ASSIGNMENT-3
BIGDATA AND ECOSYSTEM

NAME: BHAVYA BHARDWAJ

BLEND ALL STARS

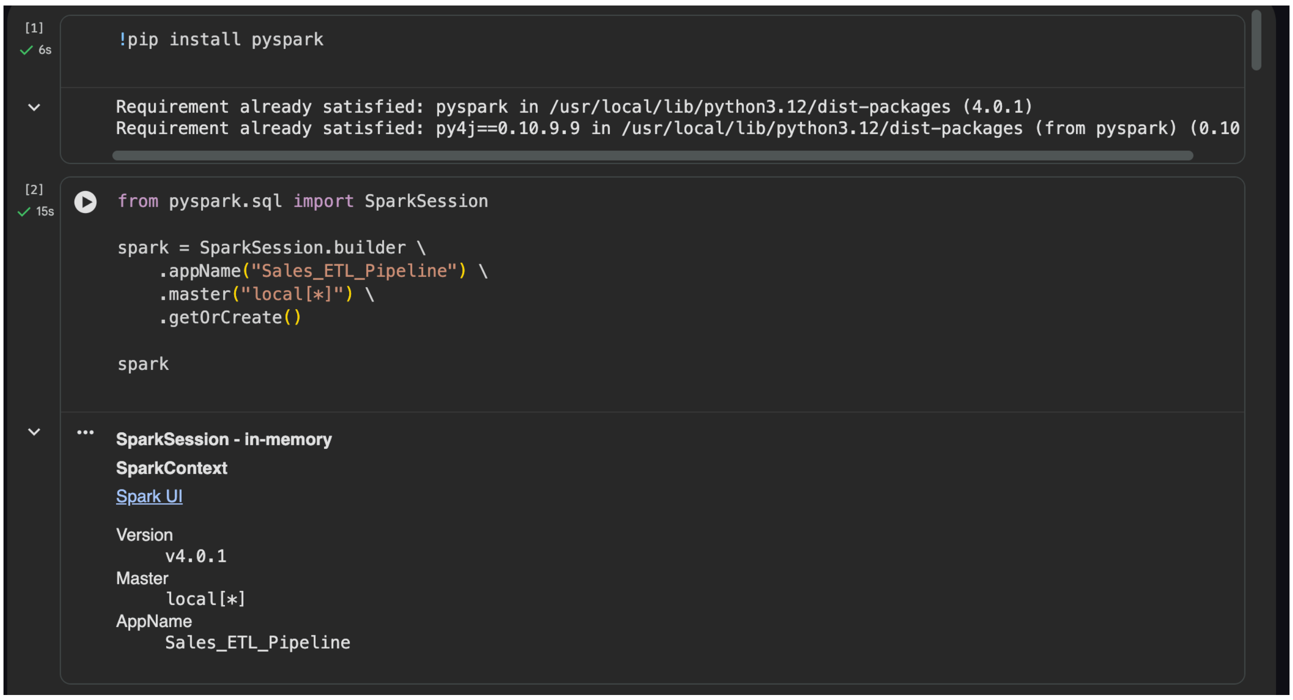
PySpark ETL Pipeline

Goal:

Align with the Python work to process large sales data using PySpark and generate enriched insights.
Tasks:
Read raw CSVs from HDFS or local folder.
Transform and clean the data (handle missing values, duplicates, etc.).
Enrich data by calculating KPIs such as:

Monthly Revenue
Profit Margin (%)
Region-wise Sales
Average Order Value
(Come up with 3–4 more KPIs as needed)

Write aggregated results to Parquet or a managed table.
(Optional) Integrate with Kafka for streaming order ingestion.
Tech: PySpark, HDFS, Kafka (optional)
Deliverables: PySpark script or notebook + pipeline diagram + screenshots (Spark job output, DAG view, Parquet output sample)
Dataset: https://wgcp-my.sharepoint.com/:f:/g/personal/ritish_jogi_blend360_com/EmxzwFjNkaxPuCw2mQ0abr0BGg6XzIPlj22VogFVtQniyg?e=aglSyr

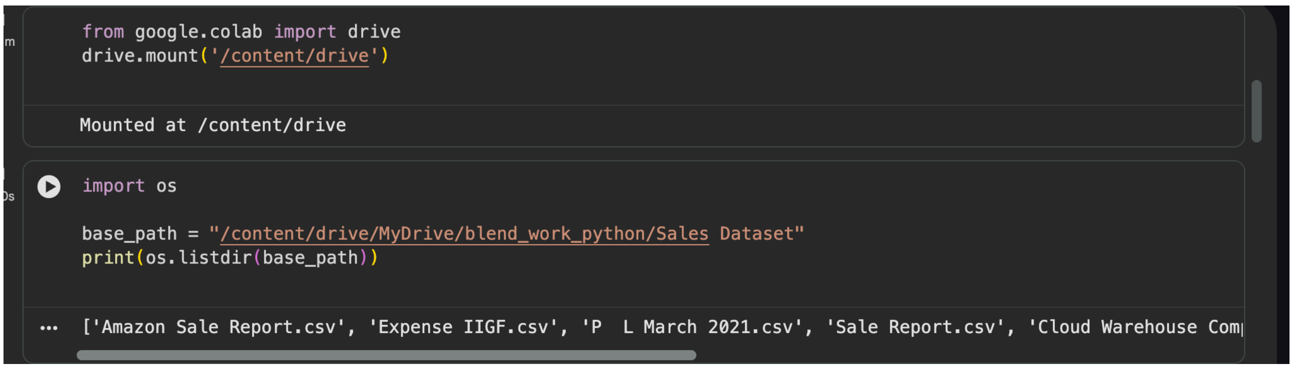


```
[1]
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.12/dist-packages (4.0.1)
Requirement already satisfied: py4j==0.10.9.9 in /usr/local/lib/python3.12/dist-packages (from pyspark) (0.10
```

```
[2]
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Sales_ETL_Pipeline") \
    .master("local[*]") \
    .getOrCreate()

spark
```

```
SparkSession - in-memory
SparkContext
Spark UI

Version
    v4.0.1
Master
    local[*]
AppName
    Sales_ETL_Pipeline
```

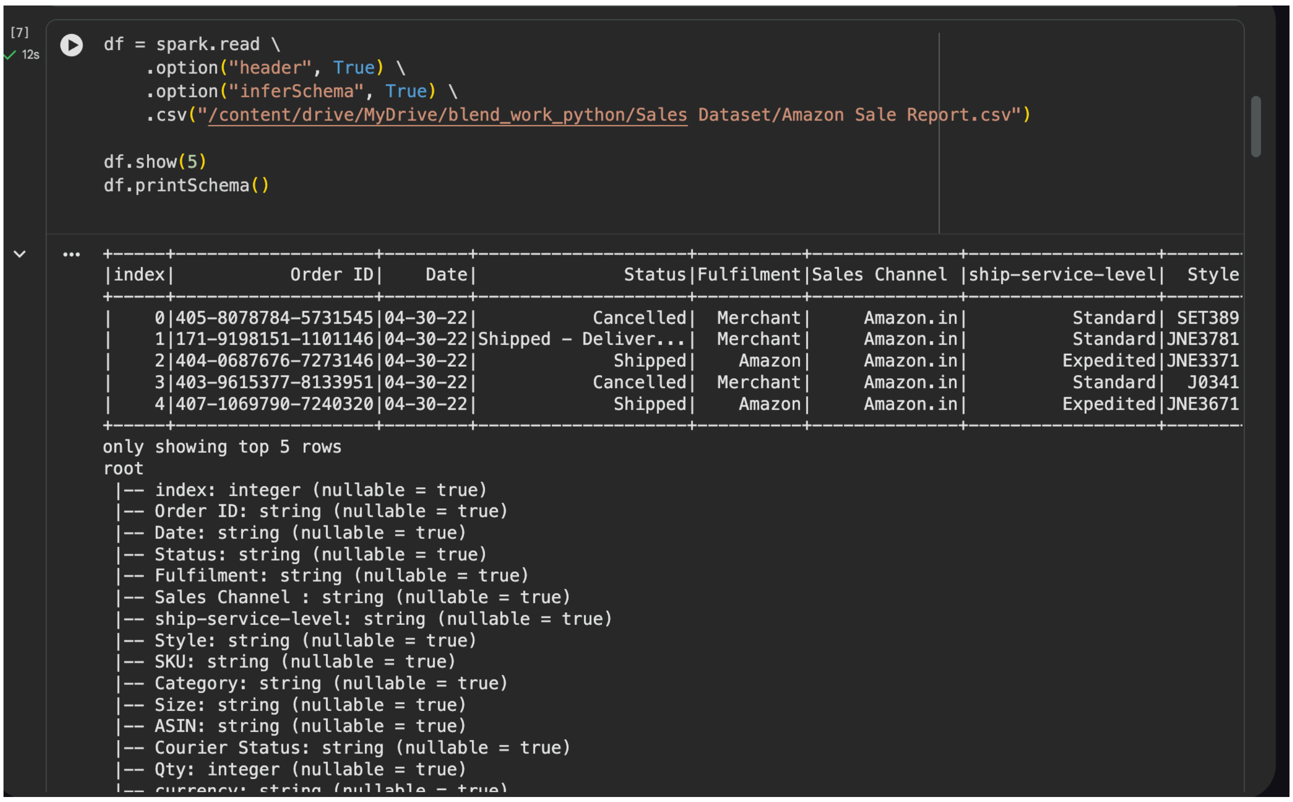


```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import os

base_path = "/content/drive/MyDrive/blend_work_python/Sales Dataset"
print(os.listdir(base_path))
```

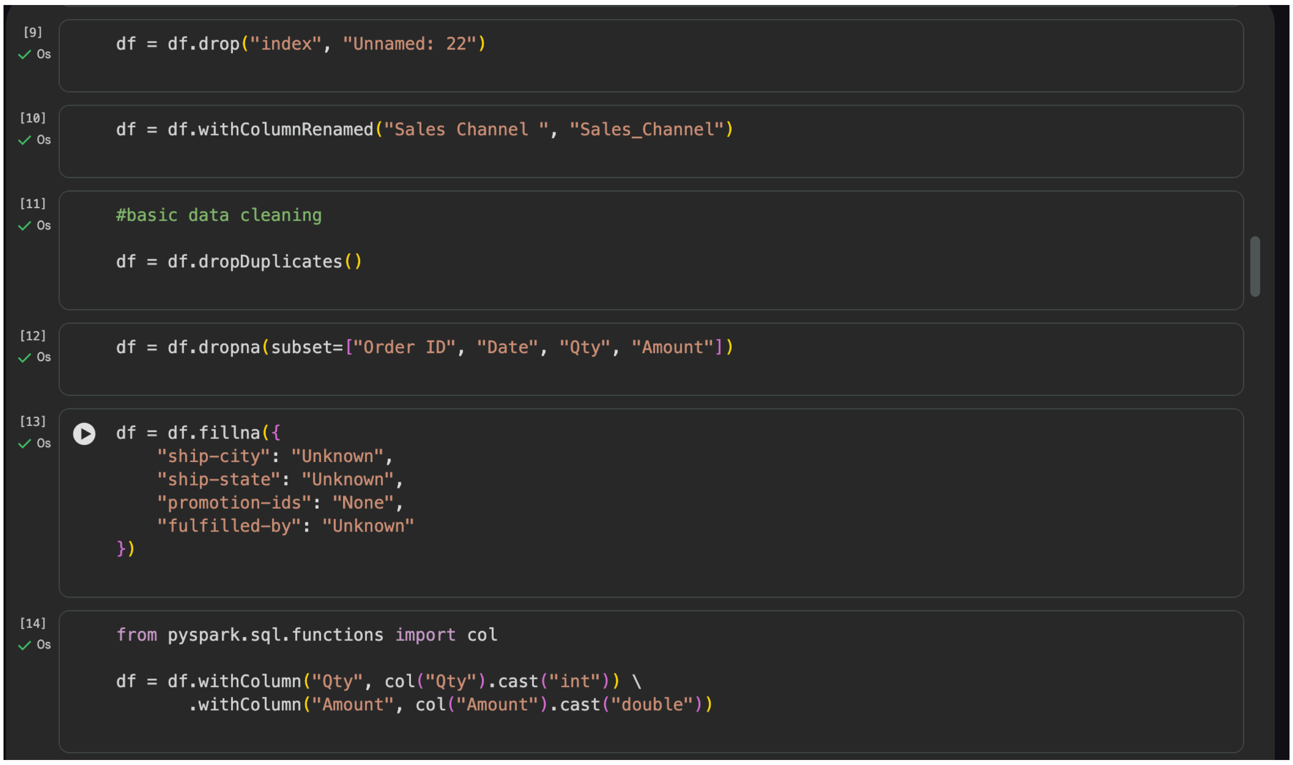
```
['Amazon Sale Report.csv', 'Expense IIGF.csv', 'P  L March 2021.csv', 'Sale Report.csv', 'Cloud Warehouse Com
```



```python
df = spark.read \
    .option("header", True) \
    .option("inferSchema", True) \
    .csv("/content/drive/MyDrive/blend_work_python/Sales Dataset/Amazon Sale Report.csv")

df.show(5)
df.printSchema()
```

```
+-----+-------------------+--------+--------------------+----------+--------------+------------------+--------
|index|           Order ID|    Date|              Status|Fulfilment|Sales Channel |ship-service-level|   Style
+-----+-------------------+--------+--------------------+----------+--------------+------------------+--------
|    0|405-8078784-5731545|04-30-22|           Cancelled|  Merchant|     Amazon.in|          Standard| SET389
|    1|171-9198151-1101146|04-30-22|Shipped - Deliver...|  Merchant|     Amazon.in|          Standard|JNE3781
|    2|404-0687676-7273146|04-30-22|             Shipped|    Amazon|     Amazon.in|         Expedited|JNE3371
|    3|403-9615377-8133951|04-30-22|           Cancelled|  Merchant|     Amazon.in|          Standard|  J0341
|    4|407-1069790-7240320|04-30-22|             Shipped|    Amazon|     Amazon.in|         Expedited|JNE3671
+-----+-------------------+--------+--------------------+----------+--------------+------------------+--------
only showing top 5 rows
root
 |-- index: integer (nullable = true)
 |-- Order ID: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- Status: string (nullable = true)
 |-- Fulfilment: string (nullable = true)
 |-- Sales Channel : string (nullable = true)
 |-- ship-service-level: string (nullable = true)
 |-- Style: string (nullable = true)
 |-- SKU: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Size: string (nullable = true)
 |-- ASIN: string (nullable = true)
 |-- Courier Status: string (nullable = true)
 |-- Qty: integer (nullable = true)
 |-- currency: string (nullable = true)
```

Basic data cleaning



```python
df = df.drop("index", "Unnamed: 22")
```

```python
df = df.withColumnRenamed("Sales Channel ", "Sales_Channel")
```

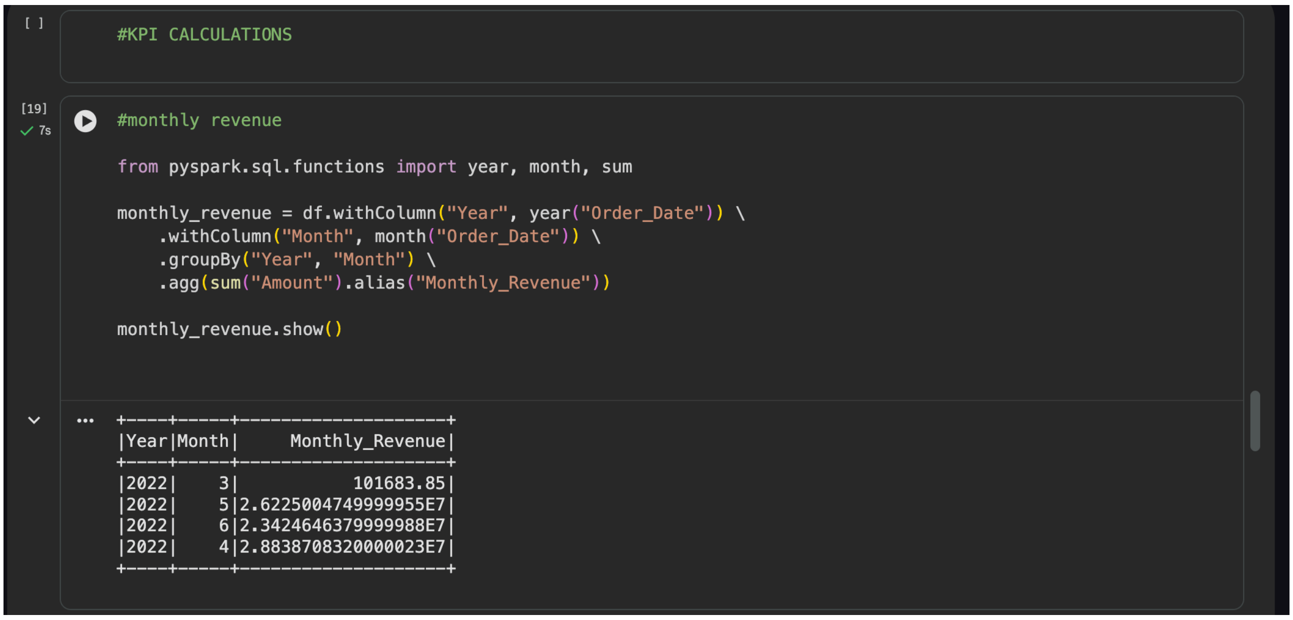
```python
#basic data cleaning

df = df.dropDuplicates()
```

```python
df = df.dropna(subset=["Order ID", "Date", "Qty", "Amount"])
```

```python
df = df.fillna({
    "ship-city": "Unknown",
    "ship-state": "Unknown",
    "promotion-ids": "None",
    "fulfilled-by": "Unknown"
})
```

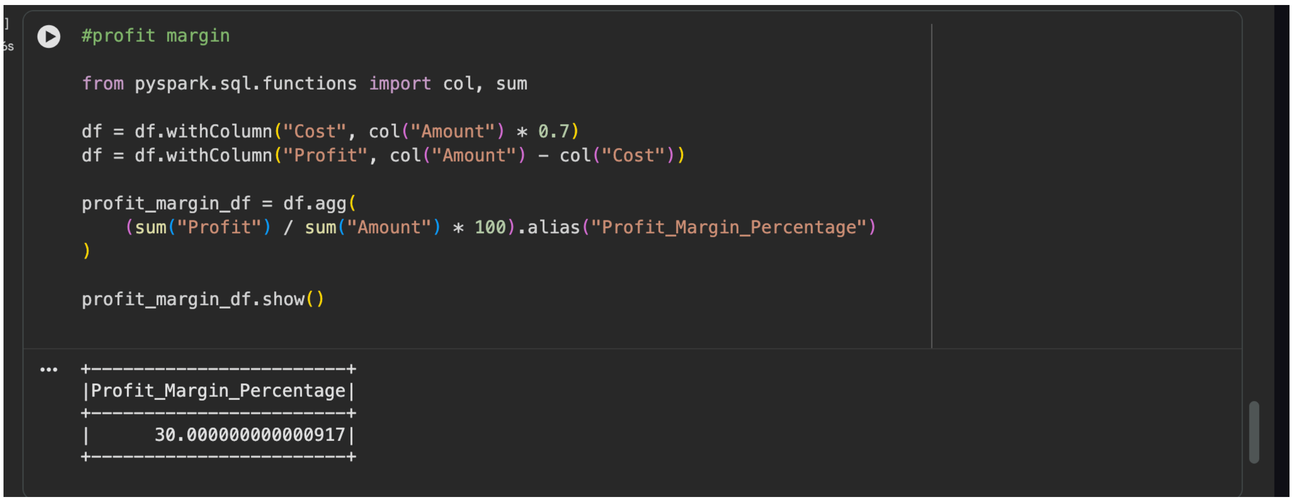
```python
from pyspark.sql.functions import col

df = df.withColumn("Qty", col("Qty").cast("int")) \
       .withColumn("Amount", col("Amount").cast("double"))
```

BASIC KPI :DELIVERABLES NEEDED



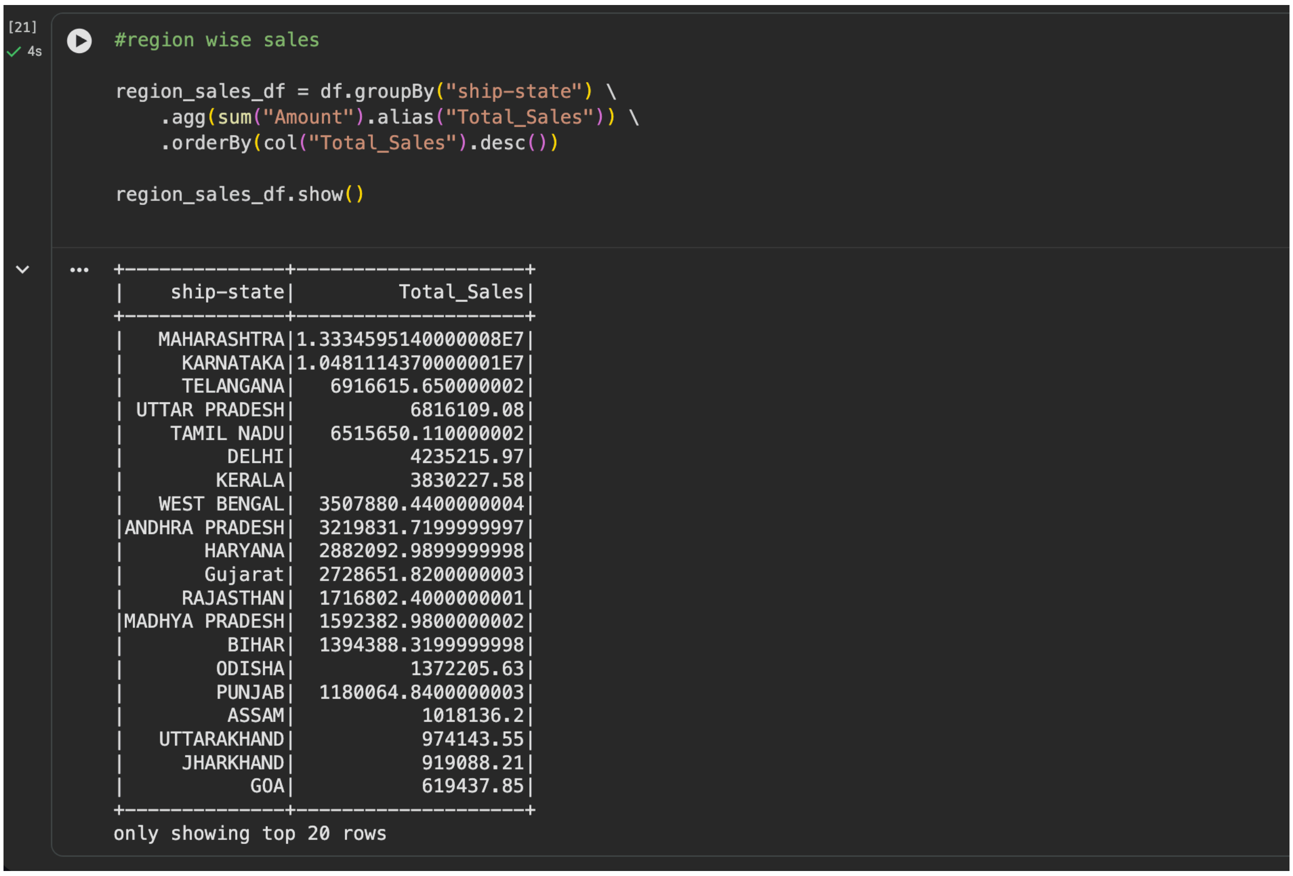
```
[ ]
    #KPI CALCULATIONS

[19]
    #monthly revenue

    from pyspark.sql.functions import year, month, sum

    monthly_revenue = df.withColumn("Year", year("Order_Date")) \
        .withColumn("Month", month("Order_Date")) \
        .groupBy("Year", "Month") \
        .agg(sum("Amount").alias("Monthly_Revenue"))

    monthly_revenue.show()
```

```
+----+-----+--------------------+
|Year|Month|     Monthly_Revenue|
+----+-----+--------------------+
|2022|    3|           101683.85|
|2022|    5|2.6225004749999955E7|
|2022|    6|2.3424646379999988E7|
|2022|    4|2.8838708320000023E7|
+----+-----+--------------------+
```



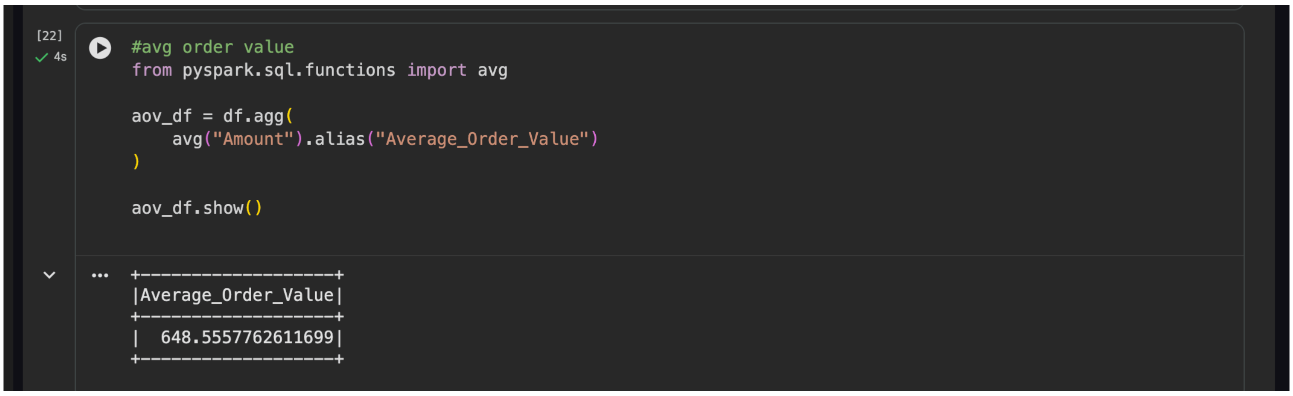
```
#profit margin

from pyspark.sql.functions import col, sum

df = df.withColumn("Cost", col("Amount") * 0.7)
df = df.withColumn("Profit", col("Amount") - col("Cost"))

profit_margin_df = df.agg(
    (sum("Profit") / sum("Amount") * 100).alias("Profit_Margin_Percentage")
)

profit_margin_df.show()
```

```
+------------------------+
|Profit_Margin_Percentage|
+------------------------+
|       30.000000000000917|
+------------------------+
```



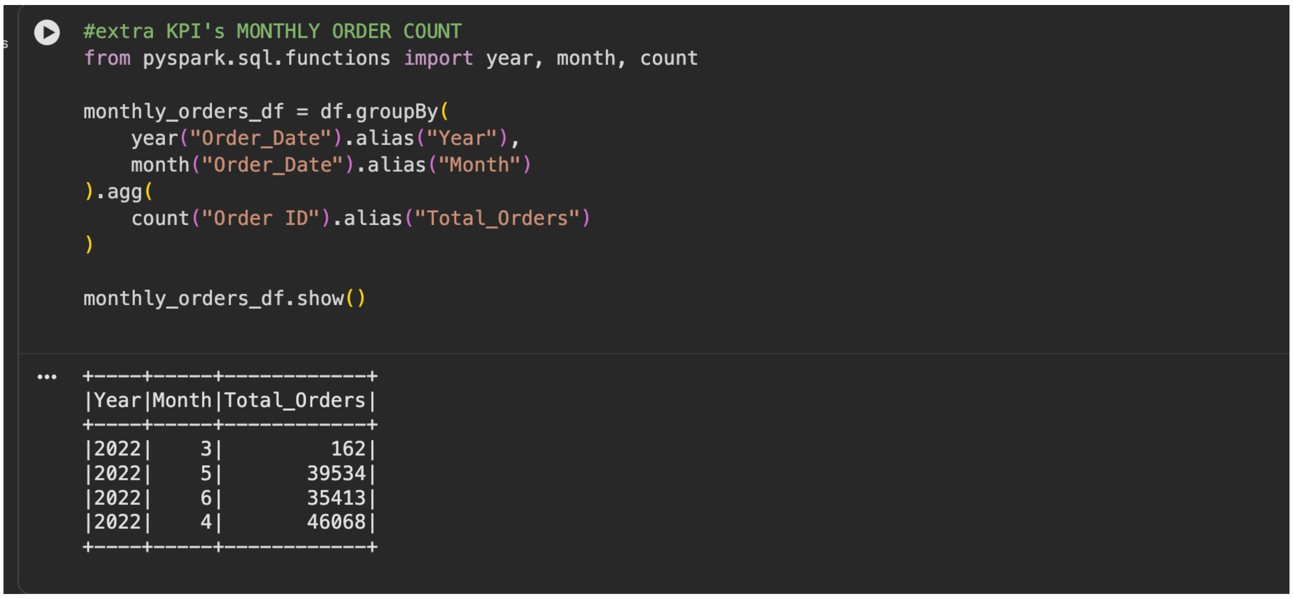
```
#region wise sales

region_sales_df = df.groupBy("ship-state") \
    .agg(sum("Amount").alias("Total_Sales")) \
    .orderBy(col("Total_Sales").desc())

region_sales_df.show()
```

```
+--------------+--------------------+
|    ship-state|         Total_Sales|
+--------------+--------------------+
|   MAHARASHTRA|1.3334595140000008E7|
|     KARNATAKA|1.0481114370000001E7|
|     TELANGANA|   6916615.650000002|
| UTTAR PRADESH|          6816109.08|
|    TAMIL NADU|   6515650.110000002|
|         DELHI|          4235215.97|
|        KERALA|          3830227.58|
|   WEST BENGAL|  3507880.4400000004|
|ANDHRA PRADESH|  3219831.7199999997|
|       HARYANA|  2882092.9899999998|
|       Gujarat|  2728651.8200000003|
|     RAJASTHAN|  1716802.4000000001|
|MADHYA PRADESH|  1592382.9800000002|
|         BIHAR|  1394388.3199999998|
|        ODISHA|          1372205.63|
|        PUNJAB|  1180064.8400000003|
|         ASSAM|           1018136.2|
|   UTTARAKHAND|           974143.55|
|     JHARKHAND|           919088.21|
|           GOA|           619437.85|
+--------------+--------------------+
only showing top 20 rows
```



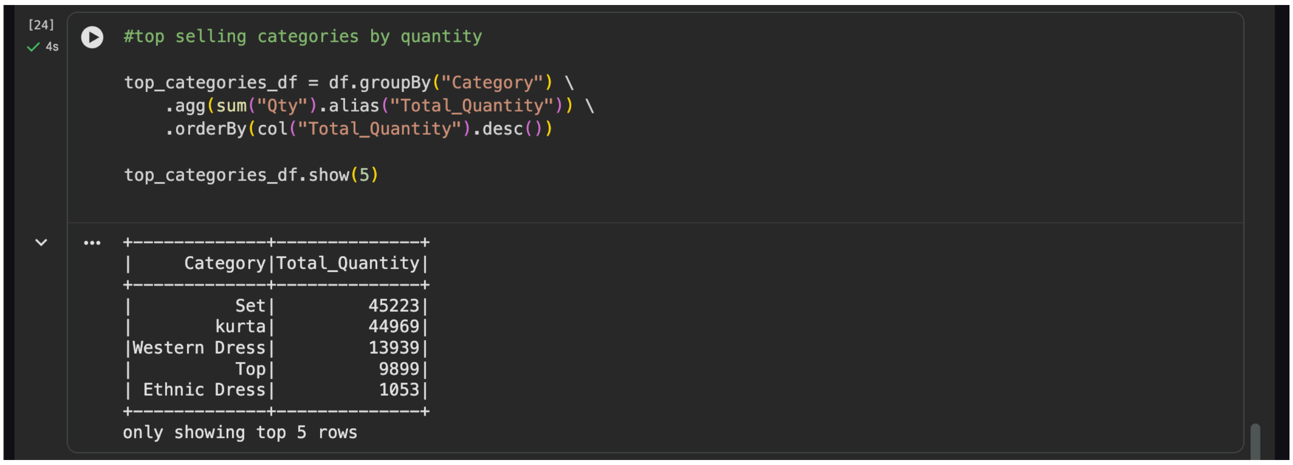
```
#avg order value
from pyspark.sql.functions import avg

aov_df = df.agg(
    avg("Amount").alias("Average_Order_Value")
)

aov_df.show()
```

```
+-------------------+
|Average_Order_Value|
+-------------------+
|  648.5557762611699|
+-------------------+
```



```
#extra KPI's MONTHLY ORDER COUNT
from pyspark.sql.functions import year, month, count

monthly_orders_df = df.groupBy(
    year("Order_Date").alias("Year"),
    month("Order_Date").alias("Month")
).agg(
    count("Order ID").alias("Total_Orders")
)

monthly_orders_df.show()
```

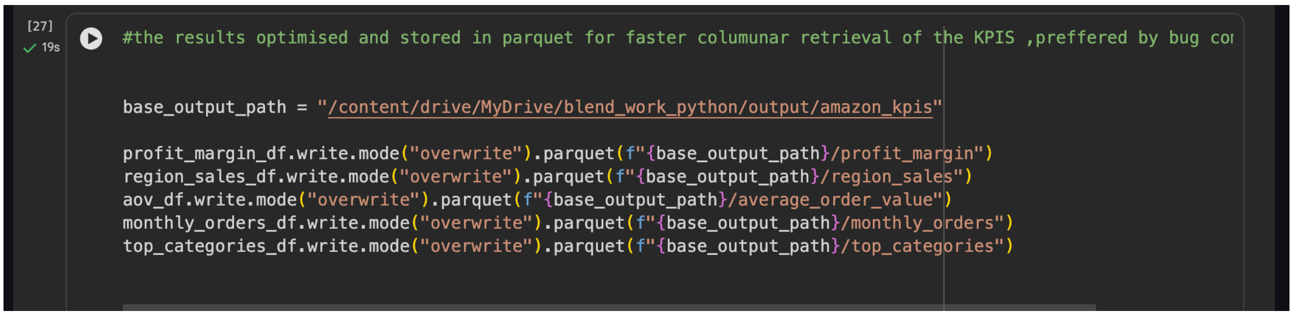
```
+----+-----+------------+
|Year|Month|Total_Orders|
+----+-----+------------+
|2022|    3|         162|
|2022|    5|       39534|
|2022|    6|       35413|
|2022|    4|       46068|
+----+-----+------------+
```



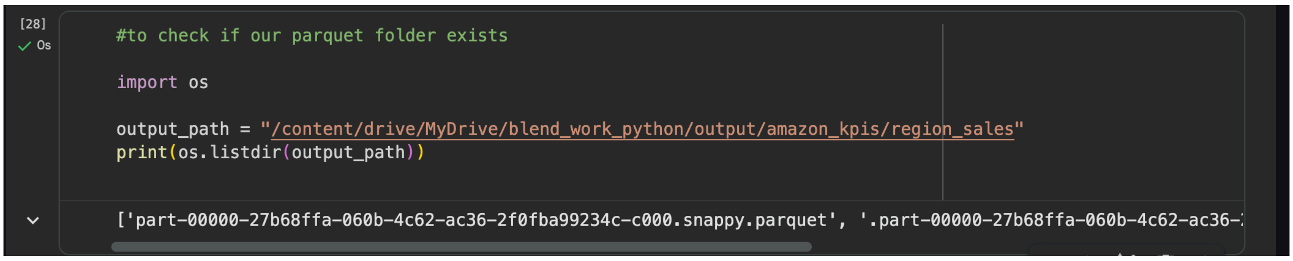
```
[24]
✓ 4s
#top selling categories by quantity

top_categories_df = df.groupBy("Category") \
    .agg(sum("Qty").alias("Total_Quantity")) \
    .orderBy(col("Total_Quantity").desc())

top_categories_df.show(5)
```

```
+-------------+--------------+
|     Category|Total_Quantity|
+-------------+--------------+
|          Set|         45223|
|        kurta|         44969|
|Western Dress|         13939|
|          Top|          9899|
|  Ethnic Dress|          1053|
+-------------+--------------+
only showing top 5 rows
```

STORING THE OPTIMIZED RESULTS IN PARQUET

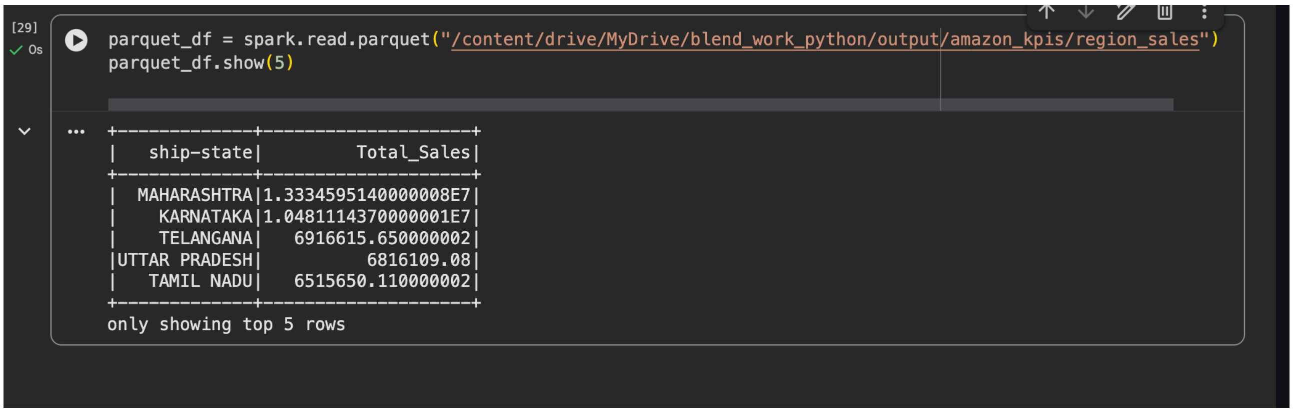


```
[27]
✓ 19s
#the results optimised and stored in parquet for faster columunar retrieval of the KPIS ,preffered by bug co

base_output_path = "/content/drive/MyDrive/blend_work_python/output/amazon_kpis"

profit_margin_df.write.mode("overwrite").parquet(f"{base_output_path}/profit_margin")
region_sales_df.write.mode("overwrite").parquet(f"{base_output_path}/region_sales")
aov_df.write.mode("overwrite").parquet(f"{base_output_path}/average_order_value")
monthly_orders_df.write.mode("overwrite").parquet(f"{base_output_path}/monthly_orders")
top_categories_df.write.mode("overwrite").parquet(f"{base_output_path}/top_categories")
```



```
[28]
✓ 0s
#to check if our parquet folder exists

import os

output_path = "/content/drive/MyDrive/blend_work_python/output/amazon_kpis/region_sales"
print(os.listdir(output_path))
```

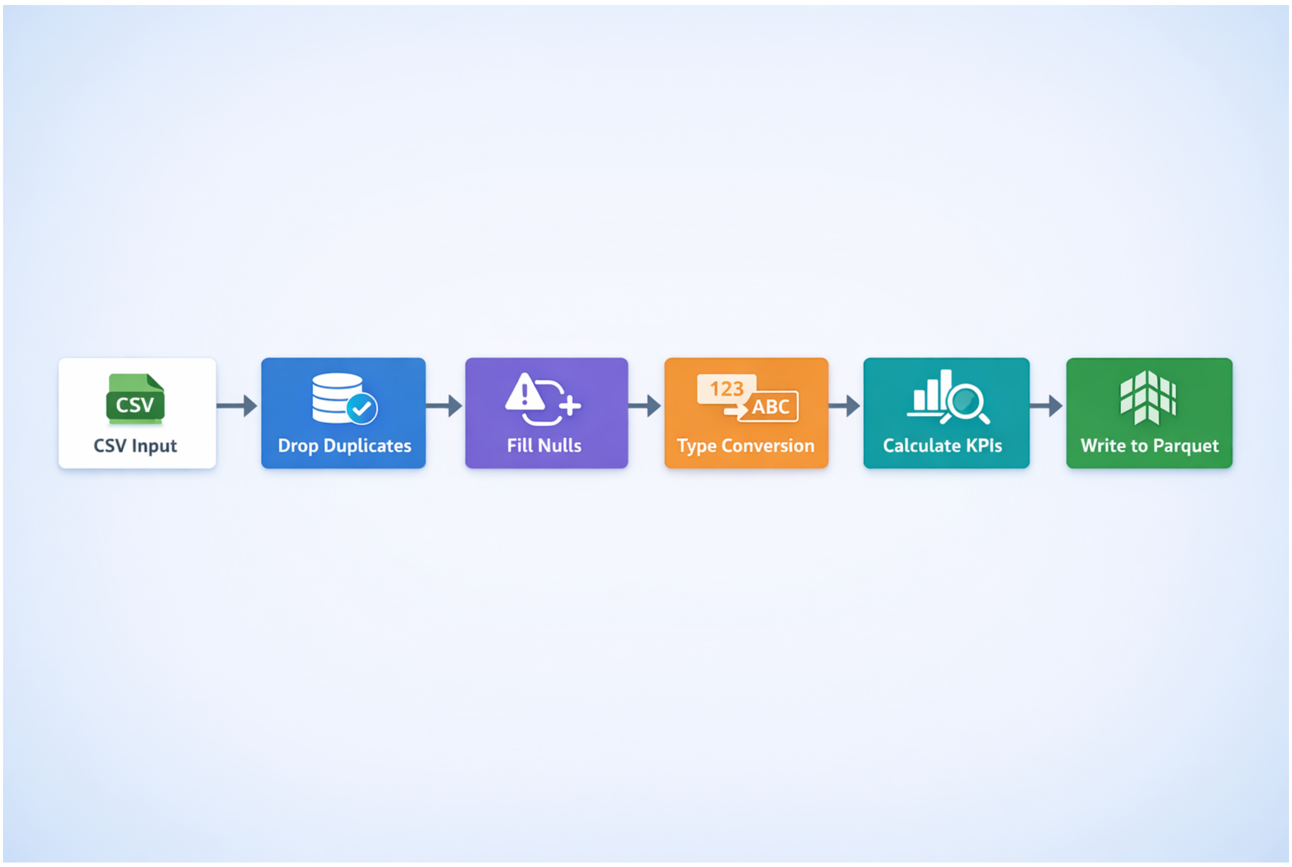
```
['part-00000-27b68ffa-060b-4c62-ac36-2f0fba99234c-c000.snappy.parquet', '.part-00000-27b68ffa-060b-4c62-ac36-
```



```
[29]
✓ 0s
parquet_df = spark.read.parquet("/content/drive/MyDrive/blend_work_python/output/amazon_kpis/region_sales")
parquet_df.show(5)
```

```
+-------------+--------------------+
|   ship-state|         Total_Sales|
+-------------+--------------------+
|  MAHARASHTRA|1.3334595140000008E7|
|    KARNATAKA|1.0481114370000001E7|
|    TELANGANA|   6916615.650000002|
|UTTAR PRADESH|          6816109.08|
|   TAMIL NADU|   6515650.110000002|
+-------------+--------------------+
only showing top 5 rows
```

DATA ETL PIPELINE DIAGRAM



PS:WE DID NOT USE HDFS AS OUR DATASET WAS SMALL AND DID NOT REQUIRE HDFS AND COULD EASILY RUN ON GOOGLE COLLAB USING LOCAL FILE STORAGE