Week 1 - Data Engineering on the Cloud

Friday, June 14, 2024 12:59 PM

6.17.24 - 1.1

Friday, June 14, 2024 12:59 PM

Hartford
1.1
Data Engineering

- What is it?
  - Making data useable and accessible
  - Movement of data (raw to usable and clean)
  - Preparing data for use at scale.
- Top skills required (2-3) - Professional Skills
  - Collaboration within team (solve issues together)
  - Adaptable (learn/pickup new tech; not just be a master at one skill)
  - Understanding the business need
- Most common tools used
  - SQL, python, Snowflake, AWS

- We learned about tableau/ thought spot and I don't really think about that as
  - Medium to communicate things better/ tell a story
  - Effective communication (issue or solution)

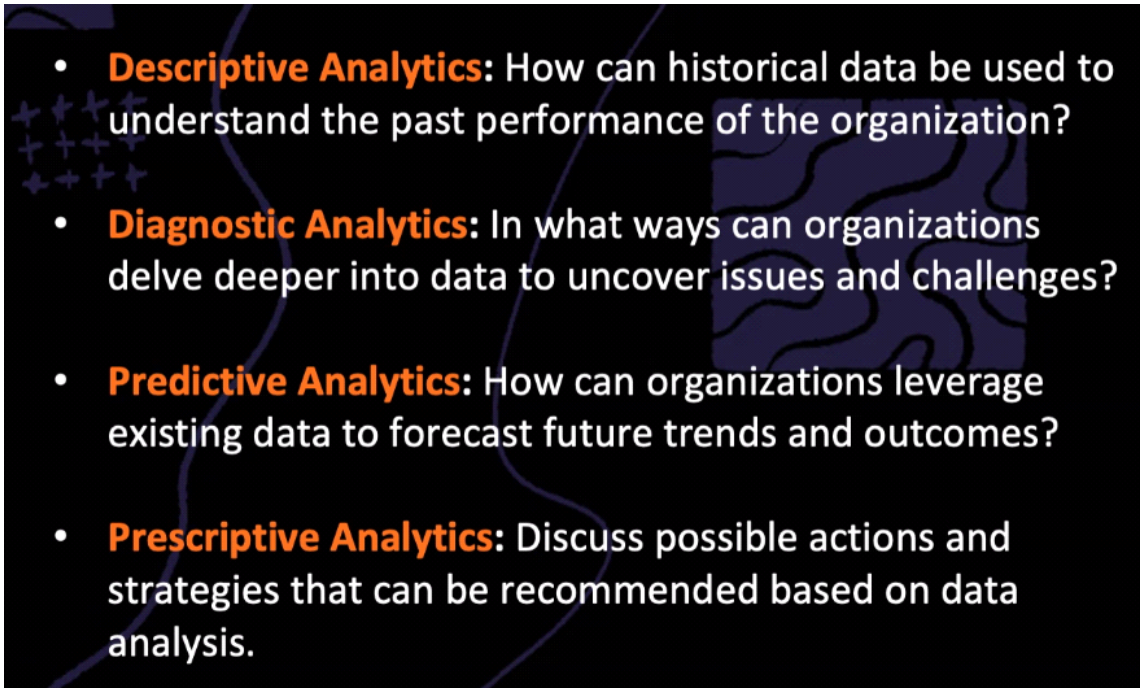
Ways an org can utilize the different types of analytics

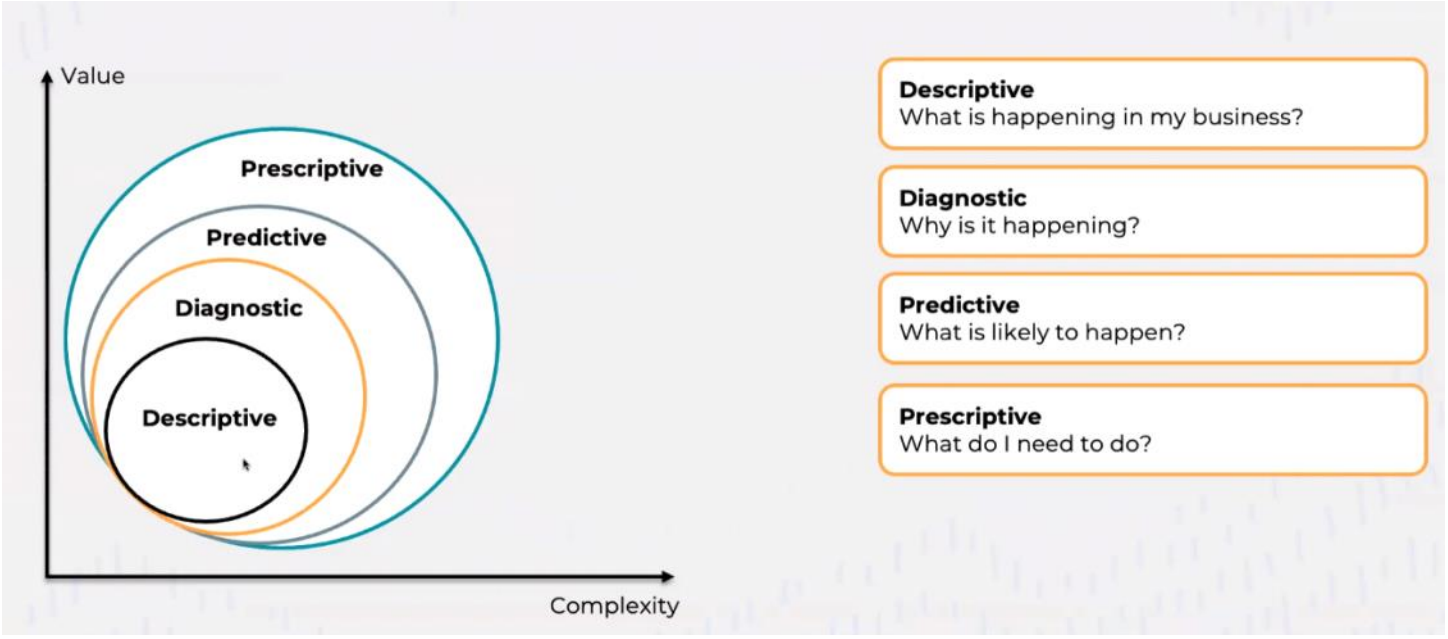
- Descriptive: Use historical data to figure out what happened.
  - Find patterns and relationships that could help describe why a success/failure occurred
  - Better understand changes in a business and if we are implementing changes we can use this data to feel more secure/confident in our decision, due diligence with our decision/ numerical evidence with our decision
  - Ex: profits month over month

- Diagnostic
  - Eval overall performance from descriptive analytics, now we can see which metrics aren't up to standard and undercover those problems
  - Bench mark for metrics and identify the cause of trends
  - Figuring out if its external or internal (not a lot of clicks on website and find out there is a problem in the code)
  - Ex: Learn that our technology is out of date or Finding that problem in the code
  - Ex: jan/feb/march profits are down in ice cream and diagnostic we can see temp is low
  - Defect analysis

- Predictive:
  - Combine external factors and existing data to identify a trend
  - Predict how data should move
  - Help in making financial goals/kpis
  - Ex; look at trend from prev year and based on how data is moving this year we can tweak the model and have better prediction for this current year

- Prescriptive:
    - Use visualizations to show what's going on and tell story and use that to guide recommendations
    - Use the predictive analytics and take our model and apply it to a different location/apply elsewhere and use this as our recommendation





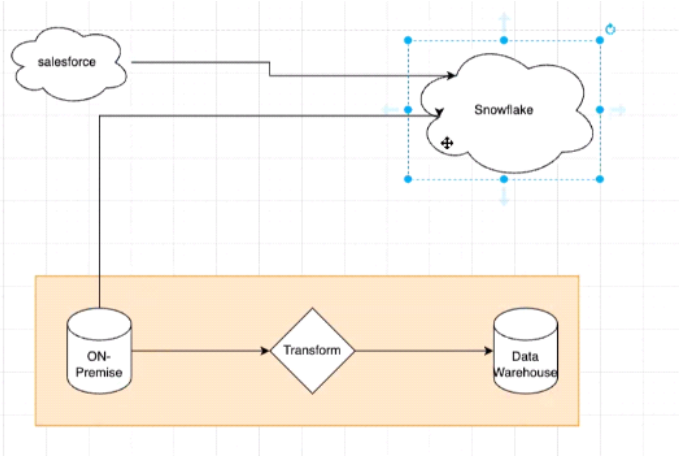
Big Data

- Massive data that can be processed using traditional methods
- 3 V's:
    - Volume
        - Huge amounts of data
    - Variety
        - Comes from lots of difference sources
    - Velocity
        - Batch Processing: a set time for data update/refresh (ex: bus waits for everyone to get on the bus to leave)
        - Real Time: data is updated continuously (ex: escalator brings people upstairs constantly)
- Latency: data can come at different speeds
- Important to undercover hidden patterns and correlations, deeper insights
- Importance lies in how it is used
- Challenge is in the storage, processing and analysis

Data Sources:

- On premises
- Cloud

-



- Types
    - Data bases (relations, non-relational)
    - File based (excel, csv, Json, xml)

Forms

- Structed: has a fixed style/organization (ex: excel sheet, sql)
    - Easily queryable
    - Has defined manner or schema, typically found in relational db
    - Organized in rows and columns, type is well defined
- Unstructured: no predefined structure (ex; email or social media post)
    - Not easily queryable without preprocessing
    - Various formats
    - Does not have a specific schema
    - Tends to be text heavy
    - Ex: video, audio files. Images, text files without a fixed format
- Semi structured: a hybrid (ex: Json, XML)
    - Elements might be tagged or categorized in some way
    - More flexible but still a bit chaotic
    - Neither raw or fitted to a conventional database system
    - Ex: xml or Json

- Ex: json files for policy information, XML files for claim processing

Modern Data Architecture

- Data lakes, lake houses, cloud
- In order to have chat gpt, chat bots, ai we need clean and trusted data that has been integrated from multiple sources
  - Goal is to bring all this data into one place that is integrated (snowflake is where we store that clean and trusted data)

- One source of truth, homogenous data (don't need to connect to each source every time), don't want to touch the source directly (decoupling),
- Data ware house - Snowflake
  - Pay for storage (cheap) it is the engine that is expensive bc running 24/7
  - Sql query engine
- Data lake - S3
  - Just paying for storage not the engine
- Lakehouse
  - Lower storage cost, less data movement
  - Simplified schema
  - Can be expensive and time consuming to build and transfer to this
  - Data lake but with additional functionality

6.18.24 - 1.2

Tuesday, June 18, 2024 7:52 AM

Review:

- End goal is to empower people to make data driven decisions and leverage different types of analytics
- AI accelerates the exposure of defects --> need to spot those errors faster

- Misconception about Big data is that it is just a lot of data (wrong)
    - 6 V's: variety, volume, velocity, value, variability, veracity
- Big Data Stack: 3 Layers
    - high volume low cost storage --> low cost scalable computing --> analytics
- Structed: organized in a schema, found in relational DB, has consistent structure, organized in rows and columns
- Unstructured: no predefined structure, not easily queryable without preprocessing, various formats
- Semi structured: has some level of structure but not as organized, elements may be tagged or categorized in some way, second most common
- Modern Day Architecture: Data Warehouse, Data Lake, Lakehouse

How to connect to a data source:

- JDBC: java db connectivity
- ODBC: open database connectivity
- Raw logs
- APIs
- Streams
- Csv. Xml, Json, Avro, parquet, orc

ETL: Extract Transform Load

- Process used to move data from source systems into a data warehouse
- Writing code gives you more flexibility
- Can use the drag and drop / simplest routes if needed
- We will be working with what people already wrote

- Extract
    - Retrieve raw data from source systems, which can be DB, CRMs, flat files, APIs or other data repos
    - Ensure data integrity during the extraction phase
        - Checks and balances to make sure nothing dropped and everything is correct
    - Can be done in real time or in batches
        - Reading data live or from steaming sources (Kafka)
        - Batch/ bulk extraction is usually daily
    - IO (input output functions)
- Transform
    - Convert extracted data into a format suitable for the target data warehouse
    - Use Data Mapping to figure out what we are doing/ what we need to do
    - Can involve various operations:
        - Data cleansing (remove duplicates, fixing errors)
        - Data enrichment (adding additional data from other sources)
        - Format changes (string manipulation)
        - Aggregates or computations (calculating totals or averages)

        - Encoding or decoding data
        - Handling missing values
- Load
    - Move the transformed data into the target data warehouse
    - Can be done in batches or in a streaming manner
    - Ensure that data maintains its integrity during the loading phase

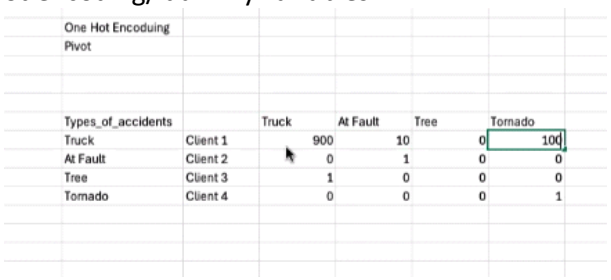
ETL, ELT, and Reverse ETL

- ETL: extract, transform, load
- ELT: extract, load, transform
    - More common with data lakes
- Reverse ETL: extract form the data warehouse, transform, load to the transactional system/third party app
    - Not common in companies

Top 5 Transformations:

- Filtering:
    - if our source has multiple different types of claims but we only want car accidents, we can filter for just the type car accidents to bring into the target source
- Sorting:
    - using an ORDER BY clause in SQL to sort by highest to lowest profit
- Aggregating:
    - if we are getting the data real time from the source every minute, we can aggregate to get the average for every day
    - Counting the total number rows after batch extraction to ensure integrity
    - Sum total values for month to ensure transferred corrected, sum of source match sum of where we brought it
- Mering:
    - combing our source with additional data from other sources to add more information and value
- Encoding:
    - if we are looking at submission of claims and want to make this a binary variable--> transform yes into 0 and no into 1

- Partitioning
    - If we have a dataset of everything from 2000 but we are only focused on 2023 and 2024, we can create a partition of just those two years. This will make everything run faster since less data to go though
- Concatenation
    - When data is partitioned by year and we can merge/stack/concatenate them on top of each other to have one big dataset rather than three mini ones
- Joins
    - Making the data wider, adding more columns
- Pivoting
    - One hot encoding/ dummy variables

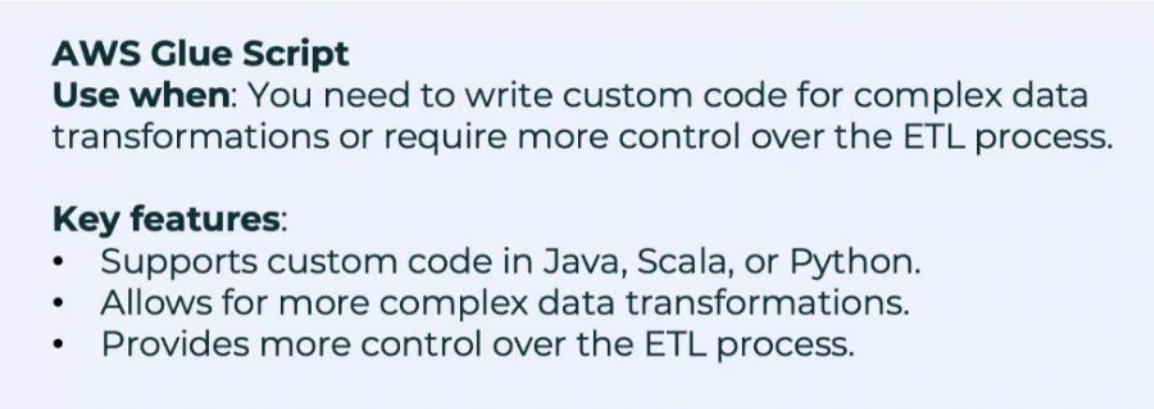


One Hot Encoduing
Pivot

Types_of_accidents		Truck	At Fault	Tree	Tornado
Truck	Client 1	900	10	0	100
At Fault	Client 2	0	1	0	0
Tree	Client 3	1	0	0	0
Tomado	Client 4	0	0	0	1

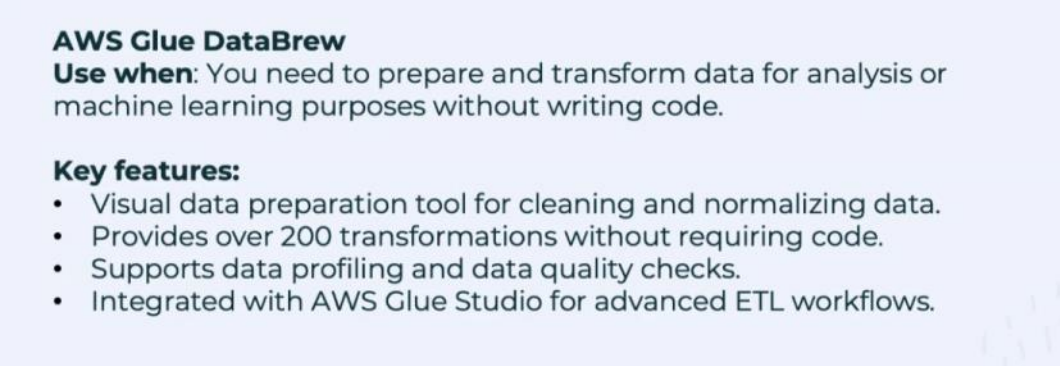
AWS Glue:

- Use when you need to write custom code for complete data transformations or require more control
- For the visual option, when it is red you know there is something wrong with it
- Integrates with data brew
-



AWS Glue DataBrew:

- Used by data scientist mostly
- No code required
- Can see distributions, missing data, data profiles
-



Dev: where you build everything
UAT/QA/Testing: where you test everything and make sure it is correct
Prod: this is when it is finalized and now we can automate it

Dev, Dev1, Dev2.. Is different GitHub branches or if multiple people are working on the same tool but different project, different groups, to keep track of billing from each group, or it is a promotion process (once satisfying req of dev then goes to dev1 then QA then prod)

Orchestration

- Process of dependency management, facilitated though automation
- Data orchestrator manages scheduling, triggering, monitoring and resource allocation
- Workflow management, automation, error handling, recovery, resource optimization, observability,

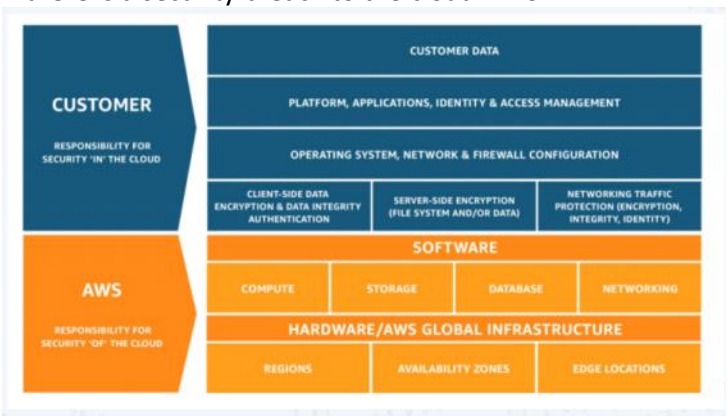
debugging, compliance/auditing, monitoring, alerting
- Amazon Step Function & Apache Airflow
    - Different services working together in synergy
- Creates dependencies
- DAGs(airflow) - one way direction/ branches of tasks that need to be done in order
    - Ex: if Joe's ETL fail then send an email to Joe he needs to fix it, once Joe's is all set then can move on to Alina's ETL/Task
- Can have event/trigger based (once this happens then start the process) tasks (or scheduled based (set time and frequency)

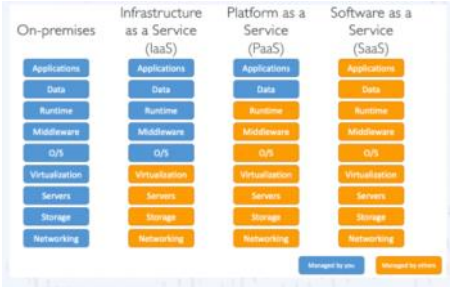
AWS Console
https://developintelligence.signin.aws.amazon.com/console
Ginamarie.Mastrorilli

AWS:
- IAM
    - identity and access management
    - Create users, permissions, resource management, access

Intro to AWS:
- Types of Cloud Computing
    - On premises
        - Everything is managed by you
    - Infrastructure as a service (IaaS)
        - The hardware is managed by them and everything else is managed by you
        - Cheapest
        - Ex: EC2 on AWS
        - Infrastructure as a Service (IaaS) contains the basic building blocks for cloud IT and typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.
    - Platform as a Service (PaaS)
        - Everything except applications and data managed by AWS/Cloud
        - Platform as a Service (PaaS) removes the need for your organization to manage the underlying infrastructure (usually hardware and operating systems) and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.
    - Software as a Service (SaaS)
        - Everything managed and we are just a user
        - If anything happens it is on the vendor not on you
        - Software as a Service (SaaS) provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email which you can use to send and receive email without having to manage feature

additions to the email product or maintain the servers and operating systems that the email program is running on.

- AWS Shared Responsibility model
  - If something happens in the cloud It is on you
  - If there is a security breach to the cloud AWS will fix
  - 

  - 

- Not every service is available in each region
  - New service might be launched in one region and only available in those zones
- Availability Zones (AZ)
  - Buildings, data zones
  - Each region at minimum has 3
  - Have to be at least 300 miles apart (estimate) to protect from natural disasters
  - Connected by fiber optic cables
  - Data flows between all 3 to protect against disaster recovery
  - Nobody knows where they are

Amazon S3

- Use Cases: Backup and storage, disaster recovery, archiving, hybrid cloud storage, application hosting, media hosting, data lakes, software delivery, static website
- A lot of companies use s3 to capture stats on different services bc you can store metadata in it
- Save everything in S3 but save only what you need in snowflake
- Buckets
  - Store object "files" in buckets, think of them as dictionaries
  - Must have a globally unique name
  - Defined at the region level
  - Much cheaper to store all data in an S3 bucket than in snowflake bc in snowflake the "engines are always running" whereas in a bucket its just where its held
- Objects (files)
  - Must have a Key

    - Key
        - Key is the FULL path:
            - s3://mybucket/myfile.txt
            - s3://mybucket/folder1/subfolder1/myfile.txt
        - Composed of a prefix + object name
            - s3://mybucket/folder1/subfolder1/myfile.txt
    - No concept of dictionaries within buckets
    - They are just keys with very long names that contain "/" slashes
    - Object values are the content of the body: Max object size is 5TB, if large it must be "multi-part" upload.
    - Version ID if versioning is enabled
    - Metadata

- To make bucket not private
    - Permissions (Amazon S3 > Buckets >my bucket)
        - Block Public Access --> uncheck block public access
- To enable version control
    - (Amazon S3 > Buckets >my bucket > my file)
    - Object management overview --> Bucket properties > enable versioning
    - Can also do on bucket level in Properties
    - If accidently delete a version/ file, can go to version and reupload it

6.20.24 - 1.3

Thursday, June 20, 2024 7:45 AM

Review:

- GitHub: aws-CloudFormation-templates

Cloud Shell Interface

- Aws --version
- Aws s3api create-bucket -- bucket gina-inclass-demo1 --region us-east-1
- Aws s3api rm :s3://gina-inclass-demo1 --recursive
- Aws s3api delete-bucket --bucket gina-inclass-demo1 --region us-east-1

CloudFormation:

- Declarative ay of outlining AWS infrastructure
- We are just declaring the resources to use
- For example, within a CloudFormation template, you say:
    - I want a security group
    - I want two EC2 instances using this security group
    - I want an S3 bucket
    - I want a load balancer (ELB) in front of these machines
    - Then CloudFormation creates those for you, in the right order, with the exact configuration that you specify
- Infrastructure as Code
    - No resources are manually created, changes to the infrastructure are reviewed through code
- Reproducibility
- Cost
    - Each resources within the stack is tagged with an identifier to see how much a stack costs
    - Can estimate with a template
    - You can save money by having automation deletion of templates at 5pm and then recreate at 8am

AWS Athena

- Sits on top of S3 and makes it act like a database, even though S3 is just storage
- Query engine
- Good for analysis or something quick
- Renting an engine to make S3 act like a database
- Serverless data lake
    - Means you aren't paying for it 24/7 only when u use it
    - Snowflake, we are paying for the engine all day everyday

AWS Glue

- Not just for ETL
- Glue is based on Spark engine
- ETL
    - Automatic code generation (example Python)

- Encryption
  - Server-side (at rest)
  - SSL (in transit)
- Can be event-driven
- Can provision additional “DPU’s” (data processing units) to increase performance of underlying Spark jobs
  - Enabling job metrics can help you understand the maximum capacity in DPU’s you need
- Errors reported to CloudWatch
  - Could tie into SNS for notification
- Transform data, Clean Data, Enrich Data (before doing analysis)
  - Generate ETL code in Python or Scala, you can modify the code Can provide your own Spark or PySpark scripts Target can be S3, JDBC (RDS, Redshift), or in Glue Data Catalog
- Fully managed, cost effective, pay only for the resources consumed
- Jobs are run on a serverless Spark platform
- Glue Scheduler to schedule the jobs
- Glue Triggers to automate job runs based on “events”

6.21.24 - 1.4

Friday, June 21, 2024 7:45 AM

Amazon Athena

- When to use:
  - Data Analysis:
    - Analyzes unstructured, semi-structured, and structured data stored in Amazon S3.
    - Supports various data formats including CSV, JSON, Apache Parquet, and Apache ORC.
  - Ad-Hoc Queries:
    - Run ad-hoc queries using ANSI SQL.
    - No need to aggregate or load data into Athena.
  - Integration with Visualization and BI Tools:
    - Integrates with Amazon Quick Sight for easy data visualization.
    - Supports generating reports and data exploration with BI tools or SQL clients via JDBC or ODBC drivers
  - AWS Glue Integration:
    - Works with AWS Glue Data Catalog for a persistent metadata store.
    - Enables table creation and querying based on a central metadata store.
    - Integrated with AWS Glue's ETL and data discovery features.
  - Ease of Use:
    - Allows running interactive queries directly against data in Amazon S3.
    - Requires no data formatting or infrastructure management.
    - Quick setup: define a table and start querying using standard SQL.
  - Ideal Use Cases:
    - Suitable for running interactive ad-hoc SQL queries against Amazon S3 data.
    - Eliminates the need to manage infrastructure or clusters.
    - Perfect for quick queries, such as troubleshooting web log performance issues.
- Terms:
  - Data source – a group of databases (sometimes Catalog)
  - Database – a group of tables (sometimes schema)
  - Table – data organized as a group of rows or column
- Create table as select (CTAS)
  - query creates a new table in Athena from the results of a SELECT statement from another query. Athena stores data files created by the CTAS statement in a specified location in Amazon S3
  - CREATE TABLE AS combines a CREATE TABLE DDL statement with a SELECT DML statement and therefore technically contains both DDL and DML. However, note that for Service Quotas purposes, CTAS queries in Athena are treated as DML
- ACID Transactions
  - Powered by Apache Iceberg
  - Ensure data integrity in database transactions
  - Atomicity
    - This means that all parts of a transaction must be completed successfully.
    - If any part fails, the entire transaction is rolled back, leaving the system as if the transaction never happened.
    - Think of it as an all-or-nothing rule.
  - Consistency
    - A transaction must bring the database from one valid state to another, maintaining all predefined rules, such as data integrity constraints.
    - This ensures that the database remains accurate and reliable throughout the

transaction process.
    - Isolation
        - Transactions are isolated from each other, meaning the operations of one transaction cannot interfere with those of another.
        - This is like making sure that if multiple people are making changes at the same time, each change is processed separately without causing conflicts.
    - Durability
        - Once a transaction is successfully completed, the changes it made are permanent, even in the event of a system failure. This ensures that once you have made a change and the system confirms it, that change will not be lost.

Amazon Sage Maker

- provides machine learning (ML) capabilities for data scientists and developers to prepare, build, train, and deploy high-quality ML models efficiently.

Glue:

- Can submit ETL as scripts (python, scala, pyspark)
- You can schedule when you want it to run
- If you want to take advantage of Crawler to structure your data
- Use when your data is structured
- Data is located in S3 buckets
- Serverless
- Quick ETL

EMR:

- Can connect to EC2 --> this way we have access to whatever we are working on in EC2
- Can connect to Hadoop
- Better for more complex situations and bigger data
- Use when your data is semi or unstructured
- Can be less expensive than Glue

3rd Option Apache Airflow or Informatica

AWS Data pipeline
AWS Redshift
AWS Kinesis
AWS migration hub

Big Data Architectures

- Lambda:
    - Founded by Nathan Mars at Back Type
- Kappa:
    - Founded by Jay Keep's at Confluent

Real Time Layer:

- AWS Kinesis, AWS Firehose

- Real time data process, make your pipeline continuous, automate this
- Every time reports are generated you are getting real time data, not the batch from yesterday
- Ex: stock trading, maybe HIMCO, self-driving cars

Batch Layer:
- AWS Glue
- When you get the data in batches, whether that's once a hour/day/week
- Ex: general details about a client

Serving Layer:
- Tableau,
- Showing the results of the data
- Send to an analyst, build your own dashboard

Kappa:
- Financial instructions/ self-driving cars
- AWS Kappa

Week 2 - Python for Data Engineering

Friday, June 21, 2024 3:02 PM

6.24.24 - 2.1

Friday, June 21, 2024 3:02 PM

Have to be in notebook_env to open jupyter notebook

- Conda activate notebook_env
- Jupyter labcon

Terminal, Git, and GitHub Basics

- Basic Linux Commands
    - Cd changes the directory
        - Just type cd to go back to home
    - Cd ~ changes to the home directory
    - Cd.. Moves up one directory
    - Ls lists files in the folder
    - Ls -a shows hidden files
    - Ls -al shows roles/permissions
    - Pwd shows the current directorypwd
    - Mkdir <folder name creates a new directory with the Folder Name
    - Touch <file name> creates a new file with the file name
        - Ex: <file name> deletes a file
    - Rm -r <folder name> deletes a folder
    - Explorer . Opens current folder on Windows
    - Explorer <file name> opens a specific file one windows
    - Clear clears the screen
    - -- version checks version
    - Info shows the info
    - Info --envs shows the environments
        - "*" will appear next to env you are in
    - Which python to show what env you are working in
    - Conda create -n dev2 to create new virtual environment called dev2
    - Conda activate dev2 to activate/switch to new virtual environment
    - Conda deactivate to go back to base environment
    - Conda install pandas numpy statsmodels
        - To install multiple packages at once just add a space in between
    - To delete env
        - Conda env remove -n dev2
        - Check conda info --envs to make sure not there anymore
    - If missing a channel when creating env/installing packages
    - Type code to open VS code
    - Type jupyer lab to open, to exit control c twice or do file shut down

- Docker
    - Think of it like a virtual machine/ container
    - Simplifies installation
    - Work with Docket Hub
    - Used when downloading data bases

Python Virtual Environments

    - Isolated buckets or folders, each a python interpreter and associated libraries
    - For each project you should create a new virtual environment
    - Virtual environments create an isolated environment for Python projects.
    - You may work on different projects that have different dependencies.
    - Different projects might also use different types and versions of libraries.
    - This virtual environment ensures you have the required dependencies for future class activities.
    - All virtual environments will be under envs/

- Conda vs Venv
    - Both all you to create multiple virtual env
    - Conda
        - provides both package dependency management and environment management for Python (and supports many other languages)
        - Conda can create env with just python interpreter (conda create -n environment name)
        - You need to install packages (conda install pandas)
        - Has good dependency checker
        - Conda Install
            - This goes to the Conda Hub and takes the latest version to install it
            - Conda hub is much smaller hub and has very specific packages
            - Is basically a subset of PyPi hub
            - All the packages here are vetted and checked
    - venv
        - is a built-in Python module which provides environment management and requires no additional installation.
        - Venv create env name
        - To install libraries need to pip install pandas
        - Need pip to install things
        - Pip install
            - Goes to the PyPi hub and installs it
            - PyPi hub is HUGEE
            - Has a lot of trash and very old things but has everything including new things

- Can create a virtual env with YAML
    - Conda env create -f env.yml
    - Conda activate tscookbook
    - Can bootstrap env and share with others

- Ipykernel
    - Nb_kernel
    - Conda create -n test python=3.8 pandas ipykernel -y
    - conda install -n python_env ipykernel
    - From one jupyter instance I can have many virtual environments

Basic Data Structures in Python

- List, tuples, dictionaries, set
- Lists have to have one data type
- Array can have multiple data types (numpy)

6.25.24 - 2.2

Tuesday, June 25, 2024 7:43 AM

! conda install pandas -y

- Allows me to pass though terminal and install things in different kernel
- Error no module found, this is how to install in this environment

Shift tab to see docstring notes in jupyter

6.26.24 - 2.3

Wednesday, June 26, 2024 7:40 AM

conda install pandas -y
conda install -c conda-forge pyarrow -y

6.27.24 - 2.4

Thursday, June 27, 2024 7:50 AM

git pull OR git pull origin main (if in dev-gina)stasgit
- need to be in main branch
- need to be in tech cat folder
git stash to save work without pushing
git branch
git switch dev-gina

Git ignore .cfg

git status
git add .
git commit -m " added file"
Git
git push

gitadd

Touch .gitignore
Code .gitignore
- Inside git ignore do *.cfg

Git status to check file is not there in red
git add README.md
Git status there should be nothing there after adding
git commit -m "my first commit read.me"
Git status should now say nothing and good to push
Git push

https://git-scm.com/book/en/v2/Appendix-C:-Git-Commands-Setup-and-Config

```
git config --global core.editor "code --wait"
```

Touch helloworld.txt
Echo "my statement is hello" > helloworld.txt
Git add helloworld

Mkdir activities - to make new folder cd
Ls to see what is in folde

6.28.24 - 2.5

Friday, June 28, 2024 8:27 AM

AWS Glue ETL Transformation

Data Prep Recipe

- Create new recipe , we are just defining steps
- IAM role Data brew role
- Time transformations
    - …, extract, date and time values, extract year
    - Extract, data and time values, based on fxn, datediff (difference between 2 dates)
        - Unit is minute
- Click recipe once done with all transformations
    - Publish (publishing the definition and functions)
- Select this for data brew recipe name

https://automatetheboringstuff.com/
https://wesmckinney.com/
https://www.w3schools.com/

https://docs.python.org/3/library/datetime.html

DBMS Normalization: 1NF, 2NF, 3NF Database Example (guru99.com)

Week 3 - SQL Foundations

Monday, July 1, 2024 7:58 AM

In python pass is just a place holder (also could just do return None)

Database

- Structured collection of data that is organized and stored in a computer system

Data Warehouse

- Is a type if database that is specifically designed for analytics rather than transactional data
- Structured and filtered data used for analytics
- Include historical data

Snowflake

- Is a data ware house
- Cloud based data platform

1. **OLTP (Transactional Database):**
   - Research what OLTP is and its key characteristics.
   - Understand why organizations need OLTP.
   - Explore the pros and cons of OLTP databases.
   - including use cases and scenarios where each is most suitable.

   - Online transaction processing database
   - Transactional data, data to day transactions
   - Relational database management system
   - Fast query processing, transactions executed in real time, effectiveness in transactions per second
   - Can have many users completing transactions simultaneously, but two users cannot change the same data at the same time
   - ACID properties
   - Not used for complex queries
   - Popular OLTP Databases: MySQL, PostgreSQL, Oracle,
   - Use case:
     - reservation systems for restaurants/hotels to manage bookings, payments
   - Pros
     - Quick query processing
     - Data integrity due to ACID properties
     - Real time processing
   - Cons
     - Limited analytical capabilities - One of the major problems with OLTP databases is their analytical capabilities, as they are built mainly for fast transaction processing rather than data analysis.

- Scalability - It is not easy to scale with very huge datasets.
- Cost - OLTP databases need High-performance hardware or software, which can be pretty costly
- Doesn't hold any historical data

2. Data Warehousing:

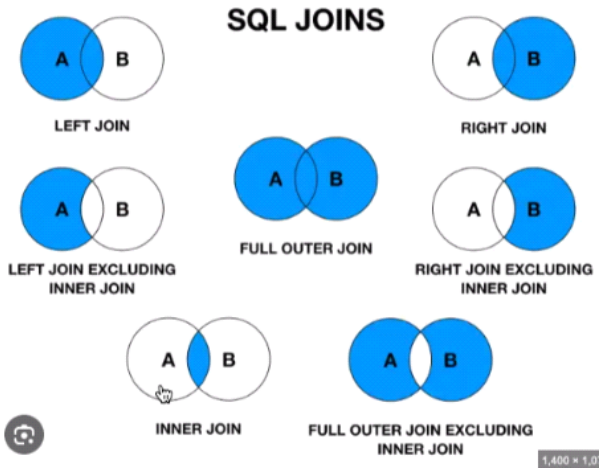
- Research what a Data Warehouse is and how it differs from OLTP.
- Explore why organizations need Data Warehouses.
- Investigate the pros and cons of Data Warehouses.
- including use cases and scenarios where each is most suitable.

- Gathers data from different sources into a central storage area
- Designed for decision making process whereas OLTP for business transaction process

- Very time consuming and costs a lot of money

JOINS
Inner and left are most common

INNER JOIN	Returns records that have matching values in both tables.
LEFT JOIN	Returns all records from the left table and the matched records from the right table.
RIGHT JOIN	Returns all records from the right table and the matched records from the left table.
CROSS JOIN	Returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables.
FULL OUTER JOIN	Places null values within the columns that do not match between the two tables, after an inner join is performed.



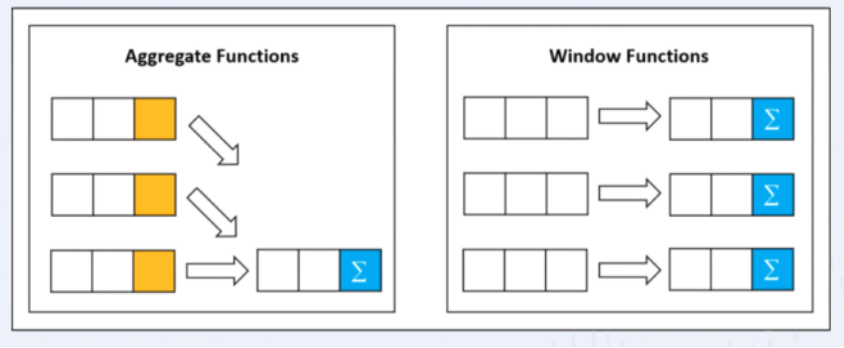
- Ex: left join if left is customer and right is orders,
    - Left would give you all customers even if they didn't make order

Week 4 - Snowflake & SQL

Monday, July 8, 2024 9:49 AM

Window Functions

- Calculates values across a set of rows and returns a value for each row
- Don't need to use a group by
- Returns multiple values, one for each row
- <window fxn name> OVER(
  Partition by
  Order by)



Conda info
Conda info --envs : to show environments
Conda activate notebook_env
Conda list nb_conda to see what packages are installed
Conda create -n dev3 python=3.11 ipykernel : to create a new environment
Which python : to check where u are

Conda install anacond::snowflake-snowpark-python
Conda install pandas -y
Just google conda snowpark python to find install code

Snowflake Architecture

- Db storage, query engine (warehouse/ engine), cloud services
- Software as a service (only responsible for application)
  - Relies on AWS/Azure/Cloud provider for infrastructure
  - Snowflake manages data storage, virtual warehouse, upgrades
- Virtual Warehouse
  - Cluster of compute resources
  - Two types
    - Standard
    - Snowpark optimized
  - Provides resources (CPU, Memory and storage)
  - Charged per second
  - Multi Cluster
    - Allows warehouse to automatically scale resources
- Loading Data
  - Bulk Loading

        - Most common
        - Uses warehouses
        - Loading from stages
        - COPY command
        - Transformations possible
        - Not something you do everyday
        - COPY INTO, Snow pipe, and third party ETL tools
        - Less frequent, bigger amount
    - Continuous Loading
        - Load small amounts of data
        - More frequent but less amount
        - Automatically once they are added to stages
        - Latest results for analysis
        - Snow pipe serverless
- Stages
    - Stage object contains location of data files where data can be loaded from
    - Db object that is created in schema
    - Internal
        - Stored files in snowflake
    - External

Git Hub Notes

- When creating repo, always have read me

Cd Desktop/techcatalyst-de-GM/
Code READ.me to open the read me in vs code
Git status to see what has changed

- Red meads haven't been added
- Green means it has been staged
- Git restore --staged read.me

Code . Open all files in that folder

Code note.txt to create new file with new name note.txt

Git add read.me

Week 5 - Data Modeling Concepts & PySpark Applications

Monday, July 15, 2024 7:59 AM

https://colab.research.google.com/drive/1yUPJmb3D0aq2Bt-8ARCqazLses5lssz5?usp=sharing

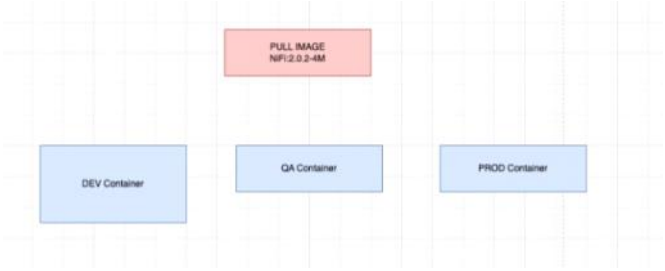
Apache Nifi

- Visual data flow developed by NSA
- Open source etl tool
- Processors to fetch objects
- Flows

Amazon Keys

Access key ID	Secret access key

Docker

- Make sure reproducible, don't run into random issues/install things
- Create a docker container that has the application you want on it
- Can run on any computer independently
- Can replicate container from dev env and use it in Prod
- Docker is more "lightweight" comparted to a VM
- Pre-packaged applications
- Image: something you can download, can create containers from the image
    - We downloaded NiFi image and then created a Nifi container
    - Can duplicate this container
    - Docker pull …
    - 

- Base Images
    - Lego pieces
    - 1st Lego is alpine Linux, 2nd is python, 3rd is jupyter…
    - Pre-set environment and we can pull this image and make containers on it
- Someone can download the same docker image and have the same environment and that way they can run my same code but on their computer

Sudo docker image ls -- to see what images you have
Sudo docker container ps -- to see what containers
Sudo docker container ps -a to see hidden ones/stopped/exited
Sudo docker container rm nifi -- to remove docker

sudo docker run --name nifi apache/nifi:2.0.0-M4 -- to run this image
sudo docker run -d --name dev-nifi apache/nifi:2.0.0-M4 -- the -d cleans us screen

sudo docker container stop dev-nifi -- to stop the container

sudo docker container start dev-nifi -- to start the container

Sudo docker run --name dev-nifi -p 8443:8443 -d -- to connect to a specific port

- Create new folder "files" in Downloads and copy path : this will point to a specific folder
- Pwd in terminal to get path

sudo docker run --name dev-nifi -p 8443:8443 -d -v /home/gmastrorilli/Downloads/files:/opt/nifi/input-files apache/nifi:2.0.0-M4

To get nifi username and password: sudo docker logs dev-nifi | grep Generated

Sudo docker exec -it dev-nifi /bin/bash -- to get inside container

- Go to cd /opt/nifi/input-files/
- Exit to get out
- Ls to see if your files are there

Apache Spark

- AWS Glue runs on Spark engine
- Spark runs on Hadoop but doesn't have o
- Uses scripts, has a rich ecosystem and is very scalable (scala, python, java r and sql)
- RDD Resilient Distributed Dataset
- Spark Streaming, Spark SQL, MLLib, GraphX, Spark Core
- Uses in memey computation instead of disk based solution (faster)
- Lazy evaluation, delays the aval of an expression until its value is needed
- Better for huge amounts of data, takes longer than pandas for small df

Spark DataFrame	Pandas DataFrame
Spark DataFrame supports parallelization.	Pandas DataFrame does not support parallelization.
Spark DataFrame has Multiple Nodes.	Pandas DataFrame has a Single Node.
It follows Lazy Execution which means that a task is not executed until an action is performed.	It follows Eager Execution, which means task is executed immediately.
Spark DataFrame is Immutable.	Pandas DataFrame is Mutable.
Complex operations are difficult to perform as compared to Pandas DataFrame.	Complex operations are easier to perform as compared to Spark DataFrame.
Spark DataFrame is distributed and hence processing in the Spark DataFrame is faster for a large amount of data.	Pandas DataFrame is not distributed and hence processing in the Pandas DataFrame will be slower for a large amount of data.
sparkDataFrame.count() returns the number of rows.	pandasDataFrame.count() returns the number of non NA/null observations for each column.
Spark DataFrames are excellent for building a scalable application.	Pandas DataFrames can't be used to build a scalable application.
Spark DataFrame assures fault tolerance.	Pandas DataFrame does not assure fault tolerance. We need to implement our own framework to assure it.

https://www.markdownguide.org/basic-syntax/

https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax

SQL Reference - Spark 3.5.1 Documentation (apache.org)

Data Modeling

Fact Table/Star Schema

- One central table that hold facts/transactions
- Things we can aggregate/measure (numbers)
- Dimension tables are used to describe the facts in the central table
- Only IDs in fact table
- Dim tables can only have unique records

Apache Hue (Hadoop User Experience)

- To choose this when setting up your emr cluster is Opting to have web UI for interacting w hadoop
- Web based interactive query editor for analyzing data with Hadoop
- Integrated with hive and impala --> provides user friendly interface for interacting with the hadoop cluster
- Web user interface designed to simplify hadoop and provides many services
- Similar to databricks, aws management console for emr, aws glue console

Apache Tez

- To choose this when setting up your emr cluster is Enabling optimized engine for running queries
- Open source framework for big data processing based on MapReduce tech
- Designed to optimize complex data processing/ data pipelines by enabling DAGs of tasks
- Alternatives are Flink

Apache Hive

- To choose this when setting up your emr cluster you are Adding data warehouse structure that allows you to perform sql queries on the data
- SQL based system that runs on top of hadoop as a data warehouse
- Facilitate analysis of large datasets in hadoops HDFS and other systems like S3
- Sql interface using Hive SQL

Week 6

Monday, July 22, 2024 8:01 AM

Spark Streaming
- Input data stream
- Spark stream
- Batches of input data
- Spark engine
- Batches of processed data
- Can take data from Kafka, Flume, HDFS/S3 Kinesis and output into GDFS, Databases, Dashboards

CDC change data capture

Amazon DMS handles CDC for you