The Exterminators - SVCE

BUILDING AN EFFICIENT ETL PIPELINE

By:

Janani	III/CSE
Allen	III/CSE
Poonam	III/IT
Akaash	III/CSE

/TABLE OF CONTENTS

/01 AIM

/02 WHY THIS PROBLEM STATEMENT

/03 SOLUTION - PHASE 2 OVER PHASE 1

/04 TECHNICAL EXPLANATION

/05 OPTIMIZATION & INNOVATION

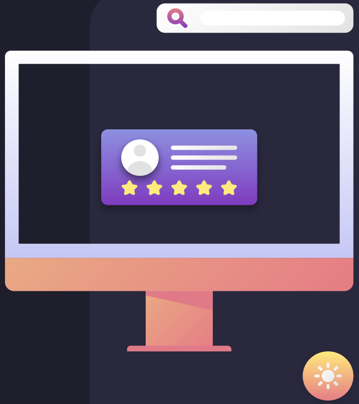
/06 VIABILITY

/07 MARKET COMPARISON & ESTIMATION

/08 FUTURE PLANS

/AIM

To build an efficient **ETL Pipeline** which Extracts, Transforms and Loads data from a destination to another. It must consist of a web dashboard which performs **visualization** on data and helps **reporting errors** occurring in the pipeline.



/WHY WE CHOSE THIS PS

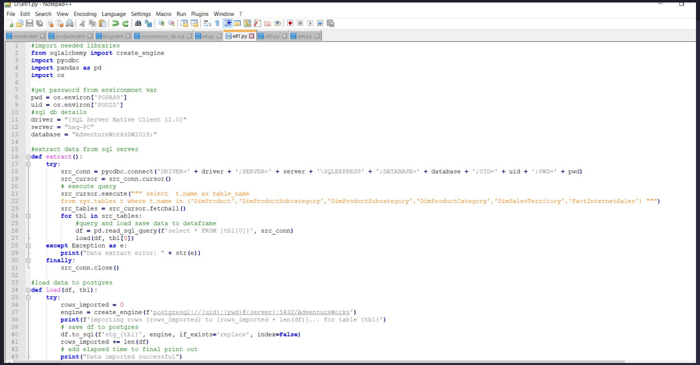
- Interesting problem statement
- ETL, Cloud, Data Warehousing is a new concept to us - we wanted to take it up as a challenge.
- There are not many resources like other problem statements- so we learnt topics from scratch by doing a lot of research.

INITIAL SOLUTION - what we tried performing in the beginning weeks

- Extracting data from SQL Server and loading data into Postgres
- Automating the pipeline using Airflow (using TaskFlow API) and using DAG
- Performing Incremental Data loading using Destination Change Comparison technique
- Alteryx- data visualization tool that works excellently with ETL

INITIAL SOLUTION 1

1. Making a basic ETL pipeline using Python



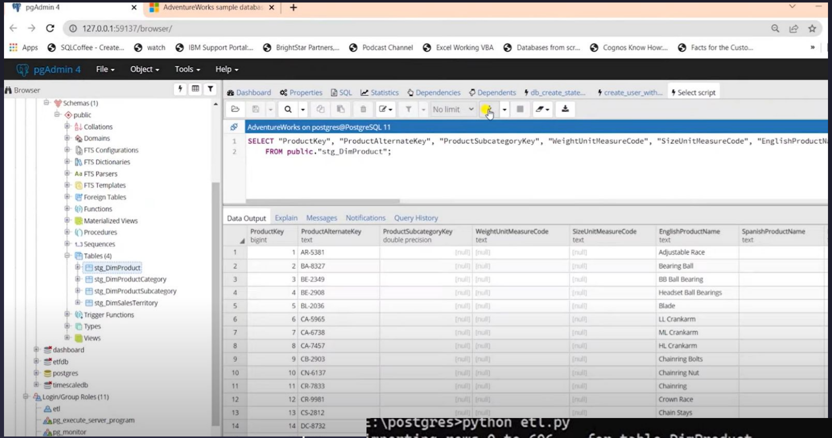
```python
#import needed libraries
from sqlalchemy import create_engine
import pyodbc
import pandas as pd
import os

#get password from environmnet var
pwd = os.environ['PGPASS']
uid = os.environ['PGUID']
#sql db details
driver = "{SQL Server Native Client 11.0}"
server = "haq-PC"
database = "AdventureWorksDW2019;"

#extract data from sql server
def extract():
    try:
        src_conn = pyodbc.connect('DRIVER=' + driver + ';SERVER=' + server + '\SQLEXPRESS' + ';DATABASE=' + database + ';UID=' + uid + ';PWD=' + pwd)
        src_cursor = src_conn.cursor()
        # execute query
        src_cursor.execute(""" select  t.name as table_name
        from sys.tables t where t.name in ('DimProduct','DimProductSubcategory','DimProductSubcategory','DimProductCategory','DimSalesTerritory','FactInternetSales') """)
        src_tables = src_cursor.fetchall()
        for tbl in src_tables:
            #query and load save data to dataframe
            df = pd.read_sql_query(f'select * FROM {tbl[0]}', src_conn)
            load(df, tbl[0])
    except Exception as e:
        print("Data extract error: " + str(e))
    finally:
        src_conn.close()

#load data to postgres
def load(df, tbl):
    try:
        rows_imported = 0
        engine = create_engine(f'postgresql://{uid}:{pwd}@{server}:5432/AdventureWorks')
        print(f'importing rows {rows_imported} to {rows_imported + len(df)}... for table {tbl}')
        # save df to postgres
        df.to_sql(f'stg_{tbl}', engine, if_exists='replace', index=False)
        rows_imported += len(df)
        # add elapsed time to final print out
        print("Data imported successful")
```

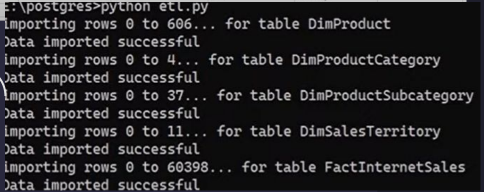
2. Results were produced in Postgre from SSMS ⇒ETL successful



AdventureWorks on postgres@PostgreSQL 11

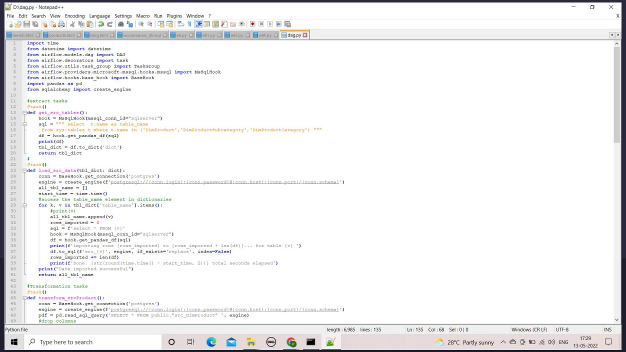
```sql
SELECT "ProductKey", "ProductAlternateKey", "ProductSubcategoryKey", "WeightUnitMeasureCode", "SizeUnitMeasureCode", "EnglishProductN
    FROM public."stg_DimProduct";
```

	ProductKey bigint	ProductAlternateKey text	ProductSubcategoryKey double precision	WeightUnitMeasureCode text	SizeUnitMeasureCode text	EnglishProductName text	SpanishProductName text
1	1	AR-5381	[null]	[null]	[null]	Adjustable Race	
2	2	BA-8327	[null]	[null]	[null]	Bearing Ball	
3	3	BE-2349	[null]	[null]	[null]	BB Ball Bearing	
4	4	BE-2908	[null]	[null]	[null]	Headset Ball Bearings	
5	5	BL-2036	[null]	[null]	[null]	Blade	
6	6	CA-5965	[null]	[null]	[null]	LL Crankarm	
7	7	CA-6738	[null]	[null]	[null]	ML Crankarm	
8	8	CA-7457	[null]	[null]	[null]	HL Crankarm	
9	9	CB-2903	[null]	[null]	[null]	Chainring Bolts	
10	10	CN-6137	[null]	[null]	[null]	Chainring Nut	
11	11	CR-7833	[null]	[null]	[null]	Chainring	
12	12	CR-9981	[null]	[null]	[null]	Crown Race	
13	13	CS-2812	[null]	[null]	[null]	Chain Stays	
14	14	DC-8732					



```
:\postgres>python etl.py
importing rows 0 to 606... for table DimProduct
Data imported successful
importing rows 0 to 4... for table DimProductCategory
Data imported successful
importing rows 0 to 37... for table DimProductSubcategory
Data imported successful
importing rows 0 to 11... for table DimSalesTerritory
Data imported successful
importing rows 0 to 60398... for table FactInternetSales
Data imported successful
```

3. DAG code for automation purposes



```python
import time
from datetime import datetime
from airflow.models.dag import DAG
from airflow.decorators import task
from airflow.utils.task_group import TaskGroup
from airflow.providers.microsoft.mssql.hooks.mssql import MsSqlHook
from airflow.hooks.base_hook import BaseHook
import pandas as pd
from sqlalchemy import create_engine

#extract tasks
@task()
def get_src_tables():
    hook = MsSqlHook(mssql_conn_id="sqlserver")
    sql = """ select  t.name as table_name
     from sys.tables t where t.name in ('DimProduct','DimProductSubcategory','DimProductCategory') """
    df = hook.get_pandas_df(sql)
    print(df)
    tbl_dict = df.to_dict('dict')
    return tbl_dict
#
@task()
def load_src_data(tbl_dict: dict):
    conn = BaseHook.get_connection('postgres')
    engine = create_engine(f'postgresql://{conn.login}:{conn.password}@{conn.host}:{conn.port}/{conn.schema}')
    all_tbl_name = []
    start_time = time.time()
    #access the table_name element in dictionaries
    for k, v in tbl_dict['table_name'].items():
        #print(v)
        all_tbl_name.append(v)
        rows_imported = 0
        sql = f'select * FROM {v}'
        hook = MsSqlHook(mssql_conn_id="sqlserver")
        df = hook.get_pandas_df(sql)
        print(f'importing rows {rows_imported} to {rows_imported + len(df)}... for table {v} ')
        df.to_sql(f'src_{v}', engine, if_exists='replace', index=False)
        rows_imported += len(df)
        print(f'Done. {str(round(time.time() - start_time, 2))} total seconds elapsed')
    print("Data imported successful")
    return all_tbl_name

#Transformation tasks
@task()
def transform_srcProduct():
    conn = BaseHook.get_connection('postgres')
    engine = create_engine(f'postgresql://{conn.login}:{conn.password}@{conn.host}:{conn.port}/{conn.schema}')
    pdf = pd.read_sql_query('SELECT * FROM public."src_DimProduct" ', engine)
    #drop columns
```

INITIAL SOLUTION 1 PROBLEMS

Postgres

- Loading takes a long time for big data
- Multiple data access becomes difficult
- Low reading speed

Airflow

- Airflow doesn’t have versioning- if you delete a task from DAG code and redeploy it, metadata related to task will be lost

INITIAL SOLUTION 2

- Writing ETL pipeline in Python.
- Apply functional programming in Data Engineering.
- Use a proper object oriented code design and meta file for job control.
- Implement a pipeline in Python extracting data Using docker image with application code on Docker Hub, using Kubernetes and Argo Workflows

PROBLEMS WRT DOCKER

- Is best for building microservices for applications
- Not good if software runs in different environments or are from multiple sources
- Lesser speed
- Data storage is complicated
- Graphical applications don’t work well
- In Docker files are created inside a container. It is difficult to retrieve out of the container for different kind of processes

SO WE DIDN’T IMPLEMENT THIS

<OUR SOLUTION?>

During our last few weeks?

> Using Cloud of course! <

/WHY CLOUD?

Simplified data process for users	Allows organizations to combine data from multiple sources	Easy optimization
Handles huge volumes of data and data migration	Easy for visualization and debugging and reduce MTTR	Easy to compute data
No need of much security patches	Maintenance	Easy to analyze unstructured data

TOOLS WE ARE USING

DBEAVER

To create and manage databases across a wide range of database management systems.

REDSHIFT

Large scale data storage and analysis, and is frequently used to perform large database migrations.

DATA PIPELINE

helps you reliably process and move data between different AWS compute and storage services, as well as on-premises data sources, at specified intervals.

GLUE

Uses ETL jobs to extract data from a combination of other Amazon Web Services and incorporates it into data lakes and data warehouses.

TOOLS WE ARE USING

CLOUDWATCH

An AWS monitoring service you can use to monitor your applications, services, and resources.

QUICKSIGHT

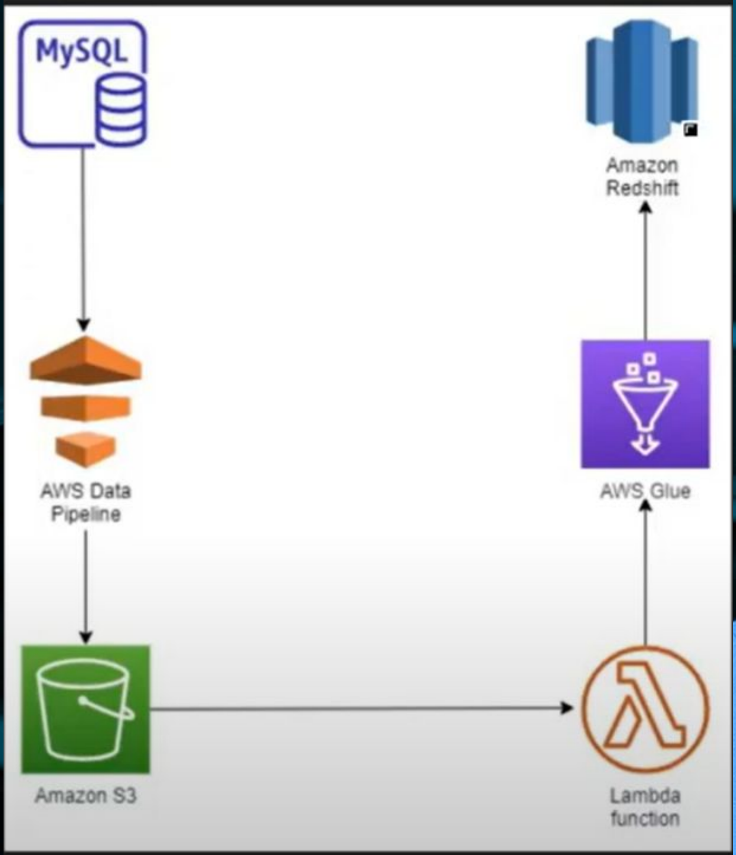
Use to deliver easy-to-understand insights to the people who you work with, wherever they are. Amazon QuickSight connects to your data in the cloud and combines data from many different sources.

PYSPARK

A standard ETL tool like PySpark, supports all basic data transformation features like sorting, mapping, joins, operations, etc.

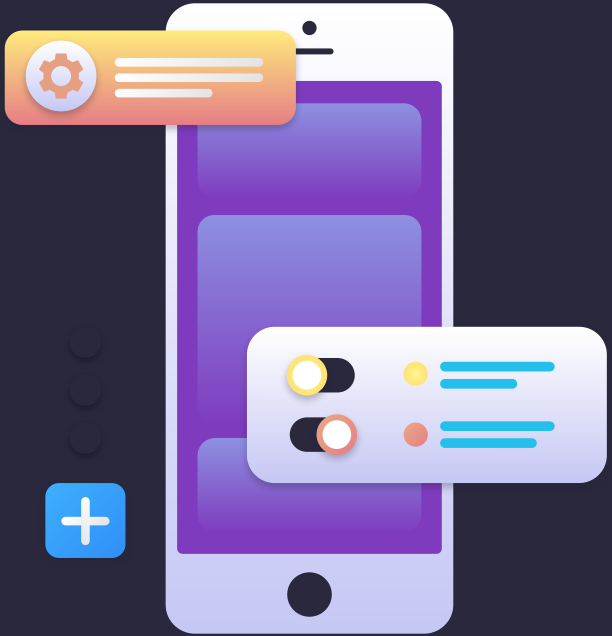
ATHENA

Performs interactive queries in the web-based cloud storage service, Amazon Simple Storage Service (S3). Athena is used with large-scale data sets. Amazon S3 is designed for online backup and archiving of data


MySQL
Amazon Redshift
AWS Data Pipeline
AWS Glue
Amazon S3
Lambda function

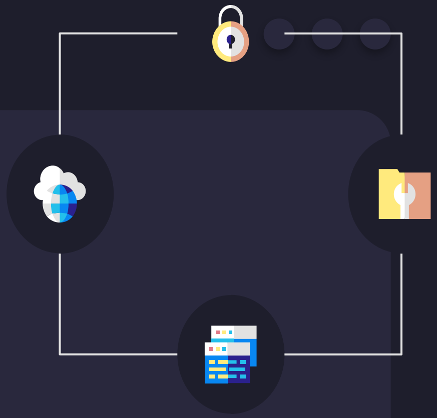
/TECHNICAL EXPLANATION

OUR SOLUTION EXPLANATION

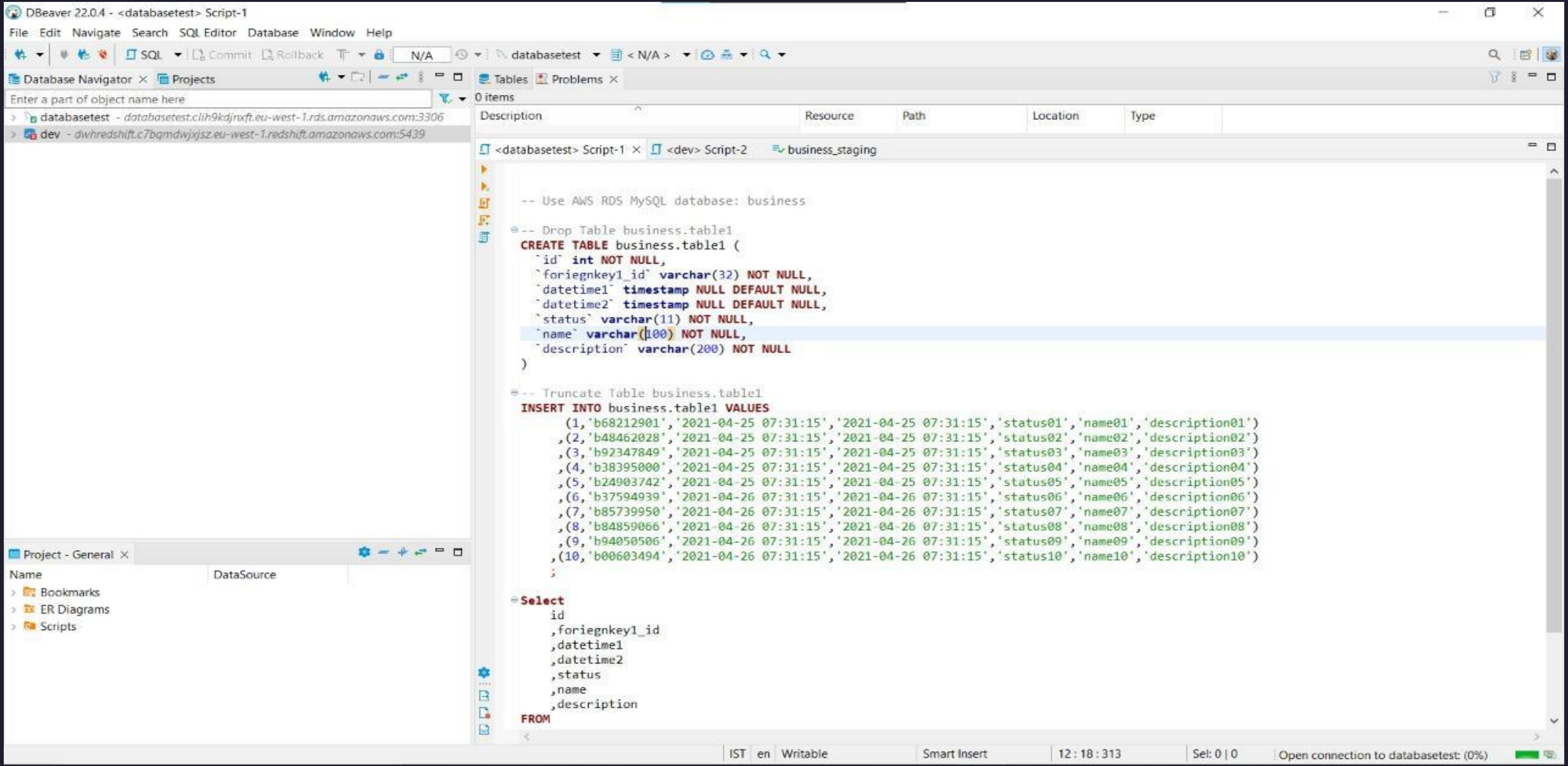


TECHNICAL EXPLANATION

- We first made a database connection between DBeaver and RDS console. DBeaver will have the entire database.
- Then we set up Redshift cluster by creating roles, editing IAM roles and VPC and configured it as per the requirement to build the ETL
- Created a S3 bucket which has an initial load and incremental load
- In Data Pipeline we configured DataNodes, table structure and created an EC2 instance



DBEAVER


DBeaver 22.0.4 - <databasetest> Script-1
File Edit Navigate Search SQL Editor Database Window Help
SQL Commit Rollback N/A databasetest < N/A >
Database Navigator Projects
Enter a part of object name here
databasetest - databasetest.clih9kdjnxft.eu-west-1.rds.amazonaws.com:3306
dev - dwhredshift.c7bqmdwjxjsz.eu-west-1.redshift.amazonaws.com:5439
Tables Problems
0 items
Description Resource Path Location Type
<databasetest> Script-1 <dev> Script-2 business_staging
Project - General
Name DataSource
Bookmarks
ER Diagrams
Scripts
IST en Writable Smart Insert 12 : 18 : 313 Sel: 0 | 0 Open connection to databasetest: (0%)

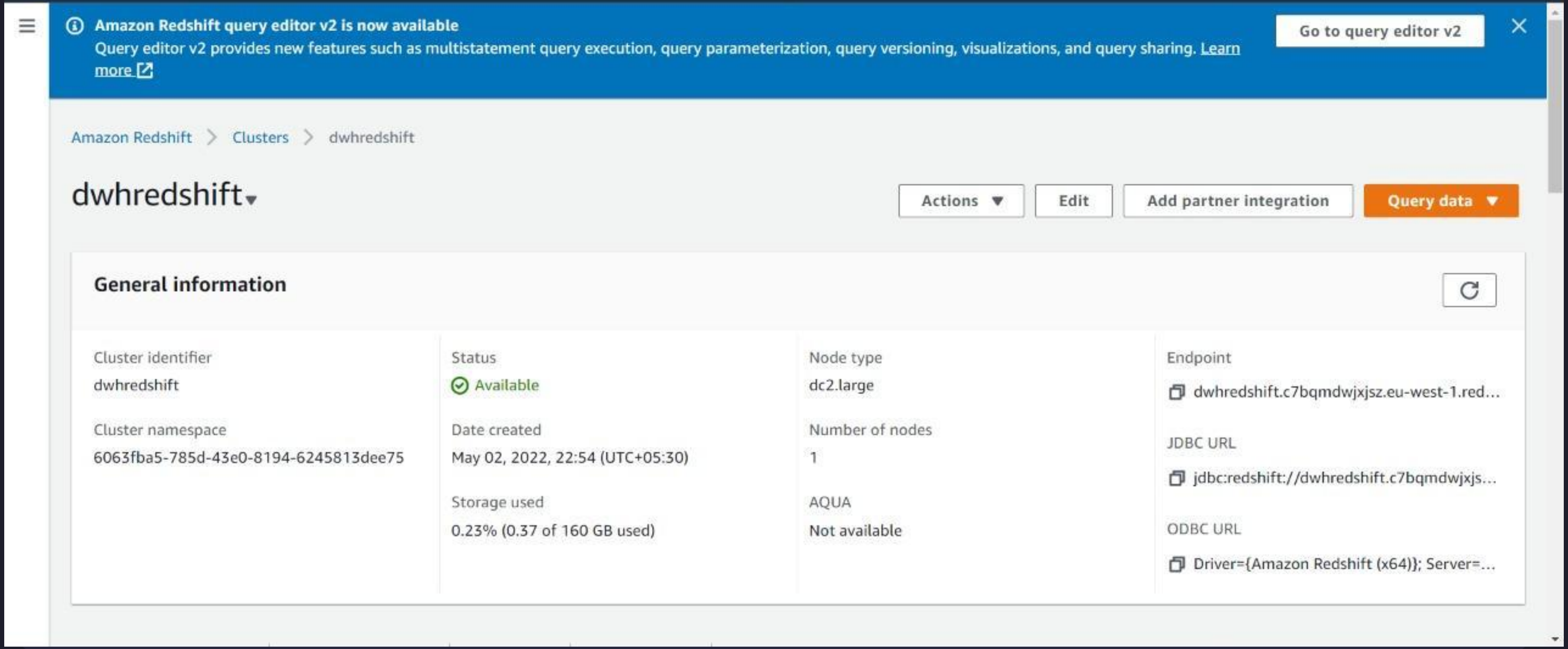
```sql
-- Use AWS RDS MySQL database: business

-- Drop Table business.table1
CREATE TABLE business.table1 (
  `id` int NOT NULL,
  `foriegnkey1_id` varchar(32) NOT NULL,
  `datetime1` timestamp NULL DEFAULT NULL,
  `datetime2` timestamp NULL DEFAULT NULL,
  `status` varchar(11) NOT NULL,
  `name` varchar(100) NOT NULL,
  `description` varchar(200) NOT NULL
)

-- Truncate Table business.table1
INSERT INTO business.table1 VALUES
      (1,'b68212901','2021-04-25 07:31:15','2021-04-25 07:31:15','status01','name01','description01')
     ,(2,'b48462028','2021-04-25 07:31:15','2021-04-25 07:31:15','status02','name02','description02')
     ,(3,'b92347849','2021-04-25 07:31:15','2021-04-25 07:31:15','status03','name03','description03')
     ,(4,'b38395000','2021-04-25 07:31:15','2021-04-25 07:31:15','status04','name04','description04')
     ,(5,'b24903742','2021-04-25 07:31:15','2021-04-25 07:31:15','status05','name05','description05')
     ,(6,'b37594939','2021-04-26 07:31:15','2021-04-26 07:31:15','status06','name06','description06')
     ,(7,'b85739950','2021-04-26 07:31:15','2021-04-26 07:31:15','status07','name07','description07')
     ,(8,'b84859066','2021-04-26 07:31:15','2021-04-26 07:31:15','status08','name08','description08')
     ,(9,'b94050506','2021-04-26 07:31:15','2021-04-26 07:31:15','status09','name09','description09')
    ,(10,'b00603494','2021-04-26 07:31:15','2021-04-26 07:31:15','status10','name10','description10')
    ;

Select
    id
    ,foriegnkey1_id
    ,datetime1
    ,datetime2
    ,status
    ,name
    ,description
FROM
```

REDSHIFT CLUSTER



Amazon Redshift query editor v2 is now available
Query editor v2 provides new features such as multistatement query execution, query parameterization, query versioning, visualizations, and query sharing. Learn more

dwhredshift

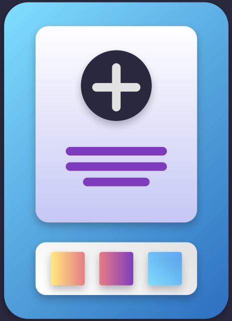
General information

Cluster identifier	Status	Node type	Endpoint
dwhredshift	Available	dc2.large	dwhredshift.c7bqmdwjxjsz.eu-west-1.red...
Cluster namespace	Date created	Number of nodes	JDBC URL
6063fba5-785d-43e0-8194-6245813dee75	May 02, 2022, 22:54 (UTC+05:30)	1	jdbc:redshift://dwhredshift.c7bqmdwjxjs...
	Storage used	AQUA	ODBC URL
	0.23% (0.37 of 160 GB used)	Not available	Driver={Amazon Redshift (x64)}; Server=...

/COMPARISON

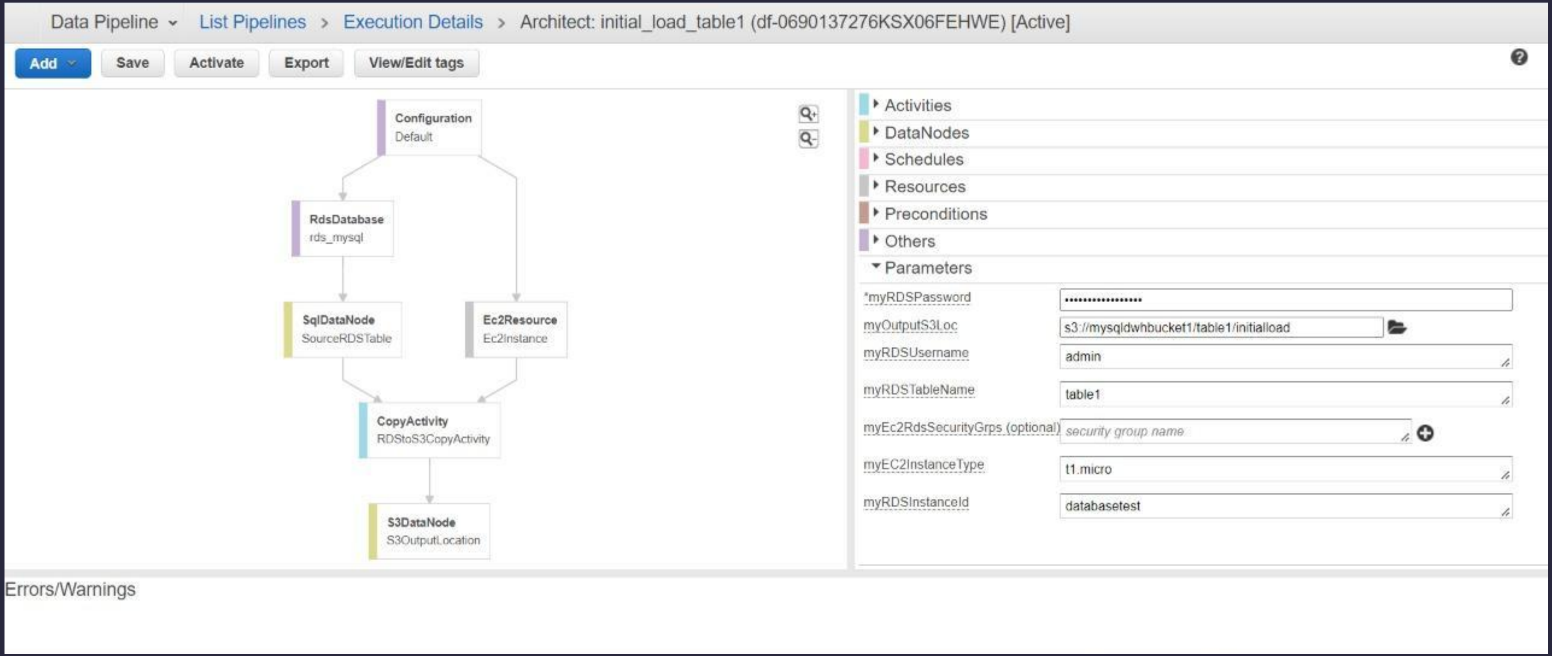
	/BIGQUERY	/REDSHIFT
Service	Serverless/Managed	Not Managed
What are they	fully managed cloud-based data warehouse which is designed for handling large-scale data set storage.	fully managed and serverless data warehouse.
Regions Available	Americas – 4 Europe – 3 Asia Pacific – 7	Americas – 8 Europe – 5 Asia Pacific – 9
Storage format	Columnar & uncompressed storage	Columnar & compressed storage
Billing	Not Predictable	Predictable

EXPLANATION



- AWS Redshift connection was established with DBeaver
- Initial data load was performed into Redshift for data load for a base object
- Incremental data load was performed into Redshift for subsequent load process requirements
- Incremental Data Pipeline from MySQL to S3 was performed by altering DataNodes and EC2 Resource
- AWS Glue Job from S3 was then configured with Redshift

DATA PIPELINE



Data Pipeline > List Pipelines > Execution Details > Architect: initial_load_table1 (df-0690137276KSX06FEHWE) [Active]

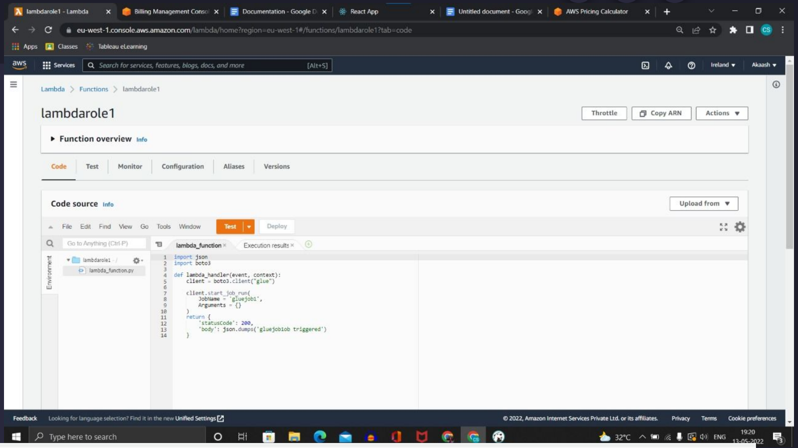
*myRDSPassword	••••••••••••••••••
myOutputS3Loc	s3://mysqldwhbucket1/table1/initialload
myRDSUsername	admin
myRDSTableName	table1
myEc2RdsSecurityGrps (optional)	security group name
myEC2InstanceType	t1.micro
myRDSInstanceId	databasetest

EXPLANATION

- Lambda Function for AWS Glue Job from S3 was configured to AWS Redshift
- Redshift Snapshot and Restore process performed
- Our ETL was ready

- We then used various tools to visualize and debug our ETL

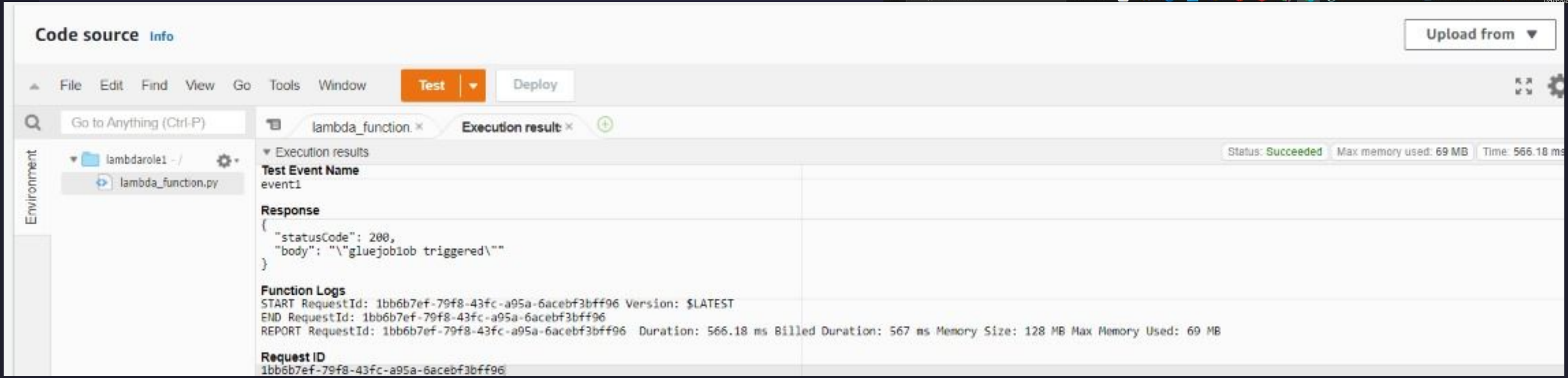
LAMBDA FUNCTION in Glue



```python
import json
import boto3

def lambda_handler(event, context):
    client = boto3.client("glue")

    client.start_job_run(
        JobName = 'gluejob1',
        Arguments = {}
    )
    return {
        'statusCode': 200,
        'body': json.dumps('gluejob1ob triggered')
    }
```



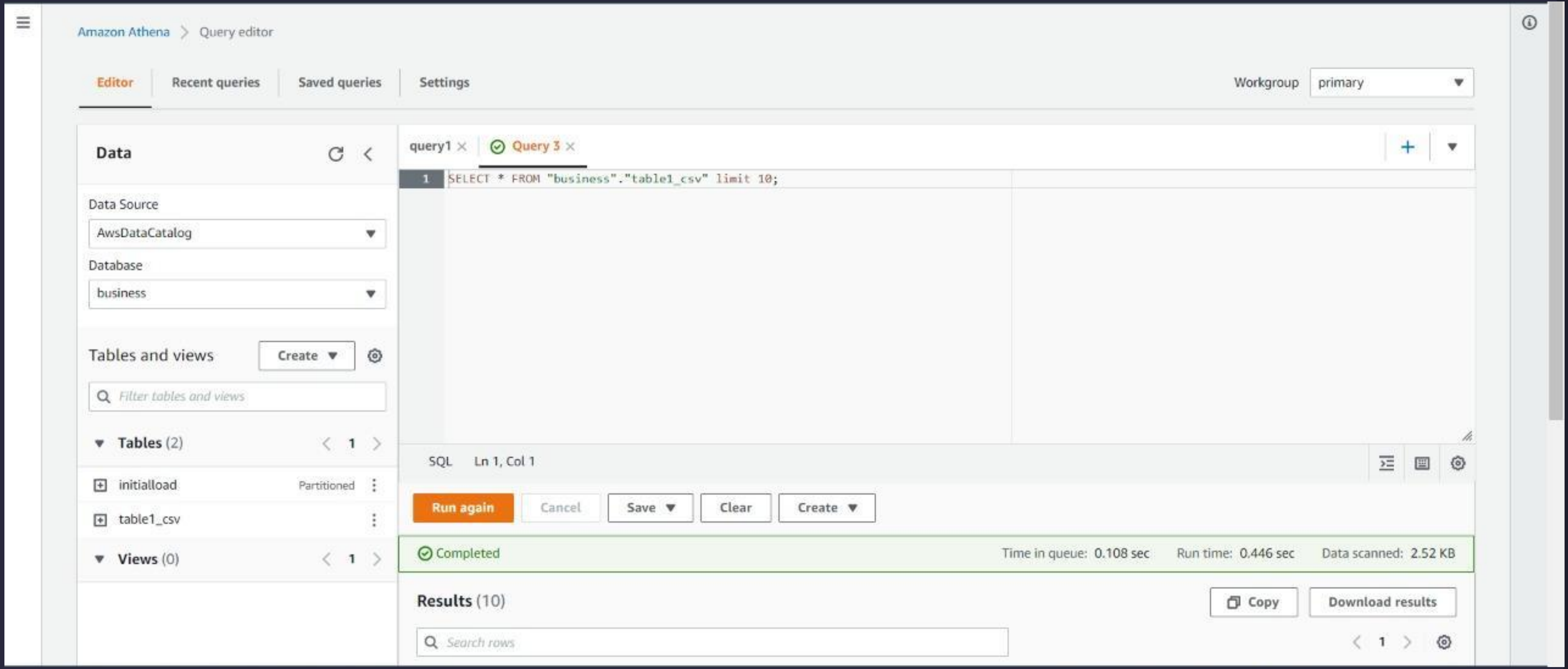
```
Execution results                                  Status: Succeeded   Max memory used: 69 MB   Time: 566.18 ms
Test Event Name
event1

Response
{
  "statusCode": 200,
  "body": "\"gluejob1ob triggered\""
}

Function Logs
START RequestId: 1bb6b7ef-79f8-43fc-a95a-6acebf3bff96 Version: $LATEST
END RequestId: 1bb6b7ef-79f8-43fc-a95a-6acebf3bff96
REPORT RequestId: 1bb6b7ef-79f8-43fc-a95a-6acebf3bff96  Duration: 566.18 ms Billed Duration: 567 ms Memory Size: 128 MB Max Memory Used: 69 MB

Request ID
1bb6b7ef-79f8-43fc-a95a-6acebf3bff96
```

ATHENA



Amazon Athena > Query editor

Editor | Recent queries | Saved queries | Settings

Workgroup: primary

Data

Data Source: AwsDataCatalog

Database: business

Tables and views

Tables (2)
- initialload — Partitioned
- table1_csv

Views (0)

query1 | Query 3

```
SELECT * FROM "business"."table1_csv" limit 10;
```

SQL Ln 1, Col 1

Run again | Cancel | Save | Clear | Create

Completed   Time in queue: 0.108 sec   Run time: 0.446 sec   Data scanned: 2.52 KB

Results (10)

Copy | Download results

/QUICKSIGHT

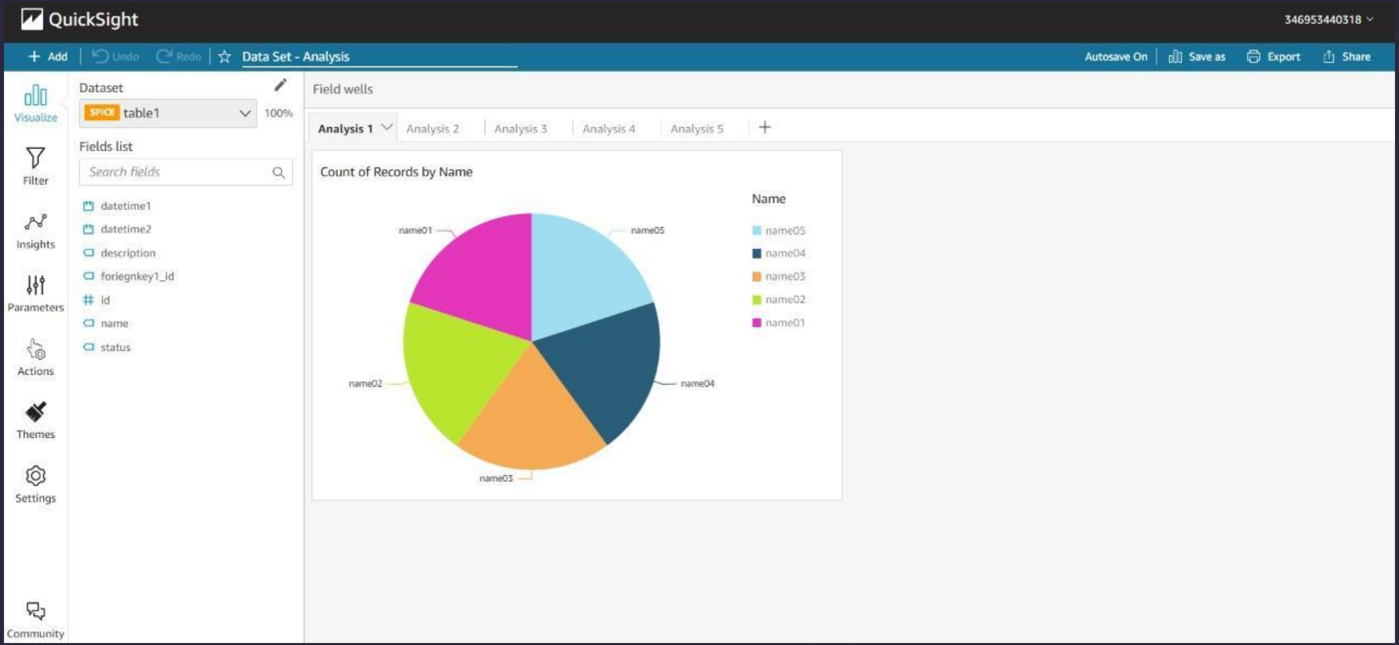
/VISUALIZATION

- Interactive dashboards
- Combine a variety of data into a single analysis.and publish dashboards
- Less expensive than Tableau
- Any data can be used

/SPICE

- Super-fast parallel in-memory calculation engine
- It’s faster and easier to retrieve than having to query the original data source each time
- Rapidly perform advanced calculations and serve the data

QUICKSIGHT


QuickSight
346953440318
Add
Undo
Redo
Data Set - Analysis
Autosave On
Save as
Export
Share
Visualize
Filter
Insights
Parameters
Actions
Themes
Settings
Community
Dataset
SPICE table1
100%
Fields list
Search fields
datetime1
datetime2
description
foriegnkey1_id
id
name
status
Field wells
Analysis 1
Analysis 2
Analysis 3
Analysis 4
Analysis 5
Count of Records by Name
Name
name05
name04
name03
name02
name01

/CLOUDWATCH

/LOGS

- Collects compute performance metrics such as CPU, memory, and network as performance events

/METRICS

- CloudWatch Metric Streams enables you to create continuous, near-real-time streams of metrics to a destination of your choice

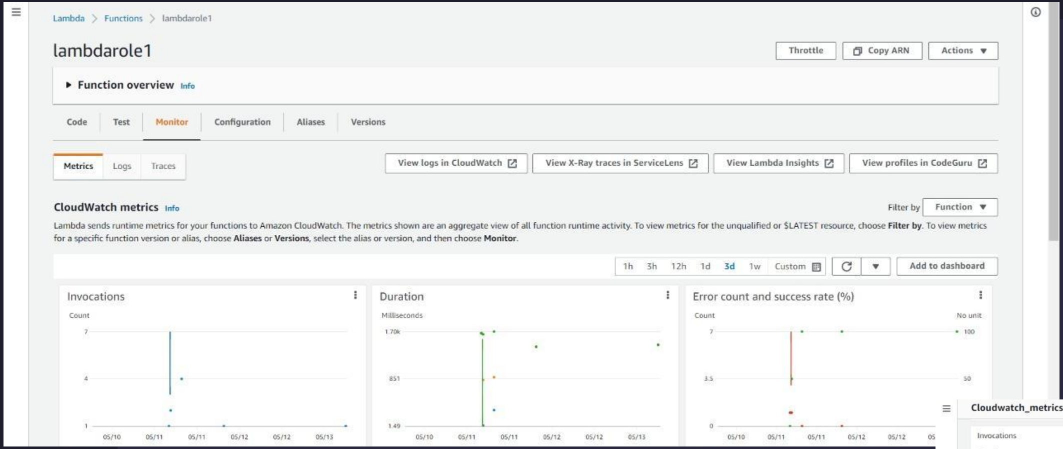
/MONITOR

- Create reusable graphs and visualize our cloud resources and applications in a unified view

/INSIGHTS

- Provides automated setup of observability for enterprise applications to get visibility into their health

CLOUDWATCH

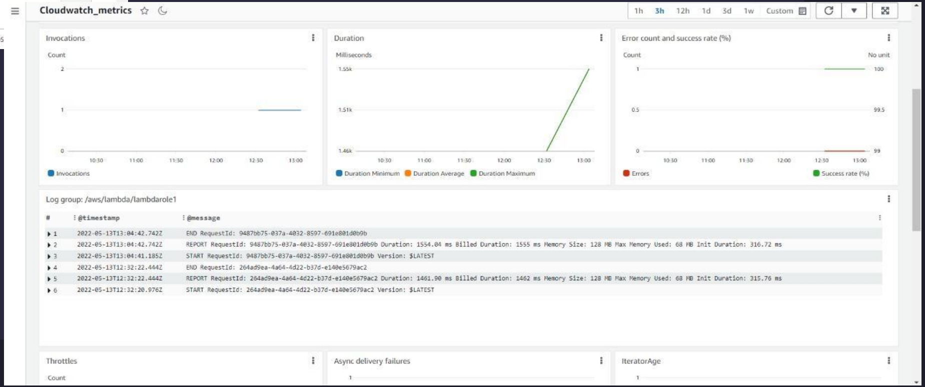


CloudWatch metrics Info

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select the alias or version, and then choose **Monitor**.

Metrics

Logs

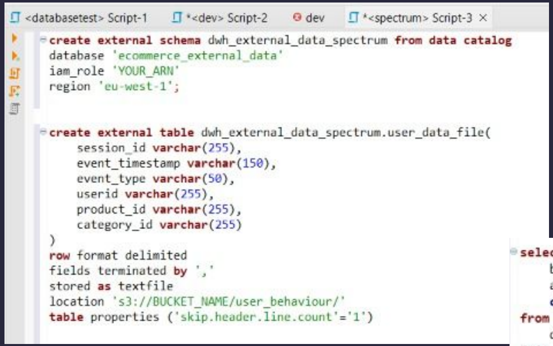


Log group: /aws/lambda/lambdarole1

```
#    @timestamp                 @message
▸ 1  2022-05-13T13:04:42.742Z   END RequestId: 9487bb75-037a-4032-8597-691e801d0b9b
▸ 2  2022-05-13T13:04:42.742Z   REPORT RequestId: 9487bb75-037a-4032-8597-691e801d0b9b Duration: 1554.04 ms Billed Duration: 1555 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 316.72 ms
▸ 3  2022-05-13T13:04:41.185Z   START RequestId: 9487bb75-037a-4032-8597-691e801d0b9b Version: $LATEST
▸ 4  2022-05-13T12:32:22.444Z   END RequestId: 264ad9ea-4a64-4d22-b37d-e140e5679ac2
▸ 5  2022-05-13T12:32:22.444Z   REPORT RequestId: 264ad9ea-4a64-4d22-b37d-e140e5679ac2 Duration: 1461.90 ms Billed Duration: 1462 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 315.76 ms
▸ 6  2022-05-13T12:32:20.976Z   START RequestId: 264ad9ea-4a64-4d22-b37d-e140e5679ac2 Version: $LATEST
```

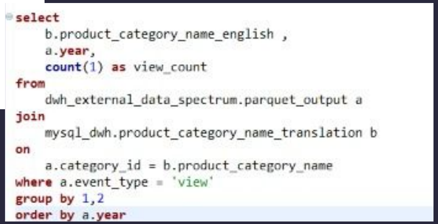
REDSHIFT SPECTRUM

Users can query data without having to load it.Multiple users can access same S3 bucket at same time.



```
create external schema dwh_external_data_spectrum from data catalog
database 'ecommerce_external_data'
iam_role 'YOUR_ARN'
region 'eu-west-1';


create external table dwh_external_data_spectrum.user_data_file(
    session_id varchar(255),
    event_timestamp varchar(150),
    event_type varchar(50),
    userid varchar(255),
    product_id varchar(255),
    category_id varchar(255)
)
row format delimited
fields terminated by ','
stored as textfile
location 's3://BUCKET_NAME/user_behaviour/'
table properties ('skip.header.line.count'='1')
```



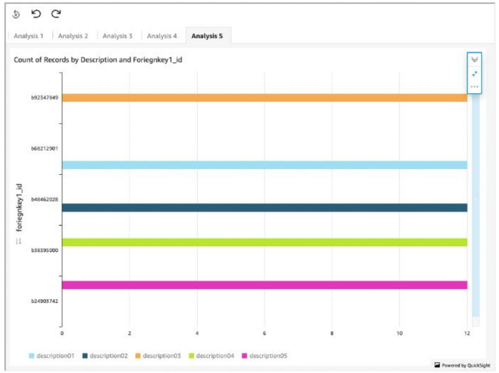
```
select
    b.product_category_name_english ,
    a.year,
    count(1) as view_count
from
    dwh_external_data_spectrum.parquet_output a
join
    mysql_dwh.product_category_name_translation b
on
    a.category_id = b.product_category_name
where a.event_type = 'view'
group by 1,2
order by a.year
```

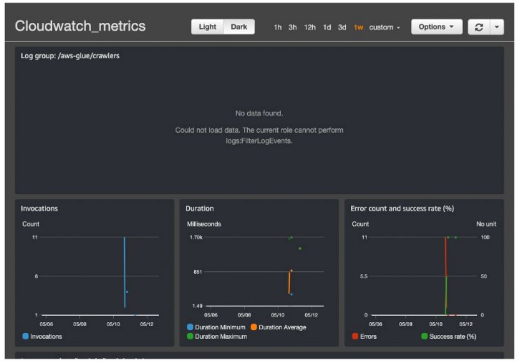
/Front-end

1. Build using ReactJS
2. Deployed in Netlify
3. Configured to a new domain
4. Added the domain in Domain and embedding section of the AWS page

/CURRENTLY WORKING APPLICATION

ETL Pipeline-Dashboard





Log group: /aws-glue/crawlers

No data found.

Could not load data. The current role cannot perform logs:FilterLogEvents.

OPTIMIZATION TECHNIQUES FOR DATA

Compound sort keys

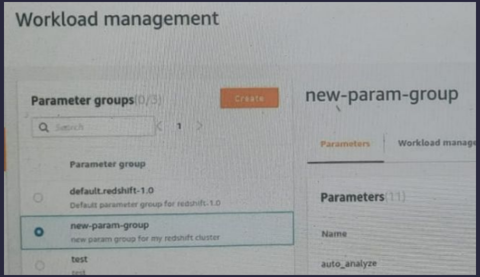
Most frequently used columns put at front for speed

Redshift DIST keys

Data distribution across slices by Leader node matching values of a specific column

Redshift Vacuuming

Reclaims disk space occupied by rows marked for deletion in previous operations⇒ compacts table


Workload management

Parameter groups (0/3)

Search

new-param-group

Parameters | Workload manage

Parameter group

default.redshift-1.0
Default parameter group for redshift-1.0

[✓ new-param-group
new param group for my redshift cluster]

test
test

Parameters (11)

Name

auto_analyze

/INNOVATION

- Using Cloud based service - Amazon Web Services (AWS) for efficiency of the ETL Pipeline.
- AWS optimizes the software in terms of time, space and complexity.

Implemented using AWS Free Tier.

/VIABILITY - INITIAL

<0.114s>

> TIME IN QUEUE

<0.627s>

> RUN TIME

<2.52KB>

> DATA SCANNED

/VIABILITY - FINAL

<0.108s>

> TIME IN QUEUE

<0.446s>

> RUN TIME

<2.52KB>

> DATA SCANNED

HOW REALISTIC IN THE MARKET?

HUGE VOLUMES OF DATA

Great for large scale businesses to work on large data sets

CENTRALIZED LOCATION

Diverse data such as unstructured data could also be used

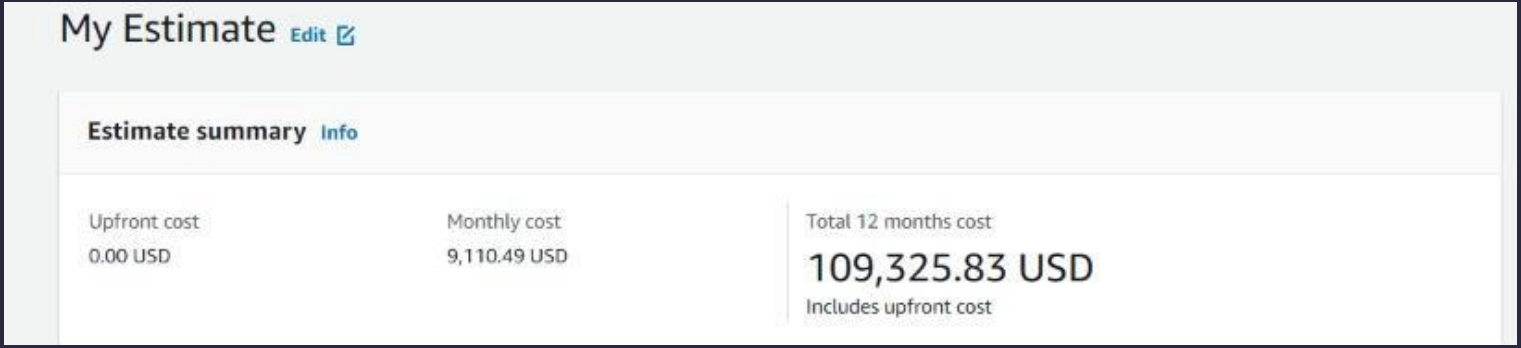
BUSINESS INTELLIGENCE

Easy to analyze using Quick Shift and Cloudwatch

EVERYTHING IS CLOUD NOWADAYS

Safety of all data sets, easy optimization and editing

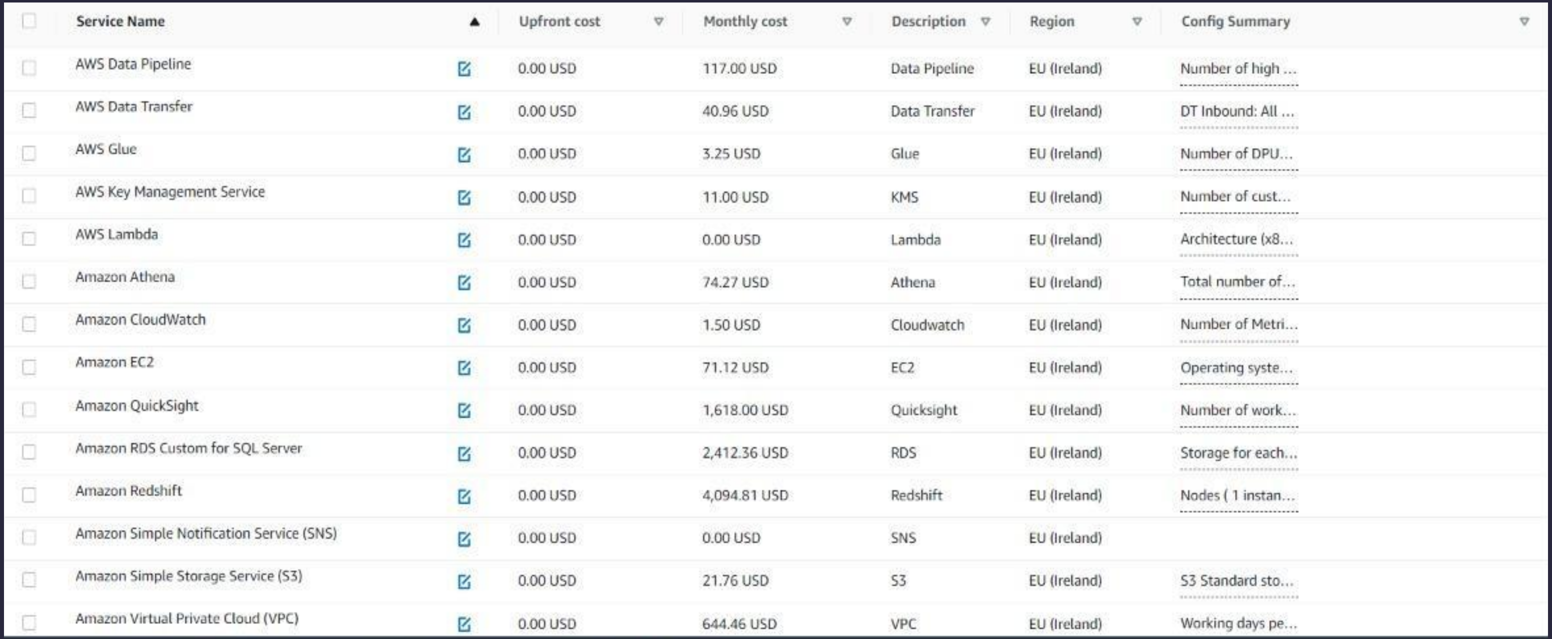
/ROUGH ESTIMATE ($USD)


My Estimate Edit

Estimate summary Info

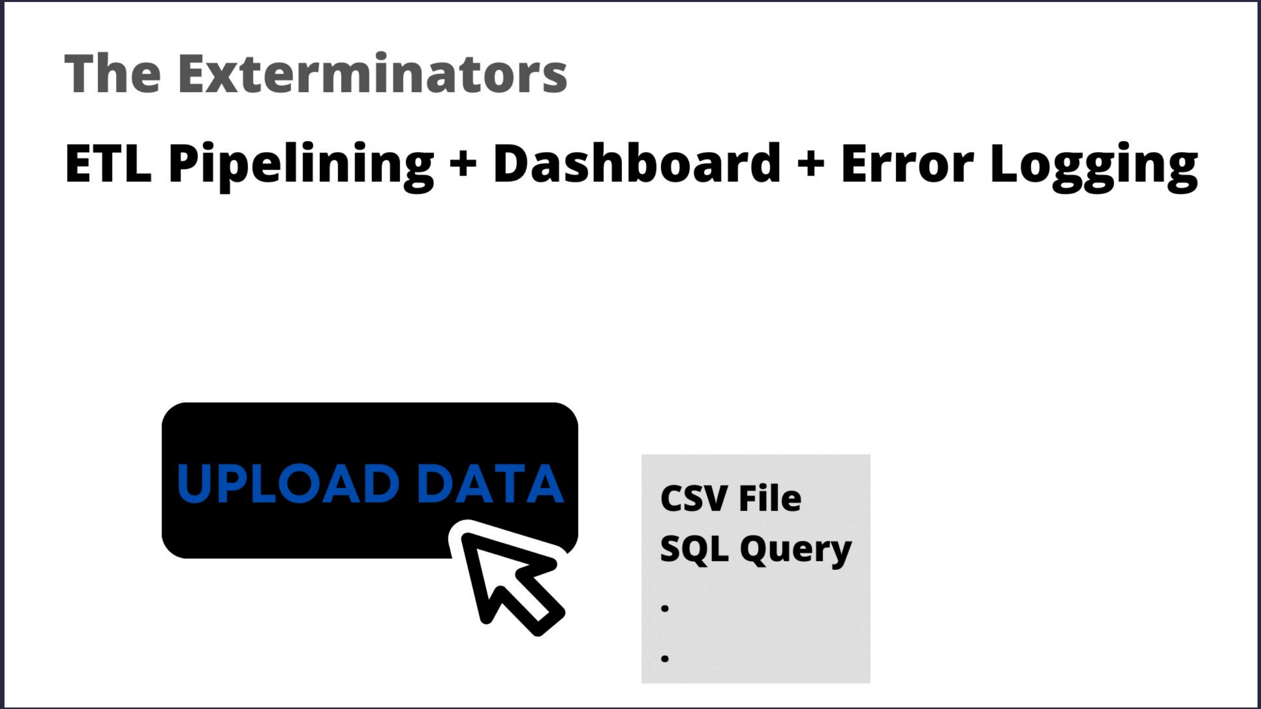
Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	9,110.49 USD	109,325.83 USD Includes upfront cost

/ROUGH ESTIMATE - SERVICES USED



Service Name	Upfront cost	Monthly cost	Description	Region	Config Summary
AWS Data Pipeline	0.00 USD	117.00 USD	Data Pipeline	EU (Ireland)	Number of high ...
AWS Data Transfer	0.00 USD	40.96 USD	Data Transfer	EU (Ireland)	DT Inbound: All ...
AWS Glue	0.00 USD	3.25 USD	Glue	EU (Ireland)	Number of DPU...
AWS Key Management Service	0.00 USD	11.00 USD	KMS	EU (Ireland)	Number of cust...
AWS Lambda	0.00 USD	0.00 USD	Lambda	EU (Ireland)	Architecture (x8...
Amazon Athena	0.00 USD	74.27 USD	Athena	EU (Ireland)	Total number of...
Amazon CloudWatch	0.00 USD	1.50 USD	Cloudwatch	EU (Ireland)	Number of Metri...
Amazon EC2	0.00 USD	71.12 USD	EC2	EU (Ireland)	Operating syste...
Amazon QuickSight	0.00 USD	1,618.00 USD	Quicksight	EU (Ireland)	Number of work...
Amazon RDS Custom for SQL Server	0.00 USD	2,412.36 USD	RDS	EU (Ireland)	Storage for each...
Amazon Redshift	0.00 USD	4,094.81 USD	Redshift	EU (Ireland)	Nodes (1 instan...
Amazon Simple Notification Service (SNS)	0.00 USD	0.00 USD	SNS	EU (Ireland)	
Amazon Simple Storage Service (S3)	0.00 USD	21.76 USD	S3	EU (Ireland)	S3 Standard sto...
Amazon Virtual Private Cloud (VPC)	0.00 USD	644.46 USD	VPC	EU (Ireland)	Working days pe...

/FUTURE PLANS

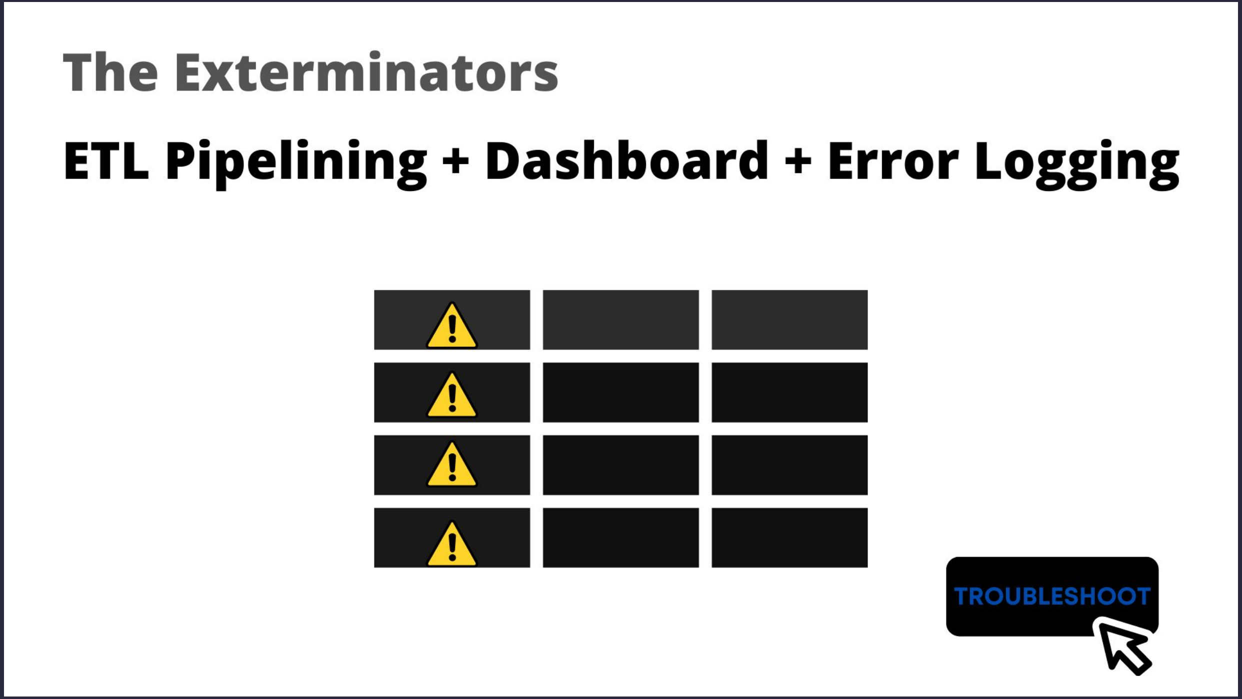


The Exterminators

ETL Pipelining + Dashboard + Error Logging

CSV File
SQL Query
.
.

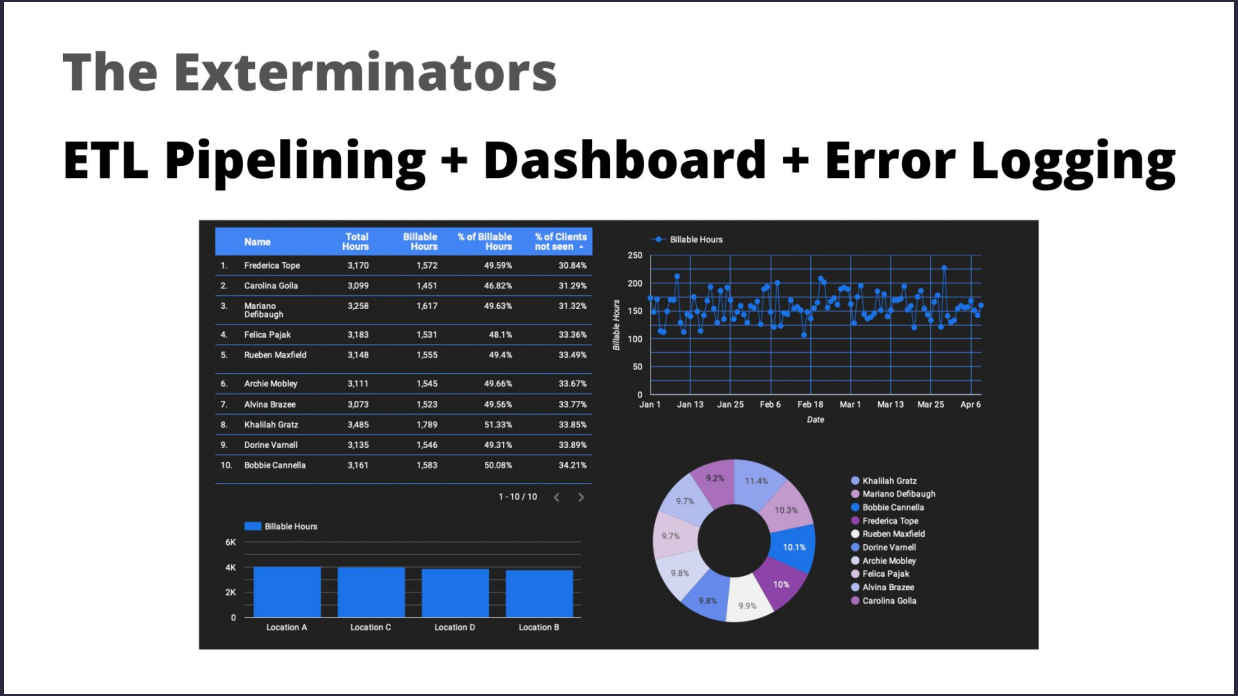
/FUTURE PLANS



The Exterminators

ETL Pipelining + Dashboard + Error Logging

/FUTURE PLANS



The Exterminators

ETL Pipelining + Dashboard + Error Logging

	Name	Total Hours	Billable Hours	% of Billable Hours	% of Clients not seen ▲
1.	Frederica Tope	3,170	1,572	49.59%	30.84%
2.	Carolina Golla	3,099	1,451	46.82%	31.29%
3.	Mariano Defibaugh	3,258	1,617	49.63%	31.32%
4.	Felica Pajak	3,183	1,531	48.1%	33.36%
5.	Rueben Maxfield	3,148	1,555	49.4%	33.49%
6.	Archie Mobley	3,111	1,545	49.66%	33.67%
7.	Alvina Brazee	3,073	1,523	49.56%	33.77%
8.	Khalilah Gratz	3,485	1,789	51.33%	33.85%
9.	Dorine Varnell	3,135	1,546	49.31%	33.89%
10.	Bobbie Cannella	3,161	1,583	50.08%	34.21%

1 - 10 / 10

<THANK YOU>