Welcome to Compass Digital!

Welcome to the Data Technology team at Compass Digital! We have compiled this documentation to assist you with your Onboarding journey at CD.

Reach out to your Manager if you need any additional assistance.

􏰀 Table of Contents

Onboarding Information

Coding Best Practices Feedback Form

✍ Purpose:

The purpose of this document is to make you familiar with the Processes of the Data Engineering and Architecture Team. The document further covers the Tools/Applications that you would require while being a part of this department. The same page will also assist you in case you do not have access or need help in understanding a certain application usage or whom to contact for support.

Getting started with GitHub

Creating SSH Keys

GitHub Best Practices

19

Ideas for this page (private)

**RRSP**

The process of getting started with RRSP contribution is not very straightforward, or at least when I joined. I had to be mindful of the 3 month period before I can make contributions, find the right person and send them the filled out forms. We can do that at onboarding time.

Some people prefer to have some or all of their annual bonus go to their RRSP – it gets taxed less that way. The procedure for that is not very straightforward. At my previous job we had an online portal (much like BambooHR) where you could change the percentage of your salary and bonus that gets deposited into your RRSP (you could do that once a month I believe, very flexible.)

**Slack:**

Since Slack offers messaging people in other organizations, proposing a Slack channel with clients (Nutrislice, Zendesk, etc.) and partners (Snowflake, Monte-Carlo, etc.) would be nice to ask quick questions and keep up if it doesn’t have to be an email – such as formal requests.

**Udemy Business**

20

Onboarding Information

Last modified: Apr 5, 2023

Team Structure  
Following is the current team structure for the Data Technology Team:

Getting Started  
Your Mac will come installed with the following programs:

Cisco AnyConnect VPN

Connect to “Compass Group, NAD w/MAF” Google Chrome  
Microsoft 365

Outlook Teams OneDrive Office

Data Technology Org Chart (as of 04/03/23)

Before downloading and installing all the tools you are going to need, it is important to set up OneDrive as your primary directory and choose your browser.

**OneDrive**

Use this to store all files. Do not save anything locally.  
Within OneDrive, make the directory /Downloads/ and set it as the default download location for your web browser and Slack.

Portals & Access Information

**HR**

1. **Clicktime**: Timesheet used for logging your hours
2. **BambooHR**: HR website and employee information
3. **Dayforce:** Payroll
4. **ClaimSecure:** Benefits

21

1. **Perkspot:** Employee Discount website
2. **Self-Service:** Website to reset passwords

For any issues related to benefits, you can send an email to benefits.services@compass-canada.com  
For any HR & Payroll-related issues you can either reach out to the HR department or your direct Manager.

**Communication**

1. Microsoft Outlook
2. Slack
3. Microsoft Teams

**Support**

Digital Support Centre  
Compass Canada Self-Service Compass Technology Digital Workplace

**Security**

1. Cisco AnyConnect (VPN): Available by default on your machine. Used for access to internal resources and controlled information assets.
2. LastPass: LastPass is a password manager that stores encrypted passwords online.

You can create a request to get access to LastPass on the sre-service-desk channel on Slack.

**Engineering**

**Developer Experience**

**Github**: A platform used for storing code, tracking changes, and collaborating on software projects.  
Create an account on GitHub using your Compass email.  
Once you’ve created your username in GitHub, Contact your Manager and ask them to add you to the project repositories. Click here to access the CDL’s common Github repositories.

You can check our GitHub Process page for more information. **Jira**: It is an agile project management and issue-tracking product.

To get access to Jira, contact your Manager or you can post your request on the Slack channel.  
**Confluence:** Confluence is a web-based wiki used in Compass Digital for internal documentation. Confluence is easily integrated with Jira allowing you to create tasks for your

internal documentation.

Contact your Manager to get access to Confluence.

**Cloud**

**AWS**

AWS is a cloud solution that provides servers, storage, networking, and compute resources. AWS is our primary cloud solution.

**Elastic Map-Reduce (EMR)**

EMR is a service offered by AWS. This service provides a cluster that can be dynamically scaled up and down. In simple words, EMR provides a Spark engine (and other tools that are used with Spark) to transform the data that is ingested by dividing and distributing the tasks to multiple worker nodes on the cluster and hence optimizing performance while minimizing costs.

EMR is a cloud big data platform for running large-scale distributed data processing jobs, interactive SQL queries, and machine learning (ML) applications using open-source analytics frameworks such as Apache Spark, Apache Hive, and Presto.

This is exclusively used during our Extract-Transform-Load process. **Elastic Kubernetes Service (EKS)**

To get access to AWS, create a request in the sre-service-desk channel on Slack. You will receive credentials through LastPass and you will have to generate the password through an internal library called Talos. Talos is an MFA-enforced credential generator.

22

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed container service to run and scale Kubernetes applications in the cloud or on-premises. This service provides a Kubernetes cluster that dynamically scales based on its load.

**What is a Kubernetes cluster?**

Kubernetes is an orchestrator for Docker containers. A docker container can be run as a standalone piece, similar to a simple computer. However, as the system grows in complexity the different pieces of the system — containers and their resources — need to be configured to continue to work with each other.

This is exclusively used during our Extract-Transform-Load process. **DynamoDB**

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

You can find more information about DynamoDB here.

**Technologies**

**Airflow**

A Python-native orchestration tool for programmatically authoring, scheduling, and monitoring workflows. Head over to our Apache Airflow section to know more. **Redshift**

Data warehousing tool provided by AWS. Use the database clients (like Dbeaver) to connect to RedShift.

Contact Data Engineering channel to submit your request for access. **PowerBI**

It creates and shares interactive data visualizations across global data centres, including national clouds to meet your compliance and regulation needs. Sign-up with your Compass email to start accessing PowerBI.

To get Pro account access, submit a request here. **Looker**

Looker is another visualization platform (similar to PowerBI) with an added self-serve layer. This platform is used internally (CD employees) only to optimize the facility of creating flexible reports. A license is required to access this platform. For more details about the setup for Looker, click here. Click here to access the Looker Development Workflow.

**DBT**

dbt helps data teams work like software engineers—to ship trusted data, faster. For more details about its setup, Head over to our Using DBT section. **Snowflake**

Snowflake is a powerful data warehousing tool that has recently become popular in the industry. It enables you to build data-intensive applications without an operational burden. Snowflake integrates seamlessly with cloud services and offers more features than its competitors.

**Docker**

Docker offers the capability to deploy a lightweight OS with only the necessary software and packages to run code. This environment is called a container and is isolated from the rest of the system. Using a docker image – a set of instructions for building the container – identical containers can be run on different systems regardless of what OS they run, as long as they support Docker. This feature allows users to recreate the environment to prevent errors due to incompatible software and packages. Further, developers can run their code in identical environments and reproduce results regardless of the platform that is hosting the container.

Got an error? Contact the IT Support Team.

23

Getting started with GitHub

GitHub - Account Setup and Configuration

GitHub is the single largest host for Git repositories and is the central point of collaboration for millions of developers and projects. A large percentage of all Git repositories are hosted on GitHub, and many open-source projects use it for Git hosting, issue tracking, code review, and other things. So while it’s not a direct part of the Git open-source project, there’s a good chance you’ll want or need to interact with GitHub while using Git professionally. Click here to get further instructions on GitHub Account Setup and Configuration.

Must-Know concepts

Branching

Git Branching is the most important feature in Git. It enables developers to collaborate in an agile manner. Instead of everyone committing to the same branch and possibly introducing bugs/conflicts, each developer creates a copy of the main branch and works on it. After their work is done and tested, the pull request is created and the new branch is merged with the main one. We have mentioned below two popular ways of utilizing this feature:

**Git Flow**

Following is a typical workflow. Each team in CDL has their own ‘flavour’ to their workflow. In this workflow, the development style is characterized by having 3 main branches:

1. Develop
2. Release
3. Main/Master

You are branching off the develop branch and when the changes are approved get moved to the release branch.  
When the release branch becomes mature enough it is merged into the main branch. This branch is called Long Term Support releases. The advantage of this flow is, that you have control over what gets released. The disadvantage, however, is the amount of time required in controlling.  
It is beneficial for open-source projects where there are several contributors of varying skills. To avoid bad code getting through to your releases, there are 3 walls of Pull Request to guard it. (Even though this hinders the development speed).

**Trunk Based Development (TBD)**

In Trunk Based Development you just have only one branch. Every committer makes a branch when they implement a feature or fix a bug.

Submit a Pull Request after making the changes. Get the code peer-reviewed, and merge it into the main branch. This development cycle allows for true Continuous Integration and Deployment making it incredibly agile.

If you need access to Github refer to our Onboarding Information section for more information.

The level of control over code quality drops significantly, but in a team where all developers know what they are doing it is highly beneficial.

24

How do you do it?  
Following are some commands that will help you to start your Git branching journey:

**Git Checkout**

Concept of a staging area: A staging area needs to be populated via git add and then committed via git commit -m 'helpful message.

The following command lets you move to another branch (or commit). For example: If you're in the master branch and want to create a new branch and change You can use the following command:

git checkout -b new\_branch

Your terminal tracks what branch you are currently on. You can then continue working on your feature, and then when you are ready to push some changes to the repository use the following command:

git push origin new\_branch You can also do a git push .

**Git Pull**

To get your local source up to speed, use git checkout master and then git pull .  
-b option is used when you are creating a new branch if the branch already exists you will get an error.

Now that you have caught up, you can git checkout -b new\_feature and start contributing again! **Git Rebase**

You must include new changes in your current feature branch when you are working on a large/difficult feature and the master branch has new changes since its creation.

The first thing would be to git checkout into master and follow the git pull part of the tutorial to catch up on master (Don't forget to either git stash or git commit your changes since when you git checkout it might drop all uncommitted changes from your current branch).

Now that you have done that, you can git checkout my\_old\_branch and do git rebase master .

References

Learn Git Branching Git - Book

This command tells Git to pretend to have branched off from the top of master instead of the middle. It keeps your current changes and sorta soft merges the source with a master in this case.

25

Creating SSH Keys

Windows OS Installation Instructions  
1. If you are using Windows OS, you can download and install Git bash here. 2. Once installed, Open Git Bash

3. Type the following command ssh-keygen in the *.ssh* folder.  
4. You will be prompted with the following message `*Enter file in which to save the key:`* Over here, you need to enter the github\_rsa . 5. You can leave the passphrase empty or for added security enter a passphrase.

Do not forget the passphrase if you have entered one as you will need it every time you commit.

6. You should now see two files that have been created in that folder as github\_rsa and github\_rsa.pub . 7. Type cat github\_rsa.pub and copy the displayed on the screen.  
8. Now go to your GitHub account and click on Settings and then on SSH GPG keys or click here  
9. Click on `New SSH Key`.

10. Paste the github\_rsa.pub in the key section and give a title.  
11. Now open the *config* file from the .your\_home\_directory/.ssh/ folder in any editor. 12. If you do not have a config file you can create one in that directory.

13. Enter the following information in your config file:

Host github.com  
IdentityFile ~/.ssh/github\_rsa

14. You should now be able to make commits to your repositories.  
15. If you want to clone a new repo please make sure you clone using the SSH URL

**Linux & Mac Users, follow these steps:**

1. Open the terminal on your Linux/Mac machine. 2. Go into your *your\_home\_directory/.ssh/*

a. If you do not have .*ssh/* folder you can create one.

Config file does not have file extensions like: .txt or .py

Check this link for more information: Adding a new SSH key to your GitHub account - GitHub Help

26

27

GitHub Best Practices Git Best Practices

**Activate main branch protection through your repository settings**

It is always a good idea to turn on git branch protection to prevent accidental direct changes to the main branch and ensure that your main branch code is deployable at all times. All commits should be pushed to main through pull requests.

**Don’t commit code as an unrecognized author**

Ensure your Git client is configured with the correct email address and linked to your GitHub user. Check your pull requests during code reviews for unrecognized commits. Sometimes you commit code using the wrong email address, and as a result, GitHub shows that your commit has an unrecognized author.

**Don't write meaningless commit messages**

they should be short and self-explanatory, to say what changes you made in the code and where they take effect, for example: For Example:

bad: "fix on index.html"  
good: "Improved responsiveness on navbar"

**Define code owners**

Use Code Owners feature to define which teams and people are automatically selected as reviewers for the repository. **Archive dead repositories**

Sometimes developers create repos for an ad hoc use case, a POC, or some other reason. Sometimes they inherit repos with old and irrelevant code. These reports were left intact. There is no development work in those reports and you want to avoid the risk of other people using them. The best practice is to archive them, or make them “read-only” to everyone.

28

**Move or mark outdated codes and directories**

Sometimes there are code snippets or directories in project repositories that are outdated and not being used due to a methodology change or other reasons. It’s a good practice to either move those directories to a separate repo and archive it or mark those directories as inactive in the readme description to make the code base easier to follow.

**Clean your commit history**

Commit frequently for code changes and make use of the “git rebase -i Head~{n}” command to clean up your commit history with more meaningful and atomic commit messages [3].

References

1. What is version control | Atlassian Git Tutorial  
2. https://www.datree.io/resources/github-best-practices  
3. https://hackernoon.com/how-to-clean-your-git-history-ryzb3ydv

29

Code Review

Code Review or peer code review is an important aspect which improves code quality, reduces overall mistakes and saves time and money for an organization. It is also highly encouraged that code review is conducted every week and any ***code should be reviewed before it is merged to any feature branch.*** Ultimately, consistent code review in an organization will help reduce mistakes and save time and money in the future.

**Philosophy**

It’s important for an organization that the code review process be a positive experience for both the reviewer and reviewee. The reviewer should focus on helping the reviewee write cleaner code and the reviewee can use this opportunity to explain the codebase or libraries they may have used.

**Guidelines for Review**

**Short**: Less than 500 lines of code at a time. No more than 1 hour. **Frequent:** We encourage reviews **once/multiple times per week.**

**Peer-to-Peer mentality**: Even the most senior engineers benefit from outside perspectives. Reviews should have an atmosphere that is hierarchy-agnostic.

**Don’t make fun of bad implementations**: Learning requires vulnerability. Getting a laugh about bad code is not worth sacrificing the positive learning environment.

**Clarify; Don’t Argue:** Your reviewer is brand new to your code and the whole point is they are a fresh set of eyes. If they are raising something as a concern it will be difficult for them to back up their point when in your domain. **As a reviewer,** it’s your job to listen actively and **ask three** follow-up questions before defending any implementation.  
**Cross Pods if possible:** In an ideal situation, the reviewer and reviewee would be working on different projects which allows the code review to be an unbiased process that allows a positive atmosphere where there is knowledge transfer.

To review code efficiently, it is important that the reviewer and reviewee follow guidelines including but not limited to the list below:

**Tasks to Identify**

Variable Names Function Names Function Length

Function Parameters

Documentation Data Structures Optimization Redundancy Modularization

**Descriptions**

Are the variables' names descriptive? (variable named as employee\_names\_df rather than df). Are the Function Names Descriptive ( get\_date\_from\_db )? Use verbs for functions names.

Is the Function length longer than ~25 lines of code? Or does one function do multiple tasks? In most cases, function length should not be more than 25 lines unless justified. This improves the overall readability of code and helps identify potential mistakes.

Are there more than 3 function parameters? Having more than 3 function parameters indicates that the function is doing more than one task and can be split into multiple functions.

Ensure there are sufficient comments for the function or class. This will help reduce the time to understand the code when read at a future date. Is the Data Structure engineered (Dictionary in a dictionary)?  
Are there unnecessary nested loops? This causes the program to run slowly and should be avoided.  
Is there redundancy in the code? If code is being duplicated it indicates that duplicated lines of code need to be their function.

Are there functions or classes that can be used in other Projects? As an organization, we want to avoid re-inventing the wheel on every project and having code that can be used in multiple projects helps the organization save time and resources for future projects.

For more information you can read the following the blogs:

1. https://medium.com/palantir/code-review-best-practices-19e02780015f 2. https://www.atlassian.com/agile/software-development/code-reviews

30

Coding Best Practices

**We have compiled information from various sources to give you some hints on how to improve developer workflow and productivity, code quality and security.**

Clean Code Best Practices  
These are some guidelines that would help make your code more readable and easier to follow:

**Use searchable variable names**

The code we write must be readable and searchable. Make your names searchable by defining them as variables.

1 import time 2

1. 3  # Instead of the following:
2. 4  time.sleep(86400)

5

1. 6  # Declare them in the global namespace for the module.
2. 7  SECONDS\_IN\_A\_DAY = 60 \* 60 \* 24
3. 8  time.sleep(SECONDS\_IN\_A\_DAY)

**Use meaningful variable names**

The name of the variables should be meaningful for the code to be easy to follow.

1 import datetime 2

1. 3  # Instead of the following:
2. 4  ymdstr = datetime.date.today().strftime("%y-%m-%d")

5

1. 6  # Use the following:
2. 7  current\_date = datetime.date.today().strftime("%y-%m-%d")

**Code Cohesion: Functions should do one thing**

When functions do more than one thing, they are harder to test and troubleshoot. When you can isolate a function to just one action, it can be refactored easily and your code will read much cleaner. Both dependency injection and dependency inversion can help you do this with ease.

1 from typing import List 2

1. 3  class Client:
2. 4  active: bool

5  
6 # Instead of the following:

7 def 8  
9

10  
11 def 12  
13  
14

email(client: Client) -> None:

pass

email\_clients(clients: List[Client]) -> None:

"""Filter active clients and send them an email.

"""

for client in clients:

31

**Use docstrings for functions and classes**

Make use of docstrings to briefly explain what each function and class does. Docstrings are more helpful for end-users of your functions than comments because they can be retrieved using Python’s help function.

1 2 3 4 5 6 7 8 9

10

def square(a):

'''Returned argument a is squared.'''

return a\*\*a

help(square)

# would show:

Help on function square in module \_\_main\_\_:

square(a)

Returned argument a is squared.

**Keep the same docstring format across your codebase**

There are various formats for Docstrings (e.g., Google’s docstring, PyDoc, etc.) that you can easily activate in your coding IDE and use across your project or codebase. To make the code cleaner and more readable, it’s better to use the same format across the repository.

**Use type annotation in function signatures**

Type annotation (type hinting) is useful for specifying a function's expected input and output types so that the end-users of your modules would know what range of values they can pass to the module to avoid errors and exceptions. Python’s built-in typing library is very useful for specifying the types of your argument.

1. 15  if client.active:
2. 16  email(client)

17  
18 # Use the following:

19 def 20  
21  
22

23  
24  
25 def 26  
27  
28  
29

get\_active\_clients(clients: List[Client]) -> List[Client]:

"""Filter active clients.

"""

return [client for client in clients if client.active]

email\_clients(clients: List[Client]) -> None:

"""Send an email to a given list of clients.

"""

for client in get\_active\_clients(clients):

email(client)

1 2 3 4 5 6 7 8 9

10 11

import numpy as np

from typing import List

# Instead of

def sum(arr):

return np.array(arr).sum()

# Use

def sum(arr: List[float]) -> float:

'''Returns sum of the values in the list.'''

return np.array(arr).sum()

32

**Use Python linting for your code**

Python linters help make your code format consistent by enforcing stylistic standards like the length of lines, order of imports, unused imports, etc. It’s good to activate a linter in your IDE to follow the same style across your codebase.

**Freeze library versions**

Freezing the library versions you use in your code in a requirements.txt file, makes your project easier to reproduce.

1. 1  # Instead of
2. 2  pandas
3. 3  sqlalchemy
4. 4  boto3

5

1. 6  # Use
2. 7  pandas==1.4.2
3. 8  sqlalchemy==1.4.35
4. 9  boto3==1.22.0

**Use loggers**

Using loggers to keep track of the flow of execution in your code, makes troubleshooting long and complex scripts much easier.

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

# Instead of

def my\_func(args):

<block 1: pulling the input data>

<block 2: training a model>

<block 3: running inference>

# Use

import logging

def pull\_data(args):

logging.info("pulling the data from redshift...")

<pulling the input data>

def train\_model(args):

logging.info("training the model...")

<training a model>

def run\_inference(args):

logging.info("running inference...")

<running inference>

def pipeline():

pull\_data()

train\_model()

run\_inference()

**Use inline commenting**

33

When a given function is long and has complex logic, using inline commenting to clarify what each block of code does, helps with the review and maintenance of the function.

**Use unit tests**

You can use unit tests to test out your modules programmatically. One of the advantages of writing unit tests over manual testing is the reusability of the tests and facilitating the PR review process. Common tools for writing unit tests in Python are unittest and pytest. You can learn more about unit tests here.

Design Best Practices

These are some tips and resources to understand how to structure your code so that it’s easy to extend, maintain, and test. Once your code reaches a certain size it can start to become very difficult to work with and starts to link and twist together like a big bowl of spaghetti. This is the anti-spaghetti toolkit. Many of the topics have well-made videos linked in the text that can be found at the bottom of the page in the references and resources.

**Dependency Injection: separate creation from use**

Dependency injection is a technique for decoupling pieces of code and making them less reliant/dependent on each other and increasing code cohesion. This technique can be improved with effective dependency inversion, as shown below:

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

*# The send\_message method is responsible for creating the client and for*

*# using it.*

*# This makes then method hard to test expecially if the client is a 3rd*

*# party client (e.g. boto3)*

**class** SMSClient:  
**def** send\_message(self, message: str):

**print**("Yeet sms: ", message)

**def** send\_message(message: str): client = SMSClient()  
**print**("sending message: ", message) client.send\_message(message) **print**("message sent!")

**def** main(): send\_message("hello world")

*# Instead:*

*# Now we can pass in some client created with in any manner we want*

*# which will help when testing. We can also use send message again without creating*

*# another client.*

**def** send\_message(message: str, client: SMSClient): **print**("sending message: ", message) client.send\_message(message)  
**print**("message sent!")

**def** main():  
client = SMSClient() send\_message("hello world", client) send\_message("it's me", client)

34

**Dependency Inversion: define once and use it many times**

Dependency inversion is when you define the desired behaviour or *interface* for an object you want to work with. This allows you to re-use your orchestrating code when you inject different dependencies that behave similarly and keep your code DRY (Don’t Repeat Yourself). Two key tools for dependency inversion are Abstract Classes and Protocols, see below! This video gives a good overview of dependency injection and inversion.

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

*# Here we have a lot of code duplication here! If we want to*

*# add many more clients then the code starts to explode and it becomes difficult*

*# to modify how the send\_<something>\_message methods work at the same time*

**class** SMSClient:  
**def** send\_message(self, message: str):

**print**("Yeet sms: ", message)  
**def** send\_sms\_message(message: str, client: SMSClient):

*# etc...*

**class** EmailClient:  
**def** send\_message(self, message: str):

**print**("Yeet email: ", message)  
**def** send\_email\_message(message: str, client: EmailClient):

*# etc...*

**def** main():  
sms\_client = SmsClient() send\_sms\_message("hello world", sms\_client)

email\_client = EmailClient()

send\_email\_message("hello world", email\_client)

*# Instead:*

*# Have a abstraction of a client that describes it's behaviour.  
# Otherwise known as an Interface. It is now a lot easier to change how send\_message # behaves for all clients and it's simple to add more and more clients.***from** abc **import** ABC, abstractmethod

**class** Client(ABC):  
@abstractmethod *# classes that inherit Client must overwrite this method* **def** send\_message(self, message: str):

**pass**

**class** SMSClient(Client):  
**def** send\_message(self, message: str):

**print**("Yeet sms: ", message)

**class** EmailClient(Client):  
**def** send\_message(self, message: str):

**print**("Yeet email: ", message)

**class** WhatsappClient(Client):  
**def** send\_message(self, message: str):

**print**("Yeet whatsapp message: ", message)

**def** send\_message(message: str, client: Client): **print**("sending message: ", message)

35

A useful thing to note in the above example is that each of the clients can have completely different initializations and collection of methods but can all act as a client if they inherit from the Client class and satisfy the client interface (send\_message).

**Abstract Classes and Protocols: understand when to use them**

Both tools are useful for implementing effective *dependency inversion* in your code and allow you to de-couple pieces of logic and re-use them more widely and effectively.

With Abstract Classes you define the behaviour of the interface and then explicitly implement many objects that implement that interface. These objects may be quite different from each other and have different init requirements and methods! There’s an example of using abstract classes above when we discuss dependency inversion.

Protocols are very similar to abstract classes in that you use them to abstract a behaviour into an interface. They differ from abstract classes as they are *intrinsically* satisfied by classes. This means you don’t need to inherit from them and can just pass the classes as arguments where the Protocol is required.

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

**class** SMSClient:  
**def** send\_message(self, message: str):

**print**("Yeet sms: ", message)

**class** EmailClient:  
**def** send\_message(self, message: str):

**print**("Yeet email: ", message)

**class** MessageSender(Protocol):  
**def** send\_message(self, message: str):

...  
**def** send\_message(message: str, sender: MessageSender):

*# etc...*

**def** main():  
sms\_client = SmsClient()  
email\_client = EmailClient() send\_message("hello world", sms\_client) send\_message("hello world", email\_client)

This affords the ability for your interfaces to be very specific to just the behaviour that method requires, which can help reduce the size of your interfaces which is always a good idea.

Since you don’t have to inherit from the Abstract Class you can have interfaces that are intended to be used by many 3rd party classes or by classes spread out through your codebase; this can be a powerful tool for decoupling parts of your code and separating them into different modules and packages.

A useful video comparing these two tools in more depth can be found here.

52

53

54

55

56

57

client.send\_message(message) **print**("message sent!")

**def** main():  
**for** client **in** [SmsClient(), EmailClient(), WhatsappClient()]:

send\_message("hello world", client)

36

Resources & References 1.

2. 3. 4. 5. 6.

Python Docstrings Tutorial : Examples & Format for Pydoc, Numpy, Sphinx Doc Strings

GitHub - zedr/clean-code-python: Clean Code concepts adapted for Python

Python Code Quality: Tools & Best Practices – Real Python

Getting Started With Testing in Python – Real Python

Dependency INVERSION vs dependency INJECTION in Python

Protocol Or ABC In Python - When to use which one?

37

Feedback Form  
A feedback form is a way to collect opinions about your company's service. The goal is **to gain a better understanding of the overall**

**documentation experience so that we can identify areas for improvement**.  
**Click here to get started on the Feedback Form below to let us know how we have done to help you.**

38

􏰀 Templates

39

Post-Mortem

**Description**

e.g., prod ran into issues

**Root cause**

e.g. front-end deployment passed but back-end deployment failed

**Fixes**

e.g. the offending commit was reverted

**Miscellaneous**

things to note

**Action items**

e.g. scope a proposal for canary deployments

40

The Insights Engine

The Insights Engine (IE) is a user-orientated analytics delivery platform. It is designed to enable Insight Creators to get their analytical work in front of the right stakeholders as soon as possible. It is the data backend for various projects like Distilr Supplier Dashboard, FMP, and partly for Operator Analytics.

The analytics for the Insights can be refreshed when new data is available. Insight Creators can specify which Insights go to which users of their App. On top of all this, it allows for content filtering. Meaning that Insight Creators can give their users the ability to keep their stream of information relevant to them.

The Insight Engine is exposed as a REST interface designed in a way that allows several integrations and applications to make use of the scalable and robust analytics orchestration.

Workflow  
As shown in the diagram below, data from the SQL database is transferred to the Insights Engine.

An Insight Creator is an Engineer working on the project. They use the Insights Development Kit to create a Generator. Generator is a piece of code that generates insight from the data. In the Insight engine app, there are multiple generators.

Workflow of Insights Engine

41

Based on the request made to the Engine by the Users, it runs a Generator for each user.

For example, If User A could requests information regarding Pepsi, then the Insights Engine would run a generator and display the required information. In this case, User A is the data stakeholder requesting for information. The same process is repeated if User B selects another set of information.

42

Concept & Terminologies

These are some core terminologies used throughout this documentation. The first mention of each on a page will be linked to these definitions for ease of browsing.

**Insight:** A combination of human-readable text and graphics that represent a useful and actionable piece of information. An example of Insight can be seen below in the Example Insight Layout section.

**Insight Creator:** Someone who writes insight generators and uses the Insight Engine to pass Insights to their Users.  
**Generator:** A Python class that creates an Insight Payload every time it is run. This class will create a query/set of queries to collect the required data, transform that data into a useful format for transcription, and create a visualization to describe the Insight.  
**Payload:** A specific instance of an Insight in JSON form that is passed to the front end. This includes all the information needed to display an insight on the front end including a Generator Title, an insight heading, a transcription/text body, a significance

**Significance**: A statistical significance is given to each insight payload by the Insight Creator during the insight generation process, specifically in the transform step.

**User:** The receiver of Insight, often the users of an external app that has integrated with the insight platform.  
**App:** An internal app. A collection of Insight Generators and Users that are linked together in the Insight Engine by a common

parameter, app\_name, passed with each integration request.

Example Insight Layout

Below is an example Insight payload from the FMP Disitlr App, along the side, are descriptions of the components that make up an Insight from the Insight Engine.

43

Workflow Related Links  
Jira Board: https://teamideaworks.atlassian.net/secure/RapidBoard.jspa?rapidView=255 Github Services Repos: Compass Digital

Education Related Links:  
Designing Microservices: Design Microservice Architectures the Right Way

44

45

Using the Insights Engine  
If you’re looking to use the insights engine in your project then you’re in the right place! However, if you’re looking to contribute to or learn

about the Insight Engine internals then hop over and have a look at our contributor pages. **Overview**

The Insights Engine was designed to be flexible and used in as many applications as possible. There are 3 key steps you need to take to get Insight into your users:

1. Prepare your Database for Insight Querying [instructions]  
In this step, you’ll be making sure your database has a suitable layout for creating insights. This involves making sure the tables/views have

the correct columns for the insights engine to filter against. 2. Add the Insight Engine to your backend [instructions]

In Step 2 you’ll use the Insight Engine SDKs or APIs to orchestrate how the Insights Engine integrates with your app. This integration will include adding, updating, and removing Users when they sign up to your app and passing the Insight Payloads from the Insights Engine to your front end.

3. Construct Generators and deploy them [instructions]

In Step 3 you’ll learn how to write an insight generator with help from the Insight Development Kit (IDK). The IDK will generate all the boilerplate for your Generator as well as specify method input and output types making it easy to create a Generator that seamlessly integrates with the Insight Engine. We’ll also cover deploying your insights to the Insight Engine and testing some of the results with fake users.

With each of those steps completed, you should be able to see Insight Payloads appearing in your application.

46

Getting Started: Tools and Access To create Insights you will need access to the following:

The Registration Service Github  
This is where you will write your analytical code in the form of Generators and where you store the configuration for your Insights

App.

**Insight Creator Tools**

CDL Insights Development Kit: cdl-idk  
The CLI toolkit that helps create Insight Generators

SDK  
You will need access to the following SDK to integrate the Insights Engine:

https://github.com/compassdigital/insights-proxy-service - Connect your Github account Contact Eden Trainor to get access to the repository.

Refer to our Onboarding Page to get more information regarding access to Github. Contact Eden Trainor to get access to the repository.

47

Preparing Your Database

Why prepare your database?

When registering users to the Insights App you will need to specify a set of Permissions which filters the data for that user based on their access level. The permissions are created by using a SQL view which is subsequently passed to the Insight Generators. This allows the user to have access to the Insights Engine for their permissions. The way you specify these permissions are:

The first step in preparing the data is to register a user through the Insights Engine API.  
While registering for the user, you can customize the permissions of data viewing for the users. The user is only limited to a certain set of data because of data security reasons.  
The Insights generator then passes the view to the user.  
If 2 users have the same permissions, the Insights engine runs just once for the sets.

How to Prepare the database?

The view is created on the SQL database and the permissions will be stored in the Insights Engine config file. In the permissions, you’ll pass a permission type and a permission value.  
The permission type is a type of column. Like the region, division, and sector.  
In the config file, the backend developer is in charge of integrating

App Configuration

In this section, we will set up the development location for Generators and also create configuration files that inform the Insights Engine on how to connect to your backend databases and how to implement RLS on the tables therein.

Let’s add the screenshot of the config file here.

Preparing the Database

**Step 1: Create The App Folder**From the base of the registration service repo **navigate** to the registration\_service and then the generators folder: ./insights-

registration-service/registration-service/generators . 1 cd ./registration-service/generators

In this folder, you should see several other folder names after Insight Apps. For example, /fmpdistilr . **Create** a new folder and name it after your app, this will be referred to as your App Folder.

1 mkdir MyCarefullyChosenAppName

This name is important as it is the accepted app\_name parameter that refers to all resources of your app from now on. You attach this parameter to every interaction with the insights engine.

48

**Step 2: The Configuration File**

Inside this folder, make a configuration file. The configuration file outlines what source tables are used to create insight and the permission requirements Listed below are the following details that are required for the configuration:

1. app\_name : Name of the app.

2. connection : List of connection objects required to create a connection to your appropriate database.

3. secrets : The location of the password on AWS secrets manager for the database connection. Passwords are stored in the AWS secrets manager.

4. views : It includes permissions required to build the SQL views for each user. This is a nested object which includes: a. insight\_views : The location of the schema where the SQL views will be built in.

b. database : The location of the database of the source tables from which the SQL views are built from.

c. schema : The location of the schema of the source tables from which the SQL views are built from.

d. permission : This object includes the permissions of the users that access the insights engine and is a nested object which includes:

i. value-key : A lookup value that maps to the specific row in the row level security from the user permission. For example, for app\_name fmpdistilir the value-key is team\_name , this team\_name maps specifically to DIV-CLV-38811 .

ii. type-key : A lookup value that maps to the type column and comes from the user permission. For example, for app\_name fmpdistilir the type-key is team\_type\_name , this team\_type\_name maps specifically to DIVISION which subsequently

maps to type-column.  
iii. type-column: A lookup value that maps the value from type-key to a column that contains the row-level security. For example for

app\_name fmpdistilir the type-column for DIVISION is team\_type\_divisioncode this column contains the row values for

row-level security.  
5. Tables: It is a list which contains the physical name of the source tables that the insights are built from. Tables is a nested object that

includes:  
a. table\_name : Name of the table from which the SQL views are created and insights are written.

b. view\_prefix : Name of the view for the source table excluding the table\_name .  
c. dev-alias : An alias for the source table that the developer wants to call when writing an insight.

**Listed below is an example app config file for app\_name** fmpdistilr :

1{ 2  
3  
4

5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

"app\_name": "fmpdistilr",

"connection": {

"ENGINE": "SNOWFLAKE",

"SNOWFLAKE\_USER": "INSIGHTS\_BATCH",

"SNOWFLAKE\_ACCOUNT": "spa05468.us-east-1",

"SNOWFLAKE\_WAREHOUSE": "COMPUTE\_WH",

"SNOWFLAKE\_ROLE": "SYSADMIN",

"SNOWFLAKE\_DATABASE": "INSIGHTS\_VIEWS",

"SNOWFLAKE\_SCHEMA": "MPOWER"

},

"secrets": {

"SNOWFLAKE\_PASSWORD": "arn:aws:secretsmanager:us-east-2:736888855894:secret:snowflake\_insights\_spa\_user

},

"views": {

"insight\_views": "distilr\_mpower",

"database": "SHARED\_DISTILR\_MPOWER",

"schema": "mpower",

"permission": {

"value-key": "team\_name",

"type-key": "team\_type\_name",

49

\_

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

"type-column": {

"DIVISION": "team\_type\_divisioncode",

"SECTOR": "team\_type\_sectorcode",

"REGION": "team\_type\_regioncode"

} },

"tables": [ {

"table\_name": "fmp\_fe\_parent\_fixed\_windows",

"view\_prefix": "fmp\_fe\_parent\_fixed\_windows\_table",

"dev-alias": "data\_table\_fixed\_windows"

}, {

}, {

} ]

} }

"table\_name": "fmp\_fe\_parent",

"view\_prefix": "fmp\_fe\_parent\_table",

"dev-alias": "data\_table\_parent"

"table\_name": "noma\_with\_codes",

"view\_prefix": "noma\_table",

"dev-alias": "data\_table\_noma"

50

Insight Generators

Insights are a python class when used will generate a payload with insightful info. This is the section where you start to create Generators. This means it is an analytical code that creates the core of your insights.

Creating an Insights Generator

On the Configuring Your App page, make a directory in the registration service repository that has the same name as your app. This is where we will be creating our generators. Creating Insight Generators is made considerably easier by the use of the Insights Development Kit (IDK) from the Getting Started Guide. Following are a few steps that are needed to make a basic insight:

1. The first step is creating a new folder in the configuration app. To create a new folder you’d first have to go to your app directory. For example, if you had to go to the fmpdistilr folder, you’d have to type the following command:

1 cd\_registration-service/generators/fmpdistilr

1. To create a new folder in the directory, use the following command:

1 mkdir <name of the folder>

1. You should now be able to see a folder created in the directory structure.
2. The next step would be to check the IDK version. Use the following command:

1 idk --version

1. To create the Insight Directory Structure use the following command:

1 idk --init

1. The directory includes files like:

a. transcribe.py : This file returns a transcription of the results of your Insight.  
b. requirements.txt : This file consists of your data frame requirements for Insight.  
c. idk.json : This is used for changing names insight\_name , insight\_file , or insight\_class . d. <filename.py> : This is the main Python file that consists of

1. The directory inherits all the files from the Insights Generator. These files include context , args , define\_query , transform , provide\_evidence .

Refer to our Example Layout in Concept & Terminologies for more information on how an Insight looks on a Dashboard after it is generated.

51

API Documentation

52

Insights Engine API

Proxy Service 0.0.1 OAS3  
This is the Insights Engine Proxy Service, the single touch point for accessing the insights engine services.

Terms of service Contact the developer Apache 2.0

insights Insights are user-oriented analytics delivery platform.

**Servers**

**https://bygplz0jqk.execute-api.us-east-2.amazonaws**

**Authorize**

|  |  |
| --- | --- |
| **GET** | **/get\_user\_insights** Get insight payloads for this user only |
|  |  |

53

Insights Engine SDKs  
The SDK is served from the private compass digital repository: https://github.com/compassdigital/insights-proxy-service - Connect your

Github account .  
Please Reach out to @Eden Trainor for access.

Python  
The source code for the python SDK can be found here.

**Installation**

To install the python SDK you’ll need a GitHub access token that can copy the repository linked above. The pip command below should install the correct release tag:

1 pip install git+https://${GITHUB\_TOKEN}@github.com/compassdigital/insights-proxy-service.git@{<release>}#egg=insi

**Documentation**

Documentation can be found on the following links:

Production/Integration Staging  
Development

Go (Golang)  
The source code for the go SDK can be found here.

**Installation**

To install the python SDK you’ll need a GitHub access token that can copy the repository linked above.

A guide to setting up your environment to be able to install modules from private git repositories can be found here, after which the following command will install the SDK:

1 go get github.com/compassdigital/insights-proxy-service/sdk-go

**Documentation**

Documentation can be found on the following link. Typescript (Axiom)

54

Contributing to the Insights Engine

**Title Creator Modified**

Getting Started: Work Environment Setup Eden Trainor May 16, 2022

N.B. - Please label all child pages with “iecontrib'

55

Getting Started: Work Environment Setup

Project Contributor Tools  
AWS CDK: https://aws.amazon.com/cdk/

56

Architecture

The Insight engine is a micro-service architecture

Orchestrator (Proxy) Service

Architecture Diagram of Insights Engine

Orchestrates request fulfilment by making calls to the other services. This is the center of the insights engine.

Generator Service

Runs the Generators whilst controlling their execution rate in order to avoid overrunning the databases

Group (Administration) Service

Stores collections of Users allowing for fetching insights in groups

Filtration (Personalization) Service

Filters insights from user specified filters

Subscription Service

Orchestrates subscribing Users to Generators

57

Apache Airflow

Apache Airflow is a Python open-source project for the workflow orchestration of dynamic, scalable data pipelines. Airflow can integrate with a number of sources like cloud stacks, databases, and filesystems. Airflow was originally created by Airbnb. It has a user-friendly UI layer.

Architecture  
The architecture of Airflow has the following components:

Airflow Architecture

1. Webserver (UI): This is the Flask app that displays the status of your jobs.

58

Airflow Webserver

2. Scheduler: It is a Multi-threaded Python process that uses the DAG object.  
3. Worker (only when in clustered mode)  
4. Executor: It is the mechanism by which work gets done. There are 2 types of Executors:

SequentialExecutor : This executes tasks sequentially, with no parallelism or concurrency.  
LocalExecutor - This executor supports parallelism and hyperthreading. It has a Single node deployment, Connected to an RDBMS

(PostgreSQL / MySQL) and runs all jobs on the same machine. 5. Metadata Database

Core Concepts

**Tasks**A Task is **the basic unit of execution in Airflow**. Tasks are arranged into DAGs, and then have upstream and downstream dependencies

set between them to express the order they should run in. In a Task, there are various other functions like:

Task Instance: It is an individual run of a single task.  
Task Relationship: It is an upstream/downstream (before/after) of a task. Task Lifecycle: This type of task has stages like success, failure, and queued.

59

**Operators**

Operators in Airflow define what actually needs to be done. Airflow has a very extensive set of operators available, with some built-in to the core or pre-installed providers. Some popular operators from the core include:

BashOperator - executes a bash command PythonOperator - calls an arbitrary Python function EmailOperator - sends an email

For a list of all core operators, see: Core Operators and Hooks Reference. Other than task dependencies, operators generally run independently.

Operators in Airflow

DAGs

A *DAG* (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.

Here’s a basic example of DAG:

DAG Example

It defines four Tasks - A, B, C, and D - and dictates the order in which they have to run, and which tasks depend on what others. It will also say how often to run the DAG - maybe “every 5 minutes starting tomorrow”, or “every day since January 1st, 2020”.

Inside Airflow’s code, we often mix the concepts of Tasks and Operators, and they are mostly interchangeable. However, when we talk about a *Task*, we mean the generic “unit of execution” of a DAG; when we talk about an *Operator*, we mean a reusable, pre- made Task template whose logic is all done for you and that just needs some arguments.

DAGs have explicit execution order, beginning, and end.

60

DAG

61

Airflow as an Orchestrator Problem

**Airflow is running all the heavy lifting for the tasks defined in a DAG**. Ideally, BIs, Data Scientists and Data Engineers that write their pipeline scripts should use Airflow to orchestrate submitting an image and entry point to a Kubernetes cluster wherein a container would be run within a Kubernetes pod. This means developing their automated pipeline, or manually creating and registering their docker image for use with MWAA. However, this automated process is feasible for some of the Data Engineers but is not in a Data Scientists tool kit by default, and the Data Scientists work on analytics, not implementation.

**Why is this a Problem?**

There is no automated process for developers to package their scripts into a docker image for use within a Kubernetes cluster. The only way for a developer to offload their work to a remote compute resource is for them to do it manually.

The cost of running a managed Airflow is high, and running heavy lifting within the Airflow deployment increases resource requirements thereby increasing costs.

If Airflow isn't configured in a way that can anticipate handling workloads coming in from DAGs, it could bring down the Airflow scheduler, and all DAGs to a halt.

For all Python scripts being run within tasks across any DAG in Airflow, it requires Airflow to install the Python dependencies that are required by **ALL Python tasks.**

**How does this affect a Developer?**

As the second classification of developers, they want their workflows to be seamlessly orchestrated by Airflow without worrying about Airflow being brought down and made unavailable because of heavy resource utilization.

As developers of any background (DS, BI, DE), we want to leverage Airflow as an orchestrator without having to deal much with baking our scripts into a Dockerfile, building an image out of a Dockerfile, publishing that image to a registry, and thus referencing that image in a Kubernetes pod submit operator.

Solution

Airflow should be kept lightweight by acting as an orchestrator where each task in a DAG simply offloads/orchestrates work to be done in a Kubernetes or EMR cluster.

**Solution Approach**We can solve this issue in **two** ways:

1. Automate the process of having a Dockerfile (that can be generated, or manually created) per DAG be built into an image & published to our ECR.

2. Use custom operators in DAGs that will use those published images to spin up a Kubernetes pod with a container containing a developer's workflow scripts & an ‘entry point’ function to tell the container what script to run.

A developer is classified into two, one, who would work on/upkeep the automation portion and then the second one who uses the automation. In this case, as someone who is developing the automated pipeline, we want to be able to design guard rails that help to enforce best practices, which in turn, ensures that the Airflow environment won't go down.

62

**First: The directory structure for enabling image automation**

To aid in automating the image creation process, we will make use of a standardized directory structure per pipeline. The directory structure has 4 main items.

1. An airflow subdirectory containing DAG files & airflow-related templates.
2. A contained\_scripts subdirectory that would have all of the scripts that would be sent to EKS to run.
3. A requirements.txt file, within the contained\_scripts subdirectory, containing all the python modules required for your workflow scripts.
4. A Dockerfile , within the contained\_scripts subdirectory, that defines installation requirements and entry points for building an image to be used in EKS.

The Dockerfile is fairly simple & would contain:

A base image to pull from.  
Copies the contents of the workflow subdirectory into the working directory of the (future) image. Installs all the libraries referenced in the requirements.txt file.  
An entry point script that will run.

**Second: Automating the image creation**

On every push to the airflow-cdl-data ops git repo, a GitHub action will run that would:

1. Go through each directory in the dags/directory of the repo.  
2. Find that Dockerfile.  
3. Build an image out of it & name it the same name as the directory that housed the Dockerfile. 4. Publish the Dockerfile to ECR.

To make referencing the image, in a DAG, trivial, the image name that would be passed to the Kubernetes operator will be interpolated at runtime based on a logic that would be aware of the ECR & the directory name to return a full image location to reference.

Process

There is a boilerplate generator called containerized\_pipeline , found in the cdl-dataops-airflow-2 repo, that will generate some directories and files to help get started with creating a containerized pipeline for Airflow to run.

To run the generator, execute:

1 python boilerplates/containerized\_pipeline/generator.py --name containerized\_pipeline --team data\_science

The generator requires two arguments, --name and --team .  
--name denotes the name of the project.  
--team denotes the team subdirectory to place the generated project within.

63

The output of the generator will show a view of what was created:

The contained\_scripts/ directory has:

A code/ subdirectory containing a driver script called main.py and two dummy submodules foo.py and bar.py that are referenced in the driver script.

main.py

foo.py

64

bar.py

A requirements.txt file that has some sample requirements

requirements.txt

A Dockerfile that contains instructions for: a base image to pull from  
Copying the requirements.txt file Installing the requirements

Copying the contained\_scripts/ directory Copying a common library called cdl\_library An entry point to the main.py script

The airflow/ directory has one DAG file:

Dockerfile

65

dag.py

A GitHub action is created for the containerized\_pipeline .

66

containerized\_pipeline\_image\_ci.yml

This workflow will build an image out of the Dockerfile and pushes it to the ECR registry.  
It publishes it with the same name as your pipeline.  
The next step would be to push the file to GitHub. An action is then generated in GitHub and it runs immediately:

67

GitHub

Finally, Airflow will be populated with your new DAG, and will run once it is turned on from the UI.

Containerized DAG

In the Boilerplate generator the example credentials that are passed to KubePOD Operator enable your container to have admin services. This will give you access to all resources.

68

Airflow / Orchestration

Useful Links:

**Github Repository**

https://github.com/compassdigital/cdl-dataops-airflow-2

**Managed Airflow (Dev)**

https://9631e326-d50d-41ad-a353-bfb03d6b9f25.c20.us-east-1.airflow.amazonaws.com/home

**Managed Airflow (Prod)**

https://bdc0ea40-c9db-47ac-9c57-8a617c0d456c.c68.us-east-1.airflow.amazonaws.com/home

This page contains guides to using the AWS-managed Airflow platform.

Overview of AWS Managed Airflow Environment

DataOps MWAA Environments

DataOps MWAA Configurations

Using AWS Managed Airflow Environment Adding Dags to MWAA: Dev Environment

Adding Dags to MWAA: Prod Environment

**Overview of AWS Managed Airflow Environment**

This AWS service makes managing Airflow environments very easy! You will specify what size of a cluster you need depending on your workload, and the maximum number of workers that are allowed to be deployed. In addition, you’ll need to have an S3 bucket where your dags will be hosted. Once all that is set up, you can start uploading dag files to your S3 bucket and see them on the AWS-managed Airflow UI (let’s call it MWAA from now on like AWS does.)

**DataOps MWAA Environments**

Currently, we have two managed Airflow environments:

**Development**

The development environment is available to everyone with access to testing DAGs and verifying that they work as intended. Users will start by adding their dags to the dev environment. Once it is confirmed that the dags are bug-free, users can move them to production as instructed below.

**Production**

The production environment is intended for dags that have been tested and approved by admins. There is no testing and debugging in this environment.

**DataOps MWAA Configurations**

The DataOps-managed Airflow is set up as follows:

69

The GitHub repository cdl-dataops-airflow-2 holds the dags and their corresponding scripts for MWAA. There are two main branches: dev and prod . The files on these branches are set to be synced to S3 at cdl-dataops-mwaa (through a GitHub action). There are two directories in the bucket, dev\_af2.2.2/ and prod\_af2.2.2/, each of which holds the dags for the managed airflow environments dataops-

dev-af-v2-2-2 and dataops-prod-af-v2-2-2 respectively. MWAA is set up to automatically pick up new dags/scripts from the S3 buckets and show them on the Airflow UI. This takes about 30 seconds.

**Using AWS Managed Airflow Environment**

**Adding Dags to MWAA: Dev Environment**

Now that we know how things work, let’s look at how we can add dags to MWAA:

1. Start by cloning the Airflow repository airflow-cdl-dataops on your local machine. 2. Create a new folder for your dag inside the dags folder.  
3. Place your dag file and other scripts inside that folder.  
4. Push your changes directly to the dev branch.

5. Head to MWAA UI and look for your Dag.

Check the FAQs page for ore information.

**Adding Dags to MWAA: Prod Environment**

Now that you have tested your dag on the dev environment, follow these steps to add it to the production environment:  
1. On your local machine, copy the working version of your dag files from the dev branch to another location out of the repository folder.

2. Create a branch out of the prod branch on airflow-cdl-dataops

3. Move back the files that you copied from the dev branch to the newly created branch. Make sure you use the same folder structure (create the same directory inside the dags folder if it doesn’t exist.)

4. Make a pull request for your branch to be merged into prod . Once the branch is merged, you will see your dag on MWAA UI.

Note: Creating a branch out of another simply means checking out the first branch - prod in this case - and then creating your new branch which will be a new branch out of the previous one.

70

FAQs  
We have added some Frequently Asked Questions about Airflow on this page:

If you’d like to add more questions to this page, please contact @Minnie / @Robert.Inkpen / @Vaz, Kris .

**Q. What is the git flow to create or update an airflow pipeline?**

A. Airflow is set up to refer to files found in our cdl-dataops-airflow-2 GitHub repo. There are two main Airflow environments:

1. Airflow dev (retrieves its files from the dev branch in the cdl-dataops-airflow-2 repo) 2. Airflow prod (retrieves its files from the prod branch in the cdl-dataops-airflow-2 repo)

When you want to create or update an Airflow pipeline, you will have to write/update code inside the cdl-dataops-airflow-2 repo following these steps:

1. Make a feature\_branch out of staging .

1. Make changes and commits in your feature\_branch
2. When ready to test in dev , execute the following commands in sequence:  
   (Note: These steps will copy files from your feature\_branch into dev without making a merge; once they are copied, it commits and pushes dev to remote)

1 git checkout dev 2

1. 3  # Note: You will need to repeat the below line (7) in case
2. 4  # you have more than one directory that you want to recursively
3. 5  # copy from your feature\_branch into dev.

6  
7 git checkout feature\_branch -- path/to/your/updated/files 8  
9 git commit -am "did stuff"

10  
11 git push

c. Rinse and repeat the above (1.a -> 1.b) for as many times you make ready-to-test updates in your feature\_branch

2. Once ready for prod :

1. (Best practice/optional/sanity step): Pull the latest staging and merge staging into your feature\_branch , so that your

feature\_branch is up to date with any changes that might have happened in staging .

1. PR for feature\_branch to staging (set DE as reviewer & relevant team member as an additional reviewer).
2. DE + team member approves PR → DE merges feature\_branch to staging .
3. PR for staging to prod is made & managed and merged by DE (perhaps this will happen on a Sunday/weekly cadence in an automated fashion; and for immediate updates, manually managed/merged by a DE).

**Q. Why use Secrets Manager?**

A. Variables in Airflow used for Python workloads need to be ported over to Secrets Manager so that a container that is running in EKS can contact Secrets Manager (service by AWS) to get their secret.

71

**Q. How do I add my DAG to the Production Server repo? (bring up in touchpoint meeting)**

A. Please create a branch in the repo, and add your DAG file and supporting files. Commit them and push them to GitHub. Once you submit a Pull Request and after review, they will be merged into the codebase. The following snippet shows how to create a branch and push it to GitHub:

1 git checkout -b { your branch / dag name }  
After this command, add your code and commit your changes

To create a pull request, go to the GitHub repo and switch to your branch and click on Pull Request .

Next, submit the pull request and tag people to review the PR.

1 git add {path to file you added}  
2 ... {repeat if you have multiple files}  
3 git commit -m "{a helpful message about your changes}" 4 git push -u origin { your branch / dag name}

72

**Q. How do I debug my DAG in production?**

A. Navigate to the Airflow Server’s URL and click on the DAG you are debugging.

73

Next, click on the task that Failed (indicated with the “Red” box),

You’ll be brought to this view, and click on View Log to be brought to the log file.

74

**Q. I need rapid iteration and need to commit directly to the production branch, can I commit directly to the master branch?**

A. Sure, but please be aware this is bad coding practice and can run into conflicts that will be your responsibility to resolve. Click here for more information.

**Q. Where can I find the custom CDL helper functions and custom Airflow operators?**

A. On the Airflow repo, there are two other directories within the dags directory that house both the operators and some helpful functions, common and plugins/operators .

Within the common directory, we see the heading for what the file may refer to but feel free to look through these files to find something more specific or add to them with many useful functions that you create. Please follow the doc string formatting found in the other files to ensure that the code is easily usable by everyone.

75

A sample from these files, specifically the db.py , can be seen below with two commonly used functions and execute\_sql\_query .

For the airflow operators, the file names are more explicit and house only one operator at a time.

check\_table\_exists

76

A commonly used function from this directory is the s3\_to\_redshift.py :

77

78

**Example:**

Within the projects directory, under mobile\_targets is the working dag, dag.py , which imports and uses a custom function and a custom operator which can be used as an example for the above concepts.

79

Additional Features  
What version of Airflow is supported for these features?

@Robert.Inkpen

Branching  
One of the simplest ways to implement branching in Airflow is to use the BranchPythonOperator. Like the PythonOperator , the

BranchPythonOperator takes a Python function as an input.

Branching

However, the BranchPythonOperator's input function must return a list of task IDs that the DAG should proceed with based on some logic.

Logic

Hooks  
A Hook is **a high-level interface to an external platform that lets you quickly and easily talk to them without having to write low-**

**level code that hits their API or uses special libraries**.

Hooks

They're also often the building blocks that Operators are built out of. They’re an interface to interact with external systems like:

HttpHook MySqlHook S3Hook

You can also enable building custom operators using internal hooks.

80

Hooks

Sensors  
Sensors are a special type of Operator that is designed to do exactly one thing - wait for something to occur. It can be time-based or waiting

for a file or an external event, but all they do is wait until something happens, and then *succeed* so their downstream tasks can run.

Sensors

XComs

XComs (short for “cross-communications”) is a mechanism that let Tasks talk to each other, as by default Tasks are entirely isolated and may be running on entirely different machines.

XComs

If two operators need to share information (filenames, small amount of data), you should consider combining them into a single operator, or write the data to an externally accessible location. If it can't be avoided, use the XCom feature for cross-communications. XComs can be ‘pushed’ or ‘pulled’ by all Task Instances.

81

More Features

SubDAGs: Dynamic nested sub dags using loops  
Backfilling: Schedule runs that are backfilled from a certain past date  
Trigger Rules: Run tasks even if previous tasks failed, at least one succeeded etc. Variables: Store variables, and configurations  
Connections: Connection credentials  
Pools & Priorities: Prioritization for tasks, and allocates tasks to pools to limit connections

XComs

References: Search — Airflow Documentation What is Apache Airflow?

82

Scalable Deployment & Services

Distributed Airflow  
Airflow is capable of running in a cluster mode using Celery and Redis.

Horizontally scalable but requires more DevOps best practices for productionizing code.  
It can run on Kubernetes (a production-grade container orchestration platform developed by Google).  
Increases scalability and performance of DAGs with overhead costs of maintaining the cluster and deployment CI/CD.

Parallelism

Airflow runs DAGs and tasks in parallel (if not specified), meaning that if you are truncating and reloading a table in a task that is running in parallel, you will likely lock the table because both tasks are trying to truncate and reload the table  
Running jobs in parallel requires more upfront time in designing your workflow to make sure that it runs correctly with no side effects.

Managed Solutions  
Airflow provides access to highly scalable environments without needing to worry about maintaining the deployment environments.

It reduces costs by freeing the time of the resource who is responsible for the deployment so that they’re doing meaningful and valuable work.

Increases upfront costs by having to pay a vendor for the managed service.

83

Using DBT

DBT (data build tool - dbt - Transform data in your warehouse ) is a command line tool that helps with building robust data transformations. Pages will be created / added to this section as we learn more about using this tool effectively.

84

DBT Setup

**Windows**

In order to run DBT and Airflow on a Windows machine you will need the following: Windows Subsystem for Linux (WSL), a Linux distro, the Docker installer and the command line interfaces. Following these instructions should get you started. It is recommended to install the WSL and Linux distro using the command line method. If you install Docker first it will direct you to the Windows store to fulfill install requirements, but we’ve found the downloads there to be unreliable.

1. Install WSL by opening a PowerShell or command prompt with admin powers, and running this command: **wsl --install**
2. Next run this command to get a list of Linux distros that can be installed from the command line window you opened: **wsl --list --online**
3. Next run this command, replacing Ubuntu-20.04 with whatever distro from the list you picked:

**wsl --install -d Ubuntu-20.04**

1. Once done downloading it will open a new command prompt that is running the Linux OS you downloaded, it'll ask you to create a username / password, and it will suggest using a sudo command to update the packages in the OS
2. Once all this is done install Docker (needed by Astro), and you can just jump right into the tutorial
3. Get the Astro CLI, a command line tool that will allow you to run Airflow locally. Instructions are here: GitHub - astronomer/astro-cli: C
4. Finally, install DBT: Make sure python 3.8 or 3.9 is installed (3.10 and 3.11 will not work, as of Nov 9th, 2022).
5. Install the redshift connector using pip: pip install dbt-redshift
6. Clone our DBT repo: https://github.com/compassdigital/cdl-dataops-airflow-2 - Connect your Github account
7. (Optional) Installing Virtual Environment. Run py -m pip install --user virtualenv

a. Create Virtual Environment directory. Run py -m venv <env name>  
b. Activate Virtual Environment. Run .\ <env name>\Scripts\activate  
c. Install packages into the environment. Run pip install -r requirements.txt d. To deactivate the virtual environment. Run deactivate

1. Create a folder named “.dbt” at the following location: C:\Users\YourWindowsUsername or into the virtual environment.
2. Copy the profiles.yml file from the project repository and copy it to this .dbt folder  
   Create Windows environment variables reflecting the variable names in profiles.yml  
   This screen can be found by opening the properties screen on “This PC”, clicking “advanced system settings” and then you can find the Environment Variables Button at the bottom of the dialogue.

Create the new variables in the top section which is just for your user. Through this process you will supply a name and a value for each new configuration, click ok then click new again. The sample values below are from our original Redshift project. These will change when we move to Snowflake.

**Variable Name Content**

LI that makes it easy to create, test and deploy Airflow DAGs to Astronomer

Installation overview | dbt Developer Hub

DBT\_DB\_NAME DBT\_ENV\_SECRET\_PASSWORD DBT\_ENVIRONMENT DBT\_HOST

DBT\_PORT DBT\_PROFILES\_DIR DBT\_TARGET DBT\_USER

cdl\_bi\_edw  
this is your database user’s password

dev\_yourfirstname\_yourlastname the address of the cloud DB

eg: cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com 5439

The full path where your profiles.yml can be found local\_dev  
Your DB username

85

Alternatively:  
Create a file named profiles.yml in this .dbt folder, and paste in the following / update the values cdl\_bi:

outputs: local\_dev:

dbname: cdl\_bi\_edw  
host: <REDSHIFT\_HOST>  
password: <YOUR\_REDSHIFT\_PASSWORD>  
port: <REDSHIFT\_PORT>  
schema: dbt\_dev\_<YOUR\_REDSHIFT\_USERNAME> threads: 1  
type: redshift  
user: <YOUR\_REDSHIFT\_USERNAME>

target: local\_dev

Note: After setup, if running a dbt command on the command line fails with an error about how no default profile was found that means there was an issue with dbt parsing the DBT\_TARGET variable. You can work around this by copying the entire local\_dev profile, pasting it in as a new profile block and changing the name to default. This will allow you to run dbt commands locally without having to explicitly target your local dev environment every time.

1. Start Airflow: Make sure the directory is on <YOUR\_LOCAL\_PATH\_TO\_REPO>/cdl-dataops-airflow-2/dags/ and run the command “astro dev start”
2. Run python make\_local\_astro\_deps.py to get local files.
3. Navigate to http://localhost:8080/home
4. Enter the menus shown here and create a new record in each (explained in the following steps)
5. For the connection, create a record named “redshift” - note that this is case sensitive a. Enter the database host address and port

b. Under schema, enter the name of the database to be used eg: cdl\_bi\_edw

c. Add your user credentials

1. For the variable, name it “dbt\_environment” and enter your dbt username following the pattern shown below

(dbt\_dev\_redshiftusername)

86

19. Create another variable “dbt\_target” and val “local\_dev”

20. Navigate back to the Airflow main page and select dbt\_dag. (Make sure you are in the correct branch) 21. Trigger job with config and input the following : {"DBT\_SELECTOR": "dim\_dummy"}  
22. Make sure the job ran successfully and check the log

**Mac**

The setup process on Mac is much simpler (to be completed by someone who has done a Mac setup)  
1. https://github.com/compassdigital/cdl-dataops-airflow-2/blob/dev/dags/business\_intelligence/docs/local\_development.md

**Troubleshooting**

It is possible for the default /.dbt directory that DBT looks for the profiles.yml files to be incorrect. There are a few means to address this as explained at Connection profiles | dbt Developer Hub (getdbt.com). On windows the permanent solution is by changing the environment variable:

1 SETX DBT\_PROFILES\_DIR C:\Users\<your username>\.dbt

87

Create DBT Models  
**Description**: Use these steps to create a dbt model and their best practices. Make sure DBT is installed and have Command Line and

Visual Studio Code opened. For full documentation around DBT : What is dbt? | dbt Developer Hub

1. Create a Jira Ticket for the project
2. Create a Github branch base off from dev in cdl-dataops-airlow-2 repo with the Jira Ticket number and project name. Example:

BIA-74-lms-certification-program

1. Navigate to the new branch and in Visual Studio Code, locate the folder: cdl-dataops-airflow- 2>dags>business\_intelligence>dbt>model

a. This will be the location to create the dbt model

b. When creating a model, there are two files that needs to be created: sql and yml files.

1. Add source data or the table that is used for transformations (if it is not added yet) to the schema yml files in cdl-dataops-aireflow-

2>dags>business\_intelligence>dbt>models>source

5.

6. Create a sql file and yml file. The naming for these files should be the same.  
a. SQL file should contain the SQL script and config. Config should specify the type of materialization: table , view , incremental

and ephemeral . For more detail around the four types of materialization:

Materializations | dbt Developer Hub

i. Best practice:  
1. Create a model for each table source which will modularize our transformations so that you can re-use models, and reduce

repeated code.  
2. Add all the table references at the top in the WITH statement to see what tables are used.

88

3. By using ref in the sql file, it will build dependencies between models.

b. YML file should contain dictionary for columns and testing when necessary. There are four generic tests : unique , not\_null , accepted\_values and The testing will return the rows where your

assertion is not true otherwise, if the test returns zero rows, your assertion passes. c.

1. Save all changes
2. In Command Line (make sure it is on the right directory)  
   a. dbt run --select <model name> or dbt run --select +<model name> . Having the + will run the sequence of models that the

selected model is depended on. b. Wait until the dbt ran successfully

1. Test for all models created before committing to Github and creating a pull request. Add tests to your DAG | dbt Developer Hub You can also view the tables in your personal dbt schema by running it in Redshift.

a. Note that manifest.json file will also be created. This single file contains a full representation of your dbt project's resources (models, tests, macros, etc), including all node configurations and resource properties. Manifest JSON file | dbt Developer Hub

1. Once the models are pushed, it will be pushed to a dev instance since the prod instance is currently not ready. The table will be created under dbt\_dev schema after the airflow job finishes (triggered or scheduled).

|  |  |
| --- | --- |
| relationships | Add tests to your DAG | dbt Developer Hub |

89

Running DBT Models

**Running a model locally from the command line (Windows)**

1. Open your command prompt  
2. Make sure that you have the branch you want to test on loaded in git  
3. In the command prompt, navigate to the dbt folder of your project eg: \cdl-dataops-airflow-2\dags\business\_intelligence\dbt 4. Use the command: dbt run --select +model\_name

**Running a subset of models from Airflow with DBT\_SELECTOR**

Note: This set of instructions is soon to be made obsolete. If you see dags named “optimus\_manual\_build”, “optimus\_manual\_run” etc then follow the directions listed here: BI Manually Triggering Optimus Commands

1. Navigate to the Airflow interface on US East (N. Virginia)
2. Open the dataops-bi-dev-af-v2-2-2 instance
3. Click the run button for **dbt-single-selector-dag** and select run with config
4. Enter the config as shown below, altering the path as necessary to get the set of models required. In this example, the + sign tells it to run all of the predecessor models that feed into the selected path. The \* tells it to run all models found in that folder. To understand what paths you would need to enter for various models, you can just examine the url hierarchy in git

)

The run is based out of the /dags/business\_intelligence/dbt folder.

1. If you want to run multiple subsets, you can enter a space after the first path and enter a second one. The space creates a union between the two model sets, so if both model paths have common predecessor tables those common models will only be run once.
2. You can also exclude models using the DBT\_EXCLUDE variable. In this example we trigger a single model rather than all models in a folder, and exclude dim\_dummy.

( https://github.com/compassdigital/cdl-

dataops-airflow-2/tree/dev/dags/business\_intelligence/dbt/models/dw/staging/p2 - Connect your Github account

90

7. Click trigger. Once this is done you can view the airflow log to see what tables have been finished, any errors etc.

**Running a subset of models from Airflow with DBT\_VARIABLES**

1. Navigate to the Airflow interface on US East (N. Virginia)
2. Open the dataops-bi-dev-af-v2-2-2 instance
3. Click the run button for **dbt-single-selector-dag** and select run with config
4. Enter the config as shown below, entering star\_schema\_source\_to\_run as the variable name and then a list of the source names we want to refresh.
5. Click trigger. Once this is done you can view the airflow log to see what tables have been finished, any errors etc.

91

DBT Exposure Process

Link to dbt exposure docs

Steps to Add New Exposure

1. Visit the exposures directory of the cdl-dataops-airlfow-2 repository on the dev branch (link: https://github.com/compassdigital/cdl- dataops-airflow-2/tree/dev/dags/business\_intelligence/dbt/exposures )

2. Select “Add a File” > “Create New File” and title the new file “dashboard\_name.yml”

3. Add the relevant metadata for the dashboard, including the dashboard’s title, link, dependent tables and owner

4. Scroll to the bottom and choose “Create new branch for this commit” and give it a relevant branch name and select “Propose New File”

92

5. This will open a window to submit a pull request, add the resource who helped figure out the table dependencies as a reviewer

6. Select “Create Pull Request” and the reviewer will review and merge accordingly

93

Upgrade DBT Version

Occasionally we will upgrade the version of DBT that we’re using to gain access to new features and fixes. This guide aims to assist with performing this upgrade in your development environment.

Index

1. First Step - Making sure you have the right version of Python and that pip is working 2. Second Step - Update or pull the current version of local\_requirements.txt  
3. Windows Users  
4. Mac Users

First Step - Python

When you’re going to upgrade to a new version of DBT, the first thing you should do is ensure that the version of Python running in your environment is compatible with the version you are upgrading to. You can find that by visiting the DBT Python Compatibility page and selecting the appropriate version from the dropdown selector: What version of Python can I use? | dbt Developer Hub

Make sure that your pip is functional. We’ve seen it happen that pip becomes broken somehow. If this happens you can fix it by running this command:

1 python -m ensurepip

Second Step - Local Requirements

Update the local\_requirements.txt file in your project repository to target the dbt\_core version we’re upgrading to, as well as any adapters or other packages that need to be upgraded at the same time. This file is kept in git so if you’re not the first person to perform this upgrade, you can skip this step and just pull the repo instead.

Windows

1. Open a PowerShell window
2. Run the following command, replacing <path> with whatever folder structure sits above the location your cloned the repository to.

1 pip install -r <path>\cdl-dataops-airflow-2\dags\business\_intelligence\local\_requirements.txt

1. If you get an error that package dependencies could not be built, you may need to manually uninstall those packages from your environment first. Run the following command on any packages that failed to install to make sure there are no version conflicts when the install runs. Once done, retry step 2.

1 pip uninstall packagename

1. Test the install by opening a command prompt and navigating to the dbt folder in your project: eg: \cdl-dataops-airflow- 2\dags\business\_intelligence\dbt  
   Run a DBT command to make sure everything is working. I like to use parse because it finishes quickly.

1 dbt parse

94

If you get an error like “Could not run dbt” or something about missing a default profile, refer to the steps regarding environment variable setup here: DBT Setup  
Odds are that the problem you are having is something to do with the profiles.yml file being used by DBT

Mac  
To be written by someone who has done a Mac upgrade

1. Open a PowerShell window  
2. Run the following command, replacing <path> with whatever folder structure sits above the location your cloned the repository to.

1 pip install -r <path>\cdl-dataops-airflow-2\dags\business\_intelligence\local\_requirements.txt

3. If you get an error that package dependencies could not be built, you may need to manually upgrade those packages first. For example, in my case, I get error about the wrong versions of botocore and boto3 package so I upgraded them by running the following commands. Once done, retry step 2.

4. Test the install by opening a command prompt and navigating to the dbt folder in your project: eg: \cdl-dataops-airflow- 2\dags\business\_intelligence\dbt  
Run a DBT command to make sure everything is working. I like to use parse because it finishes quickly.

1 dbt parse

If you get an error about env\_var not found, you reset the env\_var DBT\_PROFILES\_DIR to the path where you stored your local profile.yml.

1 export DBT\_PROFILES\_DIR=~/.dbt

1 # upgrade  
2 pip install botocore --upgrade 3 pip install boto3 --upgrade  
4 OR  
5 # Specify version  
6 pip install botocore== xxx

95

Power BI

Power BI is an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence. It is part of the Microsoft Power Platform.



96

Creating date range filtering in Power BI Report Builder

When creating a report in Power BI off of a model it isn’t really obvious how to configure date parameters in a way to allow a user to select a start and end date, rather than having to select all values in a range. In this guide we’ll set up a date range filter for the Sales Detail Report in the Mobile Reporting workspace.

1. Right click on the dataset that we want to filter and open the query designer.
2. Set up a parameter for the date field you want to filter on as shown below. This looks like you’re leaving out required fields as the filter expression is empty. What happens when you click OK is Power BI generates report parameters and hidden datasets to power them, and then updates the dataset query to use these new parameters.
3. If you right click on the Datasets folder and select show hidden, you can see the new items that Power BI created for this filter.

97

98

Power BI Import vs Snowflake Direct Query: Performance Comparison

The purpose of this page is to document the general PowerBI report performance results comparing the import of data versus direct querying snowflake.

A year’s worth of mobile\_orders data was used:

The test was performed at both small and large snowflake instance sizes. The performance was calculated as the milliseconds (ms) for each object on the report page to load. This was done with the built in Performance Analyzer in PowerBI desktop.

**Important note: many objects load in parallel. Meaning the total ms is not the total time the page took to load, but rather an indication of effort.**

1 select \*  
2 from "DATAOPS\_EDW"."DATASTORE"."MOBILE\_ORDERS"  
3 where orderdate between '2021-01-01' and '2021-12-31'

Small instance

Object  
count of orderid by sector  
count of customer\_ID by menuitemname Card  
Card  
Slicer  
Table  
TOTAL (ms)

TOTAL (s)

Large instance

Object  
slicer  
count of customer\_ID by menuitemname card  
card  
count of orderid by sector  
table  
TOTAL

TOTAL (s)

Direct Query (ms) 3319  
8401  
2439

2390 9767 10330 36646

36.6

Direct Query (ms) 10862  
12839  
2253

2290 2928 9276 40448

40.4

Import (ms) 361  
205  
204

203 1900 769 3642

3.6

Import (ms) 526  
1956  
326

326 325 1228 4687

4.6

As can be seen in terms of “effort” as calculated by total ms, direct query with either a small or large Snowflake instance is approximately 10 times greater.

99

Page Load Time  
A total timed page load was also recorded for the large instance comparison.

Direct Query (s) Page load 11.07

The direct query report page takes nearly 6 times longer to load.

Considerations

Import (s) 1.9

The PowerBI file was about 3.8GB which is a reasonable test of import performance, however the file was run on a local machine rather than on the PBI Service which may change import performance.

A full load of all POS and Mobile data would produce a much larger dataset which would put further strain on import.  
A select \* was done on the mobile table which is not recommended for import. A much thinner table leveraging numeric data types would be optimal for testing import.  
This analysis was only an assessment of report performance, which is different to the refresh performance of a dataset. As there is no associated in-memory dataset when leveraging direct query there is no refresh required.

There are some modelling limitations when using direct query:

no auto date/time capabilities some result set/row limitations

Power Query (modelling) limitations such as duplicating or changing columns or replacing values. This means modelling must be done on the source tables. CTEs are not supported, so you must use subqueries.

100

Compression and Integerizing Fields in Power BI

Including long hash id’s in a PBI dataset massively increases its size. However, the benefit to doing so is for determining accurate transaction/order distinct counts.

PBI’s underlying engine often known as VertiPaq primarily compresses data by two methods: value or hash encoding. Value encoding is generally applicable for integer fields (and two decimal place fields such as currency values). Value encoding compresses down much better than hash encoding. All other data types leverage hash encoding (we are ignoring the 3rd and least common compression type run- length). Under the hood, hash encoding creates a dictionary which effectively converting strings into integers for improved compression. By “integerizing” orderid we can prevent PBI from needing to use hash encoding.

OrderID Results

POS orderids (pos\_order\_hdr\_key) for January 2022 were loaded into a redshift table with an identity column to produces the ints forms. There were approximately 30M records. This data was loaded into PBI and the column sizes checked with dax studio.

**Field**

pos\_order\_hdr\_key pos\_order\_hdr\_key\_int

**Size (MB)**

1,985 243

**Encoding**

hash value

There was a cost savings of **88%** by converting pos\_order\_hdr\_key to an integer.

Considerations

Any field that is used in a relationship is always compressed using hash encoding. The technique of integerizing to allow value encoding would not apply if pos\_order\_hdr\_key were used in a relationship with another table.

There is a secondary potential benefit to integerizing. If the field is a massive string/hash as is the case for pos\_order\_hdr\_key converting to a shorter string or integer (while maintaining uniqueness) produces a shorter value which must be stored.

There is benefit to converting pos\_order\_hdr\_key, but the same technique could be tested for other string fields such as data\_source, erp\_entity\_id, item\_name and clean tiers.

Other String Fields Results  
The following results consider only the integerizing of data\_source, erp\_entity\_id, item\_name and tier\_4.

**Field**

**Size (MB)**

**Savings Encoding**

hash 81% hash hash 3% hash hash 5% hash hash

data\_source  
data\_source\_int  
erp\_system\_entity\_id 33.69 erp\_system\_entity\_id\_int 32.63 item\_name 110.20 item\_name\_int 104.64 tier\_4 29.52

1.31 0.25

101

There is large % savings for data\_source and marginal benefit for erp\_system\_entity\_id, item\_name and tier\_4. It should be added that data\_source is a very small field and thus the integer benefit compared to the total model is negligible.

Comparing the table in its original string format against an integer formatted table is shows only a 3% savings.

Conclusion

If including a long hash id such as orderid which is not involved in a relationship then it is highly advised to integerize.

There is an additional benefit to creating proper numeric id fields rather than relying on strings. Power BI and dax is **case insensitive**. This means that if item\_name is involved in a relationship and you have two values such as “Burger” and “burger” these look the same and you will be making a many to many relationship. Such a relationship is highly advised against. Some warehouse transformation like lower casing and creating integers may be beneficial.

Be areare that if you create a table visual in PBI with item name then do a distinct count it is **case insensitive**. The above example of “Burger” and “burger” will only show one value and distinct count of one. This can make troubleshooting a little challenging and your best bet is to rely on the warehouse.

102

Centralized RLS Management Pipeline

Context

**Current pipeline (implemented for OA only)**

1. DSMs/Internal CD members use smartsheet form to specify RLS permissions per user a. Owners: Luke/Sravya

2. Smartsheet table is ingested into our data warehouse (no processing done here) a. source.smartsheet\_oa\_rls\_intake\_form\_submissions

b. In this redshift table, each row contains RLS permissions for a single user (as a list)

c. Owners: Masoud/Anirudh (runs daily at 8am and 5pm)

3. Using this source table, we create and maintain a RLS table (dbt\_prod.oa\_rls)

1. We create a row for every user email + unit number they should have access to (i.e., if user A has access to 10 units, then there will

be 10 rows for this user)

1. If user is an admin user, then column admin = true and they receive access to all units in OA
2. Owners: David Beallor (runs daily at 9am and 6pm)

**Challenges**

1. OA onboarding pipelines are tied to the smartsheet file used to create dbt\_prod.oa\_rls
2. Adding users for different dashboard access through the same smartsheet pipeline will mean that:  
   a. Either the submitter needs to specify which AD groups (dashboards) the user should get access to

b. or the Submitter needs to be redirected to a separate form maintained for report-level access

1. Power Automate Flows can create some limitations (there’s a limit to number of rows you can copy from smartsheet to excel - maybe

2048 or 4096, requires testing)

Prerequisites for Centralized Solution

1. dbt\_prod.oa\_rls is a centralized permissions list for all users accessing internally developed PBI dashboards (i.e., mobile reporting, OA, executive dashboard etc.)
2. Reduce the number of steps it takes to add a user for dashboard / RLS access
3. OA onboarding pipelines still need to run as before (onboarding emails sent on time, user tracking for OA is still maintained)
4. Add an easier method to update an existing users access. For someone adding a new user, how to check if user already exists?

Potential Solutions

Timelines

103

Extract-Transform-Load (ETL) Platform Architecture

Purpose

The purpose of this document is to explain the architecture and process of ETL. The document will explain the details of each component for the members who are going to implement, maintain, configure, or make use of the components. This section includes discussions on what services, frameworks, and languages we can use and how to get started with them.

Extract-Load-Transform Architecture

104

**ETL Architecture Diagram**

The diagram shows that we receive data from multiple sources, which is collected in the Spark Engine. Once the data is Transformed, it is moved to the Data Lake and Computation Layer. From the Data Lake, it is then moved to the Data Warehouse and Application Storage Layer like Snowflake or Redshift. All this is controlled by the Airflow, which acts as an Orchestrator. Finally, the Transformed data is then moved to the Data Consumption and Application Layers like the API, PowerBI and other applications.

You can refer to the Terminologies section to understand each component of the Architecture Diagram.

105

Prerequisites

You can refer to our Onboarding page for more information on Tools/Applications/Software for onboarding in CDL. This section includes a list of software’s/tools/applications that you will need to get started with the ETL architecture:

Access to AWS (Amazon Web Services) Snowflake Access  
GitHub Access  
Redshift Access

Good to have:

A DB client like dbeaver  
Integrated Development Environment (IDE) like vscode.

Mac Users:  
iTerm2, Brew, ZSH, etc.

Windows Users: Powershell, Chocolatey

106

Terminologies  
This document consists of a summary of the terminologies used all over the ETL document:

**Spark:**

Apache Spark is a popular framework used for handling big data. It is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. It is also the most widely-used engine for scalable computing.

**Elastic Map Reduce (EMR):**

Elastic Map-Reduce (EMR) is a service offered by AWS. This service provides a cluster that can be dynamically scaled up and down. In simple words, EMR provides a Spark engine (and other tools that are used with Spark) to transform the data that is ingested by dividing and distributing the tasks to multiple worker nodes on the cluster and hence optimizing performance while minimizing costs.

EMR is a cloud big data platform for running large-scale distributed data processing jobs, interactive SQL queries, and machine learning (ML) applications using open-source analytics frameworks such as Apache Spark, Apache Hive, and Presto.

**Iceberg:**

Apache Iceberg is a **table format** that is used in conjunction with Apache Spark. It is a high-performance format for huge analytic tables. Iceberg brings the reliability and simplicity of SQL tables to big data while making it possible for engines like Spark, Trino, Flink, Presto, and Hive to safely work with the same tables, at the same time.

**What’s a table format?**A table format is a set of standards for storing data, and it enforces how data is partitioned and indexed for faster and easier retrieval. The

function of a table format is to determine how you manage, organize and track all of the files that make up a table.

**What is the benefit of table formats?**

Like any other language, Spark saves the data in different formats (.csv, .parquet, .avro, .json) but without a system in place for quick retrieval of these files, programs will have to scan the entire storage volume every time you need to query the data. Needless to say, the performance of such a system will degrade as the size of data grows.

**How does the Iceberg table format help Spark?**

Iceberg doesn’t have any dependency on Docker. Docker can be used to quickly spin up a container where Iceberg and Spark are installed from which you can run tests on. Docker, Spark, and Iceberg: The Fastest Way to Try Iceberg!

Parquet is the default format in that Iceberg stores the data files behind the scenes. The file format can be one of parquet, avro, orc. Doc ker, Spark, and Iceberg: The Fastest Way to Try Iceberg!

Iceberg helps partition and index the data that Spark saves to the data lake (hosted on S3 in parquet format in our case). Iceberg offers the following feature:

**Delta Load**: Iceberg rewrites data files for reading performance, and it can use delete deltas for faster updates.  
**Full Schema Evolution**: With Iceberg you can rename and reorder columns. Schema changes do not require rewriting your table.

**Hidden Partitioning**: Iceberg handles the tedious and error-prone task of producing partition values for rows in a table and skips unnecessary partitions and files automatically. No extra filters are needed for fast queries, and the table layout can be updated as data or queries change.

107

**Time Travel and Rollback**: Time travel enables reproducible queries that use the same table snapshot, or lets users easily examine changes. Version rollback allows users to quickly correct problems by resetting tables to a good state.

**Data Compaction**: Data compaction is supported out-of-the-box and you can choose from different rewrite strategies such as bin- packing or sorting to optimize file layout and size.

**Parquet:**

Parquet is a low-size column-based file format for storing tabular data. Parquet has helped its users reduce storage requirements by at least one-third on large datasets, in addition, it greatly improved scan and deserialization time. It is designed to be a common interchange format for both batch and interactive workloads. It is similar to other columnar-storage file formats available in Hadoop, namely RCFile and ORC.

**Docker:**

Docker offers the capability to deploy a lightweight OS with only the necessary software and packages to run code. This environment is called a container and is isolated from the rest of the system. Using a docker image – a set of instructions for building the container – identical containers can be run on different systems regardless of what OS they run, as long as they support Docker. This feature allows users to recreate the environment to prevent errors due to incompatible software and packages. Further, developers can run their code in identical environments and reproduce results regardless of the platform that is hosting the container.

**Elastic Kubernetes Service (EKS):**

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed container service to run and scale Kubernetes applications in the cloud or on-premises. This service provides a Kubernetes cluster that dynamically scales based on its load.

**What is a Kubernetes cluster?**

Kubernetes is an orchestrator for Docker containers. A docker container can be run as a standalone piece, similar to a simple computer. However, as the system grows in complexity the different pieces of the system — containers and their resources — need to be configured to continue to work with each other.

**Snowflake**

Snowflake is a powerful data warehousing tool that has recently become popular in the industry. It enables you to build data-intensive applications without an operational burden. Snowflake integrates seamlessly with cloud services and offers more features than its competitors.

**Airflow**

Airflow is an Orchestration tool that enables data movement from one source to another. It is a platform to programmatically author, schedule, and monitor workflows. It is used to author workflows as Directed Acyclic Graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies.

Airflow can help handle the following challenges:

1. Data needs to be gathered from different sources. Sources can be files, databases, streaming data, and API responses. Airflow has built-in features for connecting to these data sources with a single line of code.

2. Source data is updated on a schedule (monthly, daily, or every 5 minutes) and has multiple sources. That means we have to accommodate multiple schedules. Airflow keeps track of the schedule for you and retrieves source data accordingly.

108

1. Data needs to be transformed with custom logic – as outlined in a script, in the language of choice. Airflow can handle running scripts to restructure your data before saving it to your destination.
2. Similar to source handling, Airflow can connect you to your data destination with one line of code and you can save the (transformed) data to the destination of your choice.
3. You can create and manage scripted data pipelines as code (Python).

**RedShift**

Amazon Redshift uses SQL to analyze structured and semi-structured data across data warehouses, operational databases, and data lakes using AWS-designed hardware and machine learning.

**Looker**

Looker is a business intelligence software and big data analytics platform that helps you explore, analyze and share real-time business analytics easily.

109

Getting Started with the ETL Process  
This section includes tutorials for the common technologies and platforms that we use:

Airflow / Orchestration Apache Iceberg Docker

110

Boilerplate

*Boilerplate is any text, documentation, or procedure that can be reused more than once in a new context without any substantial changes to the original.*

To get started with writing a simple ETL pipeline for Airflow to process, there is a generator script that can be run to set up a boilerplate DAG folder within the airflow-cdl-dataops repo.

Requirements Installation  
There are two python modules that are required to run the boilerplate: pyyaml and jinja2 . They can be installed by running pip

install pyyaml jinja2.

Running Boilerplate  
The generator script here can be run with python path/to/generator.py The script uses a YAML file, that can be edited, to help create:

The directory structure for the pipeline  
DAG file with some interpolated values  
Extract, Transform and Load sub-directories each containing a python file.

Note: Each of the extract/transform/load sub-directory files would require further implementation as per your pipeline’s requirements.

The YAML file’s values include:

Running the script produces the following directory structure in the repo:

1 directory\_name: "test\_pipeline"  
2 owner: "TEST"  
3 email: "someone@compassdigital.io" 4 start\_date: "datetime.utcnow()"

1 2 3 4 5 6 7 8 9

test\_pipeline/

dags/

test\_dag.py

load/

load\_warehouse.py

extract/

api\_extract.py

transform/

template\_transform.py

In the dags/ directory, there is a DAG file that contains an implemented flow of the ETL process. Each of the extract/ , transform/ , and load/ sub-directories contain files that should be implemented appropriately as per your desired pipeline’s purpose. These ETL- related files are referenced within the test\_dag.py file.

111

Example of an Airflow Process created through Boilerplate

For more information on the process refer to our Airflow as an Orchestrator document.

112

Apache Iceberg

The parameters mentioned below are for testing purposes. We will recreate these resources and finalize the naming convention for the Data lake bucket, database, tables, catalog, cluster name, and other parameters that we would need.

Test Catalog: iceberg\_catalog  
Test Database: iceberg\_database\_test

**Getting started with Iceberg on AWS:  
1. Deploy an EMR Cluster with Iceberg installed:**

An EMR cluster by the name of CDL-DATAOPS-DATALAKE-MANAGED can be found in the us-east-1 region. This cluster has been configured to support the iceberg.

**2. Using the right catalog:**

Iceberg can use different frameworks as its catalog, such as Hive, Glue, etc. To make the configuration easier, we will use Glue as our catalog. The catalog iceberg\_catalog has been created for running tests. The syntax for referring to a table is

catalog.database.table . **4. SSH into the cluster:**

You can use Airflow to submit a spark job to the cluster – however, you can also SSH into the cluster using the CDL-DATAOPS-EMR-KEYPAIR private key. The command for SSHing into the cluster can be found when you click on the cluster name on the AWS EMR service page.

**3. Running Spark:**

You can use spark-sql , pyspark , spark-shell , or spark-submit to use Iceberg. Please refer to the iceberg documentation for more details.

**Saving files to Data lake in Spark using Iceberg:**

The general syntax for creating new tables is ***df.writeTo.create(catalog.database.table)*** Example:

df.writeTo.create('iceberg\_catalog.iceberg\_database\_test.employees')

The behavior of each operation corresponds to SQL statements

df.writeTo(t).create() is equivalent to CREATE TABLE AS SELECT df.writeTo(t).replace() is equivalent to REPLACE TABLE AS SELECT df.writeTo(t).append() is equivalent to INSERT INTO df.writeTo(t).overwritePartitions() is equivalent to dynamic INSERT OVERWRITE

Source: Apache Iceberg  
Below is a sample script that you can use to test Iceberg functionality:

1. 1  *# Imports*
2. 2  **from** pyspark **import** SparkContext
3. 3  **from** pyspark.sql **import** SparkSession
4. 4  **from** pyspark.sql.types **import** StructType,StructField, StringType, IntegerType

5  
6 *# Get SparkSession*

113

1. 7  sc = SparkContext(appName="Iceberg\_Test")
2. 8  spark = SparkSession(sc)

9

1. 10  *# Define table schema*
2. 11  schema = StructType([StructField("id",IntegerType(),True), StructField("name",StringType(),True)])

12

1. 13  *# Create dataframe corresponding to table*
2. 14  df = spark.createDataFrame([(1,"emp1"), (2,"emp2"), (3,"emp3")],schema=schema)

15

1. 16  *# Create and write to table*
2. 17  df.writeTo('iceberg\_catalog.iceberg\_database\_test.employees').create()

114

Docker

Docker is essentially a way to run very light VMs (Containers) on your computer. They are completely isolated from one another and build their binaries and packages. It is very helpful during development since you are not required to configure the environment where you are going to be running the code that you wrote.

**Getting started**

Docker is used in the following ways:

1. Write Code  
2. Write a Dockerfile 3. Build the Image 4. Run the Image

**Terminologies**

**Dockerfile**

A file that Docker uses to build an Image. Simply lists all the steps that Docker will take when it builds an image. The last instruction is typically a console command that will be run inside the Container when it is run.  
**Image**A built template for that is used to create a Container.

Hub for Docker images can be found here.

**Container**

A Container is a sandboxed process on your machine that is isolated from all other processes on the host machine. An instance of an Image. A small isolated VM that runs some code.

**Simple example**

First, let's start by writing a simple python script:

Next, let's work on that Dockerfile:

1 2 3 4 5

#main.py

if \_\_name\_\_ == "\_\_main\_\_":

print("Hello from Docker!")

1 #Dockerfile (yes, it's the name of the file) 2  
3 FROM python:3.8-slim  
4

5 COPY . .  
6  
7 CMD ["python3", "main.py"] 8

115

Let's break it down:

FROM python:3.8-slim

Here we are setting a base image for our image. What it means is that we want the slim version of a Ubuntu that already has Python 3.8 installed. This bare image is what we are going to be working with.

COPY . .

This instruction simply copies everything from the directory on our local machine (relative to the Dockerfile) and puts it into the base directory of the container (Usually ~/ )

CMD ["python3", "main.py"]  
Finally, this instruction tells the container to run python3 main.py in the command line.

**Best Practices**

Docker builds images automatically by reading the instructions from a Dockerfile -- a text file that contains all commands, in order, needed to build a given image. A Dockerfile adheres to a specific format and set of instructions which you can find in the Dockerfile reference.

General Guidelines and Recommendations for Docker can be found here.

**Good to know links**

Don't Panic: Kubernetes and Docker  
Learn Docker in 7 Easy Steps - Full Beginner's Tutorial https://github.com/compassdigital/cdl-di-airflow-containers - Connect your Github account

116

Extract

Methods of Extract

In the ETL process, data is extracted from a variety of sources. The source data owners (in this case, our customers) can/may store data in various ways based on their requirements and can share data in any format with us. In some cases, customers give direct access to their source data (not common for security reasons) or provide the data through a staging platform. Following are some examples:

1. Customers can share their data by saving them as files and dropping them in shared storage space to which, you and the client will both have access.

2. They can also share data by creating an HTTP endpoint – a link to a website – which you can query and get the data, often in JSON format and paginated – meaning a subset of all records is returned at a time. For example: Data for NYC Taxi Zones, which correspond to the pickup and drop-off zones, or locations, are included in the Yellow, Green, and FHV Trip Records published to Open Data. You can see which neighborhood a passenger was picked up in, and which neighborhood they were dropped off in.

**Example of a Code Snippet - Getting from API and storing as JSON file to S3:**

1.

For more information refer to ETL Jobs | SFTP and S3 Bucket Sync Process

1. 1  **import** boto3
2. 2  **import** json
3. 3  **import** requests

4 5

1. 6  **def** get\_api\_response(url):
2. 7  response = requests.get(url)
3. 8  **return** response.json()

9

10 **if** 11  
12  
13

14

15

16

17

18

19

\_\_name\_\_ == "\_\_main\_\_":

bucket = "some-bucket"

s3 = boto3.resource('s3')

bucket = s3.Bucket(bucket)

s3\_target = "path/to/target/file.json"

url = "https://api.compassdigital.com/path/to/resource"

json\_stuff = get\_api\_response(url)

bucket.put\_object(Key=s3\_target, Body=json.dumps(json\_stuff))

3. They can share data with CDL by giving (read-only) access to their database.  
Based on any of the above formats, developers can write a code to connect to the source and download the files once the data is obtained.

Mostly, the data needs to be formatted to match our applications, which is when the **Transformation** process starts.

Details of the Extract Process  
The raw data that we receive can be categorized as tabular such as:

Relational Database Tables CSV files  
xlsx files

117

And non-tabular formats such as:

JSON format files APIs

Considering that all of our data ends up in relational databases (Iceberg, Redshift, Snowflake) we convert them to tabular format regardless of their original format which makes it easy to handle tabular format.

Tabular data can be read like a list of rows in Python, or more commonly loaded into Pandas/Spark data frame. Transformation is then applied to data in the data frame, and from there exported as a file (also in tabular format) and/or loaded into tables on Iceberg, Redshift, and Snowflake.

Non-tabular data is commonly received in JSON format. Due to the nature of JSON files, we can have many nested levels of data embedded within, which is why the data needs to be flattened/normalized. This can be done using a code piece in your language of choice or by reading the JSON file to a Pandas data frame which normalizes it on the fly. Once the data is in tabular format, we can proceed with the Transform or the Load process.

118

Load

The Load phase in the ETL pipeline refers to taking some transformed data and persisting it to one or more target destinations. For example, Data warehouses like Redshift or Snowflake.

Loading from Iceberg

The most relevant source that our data warehouse (whether Redshift or Snowflake) is our tables in Apache Iceberg. Of course, it may not always be the case that the data you may want to load will exist in Iceberg, but for this document, we detail some options that are available for loading data from Iceberg to Snowflake and Redshift.

Loading Iceberg data to Snowflake  
To load tables from Iceberg to Snowflake, the current Iceberg data files need to be referenced and copied for Snowflake.

**Create a Manifest File**

A manifest file containing all the path references leading to the relevant Iceberg data files can be created so that we can use it later when loading to Snowflake.  
To get the data files for a table in Iceberg, they can be accessed via a Spark SQL by querying the {catalog}.{db}.{table}.files table Queries :

1 data\_files\_df = spark.sql(f"SELECT file\_path from {catalog}.{db}.{table}.files")  
If the table is partitioned and you only want to select data files for some subset of partitions, you can filter the files table for the relevant

files based on the partition present in the file\_path column. **Persist the Manifest to S3**

You can persist this list of files to S3 for later reference when loading to Snowflake through:

Example of a utility function that can generate a manifest file and stores it in S3: https://github.com/compassdigital/airflow-cdl- dataops/blob/dev/dags/cdl\_library/utils/manifest\_redshift.py

**Preparing Snowflake**

For Snowflake to access the files from Iceberg (remember that Iceberg stores the data files within S3), an external stage must be

created.

An external (S3) stage specifies where data files are stored so that the data in the files can be loaded into a table.

Data can be loaded directly from files in a specified S3 bucket, with or without a folder path (or prefix, in S3 terminology). If the path ends with / , all of the objects in the corresponding S3 folder are loaded.

**Using the existing Iceberg stage**

There is already an external stage that points to the bucket where Iceberg stores data in S3 under the name dataops\_edw.source.iceberg\_stage .

1 import boto3  
2 import JSON  
3 [...]  
4 data\_files = data\_files\_df.collect() 5 s3 = boto3.client("s3")

6 s3.put\_object(Body=json.dumps(data\_files), ...)

119

**Creating a stage**

If it is desired to create a stage that has a more specific path beyond the Iceberg bucket, then the stage can be created as follows:

1 2 3 4 5 6 7 8 9

CREATE STAGE dataops\_edw.source.<stage\_name>

URL='s3://<iceberg\_bucket>/<specific\_path>'

STORAGE\_INTEGRATION = s3\_integration

FILE\_FORMAT = (TYPE = parquet)

-- where iceberg\_bucket = 'cdl-dataops-datalake-dev-us-east-1'

-- specific\_path = the desired path within the bucket

-- s3\_integration = This is an existing integration containing relevant AWS creds

-- FILE\_FORMAT/TYPE = Iceberg stores files in parquet format

**Copying from stage to Snowflake table**

Accessing data from a stage is done via referencing @dataops\_edw.source.<stage\_name> .  
Further path specification can be suffixed to the stage as @dataops\_edw.source.<stage\_name>/path/to/data . Copying specific files from a stage to a table is done by:

1 copy into <table\_name> from 2(  
3 select  
4 <columns\_to\_load>

5 from @dataops\_edw.source.<stage\_name>  
6)  
7 files=(<files>) on\_error=continue force=true; 8

1. 9  -- where table\_name = the table you want to load data into
2. 10  -- columns\_to\_load = the columns from the parquet file. These are referenced via '$1:<column>' ex: $1:employee\_
3. 11  -- stage\_name = the external stage
4. 12  -- files = a list of relative paths from the stage. Ex: files = (/path/to/data.parquet', /path/to/data2.parquet

Loading Iceberg data to Redshift  
Loading data ***directly*** from Iceberg to Redshift is not possible in our setup. The reason for this is because of the limitations of Redshift’s

cross-region COPY support for parquet files. Our Redshift instance is in east-2 while our Iceberg parquet data files exist in east-1 .  
To circumvent this issue, the easiest and quickest solution is to write transformed data both to Iceberg **and** to an S3 location in an east-2

bucket (ex: cdl-bi-raw-source ).  
Writing a Spark DataFrame to an S3 location is done by:

The SQL to load data from parquet files in S3 to a Redshift table can be done as follows:

i )

1 df.write.mode(mode).parquet("s3://path/to/parquets")  
2 # df = The same DataFrame that was used to write to Iceberg  
3 # mode = Strategy for how to write data to a location if there are existing files  
4 # More on modes: https://spark.apache.org/docs/latest/sql-data-sources-load-save-functions.html#save-modes

1. 1  sql = f"""
2. 2  COPY <table\_name\_to\_load>
3. 3  from '<s3://path/to/parquets>'
4. 4  iam\_role '<iam\_role>'
5. 5  format as parquet;
6. 6  """

120

**Note:** It is important to handle de-duplication in a manner that is relevant to your pipeline’s requirements.

121

Transform

**Defining the Schema**

A schema is suggested (and most times required) to be defined to help form a valid structure of data. Schema is used to conform the data and can be used to highlight errors when expected types don’t match or to fill in the gaps when there is missing data.

Data Types - Spark 3.4.0 Documentation

A Spark schema consists of StructTypes that contain a list of StructFields. Each StructField contains a name and a type. For example:

1 2 3 4 5

schema = StructType([

StructField("employee\_id", IntegerType()),

StructField("name", StringType()),

StructField("active", BooleanType())

])

**Reading into a DataFrame**

A DataFrame can be created from data that is resolved from some source (as shown in the example below).

Source data can come from many places and is entirely dependent on your extract phase logic, but mostly, it comes from S3 in the

form of CSV or JSON file(s).  
You can either read these files into a python object OR read the files directly into a Spark DataFrame using one of the spark read options:

Option 1 - Read into a python object (JSON example):

1 2 3 4 5 6 7 8 9

10 11 12

[...]

def get\_data(bucket, key)

response = s3.get\_object(Bucket=bucket, Key=key)

if "Body" in response:

data = response["Body"].read()

return data

return None

data = get\_data("some-bucket", "path/to/file")

if data:

df = spark.createDataFrame(data, schema=schema)

[...]

Option 2 - Read directly from source to spark (JSON example):

1 df = spark.read.option("multiline", "true").schema(schema).json(path\_to\_json)

The multiline option is for when the JSON file is written with newline characters

The JSON path typically will be an S3 URI  
Once the DataFrame is created, any transformations or selecting out of data can be done from here.

**Writing to Iceberg**

In our architecture, we write this transformed data to an Apache Iceberg table.

122

To wErnitseutroeItcheabtethrgewicebneregdctaotawlorigteiswcitohrinreacntlyIcrebferregncaetdal(othgi,sawnoduwldithbiendtheefinceadtawloitgh,inwethme uspstawrkristeubtomait tcaobnlefiginuratdioantas)b,aasse.well as the database within the catalog. Apache Iceberg

It is recommended that the database exists before your Spark application tries to write to it.  
To do this, execute a spark-sql shell with the options configured to the relevant iceberg catalog, and run the appropriate sql to create the database.

For example:

The syntax for writing to Iceberg is fairly simple, there is a choice between writing directly using the DataFrame or through Spark SQL. To write with the DataFrame we use the DataFrameWriterV2 writeTo() method Writes :

For example:

1 df.writeTo("iceberg\_catalog.database.table").create()  
To write the DataFrame to Iceberg through Spark SQL the DataFrame must first be used to create a temporary view that can be

referenced within Spark SQL. With the view, it can be used to insert records into an Iceberg table: For example:

1 > use iceberg\_catalog;  
2 > create database dummy\_database;

1. 1  df.createOrReplaceTempView("df\_view")
2. 2  spark.sql(

3  
4  
5  
6  
7 8)

"""

create table iceberg\_catalog.database.table using iceberg

# PARTITIONED BY (col1, col2) <- this is an example of table partitioning

as select \* from df\_view

"""

123

Summary of Jobs

**Sub- group**

**Name**

**Schedule**

**Source**

**Source Backup**

**Transform**

**Destination**

**Redshift Table(s)**

**Environm ent**

**Link To Job**

**Link to Code**

**Group**

1

CDL

Digital Success

DXM\_Smartsh eet

30 21 \* \* \*

https://app. smartsheet .com/repor ts/5gw8cq 298cFXRf 6pwfPc8w QXvJrFjw mhCF4HJ 5F1? view=grid

S3

Yes

cdl\_bi\_edw

datamart.dxm\_smartsheet

Airflow – Production

https://994 61153- dff4-4925- a79a- 34be236be 719.c3.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=DX M\_Smarts heet

Github

2

Cleantelligent

N/A

cleantelligent\_s ftp\_load

06\*\*\*

S3

N/A

No

Airflow – Production

https://994 61153- dff4-4925- a79a- 34be236be 719.c3.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=cle antelligent \_sftp\_load

Github

cansftp.com pass- canada.com

3

4

public.cleantelligent\_allinvalidinspec tions public.cleantelligent\_dumptimes public.cleantelligent\_newinspectionc ustomanswers public.cleantelligent\_newinspection details public.cleantelligent\_newinspections

CDL (?)

Shopping Cart

datalake\_shop ping\_cart

\*/10 \* \* \* \*

cdl- datalake

No

Yes

cdl\_bi\_edw

source.datalake\_shopping\_cart

Airflow – Dev

https://c04 b78fc- 26d0-4f24- a58d- 9d6aafdba ad0.c1.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=dat alake\_sho pping\_cart

Github

DataOps Internal

N/A

ftp\_sync

\*/15 \* \* \* \*

?

N/A

No

N/A

N/A

Airflow – Production

https://994 61153- dff4-4925- a79a- 34be236be 719.c3.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=ftp \_sync

Github

5

6

7

P2

P2

P2

Tasks

Tasks

Orders

P2\_TASKS\_D AILY\_EXTRAC T

P2\_TASKS\_H OURLY\_EXTR ACT

P2\_ORDERS\_ DAILY\_ETL

0 2 \* \* \*

0 0-1,3-23 \* \* 0-5

0 1 \* \* \*

API

API

API

S3

S3

S3

Yes

Yes

Yes

Iceberg

Redshift Snowflake

Iceberg

Redshift Snowflake

Iceberg

Redshift Snowflake

source.p2\_tasks

source.p2\_tasks

source.p2\_orders

Airflow – Production

Airflow – Production

Airflow – Dev

Airflow

Airflow

Airflow

Github

Github

Github

124

8

P2 Orders P2\_ORDERS\_ HOURLY\_EXT

RACT

30 2/1 \* \* 1-6

API

S3

Yes

Iceberg source.p2\_orders Airflow – Airflow Github

Redshift Dev Snowflake

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9  10  11  12 | P2 | Menu Items | P2\_MENU\_ITE MS\_WEEKLY\_ EXTRACT | @weekly | API | S3 | Yes | Iceberg  Redshift Snowflake | source.p2\_menu\_groups\_dimensio n  source.p2\_menu\_group\_items\_dim ension | Airflow – Dev | https://c04 b78fc- 26d0-4f24- a58d- 9d6aafdba ad0.c1.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=P2 \_MENU\_IT EMS\_WEE KLY\_EXTR ACT | Github |
| SFTP | Touch The Table | touch\_the\_tabl e\_HRDash\_sft p | daily | Redshift |  | No | SFTP | datamart.hr\_dashboard\_employee \_master\_vw  datamart.marquise\_can\_payroll\_ summary\_fact | Airflow- Prod | https://994 61153- dff4-4925- a79a- 34be236be 719.c3.us- east- 2.airflow.a mazonaws. com/tree? dag\_id=tou ch\_the\_tab le\_HRDas h\_sftp | https://gith ub.com/co mpassdigit al/airflow- cdl- dataops/tre e/prod/dag s/touch\_th e\_table\_H RDash\_sft p |
| SFTP | Shelf Engine | shelfengine\_sft p\_upload | daily | Redshift |  | No | SFTP | source.cg\_dw\_cdl\_us\_source\_can \_pos\_order\_detail  source.cg\_dw\_cdl\_us\_source\_edw \_org\_hierarchy\_dim  source.cg\_dw\_cdl\_us\_source\_can \_pos\_order\_header  datamart.pos |  | https://c04 b78fc- 26d0-4f24- a58d- 9d6aafdba ad0.c1.us- east- 2.airflow.a mazonaws. com/code? dag\_id=sh elfengine\_ sftp\_uploa d |  |
| Smartsheets | SMARTSH EET\_CDL \_NA\_PRO JECT\_PO RTFOLIO\_ SUMMAR Y | ENDEAVOR\_S MARTSHEETS \_DATASCIENC E\_REPORT\_K UBE\_JOB | Daily | Smartshee t | S3 | Yes | Iceberg Redshift Snowflake | Iceberg - smartsheets.smartsheet\_cdl\_na\_pr oject\_portfolio\_summary  Redshift - source\_iceberg.smartsheet\_cdl\_na \_project\_portfolio\_summary  Snowflake - DATAOPS\_EDW.SOURCE.SMART SHEET\_CDL\_NA\_PROJECT\_POR TFOLIO\_SUMMARY | Kubernete s  EMR  Airflow - Dev | https://c04 b78fc- 26d0-4f24- a58d- 9d6aafdba ad0.c1.us- east- 2.airflow.a mazonaws. com/graph ? dag\_id=EN DEAVOR\_ SMARTSH EETS\_DA TASCIENC E\_REPOR T\_KUBE\_J OB&root= &execution \_date=202 2-05- 10T15%3A 30%3A00 %2B00%3 A00 |  |

125

13

SFTP to S3 Hourly SYNC

SFTP Maintenan ce

ENDEAVOR\_S FTP\_S3\_SYN C

Hourly

SFTP

SFTP

No

S3

Airflow Prod

SFTP server

https://994 61153- dff4-4925- a79a- 34be236be 719.c3.us- east- 2.airflow.a mazonaws. com/graph ? dag\_id=EN DEAVOR\_ SFTP\_S3\_ SYNC

https://gith ub.com/co mpassdigit al/airflow- cdl- dataops/bl ob/prod/da gs/projects /endeavor/ sftp\_maint enance/sft p\_s3\_sync \_dag.py

https://gith ub.com/co mpassdigit al/dataops- sftp- server- repo/blob/ main/maint enance\_sc ripts/sftp\_s ync\_to\_s3. py

14

MERAKI

ENDEAVOR\_ MERAKI\_REF RESH

Every 6 Hours

S3

YES

Iceberg Redshift Snowflake

source\_iceberg.meraki\_network\_cli ent\_devices

source\_iceberg.meraki\_network\_cli ents

source\_iceberg.meraki\_network\_cli ents\_device\_info

source\_iceberg.meraki\_networks

DATAOPS\_EDW.SOURCE.MERAK I\_NETWORKS DATAOPS\_EDW.SOURCE.MERAK I\_NETWORK\_CLIENTS DATAOPS\_EDW.SOURCE.MERAK I\_NETWORK\_CLIENTS\_DEVICE\_I NFO DATAOPS\_EDW.SOURCE.MERAK I\_NETWORK\_DEVICES

Airflow Prod

https://bdc 0ea40- c9db-47ac- 9c57- 8a617c0d4 56c.c68.us -east- 1.airflow.a mazonaws. com/tree? dag\_id=EN DEAVOR\_ MERAKI\_ REFRESH

https://gith ub.com/co mpassdigit al/dataops \_endeavor \_container/ tree/dev/m eraki

15

SHELF ENGINE

SHELF ENGINE

ENDEAVOR\_K UBE\_SHELF\_ ENGINE\_CDV \_DAILY\_EXPO RT

Daily

REDSHIF T

NO

SFTP

P2 Mobile Orders tables

Airflow Prod

https://bdc 0ea40- c9db-47ac- 9c57- 8a617c0d4 56c.c68.us -east- 1.airflow.a mazonaws. com/tree? dag\_id=EN DEAVOR\_ KUBE\_SH ELF\_ENGI NE\_CDV\_ DAILY\_EX PORT

16

VERIZON

THRIVE

ENDEAVOR\_V ERIZON\_THRI VE\_DATASHA RE

Daily

REDSHIF T

NO

S3

P2 Mobile Orders tables

Airflow Prod

https://bdc 0ea40- c9db-47ac- 9c57- 8a617c0d4 56c.c68.us -east- 1.airflow.a mazonaws.

MERAKI\_ REFRESH

Meraki API

126

ETL Jobs

SFTP and S3 Bucket Sync Process

**How does the process work?**

The airflow job connects to the EC2 instance hosting the SFTP server and syncs the directory (/ftp\_home/<user directory>) to S3 (s3://cdl- data-ops-ftp-sync/<user directory>).

**Are there any specific applications required for this process?**

The DAG runs in 2 steps

1. The job connects to the SFTP server using the private key file CDL-BI-EC2-FTP.pem and executes the script sftp\_sync\_to\_s3.py in the SFTP server. The script outputs the list of all the available SFTP users. And stores the output in the AWS S3 location s3://cdl-data-

ops-ftp-sync/list\_of\_dirs\_to\_sync.json .

1. The job connects to the SFTP server using the private key file CDL-BI-EC2-FTP.pem and executes the script sftp\_sync\_to\_s3.py in the SFTP server with command line parameter SFTP user directory e.g. /ftp\_home/trentu\_volante . The script takes the parameter and runs the AWS CLI S3 SYNC command for the user directory as a separate task. e.g. aws s3 sync /ftp\_home/trentu\_volante

s3://cdl-data-ops-ftp-sync/trentu\_volante .

PROD job : Airflow Dev Job : Airflow

GIT Repo: https://github.com/compassdigital/airflow-cdl- dataops/blob/dev/dags/projects/endeavor/sftp\_maintenance/sftp\_s3\_sync\_dag.py

https://github.com/compassdigital/dataops-sftp-server-repo/blob/main/maintenance\_scripts/sftp\_sync\_to\_s3.py

INFO: The AWS CLI uses the AWS configuration in the home directory .aws . The SFTP server uses ENDEAVOR Batch Access Key/Secret Key to sync the SFTP files with the S3 bucket folder.

Note: Whenever the ENDEAVOR Access Key/Secret Keys values are rotated, The AWS configure command needs to be executed with the new Access Key/Secret Keys

127

API Process

**Extract:**

This phase includes making calls to API, gathering the responses, and storing the responses in S3. Further details: Extract **Transform:**

This phase includes taking stored data from S3 (from the Extract phase), loading it into a Spark DataFrame, making relevant transformations (such as flattening, or other complex calculations), and storing the data in an Apache Iceberg table(s)Further details: Tra

nsform

**Load:**

This phase includes taking the data from an Apache Iceberg table and storing it in Snowflake or Redshift. Referencing data files for loading to Snowflake is aided by creating a “manifest” that is simply a file that contains all the relevant data files from an Apache Iceberg table that will be referenced in a Snowflake load. Further details: Load

128

**Screenshot of an Airflow DAG that performs an ETL pipeline**

129

NutriSlice

Nutrislice is a client of CD. They have an SFTP server where every day they drop the S3 data. The Data Engineer needs a job that takes that data and syncs it to an S3 bucket daily. Another job is required that takes the data from the S3 bucket and loads it into Iceberg, Redshift, and Snowflake. There are three separate load steps for the tables. The tables and their load format are below:

**Table Name**

customers\_dimension foods\_dimension locations\_dimension orders

transactions order\_items

**General Load Steps**

**Load Format**

Slowly Changing Dimensions (SCD) Type 2 SCD Type 2  
SCD Type 2  
Deduplication

Deduplication Raw load

Each load format has the same step except for the Iceberg load (where each format varies). The General Load Steps are:

1. Check for new data on the S3 bucket  
2. Download data from S3  
3. Load the CSV files into Iceberg (Spark Step)  
4. Write the entire table as a parquet to S3  
5. Truncate and reload the table in Redshift with the new parquet 6. Truncate and reload the table in Snowflake with the new parquet

**Load Formats Raw Load**

The raw load involves adding an etl\_extract\_timestamp column to the new data and appending it to the existing table. No other transformations are done on the data.

**Iceberg Load Steps**

1. Load the CSV file into a Spark DataFrame  
2. Add an etl\_extract\_timestamp column to the DataFrame 3. Append data to the existing Iceberg table  
4. Proceed to Step 4 of General Load Steps

**Deduplication Load**

The deduplication load involves adding a row\_hash column to the table. The row\_hash uses the SHA2 hashing algorithm to provide a unique value for each distinct row in the dataset. The row\_hash is then used to insure no duplicate data gets loaded in by only inserting records if the row\_hash does not exist yet.

**Iceberg Load Steps**

1. Load the CSV file into a Spark DataFrame  
2. Add an etl\_extract\_timestamp column to the DataFrame  
3. Add a row\_hash column to the DataFrame  
4. Merge into the existing Iceberg table where the row\_hash does not exist

130

5. Proceed to Step 4 of General Load Steps

**Slowly Changing Dimensions Load**

A dimension table is where each unique row can be found by a combination of one or more primary key columns. A Slowly Changing Dimension (SCD) introduces the ability to maintain a full history of the table while updating existing records over time. SCD Type 2 adds the following extra columns to the table:

**Column Name**

scd\_start scd\_end scd\_status

scd\_hash

**Iceberg Load Steps**

**Column Purpose**

The timestamp at which the record becomes active (the record is the most up to date one for a specific primary key)  
The timestamp at which the record becomes inactive (a new record has been added to the table with the same primary key)

Whether the record is the most up-to-date one for that specific set of primary key(s). True if active (most up-to-date), False if inactive (not the most up-to-date).

A column that contains the results of the SHA2 hashing algorithm to all non-SCD and ETL columns. Used to determine whether a record’s content is unique when compared against what exists in Iceberg already.

1. Load the CSV into a Spark DataFrame
2. Add SCD columns
3. For new records: Set all scd status values to “True“
4. For new records: Set all scd\_start values to the current timestamp
5. For existing records: If there exists a new record with a different scd\_hash on the same primary keys, set the currently active record

( scd\_status = True )'s to scd\_status = False to the current timestamp

1. Insert new records, set scd\_end to current timestamp for any records that have scd\_status = False and scd\_end = 9999-12-31

The Airflow script for this job does the following:  
1. Get the job definition from the jobDefinition DynamoDB table.  
2. Construct the payload for submitting the job to the EMR (on EKS) cluster using the job definition. 3. Submit the job to the EMR cluster.

**Potential Problems**

1. File not found error during Spark step:  
If it is over the weekend, you can safely ignore it. If it continues through during weekdays, reach out to Matt.

2. Merge into matching more than one record:  
a. Take a look at the data and see if you can find why more than one record is being returned in the merge b. Reach out to @Anirudh.Kilambi  
c. If step 2a fails, then contact Matt.

131

Zendesk Organizations API Datashare

Purpose

Business Analysts are starting to use Zendesk for their work. Currently, any data related to Compass Units are manually inserted by Business Analysts into Zendesk’s Organization API. This method is unsustainable and time-consuming. The Business Analysts have asked to create a datashare in Iceberg that updates the Organization API with data for all Canadian and US units.

Data needs to be extracted from the following tables:  
1. iceberg\_catalog.cg\_dw\_us.source\_org\_hierarchy\_dim

2. iceberg\_catalog.cg\_dw\_us.source\_org\_unit\_dim The mapping of zendesk\_field to iceberg\_table\_column is:

**Zendesk Field**

unit\_account unit\_country division  
unit\_name unit\_number unit\_region unit\_sector unit\_street\_address unit\_city

unit\_state unit\_zip\_code unit\_phone\_number

Code

**DAGs**

**Column in Iceberg Table**

org\_hierarchy\_dim.org\_complex\_name org\_hierarchy\_dim.erp\_system\_id org\_hierarchy\_dim.org\_division\_name org\_hierarchy\_dim.org\_unit\_name org\_hierarchy\_dim.erp\_entity\_id org\_hierarchy\_dim.org\_region\_name org\_hierarchy\_dim.org\_sector\_name org\_unit\_dim.org\_unit\_street org\_unit\_dim.org\_unit\_city org\_unit\_dim.org\_unit\_state\_long org\_unit\_dim.org\_unit\_zip org\_unit\_dim.org\_unit\_phone

This updated job runs in Airflow 2.2 Airflow - Login - Airflow (DAG ID: zendesk\_organizations\_update) What doe the Code do?

The job is split into 3 parts:

**1. Python Step**

In this step, you need to perform the required Get requests to retrieve all organization data from the Organization API. This task runs through EKS. Click here to access the code.

**2. Spark Step**

In this step, we flatten the response received in step 1 and convert it to spark DataFrame (response). Next, we Join responses to the org\_hierarchy\_dim and org\_unit\_dim .

Primary Keys: erp\_entity\_id and erp\_system\_id . erp\_entity\_id is the first word from the name field of the organization API (see example).

132

must be taken from the tags field of the API. Create the JSON object that will then be used to put/post back to the API. Click here to access the code for this task.

Example response of one organization from the API

In this case, erp\_entity\_id = 000C2 . erp\_system\_id is always one of two values: 2010 (Canada) or 1001 (united states) . **3. Python Step**

Split the JSON into chunks of 100 units, perform the put and post requests back to Zendesk to update/create records. The code for this can be found at: update\_or\_add\_organization\_data.py

Zendesk only supports the bulk update or creation of 100 records simultaneously. This is taken care of for you in step 3 of the code.

Adding more fields to the API  
To add more fields to the API, a few things need to be changed in the script. Since we flatten the JSON response in step 1, we need to

specify the schema for Spark to read.  
1. Any new field we add must be captured from Zendesk in future responses. Add the new field to organization\_fields\_schema .

erp\_system\_id

133

2. Update the fields\_map dictionary in the spark script.

The key should be the field name as Zendesk sees it, and the value should be the corresponding column name in the datalake.

The AWS secret for this job is secret\_name = "zendesk\_organization\_api", region\_name = "us-east-2"

134

Zendesk

Zendesk ticket data is provided to us through REST API. This paginated data is retrieved in batches of 100 records per page.  
Once the data is normalized using Panda's library, we filter out expired records and invalidate them. This is because CDL follows the Slowly Changing Dimension concept.  
Once the expired records are invalidated and new records are added, the data is loaded to Redshift.

**New development**

We have started using an Incremental loading API endpoint, using the cursor-based pagination. It loads the changed and newly added ticket-related data in the last 24 hours. Further information can be found here: Incremental Exports . All the tables are loaded using Cursor Based Pagination, except the Comments table which is time-based pagination.

**New Development**

We have migrated the existing pipeline to the new stack of Spark + Iceberg, and started incremental loading and implementing Slowly Changing Dimension 2 on all the tables.

You can find the Zendesk Developer API here: Tickets

135

Smartsheets

Smartsheet is a software as a service offering, One of its services is creating and managing spreadsheets and generating reports. The DAGs in this group automate the processing of the data shared by the Smartsheet application.

Steps involved in automating this process:

1. Under AWS SES an Inbound Email Receiving configuration is made for an email id for e.g. dataops\_reports@cdl-bi.awsapps.com. Under the configuration setting the Email sent to this email id will be saved in an S3 location for e.g. s3://cdl-bi-inbound-

mail/SmartSheetEmailDataShare/

1. In the Smartsheet application, The report should be scheduled to the above-configured email.
2. An ETL job is required to parse the new email files, extract the XLSX file, and format the files into CSV. It should be staged in the S3

location for the next steps to consumption.

1. Execute Spark jobs to refresh the Iceberg table with the new file.
2. Refresh respective Snowflake and Redshift tables from Iceberg tables.

136

SMARTSHEET\_CDL\_NA\_PROJECT\_PORTFOLIO\_SUMMARY

This job/DAG refreshes Smartsheet *“Datascience Report”* data in Iceberg, Redshift, and Snowflake tables.  
The following dataset contains all the project tracking data that is currently being tracked in the Smartsheet Application:

**ETL Job Details**

**DAG Name**

**Airflow Job Link**

**Script Path**

**TASK: REFRESH\_S3\_SOURCE\_FILES\_LIS T**

**TASK: ICEBERG\_REFRESH**

**TASK: REDSHIFT REFRESH**

**TASK: SNOWFLAKE\_REFRESH**

DAG script path

Python script run in EKS cluster, ETL logics to parse the email attachment and transform the data into CSV format for next steps to consume.

PySpark script runs in EMR cluster to refresh Iceberg tables.

Python script runs in EKS cluster to refresh Redshift Table.

Python script runs in EKS cluster to refresh Snowflake Table.

**ENDEAVOR\_SMARTSHEETS\_DATASCIENCE\_REPORT\_KUBE\_JOB**

Airflow

https://github.com/compassdigital/airflow-cdl- dataops/blob/dev/dags/projects/endeavor/smartsheets/datascience\_report\_dag.py

https://github.com/compassdigital/dataops\_endeavor\_container/blob/dev/smartsheet/parse\_s3\_email.py

https://github.com/compassdigital/terminus-nuclear- plant/blob/dev/endeavor/smartsheets/datascience\_report\_ib\_table\_refresh.py

https://github.com/compassdigital/dataops\_endeavor\_container/blob/dev/smartsheet/redshift\_refresh.py

https://github.com/compassdigital/dataops\_endeavor\_container/blob/dev/smartsheet/snowflake\_refresh.py

Steps involved in automating this process:  
1. Under AWS SES an Inbound Email Receiving configuration is made for email ID dataops\_reports@cdl-bi.awsapps.com. Email sent

to this ID is saved in S3 location s3://cdl-bi-inbound-mail/SmartSheetEmailDataShare/ .

2. In the Smartsheet application, The Data science report is scheduled to send an email daily at 8.00 AM PST. to dataops\_reports@cdl- bi.awsapps.com.

3. An EKS job runs to pick up the newly pushed file every day to read the email message and process the attached XLSX file to convert to CSV format and push into AWS S3 staging area s3://cdl-dataops-datalake-pipeline-

storage/smartsheets/smartsheet\_cdl\_na\_project\_portfolio\_summary/Data-Science-Export.csv .

4. Refresh the Iceberg cluster with newly refreshed file smartsheets.source\_smartsheet\_cdl\_na\_project\_portfolio\_summary .

5. Refresh the Snowflake and Redshift tables: source\_iceberg.smartsheet\_cdl\_na\_project\_portfolio\_summary DATAOPS\_EDW.SOURCE.SMARTSHEET\_CDL\_NA\_PROJECT\_PORTFOLIO\_SUMMARY .

This email is sent to an ID that can only be read and processed manually by email clients (like Outlook). To automate the processes of reading the email and consuming the data attachments, the email messages are stored in a accessible locations where the ETL script can scan the location for new emails and parse the contents of the email.

137

Operator Analytics RLS Intake Submission Form Load

Problem

With the looming launch of Operator Analytics at the end of October, the Data Science and BI teams require a new dataset from Smartsheet to be ingested into our datalake and data warehouse.  
Data needs to be extracted from this sheet into Iceberg, Redshift, and Snowflake twice on daily basis.

Smartsheet pipelines we’ve created in the past have leveraged the email automation feature of sheets to get sent data that we then transform and load into our Compass data locations. With this pipeline, we want to leverage the Smartsheet REST API to retrieve the data.

Important Information  
General information about table names and file paths can be found below:

1{

1. 2  "snowflake\_source\_database":"",
2. 3  "snowflake\_source\_schema":"",
3. 4  "snowflake\_source\_table":"",
4. 5  "dataset\_input\_path":"s3://cdl-bi-raw-source/smartsheet/oa\_rls\_intake\_form\_submissions/files\_to\_load.json",
5. 6  "source\_file\_type": "json",
6. 7  "parquet\_path":"s3://cdl-bi-raw-source/smartsheet/oa\_rls\_intake\_form\_submissions/output.parquet",
7. 8  "catalog\_database":"smartsheet",
8. 9  "catalog\_table":"oa\_rls\_intake\_form\_submissions",
9. 10  "snowflake\_database":"DATAOPS\_EDW",
10. 11  "snowflake\_schema":"SOURCE",
11. 12  "snowflake\_table":"smartsheet\_oa\_rls\_intake\_form\_submissions",
12. 13  "redshift\_schema":"source",
13. 14  "redshift\_table":"smartsheet\_oa\_rls\_intake\_form\_submissions"
14. 15  }

16

Code

**DAG**

This job runs in AF 2.2 (DAG ID: smartsheets\_oa\_rls\_intake\_form\_load ). **What does the Code do?**

The job is split into 3 parts:

**Step 1: Python**

The file can be found here.  
This step involves using the Smartsheet REST API to retrieve our data. To use the API, a few key things are needed:

Due to the nature of API return response, a column mapping is required to convert the columnIDs back into usable column names for further processing.

138

**1.1 API Key**

The API key exists in AWS SecretsManager secret\_name = smartsheet\_pmo\_sa , region\_name = us-east-1 . This API key connects to the following Smartsheet account: noreply-pmo@compassdigital.io - the owner of the sheet.

**1.2 Sheet ID**

The sheet ID is the main identifier used to access its data through the API: sheetID = 3742773426841476 .

**1.3 ColumnID Mapping**

When the data is retrieved from the API, column names are not used, only columnIDs are used. The mapping of columnID to column name is below:

1. 1  column\_mapping = {
2. 2  4103993758640004:"operator\_email",
3. 3  6321608988092292:"admin\_or\_units",
4. 4  7549062859777924:"compass\_sector",
5. 5  3049014633490308:"regions\_chartwells\_sectors",
6. 6  3649997059188612:"regions\_eurest\_sector",
7. 7  2156318331037572:"regions\_flix\_sector",
8. 8  1726155881506692:"regions\_ra\_bi\_sector",
9. 9  3273510426568580:"units\_central\_division",
10. 10  2663841966385028:"units\_northeast\_division",
11. 11  148717707782020:"units\_west\_division",
12. 12  6299168857712516:"units\_bank\_of\_america",
13. 13  4656363194345348:"units\_mid\_atlantic\_region",
14. 14  8520392799217540:"units\_mid\_east\_region",
15. 15  471767967917956:"units\_northwest\_region",
16. 16  6335970284988292:"units\_pfizer\_region",
17. 17  6327311630919556:"units\_procter\_and\_gamble\_region",
18. 18  3225947421861764:"units\_raytheon\_region",
19. 19  1069091618350980:"units\_united\_healthcare\_group\_region",
20. 20  4289950944388996:"units\_utc\_region",
21. 21  5047104286549892:"units\_wells\_fargo\_region",
22. 22  8269066110560132:"units\_dc\_metro\_region",
23. 23  2626136213612420:"units\_merck\_region",
24. 24  1139373154559876:"units\_new\_jersey\_natl\_accts\_region",
25. 25  1136723159738244:"units\_barnhart\_region",
26. 26  5630306084513668:"created",
27. 27  2648301331867524:"created\_by",
28. 28  6210615926646660:"row\_id"
29. 29  }

Once we query the API for the data, we write it to S3 as a JSON for further processing. The format of the JSON we write to S3 is:

1 [{each:dictionary, is:one, row:in, smart:sheet}]

**Step 2: Spark**

File  
Schema for the iceberg table Config for the table

This script handles loading the JSON into Iceberg and Snowflake. JSON is read from S3 into a Spark DataFrame where we then handle performing any transformations. For this dataset, we perform some string formatting on timestamps, create an

etl\_row\_hash column, and add an etl\_extract\_timestamp .  
If a table already exists in the iceberg, then we merge in records that don’t already exist. Otherwise, we load the entire dataset.

139

After the iceberg load is done, we take a DataFrame that contains all data that was loaded in on that specific job run. The DataFrame is used to write to Snowflake, and then it is written to S3 as a parquet.

**Step 3: Redshift Load Step**

File

This script handles loading the final DataFrame parquet taken in the last stage of the Spark step into Redshift.

140

myLMS

myLMS is a corporate learning program used by Compass. The DAGs in this group handle the loading of data shared from myLMS. As of 10/28/22, we get three datasets from the US team on a daily basis:

Certifications - The current certifications of Compass Associates Courses - Courses offered by Compass Group  
Programs - Programs Completed by Compass Associates

General Steps to Process LMS Data  
Currently, all myLMS data gets shared with us by Compass Digital’s US data warehouse team. The data is transferred to the following S3

bucket: s3://cg-dw-prod-exportcdl-us-east-1  
Every day, a new file is uploaded to the data-warehouse export S3 bucket for each dataset from the US DW team. The current tables must

be truncated, and reloaded with the data from the new files.

1. The data from the export bucket must be synced into a Compass Digital controlled S3 bucket, as we only have read access to the bucket above.
2. The data is a simple export from the myLMS software, transformations may be needed on the data to ensure it is easily processed by Spark and other Database applications.
3. Execute a spark job to load the Iceberg table with a new file.
4. Execute python scripts to load new data into Redshift and Snowflake

Troubleshooting  
You may come across the following situations when working on this project. We have listed the possible workarounds for each situation:

**1. File not found error at Iceberg Load**

For each load, we read the file based on the load day’s date. However, the US DW team sends the data at around midnight the night before. On occasion, the data will be uploaded at 23:59, and sometimes at 0:01. As a result, the filing date might show one extra day behind.

**Certifications**

If there is no reformatted file on the date of the load, that means that no new files were uploaded to the US DW bucket. aws s3 ls the certification bucket to see if there are any net new files there.

**Courses and Programs**

If our S3 location shows no files with today’s date, or yesterday’s. Then no file was loaded. Check the export bucket to see if there are any new files there.

If no files are coming in on a consistent basis, reach out to kerry.howe@compass-usa.com - The LMS coordinator. If you’re still facing an issue, contact @Anirudh.Kilambi for more further assistance.

141

myLMS Certifications Refresh

This DAG refreshes the myLMS certifications data in Iceberg, Redshift, and Snowflake. All LMS refreshes in this directory point to the same Airflow DAG → mylms\_data\_load (see Airflow Job Link)

Potential problems with this load are documented here

Relevant Information

**ETL Job Details**

**DAG Name**

Airflow Job Link

DAG Script Path

TASK: reformat\_certification\_files

TASK: load\_certifications

TASK: load\_certifications\_redshift

TASK: load\_certifications\_snowflake

**Step 1: Reformat Certification Files**

**mylms\_data\_load**

Airflow - mylms\_data\_load  
DAG  
Github - Reformat LMS Dataset Github - Load Certification Github - Load LMS to Redshift Github - Load LMS to Snowflake

A python script that runs in an EKS pod.  
The US DW team sends an export from the LMS program with the certification files as a zipped CSV to `s3://cg-dw-prod-exportcdl-us-

east-1/lms/certification/` . The file name is in the format Certification\_Status\_Report\_{%m%d%Y}\_312.csv.gz . The reformat involves:

1. String formatting for date columns.
2. Some columns have many values separated by a space (known as array columns ). Each of these values needs to be split up into its

own individual row in the table.  
a. The reformatted file is then placed in s3://cdl-bi-raw-source/cg\_dw\_prod/LMS/reformatted\_files/{year}/{month of

year}/{day of month}/

**Step 2: Load Certifications**

Schema can be found at: s3://cdl-bi-raw-source/LMS/table\_schemas/certifications\_schema.json A Spark script that runs as a step in EMR.  
The reformatted certification file in step one is loaded into a Spark DataFrame where:

1. The existing table in Iceberg is Truncated.  
2. ETL columns are added.  
3. Deduplication is done.  
4. Parquet is written to S3 (file path: s3://cdl-bi-raw-source/LMS/parquets/certification.parquet/ ).

142

**Step 3: Load Certifications Redshift and Load Certifications Snowflake**

A Python script that runs in an EKS pod.  
Redshift Table: source.cg\_dw\_us\_mylms\_certifications  
Snowflake Table: DATAOPS\_EDW.SOURCE.cg\_dw\_us\_mylms\_certifications

1. The existing table in Iceberg and Redshift are truncated. 2. The parquet created in step 2 is loaded into the tables.

143

myLMS Courses Refresh

This DAG refreshes the myLMS courses data in Iceberg, Redshift, and Snowflake. All LMS refreshes in this directory point to the same Airflow DAG → mylms\_data\_load (see Airflow Job Link)

Potential problems with this load are documented here

Relevant Information

**ETL Job Details**

**DAG Name**

Airflow Job Link

DAG Script Path

TASK: sync\_s3\_buckets

TASK: load\_courses

TASK: load\_courses\_redshift

TASK: load\_courses\_snowflake

**mylms\_data\_load**

Airflow - mylms\_data\_load  
DAG  
Github - Sync S3 Buckets Github - Load Courses  
Github - Load LMS to Redshift Github - Load LMS to Snowflake

**Step 1: Sync S3 Buckets**

A python script that runs in an EKS pod.  
Since the S3 bucket, the US DW team uploads the data to is not under our control, we need to sync the files in their bucket to the one that we control. The directory s3://cg-dw-prod-exportcdl-us-east-1/lms/ is synced with s3://cdl-bi-raw-source/LMS/course/ .

**Step 2: Load Courses**

The schema is inferred from the header.  
A Spark script that runs as a step in EMR.  
The reformatted certification file in step one is loaded into a Spark DataFrame where:

The existing table in Iceberg is Truncated.

The column names in the CSV are not in a SQL/Python-friendly format, so all column names need to be reformatted into snake case, and certain characters are removed from column names.

Date columns have their string dates reformatted into DateType columns.  
ETL columns are added.  
Deduplication is done.  
Parquet is written to S3 (file path: s3://cdl-bi-raw-source/LMS/parquets/course.parquet/ ).

**Step 3: Load Courses Redshift and Load Courses Snowflake**

A Python script that runs in an EKS pod.  
Redshift Table: source.cg\_dw\_us\_mylms\_courses  
Snowflake Table: DATAOPS\_EDW.SOURCE.cg\_dw\_us\_mylms\_courses

The existing table in Iceberg and Redshift are truncated.

144

The parquet created in step 2 is loaded into the tables.

145

myLMS Programs Refresh

This DAG refreshes the myLMS Programs data in Iceberg, Redshift, and Snowflake. All LMS refreshes in this directory point to the same Airflow DAG → mylms\_data\_load (see Airflow Job Link)

Potential problems with this load are documented here

Relevant Information

**ETL Job Details**

**DAG Name**

Airflow Job Link

DAG Script Path

TASK: sync\_s3\_buckets

TASK: load\_programs

TASK: load\_programs\_redshift

TASK: load\_programs\_snowflake

**mylms\_data\_load**

Airflow - mylms\_data\_load  
DAG  
Github - Sync S3 Buckets Github - Load Programs  
Github - Load LMS to Redshift Github - Load LMS to Snowflake

**Step 1: Sync S3 Buckets**

A python script that runs in an EKS pod.  
Since the S3 bucket, the US DW team uploads the data to is not under our control, we need to sync the files in their bucket to the one we control. The directory s3://cg-dw-prod-exportcdl-us-east-1/lms/ is synced with s3://cdl-bi-raw-source/LMS/program/ .

**Step 2: Load Programs**

The schema is inferred from the header.  
A Spark script that runs as a step in EMR.  
The reformatted certification file in step one is loaded into a Spark DataFrame where:

The existing table in Iceberg is Truncated.

The column names in the CSV are not in a SQL/Python-friendly format, so all column names need to be reformatted into snake case, and certain characters are removed from column names.

Date columns have their string dates reformatted into DateType columns.  
ETL columns are added.  
Deduplication is done.  
Parquet is written to S3 (file path: s3://cdl-bi-raw-source/LMS/parquets/programs.parquet/ ).

**Step 3: Load Courses Redshift and Load Courses Snowflake**

A Python script that runs in an EKS pod.  
Redshift Table: source.cg\_dw\_us\_mylms\_programs  
Snowflake Table: DATAOPS\_EDW.SOURCE.cg\_dw\_us\_mylms\_programs

The existing table in Iceberg and Redshift are truncated.

146

The parquet created in step 2 is loaded into the tables.

147

Summary of Data set by Vendor

**Vendor**

Agilysis

Raw Agilysis POS data to be used in the Loss Prevention Dashboard

agilysys\_loss\_prevention\_cancelled\_ch ecks\_detail agilysys\_loss\_prevention\_discount\_det ail agilysys\_loss\_prevention\_discounts\_co upons\_detail agilysys\_loss\_prevention\_linked\_trans actions agilysys\_loss\_prevention\_log\_on\_off\_d etail agilysys\_loss\_prevention\_manager\_ov erride\_detail agilysys\_loss\_prevention\_modifier\_det ail agilysys\_loss\_prevention\_no\_sales\_de tail agilysys\_loss\_prevention\_over\_short\_d etail agilysys\_loss\_prevention\_product\_des cription\_detail agilysys\_loss\_prevention\_sales\_detail agilysys\_loss\_prevention\_store\_detail agilysys\_loss\_prevention\_tax\_detail agilysys\_loss\_prevention\_tender\_detail agilysys\_loss\_prevention\_tender\_type agilysys\_loss\_prevention\_transaction\_ detail agilysys\_loss\_prevention\_voided\_open \_checks\_detail agilysys\_old\_layout\_transaction\_units agilysys\_old\_layout\_transactions agilysys\_transaction\_units agilysys\_transactions

**Contacts Descriptions Data warehouse tables Airflow DAG:**

148

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CDL |  | CDL app data (Thrive, Boost, Frictionless etc) | p2\_brands\_config p2\_brands\_configs\_delivery\_destinatio ns p2\_brands\_configs\_mealplan p2\_brands\_configs\_menu\_tabs p2\_brands\_configs\_promotions p2\_brands\_configs\_types\_of\_kds p2\_datalake\_users p2\_group\_location\_brands p2\_group\_locations p2\_groups p2\_kds\_devices p2\_location\_brand\_delivery\_hours p2\_location\_brand\_hours p2\_location\_brand\_is\_supported p2\_location\_brand\_menu\_hours p2\_location\_brand\_menus p2\_location\_brand\_terminals p2\_location\_brand\_type\_of\_kds p2\_location\_brands p2\_locations p2\_menu\_group\_items\_dimension p2\_menu\_groups\_dimension p2\_orders p2\_orders\_clone p2\_orders\_issues p2\_orders\_refunds p2\_orders\_temp p2\_shopping\_cart p2\_shopping\_cart\_item\_barcodes p2\_shopping\_cart\_item\_certificates p2\_shopping\_cart\_item\_option\_certific ates p2\_shopping\_cart\_item\_options p2\_shopping\_cart\_item\_options\_temp p2\_shopping\_cart\_items  p2\_shopping\_cart\_items\_is p2\_shopping\_cart\_items\_items\_discou nt p2\_shopping\_cart\_items\_nutritions p2\_shopping\_cart\_items\_options\_is p2\_shopping\_cart\_items\_options\_item s\_is p2\_shopping\_cart\_items\_options\_item s\_nutritions p2\_shopping\_cart\_items\_temp p2\_shopping\_cart\_payment\_methods p2\_shopping\_cart\_promo p2\_tasks p2\_users\_groups\_locations  cdl\_datalake\_compass\_apps\_loyalty\_e vents cdl\_datalake\_compass\_apps\_loyalty\_u sers cdl\_datalake\_compass\_apps\_order\_ite m\_options cdl\_datalake\_compass\_apps\_order\_ite ms cdl\_datalake\_compass\_apps\_orders cdl\_datalake\_compass\_apps\_shopping \_cart cdl\_datalake\_compass\_apps\_shopping \_cart\_item\_options cdl\_datalake\_compass\_apps\_shopping \_cart\_items cdl\_datalake\_compass\_apps\_transacti on\_item\_details cdl\_datalake\_compass\_apps\_transacti on\_item\_taxes cdl\_datalake\_compass\_apps\_transacti ons cdl\_datalake\_compass\_apps\_users  **[Deprecated Tables]:** |  |  |

149

api\_compass\_apps\_group\_brand\_men us  
api\_compass\_apps\_group\_details api\_compass\_apps\_group\_location\_br and\_hours api\_compass\_apps\_group\_location\_br and\_menu\_hours api\_compass\_apps\_group\_location\_br ands api\_compass\_apps\_group\_locations api\_compass\_apps\_group\_locations\_b ackup api\_compass\_apps\_group\_locations\_o ld

api\_compass\_apps\_groups api\_compass\_apps\_location\_brand\_de livery\_hours api\_compass\_apps\_location\_brand\_ho urs api\_compass\_apps\_location\_brand\_ho urs\_bkp api\_compass\_apps\_location\_brand\_m enu\_hours api\_compass\_apps\_location\_brand\_m enus api\_compass\_apps\_location\_brand\_ter minals api\_compass\_apps\_location\_brands api\_compass\_apps\_locations api\_compass\_apps\_menu\_item\_certifi cates api\_compass\_apps\_menu\_item\_option \_certificates api\_compass\_apps\_menu\_item\_option s

api\_compass\_apps\_menu\_items api\_compass\_apps\_order\_issues api\_compass\_apps\_order\_refunds api\_compass\_apps\_orders api\_compass\_apps\_orders\_backup api\_compass\_apps\_orders\_backup\_18 june api\_compass\_apps\_other\_locations api\_compass\_apps\_shopping\_cart api\_compass\_apps\_shopping\_cart\_ite m\_barcodes api\_compass\_apps\_shopping\_cart\_ite m\_certificates api\_compass\_apps\_shopping\_cart\_ite m\_option\_certificates api\_compass\_apps\_shopping\_cart\_ite m\_options api\_compass\_apps\_shopping\_cart\_ite ms api\_compass\_apps\_shopping\_cart\_ite ms\_items\_discount api\_compass\_apps\_shopping\_cart\_pa yment\_methods api\_compass\_apps\_shopping\_cart\_pro mo api\_compass\_apps\_shopping\_cart\_pro mo\_backup api\_compass\_apps\_user\_group\_locati ons

CDL

Midya Tsoy

CDL Marketing cloud backfeed files. Contain data around marketing email interactions with end users of our mobile apps

marketingcloud\_backfeed\_events marketingcloud\_backfeed\_jobs marketingcloud\_backfeed\_push\_notific ation marketingcloud\_backfeed\_subscribers marketingcloud\_backfeed\_subscribers \_email\_status marketingcloud\_backfeed\_survey\_all marketingcloud\_backfeed\_survey\_com ments

150

CDL

Legacy Mobile Apps platform for CDL (P1).

Data set is no longer update, it’s simply stored for archive and historical reporting purposes

digitalhospitality\_prod\_apps digitalhospitality\_prod\_brands digitalhospitality\_prod\_customers digitalhospitality\_prod\_groupcustomers digitalhospitality\_prod\_jdegroups digitalhospitality\_prod\_orderitemextrao ptions digitalhospitality\_prod\_orderitems digitalhospitality\_prod\_orders digitalhospitality\_prod\_promotionredem ptions digitalhospitality\_prod\_promotions digitalhospitality\_prod\_sites digitalhospitality\_prod\_unitbrands digitalhospitality\_prod\_unitcustomers digitalhospitality\_prod\_units digitalhospitality\_prod\_users

CDL

“Eat it or Delete It” – CDL Prototype App allowing users to rate different images of food. No longer in production, data is archived for historical purposes

eat\_it\_or\_delete\_it\_item\_certificates eat\_it\_or\_delete\_it\_item\_meta\_attribut es  
eat\_it\_or\_delete\_it\_items eat\_it\_or\_delete\_it\_messages eat\_it\_or\_delete\_it\_votes

Braintree Payment processing data for Braintree braintree\_transaction\_customer\_details braintree\_transaction\_other\_details

braintree\_transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| US Data Warehouse Team | Tom Maloney | Data share to us from the US Data warehouse team.  Include Hierarchy from JDE & SAP, POS Data from Agilysis, Nextep, Micros PCard concur data etc.. | cg\_dw\_can\_datamart\_can\_payroll\_su mmary\_fact cg\_dw\_canteen\_avec\_sales\_detail\_fac t cg\_dw\_canteen\_avec\_sales\_header\_f act cg\_dw\_canteen\_ocs\_sales\_detail\_fact cg\_dw\_canteen\_ocs\_sales\_header\_fa ct cg\_dw\_cdl\_us\_source\_can\_pos\_order \_detail cg\_dw\_cdl\_us\_source\_can\_pos\_order \_header cg\_dw\_cdl\_us\_source\_edw\_can\_f0005 cg\_dw\_cdl\_us\_source\_edw\_can\_f0006 cg\_dw\_cdl\_us\_source\_edw\_can\_f0901 cg\_dw\_cdl\_us\_source\_edw\_can\_f0902 cg\_dw\_cdl\_us\_source\_edw\_can\_f0911 cg\_dw\_cdl\_us\_source\_edw\_can\_payro ll\_fact cg\_dw\_cdl\_us\_source\_edw\_can\_payro ll\_summary\_fact cg\_dw\_cdl\_us\_source\_edw\_hr\_employ ee cg\_dw\_cdl\_us\_source\_edw\_org\_hierar chy\_dim cg\_dw\_cdl\_us\_source\_edw\_org\_unit\_ dim cg\_dw\_cdl\_us\_source\_fin\_client\_contr act\_dim cg\_dw\_cdl\_us\_source\_fin\_gl\_acct\_nbr \_dim cg\_dw\_cdl\_us\_source\_noma\_transacti ons cg\_dw\_cdl\_us\_source\_pos\_location\_di m cg\_dw\_cdl\_us\_source\_pos\_product\_di m cg\_dw\_cdl\_us\_source\_sc\_pcard\_conc ur\_fact cg\_dw\_cdl\_us\_source\_sc\_pcard\_spen d\_fact cg\_dw\_cdl\_us\_source\_sc\_pcard\_spen d\_fact\_test cg\_dw\_us\_org\_hierarchy\_dim cg\_dw\_us\_org\_hierarchy\_geocoded cg\_dw\_us\_org\_unit\_dim  cdl\_mylms\_certifications |  |

151

Clearview

HDL

Tim Hortons Proprietary POS transactional data.

**Ongoing-Feed**

clearview\_ej\_detail clearview\_ej\_dimension clearview\_ej\_header clearview\_ej\_servicetime clearview\_ej\_store clearview\_ej\_tax clearview\_ej\_tender

**One time data parse for historical reason:** clearview\_qst\_csv\_parsed\_data clearview\_qst\_csv\_parsed\_data\_new clearview\_qst\_xlsx\_parsed\_data clearview\_sicom\_xls\_parsed\_data clearview\_sicom\_xls\_parsed\_data\_ne w

Crothal Task data (Data related to paitience move within hospital under contract with Crothal)

crothal\_facilities crothal\_functionalareas crothal\_housecodes crothal\_locations crothal\_shifts crothal\_sitestandards crothal\_taskobjectives crothal\_tasks crothal\_timezones crothal\_units

CDL

Eatclub Dataset – Dataset on containing all eatclub transactions curently being shared to us via the Eatclub team.

eatclub\_cdl\_allergens\_v eatclub\_cdl\_auth\_user\_v eatclub\_cdl\_companies\_v eatclub\_cdl\_delivery\_locations\_v eatclub\_cdl\_items\_allergens\_v eatclub\_cdl\_items\_tags\_v eatclub\_cdl\_items\_v eatclub\_cdl\_locations\_v eatclub\_cdl\_main\_operator\_v eatclub\_cdl\_main\_refundcategory\_v eatclub\_cdl\_main\_region\_v eatclub\_cdl\_menus\_items\_v eatclub\_cdl\_menus\_v eatclub\_cdl\_orders\_items\_v eatclub\_cdl\_orders\_v eatclub\_cdl\_refunds\_v eatclub\_cdl\_restaurants\_v eatclub\_cdl\_tag\_types\_v eatclub\_cdl\_tags\_v eatclub\_cdl\_user\_profiles\_v

EXACT

Payment processing data for Exact. This vendor is no longer being used for any payment processing. Data is store for historical reporting purposes

exact\_transactions

GCP\_FIrebase

Event Related Data that was captured in GCP Firebase. This system is no longer in production and has since been replaced.

gcp\_firebase\_boost\_events

CheckOne

Checkone dataset as it relates to Hamilton Health Science. Checkone provides a cleaning audit system for various Hospital

hamiltonhealthscience\_checkone

Vericlean

Vericlean Dataset as it relates to Hamilton Health Science. Vericlean provides a cleaning audit system for various hospital

hamiltonhealthscience\_vericlean

JDE Canadian ERP system jde\_f060116 jde\_f08001 jde\_f08042

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Know’N’Act |  | Meal survey system used by Marquise, Crothal Canada. This dataset is used for the **Touch the Table** survey in Marquise & Crothal. Product is supported by CDL. | knownact\_feedback\_questions knownact\_feedback |  |

152

Maintenance Connection

Ticket data from Maintanance Connection containing data for both Crothal & SSC

mc\_asset\_documents mc\_asset\_images mc\_asset\_meter\_history mc\_asset\_specifications mc\_assets mc\_classifications mc\_companies mc\_invoices

mc\_labors mc\_lookup\_table\_values mc\_lookup\_tables mc\_part\_documents mc\_part\_locations mc\_part\_vendors  
mc\_parts mc\_purchase\_order\_line\_items mc\_purchase\_orders mc\_receipt\_line\_items mc\_receipts  
mc\_specifications

ssc\_aims\_labor\_st\_ot ssc\_aims\_phasecost ssc\_mystaff ssc\_workorder\_assignments ssc\_workorder\_documents ssc\_workorder\_laboractuals ssc\_workorder\_laborestimates ssc\_workorder\_miscactuals ssc\_workorder\_partactuals ssc\_workorder\_partestimates ssc\_workorder\_statusupdates ssc\_workorder\_tasks ssc\_workorders

Meraki

Compass Units Network related data. Contains data around which device is connected to the compass wifi network at a given unit

meraki\_device\_info meraki\_network\_client\_devices meraki\_network\_clients meraki\_network\_clients\_device\_info meraki\_network\_devices meraki\_networks

Micros

Micros raw POS transactional data

micros\_check\_detail micros\_check\_header micros\_discount\_records micros\_general\_ledger\_identification micros\_menu\_item\_price\_definitions micros\_menu\_item\_sales\_records micros\_order\_type\_records micros\_paid\_in\_out\_detail\_records micros\_pay\_identification micros\_pay\_information micros\_service\_charge\_records micros\_summary\_totals micros\_tax\_records micros\_tender\_records

153

MM Hayes

MM Hayes raw POS transaction data

mmhayes\_qc\_ccauthorizationtype mmhayes\_qc\_donation mmhayes\_qc\_loyaltyreward mmhayes\_qc\_pacombo mmhayes\_qc\_padepartment mmhayes\_qc\_padiscounts mmhayes\_qc\_paitemtaxdsct mmhayes\_qc\_paitemtype mmhayes\_qc\_paordertype mmhayes\_qc\_papaidout mmhayes\_qc\_paplu mmhayes\_qc\_paprepoption mmhayes\_qc\_paprepoptionset mmhayes\_qc\_paprintstatus mmhayes\_qc\_pareceivedacct mmhayes\_qc\_pasubdepartment mmhayes\_qc\_pasurcharge mmhayes\_qc\_patender mmhayes\_qc\_patransactions mmhayes\_qc\_patranslineitem mmhayes\_qc\_patranslineitemmods mmhayes\_qc\_patransstatus mmhayes\_qc\_patranstype mmhayes\_qc\_paymentmethodtype mmhayes\_qc\_quickchargeusers mmhayes\_qc\_revenuecenter mmhayes\_qc\_taxrate mmhayes\_qc\_terminalmodel mmhayes\_qc\_terminals mmhayes\_qc\_terminaltype

MyAdmin

Project Intake system used by CDL to track POS/Kiosk/Mobile install. This has since been replace with Smarthsheet

myadmin\_fbr\_mobile\_requests\_report myadmin\_fbr\_questionnaire myadmin\_fbr\_tech\_detail\_report myadmin\_fbr\_termination

MyDinning

MyDinning application data

mydining\_client\_special\_diet mydining\_client\_texture mydining\_facilitylist  
mydining\_los  
mydining\_menu mydining\_menu\_course mydining\_menu\_item mydining\_menu\_item\_nutrient mydining\_menu\_line mydining\_menu\_schedule mydining\_nutrient mydining\_orders\_export mydining\_orders\_export\_bakup\_24nov 2022

mydining\_patient\_length\_of\_stay mydining\_patient\_satisfaction\_scores mydining\_special\_diet mydining\_special\_diet\_menu\_item mydining\_special\_diet\_nutrient mydining\_texture mydining\_texture\_menu\_item mydining\_therapeutic\_menu mydinning\_client\_special\_diet mydinning\_client\_texture mydinning\_facility\_list mydinning\_menu mydinning\_menu\_course mydinning\_menu\_item mydinning\_menu\_item\_nutrient mydinning\_menu\_line mydinning\_menu\_schedule mydinning\_nutrient mydinning\_special\_diet mydinning\_special\_diet\_menu\_item mydinning\_special\_diet\_nutrient mydinning\_texture mydinning\_texture\_menu\_item mydinning\_therapeutic\_menu

154

Nudge

Nudge vendor data. Dataset related to “Nudge” push from the apps.

nudge\_eurest\_campaign nudge\_eurest\_costcenter nudge\_eurest\_employee\_count nudge\_eurest\_questions nudge\_eurest\_response nudge\_eurest\_sl nudge\_eurest\_user

Nutrislice

K-12 Transaction Sales data from vendor Nutrislice.

Nutrislice is an online ordering platform allowing parent to prepay for their kids meal.

nutrislice\_customers\_dimension\_iceber g  
nutrislice\_customers\_iceberg nutrislice\_foods\_dimension\_iceberg nutrislice\_foods\_iceberg nutrislice\_locations\_dimension\_iceberg nutrislice\_locations\_iceberg nutrislice\_order\_items\_iceberg nutrislice\_orders\_iceberg nutrislice\_transactions\_iceberg

Origami

Risk Audit Data from Vendor Origami

origami\_risk\_audit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Quickbase |  | Various dataset from different Quickbase Projects | quickbase\_healthcare\_ed\_accounts quickbase\_healthcare\_ed\_brands quickbase\_healthcare\_ed\_brands\_in\_s cope quickbase\_healthcare\_ed\_implementat ions  quickbase\_healthcare\_ed\_tasks quickbase\_higher\_ed\_healthcare\_acco unts quickbase\_higher\_ed\_healthcare\_bran ds quickbase\_higher\_ed\_healthcare\_bran ds\_in\_scope quickbase\_higher\_ed\_healthcare\_impl ementations quickbase\_higher\_ed\_healthcare\_task s quickbase\_patient\_nutrition |  |

Remedy  
Play Store & Apple’s app store

Technology deployment platform

Review data for CD mobile apps from Android’s Play Store & Apple’s app store

Event data from Frictionless cafe as capture by Standard Cognition

remedy\_fss\_cdl\_canada

reviews\_appstore reviews\_playstore

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SAP |  | SAP Budget, GL & Hierarchy data as well as HRIS data | sap\_budget sap\_gl\_accounts sap\_gl\_hierarchy sap\_gl\_hierarchy\_collapsed sap\_gl\_master sap\_gl\_total\_balance sap\_hris\_eurest\_eastern\_actives\_repo rt  sap\_hris\_us\_employee\_record sap\_hris\_us\_employee\_record\_old sap\_stat\_all |  |

Standard Cognition Frictionless Events

sc\_frictionless\_carts sc\_frictionless\_events

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Marketing Cloud |  | Backfeed data from Marketing cloud capture users email open rate, click rate for CD mobile application. | marketingcloud\_backfeed\_events marketingcloud\_backfeed\_jobs marketingcloud\_backfeed\_push\_notific ation marketingcloud\_backfeed\_subscribers marketingcloud\_backfeed\_subscribers \_email\_status marketingcloud\_backfeed\_survey\_all marketingcloud\_backfeed\_survey\_com ments |  |

155

Eatclub

Transactional data from Eatclub

eatclub\_cdl\_allergens\_v eatclub\_cdl\_auth\_user\_v eatclub\_cdl\_companies\_v eatclub\_cdl\_delivery\_locations\_v eatclub\_cdl\_items\_allergens\_v eatclub\_cdl\_items\_tags\_v eatclub\_cdl\_items\_v eatclub\_cdl\_locations\_v eatclub\_cdl\_main\_operator\_v eatclub\_cdl\_main\_refundcategory\_v eatclub\_cdl\_main\_region\_v eatclub\_cdl\_menus\_items\_v eatclub\_cdl\_menus\_v eatclub\_cdl\_orders\_items\_v eatclub\_cdl\_orders\_v eatclub\_cdl\_refunds\_v eatclub\_cdl\_restaurants\_v eatclub\_cdl\_tag\_types\_v eatclub\_cdl\_tags\_v eatclub\_cdl\_user\_profiles\_v

Bypass

Bypass POS data

brands\_configs\_delivery\_destinations bypass\_adjustments\_unique bypass\_employees\_unique bypass\_events\_unique bypass\_inventory\_categories\_unique bypass\_items\_unique bypass\_line\_items\_unique bypass\_locations\_unique bypass\_modifiers\_unique bypass\_orders\_unique bypass\_payments\_unique bypass\_reporting\_groups\_unique bypass\_stand\_sheet\_rows\_unique bypass\_stand\_sheets\_unique bypass\_stock\_items\_unique bypass\_terminals\_unique bypass\_tills\_unique bypass\_venues\_unique

Crothal Task data

Data Related to Crothal “Tasks”

crothal\_facilities crothal\_functionalareas crothal\_housecodes crothal\_locations crothal\_shifts crothal\_sitestandards crothal\_taskobjectives crothal\_tasks crothal\_timezones crothal\_units

Silverware

Silverware POS data

silverware\_pos\_checks silverware\_pos\_discounts silverware\_pos\_guests silverware\_pos\_item\_modifiers silverware\_pos\_items silverware\_pos\_orders silverware\_pos\_payments silverware\_pos\_taxes

Smartsheet

Various data pull from Smarthsheet

smartsheet\_cdl\_na\_project\_portfolio\_s ummary smartsheet\_configuration\_tracker smartsheet\_cost\_center smartsheet\_fee\_tracker smartsheet\_hardware\_and\_software\_tr acker smartsheet\_oa\_rls\_intake\_form\_submi ssions smartsheets\_datascience\_report

156

Supply Chain

Supply Chain data

supplychain\_contractperiods supplychain\_customerhierarchy supplychain\_customerhierarchyhistory supplychain\_customerlevelhierarchy supplychain\_customerlevelhierarchyhis tory supplychain\_distributorcontracteditems supplychain\_distributorcustomers supplychain\_distributorhierarchy supplychain\_distributoritembrands supplychain\_distributoritems supplychain\_distributortermsets supplychain\_distributorvashares supplychain\_disttransreporting supplychain\_disttransreporting\_unique \_keys

supplychain\_fiscalperiods supplychain\_listoflists supplychain\_loadperiods supplychain\_manufacturercontractedite ms

supplychain\_manufactureritems supplychain\_manufactureritemshistory supplychain\_manufacturers supplychain\_manufacturertermsets supplychain\_manufacturervashares supplychain\_mfritemattributevalues supplychain\_mfrtransreporting supplychain\_mfrtransreporting\_unique \_keys

supplychain\_nttransactions supplychain\_nttransreporting supplychain\_ntvashares supplychain\_transactions

157

Tacit

Tacit Mobile application transactional data

tacit\_address  
tacit\_allergytype tacit\_availabilityschedule tacit\_availabilityscheduledayofweek tacit\_checkpayment tacit\_checkpaymentcard tacit\_checkpaymentrefund tacit\_clientapp tacit\_clientapprestaurants tacit\_company

tacit\_companybrand tacit\_customer tacit\_customerallergyfacts tacit\_customerdiscounteligibility tacit\_customerdiscounts tacit\_customerpaymentcard tacit\_customerrestaurantfeedback tacit\_customerseating tacit\_customersession tacit\_customersurvey tacit\_deliverypoint tacit\_deliverypointtimeslot tacit\_discounts tacit\_generatedpromocode tacit\_menu

tacit\_menu\_test tacit\_menuitemallergyfacts tacit\_menuitemgroup tacit\_menuitemgroupmembership tacit\_menuitemgroupmenumembership tacit\_menuitemprice tacit\_ordermenuitem tacit\_promoofferdefinition tacit\_restaurantavailabilityschedule tacit\_restaurantdiscountmap tacit\_restaurantmenu tacit\_restaurantmenuavailabilityschedul es

tacit\_restaurantpreferences tacit\_restaurantprofile tacit\_restaurantratingscale tacit\_tablecheck tacit\_tablechecktax tacit\_tableorder tap2eat\_restaurantprofile

158

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Volante (US Morrison, Canada, On Prem Canada Deployment [Trent U] |  | Volante Transactional data for both US, Canada & On Prem Canadian deployment | **[Trent U]**  trentu\_volante\_v6\_canada\_categories trentu\_volante\_v6\_canada\_category\_d etails trentu\_volante\_v6\_canada\_credit\_card \_types trentu\_volante\_v6\_canada\_currency\_ty pes trentu\_volante\_v6\_canada\_division\_de tails trentu\_volante\_v6\_canada\_divisions trentu\_volante\_v6\_canada\_employee\_i nfo trentu\_volante\_v6\_canada\_end\_of\_da y\_close trentu\_volante\_v6\_canada\_group\_deta ils  trentu\_volante\_v6\_canada\_groups trentu\_volante\_v6\_canada\_item\_categ ory\_type trentu\_volante\_v6\_canada\_menu\_item \_details trentu\_volante\_v6\_canada\_menu\_item \_price trentu\_volante\_v6\_canada\_menu\_item \_scancode trentu\_volante\_v6\_canada\_menu\_item s trentu\_volante\_v6\_canada\_payment\_ method\_types trentu\_volante\_v6\_canada\_pos\_audit\_l ogs trentu\_volante\_v6\_canada\_price\_level \_entries trentu\_volante\_v6\_canada\_price\_level s trentu\_volante\_v6\_canada\_price\_mod \_footers trentu\_volante\_v6\_canada\_price\_modif ier\_apply\_types trentu\_volante\_v6\_canada\_price\_modif ier\_identifiers trentu\_volante\_v6\_canada\_price\_modif ier\_tracker trentu\_volante\_v6\_canada\_price\_modif ier\_types trentu\_volante\_v6\_canada\_price\_modif iers trentu\_volante\_v6\_canada\_profit\_cent er\_terminals trentu\_volante\_v6\_canada\_profit\_cent ers trentu\_volante\_v6\_canada\_sale\_types trentu\_volante\_v6\_canada\_stores trentu\_volante\_v6\_canada\_system\_par m trentu\_volante\_v6\_canada\_table\_trans actions trentu\_volante\_v6\_canada\_table\_types trentu\_volante\_v6\_canada\_tax\_type trentu\_volante\_v6\_canada\_terminal\_sy nch\_stats trentu\_volante\_v6\_canada\_terminals trentu\_volante\_v6\_canada\_transaction \_credit\_card\_type trentu\_volante\_v6\_canada\_transaction \_item\_price\_mods trentu\_volante\_v6\_canada\_transaction \_item\_voids trentu\_volante\_v6\_canada\_transaction \_items trentu\_volante\_v6\_canada\_transaction \_payment\_cash trentu\_volante\_v6\_canada\_transaction \_payment\_methods trentu\_volante\_v6\_canada\_transaction \_payment\_paid\_in\_out trentu\_volante\_v6\_canada\_transaction |  |  |

159

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | s trentu\_volante\_v6\_canada\_void\_reaso ns  **[Volante V6 (Canadian Legacy Deployment)]** volante\_v6\_canada\_categories volante\_v6\_canada\_category\_details volante\_v6\_canada\_credit\_card\_types volante\_v6\_canada\_currency\_types volante\_v6\_canada\_division\_details volante\_v6\_canada\_divisions volante\_v6\_canada\_employee\_info volante\_v6\_canada\_employee\_shifts volante\_v6\_canada\_end\_of\_day\_close volante\_v6\_canada\_group\_details volante\_v6\_canada\_groups volante\_v6\_canada\_item\_category\_typ e volante\_v6\_canada\_menu\_item\_detail s volante\_v6\_canada\_menu\_item\_price volante\_v6\_canada\_menu\_item\_scanc ode  volante\_v6\_canada\_menu\_items volante\_v6\_canada\_payment\_method\_ types volante\_v6\_canada\_pos\_audit\_logs volante\_v6\_canada\_price\_level\_entrie s volante\_v6\_canada\_price\_levels volante\_v6\_canada\_price\_mod\_footers volante\_v6\_canada\_price\_modifier\_ap ply\_types volante\_v6\_canada\_price\_modifier\_ide ntifiers volante\_v6\_canada\_price\_modifier\_tra cker volante\_v6\_canada\_price\_modifier\_typ es volante\_v6\_canada\_price\_modifiers volante\_v6\_canada\_profit\_center\_term inals volante\_v6\_canada\_profit\_centers volante\_v6\_canada\_sale\_types volante\_v6\_canada\_stores volante\_v6\_canada\_system\_parm volante\_v6\_canada\_table\_transactions volante\_v6\_canada\_table\_transactions \_bkp volante\_v6\_canada\_table\_types volante\_v6\_canada\_tax\_type volante\_v6\_canada\_terminal\_synch\_st ats volante\_v6\_canada\_terminals volante\_v6\_canada\_transaction\_credit \_card\_type volante\_v6\_canada\_transaction\_credit \_card\_type\_bkp volante\_v6\_canada\_transaction\_item\_ price\_mods volante\_v6\_canada\_transaction\_item\_ price\_mods\_bkp volante\_v6\_canada\_transaction\_item\_ voids volante\_v6\_canada\_transaction\_items volante\_v6\_canada\_transaction\_items \_bkp volante\_v6\_canada\_transaction\_paym ent\_cash volante\_v6\_canada\_transaction\_paym ent\_cash\_bkp volante\_v6\_canada\_transaction\_paym ent\_methods volante\_v6\_canada\_transaction\_paym ent\_methods\_bkp volante\_v6\_canada\_transaction\_paym ent\_paid\_in\_out volante\_v6\_canada\_transaction\_paym |  |  |

160

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | ent\_paid\_in\_out\_bkp volante\_v6\_canada\_transactions volante\_v6\_canada\_transactions\_bkp volante\_v6\_canada\_void\_reasons  **[Volante V7 Canadian Cloud Deployment)]**  volante\_canada\_v7\_computers volante\_v7\_canada\_categories volante\_v7\_canada\_category\_details volante\_v7\_canada\_computers volante\_v7\_canada\_credit\_card\_types volante\_v7\_canada\_currency\_types volante\_v7\_canada\_division\_details volante\_v7\_canada\_divisions volante\_v7\_canada\_employee\_info volante\_v7\_canada\_employee\_shifts volante\_v7\_canada\_end\_of\_day\_close volante\_v7\_canada\_group\_details volante\_v7\_canada\_groups volante\_v7\_canada\_item\_category\_typ e volante\_v7\_canada\_menu\_item\_detail s volante\_v7\_canada\_menu\_item\_price volante\_v7\_canada\_menu\_item\_scanc ode  volante\_v7\_canada\_menu\_items volante\_v7\_canada\_mi\_grouping volante\_v7\_canada\_mi\_item volante\_v7\_canada\_mi\_items\_groupin gs volante\_v7\_canada\_mi\_permission\_co nfig  volante\_v7\_canada\_mi\_price volante\_v7\_canada\_mi\_scancode volante\_v7\_canada\_payment\_method\_ types volante\_v7\_canada\_pos\_audit\_logs volante\_v7\_canada\_price\_level\_entrie s  volante\_v7\_canada\_price\_levels volante\_v7\_canada\_price\_mod\_footers volante\_v7\_canada\_price\_modifier\_ap ply\_types volante\_v7\_canada\_price\_modifier\_ide ntifiers volante\_v7\_canada\_price\_modifier\_tra cker volante\_v7\_canada\_price\_modifier\_typ es  volante\_v7\_canada\_price\_modifiers volante\_v7\_canada\_profit\_center\_term inals volante\_v7\_canada\_profit\_centers volante\_v7\_canada\_sale\_types volante\_v7\_canada\_stores volante\_v7\_canada\_system\_parm volante\_v7\_canada\_table\_transactions volante\_v7\_canada\_table\_types volante\_v7\_canada\_tax\_type volante\_v7\_canada\_terminal\_kiosk\_co nfig volante\_v7\_canada\_terminal\_synch\_st ats  volante\_v7\_canada\_terminals volante\_v7\_canada\_tl\_price\_levels volante\_v7\_canada\_transaction\_credit \_card\_type volante\_v7\_canada\_transaction\_item\_ price\_mods volante\_v7\_canada\_transaction\_item\_ voids volante\_v7\_canada\_transaction\_items volante\_v7\_canada\_transaction\_paym ent\_cash volante\_v7\_canada\_transaction\_paym ent\_methods volante\_v7\_canada\_transaction\_paym |  |  |

161

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | ent\_paid\_in\_out volante\_v7\_canada\_transactions volante\_v7\_canada\_void\_reasons  **[Volante V7 US Chartwells Deployment - Not in used]** volante\_v7\_us\_chartwells\_categories volante\_v7\_us\_chartwells\_category\_de tails volante\_v7\_us\_chartwells\_credit\_card\_ types volante\_v7\_us\_chartwells\_currency\_ty pes volante\_v7\_us\_chartwells\_division\_det ails volante\_v7\_us\_chartwells\_divisions volante\_v7\_us\_chartwells\_employee\_i nfo volante\_v7\_us\_chartwells\_employee\_s hifts volante\_v7\_us\_chartwells\_end\_of\_day \_close volante\_v7\_us\_chartwells\_group\_detail s  volante\_v7\_us\_chartwells\_groups volante\_v7\_us\_chartwells\_item\_catego ry\_type volante\_v7\_us\_chartwells\_menu\_item\_ details volante\_v7\_us\_chartwells\_menu\_item\_ price volante\_v7\_us\_chartwells\_menu\_item\_ scancode volante\_v7\_us\_chartwells\_menu\_items volante\_v7\_us\_chartwells\_mi\_grouping volante\_v7\_us\_chartwells\_mi\_item volante\_v7\_us\_chartwells\_mi\_items\_gr oupings volante\_v7\_us\_chartwells\_mi\_permissi on\_config volante\_v7\_us\_chartwells\_mi\_price volante\_v7\_us\_chartwells\_mi\_scancod e volante\_v7\_us\_chartwells\_payment\_m ethod\_types volante\_v7\_us\_chartwells\_pos\_audit\_l ogs volante\_v7\_us\_chartwells\_price\_level\_ entries volante\_v7\_us\_chartwells\_price\_levels volante\_v7\_us\_chartwells\_price\_mod\_f ooters volante\_v7\_us\_chartwells\_price\_modifi er\_apply\_types volante\_v7\_us\_chartwells\_price\_modifi er\_identifiers volante\_v7\_us\_chartwells\_price\_modifi er\_tracker volante\_v7\_us\_chartwells\_price\_modifi er\_types volante\_v7\_us\_chartwells\_price\_modifi ers volante\_v7\_us\_chartwells\_profit\_cente r\_terminals volante\_v7\_us\_chartwells\_profit\_cente rs volante\_v7\_us\_chartwells\_sale\_types volante\_v7\_us\_chartwells\_stores volante\_v7\_us\_chartwells\_system\_par m volante\_v7\_us\_chartwells\_table\_transa ctions volante\_v7\_us\_chartwells\_table\_types volante\_v7\_us\_chartwells\_tax\_type volante\_v7\_us\_chartwells\_terminal\_kio sk\_config volante\_v7\_us\_chartwells\_terminal\_sy nch\_stats volante\_v7\_us\_chartwells\_terminals |  |  |

162

volante\_v7\_us\_chartwells\_tl\_price\_lev els volante\_v7\_us\_chartwells\_transaction\_ credit\_card\_type volante\_v7\_us\_chartwells\_transaction\_ item\_price\_mods volante\_v7\_us\_chartwells\_transaction\_ item\_voids volante\_v7\_us\_chartwells\_transaction\_ items volante\_v7\_us\_chartwells\_transaction\_ payment\_cash volante\_v7\_us\_chartwells\_transaction\_ payment\_methods volante\_v7\_us\_chartwells\_transaction\_ payment\_paid\_in\_out volante\_v7\_us\_chartwells\_transactions volante\_v7\_us\_chartwells\_void\_reason s

**[Volante V7 Cloud Eurest Deployment - Used for mobile deployment])** volante\_v7\_us\_eurest\_categories volante\_v7\_us\_eurest\_category\_detail s volante\_v7\_us\_eurest\_credit\_card\_typ es volante\_v7\_us\_eurest\_currency\_types volante\_v7\_us\_eurest\_division\_details

163

Development Utilities

Below are the commonly used utilities, created by CDL Data Technology developers. These utilities can be, Operators that is used to design Airflow DAG, otherwise it can be generic python classes/methods that can be readily imported as modules with in the script and which can be used to process data, develop ETL logics, establish connections with various servers, or any other commonly used features implemented for other developers to readily use with in their python script.

Operator to submit jobs to EKS Cluster  
Operator to submit job to virtual EMR runs on EKS cluster Operator to execute SSH command  
Operator to submit job to EMR cluster

164

Operator to submit jobs to EKS Cluster

This is a custom operator used to submit jobs to Elastic Kubernetes Service (EKS) Cluster, This operator is designed to submit jobs to configured Kubernetes cluster configured in the Data Engineering Job Processing Server.

Process

The operator takes in the parameters(as displayed below), generates the Kubernetes POD YAML file and submits the POD config for execution.

It also monitors the POD execution status and when the execution finishes and writes the logs into Airflow Log.  
It then archives the log into S3 location s3://eks-dataops-fargate-main-0-dataops-emr-containers-logging/EKS-DATAOPS-

FARGATE-MAIN-0-KUBE-EXECUTIONS/ .  
Job Processing Server kube\_user@ec2-3-21-60-221.us-east-2.compute.amazonaws.com

EKS Instance https://us-east-1.console.aws.amazon.com/eks/home?region=us-east-1#/clusters/dataops-fargate-main-0?selectedTab=cluster-configuration-tab

**The Operator import path:**

1 from cdl\_library.other\_common\_utils.KubePodSubmit import KubePodSubmit **The Operator Parameters:**

1 AWS\_SECRET (str): AWS Secret name used to access job processing server 2 POD\_NAME (str): The name of the POD to be created  
3 IMAGE\_URL (str): The ECR Image URI  
4 MEMORY (list): [Min Memory, Max Memory]

5 CPU (list): [Min CPU, Max CPU]  
6 ENTRY\_POINT\_ARGS (list, optional): Array of Arguments. Defaults to [].  
7 ENV (dict, optional): dictionary of environment variables. Defaults to {}. 8 RESTART\_POLICY (str, optional): as the name suggests. Defaults to "Never". 9 NAMESPACE (str, optional): as the name suggests. Defaults to "default".

10 AWS\_REGION\_NAME (str, optional): as the name suggests. Defaults to "us-east-1".  
11 S3\_DIR (\_type\_, optional): The S3 location used to share the data to job processing server. Defaults to "s3://c

**Sample code snippet:**

d

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

extract\_opr = KubePodSubmit(

task\_id="REFRESH\_S3\_SOURCE\_FILES\_LIST",

AWS\_SECRET="DATAOPS\_KUBE\_EC2\_SSH",

POD\_NAME="SMARTSHEETS\_DATASCIENCE\_REPORT",

IMAGE\_URL="736888855894.dkr.ecr.us-east-1.amazonaws.com/dataops-endeavor-repo:v0.001",

MEMORY=['2G'],

CPU=['500m'],

ENTRY\_POINT\_ARGS=["smartsheet/parse\_s3\_email.py"],

ENV={

"secrets": {

"AWS\_ACCESS\_KEY\_ID": ["endeavor-credentials", "aws-key"],

"AWS\_SECRET\_ACCESS\_KEY": ["endeavor-credentials", "aws-secret"],

"AWS\_DEFAULT\_REGION": ["endeavor-credentials", "aws-region"]

},

"others": {

"TO\_LIST": "hari.theniah@compassdigital.io",

165

17

18

19

20

21

22

23

"CC\_LIST": "None",

"BCC\_LIST": "None",

"SLACK\_CHANNELS": "cdl-bi-endeavor-job-alerts"

} },

dag=dag, )

166

Operator to submit job to virtual EMR runs on EKS cluster

This is a custom operator used to submit jobs to the virtual EMR container on EKS Cluster, This operator is designed to submit jobs to the EMR container configured in the EKS Cluster.

Job Processing Server EKS Instance  
EMR Cluster

**The Operator import path:**

kube\_user@ec2-3-21-60-221.us-east-2.compute.amazonaws.com https://us-east-1.console.aws.amazon.com/eks/home?region=us-east-1#/clusters/dataops-fargate-main-0?selectedTab=cluster-configuration-tab https://us-east-1.console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#virtual-cluster-jobs:ts4wa4icfnbs1872yy0bgjc97

1 from cdl\_library.other\_common\_utils.EMRContainerJobSubmitOperator import EMRContainerJobSubmitOperator **The Operator Parameters:**

1. 1  Args:
2. 2  AWS\_SECRET (str): AWS Secret Name, SSH Connection Credentials
3. 3  ENTRY\_POINT (str): Script Path
4. 4  CLUSTERID (str, optional): EMR EKS Virtual Cluster ID. Defaults to "ts4wa4icfnbs1872yy0bgjc97".
5. 5  AWS\_EMR\_JOB\_EXECUTION\_ROLE\_ARN (\_type\_, optional): AWS EMR Execution Role ARN. Defaults to "arn:aws:iam::73
6. 6  RELEASE\_LABEL (str, optional): EMR Release Label. Defaults to "emr-6.5.0-latest".
7. 7  ICEBERG\_CATALOG\_S3\_URI (\_type\_, optional): ICEBERG CATALOG S3 URL. Defaults to "s3://cdl-dataops-datalake-d
8. 8  ICEBERG\_CATALOG\_REFERENCE (str, optional): ICEBERG Catalog Reference to use inside the PySpark Script. Defa
9. 9  EXECUTOR\_INSTANCES (int, optional): Total Executors required for Execution. Defaults to 2.
10. 10  EXECUTOR\_MEMORY (int, optional): Total Memory required for execution. Defaults to 2.
11. 11  EXECUTOR\_CORES (int, optional): Total CPU Cores required for Execution. Defaults to 2.
12. 12  DRIVER\_CORES (int, optional): Total Driver Cores Required for execution. Defaults to 2.
13. 13  CUSTOMR\_IMAGE\_TAG (\_type\_, optional): Custom ECR Image URL. Defaults to "736888855894.dkr.ecr.us-east-1.ama
14. 14  S3\_LOG\_URI (\_type\_, optional): S3 Log Location. Defaults to "s3://eks-dataops-fargate-main-0-dataops-emr-co
15. 15  LOG\_GROUP\_NAME (str, optional): Cloud Watch LOG Group Name. Defaults to "eks-dataops-emr-containers-logs".
16. 16  LOG\_STREAM\_NAME\_PREFIX (str, optional): Cloud Watch LOG Stream Name. Defaults to "eks-dataops-fargate-main-
17. 17  ENTRY\_POINT\_ARGUMENTS (list, optional): Arguments to be passed to PySpark Script. Defaults to [].
18. 18  SPARK\_SUBMIT\_PARAMETERS\_XTRA (list, optional): Spark runtime Parameters. Defaults to [].
19. 19  AWS\_REGION\_NAME (str, optional): AWs Region. Defaults to "us-east-1".
20. 20  PRINT\_EMR\_JOB\_STATE\_INTERVAL (int, optional): Job Status to be checked interval. Defaults to 10.
21. 21  Raises:
22. 22  Exception: AWS\_SECRET is a Mandatory Argument
23. 23  Exception: ENTRY\_POINT is a Mandatory Argument

**Sample code snippet:**

6

e u

z n

0

1. 1  opr0 = EMRContainerJobSubmitOperator(
2. 2  task\_id="EMR\_REFRESH\_S",
3. 3  AWS\_SECRET="DATAOPS\_KUBE\_EC2\_SSH",
4. 4  CUSTOMR\_IMAGE\_TAG = '736888855894.dkr.ecr.us-east-1.amazonaws.com/terminus-spaceships-emr:base',
5. 5  ENTRY\_POINT="s3://cdl-dataops-spark-scripts/dev/endeavor/mm-hayes/refresh\_iceberg\_table.py",
6. 6  ENTRY\_POINT\_ARGUMENTS=["mmhayes","s3://cdl-dataops-iceberg-datalake/prod/","s3://cdl-dataops-datalake-pipel
7. 7  SPARK\_SUBMIT\_PARAMETERS\_XTRA=["--packages org.apache.iceberg:iceberg-spark3-runtime:0.13.0,software.amazon.
8. 8  AWS\_REGION\_NAME='us-east-1',
9. 9  dag=dag,
10. 10  )

i a

167

Operator to execute SSH command

This is a custom operator used to execute SSH commands in the remote host (SSH is a secure socket shell, used to communicate with remote Unix/Linux-based server).

**The Operator import path:**

1 from cdl\_library.other\_common\_utils.SSHCommandExecutor import SSHCommandExecutor **The Operator Parameters:**

**Sample code snippet:**

1 2 3 4 5

Args:

AWS\_SECRET (str): AWS Secret with the SSH host configuration

SCRIPT\_PATH (str): Remote machine's Command/Executable Script path

COMMAND (str, optional): Command to script the Script Path. Defaults to None.

AWS\_REGION\_NAME (str, optional): \_description\_. Defaults to "us-east-1".

1 opr0 = SSHCommandExecutor(

2  
3  
4  
5  
6  
7 8)

task\_id="SFTP\_STORAGE\_REPORT",

AWS\_SECRET="ENDEAVOR\_SFTP\_SERVER\_SSH",

COMMAND="python3",

SCRIPT\_PATH="maintenance\_scripts/generate\_storage\_utilization\_report.py",

AWS\_REGION\_NAME='us-east-2',

dag=dag,

168

Operator to submit job to EMR cluster

This is a custom operator used to submit jobs directly to the EMR Cluster. The Spark jobs running on the EMR container are configured to run on the EKS cluster and get a timeout when the job execution runs more than 40-45mins. Such long-running Spark jobs can be scheduled to run directly on the dedicated EMR cluster. Hence we need an Operator to submit jobs to the EMR cluster.

EMR Cluster https://us-east-1.console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details:j-3REBHG8B0K7IP

**The Operator import path:**

1 from cdl\_library.other\_common\_utils.EMRJobSubmitOperator import EMRJobSubmitOperator **The Operator Parameters:**

1. 1  Args:
2. 2  AWS\_SECRET (str): \_description\_
3. 3  ENTRY\_POINT (str): \_description\_
4. 4  ICEBERG\_CATALOG\_S3\_URI (\_type\_, optional): \_description\_. Defaults to "s3://cdl-dataops-datalake-dev-us-eas
5. 5  ICEBERG\_CATALOG\_REFERENCE (str, optional): \_description\_. Defaults to "iceberg\_catalog".
6. 6  EXECUTOR\_INSTANCES (int, optional): \_description\_. Defaults to 2.
7. 7  EXECUTOR\_MEMORY (int, optional): \_description\_. Defaults to 2.
8. 8  EXECUTOR\_CORES (int, optional): \_description\_. Defaults to 2.
9. 9  DRIVER\_CORES (int, optional): \_description\_. Defaults to 2.
10. 10  ENTRY\_POINT\_ARGUMENTS (list, optional): \_description\_. Defaults to [].
11. 11  SPARK\_SUBMIT\_PARAMETERS\_XTRA (list, optional): \_description\_. Defaults to [].
12. 12  AWS\_REGION\_NAME (str, optional): \_description\_. Defaults to "us-east-1".
13. 13  S3\_DIR (\_type\_, optional): \_description\_. Defaults to "s3://cdl-dataops-datalake-pipeline-storage/supportin
14. 14  Raises:
15. 15  Exception: AWS\_SECRET is a Mandatory Argument
16. 16  Exception: ENTRY\_POINT is a Mandatory Argument

**Sample code snippet:**

t

g

1. 1  opr0 = EMRJobSubmitOperator(
2. 2  task\_id="EMR\_REFRESH\_ICEBERG\_TABLE\_QC\_CCAUTHORIZATIONTYPE",
3. 3  AWS\_SECRET="DATAOPS\_EMR\_DEV\_CLUSTER\_SSH",
4. 4  ICEBERG\_CATALOG\_S3\_URI= "s3://cdl-dataops-iceberg-datalake/prod/",
5. 5  ENTRY\_POINT="s3://cdl-dataops-spark-scripts/dev/endeavor/mm-hayes/refresh\_iceberg\_table.py",
6. 6  ENTRY\_POINT\_ARGUMENTS=["mmhayes","s3://cdl-dataops-iceberg-datalake/prod/","s3://cdl-dataops-datalake-pipel
7. 7  SPARK\_SUBMIT\_PARAMETERS\_XTRA=["--packages org.apache.iceberg:iceberg-spark3-runtime:0.13.0,software.amazon.
8. 8  AWS\_REGION\_NAME='us-east-1',
9. 9  dag=dag,
10. 10  )

i a

169

Operator to Create Kube Secret

This is a custom operator used to submit Create Secret request to Elastic Kubernetes Service (EKS) Cluster, This operator is designed to submit requests to configured Kubernetes cluster configured in the Data Engineering Job Processing Server.

Process

The operator takes in the parameters(as displayed below), generates the Kubernetes Secret YAML file and submits the Secret config for execution.

Job Processing Server kube\_user@ec2-3-21-60-221.us-east-2.compute.amazonaws.com  
EKS Instance https://us-east-1.console.aws.amazon.com/eks/home?region=us-east-1#/clusters/dataops-fargate-main-0?selectedTab=cluster-configuration-tab

**The Operator import path:**

1 from cdl\_library.other\_common\_utils.KubeCreateSecret import KubeCreateSecret **The Operator Parameters:**

**Sample code snippet:**

1 AWS\_SECRET (str): AWS Secret of Job Processing Machine  
2 AWS\_SECRET\_TO\_COPY (str): AWS Secret to be copied to KUBE Secret  
3 NAMESPACE (str, optional): \_description\_. Defaults to "default".  
4 AWS\_REGION\_NAME (str, optional): \_description\_. Defaults to "us-east-1".

opr0 = KubeCreateSecret(

task\_id="REFORMAT\_S3\_SOURCE\_FILES",

AWS\_SECRET="DATAOPS\_KUBE\_EC2\_SSH",

AWS\_SECRET\_TO\_COPY="dataops\_service\_email",

dag=dag

1  
2  
3  
4  
5 6)

170

DB Connector

We currently support two data warehouses between Amazon Redshift and Snowflake (where Snowflake will eventually replace Redshift entirely).

To make connecting to either of these databases easier to execute queries, there is a utility class called Connector available in the cdl\_library that streamlines the connection.

The connector class acts as a client interface between you and either Redshift or Snowflake. To create a client run the following:

Passing in “redshift” or “snowflake” creates an instance of the class appropriated with the relevant configurations to connect with the respective databases.

The client can perform 3 functions at the time of writing: 1. get\_connection()

Yields a connection context manager (with a transaction established) which can be used with a block so as to not worry about closing the connection once you’ve finished using it.  
The connection is commonly leveraged by running the execute(sql) function which will execute some SQL string against the database, returning a CursorResult object.

ex:

1.

run\_query(query\_string)

As it’s named, this function will try to run your query against the database you instantiated a client for. ex:

1.

load\_df\_to\_redshift(df, table, schema)

Takes a pandas dataframe object and inserts its data into a table in the database.

1 from cdl\_library.utils.db import Connector 2

1. 3  redshift\_client = Connector("redshift")
2. 4  snowflake\_client = Connector("snowflake")

5

1 2 3 4 5

sc = Connector(database="snowflake")

with sc.get\_connection() as conn:

resp = conn.execute("describe table source.p2\_orders")

print(resp.fetchall())

1 cursor\_result = rc.run\_query(  
2 "SELECT distinct schemaname, tablename FROM PG\_TABLE\_DEF WHERE schemaname = 'source'" 3)  
4 print(cursor\_result.fetchall())  
5

171

Volante

172

VPN Configuration  
To start with the VPN configuration follow the steps given below:

1. The first step would be to sign in to the AWS Ec2 instance.  
2. Sign in using username : ec2-user and PEM file s3://cdl-bi-raw-source/vpn-config/CDL-AWS-EC2-DEV.pem

3. Next step is to issue the following commands in sequence. These commands are used to start/restart the VPN service:

sudo su

cd

service ipsec restart service network restart service ipsec status

4. If the VPN is not successfully restarted, please check the following VPN config files in location /etc/ipsec.d root@ip-18.223.216.73 /etc/ipsec.d > l  
total 48  
-rw-r--r-- 1 root root

-rw------- 1 root root -rw------- 1 root root -rw------- 1 root root drwx------ 2 root root -rw-r--r-- 1 root root

92 Sep 13 2018 vpn\_connexions.secrets 423 Sep 13 2018 pkcs11.txt

9216 Sep 13 2018 cert9.db 11264 Sep 13 2018 key4.db

120 Feb 12 2020 policies  
641 Feb 12 2020 vpn\_connexions.conf

5. Please make sure to verify the contents of files vpn\_connexions.secrets and vpn\_connexions.conf , against the S3 location:

s3://cdl-bi-raw-source/vpn-config/ s3://cdl-bi-raw-source/vpn-config/etc/ipsec.d/vpn\_connexions.conf s3://cdl-bi-raw-source/vpn-config/etc/ipsec.d/vpn\_connexions.secrets

**Volante Contact:** Syed Abidi  
**CDL Escalation (if no response from Volante):** Edwin Lam

173

Data Dictionaries

**Schema**

Datastore

Mobile Orders and Times

**Table**

mobile\_orders

**comments**

deliverystarttime always equals pickuptime

sometimes there is no deliverytime :(

**Dictionary**

Data Dictionary datastore.mobile\_orders - Google Sheets

**deliveryordert ype**

delivery pickuptime deliverytime good pickuptime always equals deliverystarttime exists exists

pickup

scan\_and\_go

delivery

frictionless

scan\_and\_go

**pickuptimech eck**

pickuptime exists

no pickup time

pickuptime exists

pickuptime exists

pickuptime exists

**deliverytimech eck**

deliverytime exists

no delivery time

no delivery time

no delivery time

no delivery time

**makes sense**

good

good

it is what it is

good

good

generally ordertime is approximately equal to pickuptime, however there are some thousands of records where the pickuptime is before the ordertime, not typically by a lot, but in some instances pickuptime is a few hours before ordertime

for scan\_and\_go, the pickuptime, if it exists, is always equal to or just before (up to 24 seconds in one example) the ordertime

pickup no pickup time no delivery time it is what it sometimes there is no pickuptime :( is

null pickuptime no delivery time good pickuptime always equals ordertime exists

dinein pickuptime no delivery time good pickuptime always equals ordertime exists

pickup pickuptime no delivery time good sometimes the pickuptime is before the ordertime exists

takeout pickuptime no delivery time good pickuptime always equals ordertime exists

174

Job Configurations

cdl-dataops-config

The purpose of this repo is to host job-related configs and parameters (path, table name, data format, etc.) and schema files. Two GitHub actions were created to:

1. Sync the repo to the s3 bucket <<bucket name>>  
2. Write the configs to dynamodb table <<table name>>

A utility class was added to cdl\_library to provide functions that can read configs & schema info from the dynamodb table Class name: JobConfig

Functions:  
Pull all dataset names for a given datasource Pull config info for a given dataset  
Pull schema file for a given dataset

Git repo:

https://github.com/compassdigital/cdl-dataops-config - Connect your Github account

175

Scout Documentation

Purpose

This documentation aims to make you understand the processes of Scout Dashboard. This documentation includes an explanation of each component and functionality of the dashboard. We have also added screenshots and references to give more context to the explanation. We have divided the documentation in a manner that you understand the process of the project. We have started with the purpose of the project and have further explained the working of the project.

Tools & Applications:

Power BI

Servers (Digital Ocean) Production and Testing  
Scout Database (MYSQL) on AWS.  
S3 bucket and permission to run Airflow jobs and Lambda jobs. Access to Jira  
Access to Confluence

In case of any access-related issues, contact @Marcia .

176

Scout Dashboard

Scout Dashboard is a market intelligence data product. The dashboard consists of competitor pricing around Compass units or the zip codes that have Compass units. In short, the dashboard helps the Pricing Managers from the Strategic Pricing Team of Compass US understand the inflation and competitor pricing of products in the market through this dashboard. The pricing information is collected from various vendor websites and then used in Scout to compare it with the in-house product pricing. The analysis of pricing is done each month by each unit. On average, each Pricing Manager analyzes 5 units every week.

For example, let’s assume that the Pricing Manager is analyzing the pricing of Yogurt with various competitors. In this scenario, they would either select the Zip Code or the Unit from the Filter Selections to analyze the data. As shown in the highlighted box from the Competitor Price Points, the Dashboard displays the pricing of Yogurt from various brand sites.

Scout is currently available only in the United States of America. It only caters to the B&I (Business & Industry) sectors.

Scout Dashboard

177

Extract, Transform & Load Process Before You Begin

Make sure you have all the required tools and applications required for this process. Refer to the Scout Documentation to get more information.

As explained in the previous document, Scout Dashboard is a market intelligence data product. To make the dashboard intelligence several steps and tons of data collected from various sources are mapped in the dashboard. The extraction of data is done in a 3 step process: Extract-Transfer-Load. The Extract Process is further spilt into Scraping & Parsing. Whereas the Transform and Load Process is straightforward. Click on the tree below to understand the process in detail.

178

Extract Process

Step 1 - Extract Data

*This process is split into two major components, Scraping & Parsing.*

**Preparations before you Scrape:**

Note: The scraping preparation method takes place in a web browser (Google Chrome or Firefox).

1. Press F12 to enter the Developer Tools Interface on the browser. 2. Select Network → Fetch/XHR (If we are scraping JSON files)

179

3. Check through all files from the right side under Fetch/XHR, and find the JSON file that contains the information we need

4. Get the request URL (API call) and regenerate it with Regex. In the example below, the Regex:

Sometimes the website doesn’t load all back-end data into JSON files. In such a case, we scrape the HTML page source.

5. Select Elements  
6. Right-click on the information (e.g. item name) and select Inspect

7. Get the tag name from the Developer Tools Interface window

**1.1 Scraping**

Scraping is a technique where a computer program extracts data from human-readable output coming from another program. In this case, the Scraped page sources can be in HTML/JSON format (sometimes could be both, but it happens rarely). The key is to accurately locate where the website stores the required information. During the scraping process the following information is collected from the competitor's websites:

180

Item name, item price, customization item name, customization item price, item calories, and item description. For example, as shown in the screenshot below, we will scrape all the required information from the Chick-Fil-A website.

Further information like Store detailed address, zip code, city, state, latitude, and longitude is collected. For example, the screenshot below for Five Guys displays how we collect this set of information:

**1.2 Scrape Process  
Clone Wezel from Github and create a virtual environment by following the steps given below:**

a. Open terminal and install virtualenv with the command pip install virtualenv .

b. Enter the Wezel directory with the command .  
c. Create a virtual environment with the command .

Inside the Wezel project folder, there is a requiments.txt . This file holds all the project decencies make sure this file is inside of the Wezel folder.

cd ~/wezel

python3 -m venv env

181

Keep the name as env

d. Activate the virtual environment  
e. Install all the dependencies with the command pip install -r requirements.txt .

**Setting up config files:**

Wezel has configurations files located in ~/wezel/config\_file there are two files startup\_config.json and competitor\_config.json. We have displayed below examples and explanation of each file.

startup\_config:

1{

1. 2  "proxy\_path":"./proxies.txt",
2. 3  "competitor\_paths":"./config\_file/competitor\_config.json",
3. 4  "sqlite\_path": "./wezel\_db/db/wezel.db",
4. 5  "wipe\_data\_on\_start\_up":true,
5. 6  "amount\_of\_nodes":1,
6. 7  "runs\_before\_kill":1,

8} 9

Click here to expand...

proxy\_path : Location path of the proxy IP address used.  
competitor\_path : Location path of the competitor\_config file.  
sqlite\_path : Location path of the SQLite database Wezel uses to track competitors. wipe\_data\_on\_start\_up : This is a boolean trigger that if set to true wipes the Wezel database. amount\_of\_nodes : The amount of process started to scrape a competitor. runs\_before\_kill : Number of times a node uses a proxy before rotating to a new proxy.

If amount\_of\_nodes is set to 3 that means that Wezel will scrape 3 URLs of each active competitor at the same time.

competitor\_config

source env/bin/activate.

The code snippet displayed below is an example of a competitor in the config file, the file is a JSON object that holds all the companies:

1 "company\_name": 2{

1. 3  "name":"company name",
2. 4  "puppeteer\_scrape\_path":"scrapes/js\_scraps/company\_name/company\_product\_scraper.js",
3. 5  "sitemap\_path":"scrapes/sitemap/company/company\_sitemap",
4. 6  "status":"ACTIVE",
5. 7  "save\_path":"scrape\_data/company"

8}

Explanation of each component:

puppeteer\_scrape\_path : Location path of the javascript scraper for that company. sitemap\_path : Location path of the sitemap of that company.

status : If the status is set to ACTIVE that means when Wezel starts it will be scraping that company. save\_path : Location path of where the scraped data get saved.

**Running Wezel**

182

Make sure the virtual environment is active and type in terminal

**Tmux setup**

*Tmux is used to run Wezel in the background of the server and ensures that it continues running even with server and client disconnection or internet blackouts on the client-side.*

Currently, Wezel is set to run on a Digital Ocean server using Ubuntu 16.04.6 LTS

To install tmux enter the command sudo apt install tmux into the terminal.  
Create a tmux session with the command tmux new-session -s name\_of\_session\_here .

When a new tmux session is created ensure that the virtual environment is active in that session.

Active tmux session

When tmux is active start wezel with python ~/wezel/start\_up.py .  
To kill the tmux session open a new terminal and tmux kill-session -t name\_of\_session\_here .

**1.3 Parsing**

Parsing, syntax analysis, or syntactic analysis is the process of analyzing a string of symbols, either in natural language, computer languages, or data structures. Typically scraped page sources are stored by competitor name on the Digital Ocean production server ( 142.93.157.14 ).

For example, Chipotle’s file path is: '/home/adam/scrape\_output/latest\_scrape/product\_output/chipotle\_product\_pass/ '.  
Before you write a parser, you will need to define if the page source is in JSON/HTML format and import Python packages accordingly

(json/BeautifulSoup).  
Generate parsers with Python multiprocessing to extract the data from page sources.  
Standardize parsed data (Add competitor key, timestamp).  
Name the result data as “competitorname\_staging.csv“ and upload it onto the S3 bucket. For example, s3://cdl-

scout/Pipline\_Staging\_Table/ChickFilA/chickfila\_staging.csv .

python ~/wezel/start\_up.py

183

Transform & Load Process Step 2 - Transform and Load Data

*This process is to further standardize the data and match scraped restaurants locations with Compass units. We work with the Pricing Managers for this step.*

**Map Data**

Subset the result data from the previous step to only includes unique menu item names & menu item prices by each competitor. Send this data to the Pricing Team. The team will email back an updated market basket conversion file based on the data. (File location: s3://cdl-

scout/Market\_Basket\_Conversion.csv ). **QA Data**

*All code in this step can be found in Scout GitHub repo* https://github.com/compassdigital/compassdigital.ScOUT - Connect your Github account

With the new market basket conversion file, we can now merge it with the competitor stating table from the Parsing step. Overwrite the QA table in the database. Table name: PBI\_Scout\_QA\_Dashboard .  
Refresh Power BI dashboard Scout\_Data\_QA .  
Notify the pricing manager about the update. They will immediately QA the data.

**Match Data & Compass Units**

*All code in this step can be found in Scout GitHub repo* https://github.com/compassdigital/compassdigital.ScOUT - Connect your Github account

Once pricing managers are satisfied with the data, we first need to overwrite the Scout main table in the database. Table name: National\_Competitor\_Prices\_Labelled\_Revised\_2 .

Use the above table to update the below tables:

If there are any outliers, for example, $50 cookie. Or, if there are a significantly great amount of locations/items missing from scraping.

Regional\_Benchmarking\_UCA

184

Unit\_Regional\_Benchmarking\_UCA PBI\_contract\_location\_information PBI\_Scout\_Competitor\_Adress\_Lookup

Refresh Power BI dashboard Scout\_Data\_20220922 .

PBI\_ZIP\_Competitor\_Lookup

185

A Guide for Debug Competitor's Data

This documentation is a checklist for when:

Competitor data is missing from a postal code Competitor data is missing from a unit

**Before starting your process, please fork the GitHub repo compassdigital.ScOUT to your machine. Please NEVER run any of the following processes on Digital Ocean servers.**

A competitor is missing from a postal/unit Sample SmartSheet request:

“Hey! It looks like Panera Bread's data isn't coming through ScOUT right now. Could you look into this when you have a chance? A few zip code examples: 42003 40503 42701”

Steps to follow:  
Q: Do we have any Panera Bread being mapped to this postal code/unit?

To check this, the query you will use is:  
"select \* from Regional\_Benchmarking\_UCA where ZIP in (42003,40503,42701)" for postal "select \* from Unit\_Regional\_Benchmarking\_UCA where UNIT\_KEY in (1009, 1049)" for unit

Then, look for Panera Bread from the output table --

If the output doesn’t have any Panera Bread location, it is highly possible that the location is missing from the scraping stage. If you find any Panera Bread location(s):

Run "select \* from National\_Competitor\_Prices\_Labelled\_Revised\_2" and check if the output contains the Panera Bread location (by matching competitor\_key/competitor\_key\_id).

If you can't find any records, this means the table Regional\_Benchmarking\_UCA and table PBI\_ZIP\_Competitor\_Lookup  
( Unit\_Regional\_Benchmarking\_UCA and PBI\_Unit\_Competitor\_Lookup for unit) are not updated. Please update these two tables by “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py”  
If you find any Panera Bread data from the output, it means table PBI\_Scout\_Raw\_Data is not updated. Please update this table using “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py”

**In the last step, please remember to refresh the PowerBI dashboard Scout\_Data\_20200922.**

A competitor has missing menu items Sample SmartSheet request:

“Hey! Denny's is missing a large # of items that are mapped in the market basket conversion file. Here are a couple of zip code examples and a list of items in case you want to check. Thank you! Items: - Cheeseburger, Zip Codes: 80120”

Steps to follow:  
Q: What is the actual menu item name?

To check this, you will need to look into “QA\_Market\_Basket\_Conversion.csv“ which locates in the S3 bucket “cdl-scout“.

In this Denny’s case, “Build Your Own Burger“, “Build Your Own Burger - 100% Beef Patty - Burger“, and “Classic Burger“is mapped to Cheeseburger.

186

Q: Do we have the item in table National\_Competitor\_Prices\_Labelled\_Revised\_2 ?  
To do this, you can run query: "select \* from National\_Competitor\_Prices\_Labelled\_Revised\_2"

If you find any of the items (“Build Your Own Burger“, “Build Your Own Burger - 100% Beef Patty - Burger“, or “Classic Burger“) in the noted zip code from the SmartSheet request:

It means table PBI\_Scout\_Raw\_Data is not updated. Please update this table using “/compassdigital.ScOUT/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py”

If no data is being found from National\_Competitor\_Prices\_Labelled\_Revised\_2 :  
Please go to the corresponding competitor folder on S3. You should use the file with the suffix “\_staging.csv”. In this case, we are

checking Denny’s data, which means you are looking for file “cdl-scout/Pipline\_Staging\_Table/Dennys/dennys\_staging.csv” Check if the staging table contains any of the listed items:

- If the item appears in the staging table, check if the menu item name in the staging table is the same as the menu item name in the market basket conversion table. Pay extra attention to leading/trailing spaces. Please inform the pricing team (Bernardo Español) about the error.

Once the market basket conversion file is updated, please upload it to the S3 bucket '/cdl-scout'. Run logic Refresh Existing Competitor(s)

- If the staging table does not have any of the listed items, it is highly possible that the data is missing from either the parsing stage or the scraping stage.

Missing from parsing stage: You should find and check the corresponding page source by matching the target\_id, if the item is in the page source, please re-parse the file and re-run the logic Refresh Existing Competitor(s)

If you still can't find any data in the page source, it means the item(s) is missing from the scraping stage. Please make sure you check: (1) Does this Denny’s location really have this item? (2) Does this item included in the sitemap? (3) Make a note and make sure the item is included when we refresh the competitor.

**In the last step, please remember to refresh the PowerBI dashboard Scout\_Data\_20200922.**

187

A Guide for Updated Marketbasket File  
This documentation is a checklist for when there are updates on the market basket conversion file.

**Before starting your process, please fork the GitHub repo compassdigital.ScOUT to your machine. Please NEVER run any of the following processes on Digital Ocean servers.**

Sample Smartsheet request:

“Hey Marcia! I've attached the updated market basket conversion file. Some spelling corrected & all items formatted the same (no more lower-case only). When you have a chance, can you replace the current mapping with this file? Let me know if you have any questions!”

Steps to follow:

Download the file from Smartsheet, the file name is “QA\_Market\_Basket\_Conversion.csv“. Upload the file onto the S3 bucket “/cdl-scout“. The old file will be automatically overwritten.  
Run script “/Scout\_Pipeline/ETL/Scout\_Main\_Update.py”. This will update the Scout main table –

National\_Competitor\_Prices\_Labelled\_Revised\_2

Note: The function does require a list of competitor names. Please check in with the pricing team to which competitor is affected by this update.

Once the above table is successfully updated, please run function Scout\_Production\_Data\_Load() in script “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py“. This step will update Scout PowerBI back-end table PBI\_Scout\_Raw\_Data .

Then, please refresh the PowerBI dashboard Scout\_Data\_20200922. Lastly, please inform the pricing team about this update.

188

189

A Guide for Refresh/Add Competitor Data

This documentation is a checklist for when:

Existing competitor(s) got refreshed New competitor(s) are added

**Before starting your process, please fork the GitHub repo compassdigital.ScOUT to your machine. Please NEVER run any of the following processes on Digital Ocean servers.**

Refresh Existing Competitor(s)

There won’t be any Smartsheet requests, as refresh competitors are usually discussed and agreed upon by both sides during Scout touchpoint calls.

In this documentation, let’s assume we agree to refresh Boston Market and Red Robin’s data.

Steps to follow:

Update the time stamp file’s file name to the current month in the format of yyyymm. The file is located on S3 “/cdl- scout/Temp\_Page\_Source/202205.txt“  
Run Wezel to get the most up-to-date scraped page sources and parsed data.

Run the standardized logic script “/Scout\_Pipeline/ETL/Standardize\_Parsed\_Data.py“. This step will give us standardized Boston Market and Red Robin’s data. Upload them onto the corresponding S3 bucket, in this case, “/cdl- scout/Pipline\_Staging\_Table/BostonMarket/bostonmarket\_staging.csv”, and “/cdl- scout/Pipline\_Staging\_Table/RedRobin/redrobin\_staging.csv”.

Send the output to Bernie/Rachel. They will map the data with the corresponding market baskets.

The pricing team will submit a SmartSheet request once they finished mapping the market basket. Please download the **new** market basket file from Smartsheet and **overwrite the current** one on S3 (“cdl-scout/QA\_Market\_Basket\_Conversion.csv”).

Run function QA\_Dahsboard\_Data\_Load() in script “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py“. This step will update the back-end data for the Scout QA dashboard.

Refresh PowerBI dashboard Scout\_Data\_QA. Inform the pricing team about the update.

After the above step, please allow a couple of days for the pricing team to check the data. They might have random requests/questions, for example, “Boston Market’s add bacon price looks super high, could you please check it?“, or “Looks like Red Robin has missing locations/items, could you check it?“

For missing data requests, please check page sources, parser functions, sitemaps, and scrapers. For pricing outliers:

Check the page source and the competitor’s website first. In most cases, this can be fixed by adding suffixes that indicate sizes.

If there’s no way to fix the issue from the parsing/scraping side, please notify the pricing team ASAP. They will come up with a threshold to eliminate the outliers.

In this case, please add a line(s) to the very end of function assign\_mb() in script “/Scout\_Pipeline/ETL/Scout\_Main\_Update.py” and a line(s) to the very end of function QA\_Dahsboard\_Data\_Load() in script “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py“.

For example, if we agree to remove all add bacon price points for Boston Market if the price is more than 3 dollars, the code will look like this:

QA Dahsboard Data Load() :

190

Q \_ a sboa d\_ ata\_ oad()

assign\_mb() :  
1 for\_bo = for\_bo.drop(for\_bo[(for\_bo["standardized\_competitor\_name"]=="Boston Market") & (for\_bo[

You will need to run function QA\_Dahsboard\_Data\_Load() again to update the back-end database, refresh the PowerBI dashboard Scout\_Data\_QA, and notify the pricing team about the change.  
**NOTE: If you update any part of the parser/scraper, which will results a new parsed output, please upload the new output and run the standardized logic script again.**

Once the pricing team signs off, the next step is updating Scout's main dashboard – Scout\_Data\_20200922. First, run script “/Scout\_Pipeline/ETL/Scout\_Main\_Update.py”. This will update the Scout main table –

National\_Competitor\_Prices\_Labelled\_Revised\_2

Run script “/Scout\_Pipeline/ETL/Unit\_Regional\_Benchmarking.py“. This will update zip and unit matching tables Regional\_Benchmarking\_UCA and Unit\_Regional\_Benchmarking\_UCA

Once the above tables are successfully updated, run script “/Scout\_Pipeline/ETL/PBI\_Data\_Generators.py“. This will update Scout PowerBI back-end tables PBI\_Scout\_Raw\_Data , PBI\_ZIP\_Competitor\_Lookup , PBI\_Unit\_Competitor\_Lookup , and

PBI\_Scout\_Competitor\_Adress\_Lookup .  
Functions to run: Scout\_Production\_Data\_Load() , Zip\_Competitor\_Lookup() , Unit\_Competitor\_Lookup() , and

Competitor\_Address\_Lookup()

In the last step, please refresh dashboard Scout\_Data\_20200922 and notify the pricing team once it’s completed.

Scrape New Competitor(s)

There won’t be any Smartsheet requests, as adding competitors is usually discussed and agreed upon by both sides during Scout touchpoint calls.

In this documentation, let’s assume we agree to add Tropical Smoothie Cafe to Scout.

Steps to follow:  
Most of the steps are the same to Refresh Existing Competitor(s). But there are a couple of extra steps you will need to run for new competitors.  
Create a new competitor folder on S3 under “cdl-scout/Pipeline\_Staging\_Table/“. The folder name should be the formal name of the competitor with all of the first letters capitalized and remove all the spaces. In this case, we will create a new folder called “TropicalSmoothieCafe“.

After this, please follow the steps for refreshing existing competitors.

1 qa\_table = qa\_table.drop(qa\_table[(qa\_table["standardized\_competitor\_name"]=="Tropical Smoothie

191

192

Scout Database Overview  
This documentation is an overview of all Scout tables stored in the database. Please refer to other documentation under

ad Process for what you should do when debugging the tables. **Major Tables:**

Major Tables:

National\_Competitor\_Prices\_Labelled\_Revised\_2 Regional\_Benchmarking\_UCA Unit\_Regional\_Benchmarking\_UCA

PowerBI Tables: PBI\_Nielsen\_Raw

PBI\_Scout\_Competitor\_Adress\_Lookup PBI\_Scout\_QA\_Dashboard PBI\_Scout\_Raw\_Data PBI\_Unit\_Competitor\_Lookup PBI\_ZIP\_Competitor\_Lookup

**National\_Competitor\_Prices\_Labelled\_Revised\_2**

Transform & Lo

This table is the most up-to-date main Scout competitor table which includes detailed information from competitor geo locations to menu prices. This is always the go-to table when you want to search for competitor/item details.  
Table Preview:

193

**Regional\_Benchmarking\_UCA**

This is the to-go table if you want to search for Scout zip-competitor mapping. E.g. Which competitors are included in zip 00501? Which Blimpie location is mapped to zip 00501?  
Table Preview:

**Unit\_Regional\_Benchmarking\_UCA**

This is the to-go table if you want to search for Scout unit-competitor mapping. E.g. Which competitors are mapped to unit 1003? Which IHOP location is mapped to unit 1003?  
Table Preview:

194

**PowerBI Tables: PBI\_Nielsen\_Raw**

This table includes cleaned, standardized Nielsen CPG data for the PowerBI dashboard

Scout\_Data\_20200922, tab “CPG RAW DATA - NATIONAL”, “CPG RAW DATA - REGIONAL”, and “CPG RAW DATA - LOCAL”. Table Preview:

195

**PBI\_Scout\_Competitor\_Adress\_Lookup**

This is a subset table from “National\_Competitor\_Prices\_Labelled\_Revised\_2” which includes unique detailed competitor locations. It is used for PowerBI dashboard Scout\_Data\_20200922, tab “SCOUT RAW DATA“.  
Table Preview:

196

**PBI\_Scout\_QA\_Dashboard**

This table is generated directly from parsed competitor data. It is used for PowerBI Dashboard Scout\_Data\_QA Table Preview:

197

**PBI\_Scout\_Raw\_Data**

This table is a subset from “National\_Competitor\_Prices\_Labelled\_Revised\_2”, and only includes competitor menu information. It is used for PowerBI dashboard Scout\_Data\_20200922, tab “SCOUT RAW DATA“.

Table Preview:

**PBI\_Unit\_Competitor\_Lookup**

It is the subset table (also simplified version) of “Unit\_Regional\_Benchmarking\_UCA” which is used for PowerBI Dashboard Scout\_Data\_20200922, tab “SCOUT RAW DATA“.

Table Preview:

**PBI\_ZIP\_Competitor\_Lookup**

It is the subset table (also simplified version) of “Regional\_Benchmarking\_UCA” which is used for PowerBI Dashboard Scout\_Data\_20200922, tab “SCOUT RAW DATA“.

Table Preview:

198

199

A Guide for Updated Nielsen Data  
This documentation is a checklist for when new Nielsen data comes through.

**Before starting your process, please fork the GitHub repo compassdigital.ScOUT to your machine. Please NEVER run any of the following processes on Digital Ocean servers.**

Sample Smartsheet request:

“Hi Marcia, We have new Nielsen Data! Can you please update the file with this new data? Please let me know if you need anything.”

Steps to follow:

The pricing team will upload the newest Nielsen data onto Smartshee. Please download the file. Upload the file onto S3 bucket “cdl-scout/Nielsen\_Data/“.

You should already clone compassdigital.ScOUT repo. Please run script “/Scout\_Pipeline/ETL/Nielsen\_CPG.py”. This step will update the database with the newest Nielsen data.  
Please go to PowerBI and refresh dashboard “Scout\_Data\_20200922“.  
Please mark Smartsheet request as 'compelete' and inform the pricing team about the update.

NOTES:

If there are any changes in the mapping files (e.g. updated any market basket/locality mappings). In the script, you can update the code shown below:

1 marketbasket = red\_excel\_from\_s3(Input\_Bucket='cdl-scout', Input\_Key='Nielsen\_Data/CPG Data Mappings - M 2 locality = red\_excel\_from\_s3(Input\_Bucket='cdl-scout', Input\_Key='Nielsen\_Data/CPG Data Mappings.xlsx',

200

Transform and Load Data ETL on AWS  
These are the “Step 2” instructions of the AWS Transform and Load Data pipeline. For the logic design of the pipeline, please refer to Tra

nsform & Load Process . This guide focuses on the structure and operation of this pipeline implementation on AWS. The Transform and Load Data step is split into 4 separate lambda functions:

MapDataFn:

1. When the scraped competitor data is ready, upload the competitor data files to cdl-scout-aws S3 bucket.  
2. Run MapDataFn with as test event. Pass the standardized\_competitor\_name , s3\_keyname , s3\_filename lists as event input. 3. Result files are saved to Pipline\_Staging\_Table S3 bucket.  
4. Send the updated file to Pricing team.

QADataFn:

1. Receive **QA\_Market\_basket\_conversion.csv** from Pricing team and upload to cdl-scout S3 bucket. 2. Run QADataFn with as test event. Pass s3\_key\_list , s3\_filename\_list as event input.  
3. Result data is updated to AWS\_PBI\_Scout\_QA\_Dashboard database table.  
4. Refresh PBI Dashboard and wait for stakeholders to check off.

MatchLoadDataFn:  
1. After stakeholder has checked off, run MatchLoadDataFn with competitor\_list , competitor\_formal\_name as input.

2. The function updates the AWS\_PBI\_Scout\_Raw\_Data , AWS\_PBI\_ZIP\_Competitor\_Lookup , AWS\_PBI\_Unit\_Competitor\_Lookup , AWS\_PBI\_Scout\_Competitor\_Adress\_Lookup tables.

NielsenDataFn:  
1. Run NielsenDataFn with nielsen\_file\_name as input. 2. The function will update AWS\_PBI\_Nielsen\_Raw table.

AWS\_National\_Competitor\_Prices\_Labelled\_Revised\_2 ,

201

Links:

The AWS CDK code repo for this pipeline: GitHub - TomLuoYuehseng/scout-pipeline-aws

Scout-pipeline-aws CloudFormation Arn: arn:aws:cloudformation:us-east-2:736888855894:stack/ScoutPipelineAwsStack2/88b75750- d102-11ed-855f-0afc8ca382f6

202

Scout Useful Files & Tables

Scout has multiple dashboards. Each dashboard is meant for different usage and purpose. In the process of creating & updating these dashboards and reports, we need help from Strategic Pricing Team to identify key relationships between external data and internal data. For example, Mapping external menu items to internal menu items.

In this document, we have listed the locations of the files. All dashboards are currently located in a Power BI workspace named '**Scout Production**'. Please contact Marcia Ma (marcia.ma@compassdigital.io) for any access requirements.

1. Scout (Main)  
This is the main dashboard of the project.

Report Name: Scout\_Data\_20200922 Page Name: SCOUT RAW DATA

**File Location**

s3://cdl-scout/QA\_Market\_Basket\_Conversion.csv

2. Inflation Report

**Owner**

Strategic Pricing Team (Rachel.Castellucci@compass- usa.com)

**Role**

Mappings between external menu items with Compass market baskets.

This dashboard is updated monthly. The dashboard is an inflation rate (CPI & ECI) tracker. The data on this dashboard can go as far as 10 years ago.

203

Report Name: Inflation Report Page Name:

**File Location**

s3://cdl-bi-data-science/bls-gov-scrape/series-ids.csv

s3://cdl-bi-data-science/bls-gov-scrape/CPI\_ECI\_Rates\_Mapping.csv

s3://cdl-bi-data-science/bls-gov-scrape/bls-series-region.csv

**Owner**

Strategic Pricing Team (Rachel.Castellucci@compass- usa.com)

Strategic Pricing Team (Rachel.Castellucci@compass- usa.com)

Strategic Pricing Team (Rachel.Castellucci@compass- usa.com)

**Role**

List of needed CPI & ECI series IDs.

Map CPI & ECI series IDs with the corresponding category name.

Map CPI & ECI series IDs with the corresponding region.

3. ERI  
The ERI dashboard includes information about minimum wages based on location & job position.

204

Report Name: ERI Page Name: -

**File Location**

**Owner**

Strategic Pricing Team (Rachel.Castellucci@compass-usa.com)

**Role**

Hourly minimum wages data by job title and region.

The hourly wage matrix file is sent quarterly through email/SmartSheet.

**4. Scout Nielsen**

Scout Nielsen dashboard shows CPG pricing data (bottled soda/candy bar), which can be filtered based on different geo-locations.

Report Name: Scout\_Data\_20200922  
Page Name: CPG RAW DATA - LOCALIZED

205

**Owner**

**Role**

s3://cdl-scout/Nielsen\_Data/Canteen Pricing Data 13 WE 01.30.2021.csv

s3://cdl-scout/Nielsen\_Data/CPG Data Mappings\_20220426.xlsx

**5. Internal Pricing & Internal Velocities**

Strategic Pricing Team (Rachel.Castellucci@compass-usa.com)

Strategic Pricing Team (Rachel.Castellucci@compass-usa.com)

Nielsen pricing file, updated every 6 months. New files will be sent through email/SmartSheet. CDL Scout team needs to upload the file to the destination location.

The conversion file between Nielsen CPG items with Compass designed market baskets.

Internal Pricing & Internal Velocities dashboards include Compass POS data for selected units. Missing dashboard images as the data updates are temporarily paused.

Report Name: Scout\_Data\_20200922  
Page Name: INTERNAL RAW DATA/INTERNAL VELOCITY DATA

**File Location**

s3://cdl-scout/Internal\_Pricing\_Exclusions\_Tiers.xls

s3://cdl-scout/Internal\_Velocities\_Exlcusions\_Tiers.xls

**Owner**

Strategic Pricing Team & CDL Scout Team (Rachel.Castellucci@compass-usa.com)

Strategic Pricing Team & CDL Scout Team (Rachel.Castellucci@compass-usa.com)

**Role**

Map internal POS menu items with Compass designed market baskets using Regex logics.

Map internal POS menu items with Compass designed market baskets using Regex logics.

**Role**

This file includes multiple tabs, which contain information like price option logics, region mappings, Starbucks pricing by zones, CPG & Entree data mapping criteria.

This is the report template which we load when exporting the guides.

**6. Low-Touch Guide (Retail Benchmarking Guide)**

It is a report exported by 3-digit zip which the CDL Scout team generates yearly.

**File Location**

“/DataDrive/Data Science/Marcia/Re\_Opening\_Guide//data/LTG\_mappings\_202112.xl sx”

“/DataDrive/Data Science/Marcia/Re\_Opening\_Guide//data/LTG\_Export\_Template\_20 2112.xlsx”

**Owner**

Strategic Pricing Team (Rachel.Castellucci@compass-usa.com)

Strategic Pricing Team (Rachel.Castellucci@compass-usa.com)

**File Location**

206

Automated Data Pipeline

207

Parser execution on Airflow with Kubernetes

This page describes the process that has been implemented to execute parsers for the Scout project and what steps are required to include new parsers to the job.

Quick start:

Add parsers to container repo following existing examples.  
Add parsers to list in production airflow  
Get new image tag from ECR and update airflow variable with the new tag. Execute dags

NB> If a parser takes longer than 15 minutes to run the permissions expire and airflow is unable to remove the pod, marking the task as failed. This is okay, double check the expected output to see if it executed correctly.

NB> REMEMBER to add any new Competitors to S3 cdl-scout/historic\_scrape\_data/Competitor and include cdl- scout/historic\_scrape\_data/Competitor/menu, cdl-scout/historic\_scrape\_data/Competitor/location or both!

Relevant Links:  
Main Repo: https://github.com/compassdigital/cdl-dt-airflow  
Container Repo: https://github.com/compassdigital/cdl-di-airflow-containers  
Airflow Production: https://airflow.compassdigital.org/  
ECR Image Tag: https://us-east-2.console.aws.amazon.com/ecr/repositories/private/736888855894/scout\_parsing?region=us-east-2

Full walk through:

Things to keep in mind:

When creating a dev branch a repo use the exact Jira ticket name, eg “DS-294” and use the same name for both airflow production and containers.

Ensure all hard coded file paths are changed to fit in the container.  
Use “moes\_parser.py” as an example for how to import custom helper files and deal with variables. All variables are .env variables that get passed through the airflow execution dag. See the “kubernetes\_parser\_dag.py“ for an example.

208

The pull request on the container repo is managed and will always require someone else to review it. Ensure you have your flake8 linter configured properly so that you dont push linting errors into either repo.

Image tag and variables:

Ensure you click into the image repo, take the most recent image tag and then replace only the end string in the airflow variable. An example is shown in the three images below.

Local Testing:

If you want to run the items locally you will have to have docker available and the required credentials:

-e aws\_secret\_access\_key=\*\*\*\*\*\*\* -e aws\_access\_key\_id=\*\*\*\*\*\*\*  
-e region\_name=\*\*\*\*\*\*\*\*  
-e parser\_run=moes\_parser

One way to build and execute the image is to:

1. Navigate into the correct directory on your branch cdl-di-airflow-containers/scout\_parsing // DS-294
2. Build the container with docker build -t img\_name . Where the img\_name is whatever you want.
3. You can check all of the build images with docker image ls
4. Store your variables as above without the -e flag. The .env file can be stored in the scout\_parsing directory as it is ignored and may be useful for future testing.
5. You can then execute with docker run --env-file=./.env parsers with this example having used 'parsers' for the img\_name.

209

6. Alternatively you can swap the and instead pass your env variables with the -e flag.

--enf-file=./.env

210

Scrape

211

Parse

Overview

**The automated parsing pipeline includes:**

1. Sync page sources from Digital Ocean to S3  
2. Trigger competitor specified parser function once there are n page sources 3. Load parsed data into PowerBI QA dashboard  
4. Map to market baskets after Strategic Team signs off (Manually triggered) 5. Load finalized data into PowerBI Scout Production

Rules

**Rename Python script name to match with the S3 folder name**

e.g. instead of friendlys\_parser.py, change it into Friendlys.py

**Transfer parser functions from digital ocean to container**

If a competitor’s page source **contains both menu and location**:

1. In menuparser() , update target\_id regex into re.split('/', filename)[-1]

1 parsingoutput['target\_id'] = re.split('psource\_20201030/', filename)[-1]

1. In main() , update ‘prefix’ based on the timestamp file (s3://cdl-scout/Temp\_Page\_Source/202101.txt)
2. In main() , update scraped\_date, add timestamp and competitor\_rank
3. In main() , change function format, define menu\_key , add concat\_parse\_result() , and append new result to ' PBI\_Scout\_QA\_Dashboard ' for QA purpose

1 2 3 4

timestamp = get\_time\_stamp()

key\_list\_menu = get\_all\_files\_from\_bucket(

bucket='cdl-scout',

prefix='Temp\_Page\_Source/Friendlys/psource\_' + timestamp + '/')

1. 1  friend\_full = format\_data\_key(df=friend\_result, *# noqa*
2. 2  standardized\_name="Friendly's",
3. 3  competitor\_url='https://friendlys.olo.com/',
4. 4  scraped\_date = int(timestamp + '01'),
5. 5  timestamp=timestamp,
6. 6  competitor\_rank=21) *# noqa*

212

If a competitor requires a **separate menu and location parser**:  
i. In menuparser() and locparser() , update target\_id and parent\_id regex into re.split('/', filename)[-1]

ii. In main() , update ‘prefix’ to be dynamic based on the timestamp file (s3://cdl-scout/Temp\_Page\_Source/202101.txt)

1 parsingoutput['target\_id'] = re.split('/', filename)[-1] 2 parsingoutput['parent\_id'] = re.split('/', filename)[-1]

1 2 3 4 5 6 7 8

timestamp = get\_time\_stamp()

key\_list\_menu = get\_all\_files\_from\_bucket(

bucket='cdl-scout',

prefix='Temp\_Page\_Source/FiveGuys/psource\_' + timestamp + '/')

key\_list\_loc = get\_all\_files\_from\_bucket(

bucket='cdl-scout',

prefix='Temp\_Page\_Source/FiveGuys/location\_' + timestamp + '/')

iii. In main() , update scraped\_date, add timestamp and competitor\_rank

iv. In main() , change function format, define menu\_key , location\_key , conversion\_key , full\_result\_key , add concat\_parse\_result() , join\_parse\_result() , and append new result to 'PBI\_Scout\_QA\_Dashboard' for QA purpose

1. 1  five\_full = format\_data\_key(df=five\_full,
2. 2  standardized\_name="Five Guys",
3. 3  competitor\_url='https://www.fiveguys.com/',
4. 4  scraped\_date = int(timestamp + '01'),
5. 5  timestamp=timestamp,
6. 6  competitor\_rank=4)
7. 1  *# define paths*
8. 2  menu\_key = 'Pipline\_Staging\_Table/FiveGuys/fiveguys\_menu\_staging.csv'
9. 3  location\_key = 'Pipline\_Staging\_Table/FiveGuys/fiveguys\_location\_staging.csv'
10. 4  conversion\_key = 'Temp\_Page\_Source/ConversionFiles/fiveguys.csv'
11. 5  full\_result\_key = 'Pipline\_Staging\_Table/FiveGuys/fiveguys\_staging.csv'

6

1. 7  *# concat the old & the new parsing result*
2. 8  concat\_parse\_result(key=menu\_key, new\_result=fivemm\_result)
3. 9  concat\_parse\_result(key=location\_key, new\_result=fivel\_result)

10

1. 11  *# Merge menu and location*
2. 12  five\_full = join\_parse\_result(menu\_key, location\_key, conversion\_key, full\_result\_key)
3. 13  write\_df\_to\_csv\_on\_s3(five\_full, output\_bucket='cdl-scout', output\_key=full\_result\_key) *# noqa*
4. 14  append\_table\_into\_db(five\_full, 'PBI\_Scout\_QA\_Dashboard')

15

**Upload page sources, conversion files to “s3://cdl-scout/Temp\_Page\_Source/”**

**This is a temporary folder. Page sources will be DELETED after parse.**

“s3://cdl-scout/Temp\_Page\_Source/202101.txt” is the timestamp, always remember to update the **filename** to match with the most

current scraping round.

Every competitor has its own page source folder, the naming rule looks like this – “FiveGuys“, “Chickfila“, “Friendlys“. Always remember to match the name with the path in parser functions. The folder structure looks like this –

1 *# define paths*2 menu\_key = 'Pipline\_Staging\_Table/Friendlys/friendlys\_staging.csv' 3 *# concat the old & the new parsing result*4 concat\_parse\_result(key=menu\_key, new\_result=friend\_full)  
5 append\_table\_into\_db(friend\_full, 'PBI\_Scout\_QA\_Dashboard')

213

“s3://cdl-scout/Temp\_Page\_Source/ConversionFiles/” stores all conversion files for competitors who have separate menu and location page sources. Filename only contains competitor name and is in lower case, e.g. “fiveguys.csv“

214

Map and Update

**Process:**

1. Strategic Team marks a completed scrape  
2. Delete competitor from PowerBI QA dashboard  
3. Start the market basket mapping and price update process  
4. Push the updated, finalized data to Scout production dashboard

215

Competitor Web Tracker 1 Panera Bread

1.1 2021-10-06

1.2 2021-12-07 2 OLO Sites

2.1 2021-09

2.2 2021-11 3 Boston Market

3.1 2021-11-01

3.2 2021-12-06 4 Cracker Barrel

4.1 2021-11-01 4.2 2021-12-08

Panera Bread

**2021-10-06**

The website looks different from it was ~2 years ago. It gives 'fake' menu information in which the prices are the same across the country. Avoid this menu by if not re.search('https://www-api.panerabread.com/www-api/public/menu/placards/500000', url)

Location info → re.compile(r"https://www-api\.panerabread\.com/www-api/public/menu/versions/\d+") Menu info → A single API call contains a main item, customizations, and store id re.compile(r"https://www-

api\.panerabread\.com/www-api/public/menu/placards/\d+/version/\d+/\d+/en-US")

**2021-12-07**

Menu info → The same API call does not includes customizations and store id anymore.  
Store id is now in re.compile(r"https://www-api\.panerabread\.com/www-api/public/menu/versions/\d+")

Customizations are stored separately. The file name is the product name hash. re.compile(r"https://www- api\.panerabread\.com/www-api/public/menu/placard/hashes/v2/\d+/version/\d+/en-US")

OLO Sites

**2021-09**

No ReCaptcha

**2021-11**

Add ReCaptcha to most of the sites, which need cookie developments

Boston Market

**2021-11-01**

Menu info → click URLs from sitemap (https://www.bostonmarket.com/menu/family-meal-options/ ), click 'start order', the website will redirect to the map section. From there, addresses need to be entered to get the store-specific price.

216

**2021-12-06**

Menu info → The ‘start order' button does not work anymore, it is now a 'fake’ button. Instead, URLs need to be generated by item in this format (https://my.bostonmarket.com/location/1669/menu/individual-meals/half-rotisserie-chicken ).

Cracker Barrel

**2021-11-01**

Menu info → click URLs from sitemap (https://togo.crackerbarrel.com/menu/abilene-tx/products/9147411)

**2021-12-08**

Location info → needs to be scraped first.

Menu info → needs to generate product URLs in this format (https://www.crackerbarrel.com/menu/chicken-n-turkey/thursday-turkey-n- dressing?productid=44707256), and enter location information to get the store-specific price.

217

Wezel Documentation

*Wezel is a Compass Digital owned project developed by the Scout data sciences team that's main focus is on web scraping and data collection to populate the scout dashboard*

**Overview**

Wezel is a web-scraping / data collection engine built in python and javascript. Wezel’s main focus is on scraping and collecting price and location data from our competitors to populate the scout dashboard.

**Prerequisites**

in ~/wezel folder create a file named proxies.txt this file holds a list of proxies\ **Example Of proxies.txt format**

These proxies are only examples and will not work

In ~/wezel/db/ create a file called wezel.db , wezel.db and wezel.db Create a scrape\_data folder mkdir scrape\_data  
Enter the scrape data dirctory cd scrape\_data

Run command (Note for new competitors a new folder must be created) mkdir panera\_bread fiveguys chick\_fil\_a chipotle ihop dennys jerseymikes red\_robin potbelly corner\_bakery boston\_market bobevans blimpie shake\_shack jimmy\_johns au\_bon\_pain moes friendlys which\_wich

Return back to the wezel dir \home\name\wezel

Create a parse\_data folder mkdir parse\_data

Enter the parse\_data dir with cd parse\_data

Enter the command mkdir panera\_bread fiveguys chick\_fil\_a chipotle ihop dennys jerseymikes red\_robin potbelly corner\_bakery boston\_market bobevans blimpie shake\_shack jimmy\_johns au\_bon\_pain moes friendlys which\_wich

**Required packages NPM**

Node.js  
puppeteer-extra random-useragent puppeteer-extra-plugin-stealth puppeteer-extra-plugin-adblocker Pandas  
Numpy  
Flask  
Tmux

1 77.247.113.14:8800  
2 154.38.153.240:8800 3 155.94.232.118:8800 4 192.126.219.135:8800 5 172.93.233.53:8800  
6 77.247.113.81:8800  
7 193.38.241.127:8800

218

**Getting Started**

Clone Repo sh git clone https://github.com/github\_username/repo\_name.git Open the terminal and install virtualenv with the command pip install virtualenv

Inside the Wezel project folder, there is arequiments.txt. This file holds all the project decencies make sure this file is inside of the Wezel folder.  
Enter the Wezel directory with the command cd ~/wezel  
Create a virtual environment with the command python3 -m venv env

Activate the virtual environment source env/bin/activate

Install all the dependencies with the command pip install -r requirements.txt

Make sure node js is installed min version v16.15.0 Node.js instalation

Install nodejs dependencies npm install (puppeteer, puppeteer-extra, random-useragent, puppeteer-extra-plugin-stealth, puppeteer-extra-plugin-adblocker)

**Startup**

Navigate to the home directory cd ~/wezel and start the source env/bin/activate In the dir wezel/config\_file/startup\_config.json

Keep the name as env

1 ```  
2 "proxy\_path":"./proxies.txt",  
3 "competitor\_paths":"./config\_file/competitor\_config.json", 4 "user\_agents":null,  
5 "job\_sqlite\_path": "./wezel\_db/wezel\_jobs.db",  
6 "data\_sqlite\_path": "./wezel\_db/wezel\_data.db",  
7 "error\_sqlite\_path": "./wezel\_db/wezel\_error.db",  
8 "whip\_job\_database":false,  
9 "whip\_parsed\_database":false,

10 "whip\_error\_database":false,  
11 "home\_dict":"/home/adam/wezel", 12 "amount\_of\_nodes":4,  
13 "runs\_before\_kill":2  
14 ```

**Start up config Explanation:**

proxy\_path: Location path to the list of proxies used by Wezel.  
competitor\_paths: Location path for the second config file.  
job\_sqlite\_path: Location path for the SQLite database where all scraping jobs.  
data\_sqlite\_path: Location path for the SQLite database where all the parsed data is stored.  
error\_sqlite\_path: Location path for the SQLite database where all scraping and parsing errors are stored for review. whip\_job\_database: Boolean value that deletes the job database.

whip\_parsed\_database: Boolean value that deletes the parsed database. whip\_error\_database: Boolean value that deletes the error database. home\_dict: Personal path to wezel folder ‘/home/name/wezel’. amount\_of\_nodes: Controls the number of the process for each scraper.

If amount\_of\_nodes is set to 3 that means that Wezel will scrape 3 URLs of each active competitor at the same time. runs\_before\_kill: A number value set to how many times a proxy is used before getting swapped out for a new one.

219

In the dir

1 ```  
2 "company\_name": 3{

1. 4  "name":"Company Name",
2. 5  "puppeteer\_scrape\_path":"scrapes/js\_scraps/company\_name/company\_name\_scrape.js",
3. 6  "sitemap\_path":"scrapes/sitemap/company\_name/company\_name\_location\_sitemap",
4. 7  "status":"ACTIVE",
5. 8  "save\_path":"scrape\_data/company\_name",
6. 9  "parser\_path": "/parsers/company\_name\_parser/company\_name\_parser.py",
7. 10  "parser\_save\_path": "/parse\_data/company\_name/wezel\_data.csv"
8. 11  }
9. 12  ```

**Competitor config Explanation:**

name: Name of the competitor

puppeteer\_scrape\_path: Location path of the scraper.

sitemap\_path: Location path of the sitemap.

status: Status represents if the scrape is active meaning when Wezel is running is that competitor being scraped (ACTIVE / INACTIVE)?

save\_path: Location path where the scraped data will be saved.  
parser\_path: Location path where parser file is stored.  
parser\_save\_path: Location path where the parsed data would be saved a as CSV file.

Once the config files have been specified you can run the wezel with the command python3 start\_up.py TMux goes here

wezel/config\_file/competitor\_config.json

220

Parsing in Wezel

**Why Parsers are Used in Wezel**

Parsers are used to transform and extract the the raw data scraped from competitor websites into a format appropriate for populating the scout dashboard. This includes obtaining menu item cost, calorie and description information along with competitor location information from JSON or HTML source pages. Previously, parsers were ran separately from Wezel after all jobs for the competitor had been scraped and saved. They have been integrated in a way so that data is parsed in Wezel as each scrape job is run. This was done so that it can be ensured that the scraper is functioning appropriately and collecting necessary information as it is being ran.

Parser Format  
For each competitor, a parser should be made corresponding to the "parser\_path" for the competitor in the competitor configuration file.

The path of the scraped data file is passed in as a system argument from pg\_source along with the target\_id .  
The parser function will parse the file at that path and print a list of the desired menu items and modifiers along with the location information

and the target id. Some pseudocode for a general parser is shown below.

**template of parser script**

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

**import** os, sys  
**import** pandas **as** pd  
**import** re  
**from** bs4 **import** BeautifulSoup **import** json  
**import** numpy **as** np

**def** parse\_data(pg\_source, target\_id): '''

pg\_source: string, path of scrape file target\_id: string, target\_id of scrape file '''  
**try**:

**with** open(pg\_source,'r',encoding='utf-8') **as** r:  
/\*either use Json **or** BeautifulSoup depending on scrape file \*/  
/\* parse menu information to populate the following fields \*/  
menu\_list += [{'menu\_item\_name': menu\_item\_name,'menu\_item\_price\_original': menu\_item\_price\_original,'men /\***for** each modifier that has extra cost populate the following fields \*/  
menu\_list += [{ 'menu\_item\_name': addon\_name,'menu\_item\_price\_original': addon\_price\_original,'menu\_item\_ /\*add competitor location information **and** target id to menu items \*/  
**for** i **in** menu\_list:

i.update({'competitor\_name': competitor\_name, 'competitor\_address':competitor\_address, 'city':city, 'po

'competitor\_source\_lat':competitor\_source\_lat, 'competitor\_source\_lng':competitor\_source\_lng, '

menu\_df = pd.DataFrame(menu\_list)

*#remove null values for required columns and rows with no price*

menu\_df.dropna(subset=['menu\_item\_name','menu\_item\_price\_original','competitor\_name','city','postal','state records = menu\_df.to\_dict('records')  
**print**(records)

**except**:  
*#error messaging*error\_message = 'Error at {} on line {}: '.format(sys.exc\_info()[-1].tb\_frame.f\_code.co\_filename,sys.exc\_in

221

u c

s t

'

f

The parser will print a list of dictionaries containing menu or modifier information along with competitor location information. The required fields of these dictionaries is shown in the example parser output below for a scrape job. Each scrape job may produce a list with multiple dictionaries depending on the modifiers and amount of menu items being scraped at a time.

**example parser output:**

1. 1  [{'menu\_item\_name': 'Cornbread -- One', 'menu\_item\_price\_original': 1.29,
2. 2  'menu\_item\_cal': 160.0,
3. 3  'menu\_item\_des': 'Not a slice. Not a piece. But your own mini-loaf.',
4. 4  'competitor\_name': 'Boston Market Warren',
5. 5  'competitor\_address': '26021 Hoover Road',
6. 6  'city': 'Warren',
7. 7  'postal': '48089',
8. 8  'state': 'MI',
9. 9  'competitor\_source\_lat': '42.485372',
10. 10  'competitor\_source\_lng': '-83.006392',
11. 11  'target\_id': '33024ea5520a3a695c32c59c1df5edfb'}]

Wezel Parser Data Database

The database /wezel\_db/wezel\_data.db is created store the parsed data as well as track the success and failures of the jobs being parsed. The python file /wezel\_db/data\_db\_functionality.py contains functions used to manage and update the database. Each time a scrape job is parsed a new row is created in the database with the job target\_id the competitor company\_name and the parsed scrape data information fields as shown in the example parser output above. The csv\_path is also set to match the parsed\_save\_path from the competitor config file. Data is added to this database through the function parse\_scrape\_data in /nexus.py . The schema for the database is shown below.

**schema for wezel\_data database**

1. 34  error = [{"parser\_error": error\_message}]
2. 35  **print**(str(error))

36 37

1. 38  **if** \_\_name\_\_ == '\_\_main\_\_':
2. 39  pg\_source= sys.argv[1]
3. 40  target\_id = sys.argv[2]
4. 41  parse\_data(pg\_source, target\_id)

222

Accessing Parser Results with Wezel Server

The file /server/wezel\_server.py contains the server routes used to display all parsed competitor data or parsed data specific to a certain competitor.

**Different app routes in wezel\_server.py**

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

def run(self):

app = Flask(\_\_name\_\_)

@app.route("/showdata")

def showdata():

parsed = self.show\_data()

return jsonify(results = parsed)

@app.route("/showerror")

def showerror():

error = self.show\_error\_data()

return jsonify(results = error)

@app.route("/data/<name>")

def data(name):

show\_data = self.show\_data\_by\_name(name)

return show\_data

@app.route("/tocsv/<name>")

def tocsv(name):

message = self.data\_to\_csv(name)

return message

@app.route("/")

def home():

return "server is running ..."

app.run(host="142.93.157.14", port=5001, debug=False)

**Accessing Parser Results**

As shown with the app routes above, in order to determine if the server is running you can go to the url of the server and port in the format https://<host>:<port> . For example, if the production server and port 5001 is being used you can check if the server is running at

http://142.93.157.14:5001/.

For accessing the live parsed data results in a table format the url in the format https://<host>:<port>/showdata can be used. This will show the results from all active competitors.

223

**Example parser results accessed from the server**

224

In order to access a specific competitor’s results on the server you can use the url in the format https://<host>:<port>/data/<name> . For example if you wanted to access the parsed results for Red Robin on port 5001 of the production server, use the link: http://142.93.157.14:5001/data/RedRobin.

**Saving parser results to CSV**

The server route https://<host>:<port>/tocsv/<name> can be used to save all the parsed results from a competitor to a csv file specified at the address in the competitor config file. Whenever this address is refreshed the csv file is overwritten with the current results in for the competitor from the wezel\_data database. A message will be displayed with the csv path if it has been successfully created/saved.

225

Web-scraping in Wezel

**Web-scraping overview**

Wezel uses web-scraping as the backbone of collecting its data, the engine works based on passing and collecting information too and from scrapes and saving the data in the correct directory.

**Web scraping with Puppeteer**

Wezel’s scraping capabilities are powered by puppeteer due to its ability to be undetectable to bot detection software.

**Required Libraries**

NodeJS  
NodeJS Download NodeJS Docs

**Required NPM Packages**

puppeteer-extra random-useragent puppeteer-extra-plugin-stealth puppeteer-extra-plugin-adblocker

**1 - Passing information from NodeJs to Python**

Naked.io has the main function that is called on execute\_js and muterun\_js . execute\_js runs the javascript normally through the terminal but does not allow for data to be passed back to Python. muterun\_js runs the javascript in the background of the python and allows data to be passed back into Python through console.log information in python scrips data = result. stdout.decode("utf-8").

**2 - Initialization of Puppeteer**

1  
2  
3  
4  
5 6] 7 })

puppeteer.launch({

headless:true,

ignoreHTTPSErrors: true,

args: [

`--proxy-server=${proxy}`

Puppeteer Launch

headless : Headless takes in a boolean value.

ignoreHTTPSErrors : Ignores all errors of HTTP errors that might cause issues with scraping.

args : Takes in a list of arguments and exceptions that can change the way the Puppeteer operates. Wezel uses args to change the IP address to a proxy.

226

**3 - Gathering Data**

Puppeteer has two ways of collecting website data based on the website being scraped API call interception and source code HTML. One of the ways is through, Data interception. This allows capturing of API network data calls to be caught and saved as JSON files. The second method is through the source code HTML. Where we take a snapshot of the HTML code when the website loads fully and hold all the data that is being displayed.

**3.1- API call interception**

API interception works on the concept of intercepting server calls. Puppeteer listens for these requests and can save and modifying of the return calls.

**API calls**

example\_of\_api\_call:

1 /^https:\/\/location\.api\.my\.chick-fil-a\.com\/locations\/4\.0\/\d+/  
When intercepting API calls a regex expression must be created to match specific API calls as most API calls differ on use cases for

example when scraping food price data websites will call different API requests based on the location selected.

**Initialization of Puppeteer to intercept the request:**

1  
2  
3  
4 5}

page.on("response", async (response) =>{

console.log(response.url())

console.log(response.text())

Initialization of Puppeteer

response. url() : Returns all the website's API calls to load data. response. text() : Returns the data of all the calls made by the website.

1  
2  
3  
4  
5 6}

page.on("response", async (response) =>{

if(response.url() == "www.api-call-url.com"){

try{

console.log(await response.text())

}catch(e){}

**Example of how Wezel intercepts API calls and gathers data to be saved**

**3.2 - HTML scraping**

HTML scraping is loading a webpage and saving the HTML data into a file, this is mainly used if API calls are hidden or cannot be accessed puppeteer allows for saving all HTML data.

Note: Depending on the website, sometimes extra sets are taken to ensure the website loads fully. For example: Scroll to the very bottom of the page or click a button.

1 puppeteer.use(StealthPlugin()) 2 puppeteer.launch({  
3 headless: true,

227

**Example code of how to access the HTML page content**

**4 - Passing Information from Python to NodeJs**

When Wezel initializes scraps it passes job data that is required to complete scrapes. This URL could be a location address. For example, in this URL: https://www.dennys.com/location/dennys-6738/menu/salads. The number 6738 is the location address or Store ID.

Since the Wezel is built in Python and scrapers are built-in Nodejs the communication between the two languages is handled by a Python library called Naked.io.

**Example of output data passed from Python to nodejs using Naked.io.**

1[  
2 '{target\_url:',  
3 'https://order.fiveguys.com/menu/lakewood-2,', 4 'proxy:',  
5 '0.0.0.0:8000}'  
6]

4  
5  
6 7] 8

ignoreHTTPSErrors: true,

args: [

`--proxy-server=${proxy}`

1. 9  }).then(async browser => {
2. 10  const page = await browser.newPage()
3. 11  page.on("response", async (response) =>{})
4. 12  try{
5. 13  await page.setViewport({ width: 1920, height: 1080});
6. 14  await page.setDefaultNavigationTimeout(50000);
7. 15  await page.goto(url,{waitUntil: 'domcontentloaded'})

16  
17 console.log(await page.content()) //gets complete webpage content 18

1. 19  await browser.close()
2. 20  }catch(e){
3. 21  await browser.close()
4. 22  }
5. 23  })

228

Error-Handling in Wezel

**Why Error Handling is Used in Wezel**

With competitor websites constantly changing, scrapers and parsers can often fall behind and need to be updated. When scrapers or parsers no longer work, it is important that developers be aware to address the issue. Additionally, removing jobs that are consistently failing due to these errors enables the Wezel Engine to better allocate resources to other jobs.

Scraper Errors

There are many types of scraper errors, each with unique handling. When a *Page not Found* or *Html Change* error occurs it will be added to the error database and the job will be removed from the job database. In the event of an unhandled error in the scraper file, an error will be displayed directly to the terminal and Wezel must be restarted once the scraper is fixed. Examples of common error types are shown below.

**1. Page not Found:** Occurs when the url on the sitemap is no longer valid. The location could be closed or the menu item is not available at that location. This can be detected by the page or API calls returning a status code of 404.

**example check for page not found error in scraper**

1  
2  
3  
4  
5 6}

7 8 9

10

11

12

13

14

15

//checks if api call returns 404

if(menu\_data.test(response.url())){

if (response.url() == 404){

error = {"error\_message":"page not found 404"}

console.log(error)

browser.close()

} else {

console.log(await response.text())

}

//checks if webpage returns 404

if (url == response.url() && response.status() == 404){

error = {"error\_message":"page not found 404"}

console.log(error)

browser.close()

**2. HTML change:** Occurs when the html of the site has changed and the scraper can no longer find elements. Error catching logic can be incorporated into scraper when selecting elements.

**example check for html error in scraper**

1  
2  
3  
4  
5  
6  
7 8}

try {

const store\_element = await page.waitForXPath('//\*[@id="locationsList"]/div[1]/div/div[2]/button', {timeout

await store\_element.click()

} catch (e) {

error = {"error\_message":"html change"}

console.log(error)

await browser.close()

**3. Proxy Blocked:** Occurs when the site has detected unusual traffic and won’t allow the proxy being used to access the site. Response is inconsistent across different competitors but may appear as a task to verify you are human.

:

229

**4. API change:** Occurs when the url headers of the API calls for menu, modifier or location have been altered so that they are no longer a match for the regex expression used in the scraper. Headers should be checked frequently for competitors before refreshing data as slight alterations are common.

Parser Errors

Even if the webpage has been scraped successfully, the required fields may not be able to be parsed from the scraper output. This can be caused by an error within the parser file or a change to the structure of the api structure. When an exception occurs in the parser a detailed error message is printed in the parser file. This message is transferred to the error database and the job is removed from the job database.

**example logic for parser error catching**

Wezel Error Database

**Database Overview**

In the event of an error with the parser or scraper, a new row is added to the /wezel\_db/wezel\_error.db . The python file /wezel\_db/error\_db\_functionality.py contains functions used to manage and update the database. The error messages described

above will be inserted into the error field along with the date and time the error occurred and other details about the job that failed. **schema for wezel\_error database**

**Viewing Errors from the Database**

All errors from the error database can be viewed through the wezel server at the address http://<host>:<port>/showerror . Further filtering can be done by creating functions to manipulate the wezel\_error database if needed.

\_

**Blocked proxies** and **API changes** are not added to the error database. In these cases, the job will be retried with another proxy until it is successful.

If API changes can consistently be detected separately from blocked proxies in the future, error handling should be added for API changes similar to the Page not Found and HTML errors.

1 2 3 4

except Exception as e:

error\_message = 'Error at {} on line {}: '.format(sys.exc\_info()[-1].tb\_frame.f\_code.co\_filename,sys.exc

error = [{"parser\_error": error\_message}]

print(str(error))

230

231

Scraping Sitemaps

A sitemap is **a file where you provide information about the pages, videos, and other files on your site, and the relationships between them**. Search engines like Google read this file to crawl your site more efficiently. For example, https://www.dennys.com/sitemap.xml.

Sitemaps also tell the search engines which pages on the site are important like the location data and the menu item data on the website to have specific items for scraping. For Scout, website sitemaps play a crucial role to help find the list of URLs for vendor restaurant locations.

Purpose  
The purpose of this documentation is to help the team quickly locate a sitemap when needed.

**Scrape Type**

MAP: The website uses a map location system where a location must be typed in to see all the stores nearby. URL: The website's location is determined by its URL. For example (https://www.dennys.com/ location/dennys-

9002 /menu/beverages/dasani-bottled-water). **Scrape Data**

1. API: Data collected will be a JSON file  
2. HTML: Data collected by saving the HTML data

**Name**

**Location Sitemap URL**

https://locations.p anerabread.com/s itemap.xml

https://restaurants .fiveguys.com/site map.xml

**Analysis**

Scrape location address. This is necessary to get to the menu data.

Location is based in the URL, so no location address necessary.

**Scrape Type**

MAP

URL

**Scrape data**

API

API

**Notes**

**Global API call**

N/A

N/A

Panera Bread

Five Guys

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Chick-fil-A | Location listing s | No site map found but a list of all stores found on the website, must scrape all locations using this URL. | URL | API | Location listings add state short form for quicker scrape.  Store menu cant be access if closed. | N/A |
| Chipotle | https://locations.c hipotle.com/sitem ap.xml | Scrape location address, this is necessary to get to the menu data. | MAP | API | To scrape the menu use Mexican Fo od - Restaurant & Catering - Chipotle M exican Grill and website auto redirects  to enter location page and can access the API. | https://services.chipotle.com/menui nnovation/v1/restaurants/482/online menu? channelId=web&includeUnavailableIte ms=true  https://services.chipotle.com/menui nnovation/v1/restaurants/482/online meals?includeUnavailableItems=true |

IHOP

Denny’s

https://restaurants .ihop.com/sitemap .xml

https://locations.d ennys.com/sitema p.xml

Location is based in the URL, so location address necessary.

Scrape location address, this is necessary to get to the menu data.

URL

MAP

HTML

HTML

No API calls available

https://nomnom-prod- api.dennys.com/restaurants/36847/me nu?nomnom=add-restaurant-to-menu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Jersey Mike’s | nan site map must be generated by scraping All Ou  r Locations - Jers ey Mike's Subs | Location is based in the URL, so location address necessary. | URL | API |  | N/A |

232

Red Robin

MAP API

Potbelly

Corner Bakery

Bob Evans

Moe’s Southwest Grill

https://www.potbel Scrape location address, this is ly.com/sitemap\_lo necessary to get to the menu data. cations.xml

Corner Bakery Location is based in the URL, so Cafe - All Locatio location address necessary.  
ns

https://www.bobev Sitemap locations can access the ans.com/sitemap. menu.  
xml

https://locations.m Location is based in the URL, so oes.com/sitemap. location address necessary.  
xml

MAP API

URL API

MAP API

URL API

N/A

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Boston Market | https://www.bosto nmarket.com/locat ion/sitemap.xml | Store ID is based in the URL and no address required to us map. | URL | API | https://my.bostonmarket.com/location/st orenumber/menu  Generate sitemap based on the store id’s | https://my- api.bostonmarket.com/restaurants/29 085/menu?nomnom=add-restaurant-to- menu |

No need to enter the address to access the menu, but has the possibility of using the map.

N/A

N/A hidden main api call

N/A

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Blimpie | Blimpie Store L ocator | Location is based in the URL, so location address necessary. | URL | API | URL of each product in format:  https://order.blimpie.com/menu/blimpie- 108/products/17494370  where the product id is unique to each location. Can be found from global API call and then used to get to modifiers. | https://order.blimpie.com/api/vendo rs/blimpie-108 |
| Au Bon Pain | All Locations | Au Bon Pain | Location is based in the URL, so location address necessary. | URL | API | URL of each product in format:  https://order.aubonpain.com/menu/30th- street-station/products/9698748  where the product id is unique to each location. Can be found from global API call and then used to get to modifiers. | https://order.aubonpain.com/api/ven dors/30th-street-station |
| Shake Shack | https://shakeshac k.com/sitemap.xm l | Have to scrape the street address from the locations listed in the sitemap and use those to enter into the location map. | MAP | API | After typing in address, involves a few popups that ensure location is correct, especially if proxy is far from location. | https://order.whichwich.com/api/ven dors\/<vendor-id>/ |
| Which Wich | https://order.which wich.com/sitemap .xml | Location is based in the URL, so location address necessary. | URL | API | URL of each product in format:  https://order.whichwich.com/menu/which wich-christiansburg/products/52058369  where the product id is unique to each location. Can be found from global API call and then used to get to modifiers. | https://order.whichwich.com/api/ven dors/whichwich-<location-slug> |
| Jimmy John’s | https://locations.ji mmyjohns.com/sit emap.xml |  |  | API | Hard to get to final url  Url of each product in format  https://online.jimmyjohns.com/menu/jj32 59/products/33242010/ | https://online.jimmyjohns.com/api/v endors/jj0805 |

Friendly’s

Le Paris Quotidien

Lemonade

Sweetgreen

https://locations.fri Location is based in the URL, so endlysrestaurants. location address necessary. com/sitemap.xml

n/a

https://lemonadela .com/api/restaura nts

https://order.sweet green.com/

URL API

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Paris Baguette | https://www.parisb aguette.com/wp- admin/admin- ajax.php? action=store\_sear ch&lat=40.51296 &lng=-74.40854& max\_results=100 &search\_radius=1 0&autoload=1 |  |  |  | API is console logged in the webpage need to find a way of gathering this data. |  |

API

233

Peet’s Coffe

Domino’s Tropical Smoothie Cafe

https://stockist.co/ api/v1/u5687/locat ions/all

https://locations tr Location is based in the URL so URL API

Need more time to find sitemap.

234

Scraping Guides

**Tier**

**last update**

07-27-2022 07-27-2022

07-27-2022 07-27-2022 07-27-2022 07-27-2022 07-27-2022 07-27-2022 07-27-2022 07-27-2022 07-27-2022

**Scraping Guide**

Open up the urls and click Order Pickup or Delivery button and wait for the webpage to load wait for the api call let menu = /^https:\/\/nomnom-prod-api\.fiveguys\.com\/restaurants\/\d+\/menu\?nomnom=add-restaurant-to-menu/

**Scrape Type**

API

**Company Name**

1 1

1 1 1 1 2 2 2 2 2

Panera Bread Five Guys

Chick Fil A Chipotle  
IHOP  
Denny’s Jersey Mikes Red Robin Corner Bakery Boston Market Bob evans

235

Monthly Refresh Data Back Up Pipeline

This page describes the process that has been implemented to back up results and status files generated by Wezel Engine from the Digital Ocean server to AWS S3 storage. In addition, this page explains what steps are required if new folders need to be synced.

**The file sync process is executed in 3 steps with 3 separate DAG files:**

digital\_ocean\_sync : This DAG task logs in to the digital ocean local server and sets up an SFTP connection to the AWS EC2 server. It copies the Wezel-generated files recursively on to EC2 server.

ENDEAVOR\_SFTP\_S3\_SYNC : This DAG task is maintained by the Data Engineering Team and is shared by all teams working on Airflow 2.0. It is triggered every hour to sync files from the EC2 server to S3 storage. For our task, in particular, it syncs the Wezel-generated files from EC2 to a temporary folder in S3 at cdl-data-ops-ftp-sync/adith\_cdl/in .

scout\_file\_sync : This DAG task syncs the Wezel-generated files from the above-mentioned temporary location in S3 to a permanent location at cdl-scout/Scout\_Monthly\_Refresh\_Data\_Back\_Up . For each month's backup, files are saved into a new directory with trigger data as key name.

**Directories from the digital ocean that are synced in this process:**

parse\_data : Parsed CSV files from each competitors scrape\_data : Page source code of each competitor's webpage sitemap : Competitors' store location sitemaps  
error\_data : Error message reported by Wezel while running

**Relevant Links:**

GitHub location:

S3 temporary file URI: s3://cdl-data-ops-ftp-sync/adith\_cdl/in/  
S3 permament file URI: s3://cdl-scout/Scout\_Monthly\_Refresh\_Data\_Back\_Up

**File Sync Process Diagram:**

GitHub location:

https://github.com/compassdigital/cdl-dataops-airflow-2/tree/dev/dags/data\_engineering/digit

digital\_ocean\_sync

https://github.com/compassdigital/cdl-dataops-airflow-2/tree/dev/dags/data\_engineering/scout\_fil

scout\_file\_sync

e\_sync - Connect your Github account

al\_ocean\_sync - Connect your Github account

236

Testing steps for future updates

**1. DAG file structures**

Clone the cdl-dataops-airflow-2 repo to the local machine. Locate the 2 DAG files for this task under the dev branch and dags/data\_engineering folder. Each DAG includes an airflow folder where the DAG logic is stored, as well as a contained\_script folder where the specific backup process code and docker setup code are stored.

2. /Airflow/dag.py

This file initiates the DAG by specifying the dag name, default arguments, scheduled interval, and DAG logic. For example, the following code is excerpted from digital\_ocean\_sync/airflow/dag.py . There is only one task for each of the digital\_ocean\_sync and

scout\_file\_sync DAGs. As seen here, the sync task is executed with the “--script sync” command in line 15 and set to be a single task in line 20.

1. 1  **with** DAG(
2. 2  dag\_name,
3. 3  default\_args=default\_args,
4. 4  schedule\_interval=schedule\_interval,
5. 5  catchup=False,
6. 6  max\_active\_runs=1,
7. 7  tags=["scount", "sync"],
8. 8  ) **as** dag:

9

1. 10  sync = KubePodSubmit(
2. 11  task\_id="sync",
3. 12  AWS\_SECRET="DATAOPS\_KUBE\_EC2\_SSH",
4. 13  POD\_NAME="scout-sync",
5. 14  IMAGE\_URL=image\_url,
6. 15  ENTRY\_POINT\_ARGS=["--script", "sync"],
7. 16  ENV=env,
8. 17  dag=dag,
9. 18  )

19  
20 sync

**3./contained\_script**

In contained\_script, the Dockerfile has the information to create a docker image from this DAG folder. It will be automatically triggered once changes are pushed to this DAG directory on GitHub. requirements.txt includes the required package for the python code. For specific mechenism on how Airflow 2.0 works, please refer to Airflow as an Orchestrator

**4. /contained\_script/code**

Under the contained\_scripts/code directory, config.py stores the authentication information, as well as the directory path for sync files' stored location. The main.py file serves as a "driver" script that will allow for programmatically calling different functions that are defined in

237

the ./code directory and imported into this file. For each of our DAGs, there is only one task, which is the sync.py for digital\_ocean\_sync and execution.py for scout\_file\_sync.

**5. Steps for Test Runs**

1. Push changes to GitHub repo  
2. Wait for a workflow run to create a docker image in the action tab on GitHub

3. To manually trigger digital\_ocean\_sync DAG, go to Airflow UI and click on “Trigger DAG”

4. Wait for files to sync to s3://cdl-data-ops-ftp-sync/adith\_cdl/in/. This will take a while since ENDEAVOR\_SFTP\_S3\_SYNC is triggered hourly. 5. To manually trigger scout\_file\_sync DAG, go to Airflow UI and click on “Trigger DAG”

238

6. Wait for files to sync to s3://cdl-scout/Scout\_Monthly\_Refresh\_Data\_Back\_Up Test Run Step Diagram

239

Wezel AWS

Introduction Getting started

File extensions & structures

Executing Scraper How to trigger

Relevant Links:  
Development configuration for future updates

Introduction

This article describes the migration of the Wezel engine from the local server to the AWS cloud service. By deploying Wezel on AWS, we can scale up the number of competitor tasks running at the same time and drastically increase the efficiency of scraping jobs with lambda functions. The article also includes instructions on configuring a working environment for developing and deploying changes to AWS for future Wezel updates.

Getting started  
The Wezel AWS project consists of two separate scrapers:

1. For scraping sitemaps and  
2. The other one is for scraping web pages.

Both scrapers utilize AWS Step Function to sequence Lambda functions and handle errors.

**File extensions & structures**

You should keep in mind the following file extensions and what each file means and where it's used:

**File Description**.venv The python virtual environment information.  
cdk.json A configuration file for CDK that defines what executable CDK should run to generate the CDK construct tree.

requirements.txt This file is used by pip to install all of the dependencies for this CDK application. In this case, it includes the aws-cdk-lib the library for CDK supports and aws-cdk.aws-lambda-python-alpha an alpha version tool that allows CDK to create python Lambda functions

without the need to construct a docker image. app.py The “main” file.

wezel\_aws/wezel\_aws\_stack.py Where the wezel\_aws\_stack stack is defined. A stack is a unit of deployment in the AWS CDK. All AWS resources defined within the scope of a stack, either directly or indirectly, are provisioned as a single unit. Both the sitemap scraper and the website scraper are

Executing Scraper

**The sitemap scraper is executed in 4 steps:**

constructed units in the wezel\_aws\_stack.

|  |  |
| --- | --- |
| lambdas/. | This directory contains all the Lambda function codes. The parser functions and database-related functions are written in Python which is created into lambda functions via aws-lambda-python-alpha .  The scrapers are in Node.js and utilize Puppeteer which requires Docker to create a lambda function. Thus the directories contain a Dockerfile that allows CDK to build an image asset of the scraper to create a lambda function. |

1. scraperMapFn : Step Functions iterator function creates parallel scraping sequences for each competitor.

2. sitemapScraperFn : Node.js Lambda function scrapes the raw sitemap page with Puppeteer and saves it to the S3 bucket as html file. The bucket key is passed as input to the next steps.

240

3. : Python Lambda function retrieves raw sitemap html file from the S3 bucket and parses the sitemap URLs. The result is saved to the S3 bucket as a .csv file.

4. jobDbFn : Python Lambda function populates JobTable DynamoDB with sitemap URLs as the partition key. When initiating each item the status attribute is set to pending.

**How to trigger**

In this step, the function interface selects new execution and enters the input with the following format

1{  
2 "sites": [ 3{

1. 4  "url": "string",
2. 5  "competitor\_name": "string"

6 }, 7{ 8  
9

10 11 12

] }

"url": "string",

"competitor\_name": "string"

}

**The webpage scraper is executed in the following steps:**

sitemapParserFn

241

1.

2. 3.

4. 5.

: The orchestrator function uses the AWS Fargate service to control the number of jobs running at the same time and reassigns failed jobs. Unlike the Lambda function which has a limited runtime of 45min. Fargate functions can stay running and are charged based on the resource used. Since the orchestrator function uses a timer to monitor job status intermittently, the Fargate function is optimized for such a use case. The orchestrator triggers a webpage scraper Step Function for each pending sitemap job from the JobTable.

start(update pending) : Updates job status for the running job from pending to running.

webpageScraperFn : Node.js Lambda function that scrapes raw webpage with Puppeteer and saves to S3 bucket as html file. The bucket key is passed as input to the next steps.

webpageParserFn : Python Lambda function that retrieves raw webpage html file from S3 bucket and parses menu item information. The result is saved DataTable DynamoDB.

update status : When the webpage is scraped successfully, the job status is updated in the JobTable. When the job failed in one of the steps it is also updated in status.

**How to trigger**

In this step, the function interface selects new execution and enters the input with the following format

1{  
2 "sites": 3{

1. 4  "targetId": "string",
2. 5  "url": "string",
3. 6  "competitor\_name": "string"

7} 8}

orchestrator

242

**Relevant Links:**

AWS CDK GitHub location: https://github.com/TomLuoYuehseng/wezel\_aws\_cdk - Connect your Github account

Orchestrator GitHub location: GitHub - TomLuoYuehseng/wezel\_aws\_orchestrator

The developer guide on Getting started with AWS CDK: Getting started with the AWS CDK - AWS Cloud Development Kit (AWS CDK) v 2

Orchestrator guide: Containerizing and running your apps on AWS Fargate Development configuration for future updates

This project is developed with AWS Cloud Development Kit (CDK) which is a tool that lets you construct your cloud infrastructure as code in one of the supported programming languages. In this project, we use Python.

**Prerequisites:** First you need to meet the prerequisites for running AWS CDK by installing Node.js and Python. You also need to install Docker for creating images for lambda functions.

**Configuration:** Then you must configure your workstation with your credentials and AWS regions by running the $ aws configure

command or manually edit the ~/.aws/config and ~/.aws/credentials files.  
**Install AWS CDK:** Install the AWS CDK Toolkit globally using the Node Package Manager command npm **Clone and update:** Clone the CDK project from the repo.  
After cloning to your workstation create a virtualenv with

$ python3 -m venv .venv

Activate your virulent with

$ source .venv/bin/activate

Once the virtualenv is activated, install the required dependencies with

$ install -g aws-cdk

$ pip install -r requirements.txt

243

you can deploy your updates with

$ cdk deploy

244

Monthly Refresh Strategy

Sitemap Checker (Manul, already documented) Integrate parsers into scrapers

save both page sources and parsed data upload to s3 buckets

determine how the job failed → website or proxy global data

frequently jump in and manual check  
document global API call and check monthly  
using Regex to match pattern and pick the last match (as global API call always comes the first)

bad proxy  
check the status of API calls if the status is **200** was successful else it failed and we were blocked

website structure change  
if not 404, and we don’t get data, that means website structure change (scraper is bad) integrate alert  
if its a bad scraper stop running that competitor

Smart Proxy Manager  
track failed proxies so we don't run them again (competitor + location) create a new database to store proxy geolocation  
run proxies through scraper that gets their geolocation  
based on where the store is pick a proxy close to it  
california 4 proxy but 5k stores

Scraper checker  
randomly sample 10 URLs from each competitor and run scraper + parser to check if there’s any website change Incorporate into Wezel

1. Check sitemap
2. Check global API
3. Check website structure

a. if changed, update scraper & parser

1. Start scraping

245

Scout SLAs  
This document outlines the service level agreements for the services provided by the Scout team.

**QA**

QA, including fixing missing data, outdated records, and erroneous entries Case study  
Consulting (regex, data-related subject matter expertise)  
Price movement, one cycle

Guides  
Nielsen data  
User research / roadmap brainstorming, one cycle Dashboard refresh  
New data sources (except Sequentum)

In terms of escalation, Marcia is the first point of contact.

**SLA**

5 days 10 days 3 days 4 days 1 month 3 days 5 days 5 days 3 days

246

P-Card Documentation

Purpose

This project provides visibility for Professional Expense Card (P-Card) spending for all employees in Compass North America to Finance. In addition, this project aims to identify any unapproved spending like any personal, jewelry or tobacco purchases.

Workflow/Architecture diagram

**Lambda Trigger**

An AWS Lambda function that transforms JP Morgan transaction data when it is uploaded into the centralized bucket.

247

**Strategic-Project-Team (SFTP Folder)**

Members from Strategic Project Teams upload 3 files in their respective SFTP folders on a weekly cadence, following are the files:

Expense Type Category File: s3://pcard-reporting/expense\_type\_categories.xls : This file contains the mapping required to transform p-card transactional raw data. The file contains the following sheets:

a. Expense\_Type\_Catagories: All the data in this sheet gets ingested as it is to the datascience\_staging.p\_card\_category\_details table.

b. Merchant logic: All the data in this sheet gets ingested to the datascience\_staging.p\_card\_merchant\_name\_conversion table.

c. Test File Breaks: All the data in this sheet gets ingested as it is to the datascience\_staging.p\_card\_level\_3\_description\_with\_merchant table.

JP Morgan transactions data: s3://pcard-reporting/jp\_morgan/Level\_3/MonthYear\_PN.csv , This file includes all the JP Morgan credit card transactions. This file gets uploaded into the SFTP on a bi-weekly basis, the files are partitioned by months, and all the JPMC data from this folder gets ingested into the datascience\_staging.p\_card\_level\_3 table via a lambda trigger.

Permissions Data for RLS in Power BI: s3://pcard-reporting/permission/permissions.csv . This file contains users' compass emails and the divisions that they have access to. The strategic Project team provides the mapping of Userprincipalname (PowerBI username- compass Emails in this case) to the division keys. This mapping gets ingested into the

datascience\_staging.pbi\_permissions\_v2 table.

**Centralized S3 Bucket**

The Strategic Projects teams' SFTP Folders files get uploaded into the Centralized Bucket. This airflow job takes all the files from the other SFTP folders and drops them into the centralized S3 buckets multiple times a day.

**P-Card-reporting S3 Bucket**

The Airflow job (airflow-lb-200262259.us-east-2.elb.amazonaws.com) takes the data from the centralized s3 bucket and uploads it into the P-Card-reporting S3 Bucket. Additionally, the airflow job transforms data into the cleaned\_data subfolders in the p-card reporting S3 Bucket. Click here for more details.

**Data Warehouse Source Input Tables SAP Concur tables**

An expense management and travel invoice software used by employees in Compass North America to submit their P-Card transactions. All data raw data from P-Card Spend Report comes from Concur

source.cg\_dw\_cdl\_us\_source\_sc\_pcard\_spend\_fact has all the CONCUR data with fewer columns and when filtered for source\_system = 'CONCUR' .

source.cg\_dw\_cdl\_us\_source\_sc\_pcard\_concur\_fact **:** This table has all the CONCUR raw data with all the columns. source.edw\_org\_hierarchy\_dim : This table contains the organizational structure and hierarchy of the entire Compass USA Org when

filtered for erp\_system\_id = 1001 **.** The **Sector, Region, Division, and Cost Center** are all part of this table. source.sap\_hris\_us\_employee\_record : This table contains the details of the Employees in the Compass USA org, such as whether

they are active or inactive, employee ID, First Name and Last Name.

source.cg\_dw\_cdl\_us\_source\_fin\_gl\_acct\_nbr\_dim : This table contains the details about the G/L accounting associated with the Cost centers/ Units to which the P-Card Spend is reported and attributed.

248

**P-Card Summary View Table**

The datascience\_staging.p\_card\_summary\_view table is the culmination of all the source and expense type categories sheets table referred to in the preceding points. This table brings together JPMC/Level 3 financial Item Descriptions (whenever they are available) and Concur Merchant Names for each Concur Transaction ID and other Transaction details obtained from all the source tables referenced above. It is used as the input for the high-risk spend flagging code in Python.

**P-Card Travel Transaction Table**

datascience\_staging.p\_card\_travel\_transaction is the final transformed table. This is the most used table for all the visuals in the Power BI Report.

Two types of transaction flagging occur before this table is built:

**Description Search Term Logic**: Each Concur Transaction’s JPMC/Level 3 Financial Item Descriptions are tested for any REGEX pattern matches between the Qualifiers and REGEX pattern non-matches with the Exclusions. The REGEX patterns for the Qualifiers and Exclusions are provided for each of the High-Risk Categories such as High-Risk Baby, High-Risk Jewelry, High-Risk tobacco in the expense\_type\_categories.xls (Test File Breaks Sheet) file developed and uploaded into SFTP by the SPOT team (Glenn). For example, Item description: ZIONOR H3 Skateboard Helmet for Kid , ZIONOR H3 Skateboard Helmet for Kid will get flagged as High-Risk Baby based on the REGEX Qualifiers and REGEX Exclusions in the (Test File Breaks Sheet).

datascience\_staging.p\_card\_level\_3\_description\_with\_merchant is the final output table containing the flagged reason based on the JPMC financial item descriptions for each of the Concur Transactions, it has description\_id = md5 (a unique JPMC financial Item Description for each of the Concur transaction IDs) and merchant\_id = md5 (Clean Merchant Name), gets flagged due to the financial item descriptions amongst other columns.

**Merchant Search Term Logic**: Each Concur Transaction’s Concur Merchant Name is tested for any REGEX pattern matches between the Qualifiers and REGEX pattern non-matches with the Exclusions (merchant specific exclusions logic is in Development but will be promoted soon to Prod). The REGEX patterns for the Qualifiers and Exclusions are provided for each of the Clean Merchant Names (Parent Merchant Names) such as GOGO AIR, WYNDHAM, and HILTON in the expense\_type\_categories.xls (merchant logic sheet) file developed and uploaded into SFTP by the SPOT team (Glenn). For example, HTTP://WWW.GOGOAIR.COM (Child Merchant Name) gets identified as GOGO FLIGHT (Clean Merchant Name) according to the merchant\_logic sheet in the SPOT file and gets Flagged as Cell Phone - WiFi. DSWA (CENTRAL SOLID WA (Child Merchant Name) gets identified as DSW (Clean Merchant Name) which according to the merchant\_logic sheet in the SPOT file gets Flagged as Apparel +

Shoes. datascience\_staging.p\_card\_merchant\_name\_conversion is the final output table containing the flagged reason based on the cleaned Concur Merchant Name for each of the Concur Transactions, it has clean\_merchant\_name based on the raw Concur Merchant Name and merchant\_id = md5 (Clean Merchant Name), flagged reason based on the clean\_merchant\_name amongst other columns.

The final datascience\_staging.p\_card\_travel\_transaction table is effectively datascience\_staging.p\_card\_summary\_view left joined with the datascience\_staging.p\_card\_level\_3\_description\_with\_merchant and

datascience\_staging.p\_card\_merchant\_name\_conversion . Hence, this table has all the data from the

datascience\_staging.p\_card\_summary\_view table along with the Cleaned Concur Merchant Name and the JPMC financial Item descriptions and their respective Flagged Reasons if any, for each of the Concur Transaction Ids in the P Card Summary view Table.

**P-Card High Risk All Table**

The Merchant Search Term and Description Term logic outputs get unioned into one table called datascience\_staging.p\_card\_high\_risk\_all table, each concur transaction gets flagged either due to its JPMC financial Item

Description or Cleaned Merchant Name into one of the High-Risk categories that they respectively belong to. Hence, this table has the high- risk categories and subcategories for each combination of the description\_id and merchant\_id obtained from the preceding intermediate tables such as the datascience\_staging.p\_card\_level\_3\_description\_with\_merchant and

datascience\_staging.p\_card\_merchant\_name\_conversion .

249

**Power BI Report (P-Card Dashboard)**

The table datascience\_staging.p\_card\_travel\_transaction is the table for almost all of the visuals on the dashboard.

source.edw\_org\_hierarchy\_dim is the second most used table in the dashboard for selecting the Sector, Division and Sector.

datascience\_staging.pbi\_permissions\_v2 is the table used to provide the RLS on the Dashboard.

SPOT AD Group This outlook AD group is used for granting its members viewing and downloading access to the Power BI Report (Also managed by Glenn/ SPOT team).

Granting access to a new Compass employee means them being added into the datascience\_staging.pbi\_permissions\_v2 and to the SPOT AD Group .

**LaserFiche URL**: LaserFiche is another company which was contracted by the SPOT team, for viewing the Receipt images for each Concur Transaction ID. The Detailed Transactions view has the concur transaction id column formatted as URL with the

receipt\_url column's value (made in the datascience\_staging.p\_card\_travel\_transaction as the actual LaserFiche link. Effectively, any user who clicks on a value in the Transaction\_Id column is redirected to the LaserFiche Receipt Image of that Transaction, if the receipt is available and submitted by the employee. Compass SSO manages the security on their side with another set of independent AD groups managed by the SPOT group.

On approximately October 18th, 2022 a change was needed to adjust the url from:

1 https://lf.compass-usa.com/WebLink/Search.aspx?dbid=3 to

1 https://lf.compass-usa.com/WebLink/Search.aspx?repo=Treasury  
In the LaserFiche system logs on only with **@compass-usa** emails. **@compassdigital.io** is not accepted.

250

Terminologies/Definitions

The following terminologies are used frequently in the P-Card (Professional Expense Card) project. Each Terminology is explained in detail and how it’s used in the project:

1. **Level 3 / JP Morgan Feed (JPMC)**: JPMC provides us with the P-Card Transaction's detail reports containing the Item Level Descriptions of all the transactions, they are used as an Input for **Description Search Term Logic** and **Merchant Search Term Logic**(Explained Later).
2. **SAP CONCUR:** An Expense management and travel invoice software used by employees in Compass North America. All the data comes from P-Card Spend Report comes from SAP Concur
3. **Compass Hierarchy:** Our Power BI dashboard, is filtered based on compass hierarchy which includes **Sector, Region, Division, and Cost Center,** these filters are commonly used across all compass dashboards. The sector is the highest level in the hierarchy and one sector can have multiple regions. One region can have multiple divisions and one division can have multiple cost centers. Cost Center is at the bottom of the hierarchy and represents one compass location.
4. **Strategic Projects Team:** They are our main stakeholders who work for Compass USA.
5. **Laserfiche**: A third-party vendor responsible for providing an API for receipt images for the P-Card transaction. Their API is integrated into the PowerBI dashboard.

251

Tools/Applications Required  
To get started on the P-Card project, access to the following tools/applications is required:

**Power BI:** P-Card has its own workspace in Power BI. For access to the workspace and all the following dashboards contact the SPOT team Compass USA.

P-Card Spend Report - Power BI: P-Card Power BI **Prod** Dashboard. This does not have a scheduled refresh. It gets manually

refreshed by the SPOT every week after testing every week’s refreshed UAT dashboard.

P-Card Spend Report\_UAT - Power BI: P-Card Power BI **UAT** Dashboard. It is identical to the Prod dashboard in terms of the layout

and the dataset. This dashboard gets automatically refreshed every Monday morning at 5:00 am with the **Prod Dataset** which is the output of the DAG - http://airflow-lb-200262259.us-east-2.elb.amazonaws.com/tree?dag\_id=run\_pcard\_st\_logic.

P-Card Spend Report\_Dev - Power BI: P-Card Power BI **Dev** Dashboard. All the new changes are tested in this dashboard. The dataset for this dashboard contains the same tables listed on the except that all of them have tests in the end. For example, (in Prod dataset) is

datascience\_staging.p\_card\_travel\_transaction\_test (in **Dev dataset**). The Dev Dataset is the output of the DAG - http://airflow-lb-200262259.us-east-2.elb.amazonaws.com/tree?dag\_id=run\_pcard\_st\_logic\_test.

**Airflow DAGs on the Non-Managed Airflow Instances:**

**Prod** DAG - http://airflow-lb-200262259.us-east-2.elb.amazonaws.com/tree?dag\_id=run\_pcard\_st\_logic. **Dev** DAG - http://airflow-lb-200262259.us-east-2.elb.amazonaws.com/tree?dag\_id=run\_pcard\_st\_logic\_test.

**Files Syncing** DAG - This DAG transfers the files from the centralized S3 bucket to the PCard reporting S3 Bucket - http://airflow-lb- 200262259.us-east-2.elb.amazonaws.com/tree?dag\_id=sync\_pcard\_files.

To get access contact Hari Theniah.

PCard reporting S3 Bucket Contact Hari.Theniah@compassdigital.io or Kris.Vaz@compassdigital.io to get access. cdl-dt-airflow Git Repo - This repo has all the Dags in the folder dags/projects/p\_card and dags/projects/p\_card\_test. Contact dat.tran@compassdigital.io to get access to this repository.

P-Card Documentation

datascience\_staging.p\_card\_travel\_transaction

252

Mobile Reporting Dashboard

Mobile Reporting is an all-purpose Data, Insights, and Business Management dashboard. The information on the dashboard is broken into applications, locations, items, and times. The main objective of this tool is to view Mobile Sales by Sales, Transactions, and Users.

Access to the Dashboard  
To access the dashboard click on this link: Power BI **Or follow these steps:**

1. Navigate to PowerBI: Data Visualization | Microsoft Power BI .  
2. Sign in using Compass credentials.  
3. Click on Workspaces and navigate to the Mobile Reporting Dashboard workspace and open the Mobile Reporting Dashboard.

Select Workspace

253

Home

Let’s explore the Dashboard. On the Mobile Reporting Dashboard, you can perform various tasks and view multiple types of reports. Kindly refer to the training video displayed below for more information:

Training Video

0:00 / 17:54 1x

254

Home Page

The Home Page of the Dashboard displays information like the Sales, New Users, Menu Names, Brand Names, and Transactions that have taken place for that day.

You can also export data from this page. However, there are 2 different export options available on the page.  
The first option is highlighted in red in the screenshot below. Through this option, you can export Summarized data used to create your visual (for example, sums, averages, and medians).  
The second way to export is by clicking on the Export button (highlighted in blue) in the screenshot above.

This Export option lets you export the data through an Excel, PPT, or PDF version. However, this option does not export the summarized data.

255

Export

Schedule Refresh  
The dashboard refreshes once a day daily and it takes approximately 60-80 minutes for the dashboard to show the changes.

256

· Refresh time: 8:00 AM EST/EDT

Filters  
Filters are inside the page or as a separate panel. Hold ctrl to make multiple selections.

257

Filters

You can use the Search function if you’ve applied it inside the filter.  
To clear filters, click on the eraser button beside the filter headers OR at the top right of the panel.

**Filter Definition**

Location Filters:

Sector: Filter on specific Compass Sectors.  
Location: Filter on specific Mobile Locations (the University of Houston, Buffalo State, Comcast). Account: Filter on the specific Accounts (An account can be composed of numerous locations) Unit. Number: Filter on the specific Unit Number.  
Brand Name: Filter on the specific Brands.  
Delivery Area: Filter on specific Delivery Area (only applicable for Delivery Orders).

Time Filters:  
Order Time Range: Filter on ranges of time when an order is placed Pickup. Time Range: Filter on ranges of time when the order is picked up.

Item Filters:  
Menu Item: Filter on specific menu items.

Item Group: Filter on specific Item Groups (item groups consist of multiple items (Hot Beverages, Breakfast Spreads, and Bagel Add- Ons).

Order Type Filters:  
Promo Code: Filter on specific Promo Codes (this filters the report down to just orders that use this promo code). Order Type: Filter on specific orders (Pickup, Delivery, Scan, and Go).  
Tender Type: Filter on orders that have a specific tender type.  
Zero Trans Sales: Filter orders that have $0 as total sales.

258

Pickup ID: Filter on a specific Pickup ID (The same as the order ID used in Admin Panel).

Values on Charts  
You can hover or click over the bar for the values to appear on the Graphs displayed on the Home Page.

To view a chart as a table:  
Select or hover over the chart or table until more options appear on the top right. Select the ellipsis menu to bring up more options.  
Show as a table.

259

**Tables Pivot table**

Expand by selecting the "+" in the left column

260

OR

Select or hover over the chart or table until more options appear on the top right. Use the arrows to either expand or collapse the table.  
The last down arrow will expand the columns further.  
This table can be used to analyze or export data.

Download Data  
On the Mobile Reporting Dashboard, you can download up to 30,000 rows of data. To extract/download data:

Select or hover over the chart or table until more options appear on the top right. Select the ellipsis menu to bring up more options.  
Export data  
Select the file format to export and select the Export option.

Mobile Dashboard

Bookmarks  
Bookmarks capture the state of the report including filters.  
For example, to split the timeline into Breakfast, Lunch, and Dinner follow the steps given below:

Select time in Order Created Time Range. To make multiple selections, hold ctrl and select. Select the timing, 6:00 am – 11 am.  
On the top right of the page, create new bookmarks and name them accordingly.

261

This will save all filters. May need adjustments which can be done by updating the bookmark (look for the ellipsis menu beside the existing bookmark).

262

Reports & Timeslot Analysis

**Report**

The Report page on the Dashboard lets you generate various types of reports as shown in the screenshot below:

Upon landing on the Report page, select the filters and then click on View report on the right.

The first report type is the Product Mix Report . It is a pivot table to expand the levels of detail to show sales and transactions by item. For example, in the screenshot below we have generated a Product Mix Report by selecting the required filters.

263

Product Mix Report

The second type of report is the Order Level Report . It is an order level report to show sales at the transaction level For example, in the screenshot below we have generated an Order Level Report by selecting the required filters.

Order Level Report

264

The third type of report is the Sales Detail Report . It is an item-level report to show detailed information about the transaction. For example, in the screenshot below we have generated a Sales Detail Report by selecting the required filters.

Timeslot Analysis

Sales Detail Report

265

Time-slot Occupancy table: The percentage of the allocated slot time is filled by the number of customers.

calculation = average orders/ total number of time-slot

Late Transaction % by the hour of day table: Calculation using the time when order is set to pick up vs the time when it is finished. Any order that is completed after the pickup time is considered as late.

Peak Time Analysis

Peak Time Analysis is an hourly view of a sale. This option is used to analyze the trend and peak time and to see a percentage of late transactions (only applies to p2)

Peak Time Analysis

Peak Time chart - traffic hours to show transactions and late transactions % by the ordered hour or pickup hour (toggle using the arrows )

Metrics

**Name**

Net Sales  
Gross Sales  
Transaction Count Average Cheque/Checks Active Users  
New Users  
Register Order Total Conversion Rate

**Formula**

Item sales (after discounts) , before tax  
Item sales (before discounts) + promo amount, before tax  
Count of orders  
(Item sales (after discounts) before tax)/ Quantity  
Number of users who made purchase during the selected time range Number of users who made first purchase during the selected time range Item sales (after discounts) + Tax Amount  
Buying users/Registered users

266

Time-slot Occupancy Time Fill Mobile vs POS %  
Week Over Week MonthOverMonth

Average orders/ total number of time-slot offered Gross Sales/ (POS Sales + Gross Sales)  
(Current Week - Previous Week)/ Previous Week (CurrentMonth PreviousMonth)/PreviousMonth

267

Mobile vs POS

This page displays Sales data from Mobile and POS systems. POS is a Point of Sales system used by various vendors to track sales of their products.

Here we are comparing the Mobile Penetration to the POS sites that do have a mobile system set up.

Calculation: Mobile Gross Sales/ (POS Sales + Mobile Gross Sales)

Mobile and POS Sales by Day chart - Timestamp of Mobile sales and POS sales  
Mobile and POS Transactions by Day chart - Timestamp of Mobile sales and POS transactions Mobile Penetration = Mobile sales compare to the total sales (POS + Mobile)

**Name**

Average Cheque/Checks Mobile vs POS %

**Formula**

(Item sales (after discounts) before tax)/ Quantity Gross Sales / (POS Sales + Gross Sales)

Mobile vs POS

268

Promo Usage

This page is dedicated to transactions that have a promo code applied. We can also see the sales difference between non-promo transactions and promo transactions.

Sales and Promo Sales by Day chart- Transactions with promo and non-promo by day. Promo Code Used By Time Of Day chart - Transactions with a promo by the hour. Transactions by Promo Code chart - Bar chart of the top used promo code by transactions.

Promo Code Usage

269

Week Over Week

This page is dedicated to showing the currently selected week to the previously selected week comparison.  
Depending on the location of the sites, the US Fiscal Week and CA Fiscal Week will differ and need to be selected accordingly. US Fiscal Week begins on a Friday of the week and CA Fiscal Week will begin on a Thursday of the week.

The buttons on the top right will switch the filter on the “Select Fiscal [country] Week”. Mandatory to have a Fiscal US/CA Week selected for the calculation to work.

Calculation = (Current Week - Previous Week)/ Previous Week

Week Over Week

270

Month Over Month

This page is dedicated to showing the currently selected period to the previously selected period comparison. Please note that the Compass Fiscal calendar starts in October.

Oct =1 Nov =2 Dec =3 Jan =4 Feb=5 Mar=6 Apr=7 May=8 Jun=9 Jul=10 Aug=11 Sep=12

Mandatory to have a Fiscal Period selected for the calculation to work.

Calculation = (Current Month- Previous Month)/ Previous Month

Month Over Month

271

Quarter Over Quarter

This page is dedicated to showing the currently selected period to the previously selected period comparison. Please note that the Compass Fiscal calendar starts in October.

Oct -Dec=1 Jan-Mar=2 Apr-Jun=3 Jul-Sep=4

Mandatory to have a Fiscal Quarter selected for the calculation to work.

Calculation = (Current Quarter- Previous Quarter)/ Previous Quarter

272

Menu Item Insights

This page is dedicated to menu item sales which can toggle between viewing the Net Sales or Total Items Sold(quantity). We can see the the menu sales and item sold(quantity) split out by order type and item menu and modifiers.

Menu Item Sales by Order Type chart- Sales by order type.  
Total Item Orders and Item Sales chart - Sales and Item Orders(quantity) by main item menus and modifier.

273

Sagemaker Documents

**Title**

Data Science and Modeling

Operations and Administration

TEMP Spot for information that can be slotted in where appropriate. Note Book Instance Resource:

Create a Notebook Instance - Amazon SageMaker

Data Wrangler Sample and Flow:

GitHub - aws-samples/schedule-sagemaker-data-wrangler-flow

Long Format SageMaker Deep Dive:

**Creator**

Eden Trainor

Eden Trainor

**Modified**

Apr 13, 2022 Apr 13, 2022

AWS re:Invent 2019: Amazon SageMaker deep dive: A modular solution for machine learning (AIM307)

274

Data Science and Modeling  
These pages will provide help and guidance with various aspects of SageMaker and data science work.

**Title**

Notebooks: Getting Started Guide  
Notebooks: Storing and Retrieving Secret Values

**Creator**

Eden Trainor Eden Trainor

**Modified**

Oct 18, 2022 Aug 15, 2022

275

Notebooks: Getting Started Guide

This guide will show you how to get started working on notebooks in AWS SageMaker including getting access to SageMaker, making your User that will allow you to separate your work, and opening a notebook.

Getting Access  
Before you start you will need to have a few things set up for you:

1. Make sure you have access to AWS through your My Apps OR an AWS User that you can use to access the AWS console. a. If you don’t, ask your data team lead to request one for you.

2. Ask an ML Engineer if your user is part of the “Sagemaker-access” IAM group (Eden Trainor, Aditya Joshi, Robert Inkpen) 3. Make sure you can see the shared GitHub repo for data science, if not ask an ML engineer to give your GitHub user access

Adding A User  
Your SageMaker user will be what links together all your work together and separates it from your colleagues.

1. Go to AWS Sagemaker
2. Select **Open SageMaker Domain** in the top right of the page. You are now in the SageMaker domain which is where you will go to do

all your SageMaker work.

1. Select **Add user** in the top right of the *Users* panel
2. Type your user name in the **Name** box, and leave all the other fields as their default values.
3. After pressing submit on the third page you should be able to see your name in the list of users.

Make sure you’re assigned as an admin to access Studio. Else you will receive an error when you log in.

**Opening Studio**

SageMaker studio is like Jupyter notebooks on crack. The studio is where you’ll do most of your work.

1. Whilst in the SageMaker domain, in the *User* panel, select **Launch App** on the row with your user name 2. Select **Studio** on the drop-down (this may take a minute on a cold start)

a. This will open SageMaker studio as a Jupyter notebook  
b. You will be presented with the *launcher* page which contains a lot of options for starting workflows

**Importing Data Science Repo**

1. Select the **Git Symbol** on the panel on the far left of the page  
2. Select **Clone A Repository**3. Enter the **cloning link** for the repo: *https://github.com/compassdigital/ds-sagemaker.git*

4. Enter your **GitHub username** and a ***personal access token***a. If you don’t have a personal access token you can make one by following this guide (SAVE THIS SOMEWHERE SECURE)

276

**Opening a New Notebook**1. On the *launcher,* page scroll down to the section **Notebooks and compute resource**

2. Select **Notebook (Python 3)**a. Selecting different SageMaker images (just above the notebook button) will give you access to environments with different tools built

in. Data Science is good for most applications.  
3. Select your *instance to compute size* by selecting **Unknown** or “**x vCPU + y GiB”** where x/y are numbered.

a. *ml.t3.medium* is a good place to start **Working in the Notebook Environment**

N.B → Only files and data saved within the /home/ec2-user/SageMaker folder persist between notebook instance sessions. Files and data that are saved outside this directory are overwritten when the notebook instance\* stops and restarts.

\*Notebook Instance

Each notebook instance's /tmp directory provides a minimum of 10 GB of storage in an instance store. An instance store is temporary, block-level storage that isn't persistent. When the instance is stopped or restarted, SageMaker deletes the directory's contents. This temporary storage is part of the root volume of the notebook instance.

**Creating a virtual environment in the Notebook Environment**

Virtual environments help keep dependencies required by different projects separate by creating isolated python virtual environments. SageMaker Studio Lab environment comes with a base image installed using Anaconda that includes key packages and resources. We suggest creating a virtual environment created using Anaconda and then installing packages pip and then downloading packages using pip. Listed are the steps required to create virtual environments in SageMaker

1. Open the Image Terminal in the SageMaker Launcher  
a. There are many terminals in the SageMaker Launcher and only the Image Terminal can be used to create a virtual environment

2. Run this command on the terminal to create the virtual environment conda create -n <NAME\_OF\_VIRTUAL\_ENVIRONMENT> 3. Activate the virtual environment conda activate <NAME\_OF\_VIRTUAL\_ENVIRONMENT>  
4. Install pip with Anaconda conda install pip

5. Install ipykernel so that JupyterNotebook can select a virtual environment as a kernel. pip install ipykernel a. use pip to install packages as needed for your project  
b. If you have packages that require gcc we suggest you follow these guidelines to install it in your EC2 instance.

nd gcc failed with exit status 1” Error and Solution

6. Open JupyterNotebook and select notebook environment

The “error comma

277

7. Select your virtual environment in the Kernel Dropdown (this may take a couple of minutes to show up)

8. Build your models in that virtual environment.  
Follow this guide if you have further questions Manage your environment - Amazon SageMaker .

If you run into an issue please feel free to reach out to @Eden Trainor , @Adi or @Robert.Inkpen

**Querying Databases**

Querying from Redshift, or any database is the same as with a local Jupyter Notebook. A useful snippet, using psycopg2 to query Redshift, is posted below:

1. 1  **def** run\_query(query):
2. 2  **with** psycopg2.connect(\*\*redshift\_params) **as** conn:
3. 3  **with** conn.cursor() **as** cur:
4. 4  cur.execute(query)
5. 5  res = cur.fetchall()
6. 6  df = pd.DataFrame(res)
7. 7  **if** df.shape[0]>0:
8. 8  df.columns = [i[0] **for** i **in** cur.description]
9. 9  **return** df

10  
11 df = run\_query(df)

**Working With Github on Sagemaker**

Cloning GitHub repository into:

Clone a Git Repository in SageMaker Studio - Amazon SageMaker

Pushing/pulling (saving work) to a git repository can be done in the ‘git’ tab on the sidebar of sagemaker studio.  
The top of the tab display has two images of clouds one with an up arrow and one with a down arrow for pull and push respectively.:

278

In the tab displayed in the above image, you will be able to see your changes (staged/changed/un-tracked). You can switch branches by clicking on their names and creating branches using the “new branch” button.

Committing changes and pushing them can be done by scrolling down, adding a commit message, and then clicking on the commit button shown in the image below:

279

280

Notebooks: Storing and Retrieving Secret Values  
This document assists in how to store your database passwords and other sensitive information for use in your notebooks. For this, we will

use a service called AWS Secrets Manager. Storage

1. Get permissions for your AWS user to access Sagemaker Secrets on the secrets manager. a. This gives you access to any secret whose name starts with “sagemaker”

2. Store your passwords  
a. Click the link above to go to Secrets Manager  
b. Follow this tutorial for storing a secret  
c. For basic password storage, you can store the secret as a plain text value shown in the notes below  
d. Prefix your secret name with “sagemaker”, e.g. sagemaker-redshift-password or sagemaker\_github\_token .

Retrieval

Use the AWS SDK boto3 package in a python notebook to retrieve your secret. Please feel free to use this minimal code snippet to help along:

1  
2  
3  
4  
5  
6  
7  
8)  
9 **return** get\_secret\_value\_response['SecretString']

**from** boto3.session **import** Session

**def** get\_secret\_value(secret\_name: str) -> str:  
secret\_manager = Session().client("secretsmanager") get\_secret\_value\_response = secret\_manager.get\_secret\_value(

SecretId=secret\_name

This will retrieve the secret as a string and will need to be converted into a dict if the secret is in JSON form:

1 **import** json  
2  
3 secret\_string = get\_secret\_value("sagemaker-my-secret") 4 secret\_dict = json.loads(secret\_string)

To store a secret as plain text:

1. Secret Type => “Other type of secret”  
2. Store as “Plaintext”  
3. Remove empty JSON and insert secret text

281

If you need help with any of the steps please don’t hesitate to reach out to Eden Trainor.

282

Notebooks: Adding Python Libraries

283

Operations and Administration

**Title Creator Modified**

Permissions: Adding a IAM User To Sagemaker Group Eden Trainor Mar 30, 2022

284

Permissions: Adding a IAM User To Sagemaker Group

285

Everything that's written before 2022

286

Loss Prevention  
A location to store all of the information relating the the Data Intelligence Loss Prevention project.

287

Word Documents with Valuable Summary Information

|  |  |  |
| --- | --- | --- |
| **Dashboard Doc...on.docx**  28 Apr 2021, 10:40 PM | **Loss Preventio... 115.doc**  28 Apr 2021, 10:41 PM | **Loss Preventio... nd.docx**  28 Apr 2021, 10:40 PM |

288

Automated Email Reports Documentations for automated email reporting

289

Boost KPI for Georg, William <William.Georg@compassdigital.io>

We’re sending the following file to William on a monthly basis Query:

1 **With** Boost\_Location 2 **AS**3(

1. 4  **SELECT DISTINCT**
2. 5  JDEGROUPNUMBER,
3. 6  unitid,
4. 7  locationid,
5. 8  locationname,
6. 9  country,
7. 10  state,
8. 11  group\_name
9. 12  **FROM** datamart.p2\_location\_brands
10. 13  **WHERE** appname = 'Boost'
11. 14  ),
12. 15  Calendar
13. 16  **As**
14. 17  (
15. 18  **SELECT**
16. 19  P.CALENDAR\_DATE **AS** START\_DATE,
17. 20  C.CALENDAR\_DATE **AS** END\_DATE
18. 21  **FROM** static\_dimensions.fiscal\_calendar C
19. 22  **INNER JOIN** static\_dimensions.fiscal\_calendar P
20. 23  **ON** DATEADD(**day**, -30, '2019-12-01' - 1) = P.calendar\_date
21. 24  **WHERE** C.calendar\_date = '2019-12-01' - 1

25

1. 26  ),
2. 27  Total\_Register\_users
3. 28  **AS**
4. 29  (
5. 30  **SELECT**
6. 31  B.UNITID,
7. 32  COUNT(**DISTINCT** CUSTOMER\_ID) **AS** Total\_Register\_Users
8. 33  **from** datastore.mobile\_registered\_users M
9. 34  **INNER JOIN** Boost\_Location B
10. 35  **ON** M.JDEGROUPNUMBER = B.JDEGROUPNUMBER
11. 36  **GROUP BY** B.UNITID
12. 37  ),
13. 38  Total\_Register\_users\_App
14. 39  **AS**
15. 40  (
16. 41  **SELECT**
17. 42  Appname,
18. 43  COUNT(**DISTINCT** CUSTOMER\_ID) **AS** Total\_Register\_Users
19. 44  **from** datastore.mobile\_registered\_users M
20. 45  **GROUP BY** Appname
21. 46  ),
22. 47  FirstOrderByCustomer
23. 48  **AS**
24. 49  (

50

**SELECT**

290

|  |  |  |
| --- | --- | --- |
|  | 51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  customer\_id,  MIN(createddate) **as** first\_order **FROM** datamart.p2\_order\_item\_options **WHERE** grain\_lvl = 'order' **GROUP BY** customer\_id  ), ActiveCustomer **AS** (  **SELECT**  unitid,  COUNT(**DISTINCT** customer\_id) **as** Active\_Users **FROM** datamart.p2\_order\_item\_options o  **INNER JOIN** Calendar C **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE  **WHERE** istestorder = 0  **GROUP BY** unitid ),  GroupActiveUsers  **AS**  (  **GROUP BY** jdegroupnumber ),  AppLevelActiveUsers  **AS**  (  **GROUP BY** appname ),  DateRegistered  **AS**  (  **SELECT**  jdegroupnumber,  COUNT(**DISTINCT** customer\_id) **as** Active\_Users **FROM** datamart.p2\_order\_item\_options o  **INNER JOIN** Calendar C **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE  **WHERE** istestorder = 0  **SELECT**  appname,  COUNT(**DISTINCT** customer\_id) **as** Active\_Users **FROM** datamart.p2\_order\_item\_options o  **INNER JOIN** Calendar C **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE  **WHERE** istestorder = 0  **SELECT**  customer\_id, email\_id, MIN(register\_date) **as** register\_date **FROM** (  **SELECT**  customer\_id, email\_id, date\_created\_utc **as** register\_date  **FROM** datamart.p2\_users u **INNER JOIN** Calendar C  **ON** u.date\_created\_utc::**date** BETWEEN C.START\_DATE AND C.END\_DATE |  |

291

|  |  |  |
| --- | --- | --- |
|  | 109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  **WHERE** source\_ = 'boost' )a  **GROUP BY**  customer\_id,  email\_id ),  BuyingCustomer  **AS**  (  **SELECT**  o.unitid,  o.orderid,  o.customer\_id,  o.createddate\_local,  o.ordernetsales  **FROM** datamart.p2\_order\_item\_options o **INNER JOIN** Calendar C  **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE **WHERE** grain\_lvl = 'order'  AND o.customer\_id IN (**SELECT** customer\_id **FROM** DateRegistered) AND istestorder = 0  ), TotalQuantityPerUnit **AS** (  **SELECT**  unitid,  SUM(itemquantity) **as** quantity **FROM** datamart.p2\_order\_item\_options o  **INNER JOIN** Calendar C **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE  **WHERE** grain\_lvl = 'item'  **GROUP BY** unitid ),  TotalSalesAndTxnPerUnit  **AS**  (  **SELECT**  o.unitid, COUNT(**DISTINCT** o.orderid) **AS** transaction\_count, SUM(o.ordernetsales) **AS** Total\_Sales, tx.quantity  **FROM** datamart.p2\_order\_item\_options o **INNER JOIN** Calendar C  **ON** o.createddate\_local BETWEEN C.START\_DATE AND C.END\_DATE **LEFT JOIN** TotalQuantityPerUnit Tx  **ON** o.unitid = tx.unitid **WHERE** grain\_lvl = 'order'  AND istestorder = 0 **GROUP BY**  o.unitid,  tx.quantity  ), Time\_To\_Order\_RawData **AS** (  **SELECT**  o.unitnumber, |  |

292

|  |  |  |
| --- | --- | --- |
|  | 167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  o.customerid **as** email, D.register\_date, min(o.ordertime\_utc) **as** first\_order\_date, datediff(**hour**, register\_date, min(ordertime\_utc)) **AS** time\_to\_order, **CASE WHEN** datediff(months, register\_date, min(ordertime\_utc)) = 1 **THEN**  1  **ELSE**  0  **END as** order\_in\_first\_month **FROM** datastore.mobile\_orders o  **INNER JOIN** DateRegistered D **ON** o.customerid = D.email\_id AND o.appname = 'Boost' AND o.grain\_lvl = 'order' AND o.istestorder = 0  **GROUP BY**  o.unitnumber,  o.customerid,  D.register\_date  ), TTO **AS** (  **SELECT**  unitnumber, SUM(order\_in\_first\_month) **as** OrderInFirstMonth, avg(time\_to\_order) **as** TTO\_Hours  **FROM** Time\_To\_Order\_RawData  **GROUP BY** unitnumber ),  RecurringUser1Day  **As**  (  (  **SELECT**  unitnumber,  COUNT(**DISTINCT** customerid) **as** UserCount **FROM**  **SELECT**  **DISTINCT**  o.customerid, o.unitnumber, date\_trunc('day', o.ordertime\_utc), COUNT(**DISTINCT** o.customerid) **as** orderCount  **FROM** datastore.mobile\_orders o **INNER JOIN** Calendar C  **ON** o.orderdate BETWEEN C.START\_DATE AND C.END\_DATE **where** grain\_lvl = 'order'  AND appname = 'Boost'  AND istestorder = 0 **GROUP BY**  customerid,  unitnumber,  date\_trunc('day', ordertime\_utc)  **HAVING** COUNT(**DISTINCT** orderid) > 1 )  **GROUP BY** unitnumber |  |

293

|  |  |  |
| --- | --- | --- |
|  | 225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  ), RecurringUser7Day **As** (  **SELECT**  unitnumber,  COUNT(**DISTINCT** customerid) **as** UserCount **FROM**  (  **SELECT**  **DISTINCT**  customerid,  unitnumber,  date\_trunc('week', ordertime\_utc),  COUNT(**DISTINCT** customerid) **FROM** datastore.mobile\_orders o  **INNER JOIN** Calendar C **ON** o.orderdate BETWEEN  **where** grain\_lvl = 'order' AND appname = 'Boost' AND istestorder = 0  **GROUP BY**  **as** orderCount  C.START\_DATE AND C.END\_DATE  customerid,  unitnumber,  date\_trunc('week', ordertime\_utc)  **HAVING** COUNT(**DISTINCT** orderid) > 1 )  **GROUP BY** unitnumber ),  RecurringUser30Day  **As**  (  (  **SELECT**  unitnumber,  COUNT(**DISTINCT** customerid) **as** UserCount **FROM**  **SELECT**  **DISTINCT**  customerid,  unitnumber,  date\_trunc('month', ordertime\_utc),  COUNT(**DISTINCT** customerid) **FROM** datastore.mobile\_orders o  **INNER JOIN** Calendar C **ON** o.orderdate BETWEEN  **where** grain\_lvl = 'order' AND appname = 'Boost' AND istestorder = 0  **GROUP BY**  **as** orderCount  C.START\_DATE AND C.END\_DATE  customerid,  unitnumber,  date\_trunc('month', ordertime\_utc)  **HAVING** COUNT(**DISTINCT** orderid) > 1 )  **GROUP BY** unitnumber ),  DailyActiveUsers |  |

294

|  |  |  |
| --- | --- | --- |
|  | 283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  **AS**  (  (  **SELECT**  unitnumber,  AVG(ActiveUsers) **as** ActiveUsers **FROM**  **SELECT**  unitnumber, orderdate, COUNT(**DISTINCT** customerid) **as** ActiveUsers  **FROM** datastore.mobile\_orders o **INNER JOIN** Calendar C  **ON** o.orderdate BETWEEN C.START\_DATE AND C.END\_DATE **WHERE** grain\_lvl = 'order'  AND appname = 'Boost'  AND istestorder = 0 **GROUP BY**  unitnumber,  orderdate )a  **GROUP BY** unitnumber ),  MonthlyActiveUsers  **AS**  (  (  **SELECT**  unitnumber,  AVG(ActiveUsers) **as** ActiveUsers **FROM**  **SELECT**  unitnumber, date\_trunc('month', orderdate), COUNT(**DISTINCT** customerid) **as** ActiveUsers  **FROM** datastore.mobile\_orders **WHERE** grain\_lvl = 'order'  AND appname = 'Boost'  AND istestorder = 0 **GROUP BY**  unitnumber,  date\_trunc('month', orderdate)  **GROUP BY** unitnumber ),  ChurnUsers  **AS**  (  (  )a  **SELECT**  unitnumber,  COUNT(**DISTINCT** customerid) **as** ChurnUsers **FROM**  **select**  unitnumber,  customerid,  min(orderdate),  max(orderdate), |  |

295

|  |  |  |
| --- | --- | --- |
|  | 341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  datediff('week', min(orderdate), max(orderdate)) **as** datedifference **from** datastore.mobile\_orders **Where** grain\_lvl = 'order'  AND appname = 'Boost'  AND istestorder = 0 **GROUP BY**  customerid,  unitnumber **Having** datediff('week', min(orderdate), max(orderdate)) >= 8  )a  **GROUP BY** unitnumber ),  UnitLevelData  **AS**  (  **select** M.JDEGROUPNUMBER::**VARCHAR**(30) **AS** JDEGROUPNUMBER, B.unitid::**VARCHAR**(30) **AS** unitid, B.locationname, COUNT(**DISTINCT** M.customer\_id) **AS** New\_Registered\_Users, (COUNT(**DISTINCT** M.customer\_id)::**FLOAT**/TR.Total\_Register\_Users::**FLOAT**) \* 100 **AS** Adoption\_Rate, TR.Total\_Register\_Users **AS** Total\_Register\_Users, AC.Active\_Users, COUNT(**DISTINCT** BC.customer\_id) **AS** Newly\_Converted\_Customer, **CASE WHEN** AC.Active\_Users > 0 **THEN**  TS.transaction\_count::**FLoat**/AC.Active\_Users::**Float ELSE**  0  **END AS** TPAU, (COUNT(**DISTINCT** BC.customer\_id)::**Float**/COUNT(**DISTINCT** M.customer\_id)::**Float**) \* 100 **AS** Puchase\_Conv **COALESCE**(TTO.TTO\_Hours, 0) **as** Time\_To\_Orders\_Hours, **COALESCE**(TTO.OrderInFirstMonth, 0) **AS** User\_Order\_In\_First\_Month, **COALESCE**(R1.UserCount, 0) **as** OneDayReturningUsers, **COALESCE**(R7.UserCount, 0) **as** SevenDayReturningUsers, **COALESCE**(R30.UserCount, 0) **as** MonthDayReturningUsers, **COALESCE**(DAU.ActiveUsers, 0) **as** DailyActiveUsers, **COALESCE**(MAU.ActiveUsers, 0) **as** MonthlyActiveUsers, **CASE WHEN** MAU.ActiveUsers > 0 **THEN**  (DAU.ActiveUsers::**Float**/MAU.ActiveUsers::**Float**) \* 100 **ELSE**  0  **END AS** Stickiness, **COALESCE**(CU.ChurnUsers, 0) **as** Churn\_Users, TS.Total\_Sales **as** TotalSales, **CASE WHEN** MAU.ActiveUsers > 0 **THEN**  TS.Total\_Sales/AC.Active\_Users  **ELSE**  0  **END AS** ARPU, **CASE WHEN** TS.transaction\_count > 0 **THEN**  TS.Total\_Sales/TS.transaction\_count  **ELSE**  0  **END AS** Average\_Cheq, |  |

296

e

|  |  |  |
| --- | --- | --- |
|  | 399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  **CASE WHEN** TS.transaction\_count > 0 **THEN**  TS.quantity::**float**/TS.transaction\_count::**float ELSE**  0  **END AS** AverageBasketSize **from** datastore.mobile\_registered\_users M  **INNER JOIN** Boost\_Location B **ON** M.JDEGROUPNUMBER = B.JDEGROUPNUMBER  **INNER JOIN** Calendar C **ON** M.registered\_date::**date** BETWEEN C.START\_DATE AND C.END\_DATE  **LEFT JOIN** Total\_Register\_users TR **ON** B.UNITID = TR.UNITID  **LEFT JOIN** BuyingCustomer BC **ON** B.unitid = BC.unitid AND M.customer\_id = BC.customer\_id  **LEFT JOIN** ActiveCustomer AC **ON** B.unitid = AC.unitid  **LEFT JOIN** TTO **ON** B.unitid = TTO.unitnumber  **LEFT JOIN** RecurringUser1Day R1 **ON** R1.unitnumber = B.unitid  **LEFT JOIN** RecurringUser7Day R7 **ON** R7.unitnumber = B.unitid  **LEFT JOIN** RecurringUser30Day R30 **ON** R30.unitnumber = B.unitid  **LEFT JOIN** DailyActiveUsers DAU **ON** DAU.unitnumber = B.unitid **LEFT JOIN** MonthlyActiveUsers MAU **ON** MAU.unitnumber = B.unitid  **LEFT JOIN** ChurnUsers CU **ON** CU.unitnumber = B.unitid  **LEFT JOIN** TotalSalesAndTxnPerUnit TS **ON** B.unitid = TS.unitid  **GROUP BY**  M.JDEGROUPNUMBER, B.unitid, B.locationname, TR.Total\_Register\_Users, AC.Active\_Users, **COALESCE**(TTO.TTO\_Hours, 0), **COALESCE**(TTO.OrderInFirstMonth, 0), **COALESCE**(R1.UserCount, 0), **COALESCE**(R7.UserCount, 0), **COALESCE**(R30.UserCount, 0), **COALESCE**(DAU.ActiveUsers, 0), **COALESCE**(MAU.ActiveUsers, 0), **COALESCE**(CU.ChurnUsers, 0), MAU.ActiveUsers,  DAU.ActiveUsers,  TS.Total\_Sales,  TS.Transaction\_Count,  TS.quantity  ), GroupLevelData **AS** (  **select** |  |

297

|  |  |  |
| --- | --- | --- |
|  | 457  458  459  460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514  B.jdegroupnumber::**VARCHAR**(30) **AS** jdegroupnumber, 'Group Level' **AS** unitid, B.group\_name::**VARCHAR**(30) **AS** group\_name, COUNT(**DISTINCT** M.customer\_id) **AS** New\_Registered\_Users, (COUNT(**DISTINCT** M.customer\_id)::**FLOAT**/TR.Total\_Register\_Users::**FLOAT**) \* 100 **AS** Adoption\_Rate, TR.Total\_Register\_Users **AS** Total\_Register\_Users,  AC.Active\_Users, COUNT(**DISTINCT** BC.customer\_id) **AS** Newly\_Converted\_Customer, **CASE WHEN** AC.Active\_Users > 0 **THEN**  SUM(**DISTINCT** TS.transaction\_count)::**FLoat**/AC.Active\_Users::**Float ELSE**  0  **END AS** TPAU, (COUNT(**DISTINCT** BC.customer\_id)::**Float**/COUNT(**DISTINCT** M.customer\_id)::**Float**) \* 100 **AS** Puchase\_Conv **COALESCE**(AVG(**DISTINCT** TTO.TTO\_Hours), 0) **as** Time\_To\_Orders\_Hours, **COALESCE**(SUM(**DISTINCT** TTO.OrderInFirstMonth), 0) **AS** User\_Order\_In\_First\_Month, **COALESCE**(SUM(**DISTINCT** R1.UserCount), 0) **as** OneDayReturningUsers, **COALESCE**(SUM(**DISTINCT** R7.UserCount), 0) **as** SevenDayReturningUsers, **COALESCE**(SUM(**DISTINCT** R30.UserCount), 0) **as** MonthDayReturningUsers, **COALESCE**(SUM(**DISTINCT** DAU.ActiveUsers), 0) **as** DailyActiveUsers, **COALESCE**(SUM(**DISTINCT** MAU.ActiveUsers), 0) **as** MonthlyActiveUsers, **CASE WHEN** SUM(**DISTINCT** MAU.ActiveUsers) > 0 **THEN**  (SUM(**DISTINCT** DAU.ActiveUsers)::**Float**/SUM(**DISTINCT** MAU.ActiveUsers)::**Float**) \* 100 **ELSE**  0  **END AS** Stickiness, **COALESCE**(SUM(**DISTINCT** CU.ChurnUsers), 0) **as** Churn\_Users, SUM(**DISTINCT** TS.Total\_Sales) **as** TotalSales, **CASE WHEN** SUM(**DISTINCT** MAU.ActiveUsers) > 0 **THEN**  SUM(**DISTINCT** TS.Total\_Sales)/AC.Active\_Users **ELSE**  0  **END AS** ARPU, **CASE WHEN** SUM(**DISTINCT** TS.transaction\_count) > 0 **THEN**  SUM(**DISTINCT** TS.Total\_Sales)::**FLOAT**/SUM(**DISTINCT** TS.transaction\_count)::**FLOAT ELSE**  0  **END AS** Average\_Cheq, **CASE WHEN** SUM(**DISTINCT** TS.transaction\_count) > 0 **THEN**  SUM(**DISTINCT** TS.quantity)::**float**/SUM(**DISTINCT** TS.transaction\_count)::**float ELSE**  0  **END AS** AverageBasketSize **from** datastore.mobile\_registered\_users M  **INNER JOIN** Boost\_Location B **ON** M.JDEGROUPNUMBER = B.JDEGROUPNUMBER  **INNER JOIN** Calendar C **ON** M.registered\_date::**date** BETWEEN C.START\_DATE AND C.END\_DATE  **LEFT JOIN** Total\_Register\_users TR **ON** B.UNITID = TR.UNITID  **LEFT JOIN** BuyingCustomer BC **ON** B.unitid = BC.unitid AND M.customer\_id = BC.customer\_id |  |

298

e

|  |  |  |
| --- | --- | --- |
|  | 515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  **LEFT JOIN** GroupActiveUsers AC **ON** B.JDEGROUPNUMBER = AC.JDEGROUPNUMBER  **LEFT JOIN** TTO **ON** B.unitid = TTO.unitnumber  **LEFT JOIN** RecurringUser1Day R1 **ON** R1.unitnumber = B.unitid  **LEFT JOIN** RecurringUser7Day R7 **ON** R7.unitnumber = B.unitid  **LEFT JOIN** RecurringUser30Day R30 **ON** R30.unitnumber = B.unitid  **LEFT JOIN** DailyActiveUsers DAU **ON** DAU.unitnumber = B.unitid **LEFT JOIN** MonthlyActiveUsers MAU **ON** MAU.unitnumber = B.unitid  **LEFT JOIN** ChurnUsers CU **ON** CU.unitnumber = B.unitid  **LEFT JOIN** TotalSalesAndTxnPerUnit TS **ON** B.unitid = TS.unitid  **GROUP BY**  B.jdegroupnumber,  B.group\_name,  TR.Total\_Register\_Users,  AC.Active\_Users  */\* MAU.ActiveUsers,*  *DAU.ActiveUsers,*  *TS.Total\_Sales,*  *TS.Transaction\_Count,*  *TS.quantity \*/*  ), AppLevelData **AS** (  **select** 'Boost' **AS** jdegroupnumber, 'App Level' **AS** unitid, 'App Level' **AS** group\_name, COUNT(**DISTINCT** M.customer\_id) **AS** New\_Registered\_Users, (COUNT(**DISTINCT** M.customer\_id)::**FLOAT**/TR.Total\_Register\_Users::**FLOAT**) \* 100 **AS** Adoption\_Rate, TR.Total\_Register\_Users **AS** Total\_Register\_Users, AC.Active\_Users, COUNT(**DISTINCT** BC.customer\_id) **AS** Newly\_Converted\_Customer, **CASE WHEN** AC.Active\_Users > 0 **THEN**  SUM(**DISTINCT** TS.transaction\_count)::**FLoat**/AC.Active\_Users::**Float ELSE**  0  **END AS** TPAU, (COUNT(**DISTINCT** BC.customer\_id)::**Float**/COUNT(**DISTINCT** M.customer\_id)::**Float**) \* 100 **AS** Puchase\_Conv **COALESCE**(AVG(**DISTINCT** TTO.TTO\_Hours), 0) **as** Time\_To\_Orders\_Hours, **COALESCE**(SUM(**DISTINCT** TTO.OrderInFirstMonth), 0) **AS** User\_Order\_In\_First\_Month, **COALESCE**(SUM(**DISTINCT** R1.UserCount), 0) **as** OneDayReturningUsers, **COALESCE**(SUM(**DISTINCT** R7.UserCount), 0) **as** SevenDayReturningUsers, **COALESCE**(SUM(**DISTINCT** R30.UserCount), 0) **as** MonthDayReturningUsers, **COALESCE**(SUM(**DISTINCT** DAU.ActiveUsers), 0) **as** DailyActiveUsers, **COALESCE**(SUM(**DISTINCT** MAU.ActiveUsers), 0) **as** MonthlyActiveUsers, **CASE WHEN** SUM(**DISTINCT** MAU.ActiveUsers) > 0 **THEN**  (SUM(**DISTINCT** DAU.ActiveUsers)::**Float**/SUM(**DISTINCT** MAU.ActiveUsers)::**Float**) \* 100 |  |

299

e

|  |  |  |
| --- | --- | --- |
|  | 573  574  575  576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596  597  598  599  600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616  617  618  619  620  621  622  623  624  625  626  627  628  629  630  **ELSE**  0  **END AS** Stickiness, **COALESCE**(SUM(**DISTINCT** CU.ChurnUsers), 0) **as** Churn\_Users, SUM(**DISTINCT** TS.Total\_Sales) **as** TotalSales, **CASE WHEN** SUM(**DISTINCT** MAU.ActiveUsers) > 0 **THEN**  SUM(**DISTINCT** TS.Total\_Sales)/AC.Active\_Users **ELSE**  0  **END AS** ARPU, **CASE WHEN** SUM(**DISTINCT** TS.transaction\_count) > 0 **THEN**  SUM(**DISTINCT** TS.Total\_Sales)::**FLOAT**/SUM(**DISTINCT** TS.transaction\_count)::**FLOAT ELSE**  0  **END AS** Average\_Cheq, **CASE WHEN** SUM(**DISTINCT** TS.transaction\_count) > 0 **THEN**  SUM(**DISTINCT** TS.quantity)::**float**/SUM(**DISTINCT** TS.transaction\_count)::**float ELSE**  0  **END AS** AverageBasketSize **from** datastore.mobile\_registered\_users M  **INNER JOIN** Boost\_Location B **ON** M.JDEGROUPNUMBER = B.JDEGROUPNUMBER  **INNER JOIN** Calendar C **ON** M.registered\_date::**date** BETWEEN C.START\_DATE AND C.END\_DATE  **LEFT JOIN** Total\_Register\_users\_App TR **ON** TR.Appname = 'Boost'  **LEFT JOIN** BuyingCustomer BC **ON** B.unitid = BC.unitid AND M.customer\_id = BC.customer\_id  **LEFT JOIN** AppLevelActiveUsers AC **ON** AC.appname = 'Boost'  **LEFT JOIN** TTO **ON** B.unitid = TTO.unitnumber  **LEFT JOIN** RecurringUser1Day R1 **ON** R1.unitnumber = B.unitid  **LEFT JOIN** RecurringUser7Day R7 **ON** R7.unitnumber = B.unitid  **LEFT JOIN** RecurringUser30Day R30 **ON** R30.unitnumber = B.unitid  **LEFT JOIN** DailyActiveUsers DAU **ON** DAU.unitnumber = B.unitid **LEFT JOIN** MonthlyActiveUsers MAU **ON** MAU.unitnumber = B.unitid  **LEFT JOIN** ChurnUsers CU **ON** CU.unitnumber = B.unitid  **LEFT JOIN** TotalSalesAndTxnPerUnit TS **ON** B.unitid = TS.unitid  **GROUP BY**  TR.Total\_Register\_Users,  ac.active\_users  )  **SELECT** \* |  |

300

Output Data Dictionary:

Column Name  
Group Number  
Unit Number  
Unit Name  
New Registered Users Adoption Rate

Total Register Users  
Active Users  
Newly Converted Customer Trasaction Per Active Users (TPAU) Purchase Conversion Rate  
Time to Orders (Hours)

Ordering Within the First Month 1-Day Returning Users  
7-Days Returning Users 30-Days Returning Users Average Daily Active Users Average Monthly Active Users Stickiness

Churn Users  
Total Sales  
Average Revenue Per Users (ARPU) Average Cheque  
Average Basket Size

Description

Group Number

Unit Number

Unit Name

Count of users who was registered for the given Date Rage

Newly Register Users/Total Register Users

Total Registered users at the site/app/group level

Count of Active Users for the given date range

Subset of New Registered Users who made their very first purchase within the given date range

Transaction\_count/Active Users

Newly Converted Customer/New Registered Users Shown as a percentage

Average number hours required for a users to place an order. Calculated based on users who registered within the given timeframe

Count of users who registered within the given date range and place their first order within the month Count of distinct users who place more than one order per day an order in the given date range Count of distinct user who placed more than 1 orders in 7 days for the given date range  
count of distinct user who placed more than 1 orders in 30 days for the given date range

Average number of active users per day for the given date range  
Average number of active users per month since the beginning of time for the unit Average Daily Active Users / Average Montly Active Users (Shown as a percentage) Count of users who have not been active for at least 8 weeks consider all users in the app Total Sales for the given date range  
Total Sales / Active Users  
Total Sales / Transaction Count for the given date range  
Transaction Quantity / Transaction Count for the given Date Range

631 **FROM** AppLevelData 632  
633 **UNION ALL**634

1. 635  **SELECT** \*
2. 636  **FROM** GroupLevelData

637  
638 **UNION ALL** 639

1. 640  **SELECT** \*
2. 641  **FROM** UnitLevelData

301

DataSci-Code-Practices Code Practices - DataSci Team

302

Solution Design Principles

**Our business model is unique. We build solutions for a business but are employed in an agency model. Your time is paid for by the project you’re on like we’re a consultant shop - except we build products. That’s a bit weird, because of that we have specific guidelines in our solutions that might be different than you might see in the broader tech universe.**

1. Third-Party First: Only do what only we can do  
2. Be Collaborative: Nobody ever did anything great on their own 3. Leave Room for the Big-Picture

Third-Party First

**Third-party first makes sense in our business model because we can always pass along cost to our stakeholder. Your time costs money too, chances are they’ll pay less if we pay for third-party and add a layer on top than if we start from scratch.**

**Case Study: Here’s a story of how you can systematically think about how to leverage third parties and build on top of them:**

In our very early days we were anxious to provide value wherever we could. The UX research team wanted something where they could index their notes and tie it back to our data.

**Here’s where we went wrong**

We built an entire app where a researcher could create a new note, type it in, look up our unit number and store that data into a simple database.

You can probably guess why that went wrong.  
Our note functionality couldn’t keep up with other apps. ***Of course it couldn’t.*** To name only one, Evernote is a multi-million dollar company

where people do this for a living. **For every 1 advantage of your solution, your competitors have 10 on you. So what would’ve been the right thing to do?**

1. First, identify the part of the problem that only we are in a position to solve.

***What was your user/stakeholder/boss' real goal of what they asked?***

***What are the gaps between that goal and reality that uniquely you are in a position to solve?***

In this case, we wanted Compass identifying information tied into our data. We also wanted to store that data centrally so other people in the organization could search for it.

2. Ruthlessly pare down the things you personally need to solve.

***What are the products that do most of what you’re trying to do already?***

***Are there specialty features that might make one of the off-the-shelf solutions a better fit than the others?***

Evernote is one example, but there are hundreds of note-taking apps. One of the gaps our stakeholder had was they wanted other people in the organization to be able to search through their notes. So parse that into requirements. The off-the-shelf solution needs to make notes searchable. **Surely someone has made that already.** Do your research and find it.

They need to share it within the org, okay but how many people? Do we need an enterprise account access so literally everyone can see their notes? Probably not. **Ask your stakeholders key questions that can reduce the complexity of the solution.**

So now we’ve got rid of one requirement. That leaves us with how we attach Compass unit identifying information. Don’t see Evernote ever doing that... Those are our real requirements

303

3. Use the tools for the job

You could create an API call and use some note apps functionality for custom APIs. You could create a lookup for the users to search for the information themselves and then copy it into the app. This is where budget comes into play.

Tip: **In B2B don’t ever underestimate your users.** If they want something bad enough, they’ll do the work. If they don’t, then why are you building it.

Be Collaborative

1. Identify the people on the team that might know something about the problem that you don’t 2. Next, the people in the company  
3. Next the companies in the market

**Barriers in contacting these people aren’t yours to deal with alone. Your manager should be a servant first and a boss second. One of the biggest levers they can pull is connecting you with the people you need. Compass is a massive company and we can leverage that position into a ton of different possibilities that a startup just doesn’t have.**

**Collaborating Up: Know how your manager can help you**

1. Tell me what the problem is
2. Tell me exactly what you want me to do
3. Be reasonable - I can’t put you in touch with the CEO of Pepsi. But maybe I can get a company you think is doing something similar to come in and pitch their solution.
4. It’s okay not to know what to do. If you start your conversation by saying you’re not sure what to do it sets the expectation and brings others into the process

Leave Room for the Big Picture

CDL Data Technologies has a broader vision than each individual solution. Follow what’s going on with the department like you’d follow the news. Be tapped into what people are working on. You don’t need to know everything we’re doing, you just need to *know of* everything we’re doing.

**We get paid by project so in order to build toward something bigger we need each solution to help us continually grow our core capabilities. Our team’s challenge is building scalable services out of a consultant business paradigm. To do that, we have to understand the commonalities in the problems we’re tasked with, and build tools for ourselves so we are ever improving our ability to solve them.**

Further Reading

https://medium.com/the-mission/elon-musks-3-step-first-principles-thinking-how-to-think-and-solve-difficult-problems-like-a-ba1e73a9f6c0 - I really dislike Elon Musk so it kills me to link this but I can’t deny how critical it is.

https://towardsdatascience.com/designing-data-products-b6b93edf3d23 - Data Products are different than regular products. https://youtu.be/K4POcyN\_D5s?list=PLCzBNll0zhHYni\_1qQFcnWpbVy58P2quE - Humanistic AI Products https://designingforanalytics.com/ - A lot of great content here

304

Marketing Cloud Data Warehouse Feed Documentation

BI Team is currently exporting the following four files to Marketing Cloud.

**Daily File Drop (Done at 3:00AM)  
Drop Location (Marketing Cloud sFTP site)**: ftp.s7.exacttarget.com **Username**: 7230212  
**Password**: F4dMv9\*4RpbR  
Code is running via EC2 ETL Machine.

**Code Directory**: /etl\_user\_data/etl/projects/MarketingCloudIntegration/scripts **Schedule**: Job is currently schedule via Crontab

00 \* \* \* \* /etl\_user\_data/etl/projects/MarketingCloudIntegration/scripts/P2DataExtract.py &> /etl\_user\_data/etl/projects/MarketingCloudIntegration/output/crontabJob.out

**Primary Contact**: Julie Nguyen (Marketing)

File



File



File



File



File



305

Kiosk Dashboard Documentation

**Source Tables**

**Table Name**

"source".remedy\_fss\_cdl\_canada "source".edw\_org\_hierarchy\_dim

"source".sap\_stat\_all "source".sap\_gl\_total\_balance

"source".edw\_can\_pos\_order\_detail Redshift "source".edw\_can\_pos\_order\_header Redshift

**First Module: (Source SAP)  
Average KPI’s Tab Calculates the following**

**System**

Redshift Redshift

Redshift Redshift

Netizza Netizza

**Description**

Identify KIOSK Units

Source of SAP Statistical data SAP Cost and Revenue Data

POS Data POS Data

Change in Average Hours pre and post Go live Date (Weighted) Average Percent Change in Average Transactions pre and post Go Live Change in Check Average pre and post Go live Date (Weighted) Percent Change on Total Sales Excluding Catering pre and post Go Live

**Second Module: (Source SAP)**

**Details of Average KPI’s**

Second module provides details on Average KPI’s Tab Results Results can be filtered by following:

Kiosk Type Kiosk Machine Sector Name Unit name

**Third Module: (Source SAP)**

**Cost t VS Revenue**

Results can be filtered by Following:

Sector  
Unit  
Fiscal Period Fiscal Year

Detailed Results of Revenue

**Fourth Module (Source SAP) Current VS previous Revenue**

This module will compares Results of Current Period (Selected) to Previous Period Detailed Revenue Results by :

Sector Unit

**Top 10 performing Units (Revenue without Catering**

**Fifth Module (Source SAP)  
Revenue versus Budget**Revenue Versus budget module compares Actual Revenue VS Budget Following filters can be applied

Sector Name Unit Name Fiscal Period Fiscal Year

**Sixth Module (Source SAP) Cost versus Budget**

306

Revenue Versus budget module compares Actual Revenue VS Budget Following filters can be applied

Sector Name Unit Name Fiscal Period Fiscal Year

**Seventh Module (Source POS Data)**

POS Date Measure Revenue By Tender Class Following Filters can be applied

Sector Name Unit Name Fiscal Period Fiscal Year

**Eight Module (Source POS Data)**

POS Data Up Sell Opportunities

307

ETL BI Controller  
Link: http://ec2-18-222-58-69.us-east-2.compute.amazonaws.com:8080/etl\_ui/#

308

Create or replace the table (DDL)

Prerequisites:  
Create table .sql script uploaded to S3

To create a table on redshift, you can copy a preexisting drop\_n\_recreate schedule.

**Administration → Schedule → List Schedule Details**

Type ‘drop\_n\_recreate’ and select a job Select Copy Schedule  
Save new schedule name

Then edit the newly created schedule's attribute to point to S3 path for the .sql create table script.

**Administration → Attribute → Manage Schedule Attributes**

Select Edit Attributes  
Type /[schedule\_name] in search bar Select schedule  
Toggle Edit  
Enter S3 path for create table .sql script Submit

Select Job and Force Start

**Administration → Schedule → List Schedule Details**

Search for /[schedule\_name]  
Select new schedule  
Force start  
**MAKE SURE TABLE NAME IN SQL SCRIPT DOES NOT OVERLAP WITH ANOTHER TABLE NAME**

309

310

Load data from S3 to Redshift table

**Prerequisites:**

Table on redshift  
Table registered on ETL Manager .CSV of new data uploaded to s3

**This process involves four tasks:**

Copying a preexisting schedule  
Updating the schedule with the appropriate **s3 file path** Updating the schedule with the appropriate **database table** Selecting the **update frequency**

Once these steps are complete, you can force start the schedule

Start by copying a preexisting S3 update schedule.

**Administration → Schedule → List Schedule Details**

Type 'STATIC\_S3\_FILE' and select a schedule to copy Select Copy Schedule  
Save new schedule name

Change the attributes to update the S3 file path

**Administration → Attribute → Manage Schedule Attributes**

Select Edit Attributes  
Type /[schedule\_name] in search bar Select schedule  
Toggle Edit  
Enter S3 path for create table .sql script Submit

Change the DBTABLE config by deleting the old config and adding a new config

**Delete Old DBTABLE Config**

**Administration → Schedule → Edit Details**

Toggle delete mode  
Search for /[schedule\_name]  
Toggle delete for row (Detail Type = DBTABLE) Click Submit

311

**Add New DBTABLE Config**

**Administration → Schedule → Edit Details**

Select Schedule  
Select DB (Redshift0)  
Select registered table name

Select the schedule frequency

**Administration → Schedule → Modify Schedule**

Select schedule  
Select schedule frequency Submit

312

Load data from a Redshift query to a Redshift table (drop and recreate) Recreate table structure and load the data

1. **If table doesn't exist on Redshift server cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com**a. Run create table sql statement in DBeaver or any other client. Specify schema name before table name, otherwise table will be created in **public**

schema or any schema that is selected as a default OR go to Create or replace the table (DDL) b. Register table in CDL BI Controller

1. Go to Administration→ Database→ Configure Table
2. List DB Names, pick REDSHIFT0 => cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com[5439] and fill out the parameters' values and click Submit

iii. if you don't know schema name list schemas in Redshift

c. Go to Administration→ Database→ Configure Table Metadata

1. Click List DB Names tab and double click on REDSHIFT0 => cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com[5439].
2. In Search text >> type in table name and double click on a table. Use (+)Add and (-)Remove buttons to add and remove distribution keys (DSTKY) and sort keys (SRTKY) then press Submit

|  |
| --- |
| **List schemas in Redshift** |
| select s.nspname as table\_schema,  s.oid as schema\_id,  u.usename as owner  from pg\_catalog.pg\_namespace s  join pg\_catalog.pg\_user u on u.usesysid = s.nspowner  where s.nspname not like '%pg\_temp%'  order by table\_schema;  Copy to clipboard   |

313

2. **Create script and load it to S3 bucket.**a. S3 buckets: https://s3.console.aws.amazon.com/s3/home?region=us-east-2

b. Script file \*.sql should NOT have **; (semicolon)** symbol in the end

3. **CDL BI Controller :**a. Go to Administration →Schedule→ List Schedule details

i. In Search box type in "Sample" and double click **REDSHIFT\_MATERIALIZE\_SQL\_DROP\_AND\_RECREATE\_SAMPLE\_SCHEDULE.** Copy this schedule with a new name. Naming convention: REDSHIFT\_MATERIALIZE\_SQL\_DROP\_AND\_RECREATE\_[table name]

b. Go to Administration → Schedule→Edit Details.  
i. In SearchText>> type in your new schedule name then press Schedules and select schedule name.

ii. Clear SearchText>>, press DBs tab and double click REDSHIFT0 => cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com[5439]

314

iii. In SearchText>> type in destination table name and press DB-Tables tab. Double click the table and click Submit

c. Go to Administration → Attribute→Manage Schedule Attribute.  
i. In SearchText>> type in your new schedule name then press Schedules and double click schedule name.

ii. Select Edit Attributes radiobutton and double click schedule name.  
iii. Insert S3 path to your sql query as an Attribute Value (without s3://cdl-bi-raw-source/ prefix) and click Submit:

315

d. Go to Administration →Schedule→ List Schedule details  
i. In SearchText>> type in your new schedule name then press Schedules and double click schedule name.

ii. Force Start the job. In 2 minutes the log will appear on Home page.

316

Schedule Details and Attributes

This document outlines the steps required to view a particular schedule’s details.

Step 1: Navigate to the ETL Controller  
Click on the following link to navigate to the ETL Controller: http://ec2-18-222-58-69.us-east-2.compute.amazonaws.com:8080/etl\_ui/

Step 2: Navigate to the Schedule Page  
Once you are on the ETL Controller’s home page, hover over the Administration tab and click on “Schedule”.

Once you have clicked on “Schedule”, click on “List Schedule Details”.

You will be brought to the Schedule Detail page. This page allows you to search for specific properties related to a schedule.

Step 3: View Details  
Once you are in the “List Schedule Details”, the pane on the right hand side of the page is where you will search for a schedule. Follow the following steps to

1. Type a few words into the input box to the right of “List Projects”.  
2. After entering some text, click on “List Schedules” and a list of potential schedules will be listed in the area below the search box. 3. **Double Click** on the schedule name you would like to see details on.

The following image outlines which areas correspond with each step.

317

This view only shows the default attributes regarding a specific schedule.

**Default Attributes**

Schedule ID Schedule Name ProjectID Project Directory CommandID Script Name

DependencyID Dependency Name

**Description**

The unique ID of the selected schedule The name of the selected schedule The unique ID of a project  
The absolute path to the project

The unique ID of the command that is used to run the script  
The name of the script that runs the job. The absolute path to the script follows the following template “{project\_directory}/scripts/{script\_name}”, so in this example, the script lives

in: /etl\_user\_data/etl/projects/quickbase/scripts/api\_extract.py  
The unique ID of another schedule this schedule depends on. If this value does not exist, the scheduler does not depend on any jobs.  
The Schedule Name of the job that is required to complete before this job starts. If this value does not exist, the scheduler does not depend on any jobs.

**Example**

682 QUICK\_BASE\_PATIENT\_NUTRITION\_API\_EXTRACT 35  
/etl\_user\_data/etl/projects/quickbase  
143  
api\_extract.py

{blank} or {integer}  
{blank} or {Schedule Name}

To view a more detailed list of attributes, click the List Details.

This value displays some environmental variables that will be set when running the job.  
If we look at the api\_extract.py source file, we see that these environment variables are used in the program configuration:

318

In this specific example, these environment variables are used to configure S3 Bucket where the data will be sent to.  
If you do not find a particular detail in this section, click on “List Attributes” to the right hand side of “List Details” which will display other attributes regarding the job

1 AWS\_CONFIG\_1\_ACCESS\_ID = iGenSchdObj.etl\_schedule\_enviornment\_variable\_values['SCHD\_AWS\_CONFIG\_1\_ACCESS\_ID'] 2 AWS\_CONFIG\_1\_ACCESS\_KEY = iGenSchdObj.etl\_schedule\_enviornment\_variable\_values['SCHD\_AWS\_CONFIG\_1\_ACCESS\_KEY'] 3 AWS\_CONFIG\_1\_REGION = iGenSchdObj.etl\_schedule\_enviornment\_variable\_values['SCHD\_AWS\_CONFIG\_1\_REGION']  
4 AWS\_CONFIG\_1\_S3\_PATH = iGenSchdObj.etl\_schedule\_enviornment\_variable\_values['SCHD\_AWS\_CONFIG\_1\_URL']

319

Data Warehouse Documentation Redshift Database Documentation

320

Datamart

321

Test Page

This is a test page

1 This is some code that is used

322

Source

323

cdl\_bi\_edw.source.edw\_can\_pos\_order\_detail

CDL BI ETL CONTROLLER Schedule Name: REDSHIFT\_LOAD\_EDW\_POS\_ORDER\_DETAIL, ATHENA\_DELTA\_COLUMN\_STAT\_EDW\_POS\_ORDER\_DETAIL Fields Description

**Field Name**

pos\_order\_detail\_key pos\_order\_hdr\_key

fin\_fiscal\_yr fin\_fiscal\_month org\_unit\_key erp\_system\_id erp\_entity\_id

system\_id  
order\_header\_id  
item\_id  
item\_name  
item\_qty standard\_product\_class\_id standard\_product\_class\_desc product\_class\_id product\_class\_desc standard\_class standard\_item item\_long\_name sales\_amt\_gross discount\_amt sales\_amt\_less\_disc close\_datetime

close\_date close\_time pos\_source

**Description**

Key used to link back to pos\_order\_hdr. Unique for each check.

Unique key for each compass unit.

Key for compass unit. Unique when combined with erp\_system\_id.

ID tied to item name. Not unique.  
Name of item.  
Volume of line item sold.  
Old categorization scheme pre cleansed tier

Total gross revenue from line item.  
Total discount applied.  
Total revenue from line item after discount. Datetime transaction was closed  
Date transaction was closed.  
Time transaction was closed

Source of the POS (e.g., AGILYSYS, MICROS, NEXTEP)

**Is Sort Key**

Y

Y Y

Y

|  |  |  |
| --- | --- | --- |
| profit\_center | A more granular unit field. May have different meanings across sectors  Eurest/Morrison/RA - Profit Center is used to distinguish cafes within a single cost center. In some occasions, separate buildings (usually close geographically)  Canteen - Canteen Account Number |  |
| profit\_center\_name | A more granular unit field. May have different meanings across sectors  Eurest/Morrison/RA - Profit Center is used to distinguish cafes within a single cost center. In some occasions, separate buildings (usually close geographically)  Canteen - Canteen Account Name |  |

324

employee\_nbr

A more granular unit field. May have different meanings across sectors

Eurest/Morrison/RA - An employee identifier linked to the terminal of the input transaction. Varies in reliability

Canteen - Canteen Location Id

emp\_name

A more granular unit field. May have different meanings across sectors

Eurest/Morrison/RA - An employee name linked to the terminal of the input transaction. Varies in reliability.

Canteen - Canteen Location Name

order\_type order\_type\_name check\_nbr meal\_type combo\_ind

revenue\_weight meal\_period\_sort cleansed\_name cleansed\_tier\_1

cleansed\_tier\_2 cleansed\_tier\_3 cleansed\_tier\_4

record\_status record\_status\_date category\_status category\_status\_date ius\_week ius\_week\_end\_dt sold\_by\_weight\_ind uom

modifier\_origin check\_detail\_type tender\_type import\_date item\_added\_datetime pos\_location\_key pos\_product\_key

cleansed\_brand

Non Populated field

Tier 1 of the four tier item classification hierarchy (Food, Non-Food, Alcohol Beverages, Non- Alcohol Beverages)

Tier 2 of the four tier item classification hierarchy

Tier 3 of the four tier item classification hierarchy

Tier 4 of the four tier item classification hierarchy

|  |  |  |
| --- | --- | --- |
| sku | Should be barcode number, but very inconsistent. Should be 12 digit integer, but sometimes less than 12 digits, sometimes includes characters, sometimes both. |  |

Unique key at item\_name by org\_unit\_key

Tags brand name for Prepackaged Snacks and Bevereages

|  |  |  |
| --- | --- | --- |
| cleansed\_prep\_method | Tags snacks and beverages (cleansed\_tier\_1 = 'Non Alcohol Beverages' or cleansed\_tier\_2 = 'Snacks/Impulse') as Prepackaged Items (CPG) or Prepared Items. |  |

325

derived\_close\_date\_year  
derived\_close\_date\_month  
derived\_close\_date\_day Y etl\_extract\_timestamp

326

source.remedy\_fss\_cdl\_canada

**Summary:**

*remedy\_fss\_cdl\_canada* holds all the project details for POS, ETC/Kiosk projects at the units(ERP\_ENTITY\_ID) level. For example, you can find go live date, status of the project, # of kiosk/terminal requested and etc.

**Table Name**:

unit\_number status wip\_state fss\_solution category

number\_of\_sco\_pos\_terminals number\_of\_order\_n\_pay\_kiosk number\_of\_order\_only\_kiosks number\_of\_self\_checkout\_kiosks go\_live\_date

Key for linking back to unit table. Note: Merge source.edw\_org\_unit\_dim with remedy\_fss\_cdl\_canada.unit\_number= source.edw\_org\_unit\_dim.erp\_entity\_id

Key for identifying the status of the project. Note: If status ='Work In Progress' and wip\_state = 'Invoicing' then 'Completed'; or status='Completed' then 'Completed'

Key for identifying the status of the project. Note: If status ='Work In Progress' and wip\_state = 'Invoicing' then 'Completed'; or status='Completed' then 'Completed'

Note: I usually use fss\_solution=='POS' for kiosk related projects, please correct me or update this documentation if you find a better filter.

Note: I usually use category=='Implementation' for kiosk related projects, please correct me or update this documentation if you find a better filter.

Note: One unit might have several go\_live\_dates if they had several projects.

**Column Name Data Description isSortKey Type**

Additional Note: Total Kiosk=number\_of\_sco\_pos\_terminals + number\_of\_order\_n\_pay\_kiosk + number\_of\_order\_only\_kiosks + number\_of\_self\_checkout\_kiosks

**Data Source Information:**

**Server Name - Database Name - Table Name -**

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..): Schedule Detail:  
Distribution Type:  
Sort Key:**

**Job Name: Additional Notes:**

327

Source.clearview\_ej\_detail

**Table Name**: Source.clearview\_ej\_detail Column Name

Restaurant Number Transaction ID record\_sequence\_key

link\_sequence

sale\_mode

item\_id item\_name quantity  
Price Total\_Amount Discount\_Flag modifier\_flag Void\_flag

**Data Source Information:**

**Server Name - CDL FTP Drive Database Name - Flat file (CSV) Table Name - EJ\_Detail**

Data Type Integer

BigInt BigInt

Int

SmallInt

int VARCHAR(128) Double  
Double  
Double  
Bool  
Bool  
smallint

Description isSortKey

Yes

Clearview-Generated Unique Transaction ID

Clearview -Generated Sequence Number. Used internally to uniquely identify each detail record

Link back to Record Sequence Key to associate line items with a parent item. Typically used with Combos

1 = Line Item is part of a combo

POS Menu Item Code (aka PLU) POS Menu Item Description

Total amount for line item (Quantity x Price) 1 = discount, 0 = not a discount  
1 = Modifier, 0 = Not a modifier  
0 = not voided, 1 = voided, 2 = refund

|  |  |  |  |
| --- | --- | --- | --- |
| detail\_type | SmallInt | Describes the type of line item. Will be one of the following values:  10 = Regular Menu Item (PLU) 20 = Coupon Discount 30 = Combo Discount 40 = Other Discount (i.e Staff Meal) |  |

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..): Schedule Detail:  
Distribution Type:  
Sort Key:**

**Job Name: Additional Notes:**

328

SAP Budget

**Table Name**: source.sap\_budget

**Column Name**

gl\_description gl\_account\_number company\_code sector

sector\_name cost\_center  
amount reference\_document fiscal\_year\_period etl\_extract\_timestamp

**Data Type**

VARCHAR(128) VARCHAR(12) VARCHAR(8) VARCHAR(10) VARCHAR(64) VARCHAR(12) FLOAT8 VARCHAR(64) VARCHAR(10) TIMESTAMP

**Description**

**isSortKey**

Y

Y

Y

**isDistKey**

**Data Source Information:  
S3 Bucket:** s3://cdl-bi-raw-source/SAP/Budget/\*txt.gz

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta Load, Daily **Schedule Detail:** SAP\_BUDGET\_RESPONSE\_LOAD\_S3\_FILES\_TO\_REDSHIFT  
**Distribution Type:** None  
**Job Name:** SAP\_BUDGET\_RESPONSE\_LOAD\_S3\_FILES\_TO\_REDSHIFT

329

SAP GL Accounts

**Table Name**: source.sap\_gl\_accounts

**Column Name**

gl\_account gl\_account\_desc

**Data Type**

INT4 VARCHAR(125)

**Description**

**isSortKey isDistKey**

Y

**Data Source Information:** Unknown  
**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Not reloaded or scheduled **Schedule Detail:** None  
**Distribution Type:** Key Distribution  
**Sort Key:** None  
**Job Name:** None  
**Additional Notes:** None

330

SAP GL Hierarchy

**Table Name**: source.sap\_gl\_hierarchy

**Column Name**

gl\_root  
gl\_parent  
gl\_child gl\_description etl\_extract\_timestamp

**Data Type**

VARCHAR(32) VARCHAR(32) VARCHAR(32) VARCHAR(256) TIMESTAMP

**Description**

**isSortKey**

**isDistKey**

**Data Source Information:  
S3 Bucket:** s3://cdl-bi-raw-source/SAP/GL\_Hierarchy/\*.txt.gz

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta Load, Daily **Schedule Detail:** SAP\_GL\_HIERARCHY\_LOAD\_S3\_FILES\_TO\_REDSHIFT  
**Distribution Type:** None  
**Sort Key:** None

**Job Name:** SAP\_GL\_HIERARCHY\_LOAD\_S3\_FILES\_TO\_REDSHIFT **Additional Notes:** None

331

SAP GL Hierarchy Collapsed

**Table Name**: source.sap\_gl\_hierarchy\_collapsed

**Column Name**

gl\_root  
gl\_parent  
gl\_child gl\_description hierarchy\_level etl\_extract\_timestamp

**Data Source Information:**

**Data Type**

VARCHAR(32) VARCHAR(32) VARCHAR(32) VARCHAR(256) INT2 TIMESTAMP

**Description**

**isSortKey**

Y

**isDistKey**

**Server Name -** cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com **Database Name -** cdl\_bi\_edw  
**Table Name -** source.sap\_gl\_hierarchy

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Truncate & Reload, Daily **Schedule Detail:** SAP\_COLLAPSE\_GL\_HIERARCHY  
**Distribution Type:** None  
**Sort Key:** gl\_parent

**Job Name:** SAP\_COLLAPSE\_GL\_HIERARCHY **Additional Notes:** None

332

SAP GL Master

**Table Name**: source.sap\_gl\_master

**Column Name**

chart\_of\_accounts gl\_account\_number balance\_sheet\_account opendate

created\_by pl\_account\_type gl\_account\_group deletion\_indicator creation\_block\_indicator posting\_block\_indicator planning\_block\_indicator search\_term\_of\_matchcode language\_code

long\_text  
short\_text etl\_insert\_timestamp

**Data Source Information:**

**Data Type**

VARCHAR(8) VARCHAR(12) VARCHAR(8) DATE VARCHAR(16) VARCHAR(8) VARCHAR(8) VARCHAR(8) VARCHAR(8) VARCHAR(8) VARCHAR(8) VARCHAR(64) VARCHAR(8) VARCHAR(128) VARCHAR(64) TIMESTAMP

**Description**

**isSortKey**

Y

**isDistKey**

**Server Name -** cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com **Database Name -** cdl\_bi\_edw  
**Table Name -** source.sap\_gl\_master

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Truncate & Reload, Daily **Schedule Detail:** SAP\_GL\_MASTER\_REDSHIFT\_LOAD  
**Distribution Type:** None  
**Sort Key:** gl\_account\_number

**Job Name:** SAP\_GL\_MASTER\_REDSHIFT\_LOAD **Additional Notes:** None

333

SAP GL Total Balance

**Table Name**: source.sap\_gl\_total\_balance

**Column Name**

gl\_description gl\_account\_number dt  
line\_item\_text company\_code sector  
sector\_name cost\_center  
amount reference\_document allocation fiscal\_year\_period doc\_date  
post\_date document\_number line etl\_extract\_timestamp

**Data Type**

VARCHAR(128) VARCHAR(12) VARCHAR(8) VARCHAR(64) VARCHAR(32) VARCHAR(16) VARCHAR(64) VARCHAR(12) FLOAT8 VARCHAR(64) VARCHAR(64) VARCHAR(10) VARCHAR(8) DATE VARCHAR(12) VARCHAR(8) TIMESTAMP

**Description**

**isSortKey**

Y

**isDistKey**

**Data Source Information:  
S3 Bucket:** s3://cdl-bi-raw-source/SAP/TotalBalances/CDL\_TotalBalances\_L\_Q\_\*.txt.gz

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta Load, Daily **Schedule Detail:** SAP\_TOTAL\_BALANCES\_LOAD\_S3\_FILES\_TO\_REDSHIFT  
**Distribution Type:** None  
**Sort Key:** gl\_account\_number

**Job Name:** SAP\_TOTAL\_BALANCES\_LOAD\_S3\_FILES\_TO\_REDSHIFT **Additional Notes:** None

334

SAP HRIS Eurest Eastern Report

**Table Name**: source.sap\_hris\_eurest\_eastern\_actives\_report

**Column Name**

scd\_key  
scd\_date  
pers\_no form\_of\_address\_key first\_name

last\_name  
position\_  
cost\_center cost\_center\_name psubarea personal\_subarea employee\_subgroup name\_of\_manager\_om regional\_vice\_president house\_number\_and\_street address\_line\_2

city  
state  
postal\_code  
month\_  
bod  
nickname service\_award\_date division  
divisionname  
region  
regionname scd\_insert\_timestamp scd\_record\_status

**Data Type**

VARCHAR(32) VARCHAR(32) VARCHAR(10) VARCHAR(8) VARCHAR(32) VARCHAR(32) VARCHAR(64) VARCHAR(10) VARCHAR(32) VARCHAR(8) VARCHAR(24) VARCHAR(24) VARCHAR(64) VARCHAR(64) VARCHAR(64) VARCHAR(64) VARCHAR(64) VARCHAR(3) VARCHAR(16) VARCHAR(2) DATE VARCHAR(32) DATE VARCHAR(8) VARCHAR(32) VARCHAR(8) VARCHAR(32) TIMESTAMP BPCHAR

**Description**

**isSortKey**

Y

Y

**isDistKey**

**Data Source Information:  
Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** SCD, Daily **Schedule Detail:** REDSHIFT\_HRIS\_SCD\_BULK\_REFRESH  
**Distribution Type:** None  
**Sort Key:** scd\_key, pers\_no  
**Job Name:** REDSHIFT\_HRIS\_SCD\_BULK\_REFRESH  
**Additional Notes:** None

335

SAP Stat All

**Table Name**: source.sap\_stat\_all

**Column Name**

company\_code cost\_center company\_name fiscal\_year fiscal\_period gl\_account\_number stat\_account skf\_name

quantity  
doc\_type entry\_date entry\_time user\_name document\_number post\_date

dm  
cycle  
area etl\_extract\_timestamp

**Data Type**

VARCHAR(8) VARCHAR(12) VARCHAR(64) INT2 VARCHAR(4) VARCHAR(12) VARCHAR(12) VARCHAR(64) INT8 VARCHAR(4) DATE VARCHAR(12) VARCHAR(16) VARCHAR(12) DATE VARCHAR(32) VARCHAR(12) VARCHAR(16) TIMESTAMP

**Description**

**isSortKey**

Y

**isDistKey**

Y

**Data Source Information:  
S3 Bucket:** s3://cdl-bi-raw-source/SAP/Stat\_All/CDL\_Stat\_All\_\*.txt.gz

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta, Daily **Schedule Detail:** SAP\_STAT\_ALL\_RESPONSE\_LOAD\_S3\_FILES\_TO\_REDSHIFT  
**Distribution Type:** None  
**Sort Key:** cost\_center, post\_date

**Job Name:** SAP\_STAT\_ALL\_RESPONSE\_LOAD\_S3\_FILES\_TO\_REDSHIFT **Additional Notes:** None

336

Boost Event Schema

**Table Name**: **Column Name**

event\_date event\_timestamp

event\_name

event\_param\_key event\_str\_val  
event\_int\_val  
event\_float\_val event\_double\_val event\_previous\_timestamp event\_server\_timestamp\_offset user\_id

user\_pseudo\_id  
user\_prop\_key  
user\_prop\_str\_val user\_prop\_int\_val user\_prop\_float\_val user\_prop\_double\_val user\_prop\_ts\_micros user\_first\_touch\_timestamp user\_revenue  
user\_currency  
device\_category device\_mobile\_brand\_name device\_mobile\_model\_name device\_mobile\_marketing\_name device\_mobile\_os\_hardware\_model device\_operating\_system device\_vendor\_id device\_advertising\_id device\_language device\_is\_limited\_ad\_tracking device\_time\_zone\_offset\_seconds device\_browser device\_browser\_version device\_web\_info\_browser\_version device\_web\_info\_hostname app\_info\_id

**Data Type**

DATE INT8

VARCHAR(100)

VARCHAR(256) VARCHAR(256) INT8  
FLOAT8 FLOAT8

INT8  
INT4 VARCHAR(100) VARCHAR(256) VARCHAR(256) VARCHAR(256) INT8  
FLOAT8 FLOAT8  
INT8  
INT8  
FLOAT8 VARCHAR(100) VARCHAR(128) VARCHAR(100) VARCHAR(100) VARCHAR(100) VARCHAR(100) VARCHAR(100) VARCHAR(256) VARCHAR(256) VARCHAR(100) VARCHAR(50) INT4 VARCHAR(100) VARCHAR(100) VARCHAR(100) VARCHAR(256) VARCHAR(100)

**Description isSortKey isDistKey**

date of event Y

UNIX Timestamp (Milliseconds)

Name of Y event

Event key

337

app\_info\_version app\_info\_install\_store app\_info\_firebase\_app\_id app\_info\_info\_install\_source traffic\_source\_name traffic\_source\_medium traffic\_source\_source stream\_id

platform event\_dimension\_hostname

VARCHAR(100)  
VARCHAR(100)  
VARCHAR(100)  
VARCHAR(100)  
VARCHAR(100)  
VARCHAR(100)  
VARCHAR(100)  
VARCHAR(128) Y VARCHAR(50)

VARCHAR(128)

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta Load (Incremental) **Schedule Detail:** Daily  
**Distribution Type:** Even  
**Sort Key:** Interleaved ( event\_date, event\_name, stream\_id )

**Job Name:** boost\_event\_collector\_v-0.0.1 **Additional Notes: Airflow Job**

**Column Name Data Type Description isSortKey isDistKey**

338

Datastore

Create file list

339

Mobile Orders

**Table Name**: datastore.mobile\_orders

**Column Name**

id grain\_lvl

orderid  
itemid  
optionid menuitemname

brandname ordertime\_utc ordertime  
orderdate  
ca\_fiscal\_yr ca\_fiscal\_quarter ca\_fiscal\_period\_name ca\_fiscal\_week\_name us\_fiscal\_yr us\_fiscal\_quarter us\_fiscal\_period\_name us\_fiscal\_week\_name customerid pickupname pickuptime\_utc pickuptime

pickupdate appname  
sector static\_locationname

unitnumber unitname groupnumber groupname sitename  
storeid etl\_source\_system country

timezone issitelive paymenttype

**Data Type**

VARCHAR(32) VARCHAR(25)

VARCHAR(128) VARCHAR(257) VARCHAR(257) VARCHAR(512)

VARCHAR(375) TIMESTAMP TIMESTAMP DATE

INT4 VARCHAR(11) VARCHAR(20) VARCHAR(20) INT4  
INT4 VARCHAR(8) VARCHAR(24) VARCHAR(255) VARCHAR(513) TIMESTAMP TIMESTAMP DATE VARCHAR(250) VARCHAR(100) VARCHAR(256)

INT4 VARCHAR(256) VARCHAR(25) VARCHAR(100) VARCHAR(256) INT4 VARCHAR(125) VARCHAR(96) VARCHAR(96) BOOL VARCHAR(375)

**Description**

Unique ID

Table split into 3 levels: order - transaction level, item - item level, option - option level

Transaction ID

Item ID, empty for grain\_lvl='order'

Option ID, empty for grain\_lvl in ('order', 'item')

**isSortKey isDistKey**

Y

Item and option name, empty for for grain\_lvl='order'

Brand name  
UTC Order Timestamp

Canadian Fiscal Year

Location name (source storename field from static\_dimensions.cdl\_unitid\_storeid\_conversion, maintained manually)

JDE/SAP unit id Unit Name JDE/SAP group id Group Name

Site Name Volante Store ID Source database Unit country  
Unit timezone  
Is site active

Payment type (including meal plan and credit cards)

Y

Y

340

deliveryordertype deliveryareaname deliverystarttimelocal deliveryendtimelocal promocode promocodediscounttype

promoamountoff promomaxamountoff discountcode discounttype discountpercentoff ordertotalamount ordernetsales orderloyaltyamount orderpromoamount orderdiscountamount itemquantity itemnetsales servicefeeamount currency servicefeeamount\_cad servicefeeamount\_usd itemnetsales\_cad

itemnetsales\_usd

orderpromoamount\_cad orderpromoamount\_usd cad\_rate cad\_against\_usd transactiontype

istestorder  
sourcedb etl\_extract\_timestamp

**Data Source Information:**

VARCHAR(32) VARCHAR(250) TIMESTAMP TIMESTAMP VARCHAR(375) VARCHAR(32)

FLOAT8 FLOAT8 VARCHAR(128) VARCHAR(32) INT2

NUMERIC NUMERIC NUMERIC FLOAT8 NUMERIC NUMERIC NUMERIC NUMERIC VARCHAR(64) NUMERIC NUMERIC FLOAT8

FLOAT8

FLOAT8 FLOAT8 FLOAT8 FLOAT8 VARCHAR(13)

BOOL VARCHAR(3) TIMESTAMP

Pickup or Delivery order Delivery Area

Promotion Code

Is promotion discount based on amount or persentage

Order level: Sales after tax and after discount Order level: Sales before tax and after discount Order level: Loyalty discount amount  
Order level: Promo discount amount

Order level: Discount amount

Item level: Number of items in the order

Item level: Sales before tax and after discount

Service fee amount

Sales currency

Service fee amount CAD

Service fee amount USD

Item level: Sales before tax and after discount CAD

Item level: Sales before tax and after discount USD

Order level: Promo discount amount CAD Order level: Promo discount amount USD CAD forex rate  
CAD against USD forex rate

Regular, Mobile Wallet or Tap2Eat order. Mobile Wallet and Tap2Eat orders don't have item level.

Test order flag  
p2, p1 and t2e flag  
when order was inserted to the source db table

**Server Name -** cdl-bi-edw-dev0.cwcgvbnuuh0w.us-east-2.redshift.amazonaws.com **Database Name -** cdl\_bi\_edw  
**Table Name -** datamart.mobile\_orders

**Reload Type (Incremental, Truncate & Reload, Slowly Changing Dimensions (SCD) etc..):** Delta Load, not scheduled **Schedule Detail:** REDSHIFT\_MATERIALIZE\_SQL\_DATASTORE\_MOBILE\_REFRESH  
**Distribution Type:** Key Distribution ( DISTKEY (id) )  
**Sort Key:** appname, unitnumber

**Job Name:** REDSHIFT\_MATERIALIZE\_SQL\_DATASTORE\_MOBILE\_REFRESH **Additional Notes:**

**Column Name Data Type Description isSortKey isDistKey**

341

Datascience\_staging

342

Data Sophistication Tables  
Information on columns, fields, and tables created downstream of source data.

Challenger Features DS\_NIGHTLY\_CANADA\_CTS

343

DS\_NIGHTLY\_CANADA\_CTS

**Summary**

The Cleansed Tier algorithm is applied to POS data from three sources: 1) Volante Canada, 2) Volante US, and 3) Clearview. Distinct item name/item identifiers from these sources are consolidated into a single table. Three different sets of algorithms are applied to add: 1) the cleansed tier hierarchy, 2) a snack/hot beverage/ cold beverage flag, and 3) a cleansed item name (snacks and beverage only). The final consolidated table is uploaded to the cdl-bi s3 account (bucket: cdl-bi-data-science, file path: cdl\_integration/can\_volante/CDL\_BI\_CONVERSION\_MASTER\_LATEST.csv). An ETL-Manager job is set up to read this file and upload to [table name] on redshift.

**Dependencies**

Information Source Tables: cdl\_datamart.pos\_ca, cdl\_datamart\_pos\_us  
ETL Manager Job Name:  
ETL Manager Creation Job Name:  
S3 Path: cdl-bi-data-science/cdl\_integration/can\_volante/CDL\_BI\_CONVERSION\_MASTER\_LATEST.csv

**Update Frequency**

Weekly for new items  
No fixed schedule for manual corrections

**Table Fields and Descriptions Field**

PRINTNAME MENUITEMNAME MENUITEMID  
STOREID ETL\_SOURCE\_SYSTEM CLEANSED\_TIER\_1 CLEANSED\_TIER\_2 CLEANSED\_TIER\_3 CLEANSED\_TIER\_4 HOT\_COLD\_OR\_SNACK CDL\_MASTER\_NAME

**Source Queries Volante Canada:**

**Type**

string string integer integer string string string string string string string

**Description**

Print name from volante. Name printed on receipt Name of the menu item.  
ID of the menu item.  
ID of the store

Source of the item [Volante US, Volante Canada, or Clearview] 1st tier of the cleansed tier hierarchy  
1st tier of the cleansed tier hierarchy  
1st tier of the cleansed tier hierarchy

1st tier of the cleansed tier hierarchy  
Flag for hot/cold beverages and prepacked snacks Cleaned menu item name

344

1. 1  Select DISTINCT
2. 2  MENUITEMNAME,
3. 3  MENUITEMID,
4. 4  STOREID,
5. 5  CLEANSED\_TIER\_1,
6. 6  CLEANSED\_TIER\_2,
7. 7  CLEANSED\_TIER\_3,
8. 8  CLEANSED\_TIER\_4
9. 9  FROM
10. 10  cdl\_bi\_edw.datamart.pos\_ca

**Volante US:**

**Clearview:**

**Algorithm Details**

The output from the following algorithms all result in the addition of categorical columns to the POS conversion table. In all cases, POS Items are classified according to a combination out from classifier models, a set of rule/keyword-based assignments, and a manually maintained lookup table.

**Cleansed Tier Algorithm**

**[CLEANSED\_TIER\_1, CLEANSED\_TIER\_2, CLEANSED\_TIER\_3, CLEANSED\_TIER\_4]**

This is a four-tier hierarchy that separates POS items into four general categories: 1) Add-Ons, 2) Food, 3) Non-Alcohol Beverages and 4) Alcohol Beverages. The each POS item is classified according to output from a set of classifier models, a set of keyword-based searches and a final manually curated look-up table.

**Snack / Beverages - Hot or Cold  
[HOT\_COLD\_OR\_SNACK]**This flags items as: 1) Hot Beverage, 2) Cold Beverage or 3) Prepackaged Snacks.

**Cleansed Item Name**

**[CDL\_MASTER\_NAME]**

This column is currently only used for the Canadian Scorecard project, and applied only to Snacks and Beverages. The algorithm does some basic pre-processing: 1) taking lower case, 2) removing numbers, 3) removing non-letter characters. An additional layer of [experimental] algorithmic string cleaning is applied.

1 2 3 4 5 6 7

Select DISTINCT MENUITEMNAME,

MENUITEMID, STOREID, ETL\_SOURCE\_SYSTEM

FROM cdl\_bi\_edw.datamart.pos\_us

1

345

Challenger Features

346

Challenger\_Features\_CAN\_Volante

Information on columns, fields, and tables created downstream of source data.

**Summary:** Challenger features run on Canadian Volante data. Business user is CDL and the business case is that each boolean Challenger feature (listed below) highlights an individual and particular reason why an item cannot be on the app. These are combined into the POSSIBLE\_APP\_ITEM field which indicates whether or not an item can be supported by our app based on capability as of July 16th, 2019

**Source Queries Volante Canada:**

**SELECT**

CA.STOREID  
, CA.STORENAME  
, CA.MENUITEMID  
, CA.MENUITEMNAME  
, CA.CLEANSED\_TIER\_1, CA.CLEANSED\_TIER\_2, CA.CLEANSED\_TIER\_3, CA.CLEAN , CA.TERMINALID  
, CA.TERMINALNAME  
, CA.ORDER\_TYPE  
, **MIN**(enteredtime::**date**) **as** FIRST\_DATE\_SOLD  
, **MAX**(enteredtime::**date**) **as** LAST\_DATE\_SOLD  
, COUNT(**DISTINCT**(extract(dow **from** enteredtime::**date**))) **as** DISTINCT\_WEE , COUNT(**DISTINCT**(enteredtime::**date**)) **as** NUMBER\_DAYS\_REPORTED\_SALES  
, sum(CA.QUANTITY) **as** QUANTITY  
, SUM(CA.ITEMNETAMOUNT) **AS** SALES\_AMT\_GROSS  
, COUNT(**DISTINCT**(CA.TRANSID)) **as** NUM\_ORDERS  
**FROM** datamart.POS\_CA CA  
**WHERE** enteredtime::**date** >= '20171001'  
**GROUP BY**CA.STOREID  
, CA.STORENAME  
, CA.MENUITEMID  
, CA.MENUITEMNAME  
, CA.CLEANSED\_TIER\_1, CA.CLEANSED\_TIER\_2, CA.CLEANSED\_TIER\_3, CA.CLEAN , CA.TERMINALID  
, CA.ORDER\_TYPE  
, CA.TERMINALNAME

**Table Locations  
S3: cdl-bi-data-science/cdl\_integration/can\_volante/Featured\_Canadian\_Data.csv Redshift:** cdl-bi.datascience\_staging.challenger\_features\_can\_volante  
**ETL Manager Refresh Job**: REDSHIFT\_LOAD\_CHALLENGER\_VOLANTE\_CAN

S

K

S

**Table Fields and Descriptions Field**

STOREID

MENUITEMID

TERMINALID

ETL\_SOURCE\_SYSTEM ERP\_ENTITY\_ID MENUITEMNAME IS\_OPEN\_ITEM

IS\_PROMOTIONAL\_ITEM

**Type**

integer NOT NULL

integer NOT NULL

integer NOT NULL

string string string BOOLEAN

BOOLEAN

**Description**

See POS\_CA documentation. See POS\_CA documentation See POS\_CA documentation

Source of the item Volante Canada

Blank for now, to be populated after terminal conversion

See POS\_CA documentation

Field indicating if a menu item name is algorithm-determined to be a free input item

Field indicating if a menu item name is algorithm-determined to be a promotional/LTO item

347

IS\_EMPLOYEE\_DISCOUNT

BOOLEAN

BOOLEAN BOOLEAN

BOOLEAN BOOLEAN

BOOLEAN

float float

Field indicating if a menu item name is algorithm-determined to be a line-item deduction for employees (typically only for hospitals)

Field indicating if the item is soup

WIP, field indicating if the algorithm-determined to be a CPG item (e.g. Frito Lay's chips, Bottle Soda etc.)

Field indicating if a menu item name is algorithm-determined to be a pay-by-weigh item

Field indicating if a menu item name is algorithm-determined to be an item which typically rotates day to day (e.g. Chef's Table, Entree, etc)

Combination of each of the able boolean features. This column is used to calculate SAM (Serviceable Addressable Market) for mobile sites.

Used to prioritize manual validation of challenger features Used to prioritize manual validation of challenger features

Feature Creation (WIP)

IS\_SOUP IS\_RETAIL\_ITEM

IS\_WEIGHT\_SCALE\_ITEM IS\_ROTATIONAL\_ITEM

IS\_POSSIBLE\_APP\_ITEM

TOTAL\_LIFETIME\_SALES TOTAL\_LIFETIME\_QTY\_SOLD

**Algorithm Details**

Please refer to code walk-through for Challenger

348

Data Sophistication

349

Cleansed Tier Keyword Process

This outlines the process to upload keywords for tagging cleansed tiers on EDW. The file will be uploaded to a folder and S3, which will be read-in by the data update pipeline.

AWS S3  
All keyword files will be stored in cdl-envision->keyword-lookup-tables.  
Please save the files in comma-separated value file format with the following naming naming:

CLEANSED\_TIER\_KEYWORDS-LATEST.csv CLEANSED\_TIER\_KEYWORDS-YYYY-MM-DD.csv

The '-LATEST' will be used for updates, the date-stamped file is just for the recording historical changes.

Keyword File  
The keyword file should have the following fields:

**Description**

base\_tier The root cleansed tiers to search in. This is useful to narrow the scope of the search, allowing and helps control for keywords that jump multiple categories.

For example: Using the base Non Alcohol Beverages|En/ISO/Hy for the keyword ‘Monster’ will restrict the search space to only the En/ISO/Hy category, preventing the mislabeling of Monster Burritos.

**Field**

cleansed\_tiers

The target cleansed tier in the, with each tier split by a '|' (pipe separator).

For example:  
Non Alcohol Beverages|Coffee|Espresso Based|Macchiato

\*Spelling is very important here, spelling mistakes here will show up as different categories (e.g., Macchiato and Machiato)

|  |  |
| --- | --- |
| regex\_include | Keywords to include when searching for candidates, again separated by a ‘|'. In this case the pipe acts as an 'OR’. For example: monster+|mstr+|mnstr+ means:  ‘monster’ OR ‘mstr’ OR ‘mnstr’ |
| regex\_exclude | Keywords to exclude used to exclude candidates, again separated by a '|'. For example: burrito+|burr+ In this case, an items that includes ‘burrito' or ‘burr’ in the item name will be excluded, once again preventing the mislabeling of Monster Burritos. |

Example

The table below shows an example entry for the Non Alcohol Beverages|Coffee|Espresso Based|Macchiato category. In this case, we’re looking under the Coffee category only, to exclude and macchiato flavored chocolates. The keywords are macchiato, macchiatto, machiato, machiatto. No exclusionary keywords are included here

cleansed\_tiers base\_tier regex\_include regex\_exclude

Non Alcohol Beverages|Coffee|Espresso Non Alcohol Beverages|Coffee macchiato+|macchiatto+|machiato+|machiatto+ Based|Macchiato

Regular Expressions

Regular expressions are a very powerful tool for identifying patterns in character strings. There is a lot of room for complex pattern identification.

350

This is a good tutorial to get started:

https://regexone.com/

It is also helpful to use an only regex tester, to test all of your patterns:

https://regex101.com/

For our current purposes, we’ll start with two basic symbols: **pipe |** - this acts as an OR clause to chain all of your keywords **plus +** - this indicates one or more occurrences of the word

351

Write-Back

By Compass Digital Labs

**Last Updated: 2019-12-03**

URL: ec2-13-59-76-43.us-east-2.compute.amazonaws.com

Installation  
To install this application, first clone the repository assuming you have permission to clone the repo with

1 git clone git@github.com:compassdigital/write-back.git

Running Docker Containers  
Before starting the application, ensure the following properties are set in **.env** file in the root of the projet:

**Environment Varabile**

**PORT  
DB\_HOST  
DB\_NAME  
DB\_USER  
DB\_PASS  
DB\_PORT OAUTH\_URL OAUTH\_TOKEN\_URL CLIENT\_ID CLIENT\_SECRET CALLBACK\_URL SESSION\_SECRET NODE\_ENV**

**Description**

Port that the server will listen on Database hostname  
Database name  
Database username

Database password  
Port that the databases listens on  
Authorization URL for OAauth2 provider  
Access Token URL for OAauth2 provider  
Client ID from OAauth2 provider  
Client Secret from OAauth2 provider  
Callback URL that the OAuth2 provider uses handle the Authentication Flow Express-js Session Secret  
Enum (prod || dev || test)

The database singleton in this application creates a Pool connection to our PostgresRDS instance and requires all of the **DB\_\*** environmental parameters to be set in the **.env** file before running the application.

For questions about setting the **.env** file, please email me @ bradley.bonitatibus@compassdigital.io  
The current version runs the client and API seperately in Docker containers. To run the application, run the following command:

1 docker-compose up --build

Local Development Environment  
The development environment is described as follows:

Nginx used as a reverse proxy running on port 80  
React.js development server running on http://client:3000  
TypeScript Express server running on http://api:5000 running ts-node as the runtime. PostgreSQL database (RDS)

352

Production Environment  
Compiled TypeScript server running on http://api:5000

Nginx reverse proxy serving production build of the create-react-app on http://client:3000 PostgreSQL database (RDS)

Authentication

Authentication is provided by Compass Manager Owner’s Management Suite (OMS). They provide authentication service via OAuth 2, a protocol to authenticate users via HTTP Requests. The following image depicts the OAuth2 flow:

The web service is provided by OMS and the library enabling this functionality is Passport.js, a reputable and well tested authentication library.

In order to implement this, an OAuth2Strategy needs to be created which specifies the following:

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

**const** client = **new** OAuth2Strategy({  
authorizationURL: String(process.env.OAUTH\_URL),  
tokenURL: String(process.env.OAUTH\_TOKEN\_URL),  
clientID: String(process.env.CLIENT\_ID),  
clientSecret: String(process.env.CLIENT\_SECRET),  
callbackURL: String(process.env.CALLBACK\_URL),  
(\_accessToken: any, \_refreshToken: any, \_params: any, profile: IUserAttributes, done: Function) => {

},

);

UserModel.findOrCreate(profile)

.then(user => {

done(**null**, user) })

.**catch**(err => { done(err)

}); }

353

An undocumented feature of the **passport-oauth2** library is that the application needs to implement the **userProfile** method on the strategy object.

1 2 3 4 5 6 7 8 9

10 11 12

client.userProfile = **function** (accesstoken, done) {  
**const** url = String(process.env.PROFILE\_URL); axios.get(`${url}?access\_token=${accesstoken}`).then(res => {

**const** userId = res.data["id"];  
**let** user = res.data["attributes"]; user["id"] = userId;  
done(**null**, user);

})  
.**catch**(err => {

done(err);

});

}

This method calls the access token url and fetches the profile characteristics of the user and returns the callback with the user object. The user object is then passed as **profile** in the handler that the OAuth2Strategy uses.

Documentation

The REST API documentation follows the Open API Spec 3.0 for Swagger. To view the API documentation, copy and paste this document (https://github.com/compassdigital/write-back/blob/master/api/docs/openapi.yaml ) into a Swagger Editor (https://editor.swagger.io/) to view end points, required parameters, and expected responses from the API.

354

Deploying a new version

To deploy a new version of this application, you can SSH into the VM in which this application is running on (EC2) and run the following command to redeploy:

These can be run from directly upon SSHing into the VM as the codebase is cloned into ~/write-back . The command “spins down” the current docker-compose setup, rebuilds it with the new code, and then redeploys it in detached mode, which is essentially daemonizing the docker-compose process.

1 cd ./write-back 2 git pull  
3 ./redeploy.sh

355

Documents

356

Scroll Documents On-boarding

We're delighted that you've chosen to get started with Scroll Documents.  
This document will be your getting started guide and contains an overview of Scroll Documents' key functionality. Use this document to learn how to:

1. Create a new document  
2. Open the document in the viewer 3. Edit the document  
4. Save a version  
5. Set a document status  
6. Export the document  
7. Delete the document

357

1. Create a new document

Firstly, let's create a brand new on-boarding document.  
Navigate to the Documents app in your space by clicking the Documents sidebar item. You can find it sitting alongside Pages and Blogs.

Click **New document** to launch the **Create Wizard.**

Select **On-boarding document** from the list of options, and click **Next.**

You may choose to give the document a new title or leave the defaults as is, and then click **Create.**

358

Your document is created and will appear in the **Documents Overview.** This Overview is the home for all documents created within your space. It enables you to access, organize, classify, and filter documents from one place in Confluence.

Selecting a document in the overview will expand the **Document Details View** where you see some metadata about the document, including the title and summary, contributors, version information, and status.

359

2. Open the document in the viewer

Navigate to the newly created document by clicking **Open in viewer.**

The **Document Viewer** is a dedicated, scrollable view of all of the pages of your document. In addition to optimizing readability, you can also view and navigate your document using the outline, jump directly into the Confluence editor to edit content, set the document status, and save and manage versions.

360

3. Edit the document

Scroll down to the section in the document that reads **Edit the document .** Alternatively, click **Edit the document** in the **Document Outline** in the left navigation panel.

Now, click the **Edit** button as highlighted in the screenshot.

This will open up the Confluence editor. Make some changes to the content, and click **Update.**You will be redirected to the **Document viewer** where you will see the changes reflected in your document.

361

4. Save a version

When you **Save a version** of a document, you're saving a snapshot of that document's page structure and content at a certain point in time. You can use the versioning feature to version multiple pages as a single unit – for example, to track changes to documents throughout the document lifecycle.

To save a version of this document, click the actions menu ••• in the viewer and select **Save a version:**

Assign the version a name and optionally add a comment, then click **Save:**

The new version will appear in your **Version History** below the current working version of the document. Each saved version also contains a date and time stamp and the name of the user who saved the version:

362

Now that this version is saved, you can compare this version to the current working version or any future version of the document by clicking through the version cards in the version history.

363

5. Set a document status

With Scroll Documents, you can assign a status to your document to indicate what state it is in. By default, all documents are set to **In Progress**, but you can choose to set this to **Under Review** or **Approved**. Note that these statuses apply to the document itself and not its individual pages.

Next, click the status indicator in the document header to open the status picker, as shown in the screenshot.

Use the status picker to change the status from **In Progress** to **Under Review** and click Save.  
You'll see that the status of the document has been changed and is now also visible in the metadata of the document:

**Note**

To change the status of this document, first make sure that you have selected the current version of the document. To do this,

1. Go to the actions menu ••• in the Document viewer and click 'Version history' 2. This will open the Version History in the left navigation bar  
3. Click the 'Current' version card to select the current version of the document



**Note**

Statuses also apply to the saved versions of your document and can be updated in the same way.



364

6. Export the document

You can export your multi-page documents to PDF or Word format and use the powerful customization functionality to fully style the output thanks to an integration with Scroll PDF Exporter and Scroll Word Exporter,

Scroll PDF Exporter

To export the document to PDF, click the actions menu ••• and select **Export to PDF**:

The Scroll PDF Exporter dialogue will appear where you can select your **Template** and **Export Scope**. Since this document already consists of a page and its children, the Export Scope has been pre-defined and cannot be edited:

Select from one of Scroll PDF Exporter's bundled templates or choose one of your own custom templates and click **Export.**Your formatted PDF will be available in your downloads. For more information about Scroll PDF Exporter, please visit the documentation for this app.

Scroll Word Exporter

To export to Word, click the actions menu ••• and select **Export to Word**:

**Note**

In order to take advantage of the export features in Scroll Documents, you need to have these apps installed. If you don't have them on your system, you can try Scroll PDF Exporter and Scroll Word Exporter for free on the Atlassian Marketplace.

The alternative is to export the pages of your document using Confluence's export functionality.



365

The Scroll Word Exporter dialogue will appear where you can select your **Template** and **Export Scope**.

Select from one of Scroll Word Exporter's bundled templates or choose one of your own custom templates and click **Export**:  
Your formatted Word document will be available in your downloads. For more information about Scroll Word Exporter, please visit the documentation for this app.

Export a Document from the Documents Overview

You can export a document directly in the Overview without having to open the document in the Viewer.  
From the overview, select the document you would like to export to open the details view. In the details view, you will find the export actions in the more menu:

Follow the standard export steps through the PDF and Word Export dialogue and your exported document will be available in your downloads.

366

7. Delete the document

It's easy to delete a document or any of its versions at any time.  
To delete a document from the Viewer, open the more menu and select **Delete document**:

In the Delete document dialogue, you can decide whether you delete only the document metadata or the document metadata and its associated Confluence pages. Deleting the Document only removes the metadata itself. In order to delete the document and its pages entirely, the **Delete associated Confluence pages** checkbox must be checked.

Select the checkbox, and click **Next**.

Review the pages that will be deleted. These are the **Document Pages** which were created along with the document.  
Click **Delete.**Once the document is deleted, you will be redirected to the Documents Overview where you will see a success flag confirming the deletion.

367

References Glossary

Here you'll find explanations of the core concepts and terminology of Scroll Documents. A root page and its descendants

A page where a document has been defined

A snapshot of the content and page structure of a document at a specific point in time

The full screen viewer that allows you to view and interact with the multiple pages of your document at once

The scrollable, interactive outline of your document that appears in the viewer and contains all page titles and headings within a document

The page accessible from the space level sidebar that contains a browsable overview of all the documents in the space A flexible status field to indicate what state your document is in

|  |
| --- |
| **Document** |
| **Document root page** |
| **Document version** |
| **Document viewer** |
| **Document outline** |
| **Documents overview** |
| **Document status** |

368

Web Based Tools

1. Write Back Tool

369

Platform Architecture  
The following images describe the Data Intelligence system architecture for our data platform.

370

Airflow Architecture

371

Datameter Architecture

372

373

Redshift Query Performance

**Summary**

In this document will lie tips on improving query performance as well as define the process of optimizing queries. Some general tips for identifying slow running queries:

Run explain on your query to see the query plan to see how Redshift will execute the SQL (https://docs.aws.amazon.com/redshift/latest/dg/c-the-query-plan.html) Try to JOIN on columns that are DIST or SORTKEYS  
Make sure table stats are up to date before querying  
Avoid using where clause on columns that are not SORTKEYS

For more, please visit https://docs.aws.amazon.com/redshift/latest/dg/query-performance-improvement-opportunities.html **Query Warnings**

Redshift has several types of query warnings that it tracks on queries executed on the cluster. Each warning has a cause and also a description on how to fix the problem. All the information on this table is from the AWS Redshift documentation found at https://docs.aws.amazon.com/redshift/latest/dg/query-performance-improvement-opportunities.html

**Warning Type**

Mississing Statistics Nested Loop Join

Very Selective Filter  
Large Distribution & Large Broadcast Serial Execution

**CDL Common Warnings**

**Problem**

The Query Planner’s statistics do not match the table’s statistics Slowest possible join in query.

The ratio of rows returned to rows scanned is less than 0.05. and likely the column is not a sort key More than 1,000,000 rows were redistributed for hash join or aggregation.

A DS\_DIST\_ALL\_INNER redistribution style was indicated in the query plan, which forces serial execution because the entire inner table was redistributed to a single node.

**Solution**

Run analyze on the table to update the statistics

To fix this, review your query for cross-joins and remove them if possible. Cross-joins are joins without a join condition that result in the Cartesian product of two tables. They are typically executed as nested loop joins, which are the slowest of the possible join types.

Add the column as a sort key  
Pick a better distribution style for the table Pick a better distribution style for the table

We have setup an Airflow job that aggregates warnings per day in Redshift and stores them in datameter.table\_warnings . We also have a Grafan dashboard that displays the warning types as a time series.

Our biggest warning types are: Very Selective Filter

Missing query planner statistics  
Distributed a large number rows across the network

In the table that stores our warning timeseries data, we collect the number of minutes the warning cost and can put a numeric value on how much the warnings cost.

1. 1  **select**
2. 2  "schema" || '.' || "table" **as** t,
3. 3  warning\_type,
4. 4  sum(warning\_cost\_minutes) **as** mins,
5. 5  sum(frequency) **as** freqs
6. 6  **from**
7. 7  datameter.table\_warnings tw
8. 8  **group by**
9. 9  "schema",
10. 10  "table",
11. 11  warning\_type
12. 12  **order by**
13. 13  mins **desc**,
14. 14  freqs **desc**;

The top 5 results (as of 2020-07-24 ) are:

**Table Name Warning Type Minutes Cost Frequency** source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_detail Very selective query filter 86,341,056 160

374

source.cisco\_meraki\_router\_api\_extract source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header source.micros\_check\_detail

Very selective query filter 246 50,426 Very selective query filter 202 80 Very selective query filter 46 48

To fix these very selection, we as a team need to define important queryable attributes for each table so we can add them as sort keys.

**Fixing Broadcasting & Distributing of a table**

The query plan is the best way to identify why a query is slow. The query plan shows you how exactly Redshift will execute this query and will provide the cost of each step in the query as well as the number of bytes scanned, and the number of rows returned.

The following is an example query along with a sample query plan:

**TL;DR;**

It’s more performant have tables that join on dist keys than it is to temporarily copy the data to temp tables. The generic patter is:

create a temp table and define the distribution key  
load the required data into the temp table  
perform joins on the temp table with the proper distribution style

This does not apply to all scenarios especially ones that require complex business logic, but we should all strive for writing optimized queries.

1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

**explain with** valid\_keys **as** ( **select**

h.pos\_order\_hdr\_key,

h.erp\_entity\_id

**from**

**where**

h.record\_status != 'DELETED' and  
h.check\_outlier != 'Yes' and

h.erp\_system\_id = 1001 ) **select**

p.pos\_order\_hdr\_key, p.pos\_order\_detail\_key, p.org\_unit\_key, p.erp\_entity\_id, p.pos\_source, p.close\_datetime, p.item\_id, p.item\_name, p.item\_qty, p.pos\_product\_key, p.sales\_amt\_gross

**from** "source".cg\_dw\_cdl\_us\_source\_can\_pos\_order\_detail p **left join** valid\_keys h **on**

p.erp\_entity\_id=h.erp\_entity\_id  
and p.pos\_order\_hdr\_key=h.pos\_order\_hdr\_key

**where**

p.close\_date >= '2020-01-01';

source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header h

1 2 3 4 5 6 7 8

XN **Hash Left Join** DS\_DIST\_OUTER (cost=27493021.03..60991124901553.40 **rows**=358720715 width=125)  
**Outer** Dist **Key**: p.pos\_order\_hdr\_key  
**Hash** Cond: ((("outer".pos\_order\_hdr\_key)::**text** = ("inner".pos\_order\_hdr\_key)::**text**) AND (("outer".erp\_entity\_id)::**text** = ("inner".erp\_entity\_id)::**text**)) -> XN Seq Scan **on** cg\_dw\_cdl\_us\_source\_can\_pos\_order\_detail p (cost=0.00..29333648.00 **rows**=358720715 width=125)

Filter: (close\_date >= '2020-01-01'::**date**)  
-> XN **Hash** (cost=21542606.40..21542606.40 **rows**=1190082925 width=24)

-> XN Seq Scan **on** cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header h (cost=0.00..21542606.40 **rows**=1190082925 width=24)  
Filter: (((check\_outlier)::**text** <> 'Yes'::**text**) AND (erp\_system\_id = 1001) AND ((record\_status)::**text** <> 'DELETED'::**text**))

This is an example of a poorly optimized query because it has an extremely high total cost ( 60991124901553.40 ) along with other key indicators of bad performance: DS\_DIST\_OUTER

Hash Left Join

The root cause of this is because the outer table ( cg\_dw\_cdl\_us\_source\_can\_pos\_order\_detail in this case) does not have it’s distribution key set to pos\_order\_hdr\_key . To remedy this, you can use create temp tables, which allows you to define the distribution style / key, load data into it temporarily, and then proceed to join the tables now that both tables are joining on DISTKEY s

1 **create temp table** tmp\_valid\_headers (

1. 2  pos\_order\_hdr\_key **VARCHAR**(40),
2. 3  erp\_entity\_id **VARCHAR**(20),
3. 4  check\_outlier **VARCHAR**(10)

5)

1. 6  DISTKEY(pos\_order\_hdr\_key)
2. 7  COMPOUND SORTKEY(
3. 8  pos\_order\_hdr\_key,
4. 9  erp\_entity\_id,
5. 10  check\_outlier
6. 11  );

375

|  |  |  |
| --- | --- | --- |
|  | 12   1. 13  **explain insert into** tmp\_valid\_headers( 2. 14  pos\_order\_hdr\_key, 3. 15  erp\_entity\_id, 4. 16  check\_outlier 5. 17  ) 6. 18  **select** 7. 19  d.pos\_order\_hdr\_key, 8. 20  d.erp\_entity\_id, 9. 21  d.check\_outlier 10. 22  **from** source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header d 11. 23  **where** 12. 24  d.record\_status != 'DELETED' 13. 25  and 14. 26  d.check\_outlier != 'Yes' 15. 27  and 16. 28  d.erp\_system\_id = 1001 17. 29  and 18. 30  d.close\_date > (**current\_date** - **interval** '18 months');   31   1. 32  **create temp table** tmp\_orders ( 2. 33  pos\_order\_hdr\_key **VARCHAR**(40), 3. 34  pos\_order\_detail\_key **VARCHAR**(40), 4. 35  org\_unit\_key int8, 5. 36  erp\_entity\_id **VARCHAR**(20), 6. 37  pos\_source **VARCHAR**(20), 7. 38  profit\_center\_name **VARCHAR**(100), 8. 39  close\_datetime **TIMESTAMP**, 9. 40  item\_id **VARCHAR**(20), 10. 41  item\_name **VARCHAR**(150), 11. 42  item\_qty int4, 12. 43  pos\_product\_key int8, 13. 44  sales\_amt\_gross **NUMERIC**(18,2) 14. 45  ) 15. 46  DISTKEY (pos\_order\_hdr\_key) 16. 47  COMPOUND SORTKEY ( 17. 48  pos\_order\_hdr\_key, 18. 49  pos\_order\_detail\_key 19. 50  ); 20. 51  **explain insert into** tmp\_orders ( 21. 52  pos\_order\_hdr\_key, 22. 53  pos\_order\_detail\_key, 23. 54  org\_unit\_key, 24. 55  erp\_entity\_id, 25. 56  pos\_source, 26. 57  profit\_center\_name, 27. 58  close\_datetime, 28. 59  item\_id, 29. 60  item\_name, 30. 61  item\_qty, 31. 62  pos\_product\_key, 32. 63  sales\_amt\_gross 33. 64  ) 34. 65  **select** 35. 66  d.pos\_order\_hdr\_key, 36. 67  d.pos\_order\_detail\_key, 37. 68  d.org\_unit\_key, 38. 69  d.erp\_entity\_id, 39. 70  d.pos\_source, 40. 71  d.profit\_center\_name, 41. 72  d.close\_datetime, 42. 73  d.item\_id, 43. 74  d.item\_name, 44. 75  d.item\_qty, 45. 76  d.pos\_product\_key, 46. 77  d.sales\_amt\_gross 47. 78  **from** source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_detail d 48. 79  **where** close\_date > (**current\_date** - **interval** '18 months');   80   1. 81  **explain insert into** pbi\_operator\_assistant.new\_normal\_peak\_time ( 2. 82  pos\_order\_hdr\_key, 3. 83  pos\_order\_detail\_key, 4. 84  org\_unit\_key, 5. 85  erp\_entity\_id, 6. 86  pos\_source, 7. 87  profit\_center\_name, 8. 88  close\_datetime, 9. 89  item\_id, 10. 90  item\_name, 11. 91  item\_qty, 12. 92  pos\_product\_key, 13. 93  sales\_amt\_gross 14. 94  ) |  |

376

1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19 20 21

*-- fact table load*

XN Seq Scan **on** order\_fact d (cost=0.00..28255957.72 **rows**=223328367 width=30) Filter: ((close\_date <= '2019-01-16'::**date**)

AND (close\_date >= '2018-08-01'::**date**)  
AND ((check\_outlier)::**text** <> 'Yes'::**text**)  
AND (system\_id = 1001)  
AND ((record\_status)::**text** <> 'DELETED'::**text**))

*-- dimension table load*

XN Seq Scan **on** order\_dim d (cost=0.00..35200377.60 **rows**=457887013 width=145)  
Filter: ((close\_date <= '2019-01-16'::**date**) AND (close\_date >= '2018-08-01'::**date**))

*-- final insert*

XN **Merge Join** DS\_DIST\_NONE (cost=0.00..37256218.21 **rows**=682379503 width=588)  
**Merge** Cond: (("outer".order\_id)::**text** = ("inner".order\_id)::**text**)  
**Join** Filter: (("outer".location\_id)::**text** = ("inner".location\_id)::**text**)  
-> XN Seq Scan **on** tmp\_filtered\_dim o (cost=0.00..8750834.17 **rows**=875083417 width=588) -> XN Seq Scan **on** tmp\_filtered\_fact (cost=0.00..4.50 **rows**=450 width=126)

The total cost of these 3 operations is significantly less than. The old total cost was 60,991,124,901,553.40 and the cost of the newly implemented query is 100,712,553.53. The old query is 605,596x more expensive because it has to broadcast the table across the network and uses merge join to merge the data in memory on each node.

In terms of compute time, this optimization ran 84.1463% faster (82 minutes down to 13 minutes)

1. 95  **select**
2. 96  o.pos\_order\_hdr\_key,
3. 97  o.pos\_order\_detail\_key,
4. 98  o.org\_unit\_key,
5. 99  o.erp\_entity\_id,
6. 100  o.pos\_source,
7. 101  o.profit\_center\_name,
8. 102  o.close\_datetime,
9. 103  o.item\_id,
10. 104  o.item\_name,
11. 105  o.item\_qty,
12. 106  o.pos\_product\_key,
13. 107  o.sales\_amt\_gross
14. 108  **from**
15. 109  tmp\_orders o
16. 110  **left join**
17. 111  tmp\_valid\_headers h
18. 112  **on**
19. 113  o.pos\_order\_hdr\_key = h.pos\_order\_hdr\_key
20. 114  and
21. 115  o.erp\_entity\_id = h.erp\_entity\_id;
22. 116  **where**
23. 117  h.check\_outlier != 'Yes'

377

Datameter Documentation Project documentation for the Datameter REST API

378

Metric Calculations  
This page documents how each metric is calculated and the processes that enable the calculations.

**Terminology**

TimeFrame: enumeration of week , month or day  
Query Parameters: location\_group , brand\_id , start\_date , end\_date , time\_frame , and order\_type

Processes & Pre-Aggregation

Each night, we have an Airflow job that pre-aggregates the metrics based on specific time frames (i.e. Month, Week, Day) as defined in the original specification for this project. See Original Requirements page for specifics. The data is currently stored in pre-aggregated tables where the API reads from and returns the data as JSON.

Top-Items

**Pre-Aggregation**

The pre-aggregation process groups the items sales for a particular day for a location group + brand\_id+ order type and stores them in a downstream table datameter.p2\_top\_items .

**Production Query**

Based on the user query parameter inputs, it calculates the sum of the itemnetsales and returns the top 10 selling items in descending order.

**Average Bill Pre-Aggregation Query**

This query calculates the average check for the query parameters:

1 SUM(itemnetsales) / COUNT(DISTINCT orderid) **Production Query**

The pre-aggregation query enables fast lookups for the Average Bill query and filters based on the query parameters.

**Number of Transactions Pre-Aggregation Query**

This query loads data into a downstream table datameter.p2\_transactions where it calculates the number of transactions based on the query parameters as:

1 count(**distinct** orderid)

**Production Query**

The production query also just does filtering based on the API query parameters for faster lookups of the data

379

**Total Sales Pre-Aggregation Query**

The pre-aggregation query just calculates the sum(itemnetsales) based on the query parameters and stores them in a downstream table datameter.p2\_total\_sales

**Production Query**

The production query just filters the data based on the API query parameters.

380

Original Requirements

1. Top 3 items per station at each site across North America a. Fiscal calendar  
b. Local to the site – per brand/station specific to the menu c. JSON Response

d. Based on Monthly, weekly and daily occurrence.  
i. Users will only be able to pull data available specific to the previous week/month.

ii. Example user cases I want to support – FYI (some insight, so you understand what I am trying to accomplish). 1. As a site operator at Houston University, I want to know what my top 3 items were for last month.

2. As a site operator at Houston University, I want to know what my top 3 items were for last week.

3. As a site operator at Houston University, I want to know what my top 3 items were yesterday or a specific date.

e. Total Sales per station/brand at a site and per cost center. i. I would like to give users flexibility to view:

1. MoM 2. WoW 3. YoY

f. Average bill per station/brand at a site based on: i. Per month

ii. Per week

g. Total number of transactions: i. Per month

ii. Per week iii. Per day

381

382

API Specification

Please use https://validator.swagger.io/ and enter https://datameter.compassdigital.org/docs in the “search” bar to view our API Specification in a more visual way.

383

Development & Deployment

**Codebase & Infrastructure**

The API is written in golang and deployed onto Amazon Web Services' (AWS) Elastic Container Service (ECS) Fargate (Serverless container management platform). We are using gin framework for the REST API and using the db/sql package built into golang for our database connection.

**Logging**

We take advantage of ECS CloudWatch Logs exporter to write our container logs to CloudWatch. We also have instrumented our API with prometheus metrics and have a Prometheus and Grafana deployment monitoring the metrics from the API. For API Usage, we stream

our logs to Kinesis Firehose where they ultimately end up in our Redshift cluster for tracking usage by API Keys.

**CI/CD**

We use AWS CodePipeline to run our CI/CD process. The process is as follows:

1. All code merged to master branch requires a code review
2. When merged to master, CI/CD Pipeline is triggered
3. Runs a build process in CodeBuild where it runs our test suites and pushes our container to ECR and generates some artifacts

a. Dynamically generates new taskdef.json file with the newly pushed container URL b. Dynamically generates appspec.yml based on the newly generated taskdef.json c. Runs a JUnit report on our test suite

1. Deploys new infrastructure using AWS CodeDeploy with Blue/Green deployments. For 10 minutes, 10% of production traffic is sent to our newly deployed service and is monitored for any failures / errors. If errors occur, the deployment is terminated and the the previous version of the API is still available without downtime. If the deployment passes, the previous deployment is terminated and 100% of traffic is routed to the new deployment for 15 minutes and we can rollback the deployment if any lingering errors occur.

**Authentication**

The API uses Bearer tokens to verify users and protect routes that return data. The keys are stored in DynamoDB.

**Persistent Storage**

Each day, we pipe the data that backs this API from Redshift into PostgreSQL (RDS) to improve performance of reads.

Due to the size of the mobile orders table, only the specific units of being queried are loaded to Postgres and any new units needed to be added to the API, will require to update the Container that runs this job. ( https://github.com/compassdigital/cdl-di-airflow-containers/tree/

master/datameter\_metrics - Connect your Github account )

384

How To's  
How to add new user to API:

1. Add user record to the datameter\_api\_creds table with the following attributes: a. api\_key => UUID V4 generated (search for UUID V4 generator)

b. date\_created => ISO Timestamp when the record is added  
c. is\_active => boolean value indicating if a user is active  
d. is\_test => boolean value indicating if the credentials are for a test user e. user => username

f. uuid => UUID V4  
2. Add record to Redshift dim table for reporting

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

**INSERT INTO** datameter.dim\_user ( uuid,

username,

apikey,

datecreated,

istest,

isactive

) **VALUES** (  
{uuid **from** dynamo}, {**user from** dynamo}, {api\_key **from** dynamo}, GETDATE(),  
{is\_test **from** dynamo}, {is\_active **from** dynamo}

)

3. Give API key to new user along with link to swagger docs.

Adding more Unit IDs to Mobile Orders  
1. Update the ETL Pipeline to get the correct unit ids => https://github.com/compassdigital/cdl-di-airflow-

containers/blob/master/datameter\_metrics/queries.py#L391

Updating user’s access to view unit ids

1. Update the Authentication middleware for the Mobile Orders to specify the username + unit id slice of integers  
a. For example, if testuser needs access to the following unit ids 1,2,3,4,5,6 , the corresponding value in the Auth HashMap

should be

1 "testuser": []int{1,2,3,4,5,6}

385

Datamater Project Documentation  
This document outlines the usage guide for Compass Digital Labs: Data Intelligence REST API.

The production URL is: https://datameter.compassdigital.org . Whenever referring to the endpoint , please remember that this is the base URL. All documentation will be relative to this base URL.

Authentication & Authorization  
To access resources that require authentication & authorization, you must adhere to the following:

The HTTP Header needs the Authoization header set  
The value of the Authorization header needs to be in the form of Bearer {api\_key} .  
If you need a new API KEY or would like to gain access to other endpoints, please contact bradley.bonitatibus@compassdigital.io

API Specification  
Our API has an endpoint that returns our Swagger Documentation as JSON. You can use the Please go to

https://datameter.compassdigital.org/docs to view the document, or see the attached YAML specification.

For an interactive experience exploring the API Specification, please go to https://validator.swagger.io and enter the following URL into the searchbar: https://datameter.compassdigital.org/docs

386

This website will generate an interactive UI to explore our various endpoints, and how to call them, and with what parameters.

Usage Rates & Logging Information  
We log each API requests metadata based on your API Token to track usage to resources. The metadata being collected is the following:

HTTP Method  
HTTP Request URL  
HTTP Request Parameters Hostname & IP Address API Token

Our centralized logging system keeps system logs for 3 months. After 3 months, the log files are deleted. Any routes that call a resource that requires authenticate, have the API event streamed into our data warehouse collects events and stores for up to 7 years for billing purposes.

387

How to Annotate and API (OLD)

Source Material for Swagger 2.0: Example:  
Installation:  
Document Generation:

YouTube Tutorial Summary:

Makefile Start:  
Service layout & General Information Section: Documenting an Endpoint:

Source Material for Swagger 2.0:

Github:  
Docs Version: Sample/Viewer: YouTube Tutorial:

Example:  
Check the following link to see all of the annotations that can be utilized in conjunction with the docs.

go-swagger/fixtures/goparsing/petstore at master · go-swagger/go-swagger

Installation:  
For mac the commands are:

Otherwise check out the installation page found here: Installation · GitBook

Document Generation:

“Swagger UI is a collection of HTML, JavaScript, and CSS assets that dynamically generate beautiful documentation from a Swagger- compliant API.“

The insights engine project aims to follow the Open API Specification (OAS)-Version 2 to guide our API development. OAS now has a version 3.0 which refers to the latest standard, however the swagger hub tools are laid out and defined with OAS version 2 in mind.

GitHub - go-swagger/go-swagger: Swagger 2.0 implementation for go

Primer · GitBook

Swagger Editor

Building Microservices with Go: 7 Documenting RESTful APIs with Swagger

1 brew tap go-swagger/go-swagger 2 brew install go-swagger

388

This tool can be used to generate the actual documents for customers or users of the API endpoints as needed. Otherwise, the swagger doc can be copied into the above Swagger Editor to quickly view a preview.

YouTube Tutorial Summary:

Top-level documentation information can be found in the Docs under swagger:meta and is used to define the intention of the service. This definition is placed at the top of the file that contains the main handler or in separate documentation.go file.

Makefile Start:

Rather than having to run the swagger tool every time a change is made, ensure that the makefile of each service includes the running of swagger as shown below:

If that item is in the make file you can regenerate the docs locally by being in the same directory as the microservice's Makefile and running:

1 make swagger  
To make the Makefile more robust we can include a check for swagger and install it if it is not on the current system. Add the following to the

Makefile.

Service layout & General Information Section:

N.B → The package must be accessible by the main.go item for this method to function correctly and the entire service must have its own go.mod and go.sum.

The go.mod for the service does not need to include import paths for each endpoint and can be as simple as.

A directory name documentation should now have a dog.go and a swagger.json or swagger.yaml. This doc.go contains the general information about the service. For example:

1 swagger:  
2 GO111MODULE=off swagger generate spec -o ./swagger.json --scan-models

1 check\_swagger\_install:  
2 which swagger || GO111MODULE=off go get -u github.com/go-swagger/go-swagger/cmd/swagger  
3 swagger: check\_swagger\_install  
4 GO111MODULE=off swagger generate spec -o ./documentation/swagger.yaml --work-dir=./ --scan-models

1 module administration-service/documentation 2  
3 go 1.16

GitHub - swagger-api/swagger-ui: Swagger UI is a collection of HTML, JavaScript, and CSS assets that dynamically generate beautiful d

ocumentation from a Swagger-compliant API.

1 // 2 // 3 // 4 // 5 // 6 // 7 // 8 // 9 //

Package main Administration Service

Documnetation defining the implementation and paths for the insights engine

administration service.

Schemes: https

BasePath: /

Version: 1.0.0

10 //

Consumes:

389

Documenting an Endpoint:

We can include the code . trick in vs code from the directory that includes the endpoint. this makes it easier to look at responses and work with the code that is already existing.

N.B → As a team we have decided to put the general route information in the main.go and then put any documentation annotation in the handler section next to each func. Putting the codse next to the fucntion means that any future function changes allows for quick and easy document updates.

An Ex for the main.go:

1 2 3 4 5 6 7 8 9

10 11 12 13

// Handler just deals with routing request to the correct handlers

func handler(ctx context.Context, request \*events.APIGatewayProxyRequest) (\*events.APIGatewayProxyResponse, err

fmt.Printf("REQUEST: %+v\n", \*request)

switch request.Resource {

case "/group":

switch request.HTTPMethod {

case http.MethodDelete:

//swagger:model

// // // // //

swagger:route DELETE /group group DeleteGroupsResponse

Deletes a all of a users groups.

responses:

200: deleteGroupResponse

400: errorResponse

Where the pattern for swagger:route is METHOD /PATH tag OperationId

o

11 //  
12 //  
13 //  
14 //  
15 //  
16 package main

- application/json

Produces:

- application/json

swagger:meta

390

Data Technology Lunch and Learns

A new series of departmental Lunch & Learns for Data Technology will begin in August and run biweekly attendance optional  
Come to learn something new or share something exciting  
Please indicate your interest in presenting a topic in the table below

*or* request a topic you’d like to learn more about in the second table below See *#data\_technology* for updates and reminders

**Presentation Schedule  
Presenter Topic**

**Supplements**

**Date**

Aug 16, 2022 @Denis Osipov Time series analysis with pandas Slides

Aug 30, 2022

@david.beallor

Typing and data classes in python

Recording Slides Repo

Sep 13, 2022 @Ronald Chu BERT Recording Repo

Oct 18, 2022 @Rebecca Judges Psychology of user behaviour Recording Slides

Oct 25, 2022 @Minnie Importance of documenting ANYTHING! Recording Slides

Nov 8, 2022 @Craig O'Connor Macro economic indicators Recording Slides

Nov 22, 2022 @Brian Nguyenvo Streamlit Apps Recording Slides

Jan 17, 2023 @Reyhaneh Ghoreishiamiri Pyspark for EDA and feature engineering Slides Repo

Jan 31, 2023 @Brett McKitterick PowerBI Custom Visuals Recording

Feb 28, 2023 @Masoud Afghah Apache Iceberg Recording Slides

Mar 14, 2023 @Adi Quicksight embedding Recording Code

Mar 28, 2023 @Jacob George Active learning pipelines- how to train better machine learning Recording

models with less data

Code(reachoutto @JacobGeorge ifthe link doesn’t work)

May 9, 2023 @Denis Osipov Pandas 2.0 Recording

May 8, 2023 @Adam.Kita Web Scraping

**Requested Topics**

391

**Requested Topic Votes: (add to vote)** *[example] PowerBI gateways*

392

DynamoDB - Effective Leveraging of a NoSql DB

A collection of knowledge and resources for the proper implementation of a DynamoDB table.

Documents:

A PowerPoint presentation from the AWS innovation week. (Also covered in the videos posted below)

Packages:

Python: Golang:

**Building\_mode... d\_2.pdf**

07 Apr 2022, 04:54 PM

GitHub - pynamodb/PynamoDB: A pythonic interface to Amazon's DynamoDB

GitHub - guregu/dynamo: expressive DynamoDB library for Go

393

Data Engineering

394

Data Engineering TouchPoint Meetings

[ 1 Meeting Date: ] [ 2 Meeting Date:May 11, 2023 ] [ 3 Meeting Date: ] [ 4 Meeting Date: ] [ 5 Meeting Date: [ 6 Meeting Date: ] [ 7 Meeting Date: ] [ 8 Meeting Date: ] [ 9

May 18, 2023

May 4, 2023

Apr 6, 2023 ]

Meeting Date: ] [ 10 Meeting Date:

] [ 11 Meeting Date: ] [ 12 Meeting Date: ] [ 13

Mar 23, 2023

Meeting Date:  
Meeting Date: ] [ 18 Meeting Date: Meeting Date: ] [ 22 Meeting Date

Meeting Date: May 18, 2023 **Technologies Discussed:  
Attendees:  
Meeting Notes:**

Meeting Date: May 11, 2023  
**Technologies Discussed:** Elastic Search, AWS, Redshift, S3  
**Attendees:** @Masoud Afghah , @Vaz, Kris , @Lokesh.Kumar , @Akhila Sree Maddukuri , @Minnie **Meeting Notes:**

@Masoud Afghah : Any progress with that with AWS?  
@Lokesh.Kumar : I don't know why the UI isn't able to pick the data.  
@Masoud Afghah : It doesn't autocomplete, but if you go to the search bar and type test then it will show you.

] [ 14 Meeting Date

] [ 15 Meeting Date ] [ 19 Meeting Date:

] [ 16 Meeting Date

] [ 17 ] [ 21

Mar 16, 2023

Mar 2, 2023

Feb 9, 2023

Jan 19, 2023

:Jan 5, 2023

Dec 8, 2022

] [ 20 Meeting Date ] [ 23 Meeting Date: ]

Nov 24, 2022

Mar 9, 2023

Jan 26, 2023

:Dec 8, 2022

:Nov 17, 2022

Nov 10, 2022

Feb 16, 2023

:Jan 12, 2023

Dec 1, 2022

:Nov 3, 2022

Oct 27, 2022

@Lokesh.Kumar : I mean, I just found this right now, so if you go to that Python script, we have to make some of the changes in order to connect our snowflake. Only we have to make proper connections between Amazon and the NEO 4 J. But I'm still worried about the UI itself.

I think that rule that we might not use the batch user, we have to create a new one for testing and then we give that credentials inside the snowflake Python.

@Masoud Afghah : Yeah, the batch user has admin access, so we have to create one that is read-only.

@Vaz, Kris : So that nuclear repo is supposed to be deprecated and that all of our scripts remain within data OPS along with our pipelines. So what Anirudh was doing, was with that CDL library was since these Spark scripts lived in that repo, it got synced to the Spark scripts bucket in S3. He needed access to the CDL library functions, so he manually would zip it locally, like on his machine on his Mac, and then think it to S3 to that same bucket where the repo gets synced.

Apr 13, 2023

**Data Engineeri... 1).docx**

18 May 2023, 01:58 PM

395

Meeting Date: May 4, 2023

**Technologies Discussed:**

**Attendees:** @Dat Tran , @Hari Prasad Theniah , @Masoud Afghah , @Nabegh Abed , @Akhila Sree Maddukuri , @Sreeda.Deepak , @Minnie , @Dondre' Stevens , @Vaz, Kris

**Meeting Notes:**

@Nabegh Abed : I will explain in general the Eat Club details. So three things here in DMS, there are three main parts. You have your endpoints and this can be a source endpoint or a target endpoint like where you're reading from and where you're you're writing to.

For the migration task, you define two endpoints source and target. You define the migration task and then you give the replication instance that migration task and tell their application instance to use this task to pull the data from point A and write it to point B. Now let's go through the steps of creating an endpoint. when you create an endpoint, as I said, it could be either a source or a target. The last thing you will need to define is your migration task. In your endpoint, when you have an S3 target. Because it's an S3 and there are special, you know, settings that I can set like which folder I want the data to be written to which bucket I want, what kind of format I want the output to be. I want to include the time stamp column and additional columns in the data. So I have four columns where this is gonna do is it's gonna create a fifth column and it's gonna put the time stamp. Yeah, there are other things you can do. It's all documented in on AWS if you want to play with that, you can do that. You can also have you know tasks settings you enable or disable logs and then here.

@Vaz, Kris : A uniformed target as it currently in your example, it's S3. Are we all OK with that? Do we want to try something else? Like I don't know anything else, another database or kinesis or one of the other target dot engines?

@Nabegh Abed : We can, if we want to, you know, for example, support new real-time. If we wanna do, you know, near real-time support, then we can push it to a stream, and then from the stream we can write it to a stream. Something similar to what Akhila has been working on with the API streaming thing.

@DatTran :It'sdefinitelymorethanP2becauseifyouwannahavealookatsayPOSorderdetailandheader,there'sacolumnandthen it called POS source and you probably do a group by close date and that will give you an approximate transaction that didn't per day, but definitely more than P2.

Meeting Date: Apr 13, 2023

**Technologies Discussed:**

**Attendees:** @Vaz, Kris , @Hari Prasad Theniah , @Nabegh Abed , @Lokesh.Kumar , @Akhila Sree Maddukuri , @Dat Tran , @Dondre' Stevens , @Sreeda.Deepak , @Adith Rajiv @Masoud Afghah , @Minnie

**Meeting Notes:**

@Vaz, Kris : One of the pipelines I wrote out for the P2 dimensions-related loads or pipelines. I have tried to make my work easier by refactoring it as much as possible. So it's like fairly generic and reusable across the different tables that can get loaded from under the set of P2 dimensions. So there are 3 main files that sort of achieve what I want to have done.

**Data Engineeri... 1).docx**

11 May 2023, 01:39 PM

396

@Vaz, Kris : There are 2 subfolders, the dimensions, and facts inside of dimensions that have this group's folder. Everything from here is the same as you would have normally seen from the boilerplate generator. There's that airflow contained scripts and spark scripts-related directories.

There are three main files, a driver, a runner, and a transformer, and I was trying to like to figure out what a good analogy for this would be. the driver is as we've normally seen. It's an entry point into the sparks script, exceptional arguments, and then runs a program depending on what was passed. So when I'm passing my arguments to the runner, the constructor for the runner class, the main things that are of interest to it are the data source and data set and those data sources and data sets are tightly coupled 1 to one with what we have over here in the config repo right? The data sources these top-level files for either the configured schema And then the data sets themselves or the actual file names which correlate with a table.

Types setup that can be inherited and you know you can apply transformations based on the custom class that you create. And there does seem to be one the in the Spark ML library, but it's specific to only making 11 column changes at a time that you there's an input set of columns or an input column that you provide and it results in an output column. So you it's only specific to like making one change.

@Sreeda.Deepak : I think we have a lot of repetitive work on each pipeline. So I think it will be really good to save a lot of time.

@Vaz, Kris : We've had this discussion many times before with the CDL library and the spark-related portion of our library. There are things that we could use like CID for this CD transformer. I called it CD transformer just to be specific about you know what it's doing or the class name to be specific or logical.

@Vaz, Kris : I wanted to actually talk about something specific, not about this overarching paradigm, but I did want to bring up something to DAT and the team.

The start date corresponds to my dimensions is it corresponds to the start date of the raw files to transform from. March 28th up until today so far. So if I wanted to say reprocess the data or apply a CD in like you know recreate. Well, either recreate the table or recreate a portion of the table for the table from starting point. The way I wrote the program or the script for the transformation can enable that. So how was to choose like you know from? The pipeline is written in a way that would do that so.

The run function is going to iterate over all of the paths that are returned from. You know, the files that satisfy that date requirement, so it's gonna that transformer class has this resolve. Raw Jason's path, which is common across all of my pipelines for retrieving the source files. Well, where the date is either greater than or equal to the start date provided to the function and this is going to start applying CD which is gonna iteratively go through each path and apply the SCD. I didn't wanna make it too complicated where it reads all of the files all at once and does some. I know this is actually where data frames are amazing at doing these sorts of friend formations but I didn't want to spend too much time doing those. You know taking a data frame. To each transformation step where it's able to apply the SCD like with Windows

397

and partitions, all of that is definitely possible, but I didn't wanna overcomplicate right now, so it iteratively applies SCD as if it was doing it on a daily basis. So it'll go to April 3rd. Because I think that's useful. Useful how? I'm not sure. I don't know. I don't think we've ever actually had a case where we needed to reprocess data for dimensions, at least for P2.

@Dat Tran : Yes, I think the biggest that we, you're right, we haven't, we never really had to reprocess a lot of data or dimensional tables before, but one mainly that is because we have been doing a lot of truncate and reload on the dimension on the table. We only started recently doing SCD. So now having the ability to reprocess and rebuild that SCD table is as nice as it's a nice thing to have in our back pockets I think.

Meeting Date: Apr 6, 2023

**Technologies Discussed:** AWS, EKS, ECS

**Attendees:** @Dat Tran , @Nabegh Abed , @Lokesh.Kumar , @Masoud Afghah , @Akhila Sree Maddukuri , @Dondre' Stevens , @Minnie , @Hari Prasad Theniah , @Sreeda.Deepak , @Adith Rajiv , @Seye Ologunja

**Meeting Notes:**

@Lokesh.Kumar : The task that we were working on was removing the expired snapshots. And I've created a DAG for that. We need some suggestions on the config file.

**Data Engineeri... 13.docx**

20 Apr 2023, 01:46 PM

DAG

@Lokesh.Kumar : The DAG is just going to see if the table is present in that database or not it's going to check that once if it is checked then it's going to remove the expired snapshots.

This is just a sample table that I've created, but I might need to Add all the tables what we have in all our databases in the form of array. But we were confused. I mean we were thinking that some table might have less frequency and some tables might have a higher frequency to remove the snapshots.

@Hari Prasad Theniah : Maybe we can keep the snapshot for couple of weeks and then we can remove anything older than couple of weeks just in case we want to fall back to an older version of the table.

398

@Nabegh Abed : The question or the idea here is, when we want to have the same frequency across all tables? Or do you want to make it into something that can be configured? and How often do we need to go over that thing to check?

@Hari Prasad Theniah : Yeah, if you have an option to maintain snapshot for some sources that would be good, but default we can have a default time or snapshot OK expiring time stamp, but if we don't have that then. if you give a specific time period, then we can use that. Otherwise we can fall back to a a default value, right? So that's a good to have option I think.

@Nabegh Abed : OK, so I think instead of using the frequency we can probably Days to maintain, meaning that If I put in that value 30 days, Then you will go take that 30 days due today minus 30 days and then use that as the time stamp when you're calling that procedure. Maybe we can default it to 30 if it wasn't provided. Meaning that we always maintain.

@Lokesh.Kumar : OK, but this is going to be across all the all the tables. What we have, right?  
@Nabegh Abed : The thing that we were talking about trying, Kind of finding out is how do we how do we manage the DAG frequency, the

DAG run frequency versus the snapshots? How to expire in one to expire? Maybe we could have like a Dynamo DB table.

@Hari Prasad Theniah : I think we can have a one single config file and let's assume I if I'm developing a ad pipeline, it's my responsibility to put the table names there, right? You can do something like that so that it the developer can choose to decide.

@Dat Tran : Lokesh, for the the air flow piece that you booted up in ECS. What's the back end database that we're using for it? @Masoud Afghah : We can config that on that to have like a bigger instance or smaller one with Kubernetes. All these parameters should

be treatable, but I'm looking at documentation says supported database backends or Postgres and MySQL.

@Dat Tran : As long as I think as long as we can connect to that database and query it, we should be good.

@Masoud Afghah : Yeah, I think we should be OK because these are all docker like containers, so we can go inside each and then see what's going on there.

Meeting Date: Mar 23, 2023

**Technologies Discussed:**

**Attendees:** @Masoud Afghah , @Vaz, Kris , @Nabegh Abed , @Hari Prasad Theniah , @Akhila Sree Maddukuri , @Seye Ologunja , @Lokesh.Kumar , @Minnie , @Dondre' Stevens

**Meeting Notes:**

@Vaz, Kris : Denis had got gotten in touch with me to have a look at our S3 usage in terms of cost. I noticed that at the time, The CDL BI raw source bucket was the largest player in terms of usage of S3. That was because there was versioning enabled on the bucket, especially in the context of P2 a lot of files that get rewritten every hour. For 10 days, at least for the context of our daily extracts that accrues a lot of storage very quickly. So my solution at the time was, Turn off the versioning on the bucket and prune. Set up a lifecycle policy on the bucket to prune out all noncurrent versions older than one day just to like. Clear out all of the noncurrent versions. And that fixed it.

Now the biggest sort of heavyweight in bucket usage or account usage in terms of buckets is our iceberg data leak bucket. I looked at a couple of our buckets and our account at that time, you know, reduced our storage to like 60 terabytes. So this is big. The end result of that particular prefix within that bucket had grown to and I sold. I have the files sort of estimate the that have like you know the number of files and the size of like the storage used within that prefix.

**Data Engineeri... 06.docx**

13 Apr 2023, 01:55 PM

399

@Nabegh Abed : Lokesh is actually looking into that. There's a bit of a learning curve for him. So he's just getting familiar with Iceberg and all the details, so I'm helping him on that part. This problem you described is an iceberg problem, not an S3 problem, right?

@Vaz, Kris : Before it was an S3 versioning issue that we were OK with removing versioning on that bucket and now it's the iceberg version of versioning via snapshots.

Meeting Date: Mar 16, 2023 **Technologies Discussed:**

**Attendees:** @Vaz, Kris , @Masoud Afghah , @Minnie , @Hari Prasad Theniah , @Akhila Sree Maddukuri , @Lokesh.Kumar , @Dat Tran , @Seye Ologunja , @Sreeda.Deepak , @Dondre' Stevens

**Meeting Notes:**

@Vaz, Kris : I think we agreed last time that this is probably best enabled via PRs on GitHub. I think that would be via feature branches and then like using feature branches as a way to create a PR. I'm thinking that Dev might be more tricky for peer review; for the dev, layer to have this PR strategy. I think the easiest would be on staging, cause that already is well documented for doing that.

@Masoud Afghah : So there's this thing in GitHub, it's draft PR. We can create draft PR and we don't have to make the PRs in need of approval so that they would be merged into Dev. We can just push and you can have like direct push to Dev, but once you wanna merge it to production, that's when you would make the PR. So before that, how how do the branches work right now if I create a branch, it's not going to show up in the dev environment, is it?

@Vaz, Kris : Let me try to find that document and explain at the same time. Like your feature branch, you create a feature branch out of staging and to test your changes you essentially. Commit your staging or your feature branch changes into dev to like run tests against and then once you're ready. You know your feature branch is ready. It works in Dev, then you promote that feature branch as a. A PR for your feature branch in the staging and then once that's approved then you know it gets into staging and then a release cycle staging gets released to production.

@Hari Prasad Theniah : The thing is right, so all our work is independent, right? It's not a single application. So every once work is interdependent.

@Vaz, Kris : if you read through the document I just posted. For testing out your changes in Dev, you're not going to merge your branch. Your feature branch into Dev. You are going to copy those files you've changed. Hopefully, they're easy to navigate. You can do it at a folder level or specific file level into Dev, which is something that would go along the lines of your checkout into the dev branch and then you do the Git Checkout feature, branch, Dash, Dash, and then the path to your files. Your updated files. That way it doesn't actually bring staging into Dev at all, it just brings the things you've changed or you want to have tested into DEV. And you repeat that process as many times as you want.

@Akhila Sree Maddukuri : We can also do cherry-picking, right?

@Vaz, Kris : If that commit happens to touch many files that you don't want to necessarily have in dev, of course, that is a possibility. You could do that. Actually, that may be simpler for many use cases, because you tend to just work on your own things. So you just cherry-pick the commit into Dev.

@Akhila Sree Maddukuri : So one control can be when we do a cherry picking, it can be like a PR or something where somebody just checks whether other files are getting affected. And if it's not, then it can be just uprooted and pushed. Wherever the sync needs to happen, just to make sure that other files don't get affected.

@Vaz, Kris : I mean off the top of my head, the immediate cost of that would be like slowed down development time because now you have to wait on someone else to approve your PR just to test or changes in Dev.

**Data Engineeri... 23.docx**

31 Mar 2023, 04:15 PM

400

@Seye Ologunja : They actually write codes on locale then raise a PR, get the PRs approved and once it's approved they merge to developments, and from there two people have proved. I don't know if that is what we are trying to do here. Two people approved in the process of two people approved they are aware of the code, and they understand the code. If there is a need to change anything there will be able to work on it. After managing staging to Dev Environment there, the test engineer will test if everything is good, and they move to staging after moving to staging they do regression testing. Then they move to production, but I know it's quite different with data so I'm just sharing this example in case you see if it fits in or not.

@Vaz, Kris : I mean, I understand. Like I've gone through that same process as well and that is a very standard process. It just doesn't we could use it, but it would just slow down our development time for testing changes because a lot, unless we have a local deployment version, enabled where we can test everything locally on our machine without having to you know.

Meeting Date: Mar 9, 2023

**Technologies Discussed:** Redshift, Iceberg

**Attendees:** @Nabegh Abed , @Hari Prasad Theniah , @Minnie , @Vaz, Kris , @Dat Tran , @Masoud Afghah , @Lokesh.Kumar , @Akhila Sree Maddukuri , @Adith Rajiv , @Sreeda.Deepak

**Meeting Notes:**

@Nabegh Abed : Talk about the iceberg problem that Kris and I worked on fixing.

@Vaz, Kris : I was working on backfilling their P2 order data from redshift back into our iceberg table that we have for it. So a while ago. Late last year, I unloaded the redshift table from P2 orders, where the pickup time stamp was. You know, up until essentially the beginning of time, which was like 2018, January 8th, I think up until November 30th. unloaded that data in parquet format into S3 which was done actually late last year. I ran into issues when I was trying to do this back last year and I parked it until now. With that unloaded data, I loaded it into an iceberg table called P2 orders unload. I'll just I'll drop the P2 prefix so there was a table in iceberg called orders to unload. Now the orders table is in Iceberg. I made a backup of all the data, so I select a star and created a new table called errors backup.

Now the application that we were or that I was trying to run or the logic that I was trying to run was essentially a merge into a very simple SQL query supported by iceberg where I'm merging the orders unload table into the orders backup table joined on the order ID column. Initially, I joined on both the order ID and pickup time stamp and there was a little bit of a transformation on the pickup time stamp to convert it to date. Format Instead of timestamp format and my initial thinking for that was since the orders backup table like the orders table is partitioned by date, I figured that it might have been simpler for the spark query planner to find matching partitions or do you know find the partition to try to match against.

I ran that job without initially providing any configurations to the spark application, meaning just running by default, at which point we ran or. I ran into an out-of-memory issue. The program or application had exited with 137 status, which is an out-of-memory issue. Tried to increase the memory, but wasn’t able to figure it out. After reaching out to Nabegh, trying to increase executor memory.

There was a select from either state, uh from both tables a merge, and then a subsequent select from the orders unload table again and then emerge again which seemed redundant. But what was interesting was looking at the partitions. Reduced by default. Initially, the tables were read into 1000 partitions each and then there was an optimization step as part of the spark where you plan that would reduce the partition size.

@Nabegh Abed : When you go and try to maybe add more exacters, maybe increase the memory limit for each executor. Sometimes partitioning the data into smaller partitions. Sometimes the data is skewed. I was checking something and I saw that the driver was receiving those events from the executors and the driver was taking like sometimes up to 100 seconds to process those events. So the executors in Spark do send a heartbeat. Along with these heartbeats, they record status. So these are the Spark event, and the history events and The driver taking all this time to process these events is something that shouldn't happen.

**Data Engineeri... 16.docx**

23 Mar 2023, 01:11 PM

401

We shifted our focus from the executors to the driver. Now, when looking at the driver, we figure out we figured out that this memory limit is actually a driver issue, not the exacters. The executors were fine, but the driver was going out of memory. We tried increasing the driver limit memory limit. That helped a bit, but it was still failing. Tried to optimize the number. There's a queue that the driver. We tried to decrease the size to see if the driver is being overwhelmed by a large number of events and that's why it's taking too long to process them. That did not change anything, which means that that's not the case. We so we kept trying to look at that. The last thing we did was to just, you know, get a heap dump from the driver and see what's going on in there. Why is the memory increasing and why is the driver CPU is going up to 100%?

We were just leaving Spark and iceberg to just use the default and it was doing 500 partitions. And 500 partitions for each partition of that date field, which means 500 we have around the I think 1800 of those dates so was 500 times that 1800 or was approximately the number of output files that this job was trying to produce. So we limited the number of output partitions we set the parameter in the spark job to use 10. And once we did that, then the job was completed in like 5 minutes.

@Hari Prasad Theniah : I faced a similar issue. I was trying to partition based on a column, but when I didn't have the right column to distribute. so I started using bucket partitioning so it's based upon a column, right? So basically it's a hash partitioning. It brought down the number of files and partitions. So because Maraki’s job had billions of records, so I think it was very huge. So I was able to fix it. So it took some time for me.

@Vaz, Kris : Yeah, that actually brings up a good point. Uh. So you were saying there was the site that you partitioning by and then followed by the date or the transaction timestamp, the order of the partitioning also matters if you did it the other way around, it would look very different as well, which is what I was doing for Pete. For the P2 orders, I was partitioning by pickup time stamp and then by app name. But the other direction would have made even more partitions and even more files. There are a couple of parameters or table configurations for iceberg for an iceberg table specific as you know in the topic of this discussion, I wanted to bring up the right related configurations.

I'm curious to see how much of a difference if I was to set that configuration back. If we were to rerun this job, one of the configurations I'd set was what's called a fan-out writer. So when we were testing, when we were using iceberg last year, and when I had implemented the P2 orders, I think I might have run into this issue with tasks 1st P2 tasks, but I know I ran into this issue with P2 orders. I think we were using 0.12 or 0.13 the version of Iceberg.

@Hari Prasad Theniah : I always write my data frame to a staging file and I do spark SQL to do the merger insert right? So the sorting is taken care of by the Spark SQL so I don't have to express it explicitly, and sort the data frame. So when I create a staging table and then I do insert it into my target table from this staging table.

The initial extract dense stamp and then the modified timestamp so it looks more like slowly changing dimension of type 0 or type one I think so without reading the history but it just captures when the first record arrived and then if there is no change to the record from the first date till the till today so there won't be any change to that record. Iceberg is doing that really well. It's not touching the partition if there is no change. It's really powerful compared to what I was doing with Redshift, and also to do this kind of processing redshift, it will take lots of CPU and memory. But Iceberg is doing that very easily.

**Data Engineeri... 09.docx**

16 Mar 2023, 02:28 PM

Meeting Date: Mar 2, 2023 **Technologies Discussed:** Iceberg, Snowflake, AWS

**Attendees:** @Masoud Afghah , @Dat Tran , @Vaz, Kris , @Nabegh Abed Maddukuri , @Seye Ologunja

**Agenda:**

@Hari Prasad Theniah , @Lokesh.Kumar , @Akhila Sree

@MasoudAfghah DiscussIcebergtablesnapshotretentionpolicyasitaffectsperformanceandleadstogrowingcostsifwedonot expire them.

402

How to do it  
Frequency (based on SLAs)

**Meeting Notes:**

@Masoud Afghah : Discuss the Iceberg Table snapshot retention policy. We can set them to expire and then that's gonna improve performance. I don't know if we're suffering any performance issues right now.

@Vaz, Kris : Can we compress the tables into one file?

@Masoud Afghah : I can only assume that like every time we make changes, we update the table we write new data to it. It creates a new snapshot. And if we don't expire them. I hope there's a default number of snapshots that it keeps, otherwise, they might have been growing and as we write more to them than it is, it's going to be slower.

@Vaz, Kris : Most of the distribution data storage is being used across our account on S3 and the majority of it was in CDL BI Ross source. And the reason for that is that we enabled versioning on that bucket and for a lot of the P2 jobs, like between Hari and me.

We don't have anything in our process right now to reprocess old versions of files. We only ever look at the current version. But, If you do want to do something like that, technically speaking via iceberg, we have that feature to do time travel to reprocess or to look at, you know, previous versions of data which would capture essentially what we had in this S3 versioning.

@Masoud Afghah : But if we want to keep different versions of that, is there a need for that as well?

@Vaz, Kris : They don't know how much versioning we have on the tables and really the amount of versions per table depends on how often that table gets updated or if anything happens to that table. So like if there's one update per day then we'll have one version per day. But if there are like 10 or 24 updates per day then we'll have 24 versions every day.

@Nabegh Abed : I think we can set up a job that runs maybe every day, and this job will basically have a config file. You can put filters and then you can put like target file size or something like that that we can build some kind of a YAML or Jason file and we list all these and then that job would go and read that file and based on the entries there would go and run this maintenance.

@Hari Prasad Theniah : You can just enable auto-delete after a certain number of snapshots.

There is an option when creating the iceberg table right, we can set the table properties to delete after the commit is enabled. So, it would retain a certain number of snapshots. We can set the number so this process doesn't have to be manual so it can be implemented during the creation of the table so that automatically when the number of snapshots exists that limit it will tell the oldest file it seems.

@Hari Prasad Theniah : To connect Spark with the DB to the client, I think it was working only with the previous version. It's not working with the current version of Spark. I don't know. I tried to fix it but it didn't work for me. So there is a Yammer cluster that's running for the DB connector. I think if we can come club that with the managed cluster it will be really helpful.

@Masoud Afghah : There is a script that has to be run on the, it's like a thrift server.  
@Hari Prasad Theniah : What's that spark Library file is not compatible with the thrives file, so this David enter. Yeah, that's the version.

There is some kind of mismatch with the libraries that are not working, so it's worth it. It is just work working only with this version of Spark.

@Akhila Sree Maddukuri : I just wanted to ask so altering the names in the iceberg tables should not be a problem, right? Or does it affect the schema involution or something?

@Vaz, Kris : It doesn't affect it in the context of the iceberg, but if you change a table name and you're ever referring to those specific S3 data files for that table, it does matter because the iceberg won't change. It'll only change it and its metadata files, so when you're like querying iceberg or you're writing a data frame from iceberg to somewhere else.

@Dat Tran : Do we have a masking solution? Think we can implement an iceberg for what to do in Snowflake.

@Hari Prasad Theniah : I think if we have a solution that's standard across.

@Dat Tran : And we decrypt in Snowflake without going to Iceberg? Could there be a PDF or something in Snowflake that does the decryption for us?

To decrypt the data and if they do need to decrypt the data, maybe they go through us and then we load a temp table into snowflake for them, and then after a while, we delete that table. We just store the so we we will not even put the, we just need to mask any like a couple

403

of PII columns nowadays. So like the last four digits of the credit card number, we should mask the e-mail address we should mask. So I'm thinking we masked the data when it hit Iceberg and then we just push that mass data over to Snowflake. Already mass and whatnot and hash. And then any analysis in Snowflake will be on that hash data and that will work for like 90% of use cases.

@Nabegh Abed : Do the masking and then maybe we can create as you said that function in Snowflake that only privileged users can access that function to do the unmasking.

@Dat Tran : If it's a function that Snowflake, you'll probably run a lot better too because it's hope it's not running row by row like it's reading through it.

@Nabegh Abed : I don't think we can go ahead and delete the third column from the schema because that will mess up the older data that we process.

Meeting Date: Feb 16, 2023  
**Technologies Discussed:** Airflow, EMR, EKS, S3, Redshift, Snowflake, phData Toolkit

**Attendees:** @Nabegh Abed , @Dat Tran , @Masoud Afghah , @Seye Ologunja , @Lokesh.Kumar , @Minnie , @Akhila Sree Maddukuri , @Hari Prasad Theniah

**Meeting Notes:**

@Nabegh Abed : Lokesh, has created a script to start Airflow, NWA, EKS, EMR, and S3 buckets.  
@Hari Prasad Theniah : The thing is, right, without the public bucket, we won't be able to push the Data from our current account into that

account, right? So because right now all our data is outside the compass network, so we need at least one bucket maybe.

@Nabegh Abed : Yeah, I think there's a cross-account sharing that can be done. I don't think that requires the bucket to be public.

@Hari Prasad Theniah : I'm not sure about that thing, but if it is possible to do cross-account bucket access without Public, then, that should be good.

@Dat Tran : So can we use that AWS family transfer thing as the intermediary? So that will always be publicly accessible. If we drop files in there and have a sync to an S3 bucket like that, that should work as a bit of a roundabout way to do it.

@Hari Prasad Theniah : we are unloading a bunch of data from our redshift into an S3 bucket for Distilr, so it can be synced to the Snowflake.

@Dat Tran : But unless we need it, the plan is to not use it at all. If we still need it, then sure. But if we don't need it then I'll contract is finishing at the end of this September. Our three-year terms end when we're not gonna renew for another three years. So if we need it for a couple of months, we can keep paying for it.

@Lokesh.Kumar : I think we can still create cross-account S3 public buckets. But for that, we might need a bucket policy that allows us public access. And after giving that, probably we'll need the IBM role, which grants access to the other account that we are trying to work on. So probably once these two steps are done, I think we can just grant access to other accounts.

@Hari Prasad Theniah : Public just for data movement between another account. So if there's no if there is no data movement between two different accounts, one in public, and one in private, then we may not require public access at all.

@Dat Tran : Well, I mean if that's the case, like have we, if we need a movement of data, let's just use the SFTP.

@Hari Prasad Theniah : The fact is right that the SFTP push will be won't be as fast as history I think. So pushing 2 or 3 GB files SFTP push won't be that fast.

**Data Engineeri... 02.docx**

16 Mar 2023, 02:29 PM

404

@Lokesh.Kumar : We were working on three different resources EMR, EKS, and Airflow. The first one is the MWAA, like the Apache Airflow yaml, so it's just like creating an MWAA environment along with the required AWS resources such as an S3 bucket, security groups, and some other resources. So mainly the parameters that we've set but these are according to the compass standards so we are following the compass standards.

@Dat Tran : We wanted Different roles for the AWS account, so obviously, we don't want everyone to be admin. Maybe we have an admin role, maybe we have like a data engineering role, or we have a BI role because I think for single sign-on to work you have to get those rows created within the ad group and then have those ad groups associated with the role in AWS. So you know when you log into AWS, it's asking you to pick super admin or admin within our account right now.

@Dat Tran : I wanna get your opinion on proposing setting up roles in Snowflake. Set up access role and then we would have functional roles. So, for each database or your schema we can have access role of read, Rewrite. You could then have those roles assigned to a data engineer, a BI analyst, a compass group, Canada analyst. But then what that allows us to see is it, if you go into Snowflake it will allow you to see all the permissioning that a particular user or particular role has without diving too much into the data.

Meeting Date: Feb 9, 2023  
**Technologies Discussed:** jfrog, EKS, EMR, GitHub, S3, Docker, Airflow

**Attendees:** @Masoud Afghah , @Dat Tran , @Nabegh Abed , @Sreeda.Deepak , @Adith Rajiv , @Akhila Sree Maddukuri , @Seye Ologunja , @Lokesh.Kumar , @Minnie

**Agenda:**

@MasoudAfghah CheckingPython(andmvn/sbt/etc.)packagesforsafetyusingtoolssuchas: **Safety** )  
**pip-audit**

**Meeting Notes:**

@Masoud Afghah : Looking to install the PIP packages and check the packages. One of the places where the packages can be installed is in Airflow. That's one place where we could run this command. It's just one command that runs and then checks all the packages that you have installed.

@Dat Tran : Nobody's adding packages directly to the requirement.txt and would this be part of the docker that Kris built for the Boilerplate?

( Safety CLI - Security for your Python dependencies

**(** GitHub - pypa/pip-audit: Audits Python environments and dependency trees for known vulnerabilities )

405

@Masoud Afghah : So I think when we run the GitHub actions and it syncs it with S3 and uploads the code on S3, that's where we can check it. So technically it would run on GitHub. So GitHub actions you have like different steps like Packaging the repository as a zip file and then uploading it. There's this would be all before the Docker image gets built and run.

@Dat Tran : I think everyone here should have access to this jfrog tool, The way we do CI CD, is gonna change moving forward. So right now everything is in GitHub, but I think the flow goes for it. We have to check in the artifacts through to this jfrog platform and then from here, it's gonna we will have a CD pipeline that would get this data over to the new data bias account. This will be a tough spot, like that CD part would be a good spot around your checks.

@Nabegh Abed : The tool that Masoud is suggesting it's gonna be set between our request and the response we get from the web and it's gonna try to filter any kind of known or suspicious packages.

@Dat Tran : We're gonna have to migrate, Umm, a couple of things over anyway, so be the place to test it.  
@Nabegh Abed : There is a tool When we migrate to the new account, that does code scan, Not package scan I don't know if they do

package scans.

@Masoud Afghah : I think the main advantage of jfrog is to control where are you reading. Pulling your dependencies from. Instead of going to the Internet and pulling things from there, you would host them locally and you would know what you're pulling because you are the one who uploaded those packages into the artifactory.

@Nabegh Abed : Next week, maybe we can set up a meeting location with our team just to review where we are and what we have. And then after that, we can discuss this with Tom this Monday we're gonna do the same thing with him and Gabe.

Meeting Date: Jan 26, 2023

**Technologies Discussed:** Collibra, AWS, GitHub, Apache, DBT, Snowflake

**Attendees:** @Dat Tran , @Seye Ologunja , @Sreeda.Deepak , @Vaz, Kris , @Hari Prasad Theniah , @Akhila Sree Maddukuri , @Minnie , @Nabegh Abed

**Meeting Notes:**

@Seye Ologunja : Discussed the Definition of Done on a Miro board with all the team members.

The idea is when you're working on data migration. When will you move that tickets? What are the things you would have done for you to say you are done working on this particular or on the particular migration you can just list them out. Manage everything to get that. Then if you have a new data source, if you are doing ingestion, all those things when will you see you are done doing them? At what point would you say you are done so everything you write it down also? Had we had different kinds of support requests but which one? Can you remember when we say we are done?

@Dat Tran : Update on that AWS account. We did speak a bit well that last time. So we did find a new Rep. So hoping to get a meeting set up with him next week sometime and we can talk through the MWAA cost and all that.

@Masoud Afghah : Discussed Collibra: Data Catalog, Data Governance & Data Quality

@Dat Tran : One of the key things is to track dependencies; especially with the data leveraging piece. Track the dependencies of different jobs. Like this airflow, DAG is loading to this table which is feeding into these many views and is feeding into like 3 views and one DBT model. And that's building this table and basically being able to track it. From the DAG level all the way to ideally the looker model level or the power BI dashboard level. But you know we can stop at Snowflake if we need to.

@Dat Tran : Made a quick ticket with this just so we can not forget about it. And then when you have some spare time. Let's get up and running. Ideally in the ECS cluster, we have this new account we'll become more of a KS shop like they would have done anything. So that's gonna be where we're hosting the application. Get away from the CT side.

406

Meeting Date:

**Technologies Discussed:  
Attendees:** @Dat Tran , @Nabegh Abed , @Masoud Afghah , @Vaz, Kris , @Adith Rajiv , @Hari Prasad Theniah ,

@Sreeda.Deepak , @Minnie **Meeting Notes**

@Dat Tran : Update: A bunch of data source development and another potential streaming feed coming our way.

@Masoud Afghah : Working on the DL master data for the SAP table, so I've loaded dash, and then Tom sent me a message saying how he was able to connect to this code database on the Informix point. So yeah, I tried connecting to that. He actually connected using JDBC because of the guide that stack. I think the BBC. Yeah, but I'm able to connect to that using DB or on VPN, but I'm trying to connect to that using Python.

Whenever we have time to discuss the streaming application that we have. I wanted to explore using Flink.

@Vaz, Kris : Nabegh did put together a sort of diagram of all of that, we didn't select Flink, I'm not sure if there was an explicit reason, but the architecture setup or POC removes the requirement to use a spark structured streaming.

@Masoud Afghah : With Flink, we can use kinesis like data analytics. The WS service, we can use that for Interesting the data so it will be easier than having everything through infrastructure for hosting. But also I think it's more capable, so maybe if in the future we're gonna have more training data, we can get started with that.

@Vaz, Kris : I'm pretty sure Flink is the go-to stream processing engine these days.

@Masoud Afghah : And if it's like other capabilities, so one of them is that if your application gets billed for any reason, you'll just pick up from where it left off. So like even if you have like spark job that doesn't get interrupted. So and you know you have a lot of like problems there. Then you can use and are you gonna read that some companies use Flink because they want to? We are your talking section and then they'll just have their job around on the plane and whatever spot instantly get killed. They'll just find another one. Or even if there's buttons are not getting killed, they'll just constantly look for cheaper spot instances. So they'll intentionally like kill their applications so that they would run on the cheapest parking.

But the link allows them to do that because it'll pick up where it left off.

@Dat Tran : We try Spartans with Spike and without an AWS account right now would they buy the saving plan and whatnot, which actually covers the regular instance, but it doesn't look like it covers spot instances. So if we go with Spot, the bill would actually jump up a bit more.

How would you host it though? What it is it hosted on an EC2 machine or is it hosted on the EKS? How would you host it?

@Masoud Afghah : We can have, like a job posted, but essentially like KD a kinesis data stream which is an AWS service. You can just upload your print link applications into a jar file. We just uploaded it to that and then they will take care of scaling it up and down.

@Nabegh Abed : That instance Hari, you showed me the other day you connected to through VPN is that related to my dining?

@Hari Prasad Theniah : No, that's for Silos. Mostly for internal conversations.

@Nabegh Abed : Because I did reach out to Tom to ask him about, how they got that machine set up, he gave me a lot of details, and he was saying that they do have a few of those like for example, my dining. Routing people through a single host. But he said these are windows instances, is that one the one that you connect to? Is it windows?

@Hari Prasad Theniah : Maybe they route the access through Windows because it's an AAD account. That could be the Windows route, but where we are logging it's an Amazon Linux. Maybe you can share the first name, so we need to set up something.

@Vaz, Kris : Random Fact; I found out recently that in Snowflake or Redshift if you're doing a merge and if there's a match from your data set to the table and you want to update the row in the table. You have to update all the columns and if you're to do it in snowflake, you have to specify all of the column names. But in Iceberg or Spark SQL, the users have to specify or you just have to write out the update set \* and it updates all of the fields. So if anyone's doing any on the Iceberg side any explicit column setting specifically in the case where you're updating all the columns to make your life easier, you can just Pass an asterisk.

Jan 19, 2023

407

I ran into yesterday an issue when I was loading Snowflake. A temporary table in Snowflake for the FoodBuy stuff. I realized that create or replace When passed through the spark application via the connector to Snowflake doesn't actually do a create or replace, it does. It does it create if not exist. When I looked at the logs for Snowflake I noticed that it doesn't actually replace the table. So what that looked like when I was running my Spark app was that when I was loading to this temporary table, you know on 2 subsequent runs it was actually not recreating that temporary table. It was just using the same and for whatever reason, I guess the session was still live. The temporary table wasn't dropped. And so it just appended data onto that existing temporary table.

So just watch out for that. If anyone doing a similar sort of logic using the temporary table, the workaround sticks precisely to drop that table.

Meeting Date: Jan 12, 2023

**Technologies Discussed:** Gather, AWS, EMR, GitHub, EMS

**Attendees:** @Masoud Afghah , @Hari Prasad Theniah , @Vaz, Kris , @Sreeda.Deepak , @Dat Tran , @Minnie , @Lokesh.Kumar , @Robert.Inkpen

**Meeting Notes**

@MasoudAfghah discussedtheuseofGatherandaskedeveryoneontheteamtojoinandusetheforumtohavebreak-outsessionsand discussions.

@Vaz, Kris : David found the Boilerplate document on Confluence useful for DBT. Got him set up with that. I had to update the airflow BI instance to have the required. David brought up a question about the ECR images between Dev and Prod before I even got a chance to tell him that there there is an issue if you use the same name between Dev and Prod and you make a change. You know, and you're container scripts in Dev. If the names are the same, it'll also affect your prod image. So I told him I think we discussed this a while back here at the touchpoint, but I think the easiest course of action for right now is to just suffix your dev code or your dev image with like an underscore dev.

@Dat Tran : Are we gonna do that as part of the boilerplate or would that be something that the developer has to kind of set up?  
@Vaz, Kris : It can easily be added to the boilerplates during the PR process just from anywhere into staging. At that point, we all look at

that cause, that code that gets merged to staging and thereby to prod would have to have a different image name.

@Lokesh.Kumar : Had a call with Gabe, they asked to follow the templates that they have, but it's already in TypeScript. Actually, I thought of using the same libraries in Python as well, That will be able to support the TypeScript libraries that will not be able to support Python. So, I'm trying to figure out ways that I can migrate the icons. One is that either I have to write the code in TypeScript instead of Python or maybe the libraries already we have in TypeScript I have to import into Python And we still write the Python code using the Python libraries that I have written. We have the templates for the IM and S3 and some of the services but I did not have a template for creating an airflow yet. Tom’s working on and he might be giving the template, maybe today or the next Monday.

@Dat Tran : Is it feasible to use a TypeScript library with Python TypeScript is it getting compiled into some format or is it just a TypeScript? @Lokesh.Kumar : I've also reached out to Justin regarding that if he can provide me an example where we can import those types of

scripts to Python.

For the Airflow one, we still have to create a new S3. I'll still have to use the three libraries for the airflow as well. So firstly, I'm trying to import, I'm trying to migrate one of the dev Dev accounts from the airflow and migrate it to the sandbox account. And once that is done, probably the next two environments would be easy for me.

@Dat Tran : What are the requirements?

@Nabegh Abed : S3, the subnet, the security groups, and the roles. EMR EKS, subnet security group.

@Robert.Inkpen : Question regarding the CI/CD process in GitHub for the new CCNA accounts. Or like what is your process for the gonna be for that?

@Dat Tran : We're going to move away from GitHub, as your DevOps pipeline, that's kind of the directions we're getting from Gabe. And the enterprise architecture team wants us to lay on top of that, that code scanning service. The CI CD stuff will have to be run as an ADO

408

pipeline.

@Hari Prasad Theniah : If we are all going to use that machine to do development, we need to have a separate development path right each developer should have their own path so that.

@Masoud Afghah : If we can get the AWS access key and secret that also expire or like rotate with the same frequency?

@Dat Tran : I think like I think that's a good setup. I don't know if Compass has the infrastructure to support that right now because from my understanding, I talked with Gabe and Tom a while back and they're trying to push security into this direction where because right now like if we log into that account, we can't even see the IM tab apparently.

Meeting Date: Jan 5, 2023

**Technologies Discussed:** Snowflake,

**Attendees:** @Dat Tran , @Vaz, Kris , @Nabegh Abed , @Hari Prasad Theniah , @Minnie , @Masoud Afghah , @Seye Ologunja , @Lokesh.Kumar , @Adith Rajiv , @Akhila Sree Maddukuri , @Sreeda.Deepak

**Meeting Notes**

The focus for January would be to get Snowflake up and running.  
AWS accounts cannot be a part of the Compass Firewall as of now since there’s no known way for Compass to bring the Snowflake instance into the network. If we go with the private link options any resource coming from AWS will be fine.

@NabeghAbed discussedtheabovediagramonhowwecandoIngestion,monitoring,anddataquality. We have jobs scheduled in the airflow. Different data sources come from different formats.

We pull those jobs and put them in our landing bucket, called source. And then from there, we have a job also scheduled an airflow that triggers an EMR job to read that through data and write it to Iceberg. We also take a copy to Snowflake.

Through this what we want to know is:  
The ingestion monitoring  
Whether the job failed, or it was completed successfully as expected The data quality

The idea is to do an entry. We can do exit or we can do exit only. Or we can bake it inside the task that's running in the airflow. And there are a few different ways of doing it. Each one has advantages and disadvantages. But the main idea is to have the ingestion job report some metrics to a database. And in that database we can generate from that database we can generate reports and do some kind of daily summary.

On the EMR side, we can do data quality. To the same database, so we can have a full view of that data source. So the thing we have a data source that's coming from the SFTP server. You can report how many files we processed, and how big is the data. And some other, you know, aggregates. And then once we process that data, then we can also report on the quality of the data itself.

409

@Dat Tran : What kind of information would we get If we were to connect to the Airflow POSTGRES database @Nabegh Abed : We can utilize the airflow database, we will only get the job status.  
@Dat Tran : How do we track and maintain the SLAs?  
@Nabegh Abed : The table refresh is gonna happen in another DAG, so we can also have SLA on that.

@Dat Tran : One of the big requirements will be this entry access stuff. If we can build it in a way where no as we onboard other folks from Andy's team and we have like a baseline level monitoring across every single day in our environment.

Meeting Date: Dec 8, 2022

**Technologies Discussed:** Monitoring API, Data Lineage, Data Catalog, Airflow on EKS vs MWAA

**Attendees:** @Nabegh Abed **,** @Vaz, Kris **,** @Anirudh.Kilambi **,** @Hari Prasad Theniah **,** @Akhila Sree Maddukuri **,** @Dat Tran **,** @Sreeda.Deepak

**Meeting Notes**

1. 1  Nabegh - Demo Design diagram for monitoring API next Wed, Dat to setup meeting
2. 2  Masoud - Research Data Linage Tool & Data Catalog tool
3. 3  - Possible tools (OpenMetaData, OpenLineage, Atlan)
4. 4  New AWS account will levarage MWAA for speed of execution
5. 5  Long term we want to move towards a self hosted solution on EKS
6. 6  - Allow us access to the Airflow Scheduler DB
7. 7  - Access to the EBS storage volume
8. 8  - Access to the Metric API to track DAG success/Failure Ration
9. 9  Dat to setup call with AWS Architect to review Cognito Security setup and MWAA cost optimization

Meeting Date: Dec 8, 2022

**Technologies Discussed:** Standard Development Process

**Attendees:** @Nabegh Abed **,** @Vaz, Kris **,** @Anirudh.Kilambi **,** @Hari Prasad Theniah **,** @Akhila Sree Maddukuri **,** @Dat Tran **,** @Emily.Ripley , @Sreeda.Deepak

**Meeting Notes**

1 2 3 4 5 6 7 8 9

10 11 12 13 14

1) Create a DDL Report for onboarding new dataset.

A. Developer will have to create a config file (1 per table)

- File format for config file tbd (JSON, YAML, CSV)

- Config file will contain table schemas as well as data type for each columns

B. CI/CD Pipeline will run upon commit to create or alter the relevant tables in Snowflake

2) Developer will have to build the Spark Script & DAG (DAG will utilized given Custom Operator for EMR)

3) Developer will add entry into the Config repo as needed

To Do:

1) Setup the DDL repo structure and CI/CD Pipeline

2) Implement an automatic prefixing of data into the Namespace layer of Iceberg so that we can identify the tea

To Test:

1) Test if Two Iceberg Catalog can write to the sane glue catalog (not needed if we're going with multiple name

m

s

410

Meeting Date:

**Technologies Discussed:** DEV/PROD Environment

**Attendees:** @Nabegh Abed , @Vaz, Kris , @Adith Rajiv , @Anirudh.Kilambi , @Hari Prasad Theniah , @Akhila Sree Maddukuri , @Dat Tran

**Meeting Notes**

Separate dev/prod env (separate AWS Environment)

Script to sync data from prod to dev (airflow jobs or Iceberg Jobs) - Sync the latest dag in dev to prod  
- table schema in sync  
- Script to run up EMR clustered (CDK script)

Separate repo Create/Replace tables:  
Background jobs will refresh tables in both prod environment  
When someone wants to work on dev; they have to sync the prod branch to dev and run that to create a pipeline in dev manually.

Meeting Date: Nov 24, 2022  
**Technologies Discussed:** DynamoDB, Lambda, DBT, AWS

**Attendees:** @Dat Tran , @Vaz, Kris , @Minnie , @Nabegh Abed , @Anirudh.Kilambi , @Hari Prasad Theniah , @Akhila Sree Maddukuri , @Seye Ologunja , @Srinivasan Rajivelu

**Meeting Notes**

@HariPrasadTheniah tocompletedocumentationforsnoflake\_db\_maintenance. https://github.com/compassdigital/dataops\_de\_snowflake\_maintenance - Connect your Github account

Hard delete has to be done manually  
@DatTran discussedtheusageofspotinstances,cost-savingandeffectiveoption  
@NabeghAbed canincreaseon-demandECRs  
@Hari Prasad Theniah , @Dat Tran discussed about the Airflow costs and what can be done to bring them down.

Improvise AWS account

Meeting Date: Nov 17, 2022

**Technologies Discussed:** DynamoDB Table, Great Expectations

**Attendees:** @Anirudh.Kilambi , @Vaz, Kris , @Hari Prasad Theniah , @Nabegh Abed , @Akhila Sree Maddukuri , @Adith Rajiv , @Seye Ologunja , @Dat Tran

**Meeting Notes**

@Hari Prasad Theniah : Database configuration  
Created a new repo in GIT https://github.com/compassdigital/dataops\_de\_snowflake\_maintenance - Connect your Github account

1 - Copy data over Prod to dev (optional)

1. 2  - Find out what event or update BI needs for the entire dataset
2. 3  - Test out Iceberg Table from Prod to Dev
3. 4  - Set up bronze S3 layer to house all the source level data for all datasource

Dec 1, 2022

411

Every database will have its own JSON.  
updates DynamoDB table  
Python Is used.  
JSON files will get synced with the DynamoDB table The config will be set to false

Access can be provided as required  
Captured by Lambda and uploaded to the Snowflake User Role Mapping, Password setup

@Nabegh Abed :  
A library that is used to run expectations on the data. Certain values should fall within a certain range. Adjusts the data. You can set rules in the form of a code and it will generate a report for you  
Getting started with the Great expectations library and running that in Spark  
GEL can have multiple engines.  
Tutorial of their website and comes with a CLI  
Comes with cloned data from Uber.  
Run the notebook, you can change the configs.  
The next step is to create expectations.  
Run the boilerplate  
The result/output is displayed.  
The next step is to run expectations on  
A checkpoint is a group of expectations.  
Details are displayed when a file fails  
Can we have a portal that will show us the status and results of the rules?  
What are the next steps? A sample job can be taken and run some basic rules on it

@Anirudh.Kilambi : Bump up R5 nodes and increase the number of instances. 24 is max.

Meeting Date: Nov 10, 2022

**Technologies Discussed:** MongoDB, AWS, Debezium, GitHub, Amazon Open Search

**Attendees:** @Nabegh Abed , @Vaz, Kris , @Hari Prasad Theniah , @Anirudh.Kilambi , @Akhila Sree Maddukuri , @Dat Tran , @Srinivasan Rajivelu , @Minnie , @Seye Ologunja

**Meeting Notes**

@Vaz, Kris

Spoke about the streaming architecture

Most likely will funnel data into kinesis firehose which will go to an S3 bucket where a batch job will process the data at some interval. For real-time access, we have the option of connecting kinesis to opensearch for real-time access. Or maybe opensearch would be

412

loaded from the S3 bucket that kinesis sends data to.  
For CDC events, in Spark, we should be able to apply operations for CDC via merge statements.

We must make sure to sort the data by date to remove the possibility of incorrect order of event processing

@Dat Tran

Explore CDC options to see if this can be applied to databases (in particular Volante’s new MongoDB deployment). Review Streaming Architecture as proposed by Nabegh, Kris & Akhila

@Nabegh Abed , please add the proposed architecture diagram.  
@Hari Prasad Theniah , please add the details that you discussed in the meeting.

Meeting Date: Nov 3, 2022  
**Technologies Discussed:  
Attendees:** @Vaz, Kris , @Dat Tran , @Anirudh.Kilambi , @Akhila Sree Maddukuri , @Minnie , @Srinivasan Rajivelu **Meeting Notes**

@Anirudh.Kilambi

Managed scaling cluster is up and running. The workflow for spark submitting is exactly the same as it is for cdl-dataops- datalake-main

Job flow ID exists in emr\_constants as CDL\_DATAOPS\_MANAGED\_SCALING\_JOB\_FLOW\_ID Able to allocate up to 28 executors with up to 24 GB of executor-memory

Meeting Date: Oct 27, 2022  
**Technologies Discussed:** Docker, EMR Clusters, AWS, Snowflake  
**Attendees:** @Minnie , @Nabegh Abed , @Akhila Sree Maddukuri , @Adith Rajiv , @Vaz, Kris **Meeting Notes**

@Nabegh Abed

Reviewed the current setup of the managed EMR cluster.

Right now there is no Notebook functionality available. Docker image locally to start a local spark session that has a notebook or a script can be made to enable a cluster within the notebook.

Discussed whether dynamicAllocation should be enabled or not. The decision is to keep it **On**. @Vaz, Kris

Reminder to start thinking about merging jobs that have been running fine in AF 2.2 dev over to prod → see FAQs for information on the merge process

Look into the snowflake dimension sync option from the iceberg wherein we capture all active and “newly” inactivated records into a data frame. We then delete all active records from snowflake and insert this data frame into the table to accurately capture pre- existing active, new activities, and newly inactivated records.

@Anirudh.Kilambi

start looking at using a spark\_scripts subdirectory within the contained\_scripts in `cdl-dataops-airflow-2`. How should we handle zipping up the relevant files?

413

Post-mortems

414

MWAA 2.0 - Scoping

Some of the issues’ we’re trying to address are the following.

1. DBT, has interdependencies in the BI space with multiple chains of logic. We want to be able to test with these conditions, the requirement is that the source and downstream tables need to be able to be tested upon any changes.
   1. DBT – is interconnected data mesh, relying on multiple data sources. We need to properly mimic a production environment, so that testing can be ran on similar data.
   2. Models built ontop of models, Possible down stream implications.
   3. Needs end-to-end testing on staging instance before approval for prod promotion (run model and descendants, test model and descendants)
   4. All DBT environments can source from Prod databases (no need for data source replication), Possible to leverage the zero-copy cloning feature on snowflake.
   5. DBT can segregate schemas for environments ( downstream tables can be schema; dev, staging, dtb prod)
   6. Makes sense to leave source data from Iceberg (no row level permissioning)
   7. Source testing → not for dbt, should be assess upon data ingestion
2. Data Engineering – Airflow Stability. We want to ensure that prod environments aren’t affected by untested staging dags. a. Goal here is to ensure environment that can mimic Prod behaviour without using prod as an output sink.
3. Data Science / ML use cases
   1. Data transformations from fact tables should be implemented into DBT, so that DS scripts can load from a materialized table, instead

of replicating sql queries between multiple models ( can use generated dbt metrics as features/inputs)

* 1. Rule of thumb all outputs to be added to iceberg (source of truth), stored as dataframes, can automate with a custom Operator to load into Iceberg
  2. treat output data as new source, ex. Cleansed tiers, to be used in other analysis

1. Proper Promotion. How to confidently promote a staging dag and have the same expected result in Prod.
   1. Can look to add automated testing logic, and defining code-review process.
   2. Quasi-shared dev/staging branch to test shared integrations, cross dependancies (ex. trigger based on completed job, need to test if

particular trigger activates)

* 1. Current workflow, decisions left to individual teams i. Branch Prod → Feature Branch,

ii. Merge Feature Branch with DEV, iii. test integrations  
iv. merge Feature Branch → Prod

v. To be iterated upon.

MWAA Limitations  
25 Concurrent workers

Will host Airflow on EC2  
Dev instances for BI/DS/DE using mwaa, reasoning is less management Possible to host dev airflow also on EC2  
Astronomer.io ? For managed instances (nah too much work)  
Current (deployed) EC2 Airflow Architecture has Stack of EC2+RDS

415

1. If a Staging Database is necessary ( probably not with current proposal ) a. Idea is to use prod data sources, and write to dbt\_env\_schema

2. The proper workflow of promoting to production

3. Define the code-review requirements  
a. We’ve got a good starting point here.

i. Data Engineering PR reviewer – will ensure proper use of Operators + Proper compute methodology ii. Intra-team code review – Will ensure logic within dag is upto requirement standard.

Action Points  
Test integration points between dbt + iceberg.

dbt - for table/view/schema materialization  
dbt - Shared compute resources on redshift possible overload if three environments pull from this source mandate version locking  
Get dbt into Airflow EC2 requirements.txt  
dbt-core==1.1.0  
dbt-redshift==1.1.0  
v2, get dbt to run with kube operator

416

Volante Data Load Issue

**Description**

We started receiving reports of data not being fresh in the Quicksight extraction tool linked to datamart.pos\_ca. Upon further investigation, the root cause was identify as a data freshness issue on the source volante tables.

**Root cause**

Source volante table jobs were not refreshing due to an error in the VPN service between our CDL-BI accounts and Volante The VPN instance (https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#InstanceDetails:instanceId=i-

0c774f60b45b1d308) restarted - so the VPN service went down **Fixes**

Restart the VPN service

**Miscellaneous**

**Action items**

417

Missing Shopping Carts because of Deactivated Brand IDs

**Description**

A group of transactions did not have matching transaction details in the data warehouse. No matching shopping carts for the order IDs.

**Root cause  
P2 Dimension LOCATION\_BRAND\_ID** is a Store ID (multiple stores). For the system to track a certain store, a store ID is used to

identify the exact store for the transaction.

11J3gKPg8BCNZNQ1zlGrIPLQPZpZa9sm0yr01RzQHwYoq3d5PJFjP92YmoB6HO3rLOYljdHQRWN (Location Brand ID/StoreID) got deactivated on 1st of October, which made the P2 jobs not to extract the fact details for the BRAND\_ID above.  
However, order\_id=1Bl9YpOM1Lt3XEqP2KD5IpeNyG1Gpkuo5B5BYYEaiw6800BG9asjB1zGrLYMH48EQd0 for the same BRAND\_ID was already extracted because the transaction happened on 30th September.

**Fix**

Fixed the P2 Fact extract scripts to include deactivated BRAND\_IDs during the extraction process, for the Order IDs that are extracted before the deactivation date.  
The P2 transaction data refresh process is fixed to include deactivated store IDs when the order IDs are already refreshed but the shopping cart data is not extracted.

**Conclusion**

Shopping carts need to be extracted whether the associated LocationBrandID is active or not.

418

POC

419

Webhook Authorization

Use case:

Architecture:

Pro:

1. g 2. h 3. I

Resources:

420

Optimus (Star Schema)

---- DW Schema ------ Site -> Location → Units [Orders Star Schema] DIM\_LOCATION

Key (Hash) Unit\_ID ERP\_SYSTEM\_ID Accounts

Region  
Address  
Timezone  
....  
HasKiosk (Boolean) HasMobile (Boolean) HasPOS (Boolean) HasNutrislice (Boolean)

DIM\_LOCATION\_TERMINALS\_AND\_BRANDS Key (Hash)  
DIM\_LOCATION.KEY BrandName/TerminalName

Etc....

**DIM\_Location (SCD)**

(Master list of all compass locations for all channels)  
- Key (hash)  
- Brands - Filled out Mobile, Eatclub if not available fill in with “Not Applicable” - Terminals #- Filled out where available, and “Not Applicable” everywhere else - Terminal Name  
- Site\_Mobile  
- Location\_Mobile (Nulls can be code in as N/A)  
- Unit #- Org\_hierachy  
- Unit Name - Org\_Hierachy  
- District - Org\_Hierarhy  
- Sector - Org\_Hierachy  
- Region - Org\_Hierachy  
- Division - Org\_Hierachy  
- Accounts - Client\_Unit\_Dim  
- Timezone (Amercia/New\_York, America/Chicago)  
- Valid\_Start\_Date (Phase 2)  
- Valid\_End\_Date (Phase 2)  
- ERP\_SYSTEM\_ID

421

- Country - Source

Update 03/23/2022 - Remove Location Column from Dw.dim\_location  
change dim\_location.site to dim\_location.mobile\_sitename  
Rename dim\_location\_terminals.location to be dim\_location\_terminals.location\_key  
Change the join key from Dw.dim\_location & dw\_dim\_location\_terminals to be dim\_location.key = dim\_location\_terminals.location\_key

**DIM\_Items (SCD)**

(Master List of all compass items + cleansed tier for all channels) - Key (Hash [MenuItemID, OptionItemID, Brandid])  
- MenuItemID  
- OptionItemID

- MenuItemName - OptionName  
- Valid\_Start\_date - Vaid\_End\_Date - Source

**DIM\_Customer**

(Master List of all customer related data (register users, hash credit card identifier etc...) across all channels) - DIM\_Customer.Key  
- LT Value $  
- Segmentation

- Registration\_Date  
- First\_Transaction\_Date - Last\_Transaction\_Date

**DIM\_Fiscal\_Calendar\_datetime** (goes down to minute) (Hari has script, might need tweak) - CalendarDate\_DateTime (To the minute (YYYY/MM/DD HH:MM)  
- Fiscal Calendar (US/Canada) Fields  
- Same fields as static\_dimension.fiscal\_calendar

**DIM\_FISCAL\_CALENDAR** (Replica of static\_dimensions.fiscal\_calender)

**Fact\_Employee\_Data**

(list of all front-line employees across compass and the units they are current associated with. Being able to track an employee movement as they transfer across units)  
- Employee ID  
- Country

- Employee status - Job Title  
- DIM\_location.key - Date Stamp

- GL\_Account

**Fact\_Order\_items**

(Item level transactional data for all channels)  
Hash Transaction Key, TransactionID, DIM\_ITEM.key, DIM\_Customer.key, dim\_location.key, dim\_date\_utc, OrdeDate, TransactionType, Tax, Promo  
Mobile, Kiosk, POS, Frictionless

**Fact\_Orders**

(Order level transactional data for all channels) [Add in sum of item level stats as columns here to avoid joining to the item table]

422

Hash Transaction Key, TransactionID,

**Fact\_PaymentMethod**

(Payment method data for all transactions across all channels)  
Hash\_Transaction\_Key, Tender Name, Amount, Last4Digit, Expiry Date, Customer\_Key Visa, 4.5, 1234, 1231, 123,  
Cash, 0.5, NULL, NULL, 123,

Fact\_Transaction\_Taxes  
(Transaction Tax data for all transactions across all channels) Hash\_Transaction\_Key

**Fact\_GL\_Accounting\_Summary**

dim\_location.key, GL\_PARENT, GL\_PARENT\_DESCRIPTION, FISCAL\_PERIOD, Amount, Source

Sample:  
28f7sss, F-155, Gross Profit, 202001, 40000, SAP 28f7sss, F-99, 202001, Total Sales, 15000, SAP 287sss, F-14, 202001, Catering Sales, 2000, SAP

----- DW Staging layer -----

**Transaction Staging:**

Eatclub\_transactions Nutrislice Mobile\_Transactions

Source  
-P2  
-Tacit  
-MM Hayes  
-Clearview (if we want to sepearte Tim Apps) -MyMeal (MyDinning mobile apps)

Kiosk\_Transactions (This exists) Source

-POS\_ORDER\_HEADER (Done) -POS\_ORDER\_DETAIL (Done) -MM Hayes  
-Volate\_CA

-Clearview  
-P2 (where appname = 'FoodWorkKiosk')

POS\_Transactions (This Exists)

Source -POS\_ORDER\_HEADER (Done) -POS\_ORDER\_DETAIL (Done) -MM\_HAYES  
-VOLANTE\_CA  
-CLEARVIEW

**Location Staging: (We flag out the transaction type (Mobile, Kiosk, POS etc...)**

P2\_Location (Angela) Tacit\_Location (Dat) Eatclub\_Location (Akhila)

423

US\_DW\_POS\_LOCATION (Kris S.) MM\_HAYES\_Location (Kris S.) Volante\_Canada\_Location (Brett) Clearview\_Location Silverware\_Location

Data source Contact:  
Eatclub - Leon Barovic <leon.barovic@myeatclub.com> MM Hayes - Dylan J. Keyer <djkeyer@MMHAYES.com>

**Draft Data Dictionary**

**data\_dictionary.xlsx**

07 May 2022, 01:24 AM

424

Overview / Design

This page is intended to explain what the star schema is and as a guide for writing queries against it.

Index

A Single Source of Truth - An explanation of the genesis of this system and the problems it solves Directional Value vs Penny Accuracy - What to expect from the star schema  
Design - A high level description of the overall design  
Fact Tables - A detailed explanation of what a fact table is and their design principles

Dimension Tables - A detailed explanation of what a dimension table is and their design principles

A Single Source Of Truth

What is this star schema, why do we need it and what problems does it solve? To answer these questions we need to understand what the previous state looked like. This system is, essentially, an evolution of our data architecture. Previously our source data was imported into a cloud database, then transformed into a more query friendly structure on a vendor by vendor basis in a schema called datamart. For example one table for MM Hayes data, another for P2 data etc. This was further refined by creating new tables in the datastore schema which were broken down by order origin (mobile, kiosk etc).

This created several issues.

1. A large number of moving parts that were difficult to monitor and maintain, with variance in how key business concepts were implemented (causing lots of data consumer questions around how a particular field was calculated)
2. The output of these processes were not compared to each other, meaning that when logic for classifying orders as mobile / web / kiosk / pos was updated, it had to be updated in many places and mistakes could cause a single order to appear in more than one datastore table.
3. Occasionally, the filters used could cause subsets of order data to be ignored
4. Difficulty in working with more than one type of order data at a time, due to fields not lining up neatly between different datastore tables and performance issues in merging the datasets in a query (eg: Redshift needing to transmit large volumes of data around the network). Sometimes people have worked around this by querying the source tables directly, which creates performance issues because it isn’t easy to optimize them for all the kinds of joins that are required

The star schema aims to solve these problems by:

1. Creating a set of transformations that are more easily monitored with tests on the output to alert us to strange results and standardizing key fiscal metrics. No more wondering which sources include discounts or taxes in net sales etc.
2. Classifying order origins (mobile / kiosk etc) during the initial transformation from source, to prevent double counting
3. Simplifying the initial transformation logic as much as possible to ensure nothing is excluded from the output
4. Organizing the resulting data in a way that takes full advantage of our cloud columnar database architecture, making it possible to query much larger volumes of data in a shorter time. This is partly through structuring the output tables so that minimal network transmission is required, and by taking the output of expensive source queries and materializing the results so that this work doesn’t need to be repeated on a regular basis. The hope is that as we expand the ability of this star schema to answer more types of business questions, the fewer queries against raw source data will be required. As the variety of query patterns against source diminishes, it should be possible to tune those tables to further improve performance.

The result is a data source that should be easier to use and trust for directional analysis.

425

Directional Value vs Penny Accuracy

Note: Errors that exist in the source data will occasionally be carried through into the star schema, meaning that although we strive for it we cannot guarantee to the penny accuracy. We have made significant efforts to detect broadly occurring data issues and correct them in a way that makes sense when possible. For example; sometimes the order header information provided was missing values for key fields like subtotal or net sales. so we tried as much as possible to calculate the values we expected to see using other values that **were** provided in the order header, or by generating the missing values from the line item details where performance allowed. However, attempting to correct source data during transformation will often cause errors in rows whose figures previously agreed with one another. The true and proper fix for these issues is to correct them at source.

We employ tests during transformation, such that if the number of rows where the figures provided make no sense rises above a given threshold we will be alerted and begin an investigation into the cause. Documentation around the specifics of these tests is generated when the models run, and will be made accessible once we’ve worked out how to host those files.

Design

The star schema is heavily inspired by Kimball architecture. I say inspired because that architecture was invented before cloud and columnar databases, and we can gain performance by breaking some of Kimball’s design rules. The key thing to understand is that everything is broken down into two types of tables: facts and dimensions. Our first release includes two fact tables for order data, and dimensions for dates, times and two levels of location data. The diagram below gives a high level overview of how these tables relate to one another, at time of initial release. This diagram will expand over time as we add to the data model.

What is a Fact Table?

A fact table is one that is designed the capture the measurements resulting from a business process event. For example, for our initial work we selected “a customer places an order” as our business process event. Every fact table has a **grain**, which defines what level of granularity exists within the data. In order for the data to be sensible, all data added to the table must be aggregated to the selected grain. The finest grain that exists for the “order placed” event is the individual line items that make up the order, and you will find data at this grain in the Order\_Details table. We also frequently require data that is summarized to the order level, so we also roll the raw data up to this level and store it in the Orders table.

426

What is a Dimension Table?

A dimension table contains the descriptive properties that will be presented in a BI front-end for filtering and grouping fact data. These provide context for the events, and answer The Five W’s of an event - who, what, where, when, why (and how). The design and content of dimension tables heavily affect the experience of a user’s BI experience.

This approach to storing descriptive data has a variety of benefits.

1. One is that it helps us to standardize the values in use and simplify the dropdowns users see when they try to filter their dashboards. This also reduces the volume of data that needs to be recorded in the database, since instead of having 5+ large fields repeated for every order or order item we can have one row that is linked to the event with a foreign key.
2. Another is that if a descriptive value needs to be modified because our business process has been updated, or if a new descriptive value is added; then the process of adding or changing that value becomes very simple. The larger the volume of fact data is the more important this quality becomes.

3. Another possibility that opens up with this approach is applying a historical lens to descriptive data - eg: what was the value of this descriptive tag at the time the event was recorded. In short, we can be both more and consistently descriptive about our business process events, and evolve these descriptions without having to disturb the core data of the event itself.

*Architectural Note: You may still see descriptive values stored directly in our fact tables. The above represents an ideal to strive towards, and the “rules” around this concept were created in a time where everything was on-prem making disk space a primary concern, and when distributed computation wasn’t a consideration. If you encounter descriptive values in a fact table it will be because either - we have yet to design a sensible dimension to contain that data or we are consciously leaving that value in place to allow it to be accessed without requiring a potentially expensive join to another table. This is the foundational issue in a discussion around using Kimball / Star Schema design vs what is know as One Big Table or Wide Table design. The latter squashes all of the descriptive values from dimensions into the fact tables and prevents you from needing to do any joins at all in your queries. Which approach is better is a discussion that is far from settled, but my feeling is that there is significant data management value to storing descriptors in dimensions - as changing them in OBT design requires performing a full reload of all fact data. As a compromise, we will likely selectively violate the “descriptors in dimensions” rule.*

427

Data Dictionary Index

Dimensions

Dim\_Customers: Data Dictionary | Dim\_Customers  
Dim\_Locations: Data Dictionary | Dim\_Locations  
Dim\_Location\_Terminal\_Brands: Data Dictionary | Dim\_Location\_Terminal\_Brands Dim\_Fiscal\_Date: Data Dictionary | Dim\_Fiscal\_Date  
Dim\_Fiscal\_Datetime: Data Dictionary | Dim\_Fiscal\_Datetime

Facts

Orders: Data Dictionary | Orders  
Order\_Details: Data Dictionary | Order\_Details Order\_Discounts: Data Dictionary | Order\_Discounts

**Dimensions**

Dim\_Customers

This table contains only the most basic information about customers that can be pulled directly from source without complex analysis. Not all records in this table represent a customer we could identify. Each data source has one or more “unknown” customer keys that are assigned when identifying information was unavailable. You can identify those records via the is\_unknown\_user field.

**Field name Type** key (primary key) GUID

has\_t2e\_source bool has\_eatclub\_source bool

has\_p1\_source bool

has\_p2\_source bool

has\_mmhayes\_source bool

has\_usdw\_source bool

**Sample Values**

**Comment**

Use this field to link to other tables that have the customer\_fk field

|  |  |  |  |
| --- | --- | --- | --- |
| is\_unknown\_user | bool | true / false | Use this field to filter out unknown customer records. There are a lot of them, which could cause a lot of skew when calculating metrics around customer behavior. Please note that at launch, all orders coming from the US Data Warehouse are from unknown customers. |
| date\_created | date |  | This is the date when we first encountered the customer. It is either the date when the customer created their account or, when that concept does not apply, it will be the date of the first order associated with the customer. |

true / false true / false

true / false

true / false

true / false

true / false

Marked

Marked Eatclub

Marked P1 app

Marked P2 app

Marked via MM

true if we know for certain that the customer has used T2E true if we know for certain that the customer has used

true if we know for certain that the customer has used a

true if we know for certain that the customer has used a

true if we know for certain that the customer has ordered Hayes

Marked  
in US DW data

true if we know for certain that the customer has appeared

428

Dim\_Locations

This table includes one row for every known combination of unit id and erp system id. At release you need to join to this table in order to add the unit id or any other location information to your resultset. Other examples of data you can find here are the country or state the unit is located in, whether it is known to have kiosk / mobile / pos / web orders or any org hierarchy data (eg: sector, division etc).

**Field name Type** key (primary key) GUID mobile\_site varchar

**Sample Values**

**Comment**

Use this field to link to other tables that have the location\_fk field

This field was inherited from some of the original source transforms but is not relevant to most sources and will likely be removed.

|  |  |  |  |
| --- | --- | --- | --- |
| unit\_id | varchar | 24934 42342 50702 | The unit number / id of the location, corresponding to erp\_entity\_id in most sources. |
| unit\_name | varchar | JPMC Lake Vista UH Main SC Chick-Fil-A Carlow University Franks | The name of the unit, typically taken from the org hierarchy source table. |
| district | varchar | Fucsko, M DMF Pereira, C DMF Mottola, B | The district name from the org hierarchy source table |
| sector | varchar | Eurest Sector Chartwells Sector | The sector name from the org hierarchy source table |

region varchar

TX  
Whitesell, M Region

The region name from the org hierarchy source table

|  |  |  |  |
| --- | --- | --- | --- |
| division | varchar | South/Oil and Gas Division Higher Education Division | The division name from the org hierarchy source table |
| accounts | varchar | JPMorgan Chase Bank NA University of Houston System Carlow University | The account name found in the source table cg\_dw\_cdl\_us\_source\_fin\_client\_contract\_dim |
| timezone | varchar | America/Chicago America/New\_York Eastern Standard Time | The time zone the unit is located in, if that information was available. Currently if this was not available, you will see a question mark. |

country varchar

United States Canada

|  |  |  |  |
| --- | --- | --- | --- |
| longitude | numeric | -73.97987365722656 | The longitude of the unit. This value is either retrieved directly from the source data, when available, or extrapolated based on the address on file. Combined with latitude, we can use this to plot our units on a map. |
| latitude | numeric | 40.76171112060547 | The latitude of the unit. This value is either retrieved directly from the source data, when available, or extrapolated based on the address on file. Combined with longitude, we can use this to plot our units on a map. |
| data\_source | varchar | US Data Warehouse T2E Data Source P2 Data Source | This field is a signifier of which source tables a row originated from. Very helpful for troubleshooting. |

erp\_system\_id int

has\_kiosk int

has\_mobile int

has\_pos int

1001 2010 0  
1

0 1 0 1

The id of the erp system the data came from. The combination of this with unit id should be unique, unless a unit id was recycled.

Indicates if kiosk orders have been detected at this unit, 0 for false and 1 for true

Indicates if mobile orders have been detected at this unit, 0 for false and 1 for true

Indicates if pos orders have been detected at this unit, 0 for false and 1 for true

429

has\_web

is\_deactivated

etl\_extract\_timestamp

int 0 1

int 0 1

timestamp 2022-04-07 07:43:57.909

Indicates if web orders have been detected at this unit, 0 for false and 1 for true

Indicates whether or not a unit is known to have been deactivated. Default value for all rows is 0 for false. If deactivation logic was available, then 1 for true.

This field indicates when the source data for this row was refreshed.

Dim\_Location\_Terminal\_Brands

This table represents a finer grain of location data, with one row for every terminal or brand found at a unit. Join to this table to find out specifically where the order was processed. You can also join this dimension to Dim\_Locations to build a resultset that shows all of our location data, but if you do this do not join to a fact table such as orders because each order will be displayed once for every terminal / brand that exists at the unit. This functionality is provided merely for exploratory purposes.

**Field name**

key

**Type**

GUID

**Sample Values**

**Comment**

Use this field to link to other tables that have the location\_terminals\_fk field

|  |  |  |  |
| --- | --- | --- | --- |
| location\_fk | GUID |  | Use this field to join to Dim\_Location on key = location\_fk  **CAUTION** - Provided only to allow easy querying of only location data. **Do not join on this field if a fact table is involved**. Doing so is highly likely to inflate your results. |
| brands | varchar | Cafeteria Dillons Coffee Allianz #19804 Multiple Brands | Identifies the brand linked to the order (mobile orders) or the profit center (pos orders). In other words, it tells us which restaurant an order came from.  Currently, if an order was split across multiple brands, this field will read “Multiple Brands” |

terminal\_id terminal\_name mobile\_site

location unit\_id

unit\_name

region

varchar varchar varchar

varchar varchar

varchar

varchar

/ GUID

This field was inherited from some of the original source transforms but is not relevant to most sources and will likely be removed.

The unit number / id of the location, corresponding to erp\_entity\_id in most sources.

The name of the unit, typically taken from the org hierarchy source table.

The region name from the org hierarchy source table

10694 25325

M9449 Regional Medi SHU Library Cafe

TX  
Whitesell, M Region

|  |  |  |  |
| --- | --- | --- | --- |
| district | varchar | Fucsko, M DMF Pereira, C DMF Mottola, B | The district name from the org hierarchy source table |
| sector | varchar | Eurest Sector Chartwells Sector | The sector name from the org hierarchy source table |

division

varchar

South/Oil and Gas Division Higher Education Division

The division name from the org hierarchy source table

accounts

varchar

JPMorgan Chase Bank NA University of Houston System Carlow University

The account name found in the source table cg\_dw\_cdl\_us\_source\_fin\_client\_contract\_dim

timezone

varchar

America/Chicago America/New\_York Eastern Standard Time

The time zone the unit is located in, if that information was available. Currently if this was not available, you will see a question mark.

430

erp\_system\_id

country

has\_kiosk

has\_mobile

has\_pos

has\_web

etl\_extract\_timestamp

Dim\_Fiscal\_Date

int 1001 2010

varchar United States Canada

int 0 1

int 0 1

int 0 1 int 0 1

timestamp 2022-04-07 07:43:57.909

The id of the erp system the data came from. The combination of this with unit id should be unique, unless a unit id was recycled.

Indicates if kiosk orders have been detected at this unit, 0 for false and 1 for true

Indicates if mobile orders have been detected at this unit, 0 for false and 1 for true

Indicates if pos orders have been detected at this unit, 0 for false and 1 for true

Indicates if web orders have been detected at this unit, 0 for false and 1 for true

This field indicates when the source data for this row was refreshed.

|  |  |  |  |
| --- | --- | --- | --- |
| data\_source | varchar | US Data Warehouse T2E Data Source P2 Data Source | This field is a signifier of which source tables a row originated from. Very helpful for troubleshooting. |

|  |  |  |  |
| --- | --- | --- | --- |
| is\_deactivated | int | 0 1 | Indicates whether or not a unit is known to have been deactivated. Default value for all rows is 0 for false. If deactivation logic was available, then 1 for true. |

This table exists as a copy of static\_dimensions.dim\_fiscal\_date, and was created to protect against failures if the owner of static\_dimensions ever changes the table unexpectedly, and allows us to customize the dist and sort for performance. As of writing, this table is missing from dbt\_dev and you can replace its functionality by joining to static\_dimensions.dim\_fiscal\_date on date\_name = \*\_date\_fk

Sample values in this table all taken from a single record representing data for 2022-08-25 eg:

1 select o.key, d.fiscal\_quarter\_name  
2 from orders o  
3 inner join static\_dimensions.dim\_fiscal\_date d on d.date\_name = o.order\_date\_fk

**Field name**

date\_name

date

2022-08-25

This is the primary key and you will use it to join on fields named things like:  
order\_date\_fk  
pickup\_date\_fk

date\_no  
dayname dayname\_short cal\_fis\_month cal\_fis\_month\_year cal\_fis\_month\_year\_sort

dayofweek dayofweek\_monday fiscal\_dayofweek fiscal\_period\_id

**Type Sample Values**

bigint 20220825 varchar Thursday varchar Thu varchar Jul varchar May 2031 int 203105

int 5  
int 5  
int 6  
int 202211

**Comment**

Unsure what purpose this field was created for.

used to enforce correct sort order on field cal\_fis\_month\_year in Power BI

1 = Sunday 1 = Monday

431

fiscal\_period\_name prior\_fiscal\_period\_id prev\_period\_name fiscalperiodweek fiscal\_quarter\_id fiscal\_quarter\_name prev\_quarter\_name fiscal\_week\_id fiscal\_week\_date fiscal\_week\_name prev\_week\_name fiscal\_year\_id fiscal\_year\_name prev\_year\_name fiscal\_period\_start\_date fiscal\_period\_end\_date fiscal\_start\_date fiscal\_end\_date fiscal\_start\_week fiscal\_end\_week fiscal\_quarter\_start fiscal\_quarter\_end

mth  
wk  
q prev\_week\_start prev\_week\_end prev\_period\_start prev\_period\_end prev\_quarter\_start prev\_quarter\_end prev\_year\_start prev\_year\_end

Dim\_Fiscal\_Datetime

varchar P11-2022  
int 202210 varchar P11-2021 varchar P11,W47-2022 int 20224

int Q4-2022 varchar Q4-2021 int 202247 date 2031-06-30 varchar W47-2022 varchar W47-2021 int 2022

int 2022  
int 2021  
date 2022-07-30 date 2022-08-26 date 2021-10-02 date 2022-09-30 date 2022-08-20 date 2022-08-26 date 2022-07-02 date 2022-09-30 varchar 11  
varchar 47  
varchar 4  
date 2021-08-14 date 2021-08-20 date 2021-07-24 date 2021-08-27 date 2021-06-26 date 2021-10-01 date 2020-09-26 date 2021-10-01

date representing the Monday of the week of the row’s date

This table exists largely as a copy of Dim\_Fiscal\_Date with the addition of an extra row for every hour and minute of the day, as well as a custom field that breaks each hour into one of four 15 minute intervals. If the type and sample values are not filled out in this table below, please refer to the above Dim\_Fiscal\_Date table. We should consider trimming down this table considerably since so much of the data is already found in the other table.

**Field name Type Sample Values Comment**

432

key

datetime

2015-09-26 00:18:00.000

This is the primary key and you will use it to join on fields named things like:

order\_fiscal\_date\_fk pickup\_fiscal\_datetime\_fk

Note that the seconds and milliseconds are truncated from the timestamp, as we aren’t interested in that level of granularity. All timestamps in the star schema are rounded down to the nearest minute.

date\_name date\_time

date\_no  
day\_name cal\_fis\_month day\_of\_week fiscal\_day\_of\_week fiscal\_period\_id fiscal\_period\_name prior\_fiscal\_period\_id prev\_period\_name fiscal\_period\_week fiscal\_quarter\_id fiscal\_quarter\_name prev\_quarter\_name fiscal\_week\_id fiscal\_week\_name prev\_week\_name fiscal\_year\_id fiscal\_year\_name prev\_year\_id prev\_year\_name fiscal\_period\_start\_date fiscal\_period\_end\_date fiscal\_start\_date fiscal\_end\_date fisal\_start\_week fiscal\_end\_week fiscal\_quarter\_start fiscal\_quarter\_end quarter

month week day hour minute

datetime

2015-09-26 00:19:00.000

Now that the key itself is a datetime, this field is a duplicate and will likely be removed.

|  |  |  |  |
| --- | --- | --- | --- |
| 15\_min\_interval | int | 0 15 30 45 | This field represents the minute value of a timestamp, rounded to the nearest 15 minutes. This allows us to easily slot orders into 15 minute timeslots, and do things such as highlight the busiest part of an hour. |

prev\_week\_start

433

Facts

Orders

This table shows one row for every order processed, and includes summary financial data. Net sales, total sales, taxes collected, number of items purchased etc. If you’re not interested in the specific details of the items in an order, this is the only fact table you need to use.

**Field name Type** key GUID grain\_lvl varchar

order\_id varchar

location\_fk GUID

location\_terminals\_fk GUID

customer\_fk GUID

**Sample Values**

000016dceedd2df0c0b7b5397b78ffe6 order

1192236957AVEC 6449595

2bdab6d14a301a6a7a21e8290132f307

d03d006b833ebe4af9ae41e1f184842d

d03d006b833ebe4af9ae41e1f184842d

**Comment**

Used to join on order\_details on key = order\_fk

Will always be “order”, left in to reflect the multi- grain architecture we’re migrating away from.

The order id from the source system. Will not be unique unless you’re only looking at a single location.

Links to dim\_locations on location\_fk = key

Use this field to join to Dim\_Location\_Terminals\_Brands on location\_fk = key

Use this field to join to Dim\_Customers on customer\_fk = key

|  |  |  |  |
| --- | --- | --- | --- |
| order\_origin | varchar | kiosk mobile pos web | During the initial transform from source we apply known logic for that source to classify an order with one of these values. POS is the default value. |

|  |  |  |  |
| --- | --- | --- | --- |
| order\_fiscal\_date\_fk | date | 2022-08-02 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Date, which you can join to in order to pull financial calendar fields such as fiscal week, quarter etc. |
| order\_fiscal\_datetime\_fk | datetime | 2020-10-19 10:07:00.000 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Datetime, which you can join to in order to pull date parts, 15 minute interval values etc. Currently it may be more performant to perform transformations on the datetime field itself rather than join to the dimension. We may eventually devolve the interesting fields from the dimension directly into fact table fields. |
| pickup\_fiscal\_date\_fk | date | 2022-08-02 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Date, which you can join to in order to pull financial calendar fields such as fiscal week, quarter etc. |
| pickup\_fiscal\_datetime\_fk | datetime | 2020-10-19 11:58:00.000 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Datetime, which you can join to in order to pull date parts, 15 minute interval values etc. Currently it may be more performant to perform transformations on the datetime field itself rather than join to the dimension. We may eventually devolve the interesting fields from the dimension directly into fact table fields. |
| app\_name | varchar | Tap2Eat eatify AVEC AGILYSYS Unknown | For mobile orders this represents the recorded name of the app that the order came from. If no app was involved, it will show the pos source that generated the order. If nothing appropriate can be found then Unknown will be displayed. |

434

order\_type

varchar

Delivery Sale Normal Sale Delivery Frictionless

This field is similar to transaction\_type, but contains a more detailed classification than transaction\_type. MM Hayes, P2 and T2E all have useful values in this field.

US DW seems to have some strange values in this that look like unit names with a few mixed in that make sense such as Mobile Order, Take Out etc. Needs refinement.

transaction\_type

varchar

Sale  
Refund  
Cancel  
Void  
Received on Account

Use this field to find out what sort of transaction the order represents.

At release only MM Hayes has useful data in this field, other sources are marked “Unknown” or “Not Applicable” - it is likely more work is needed to discover where this data lives in the source systems.

payment\_method

varchar

Cash  
Credit Card  
On Account Master CardEMV Visa

This field displays whatever payment method information was stored at source.

employee\_id

varchar

3714 96001350

This field presents an the id of the employee who processed an order. Very often unknown. The values that are not unknown are not transformed from the source system, other than to make them a string type.

employee\_name

varchar

Real names  
Site names (eg: Exxon) Terminal name

This field has questionable value and is a likely candidate for removal. Possible PII issue from real names (although, lacking much in the way of context), but vendors also often have a single user for a terminal that is always logged in.

subtotal

numeric

10.75

The gross sales of the order, before discounts, taxes or fees are accounted for. Generally speaking this equals the sum of item prices multiplied by their quantities.

service\_fee

delivery\_fee

total\_tax net\_sales

total\_sales

data\_source

Order\_Details

numeric

numeric

numeric numeric

numeric

varchar

1.99

1.99

1.99 9.99

10.99

US Data Warehouse T2E Data Source

The amount of service fees collected for the order, if applicable. Displays 0 if not applicable.

The amount of delivery fees collected for the order, if applicable. Displays 0 if not applicable.

Total tax collected on an order.

Subtotal - discounts + delivery fees + service fees In other words, the pre-tax amount that we charged to the customer.

Net sales + tax  
The amount that the customer paid

This field is a signifier of which source tables a row originated from. Very helpful for troubleshooting.

|  |  |  |  |
| --- | --- | --- | --- |
| discount | numeric | 1.99 | Dollar value discounts applied to the order. Note that this amount can differ from the discounts applied at item level - some vendors make a distinction between the two. |

|  |  |  |  |
| --- | --- | --- | --- |
| item\_quantity | numeric | 1 2 3 4 | The count of items sold in the order |

This table includes one row for every item and option in an order (side note: this is the intent, there are occasions where the same item name can be repeated in multiple rows of an order - work in progress). Generally speaking the sums of the financial figures in this table should be the same as the corresponding row in the orders table - with one caveat. Discounts and taxes are often only available at the

435

order header level, and this not recorded in this table. For truly accurate summary data, I would recommend relying on the data in the orders table.

**Field name**

key

order\_fk grain\_lvl

location\_fk

location\_terminals\_fk

lineitem\_id

parent\_lineitem\_id

**Type**

GUID

GUID varchar

GUID

GUID

varchar

varchar / GUID

**Sample Values**

000000dcf90028d58907140dbe6d7c1d

00000003106baf3269b97c0088fdfc9e item  
option

2bdab6d14a301a6a7a21e8290132f307

d03d006b833ebe4af9ae41e1f184842d

1062008 4559 1062008 4559

1.99

2

**Comment**

A unique id for the row. At this time it is not used for any kind of joins, and is not necessary to include within a select unless testing for row uniqueness.

This corresponds to the key field in the orders table

This field tells us if the row represents a menu item or a modifier to a menu item

Links to dim\_locations on location\_fk = key

Use this field to join to Dim\_Location\_Terminals\_Brands on location\_fk = key

The id of the item from the unit’s menu

Use this to link an option to its parent item. If the item is the parent item, then this value should match lineitem\_id

The base price of the item as charged to the customer.

The amount of the item being purchased. Note that this can be a decimal - this happens when the item is sold by weight, eg: at a salad bar.

|  |  |  |  |
| --- | --- | --- | --- |
| order\_id | varchar | 1192236957AVEC 6449595 | The order id from the source system. Will not be unique unless you’re only looking at a single location.  This is a duplication of the order\_id field in the orders table, and is mainly left in for the purposes of troubleshooting specific orders, in combination with data\_source (eg: going back towards the source data to compare) |

|  |  |  |  |
| --- | --- | --- | --- |
| lineitem\_name | varchar | Dasani Water (20oz) Salad Bar - ToGo Fresh Jalapenos | The name of the item from the menu |
| lineitem\_description | varchar | Meat Ball Sub White Hoagie 3oz Unknown Not Applicable | If the item’s menu description was available to us you can find it here.  If it was not available you will see Unknown or Not Applicable (note: this needs to be standardized / cleaned up) |

lineitem\_price

quantity

numeric

numeric

|  |  |  |  |
| --- | --- | --- | --- |
| customer\_id | varchar |  | Not really ready for use, in time it will become a GUID key once we have a sensible dimension defined. Sometimes you’ll see a GUID value in this field that was generated by the original datastore / datamart logic. For most rows this field will be empty. |
| order\_fiscal\_date\_fk | date | 2022-08-02 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Date, which you can join to in order to pull financial calendar fields such as fiscal week, quarter etc. |
| order\_fiscal\_datetime\_fk | datetime | 2020-10-19 10:07:00.000 | This field acts as both a date and a foreign key to Dim\_Fiscal\_Datetime, which you can join to in order to pull date parts, 15 minute interval values etc. Currently it may be more performant to perform transformations on the datetime field itself rather than join to the dimension. We may eventually devolve the interesting fields from the dimension directly into fact table fields. |

lineitem\_subtotal

numeric

5.99

The gross sales of the item, before discounts or taxes are accounted for. Generally speaking this equals the sum of item price multiplied by quantity.

436

lineitem\_discount

1.00

The discount given for the item. Note that the sum of discounts for all items in an order may not match the sum of discounts in the order header data - some vendors treat these as entirely separate.

lineitem\_tax

lineitem\_net\_sales

lineitem\_total\_sales

tlttti t

Order\_Discounts

numeric

numeric

numeric

dtti

1.15 The amount of tax charged on the item. May show 0 because no tax was applicable, or because no

data was collected.

9.99 Subtotal - discounts  
In other words, the pre-tax amount that we

charged to the customer. 10.99 Net sales + tax

20220407074357909 Thifildidit hth dtfthi

numeric

|  |  |  |  |
| --- | --- | --- | --- |
| lineitem\_tax\_recorded | boolean |  | Some vendors do not record taxes at the item level, so we created this field to understand if a 0 in lineitem\_tax was because the data was not recorded or because it was tax free. |

|  |  |  |  |
| --- | --- | --- | --- |
| data\_source | varchar | US Data Warehouse T2E Data Source P2 Data Source | This field is a signifier of which source tables a row originated from. Very helpful for troubleshooting. |

This table provides detailed information about the discounts, promos, loyalty rewards etc (also known as contra-revenue) that were applied to an order. There is one row per discount, which means that when joining orders to order\_discounts you will need to be careful not to double-count order header amounts. This table includes order subtotal discounts as well as line item discounts.

**Field name Type** key GUID

order\_fk GUID

discount\_type varchar

**Sample Values**

000000dcf90028d58907140dbe6d7c1d

00000003106baf3269b97c0088fdfc9e

Discount, Promo, Loyalty

**Comment**

A unique id for the row. At this time it is not used for any kind of joins, and is not necessary to include within a select unless testing for row uniqueness.

This corresponds to the key field in the orders table.

The specific type of discount we retrieved from the source data.

|  |  |  |  |
| --- | --- | --- | --- |
| discount\_method | varchar | Amount, Coupon, Percent | Generally understood as either Percent or Amount. Some vendor systems have specific terminology for non-percentage based discounts that has been preserved, so you may see many synonyms of Amount.  I left the original terminology in place under the assumption that the end users will be more familiar with that, but there is an argument to be made for flattening this down to only Percent / Amount. |
| discount\_value | numeric | 10, 20, 0.99 | The theoretical value of the discount. Understanding the meaning requires taking discount\_method into account. If the method is Percent then this value tells us what percent off the discount gives. Otherwise this field represents a max coupon / amount value. Eg: a $5 coupon Note the relation between discount\_value and order\_discount\_amount - it is possible (and not uncommon) for the former to be larger than the latter. Eg: a $5 coupon applied against a $3 item, resulting in a free item. |

code varchar

feedme20, boosttowin

The discount / promo code that was used to get the discount.

|  |  |  |  |
| --- | --- | --- | --- |
| order\_discount\_amount | numeric | 2.50, 1.99 | Dollar value of the discount. If you add up all of the values in this field for a particular order, it should match the value found in the discount field from the orders table. |
| data\_source | varchar | US Data Warehouse MM Hayes Data Source P2 Data Source | This field is a signifier of which source tables a row originated from. Very helpful for troubleshooting. |

437

USDW erp\_entity\_id cleaning logic

Some records out of USDW (eg. source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header ) have a profit center (PCN) name which contains an erp\_entity\_id (EEI). This PCN EEI may differ with the record’s EEI. If the PCN EEI exists in the

source.cg\_dw\_cdl\_us\_source\_edw\_org\_hierarchy\_dim table we can extract the PCN EEI instead. However, this may not always be the correct.

Currently, the following logic is applied for this cleaning process.

1. Join usdw records to analytics.sjsu\_profitcenter\_map which is a lookup table of manually verified proper\_EEIs. This join is done on both the EEI and PCN as some ids have multiple PCNs.

2. After proper\_EEI is extracted, simply use the existing usdw record’s EEI.  
3. When EEI is NULL attempt to extract the PCN EEI and check if it exists in the hierarchy table.

438

How to Query

This document is intended to aid developers in writing queries against the star schema. If you are not familiar with the concepts of fact and dimension tables, it may help to first read the Overview / Design document. Note that in this document, code snippets will be written targeting the dbt\_dev environment because we have not yet released to dbt\_prod. Once we have gone live make sure that any user facing queries are targeting the dbt\_prod schema in order to prevent any wild developer shenanigans from affecting user experience.

Table Names and Keys

Rather than reproduce an exhaustive list of tables (which can be found in the Data Dictionary) I’ll instead explain the naming conventions used.

Fact tables are named for both the business process event that they describe and the grain that the data is presented at. For example, order data presented at the detailed item level is named order\_details and order data presented at the order grain is simply called orders.

Dimension tables always begin with the prefix “dim\_” followed by a word or phrase that attempts to describe the contents within. Eg: dim\_fiscal\_date contains date descriptors, and dim\_locations contains descriptions about the location / unit where an order occurred. Note that it is possible for a type of dimension data to be represented at more than one grain level. At release the only example we have of this is dim\_locations\_terminals\_brands, which provides descriptors about which terminal or brand within a unit processed the order.

Improving Query Performance

We’ve applied compound sort keys to our tables, and if you use these fields in your where clause or joins they can significantly improve the speed of your query by allowing the database to minimize the amount of disk I/O and network broadcasting it needs to do.

**Orders**: key, order\_id, location\_fk, location\_terminals\_fk, order\_fiscal\_datetime\_fk, pickup\_fiscal\_datetime\_fk, order\_origin, order\_fiscal\_date\_fk, data\_source

**Order\_Details**: order\_fk, order\_id, location\_fk, location\_terminals\_fk, order\_fiscal\_datetime\_fk, order\_fiscal\_date\_fk, data\_source, lineitem\_name, key

**Dim\_Locations**: key, unit\_id, erp\_system\_id, sector, country

**Dim\_Locations\_Terminals\_Brands**: location\_fk, key, unit\_id, terminal\_id, brands, erp\_system\_id, sector, country

**Dim\_Fiscal\_Date**: key, fiscal\_year\_id, fiscal\_quarter\_id

**Dim\_Fiscal\_Datetime**: key, date\_name, hour, minute

Avoid using the order by clause in your queries against this data model if possible, as this requires your results to first be sent to the leader node which can significantly increase processing time.

Examples

Querying Order Summary Data

Queries for order summary data will center around the orders table. In this example we will pull the net sales for August 2022 as well as the count of orders.

1 select

1. 2  o.order\_fiscal\_date\_fk as order\_date
2. 3  , o.order\_origin
3. 4  , sum(o.net\_sales) as net\_sales

439

Notes:

1. Key is a unique field for every order. When you want a count of unique orders and you’re pulling data from multiple units, it’s important to use this rather than order\_id. The order\_id field contains the original order id from the unit’s source system, and is not guaranteed to be unique across systems.
2. Order\_origin tells us how the order was placed and tends to be important to users. Eg mobile orders, pos orders etc. Often when people think there are more orders / sales than expected, it’s because their expectations were founded on knowledge of a single order origin’s volume. This data model unifies all of those into one presentation.

Querying Order Summary Data - Selected Units  
A query of this type will involve both the orders table and dim\_locations. This example pulls all of the August 2022 data for a single unit.

1. 1  select
2. 2  l.unit\_id
3. 3  , l.erp\_system\_id
4. 4  , o.order\_fiscal\_date\_fk as order\_date
5. 5  , o.order\_origin
6. 6  , sum(o.net\_sales) as net\_sales
7. 7  , count(o.key) as num\_orders
8. 8  from dbt\_dev.orders o
9. 9  inner join dbt\_dev.dim\_locations l on l.key = o.location\_fk
10. 10  where o.order\_fiscal\_date\_fk between '2022-08-01' and '2022-08-31'
11. 11  and l.unit\_id = '44964' and l.erp\_system\_id = 1001
12. 12  group by l.unit\_id, l.erp\_system\_id, o.order\_fiscal\_date\_fk, o.order\_origin

Notes:

1. We select / filter by both unit\_id and erp\_system\_id when writing a query where locations are relevant, because unit ids are not unique, but the combination of unit id and system id **are** unique (or they should be, if this is ever not the case it is likely because someone goofed when configuring a new unit).
2. We join to dim\_locations in order to access and filter by the unit\_id / erp\_system\_id. We could also use this joined table to pull the org hierarchy data for this unit, such as sector or region.  
   Note: This query takes twice as long as the first example to run because the join to dim\_locations is a bit expensive. We are likely to update our order fact tables to include the unit\_id and erp\_system\_id, so that this kind of basic filtering can be done without joins.

Querying Order Details - Selected Units

A query of this type will start with the Orders table and join to Order\_Details. This join is extremely fast because the tables are organized in a way that allows merge joins. Then we join to Dim\_Locations to allow filtering on the unit, and pick a date range to limit the volume of data retrieved to a reasonable amount.

5 , count(o.key) as num\_orders  
6 from dbt\_dev.orders o  
7 where o.order\_fiscal\_date\_fk between '2022-08-01' and '2022-08-31' 8 group by o.order\_fiscal\_date\_fk, o.order\_origin

1. 1  /\*This query pulls ~163,000 orders for a single unit\*/
2. 2  select
3. 3  l.unit\_id
4. 4  , l.erp\_system\_id
5. 5  , l.unit\_name
6. 6  , o.order\_id
7. 7  , od.lineitem\_name
8. 8  , od.quantity
9. 9  , od.lineitem\_subtotal

440

10

11

12

13

14

15

16

, od.lineitem\_net\_sales

, od.lineitem\_total\_sales

from dbt\_dev.orders o

inner join dbt\_dev.order\_details od on od.order\_fk = o.key

inner join dbt\_dev.dim\_locations l on l."key" = o.location\_fk

where o.order\_fiscal\_date\_fk between '2022-08-01' and '2022-08-31'

and l.unit\_id = '44964' and l.erp\_system\_id = 1001

441

SQL Template for Optimus / Star Schema

1. 1  select
2. 2  o.order\_origin,
3. 3  o.app\_name,
4. 4  o.data\_source,
5. 5  l.unit\_id,
6. 6  o.customer\_fk, /\* applies to o.order\_origin in ('mobile', 'web') \*/
7. 7  o.order\_fiscal\_date\_fk order\_date,
8. 8  o.order\_fiscal\_datetime\_fk order\_datetime,
9. 9  o.pickup\_fiscal\_datetime\_fk pickup\_datetime, /\* applies to o.order\_origin in ('mobile', 'web') \*/
10. 10  o.order\_id,
11. 11  o.subtotal,
12. 12  od.grain\_lvl, /\* applies to o.order\_origin in ('mobile', 'web') \*/
13. 13  coalesce(tier\_mobile.cleansed\_tier\_1, tier\_pos\_kiosk.cleansed\_tier\_1, 'unknown') tier\_1, /\* mobile tier tab
14. 14  coalesce(tier\_mobile.cleansed\_tier\_2, tier\_pos\_kiosk.cleansed\_tier\_2, 'unknown') tier\_2,
15. 15  coalesce(tier\_mobile.cleansed\_tier\_3, tier\_pos\_kiosk.cleansed\_tier\_3, 'unknown') tier\_3,
16. 16  coalesce(tier\_mobile.cleansed\_tier\_4, tier\_pos\_kiosk.cleansed\_tier\_4, 'unknown') tier\_4,
17. 17  od.lineitem\_name,
18. 18  od.lineitem\_subtotal,
19. 19  od.quantity
20. 20  from dbt\_dev.orders o
21. 21  left join dbt\_dev.order\_details od on od.order\_fk = o.key

22

1. 23  /\* location joins.
2. 24  only use dim\_locations for unit\_names/region/sectors etc. unless you specifically need brands then just use dim
3. 25  left join dbt\_dev.dim\_locations l on l.key = o.location\_fk /\* = od.location\_fk \*/
4. 26  left join dbt\_dev.dim\_locations\_terminals\_brands lt on lt.key = o.location\_terminals\_fk /\* = od.location\_termin

27

1. 28  /\* date and time joins.
2. 29  the foreign keys (fks) for order\_fiscal\_date\_fk and order\_fiscal\_datetime\_fk are already date and datetime fiel
3. 30  these tables have useful precalculated fields such as fiscal period and quarter \*/
4. 31  left join dbt\_dev.dim\_fiscal\_date d on d.key = o.order\_fiscal\_date\_fk /\* = od.order\_fiscal\_date\_fk \*/
5. 32  left join dbt\_dev.dim\_fiscal\_datetime dt on dt.key = o.order\_fiscal\_datetime\_fk /\* = od.order\_fiscal\_datetime\_f

33

1. 34  /\* tier joins. 1st is pos\_kiosk tiers. 2nd is mobile tiers
2. 35  pos\_kiosk tiers are for o.order\_origin in ('pos', 'kiosk')
3. 36  mobile tiers are for o.order\_origin in ('mobile', 'web') \*/
4. 37  left join datascience\_item\_dim.consolidated\_item\_attributes\_dbt tier\_pos\_kiosk on od.lineitem\_name = tier\_pos\_k
5. 38  left join pbi\_operator\_assistant.consolidated\_item\_option\_attributes\_mob\_dbt tier\_mobile on
6. 39  o.order\_id = tier\_mobile.orderid and
7. 40  od.parent\_lineitem\_id = tier\_mobile.itemid and
8. 41  od.grain\_lvl = tier\_mobile.grain\_lvl and
9. 42  od.lineitem\_name = tier\_mobile.menuitemname

43

1. 44  /\* generally should apply the order\_origin and transaction\_type filters below \*/
2. 45  where
3. 46  o.order\_fiscal\_date\_fk between '2022-01-01' and '2022-01-02'
4. 47  and o.order\_origin in ('pos', 'kiosk', 'mobile', 'web') /\* ('funding transaction myqc', 'other', 'non-mmhay
5. 48  and o.transaction\_type in ('Sale', 'Unknown', 'Other') /\* ('Cancel', 'Refund', 'Partial Refund', 'Void') \*/
6. 49  and l.region = 'Bank of America Region'

l

\_ a

d

k

i

e

442

Star Schema DevTeam Workspace

Pages under this section are written by and for the development team working on the star schema. Testing notes, design notes, records of key field definitions etc.

443

Field Definitions

On this page we will document any specific business rules / calculations etc around what fields mean, which will help us when it comes to transforming raw source data into fields with a consistent meaning.

**Subtotal:** Gross sales  
**Discounts (sometimes referred to as “contra-revenue”):** Discounts + promotions **Net Sales:** Subtotal - discounts + service fee + delivery fee  
**Total Sales:** Subtotal - discounts + service fee + delivery fee + tax

**datastore.mobile\_orders**

approximate sales formulas  
**itemsubtotal** = itemnetsales + itemdiscountsubtotal + itempromosubtotal  
**ordertotalamount** = ordernetsales + ordertaxamount  
**ordernetsales** = itemnetsales (items) + itemnetsales (options) + servicefeeamount + deliveryfeeamount **orderdiscountamount** = itemdiscountamount  
**orderpromoamount** = itempromosubtotal

Issues with his include records from Nourish. There are item level records which are themselves a promotion with negative itemsubtotal. This value feeds into the order’s orderpromoamount

444

Data Validation

Jira epic to collect and track tickets created by this testing: https://teamideaworks.atlassian.net/browse/BIA-106 - Can't find link The full time period for mobile testing is 2019-10-01 through 2022-09-30  
comparing datastore.mobile\_orders to dbt\_dev.orders

1. Mobile checks diff (ticket - https://teamideaworks.atlassian.net/browse/BIA-107 - Can't find link ) dw has the following unit as POS

1 46144 - Limetree Bay Retail-Bar & Conv checks difference:

1. 1  137 checks:
2. 2  select count(distinct orderid) checks
3. 3  from datastore.mobile\_orders
4. 4  where orderdate = '2019-10-01' and unitnumber = 46144

5

1. 6  105 checks:
2. 7  select count(\*) from dbt\_dev.orders o where
3. 8  o.order\_fiscal\_date\_fk = '2019-10-01' and
4. 9  o.location\_fk = '113b54cd144fe70f78953b93f353a4b2'

2. “Unknown” unit in dbt\_dev for T2E that is not in mobile\_orders (ticket - nd link )

https://teamideaworks.atlassian.net/browse/BIA-108 - Can't fi

1. Unit 99999 in mobile\_orders not in dbt\_dev
2. should the order origins of ‘ios’ and ‘android’ be ‘mobile’? (there are very few records with OS type)
3. bunch of units in datastore.mobile\_ordesr that are not in dbt\_dev (csv below)
4. bunch of units in dbt\_dev that are not in datastore .mobile\_orders (csv below)
5. pickup time issue is I think the root cause of much of the mobile discrepancy:

joining to dim\_fiscal\_datetime will sometimes return two date\_times

8. All these units can be found in datastore.mobile\_orders table (looker link) but they are not found in dbt\_orders for order\_origin = ‘mobile’ (ticket - https://teamideaworks.atlassian.net/browse/BIA-110 - Can't find link )

(1942, 11647, 23437, 31569, 40040, 56030, 56124, 57839)

1 select distinct o.order\_id ,o.pickup\_fiscal\_datetime\_fk ,t2.key ,t2.date\_name pickup\_date  
2 ,t2.date\_timefrom dbt\_dev.orders o  
3 left join dbt\_dev.dim\_fiscal\_datetime t2 on t2.key = o.pickup\_fiscal\_datetime\_fkwhere o.order\_id = '03o9YkaM1L

|  |  |
| --- | --- |
| **units in dbt\_de... ers.csv**  01 Nov 2022, 07:53 PM | **units in datasto...dev.csv**  01 Nov 2022, 07:48 PM |

result: 0

445

9. repeating lineitem records: (ticket - https://teamideaworks.atlassian.net/browse/BIA-109 - Can't find link )

the only difference in the records is the key. Originally I was looking to confirm that an itemname was unique in a given order, I think it should be?

10. Location unit is active check to create a flag to see if the location is live. P2:

1. Location name have ‘Do not use' but is showing ‘A’ (Active) in scd\_record\_status. Also ‘I’ or 'D’ in scd\_record\_status means Inactive and Delete

select appname, locationname, unitid, scd\_record\_status from "source".p2\_group\_locations where upper(locationname) like '%NOT%'

2. Location name shows twice for unit 19815 and marked as active.

select distinct locationname,unit,unitid,scd\_record\_status from "source".p2\_group\_locations plb where unitid=19815

T2E: shows ‘Do not use’ in the name but isenabled is checked off (not active)

select cb.name as unitname , rp.companybrandid as unitid , cb.isenabled from source.tap2eat\_companybrand cb left join source.tacit\_restaurantprofile rp on cb.id=rp.companybrandid where unitname like '%NOT%'

1 select \* from dbt\_dev.order\_details 2 where order\_id = '538860018MIC'

1 select count(distinct unit\_id)  
2 from dbt\_dev.orders o  
3 inner join dbt\_dev.dim\_locations u on u.key = o.location\_fk  
4 where u.erp\_system\_id = 1001 and order\_origin = 'mobile'  
5 and u.unit\_id in (1942, 11647, 23437, 31569, 40040, 56030, 56124, 57839)

446

11. Savour app difference between datastore.mobile\_orders and dbt\_dev.orders

appname Savour

app\_name SAVOUR

transactions sales 139 884.56

transactions sales 286,583 2,005,530.27

dbt\_dev.orders have added extra filters in close\_term\_id =APP and THRIVEAPI compare to datastore.mobile\_orders. Will need to make the changes in datastore.mobile\_orders to match.

upper(coalesce(oh.close\_term\_id, '')) in ('API', 'APP', 'THRIVEAPI')

12. subtotal from orders and lineitem\_subtotal slight variance.

order\_id lineitem\_subtotal subtotal 12473187 14 0

When subtotal is calculated (quantity \* price) the source data between the item and orders do not match as some prices include discounts and some do not. We do expect some variances but minimal.

POS+Kiosk Data Checks (Sravya)

Time Range: 2019-01-01 to 2022-09-30  
Units: all units in OA list ( pbi\_operator\_assistant.oa\_units\_master\_copy ) Data pull from DBT table:

1 2 3 4 5 6 7 8 9

10 11 12

select

u.unit\_id as erp\_entity\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM') as month

,(sum(isnull(discount,0)) + sum(isnull(subtotal,0))) as sales

,count(distinct o.order\_id) as checks

,sum(item\_quantity) as item\_qty

from dbt\_dev.orders o

inner join dbt\_dev.dim\_locations u on u.key = o.location\_fk

inner join pbi\_operator\_assistant.oa\_units\_master\_copy m on u.unit\_id = m.erp\_entity\_id

where

447

Data pull from source pos\_header table:

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

SELECT

oh.erp\_entity\_id

,to\_char(oh.close\_datetime,'YYYY-MM') as month

,sum(isnull(sales\_amt\_gross,0))+sum(isnull(abs(discount\_amt),0)) as sales

,count(distinct pos\_order\_hdr\_key) as checks

,sum(total\_item\_qty) as item\_qty

FROM source.cg\_dw\_cdl\_us\_source\_can\_pos\_order\_header oh

inner join pbi\_operator\_assistant.oa\_units\_master\_copy m on oh.erp\_entity\_id = m.erp\_entity\_id

WHERE

GROUP

erp\_system\_id = 1001 and

trunc(close\_datetime) between '2019-01-01' and '2022-09-30' and

UPPER(oh.check\_outlier) = 'NO' AND UPPER(oh.record\_status) <> 'DELETED' AND

UPPER(oh.erp\_entity\_id) <> 'TEST' AND /\*remove test data\*/

UPPER(coalesce(oh.agilysys\_void, '')) <> 'VOIDED CHECK'

BY oh.erp\_entity\_id

,to\_char(oh.close\_datetime,'YYYY-MM')

Data pull from MMH pos/kiosk

1 2 3 4 5 6 7 8 9

10 11 12 13

SELECT revenue\_unit\_number as erp\_entity\_id

,to\_char(transactiondate,'YYYY-MM') as month

,sum(isnull(subtotal,0))+sum(isnull(abs(discount),0)) as sales

,count(distinct patransactionid) as checks

,sum(itemquantity) as item\_qty -- this is not helpful here for order level so not checking qty

FROM datamart.mmhayes\_lineitem\_mods mmh

inner join pbi\_operator\_assistant.oa\_units\_master\_copy m on mmh.revenue\_unit\_number = m.erp\_entity\_id

WHERE -- erp\_system\_id = 1001 and

trunc(transactiondate) between '2019-01-01' and '2022-09-30' and

order\_origin in ('pos','kiosk') and grain\_lvl = 'order'

GROUP BY revenue\_unit\_number

,to\_char(transactiondate,'YYYY-MM')

**pos\_data\_checks.xlsx**

02 Nov 2022, 10:42 PM

all differences can be seen highlighted in the excel file

Highlighting biggest differences :

1. Unit 55174, August 1st - 30th 2022: (solved - the 8957 count includes mobile orders) a. checks from pos\_order\_header (i.e., pos and kiosk) = 8957

b. checks from dbt\_orders (for pos+kiosk) = 6869

13

14

15

16

17

18

19

o.order\_fiscal\_date\_fk between '2019-01-01' and '2022-09-30' and

u.erp\_system\_id = 1001 and

order\_origin in ('pos','kiosk')

group by

u.unit\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM')

448

c. so 23% checks missing  
2. Unit 44289, June 1st to June 30th 2022: (ticket -

a. checks from mmh datamart: 1472 b. checks from dbt\_orders: 2944  
c. almost 100% increase in orders

3. Unit 45045, April 1st - April 30th 2022: (ticket -

1. sales from pos\_order\_header = $1.175 million
2. sales from dbt\_orders: $897K
3. 24% missing sales

https://teamideaworks.atlassian.net/browse/BIA-111 - Can't find link )

https://teamideaworks.atlassian.net/browse/BIA-112 - Can't find link )

d. Note: @Kris Spinney it’s very likely my sales metrics are not aligning and I’m not pulling sales the same way from pos\_order\_header and dbt\_orders, please let me know if this is the case!

4. Unit 42551, Oct 1st - Oct 31th 2020: (ticket - a. sales from pos\_order\_header = $259,981

b. sales from dbt\_orders: $25,750 c. 90% of missing sales

5. Unit 27306, June 1st - June 30th 2020:

1. sales from mmh table in datamart = $2770
2. sales from dbt\_orders: $136 million!

https://teamideaworks.atlassian.net/browse/BIA-113 - Can't find link )

c. I think mmh table in datamart must have some way to remove test orders that we are missing. Cause the number of check for this unit/month is only 482 (checks is matching between mmh datamart table and dbt\_orders table)

6. Unit 17945, Aug 1st to Aug 31st 2022:  
a. sales from mmh table in datamart = $27,498

b. sales from dbt\_orders: $48 million c. Same issue as above (#5)

Continuing checks: November 9th after latest refresh of star schema tables  
7. Several units where **sales ($)** is off between pos\_order\_header and dbt\_orders. sales = abs(discount) + subtotal

One example:  
Unit 47912, Mar 1st to Mar 31st 2022: sales from pos\_order\_header: $10,555 sales from dbt\_orders: $16,969

Another example:  
Unit 42041, month = oct 1st to 31st 2020 sales from pos\_order\_header = $3324

449

sales from dbt\_orders = $72,652  
(lots of cases like this, just picked a random one)

8. Units that are in **pos\_order\_header** but missing from **dbt\_orders** (for pos/kiosk data)  
3356 4807 6103 8109 9761 19559 22010 22209 25534 30247 43986 44225 44452 44527 46709 51136 51744 51788 52843

53334 55470 56030 57400 58420

9. Several units where pos/kiosk order counts between **pos\_order\_header** and **dbt\_orders** are off by more than 5% (fairly sure these are not mmh units).

Discrepancy could be for some or all months between 2019-10 and 2022-09

Example unit 56030, month = Sept 1st to 30th 2022 Checks from dbt\_orders (pos/kiosk): 4885  
Checks from usdw pos\_order\_header: 1509

Sales is also off for the same units

10. Unit 40639 has orders for pos/kiosk in dbt\_orders but this unit is not in pos\_order\_header table or in mmh source. Wondering where it’s coming from

For dates: (between 2019-08 to 2022-08). In 2022-09 the unit can be seen in pos\_order\_header

1 2211,4339,4807,14505,14516,14994,16475,16476,16477,16479,16481,20353,21187,22279,  
2 25908,28329,28452,30523,32002,32077,40153,40639,41368,42041,42527,42730,42824,43697, 3 43986,44712,46106,46716,47912,51136,56030,56084,56126,57400,57562,58420

**poskiosk\_order... ing.csv**

11 Nov 2022, 03:27 PM

450

Data Validation - OA Tables

Mobile Checks dbt orders:

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

select

u.unit\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM') as month

,sum(isnull(subtotal,0))) as sales

,count(distinct o.order\_id) as checks

,sum(item\_quantity) as item\_qty

from dbt\_dev.orders o

inner join dbt\_dev.dim\_locations u on u.key = o.location\_fk

inner join pbi\_operator\_assistant.oa\_units\_master\_copy m on u.unit\_id = m.erp\_entity\_id

where

o.order\_fiscal\_date\_fk between '2019-01-01' and '2022-09-30' and

u.erp\_system\_id = 1001 and

order\_origin in ('mobile','web')

group by

u.unit\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM')

OA mobile table:

1 2 3 4 5 6 7

select unit\_id,to\_char(order\_date,'YYYY-MM') as month,

sum(transactions\_order\_level) as transactions\_order\_level,

sum(sales\_amt) as sales\_amt

from pbi\_operator\_assistant.orders\_mobile\_dw d

inner join pbi\_operator\_assistant.oa\_units\_master\_copy oa on d.unit\_id = oa.erp\_entity\_id

where order\_date between '2020-01-01' and '2022-09-30'

group by unit\_id,to\_char(order\_date,'YYYY-MM')

--  
There are 2 reasons why checks are off:

1. Unit 42041, all months the order counts are off for this unit.  
2. Order counts are off only for month 2022-06-01 to 2022-06-30

451

example, unit 22928, june 1st to 30th 2022, checks in dbt = 1430  
checks in orders\_mobile\_dw = 1298

example, unit 28826, june 1st to 30th 2022, checks in dbt = 3538  
checks in orders\_mobile\_dw = 2394

example, unit 31917, june 1st to 30th 2022, checks in dbt = 1387  
checks in orders\_mobile\_dw = 1042

example, unit 31906, june 1st to 30th 2022, checks in dbt = 1809  
checks in orders\_mobile\_dw = 1364

3. Sales are off for several units (even though order counts are matching) Currently using sales from dbtorders from column = subtotal

some examples:

unit 31927, june 1st to 30th 2021, sales in dbt = $2461.8  
sales in orders\_mobile\_dw = $1612.3

unit 40040, aug 1st to 31st 2021, sales in dbt = $2023.6  
sales in orders\_mobile\_dw = $1412.3

Full list of units:

1 "1021,1769,1790,1793,6103,6139,6220,8109,8755,8808,10114,11642,11652,11654,11935,  
2 12674,12679,12683,13343,13605,13660,13669,13672,13735,14213,14491,14505,14994,15005, 3 16403,16433,16504,17945,18359,18466,19559,20170,20224,20225,20226,20233,20353,20430, 4 20548,20701,21792,22002,22005,22010,22240,22279,22462,22463,22491,22522,22528,22530, 5 22532,22924,23265,23500,23701,23702,26065,26325,26910,26935,27304,27495,28153,28199, 6 28753,31547,31574,31805,31811,31854,31896,31897,31898,31899,31901,31902,31903,31904, 7 31905,31906,31907,31908,31909,31910,31911,31913,31914,31915,31916,31917,31921,31922, 8 31923,31926,31927,32092,32126,32545,40040,40835,41145,41151,41495,42041,42822,43802, 9 44452,44818,44823,45027,45048,45049,45050,45051,45052,45053,45054,45055,45058,45059,

10 45060,45062,45071,45509,45986,46251,46740,47571,47802,47896,50242,51136,51759,52204, 11 52311,52314,52315,52317,52389,52594,52810,52811,53334,55470,55478,56084,56105,56108, 12 56109,56124,56787,57562,57753,57754,57836,57837,57838,58298,58599,58656,58742,59477, 13 59672,59947"

452

CSV file:

**salesmob\_dw\_... dbt.csv**

14 Nov 2022, 04:19 PM

POS Checks dbt orders:

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

select

u.unit\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM') as month

,sum(isnull(subtotal,0))) as sales

,count(distinct o.order\_id) as checks

,sum(item\_quantity) as item\_qty

from dbt\_dev.orders o

inner join dbt\_dev.dim\_locations u on u.key = o.location\_fk

inner join pbi\_operator\_assistant.oa\_units\_master\_copy m on u.unit\_id = m.erp\_entity\_id

where

o.order\_fiscal\_date\_fk between '2019-01-01' and '2022-09-30' and

u.erp\_system\_id = 1001 and

order\_origin in ('pos','kiosk')

group by

u.unit\_id

,to\_char(order\_fiscal\_date\_fk,'YYYY-MM')

orders\_pos\_dw:

1 2 3 4 5 6 7

select unit\_id,order\_date,sales\_type,

sum(transactions\_order\_level) as transactions\_order\_level,

sum(sales\_amt) as sales\_amt

from pbi\_operator\_assistant.orders\_pos\_dw d

inner join pbi\_operator\_assistant.oa\_units\_master\_copy oa on d.unit\_id = oa.erp\_entity\_id

where order\_date between '2019-01-01' and '2022-10-31'

group by unit\_id,order\_date,sales\_type

1. A couple units where in DBT, order counts > 0 but sales = 0. These are not in order\_pos\_dw. Examples:

Unit 59947, month Sept 1st to 31st 2022:  
dbt order count: 2597 (dbt subtotal column = 0) order\_pos\_dw order count: 0

Unit 32558, month Nov 1st to 30th 2020:  
dbt order count: 1989 (dbt subtotal column = 0)

453

order\_pos\_dw order count: 0

2. Order counts not matching up between dbt and order\_pos\_dw for following units:

1 "1112,1326,2395,6020,8734,12674,12683,13669,13672,14213,14219,14491,16292,16294,  
2 16295,16296,16301,16306,16307,16320,16403,16501,17697,19967,20225,20353,20548,20701, 3 20705,20976,20978,20991,22002,22005,22010,22528,22530,22532,22812,22814,23267,23792, 4 26935,27306,27435,27495,28153,28154,30223,30279,30342,30509,30909,31923,32001,32136, 5 32366,32451,32493,32547,32548,32555,32556,32557,32558,32616,36151,40070,40153,40639, 6 41287,41291,41368,41495,44452,44966,45045,45047,45050,45052,45053,45055,45058,45059, 7 45071,45408,45986,46223,46739,46740,50745,52861,53379,55202,56107,57562,58583,58585, 8 58586,58587,58588,58589,58590,58593,58599,58683,59947"

Specific example:

Unit 45058, month Sept 1st to 31st 2022: dbt order count: 97,844  
order\_pos\_dw order count: 1903

Unit 28153, month Aug 1st to 31st 2022: dbt order count: 3923  
order\_pos\_dw order count: 1095

CSV FILE:

3. sales is off for very few units:  
Unit 16501, month May 1st to 31st 2019: dbt subtotal: $306,279.98  
order\_pos\_dw sales\_amt: $258,995.03

Unit 22814, month Jan 1st to 31st 2021: dbt subtotal: $19216  
order\_pos\_dw sales\_amt: $18393

**checks\_not\_ma...\_dw.csv**

14 Nov 2022, 05:01 PM

454

455

Star Schema Architecture

**This page provides a general overview of how the star schema works from a design perspective. This is primarily intended for a developer audience.**

Database Schemas

Everything is organized into two schemas: DW and DW\_Staging (during the prototyping phase, schemas subject to change during productionalization)

DW\_Staging  
This is the first layer of processing, where we take our first pass at transforming raw source data into a model that more closely resembles the final output. This data is meant only for consumption by star schema transformations.  
Tables and / or views are created for facts and dimensions from each raw source.  
For example - **orders\_t2e** contains all of the order header data (a fact) we’ve received from T2E, and **order\_details\_t2e** contains the line item details (another fact, at a different grain) for all of the orders we’ve received from T2E. Likewise, **t2e\_locations** contains all of the unit and “terminal” location data found in T2E source (a dimension).  
Note: It is not necessary for a staging table to be created for every dimension we want to extract from a source. It may be possible to perform a dimension transformation out of the staging fact table: eg - we could extract customer dimension data from a table like orders\_t2e. The main reason to create a separate set of dimension tables at the staging layer is if it is easier / faster to get to the end result without processing the entire set of facts from a source.

DW  
The tables and views in this layer contain data from all of the various sources in a consistent format. Creating these tables involves unioning the various staging tables, performing transformations as required to get field types and sizes into a state where the results can be merged together. We may want to provide direct access to these tables for consumers, or we may want to create a final presentation schema that further cleans up the data and generates slices of the data to improve performance for specialized use cases. Still to be determined.

**Naming Conventions:**Dimension table / view names begin with “dim\_”, eg: **dim\_location**Fact table / view names begin with the word that describes the business process data being captured. For example, **dw.orders** contains summary data for every order and **dw.order\_details** contains line item details for every order.  
**Key:** Every table / view will contain a column named “**key**” - this is intended to be a unique value for that row. Typically we achieve this by performing an MD5 hash that takes in a list of string values. When we work on adding a new source of data to this schema, special care must be taken to ensure that this key value is unique. First we must ensure that the key is unique within the set of new data that is being added, and then ensure that the keys are also unique within the overall final dataset. This typically involves a bit of iteration and investigation into what we need to add to the hash in order to get a unique value.  
**Foreign Keys:** Fact tables will contain columns in the following naming pattern - DimensionTableName\_fk (eg: **location\_fk** or **fiscal\_date\_fk**). These can be used to link a single fact with detailed dimension information for the purposes of filtering or providing more data in a presentation layer. When we are inserting new data to the table, rather than performing a lookup on the dimension table it is preferable to generate the hash value using the same MD5 hash that was used for the relevant source in the dimension table. We simply need to look up what values were used in the dimension script, and then ensure that our staging data contains all of the relevant values to perform the same hash. This avoids network and disk I/O overhead during processing.

456

Adding a New Data Source to Star Schema

This is intended to serve as a guide for adding a new source of data into the star schema. Initially this guide will heavily reference the orders fact tables as this is what the model started with, but the same general process should also make sense when other fact tables are introduced. This is an iterative process involving discovery of what data is available in a new source, figuring out how to map fields within to required fields in the destination tables, how to extrapolate missing data if necessary, and finally flagging and filtering unwanted test data.

Note: Document is a work in progress

Initial Queries for Facts / Dimensions

The first step is to create queries off of the new data source, which pull data sets for the relevant fact and dimension tables that will be needed. These will be rough drafts, and will become what we call staging tables. These should represent data as it is shown at source - if transformations are required to extrapolate data or change how a formula works, it should be done in an entirely new field. The idea being that if we need to troubleshoot we can trace backwards from the star schema and see at what point an error occurs. This is especially important because if an error occurred at source, then we don’t want to spend a lot of time troubleshooting transformations to discover that. Querying the staging fact table will also be easier and faster than querying source itself, leading to less time spent on troubleshooting.

So as an example, when I began working on our Eatclub order data I started by taking a known Eatclub order query and materialized it to my eatclub\_orders\_staging table. Then I figured out how to write a queries that would retrieve all unit data and detailed location data (terminals / brands found within the unit).

Example

The very first version of the Eatclub orders query looked like this. Just a straightforward select statement which was later evolved through several iterations.

1. 1  select distinct
2. 2  'order' as grain\_lvl,
3. 3  o.id as order\_id,
4. 4  op.unit\_id,
5. 5  op."name" as operator\_name,
6. 6  h.erp\_system\_id,
7. 7  h.org\_district\_name as district,
8. 8  coalesce(r.brand\_name, 'Unknown Brand') as brands,
9. 9  o.transaction\_id,
10. 10  c.id as company\_id,
11. 11  coalesce(cast(o.user\_id as varchar), 'Unknown') as customer\_id ,
12. 12  cast(o.order\_datetime as date) as order\_date,
13. 13  o.order\_datetime,
14. 14  case when source is null then 'Web' else source end as order\_origin,
15. 15  o.location\_id,
16. 16  o.delivery\_date,
17. 17  o.delivered\_at,
18. 18  o.status,
19. 19  o.accounting\_type,
20. 20  o.sub\_total as net\_sales,
21. 21  coalesce(o.tax,0) as tax,
22. 22  coalesce(o.subsidized\_sub\_total,0) as discount,
23. 23  o.sub\_total - o.subsidized\_sub\_total+tax as total\_sales,
24. 24  o.etl\_extract\_timestamp,
25. 25  'Eatclub' as data\_source
26. 26  from source.eatclub\_cdl\_orders\_v o

457

Attempt to Union the Staging Data

Next we attempt to create a select statement from each of the staging tables to add to the union all query for the main fact / dim tables. As these tables are likely already defined, we can just take a select statement from another source and modify it to point to the new staging table. At this point it is highly likely that you will discover that some of the fields we’re trying to populate in the destination tables either don’t neatly exist in the original query from source, may exist but are unknown, don’t make sense in the context of the source data or are simply unavailable. This is where the iteration begins.

We now go down through the list of fields being pulled to insert into the destination table and anywhere that the field isn’t neatly and clearly mapped to a column in the staging table, we investigate the source tables to see if we missed anything. This may require contacting data engineering to locate unknown fields, or even a support agent for the vendor itself. It can also be that the required data can be calculated from available fields, in which case we want to update the staging table query to add this. We have a strong preference for keeping as much business logic as possible in the staging tables so that everything we need for investigating issues is in one place.

If a destination field cannot be calculated and the data just isn’t available, then mark it as such in a way that matches up with how other sources are flagging data as unknown / unavailable.

Simply continue cycling through this process over and over until all of the fields have something in them that makes sense.

Example

As an example we again take the Eatclub orders data. At the end of it, the (DBT-ized) union for this data looked like the sample below. We needed to work on making the MD5 hashes on lines 4, 7 and 8 produce fully unique values within the results. This had been complicated by Eatclub having the ability to include several brands within a single order. This meant every multi-brand order would generate one row per brand included in the order - we had to find a solution for that which involved adding a CTE and a count to the prototype query. That allowed us to detect the multibrand orders and collapse them into one row per order. We also didn’t have an item count available in the prototype query because that data wasn’t provided at the order header level by Eatclub. This resulted in an item\_details CTE to pull the count from the detail records for each order.

27

28

29

30

31

32

33

34

35

36

37

38

39

inner join source.eatclub\_cdl\_main\_operator\_v op on op.id = o.operator\_id

left outer join source.eatclub\_cdl\_orders\_items\_v oi on oi.order\_id = o.id and oi.operator\_id = op.id /\*we join

left outer join source.eatclub\_cdl\_items\_v i on i.id = oi.item\_id and i.operator\_id = op.id

left outer join source.eatclub\_cdl\_restaurants\_v r on r.id = i.restaurant\_id

inner join source.eatclub\_cdl\_companies\_v c on c.id = op.main\_company\_id /\*if op.main\_company\_id is null, we're

inner join source.eatclub\_cdl\_delivery\_locations\_v dl on dl.company\_id = c.id /\*if this join filters an order o

inner join source.eatclub\_cdl\_locations\_v l on l.id = dl.location\_ptr\_id

left join source.cg\_dw\_us\_org\_hierarchy\_geocoded h /\*left join here, because some main\_operator records have nu

on h.erp\_entity\_id = op.unit\_id

and (

(ceiling(h.geocoded\_lng) = ceiling(l.coordinates\_x) and ceiling(h.geocoded\_lat) = ceiling(l.coordinates

or (lower(h.org\_unit\_city) = lower(l.city) and lower(h.org\_unit\_state) = lower(l.state) )

)

1. 1  union all
2. 2  select
3. 3  cast('order' as varchar) as grain\_lvl
4. 4  , MD5(coalesce(cast(e.order\_id as varchar), 'UnknownOrderID') + 'Eatclub' + coalesce(cast(e.unit\_id as varc
5. 5  , cast(e.order\_id as varchar) as order\_id
6. 6  , e.order\_origin
7. 7  , MD5(coalesce(cast(e.unit\_id as varchar), 'UnknownEatclubUnit') + coalesce(cast(e.erp\_system\_id as varchar
8. 8  , MD5(coalesce(cast(e.brands as varchar), 'UnknownBrand') + coalesce(cast(e.unit\_id as varchar), 'UnknownEa
9. 9  , cast(e.customer\_id as varchar) as customer\_id /\*to be replaced at some point by a hash to a customer dime
10. 10  , cast(e.order\_date as date) as order\_fiscal\_date\_fk
11. 11  , date\_trunc('minute', e.order\_datetime) as order\_fiscal\_datetime\_fk
12. 12  , date\_trunc('minute', e.delivered\_at) as pickup\_fiscal\_datetime\_fk
13. 13  , cast('Eatclub' as varchar) as app\_name

458

u l

\_

h

) t n

Data Cleanup

Get your mop and bucket, it’s time to put on your data janitor hat! We want to write some queries that will test the internal consistency of the data. As an example, in our order data we know that net sales should be the subtotal plus applicable fees (delivery and service typically), minus any discounts. If you write a query that calculates this for each row, you may find orders where the result is not what is expected. This sort of testing can reveal unexpected behavior in how the source system records data, and help you modify the staging queries to be more accurate. Another example we ran into was figuring out that some sources recorded order times in UTC and others did not. By comparing datetime fields to each other we were eventually able to work out which ones needed to be converted. This overall process will also help you in figuring out what sort of DBT tests can be added to the yml file so that data quality can be monitored.

Note that this is an ongoing process that will likely be revisited from time to time as users discover issues in the data.

Discovering Test Data

Most of our data feeds include test data in them that we probably want to remove before it reaches the point that end users will see it. This is another detective game where we use source provided flags that indicate test data, or email addresses that are obviously not from real customers. Eg: an order submitted by CompassTestUser1@compassdigital.io is highly likely to be something we don’t want to include in our financial figures. As you evolve a heuristic that can detect test data without also catching non-test data, we will add this into a new field in the staging table that represents a true / false value. Then in the final union to the destination table we want to filter out the test rows. Note that this process is typically only important for the fact tables. Test data within dimensions won’t really do anything more than possibly add a few extra values to any dashboard filters that are populated by the dimension.

Note that this is an ongoing process that will likely be revisited from time to time as users discover unwanted data.

1. 14  , cast('Not Applicable' as varchar) as order\_type
2. 15  , cast('Not Applicable' as varchar) as transaction\_type
3. 16  , cast('Eatclub' as varchar) as payment\_method
4. 17  , cast('Not Applicable' as varchar) as employee\_id
5. 18  , cast('Not Applicable' as varchar) as employee\_name
6. 19  , e.subtotal
7. 20  , 0.00 as service\_fee
8. 21  , 0.00 as delivery\_fee
9. 22  , e.discount
10. 23  , e.tax as total\_tax
11. 24  , e.net\_sales
12. 25  , e.total\_sales
13. 26  , item\_quantity
14. 27  , cast('Eatclub Data Source' as varchar) as data\_source
15. 28  , e.etl\_extract\_timestamp
16. 29  from {{ ref('eatclub\_orders\_staging') }} e

459

Wezel Cloud Architecture Development

**Job Creator**

**key Components:**

Sitemap Parser

Create the Lambda function for sitemap parser SNS for failed sitemaps

Set up SNS Topic & Subscriptions

Create policies & role (?) Add Trigger & Destination

**Key Components:**

Dynamo DB Job Scheduler  
Table structure  
Choose correct Region  
Choose partition key & Sort Key(?)

PowerBI Set Up  
Dynamo db → (?) → PBI

Lambda Function Set Up (Failed jobs to SNS) Trigger

**Job Scheduler**

460

**Job Runner & Result Storage**

**Key Components:**

Step Function

Lambda Function Batches for Scrapers (js? typescript?)

Job Handler

Proxy manager

Lambda Function Batches for Parsers (python?typescript?)

Update parser results into JSON payloads

Push successful jobs & failed job into separate bucket for later uses

SNS Services

Deliver massages to Lambda function which update Dynamo DB in Job Scheduler Section

Deliver massages from parsers to Lambda function which stores successfully parsed data

**“Wrap Up”**

Transform the parsed data into standardized data table format Transfer the data into serving account

461

462

Data Technology Levels

February 2022

Overview

This document outlines the structure of the career ladder within DT, and its purpose is to guide career progression discussions. It’s represented by the two tables below, one for each of the 2 parallel career tracks. Parallel in this context means that neither track is meant to be incentivized over the other. Within each table, rows represent titles of progressively increasing seniority, and columns represent core skill sets for each track. In the case of engineer titles, descriptors (“data”, “data platform”, “machine learning”) were omitted for clarity’s sake. The stars in each cell represent the minimum bar for advancing into the level, and communicate relative scale. In terms of specific positioning, it’s a combination of quantitative metrics + qualitative feedback from the direct manager. This document is expected to be amended.

Tracks

**Technical track**

Level 5  
4  
3

2 1

Title  
Principal Data Scientist or Principal Engineer Staff Data Scientist or Staff Engineer Technical Lead  
Senior Data Scientist or Senior Engineer Data Scientist or Engineer

Primary technical skill\* Secondary technical skill\*\* Product Autonomy exceptional exceptional exceptional exceptional ★★★ ★★ ★★★ ★★★ ★★★ ★ ★★★ ★★★ ★★ ★ ★★ ★★★

★ ★ ★

Analytics skills for a data scientist, engineering skills for an engineer \*\* Engineering skills for a data scientist, analytics skills for an engineer

**People track**

Level Title  
6 Vice-President 5 Senior Director 4 Director  
3 Senior Manager 2 Manager

Definitions

**Autonomy**

Analytics / Engineering exceptional  
★★  
★★

★ ★

Leadership Product Autonomy exceptional exceptional exceptional ★★★ ★★★ ★★★ ★★★ ★★★ ★★★ ★★ ★★ ★★

★ ★★ ★★

As the name implies, autonomy represents the ability to complete increasingly complex units of work (ticket, collection of tickets, milestone projects) without or almost without supervision. This definition also includes relying on subject matter experts where appropriate, and escalating blockers.

463

**Product**

This pillar refers to the development of the capacity to work backwards from a customer problem towards a technical solution. Product also includes the ability to use one’s understanding of CDL (and the industry) in order to facilitate prioritization and the choice of the right technical solution, sometimes favouring simpler options over more complicated ones.

**Engineering**

A data scientist in DT at CDL is expected to have a good command of standard software engineering practices including authoring DRY code and versioning it, interacting with the cloud, and using databases/warehouses, while exercising ownership over their work from prototype to production. An engineer in DT at CDL is expected to be able to perform a number of functions ranging from sound architecture development, supporting ETL processes, improving the platform (incl. CI/CD + release management), and lending support to pods as needed.

**Analytics**

This pillar represents the ability to find answers to questions through iteration, typically by using descriptive or predictive models. It also involves the capacity for exercising a reasonable amount of skepticism, and being mindful of model generalizability. In terms of techniques, it’s firmly rooted in descriptive statistics, and could involve a variety of approaches ranging from straightforward SQL pulls to statistical tests or machine learning models.

**Leadership**

At its core, the leadership skill is about being focussed on serving the team, and determining optimal solutions under constraints. Operationally, it’s centred around communication; more specifically, feedback delivery, effective stakeholder management, deferring/escalating/working laterally, and routing traffic. This pillar also encompasses owning the career development of one’s direct reports and being directly invested into their growth.

464

SQL Developer Guide  
This section contains guides that aim to help you in writing queries that are easy to read and debug, maintain or modify and run well.

1. SQL Style Guide - This guide aims to help you write code that is easier to read and debug, making maintenance easier.

2. SQL Performance Guide - This guide contains some tips to improve performance as well as a deeper discussion on database architecture and table design.

465

SQL Style Guide

The way that your SQL is structured and presented has a large impact on how easy it is to read, as well as how quickly someone can read / understand it. These tips will make your queries easier to debug and modify.

SQL Keywords

It is traditional to write them keywords (eg: SELECT, FROM, WHERE etc) in upper case, as this draws the reader’s attention to the important actions in your code. For example if I’m reading a query and want to know where a particular field in the select list is coming from, I can pick out the alias for the table name and jump to the from clause to see which table it is. If this is written as FROM while everything else like table and field names are lower case, then I can much more quickly find what I’m looking for. Most SQL editors that have coding assistants will automatically capitalize these keywords.

A style I have frequently used in the past is to use upper case for keywords in the main part of the query, and use lower case for keywords that are in subqueries. This helps to make it more obvious to the reader that the code within the subquery is separate from the main query.

Object and Field Names

Always use snake case - Names like order\_pickup\_date are much easier to read than orderpickupdate. If you’re writing a transform that

selects from a source table that is not using snake case, then rename the field in your destination table.

Lower case only for names - Redshift enforces this already by changing names to lower case when creating tables, but if we write scripts that create objects with capital letters in them and migrate to another platform that allows for this, we could find ourselves with a lot of broken code.  
Descriptive names: a boolean should be named something like is\_duplicate - this both indicates to someone writing a query that the field is a boolean, and what a true value means. A date or datetime field should indicate both what the date represents and the fact that it is a date. Eg: order\_date or pickup\_datetime. This prevents a developer from having to look up the table schema while prototyping a query.

Joining Tables

Always alias your tables, and use the alias when pulling a field from that table even if it would not be an ambiguous reference. Not doing this will create headaches down the road when you have reason to join another table that has identical column names, or when you're debugging and need to comment out a join. Having and using these aliases makes it possible to identify and comment out any lines linked to the table you want to remove from the join list.

Always use the full join description: inner join, left join, right join. This makes it very clear to the reader what logic is being applied. Eg: if you just write join and omit "inner", then when someone is scanning the query their eyes will pick up on the join but then they need to go looking for left / right to understand what type of join is being done. It's faster to see if "inner" exists than does not exist.

Indenting

Use indenting, and newlines where appropriate, to visually separate fiddly logic from the main part of the query. For example when I create a CASE statement I'll have WHEN / THEN / ELSE statements indented one level further than the CASE and END lines which makes the logic easy to read and edit. Additionally, I put the END statement on it's own line following the final line of logic so that the reader can tell at a glance where the logic has ended and to make it easier to add more lines if necessary.

Use only tabs for indenting, not spaces. This helps to keep the indenting consistent, and provides a large enough visual difference to make it obvious when something is being separated. At a glance it is very easy to overlook indenting of one or two spaces, which is important when you're scanning a long query trying to identify a troublesome area.

466

Use indenting and newlines to make it clearer which bits of code are contained within a pair of brackets, where the list of fields selected starts / ends and the joins begin etc.  
In WHERE clauses, give each separate condition its own line. This makes is easier to see which filters are being applied, and to comment out / modify them when testing a query.

Examples CTEs

1 2 3 4 5 6 7 8 9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

**with**

*/\**

*\* Note here that we use one tab for the entire content of the cte, and also use an*

*\* additional tab for the content of the select list with each field on it's own line.*

*\* Another tip for making things easier is placing the commas before the fields.*

*\* This is helpful when you're debugging joins and need to comment out one or more lines.*

*\* When you place the comma at the end of the line you often wind up having to comment*

*\* out only the comma on the previous line when debugging.*

*\*/*

avg\_pickup\_time\_diff **as** ( **SELECT**

avg(datediff(**minute**, tor.createddatetime, tor.pickuptime)) **as** avg\_time\_diff **FROM** source.tap2eat\_tableorder tor  
**WHERE** tor.createddatetime <= tor.pickuptime

)

*/\* \* \* \*/*

*Note that every field is using a table alias*

*This is extremely helpful when debugging or modifying a query.*

, t2e\_orders **as** ( **SELECT**

tor.id **as** orderid  
, tc.id **as** tablecheckid  
, tor.restaurantid **as** brandid  
, tor.posorderid **as** volantetransactionid , tor.createddatetime **as** orderdatetime\_utc , tor.ordertype  
, tor.orderstatus  
, tor.pickuptime  
, ap.avg\_time\_diff  
, tc.poscheckprintlines  
, tcp.paymentmethod  
, tcp.cardtype  
, tc.discountamount  
, tc.subtotalamount  
, tc.servicecharge  
, tc.totalamount  
, tor.etl\_extract\_timestamp  
*/\**

*\* Note the structure of this CASE statement. The logic is easy to read / edit, and*

*\* if you need to comment out the entire block for testing it's easy to see where*

*\* it starts and ends.*

*\*/*

, **CASE  
WHEN** tor.clientappname IN ('Tap2Eat', 'Tap2Us') **THEN** 'Tap2Eat'  
**ELSE** tor.clientappname

**END as** appname

467

CASE statement

1 */\**

2  
3  
4 *\*/*5 **CASE** 6  
7  
8  
9

10  
11  
12  
13  
14  
15  
16  
17  
18 **END**

*\* This is a bit of a longer case statement, so we separate the logic and the output*

*\* values onto their own lines.*

**WHEN** terminals.terminalmodelid = 105 AND lio.poskioskterminalid IS NOT NULL **THEN** 'kiosk'

**WHEN** (terminals.terminalmodelid = 105 OR terminals.terminaltypeid = 7) AND lio.poskioskterminalid IS NULL **THEN** 'mobile'

**WHEN** terminals.terminalmodelid in (102,103) **THEN** 'pos'

**WHEN** terminals.terminaltypeid = 4 **THEN** 'non-mmhayes pos'

**WHEN** terminals.terminaltypeid = 8 OR lio.transtypeid = 11 **THEN** 'funding transaction myqc'

**ELSE**

'other'

**as** order\_origin,

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

**FROM** cdl\_bi\_edw.source.tap2eat\_tableorder tor  
**LEFT JOIN** source.tap2eat\_tablecheck tc **on** tor.id=tc.tableorderid and date\_trunc('minute',tor.createddatetim */\**

*\* Note again the indentation and newlines here on the subuery join. Newline after the open bracket, closin*

*\* with the entire select indented so that it's obvious the code inside has nothing to do with the main par*

*\*/*

**LEFT JOIN** (  
**SELECT DISTINCT**

cp.tablecheckid  
, c.tableorderid  
, LISTAGG(**DISTINCT** cp.paymentmethod, ', ') **WITHIN GROUP** (**ORDER BY** cp.cardtype) **as** paymentmethod , LISTAGG(**DISTINCT** cp.cardtype, ', ') **WITHIN GROUP** (**ORDER BY** cp.cardtype) **as** cardtype

**FROM** source.tacit\_checkpayment cp  
**INNER JOIN** source.tap2eat\_tablecheck c **on** c.id = cp.tablecheckid and date\_trunc('minute',c.createddatet **GROUP BY** cp.tablecheckid, c.tableorderid

) tcp **on** tcp.tablecheckid = tc.id and tcp.tableorderid = tor.id  
**LEFT JOIN** source.tap2eat\_discounts d **on** d.id=tc.discountid  
**LEFT JOIN** source.tacit\_customerdiscounts cd **on** tor.id = cd.tableorderid and cd.discountid = d.id **INNER JOIN** avg\_pickup\_time\_diff ap **on** 1=1

),

468

e

*g t*

i

SQL Performance

Writing fast and efficient SQL is a bit of a black art. This is partly because every database has query optimizer that tries to figure out what your sql is asking and attempts to rewrite parts of the queries it is running to speed up their processing\*. This means that for the most part it isn’t something we have to think much about and thus, never really have an opportunity to learn. The bigger part of why this so mysterious is that making queries really fast requires understanding all of how the sql is processed, how the data is stored on the physical disks and how that data is organized. Approaching the problem of writing queries or designing tables with only part of this understanding will result in queries that work, but not as well as they could.

General Tips - A collection of SQL knowledge to get you started without taking a deep dive Physical Storage - A deep dive into how row and column based databases work  
Logical Storage - A deep dive into how our data can be organized

Traditional RDBMS - Indexes within SQL Server, Oracle and the like

Columnar DBs - Discussion of distribution and sort styles Putting It Into Practice - Where the rubber meets the road

EXPLAIN It To Me - A brief rundown on the black art of reading EXPLAIN plans  
Operators - A partial list of the more interesting operators you will see in plans, and some tips on what could potentially be done to

improve performance.

*\* Note: The database can’t always do this well, either because the query is complex in a way that makes it difficult to rewrite or because of a limitation. For example, we discovered that when Airflow / DBT generates a “CREATE table AS select stuff from tables” query, Redshift has trouble simplifying the query and just runs the sql as you wrote it. So for the purposes of writing transforms that will create new tables, understanding performance is rather important.*

General Tips

1. **Avoid the use of SELECT \***This will be explained more in the section on database structure, but this is particularly important in columnar databases like Redshift and Snowflake. Selecting only the columns you actually need will be faster than selecting all of them. Explicitly naming columns can also help avoid ambiguous column references.  
   SQL IDEs often have a shortcut to take a SELECT \* FROM table query and expand the field list. In DBeaver you just need to place your cursor after the \* and hit ctrl-space. Then you can just remove the unwanted fields from your query.
2. **DISTINCT can slow you down**This can be a problem with very large resultsets, as it requires the entire query to be completed and stored in memory before the database can begin squashing out duplicates. If you’re having trouble with duplicate rows it’s better to investigate the root cause and find a way to filter out the unwanted rows. For example in a slowly changing dimension you may only want the most recent row, so you could use a rank window function to number the rows in descending value and then filter on row\_num = 1. Or there might be a field that you can filter on, eg: where active is true.
3. **UNION vs UNION ALL** - use the latter if possible, UNION applies a distinct to the entire query
4. **WHERE IN ( ... )** - When using this filter pattern try to keep it to reasonable list of hard coded values. If a subquery is needed to retrieve the list of values try to make it a fast query that returns a small resultset. If that query will return a large resultset consider using WHERE EXISTS instead.
5. **WHERE EXISTS** - this clause is useful where you need to check whether the data exists in another table, and can take the place of a complex WHERE IN statement. The benefit with WHERE EXISTS is that the inner query will stop processing when it reaches the first matching row.

469

Physical Storage

Understanding how data is stored on disk helps us to understand what operations our queries are causing, which of these can make our queries slower, and why we may want to alter how the data is organized on the disk. Each table we create is represented by a file on a server. Or in the case of a massive parallel processing system like Redshift, a file that is split up across different disks / servers. How the data is stored is important because disk reads are typically the slowest part of a query and we want to avoid full table scans as much as possible. Another significant bottleneck is network broadcasting of data. Understanding physical storage helps us understand when these might occur.

Traditional RDBMS (SQL Server, Oracle etc)

The most common type of database you’ll come across organizes these files in a row by row structure. If you select the top 10 rows from a table in this type of database, the server will perform 10 disk operations and return all of the fields you requested. The image below represents a hypothetical table in such a system, with each color representing a single disk operation.

A simple table in a row oriented database

Columnar Databases (Redshift, Snowflake etc)

These types of databases are very popular for analytical usage because they organize the data by column instead of row (thus, columnar). This means that to perform a sum of an entire column you only need a single disk operation, and those are by far the most expensive part of a query. So we wind up with a single disk operation that stuffs all of the requested data into fast memory, performs the aggregate and spits out the result. The image below represents the same table as stored in a columnar database, again with each color representing what can be pulled with a single disk operation. Since these tables can contain a very large amount of data, it is best to select only the fields you actually need. One extra disk read doesn’t seem like much, but that one disk read can potentially add hundreds of MB or more to the volume of data being transmitted.

A simple table in a column oriented database

In truth, Redshift makes the picture slightly more complex because it is also a distributed system. This means we’ll actually have multiple files, each containing a portion of the overall data. How the data is distributed and sorted in those files can have a massive impact on performance since they affect how much data needs to be transmitted to other servers and how much time is spent on disk reads.

Logical Storage

Traditional RDBMS (SQL Server, Oracle etc)

To speed up filtering and joins these databases allow you to create indexes either on individual fields or combinations of them. This allows the query engine to jump to the right point in a file being processing rather than running a disk read on every line in the file. Often, then, this means in these systems you can speed up a slow query by carefully editing the indexes on your tables.

470

Columnar Databases (Redshift, Snowflake etc)

An implication of columnar database design is that we can’t just play with indexes to speed up our queries, because indexes don’t exist (or make sense). Each line of a file represents a single field and all possible values found within it, so being able to jump to a specific spot in a file is just selecting a field. The key concept we need here becomes **distribution and sort styles**.

**Distribution Styles**

There are four styles we will discuss. The key thing to understand is that queries are run against each chunk of a file in the network and then the results are stitched together. This means if the tables are distributed evenly then we get the full benefit of multiple processors chewing away on requests, and if related data in separate tables are stored on the same nodes then we can avoid transmitting that data across the network.

1. **Key** - Dist keys, or distribution keys, tell the database how to allocate data among the various chunks of the table files. In the simplest approach all we care about is that the list of possible values is highly unique, and that there isn’t much variation in the number of rows where each value is present. In the above example, if we used “key” as the dist key then it would try to assign each row to its own server / file chunk. If this table will be joined to other tables, consider using the most commonly joined on column the key.

2. **Even** - This distribution style hands full control over distribution to the database and ensures that all nodes will have an equal share of the data. This isn’t very performant however, as the server knows nothing about how the data will be used and cannot help us by cleverly storing data. Only really useful if it just isn’t possible to select an efficient dist key.

3. **All** - This distribution style replicates the entire table on every node, meaning that the data contained within will never need to be transmitted to other nodes when joining with another table. This is a tradeoff between storage space and performance. Generally speaking you only want to use this on small tables, such as dimensions with a relatively small number of rows.

4. **Auto** - Don’t use this. Theoretically Redshift will analyze queries being run against your table and then determine if changing the dist key will improve performance, but I have yet to see it work well. It may be that this feature only works properly on systems with very high query volume.

**Sorting Styles**

There are three approaches to sorting. Sorting somewhat emulates the index feature on a row based database, allowing the query engine to skip over sections of the file it knows it doesn’t need for your query.

1. **Compound** - This is my preferred sorting method for sorting because it helps when you need to join your table, group / order by or use window functions. You can add as many sort keys as you like, and the order you provide the keys in matters. The first key in the list has the greatest impact on the ordering of the data, the second key sorts on that field within the distinct values of the first key and so on. If the first key in the list is also the dist key and is also the field used to join to other tables, then you unlock the power of the merge join.

2. **Interleaved** - This method allows you to specify up to 8 key fields, and gives equal weighting to each of them. This may work well in huge tables where only a few fields will be used for filtering, with a high variety of values. Probably best used on very wide tables that don’t require joins to other tables. One downside to this seems to be the initializing this sort style takes a very long time due on larger tables, and the sorting degrades quickly with a large enough incoming volume of data.

3. **Auto** - Don’t use this. Theoretically Redshift will analyze queries being run against your table and then determine if changing the sort key will improve performance, but I have yet to see it work well. It may be that this feature only works properly on systems with very high query volume.

Putting It Into Practice

So now that we understand how data storage and organization affect performance, how do we put this into practice in Redshift? We start by learning how to take a slow query and determine what is slowing it down. Often you can just look at the query and get an idea of some changes that will help, but to really get into it we can use the EXPLAIN keyword.

471

EXPLAIN It To Me

Putting the EXPLAIN keyword before the start of your select and running it will generate a description of the operations Redshift will run. This includes the type of operators and the cost to execute them. Below is an example query plan for a group by and count query, showing two steps, in the format: *XN operator (cost rows width)*

XN HashAggregate (cost=131.97..133.41 rows=576 width=17)  
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=17)

I’ve highlighted the most interesting parts of these steps in green. On the first line we see the type of operator, a hash aggregate. Cost is a relative value that we can use to compare operations within a single plan. The cost of the step, including all descendent steps, is represented by two floating point numbers separated by two periods. The first number, which on line 1 is 131.97, represents the cost to retrieve the first row of the results. The second number, 133.41, represents the cost to retrieve all rows. Width is the estimated width in bytes of the average row.

When the operator is a type of join, on the next line you’ll see what fields are being used to make the join happen. We can use this to figure out what part of our query this is referring to. Eg:

Hash Cond: ("outer".catid = "inner".catid) -> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=35) -> XN Hash (cost=0.11..0.11 rows=11 width=49)  
-> XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)

Now we know how to read the explain plan and how to track down which part of our code is responsible for each line of the plan. The next step then is to scan through our plan to find the step(s) that are having the greatest impact on our query performance and figure out if there is anything we can do to improve them. Certain operator types are best avoided as much as possible, and if one step seems like it has an inordinately large impact on the performance we might be able to change the table organization to improve it.

Operators

The tables below show a selection of operator types you may have optimization issues with, and some that indicate that the query and tables are working very well.

XN Hash Join DS\_BCAST\_INNER (cost=0.14..6600286.07 rows=8798 width=84)

**Join Operators**

Nested Loop (nloop)

**Description**

Slowest join - seen on cross joins and some inequality joins

**How to optimize if cost is high**

To fix this review your query for cross-joins and remove them if possible. Cross- joins are joins without a join condition that result in the Cartesian product of two tables.

|  |  |  |
| --- | --- | --- |
| Hash Join (hjoin) | Faster than nested loop. This type of join reads the outer table, hashes the joining column(s) and then finds matches in the inner hash table (which can be a disk based operation rather than memory)  Note that sometimes slow joins like these are inevitable because the dists and sorts were configured to allow merge joins with other tables. Sometimes getting the fastest results for some use cases requires a trade-off like this. | 1. Rewrite the query to use a merge join if possible 2. If the HJOIN step in SVL\_QUERY\_SUMMARY has a very high value in the rows field compared to the rows value in the final RETURN step in the query, check whether you can rewrite the query to join on a unique column. When a query does not join on a unique column, such as a primary key, that increases the number of rows involved in the join. |
| Merge Join (mjoin) | The fastest type of join, which requires both tables being joined to have the join column be the dist key (so the joining data is on the same disk) and a compound sort key where the first key is the same column as the dist key. As long as the tables have been vacuumed recently (no more than 20% unsorted), then Redshift can take the two files on each node, line them up side by side and stitch them together. | You don’t! This join is already running fast enough to leave fire tracks behind it. |

**Network Operators**

DS\_BCAST\_INNER

Indicates the entire inner table is being broadcast to all compute nodes, which is fine if the table is small.

Redesign the table distribution configs so that related data is located on the slices where it is needed, or change the join to use the existing dist keys. Note that sometimes changing your table distribution to avoid this operator would mean losing a merge join elsewhere in your query. Sometimes tradeoffs are just required.

DS\_DIST\_NONE / DS\_DIST\_ALL\_NONE

These are good, no need to change anything.

**Description**

Indicates the workload requires no broadcasting - all related data is on a single slice.

**How to optimize if cost is high**

472

DS\_DIST\_ALL\_INNER

DSDISTBOTH

**Misc Operators**

Scan

Sort

The entire inner table is redistributed to a single slice because the outer table uses DIST ALL, meaning we aren’t getting the benefit of parallel execution.

Idihdiibifbhbl

**Description**

A sequential scan of the table from beginning to end

Represents an ORDER BY or other sorting required by UNION, DISTINCT or window functions

Consider changing inner table to use a dist key rather than dist all, or using dist even on the outer table

Tdihji hdiibikd/idhih

**How to optimize if cost is high**

If this step is expensive look into adding the filters in your WHERE clause to the sort key. If they already exist there then the table may need to be vacuumed.

If this step is expensive look into adding the filters in your WHERE clause to the sort key. If they already exist there then the table may need to be vacuumed.

473

Roadmaps

474

FY23 Roadmap Briefs

Q2 Q3 Q4 (Jan—Mar) (Apr—Jun) (Jul—Sep)

**Data Engineering**

**Decision Science**

**Scout**

**Distilr**

475

Healthcare Migration Emily edited! Yay!

476

July Deliverable Tracking

Task Name

Business Attributes

HDL Data

Status

In Progress

Start

12-12-22

Finish Duration

07-31-23 166d

Assigned To

Require Note Compass  
AWS

Next Step

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Compass Healthcare Data Warehouse SnowFlake Requirements |  | In Progress | 12-12-22 | 07-31-23 | 166d | Dat Tran | N |  |  |
| Surfacide |  | Scoping | 01-09-23 | 07-31-23 | 146d | Dat Tran | N | Arriving as CSV via ESB  To Do:   1. Get ESB to reroute this data to CD sFTP Server 2. Chase down vendor for Historical data (1-2 Tables) 3. Sreeda to provide contact (Gary.nutbeam@surfacide.com) | ESB |

LarserFiche

PeopleHub Data

IBM Inforix - Cisco UCCX DB

Relay

NPC call centre tech

Associate tracking

Scoping

Scoping

Ready for development

In Development

01-09-23

01-09-23

01-09-23

01-09-23

07-31-23 146d

07-31-23 146d

07-31-23 146d

07-31-23 146d

Masoud Y Afghah

Dat Tran N

Masoud Y Afghah

Masoud N Afghah

Pending Compass' Managed AWS Account

Follow up with Michael to confirm he is sending us data

MyDinning (Daniel Kuhner) require a real time feed of the Relay Data into the MyDinning application

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| myResults - Oragami Data | Safety Reports / safety org trend / safety org detail from myStaff / Safety Penalty | Ready for development | 03-06-23 | 07-31-23 | 106d | Dat Tran | N | SFTP Created ESB is forwarding data from Originami to CD sFTP Need to request historical data from Data Services | ESB |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team Apps (Internal) [Team Coach, Team Fin (upgrade version of team coach)] via Daniel Kuhner's team |  | Scoping | 01-09-23 | 07-31-23 | 146d | Dat Tran | Y? | To Do:   1. Reach out to Arun K. cc Daniel Kuhner to get more information and details into how we pull this data down 2. Might be able to go through csv file via sFTP to retrieved data | Follow Up |
| BVFP Pulse |  | Scoping | 01-09-23 | 07-31-23 | 146d | Dat Tran | N | To Do:   1. Chase down a contact from BVFP Pulse (maybe via Chris D?) 2. Figure out logistics to get data flowing 3. Fall back option is to grab the data via E15 |  |
| SAP Morrison | PL Transactions Actuals Budgets Morrison Hierarches | Not Started | 12-12-22 | 07-31-23 | 166d | Dat Tran | Y | TO DO  1. Reach out to DBA team to get service account created  2. Sreeda is currently working on creating a service account  Sreeda to provide list of tables to be loaded | DBA |

477

eMorrison

Budgets Actuals Items Payroll Forecasts Bills/Invoice Inventory Transfers Hierarchies Retail

Not Started

12-12-22

07-31-23

166d

Dat Tran

Y

TO DO

1. Reach out to DBA team to get service account created

2. Sreeda is current working on creating a service account Sreeda to provide list of tables to be loaded

DBA

Huron

Not Started

01-09-23

07-31-23

146d

Dat Tran

N

"- Arriving as CSV via ESB To Do:

1. Get ESB to reroute this data to CD sFTP Server

2. Retrieve historical data from current SQL DW (Sreeda to pull)

ESB

478

DS+BI Touchpoints

**Feb 7th 2023**

**Attendees:**

@Brett McKitterick @Craig O'Connor @Sravya.Atluri @david.beallor

**Action Items:**

1. @david.beallor on OA board)

4. @Brett McKitterick to work with Angela and improve Mobile Reporting performance (reduce impact on OA) 5. @Sravya.Atluri to reach out to Alexis / Marta on following topics:

a. Getting direct access to more reporting portals (micros / agilysis) so we have a source of truth to compare our numbers against) b. Getting access to Nextep/Agilysis menu manager data

**March 7th 2023 Attendees:**

@Brett McKitterick @Craig O'Connor @Sravya.Atluri @david.beallor

**Action Items:**

1. @david.beallor to add Catertrax into DBT tables  
a. Jira ticket on OA board  
b. waiting on catertrax to send us historical data loads (last 3 years)

2. RLS document has been updated with following content:  
a. Edge cases are causing issues because of overwrite functionality. Because the latest record updates

3. @Sravya.Atluri to schedule some OA drop-in sessions in preparation for upcoming QBR for early April 4. @david.beallor to follow-up with Sarah Gordon’s team regarding myStaff

to add Catertrax into DBT tables (once we get the go-ahead from Dat’s team). For now, we have created a Jira ticket to prepare a working document about future improved implementation for RLS permissions: found here

to get access to Andy Fox team’s repo and see if we can identify the implementation logic of check\_outlier. David will let Craig/Sravya know if we need to work together on a new implementation of this logic

2. @Sravya.Atluri 3. @david.beallor

479

5. @Brett McKitterick @Sravya.Atluri look into usdw code for check\_outlier logic. @david.beallor to follow-up about missing tables ( CORE.POS\_CHECK\_OUTLIERS\_XREF )

**April 4th 2023 Attendees:**

@Brett McKitterick @Craig O'Connor @Sravya.Atluri @david.beallor

**Action Items:**

1.

2. 3. 4. 5. 6. 7.

8.

datastore.mobile\_orders potentially missing tiers for several itemnames @david.beallor to check how much sales we are missing tiers for  
Addition of an outlier flag column to mark orders with average check higher than 2 standard deviation: @david.beallor

@Sravya.Atluri to send an email update about Catertrax status to David and team

@david.beallor to follow-up on myStaff data source locations and to get the team access  
Canteen data to be added as a separate task from Catertrax (side note: should ‘vending machine’ be a separate type of order origin?)

@Craig O'Connor to request access for WasteNot portal given the contact: becky.high@compass-usa.com backlog DS item: can we create a model to tag test users for mobile

a. Sample training set: https://github.com/compassdigital/cdl-dataops-airflow- 2/blob/dev/dags/business\_intelligence/dbt/models/dw/staging/p2/p2\_test\_users.sql

@david.beallor to provide an update once we get more info on roles + RLS from myFinance + Compass IT team

480

Data Warehouse

481

Snowflake Maintenance

We have automated the commonly used Snowflake maintenance processes like:

User Creation and Modifications Database/Schema creation  
Modifications and Accesses  
User Role creations (level of access for the user)

The automation is performed through the GitHub CICD pipeline and the AWS LAMBDA process. Below is the GitHub repo:

https://github.com/compassdigital/dataops\_de\_snowflake\_maintenance - Connect your Github account Each Snowflake account can have a different branch to automate these common maintenance use cases.

Sample user JSON

1{

1. 2  "db\_object\_type": "user",
2. 3  "db\_object\_name": "hari.theniah",
3. 4  "config": {
4. 5  "default\_role": "dataops\_all\_db\_read\_role",
5. 6  "other\_roles": ["test\_db\_read\_role"],
6. 7  "password": null,
7. 8  "login\_name": null,
8. 9  "display\_name": null,
9. 10  "first\_name": null,
10. 11  "middle\_name": null,
11. 12  "last\_name": null,
12. 13  "email": null,
13. 14  "disabled": null,
14. 15  "default\_warehouse": null,
15. 16  "default\_namespace": null,
16. 17  "comment": null
17. 18  },
18. 19  "comment": "DataopsSource Database Read Access"
19. 20  }

Sample database JSON

1{ 2  
3  
4

"db\_object\_type": "database",

"db\_object\_name": "test\_db",

"config" : {

"schemas" : [

5  
6  
7]  
8 , "data\_retention\_time\_in\_days" : "30" 9 },

10 11 }

"comment" : "DATA-OPS Raw Data Source"

"datamart",

"datastore"

Sample role JSON

1{  
2 "db\_object\_type": "role",

482

Sample Network Policy JSON

1{ 2  
3  
4

5 6 7 8 9

10 11 12 13

"db\_object\_type": "network\_policy",

"db\_object\_name": "DATAOPS\_CDL\_USER\_IPS",

"config" : {

"allowed\_ip\_list" : {

"69.158.246.188" : "dat"

},

"blocked\_ip\_list" : {

} },

"comment" : "IP addresses to be whitelisted"

}

Below is the LAMBDA process, that needs to be cloned for each Snowflake account that needs to be automated:

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/dataops\_endeavor\_manage\_snowflake\_objects? tab=code

3 4 5 6 7 8 9

10 11 12 13 14

"db\_object\_name": "dataops\_all\_db\_read\_role",

"config" : {

"parent\_roles" : ["dataops\_edw\_read\_role"

, "dataops\_source\_read\_role"

, "cgna\_e15\_group\_e15\_cdl\_share\_read\_role"

, "dbt\_dev\_read\_role"

, "dbt\_prod\_read\_role"

],

"warehouses" : {"COMPUTE\_WH" : ["MONITOR", "USAGE", "OPERATE"]}

},

"comment" : "Dataops Read Role Chaining roles"

}

483

SAP Financial Data

Unmet needs

The SAP tables in the source schema have an ‘amount’ field that provides both sales and cost data. This field allows us to gauge the cost of food as well as labor. Using this field we can get profit and cost information that we don’t already have.

Objectives  
Add cost data for reporting using source.sap tables and provide coverage for existing sales in the orders table.

Jobs we want to cover

When I filter on a sector, I want to be able to view food and labor costs, cost% to sales, sales, profit, as well as digital penetration, so I can view key metrics for each sector.

**Key Terms**

unit\_id  
gl\_parent  
gl\_child terminal\_accounts dim\_accounts\_hierarchy

F-10 F-600 F-136 F-179

Some history

**Meaning**

Originally cost\_center in source.sap but with leading zeroes Parent function for hierarchy table  
Child of gl\_parent  
The account\_number for given parent function

Family tree hierarchy with each terminal account/account\_number for parent functions. Has 13 levels total

Gross Sales  
Amount Due to Account Gross Product Cost Total Labor Costs

The SAP data had already existed in the data warehouse but was hardly used since the orders fact table was created. It also had several fields and tables that needed cleaning. Several new models were created using what was in these existing schemas.

Constraints  
There is a data quality issue when comparing the orders units to the units in the SAP tables (nearly 400 units missing in orders and roughly

600 missing in the sap\_budget table ). This makes it difficult to provide a coverage map as the units aren't 1:1. Explorations + Decisions

The SAP models have several outliers and null values that make things difficult when compared to the orders table. To combat this there is a case statement to filter out the null values and a check outlier field that excludes some bogus units with drastic numbers.

The sales amount in SAP is also a lot larger than the order's sales, so there is a 12-month average measure that is used to be able to compare the two fields from each table. This allows us to see the correlation between the two fields.  
The sales amount in SAP doesn’t account for taxes (separate gl\_parent) so when reporting it is best to compare to the subtotal field in orders.

484

Using the data\_source field in orders we can determine the digital penetration in relation to SAP costs and as a percent of sales. Costs in the SAP table are higher than sales unless adding ‘amounts due to account’.

Next steps

**IN PROGRESS**

Determine the relationship between labor costs and digital penetration

485

AWS Account migration

CDL is currently working on migrating CDL managed AWS account into Compass managed account. The first step is to gain access to a nonprod account and ADO pipeline project. This page will be used to track progress.

**Proposal Slides**

**CloudFormation templates**

https://github.com/compassdigital/terminus-factory/tree/CloudFormation\_Templates - Connect your Github account **Next steps**

ADO pipeline migration will happen after account provisioning  
S3 link needs to be established for data migration  
Implement SonarQube integration  
Use Artifactory for container images management and common library packages

**CDL\_AWS\_Acc... st.pptx**

13 Mar 2023, 03:35 PM

486

Google Cloud

487

Snowflake vs. BigQuery Overview

**Business / Technology Driver**

We want to stay up-to-date with the current landscape of options for data warehouse solutions.

**Solution**

We will conduct a small, POC-grade A/B test of Snowflake’s performance against that of BigQuery. Specifically, we are going to emulate a real-life workload on Distilr using invoices from Supplychain and transactional data from Canteen.

**What success looks like**

A one-pager summary outlining performance against cost + comparing and contrasting unique features provided by either product. Preliminary comparison

**Access control model**

Snowflake supports role hierarchy, while IAM offers resource-based hierarchy for roles. Both support table-level access control as well as RLS.

**Multi-Cloud**

Snowflake is multi-cloud by design, and offers multi-cloud failover.

Within the BigQuery world, Anthos Multicloud API can be used as a way to span clouds (limited to a small number of operating regions). BQ Omni allows data ingest from AWS S3 and Azure Blobs.

**Scaling**

Automatic scaling was recently introduced to BigQuery. Its pricing model is based on fixed resource allocation, potentially posing the risk of leaving some concurrency unused. When spot increases in concurrency are required, availability may be limited.

Snowflake offers credit-based, pay-as-you-use pricing.

**Data Modelling**

The conventional way to use BigQuery would be to rely on denormalized tables, as opposed to star-schema tables as they are join-heavy. There may be a ceiling to row-level inserts on BQ.

**Cloning**

With BigQuery, there is really no equivalent to Snowflake’s Zero-Copy Cloning.

**Data Sharing**

Snowflake offers a high-concurrency API for sharing data. BigQuery’s primary data sharing option is mediated by IAM policies (e.g. adding users to instance).

**Caching**

With Snowflake, Distilr is currently using the 24-hr user-level caching functionality; there is no known BQ equivalent.

488

POC timelines TBD

Further reading

BigQuery vs Snowflake: The Definitive Guide  
Cloud Data Warehouse Benchmark | Blog | Fivetran  
Snowflake Vs BigQuery: Comparing Pricing, Performance & More  
From BigQuery Director to Snowflake user—with Dan Delorey, VP of Data at Sofi  
Benchmarking , Snowflake, Databricks , Synapse , BigQuery, Redshift , Trino , DuckDB and Hyper using TPCH-SF100 Migrating from BigQuery to Snowflake

**BQ Tech Comp... GNA.pdf**

30 Mar 2023, 08:57 PM

489