Capstone Project

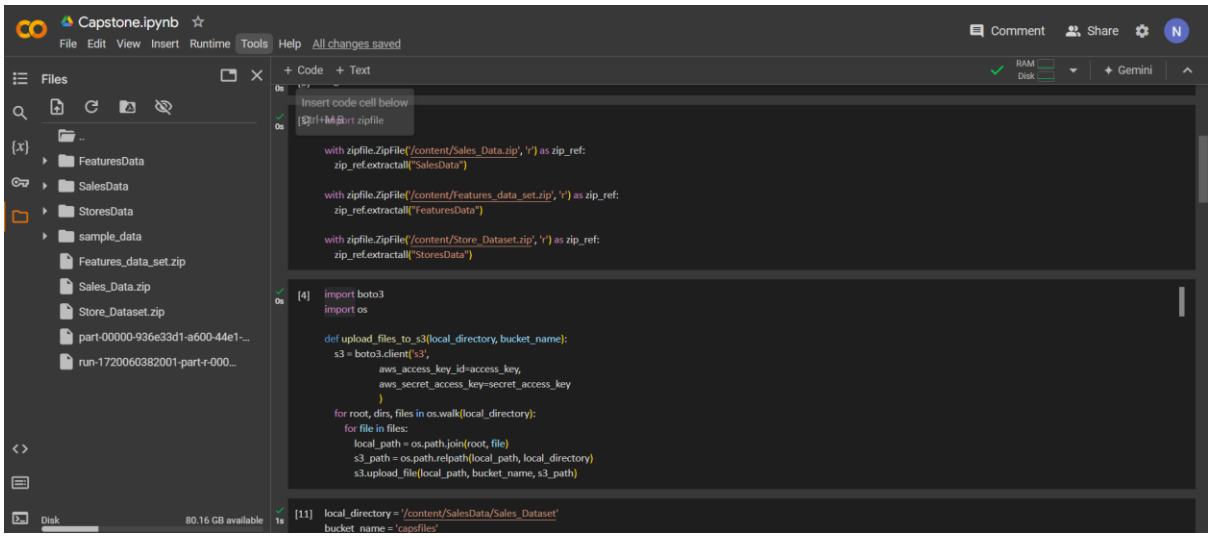
Problem Statement:

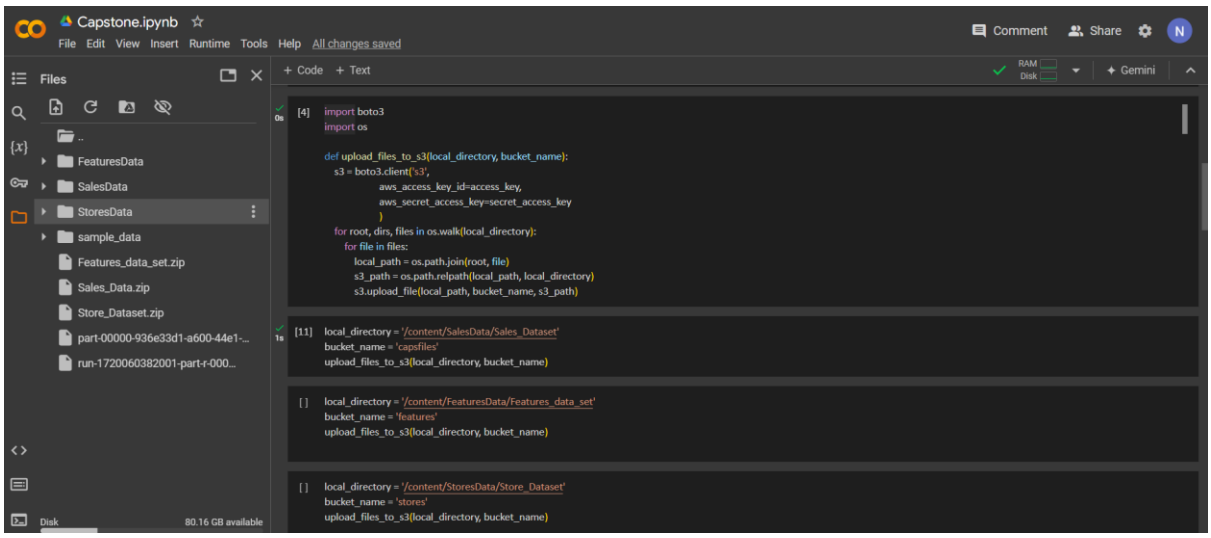
Develop a pipeline where sales data, features data, store is passed to RDS. Then Fetaures data is passed to HDFS and then to HIVE and sales data is passed to HIVE using SQOOP.

Code Working:

Step 1: RUN the Captsone.ipynb file to Run the code.

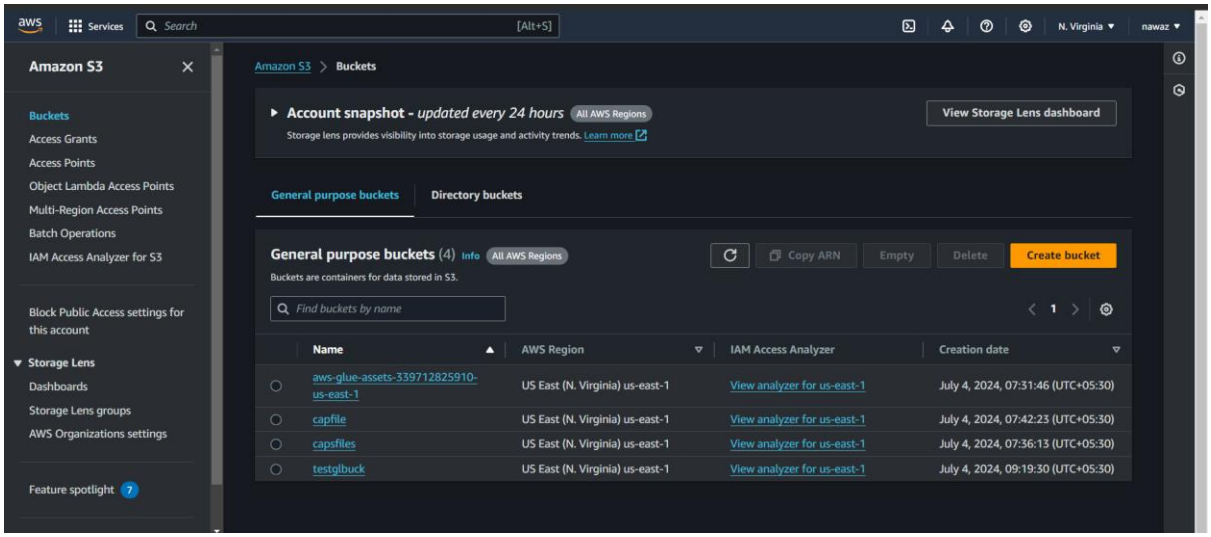
Here we pass the data to S3 bucket





Step 2: Before this we need to create buckets in S3, Glue Job and MYSQL RDS Database & Lambda function

S3: You have Create 2 buckets, 1st Contains the all file in specific folder and another bucket to store Merge file from GLUE JOB.



GLUE: Create a ETL Job in Glue and add this script in it.

```
import sys

from awsglue.transforms import *

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job


args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark_session

job = Job(glueContext)

job.init(args['JOB_NAME'], args)


# Script generated for node Amazon S3
```

```
AmazonS3_node1720059575737 =
glueContext.create_dynamic_frame.from_options(format_options={"quoteChar": "\"",
"withHeader": True, "separator": ",", "optimizePerformance": False}, connection_type="s3",
format="csv", connection_options={"paths": ["s3://capsfiles"], "recurse": True},
transformation_ctx="AmazonS3_node1720059575737")


# Script generated for node Change Schema

ChangeSchema_node1720059587518 =
ApplyMapping.apply(frame=AmazonS3_node1720059575737, mappings=[("store", "string", "store",
"string"), ("dept", "string", "dept", "string"), ("date", "string", "date", "string"), ("weekly_sales",
"string", "weekly_sales", "string"), ("isholiday", "string", "isholiday", "string")],
transformation_ctx="ChangeSchema_node1720059587518")


repar = ChangeSchema_node1720059587518.repartition(1)


# Script generated for node Amazon S3

AmazonS3_node1720059579346 = glueContext.write_dynamic_frame.from_options(frame=repar,
connection_type="s3", format="parquet", connection_options={"path": "s3://capfile/finalsales/",
"partitionKeys": []}, transformation_ctx="AmazonS3_node1720059579346")


job.commit()
```

This will merge sales data and places merge data as parquet or either csv in the mentioned bucket.

RDS: Create a RDS Database with MYSQL Database as free tier.

LAMBDA: Now we have to create 3 lambda jobs. Each will get triggered to pass sales, features and store data to RDS.

SALES DATA LAMBDA: Create a Lambda trigger with prefix as 'r'

```
import json

import boto3

import csv

import mysql.connector

from mysql.connector import Error
```

```
from mysql.connector import errorcode

s3_client = boto3.client('s3')

def lambda_handler(event, context):

    # TODO implement

    bucket = event['Records'][0]['s3']['bucket']['name']

    csv_file = event['Records'][0]['s3']['object']['key']

    csv_file_obj = s3_client.get_object(Bucket=bucket, Key=csv_file)

    lines = csv_file_obj['Body'].read().decode('utf-8').split()

    results = []

    for row in csv.DictReader(lines):

        results.append(row.values())

    print(results)

    connection = mysql.connector.connect(host='salescaps.cty66u0swnw5.us-east-
1.rds.amazonaws.com',database='salescaps',user='admin',

    password='UlMysql$')

    mysql_empsql_insert_query = "INSERT INTO sales (store, Dept, Dates, weekly_sales, isHoliday)
VALUES (%s, %s, %s, %s,%s)"

    cursor = connection.cursor()

    cursor.executemany(mysql_empsql_insert_query,results)

    connection.commit()

    print(cursor.rowcount, "Record inserted successfully into employee table")

    return {

        'statusCode': 200,

        'body': json.dumps('Hello from Lambda!')

    }
```

FEATURES DATA LAMBDA: Create a Lambda trigger with suffix as .csv

```
import json

import boto3

import csv

import mysql.connector

from mysql.connector import Error

from mysql.connector import errorcode

s3_client = boto3.client('s3')

def lambda_handler(event, context):

    # TODO implement

    bucket = event['Records'][0]['s3']['bucket']['name']

    csv_file = event['Records'][0]['s3']['object']['key']

    csv_file_obj = s3_client.get_object(Bucket=bucket, Key=csv_file)

    lines = csv_file_obj['Body'].read().decode('utf-8').split()

    results = []

    for row in csv.DictReader(lines):

        results.append(row.values())

    print(results)

    connection = mysql.connector.connect(host='salescaps.cty66u0swnw5.us-east-
1.rds.amazonaws.com',database='salescaps',user='admin',

    password='UlMysql$')

    mysql_empsql_insert_query = "INSERT INTO features
(Store,Date,Temperature,Fuel_Price,MarkDown1,MarkDown2,MarkDown3,MarkDown4,MarkDown5
,CPI,Unemployment,IsHoliday)    VALUES (%s, %s, %s, %s,%s, %s, %s, %s, %s,%s, %s, %s)"

    cursor = connection.cursor()

    cursor.executemany(mysql_empsql_insert_query,results)

    connection.commit()

    print(cursor.rowcount, "Record inserted successfully into table")

    return {
```

```
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

STORES DATA LAMBDA: Create a Lambda trigger with suffix as .csv

```
import json
import boto3
import csv
import mysql.connector
from mysql.connector import Error
from mysql.connector import errorcode
s3_client = boto3.client('s3')
def lambda_handler(event, context):
    # TODO implement
    bucket = event['Records'][0]['s3']['bucket']['name']
    csv_file = event['Records'][0]['s3']['object']['key']
    csv_file_obj = s3_client.get_object(Bucket=bucket, Key=csv_file)
    lines = csv_file_obj['Body'].read().decode('utf-8').split()
    results = []
    for row in csv.DictReader(lines):
        results.append(row.values())
    print(results)
    connection = mysql.connector.connect(host='salescaps.cty66u0swnw5.us-east-
1.rds.amazonaws.com',database='salescaps',user='admin',
    password='UlMysql$')
    mysql_empsql_insert_query = "INSERT INTO stores (store, Dept, Dates, weekly_sales, isHoliday)
VALUES (%s, %s, %s, %s,%s)"
    cursor = connection.cursor()
    cursor.executemany(mysql_empsql_insert_query,results)
```

```
connection.commit()

print(cursor.rowcount, "Record inserted successfully into table")

return {

    'statusCode': 200,

    'body': json.dumps('Hello from Lambda!')

}
```

<u>STEP2</u>: Open the Test folder in Docker and build the docker image for Docker file.

docker build -t capsdata-image .

docker run -it --name capsdata-container -p 8080:8080 -p 50070:50070 -p 8088:8088 -p 10000:10000 bigdata-image