

# Capstone Project

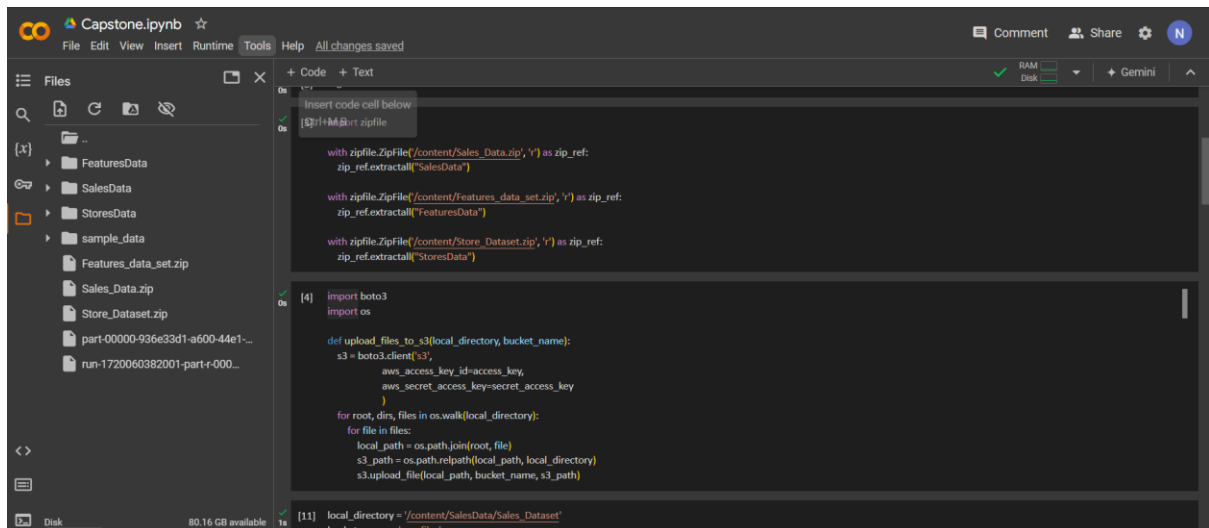
## Problem Statement:

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

## Code Working:

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

Here we pass the data to S3 bucket



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with folders like 'FeaturesData', 'SalesData', and 'StoresData', and files like 'Features\_data\_set.zip', 'Sales\_Data.zip', and 'Store\_Dataset.zip'. The code editor contains the following code:

```
with zipfile.ZipFile('/content/Sales_Data.zip', 'r') as zip_ref:
    zip_ref.extractall("SalesData")

with zipfile.ZipFile('/content/Features_data_set.zip', 'r') as zip_ref:
    zip_ref.extractall("FeaturesData")

with zipfile.ZipFile('/content/Store_Dataset.zip', 'r') as zip_ref:
    zip_ref.extractall("StoresData")
```

Below this code, there is a code cell with the following code:

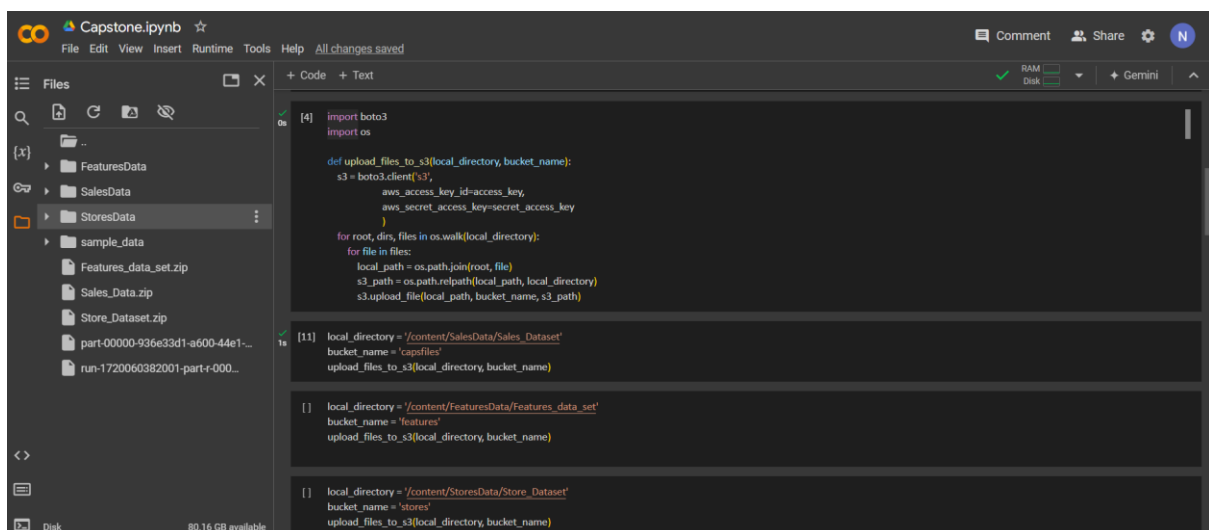
```
[4] import boto3
import os

def upload_files_to_s3(local_directory, bucket_name):
    s3 = boto3.client('s3',
                      aws_access_key_id=access_key,
                      aws_secret_access_key=secret_access_key)

    for root, dirs, files in os.walk(local_directory):
        for file in files:
            local_path = os.path.join(root, file)
            s3_path = os.path.relpath(local_path, local_directory)
            s3.upload_file(local_path, bucket_name, s3_path)
```

At the bottom, there is a code cell with the following code:

```
[11] local_directory = '/content/SalesData/Sales_Dataset'
      bucket_name = 'capfiles'
```



The screenshot shows the same Jupyter Notebook interface as the previous one, but with the 'StoresData' folder selected in the file explorer. The code editor contains the following code:

```
[4] import boto3
import os

def upload_files_to_s3(local_directory, bucket_name):
    s3 = boto3.client('s3',
                      aws_access_key_id=access_key,
                      aws_secret_access_key=secret_access_key)

    for root, dirs, files in os.walk(local_directory):
        for file in files:
            local_path = os.path.join(root, file)
            s3_path = os.path.relpath(local_path, local_directory)
            s3.upload_file(local_path, bucket_name, s3_path)
```

Below this code, there is a code cell with the following code:

```
[11] local_directory = '/content/SalesData/Sales_Dataset'
      bucket_name = 'capfiles'
      upload_files_to_s3(local_directory, bucket_name)
```

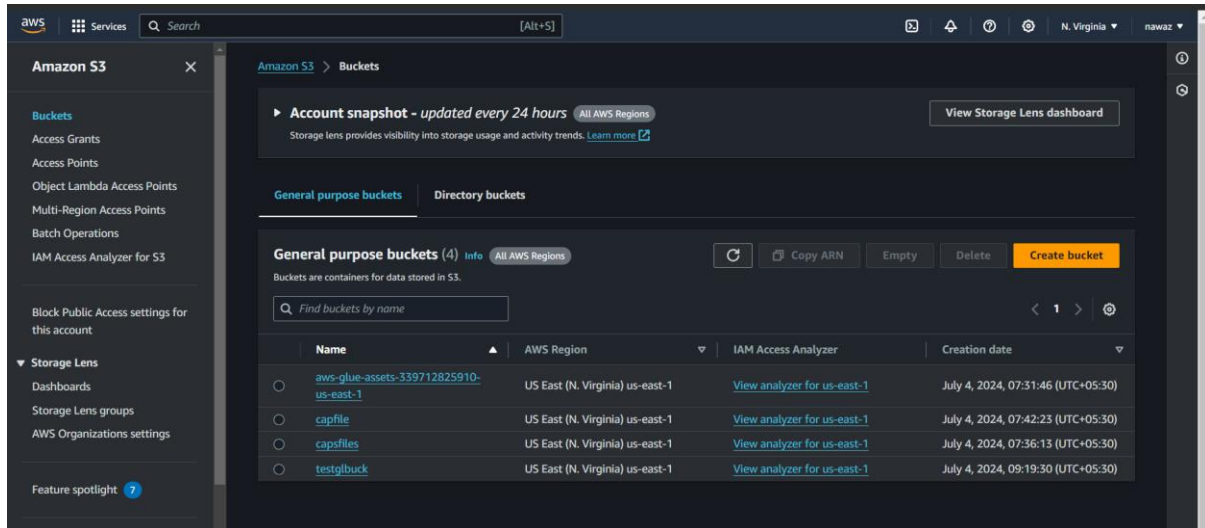
At the bottom, there are two more code cells, each with the following code:

```
[ ] local_directory = '/content/FeaturesData/Features_data_set'
     bucket_name = 'features'
     upload_files_to_s3(local_directory, bucket_name)
```

```
[ ] local_directory = '/content/StoresData/Store_Dataset'
     bucket_name = 'stores'
     upload_files_to_s3(local_directory, bucket_name)
```

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, 1<sup>st</sup> 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='UIMysql$')

    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='UIMysql$')

    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='UIMysql$')  
  
    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!')

}

```

**STEP2:** 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`

