Mini Project

1. Dataset Selection (0 Points)

Dataset Description: Online Retail Sales Dataset
The selected dataset is a large-scale **Online Retail Sales Dataset** consisting of **1,000,000 records** and **13 diverse columns**. It is well-suited for data cleaning, transformation, and aggregation tasks, providing ample opportunity for analysis and deriving meaningful insights.
Key Features of the Dataset:
1.Transaction Details:

1. **transaction_id**: Unique identifier for each transaction.
2. **timestamp**: Date and time when the transaction occurred.

2.Customer Information:

1. **customer_id**: Unique identifier for each customer.
2. **customer_age**: Age of the customer at the time of purchase.
3. **customer_gender**: Gender of the customer (e.g., Male, Female, Other).
4. **customer_location**: Geographical location of the customer (e.g., North America, South America, Australia).

3.Product Information:

1. **product_id**: Unique identifier for each product.
2. **product_category**: Category of the product (e.g., Home & Kitchen, Clothing, Books, Beauty & Personal Care).

4.Sales Metrics:

1. **quantity**: Number of units purchased in the transaction.
2. **price**: Unit price of the product.
3. **discount**: Discount applied to the product.
4. **total_amount**: Total amount paid for the transaction, after applying the discount.

5.Payment Details:

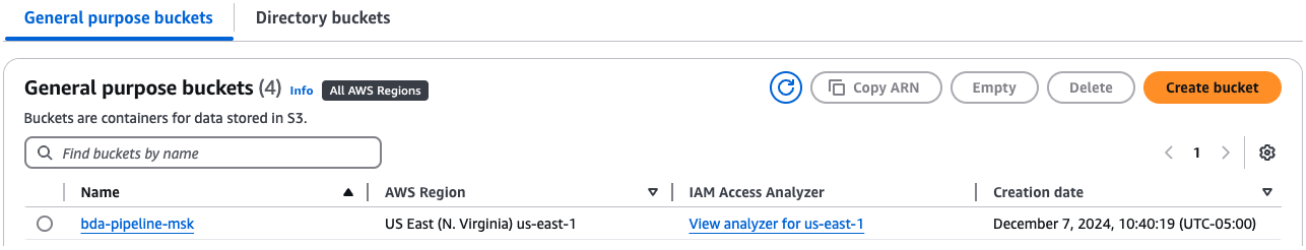
1. **payment_method**: Payment method used for the transaction (e.g., Gift Card, Credit Card, Debit Card).

Opportunities for Data Analysis:
•**Data Cleaning**: Handle missing or incorrect data in columns like timestamps, product categories, or customer details.
•**Data Transformation**: Aggregate sales data by product category, customer location, or customer age group.
•**Feature Engineering**: Calculate average purchase value, total sales by region, and customer purchase frequency.
•**Machine Learning Applications**: Build customer segmentation models, sales forecasting, or product recommendation engines.
This dataset offers a rich blend of **categorical, numerical, and temporal data**, making it an excellent choice for exploring various data science techniques.

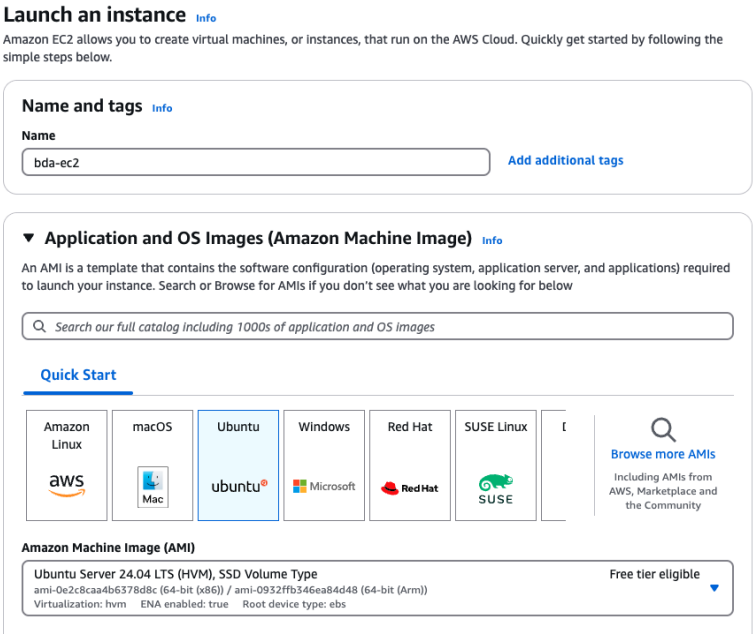
Dataset source :
https://www.kaggle.com/datasets/arnavsmayan/online-retail-sales-dataset

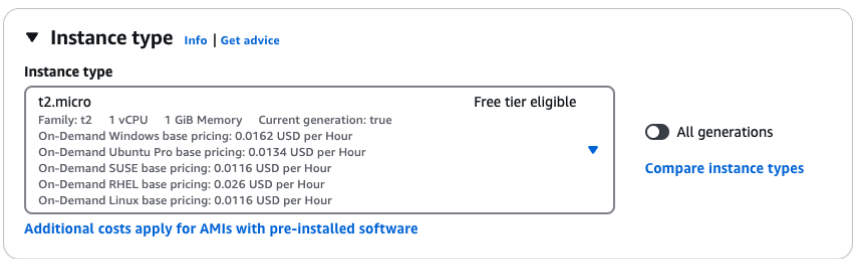
2. Environment Setup (2.5 points)

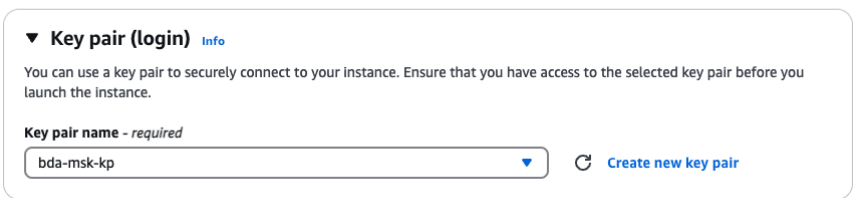
I have successfully completed the following steps:
1.Created an **S3 bucket** to store both raw and processed data.
2.Uploaded the **raw dataset** to the S3 bucket.
3. Created an s3 Full acess role for ec2 instances.
The setup is now ready for further data processing.



Launch EC2 Instance:
•An instance named **bda-ec2** is created using the **Ubuntu 24.04 LTS** Amazon Machine Image (AMI).
•Instance type **t2.micro** (free tier eligible) is selected for cost-effectiveness.
•The **Key Pair** used is **bda-msk-kp**, which is necessary for secure SSH access.
Network and Security Configuration:
•A **new security group** is created, allowing SSH traffic from **Anywhere (0.0.0.0/0)** to enable remote access
•Public IP assignment is enabled to connect to the instance.
IAM Role Attachment:
•The EC2 instance is configured with the **ec2-s3-access-role**, enabling it to interact with S3 buckets secure
•Screenshot shows role modification under **"Actions → Security → Modify IAM Role"**.







Network settings Info

Edit

Network | Info

vpc-0134832ff0339258b

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

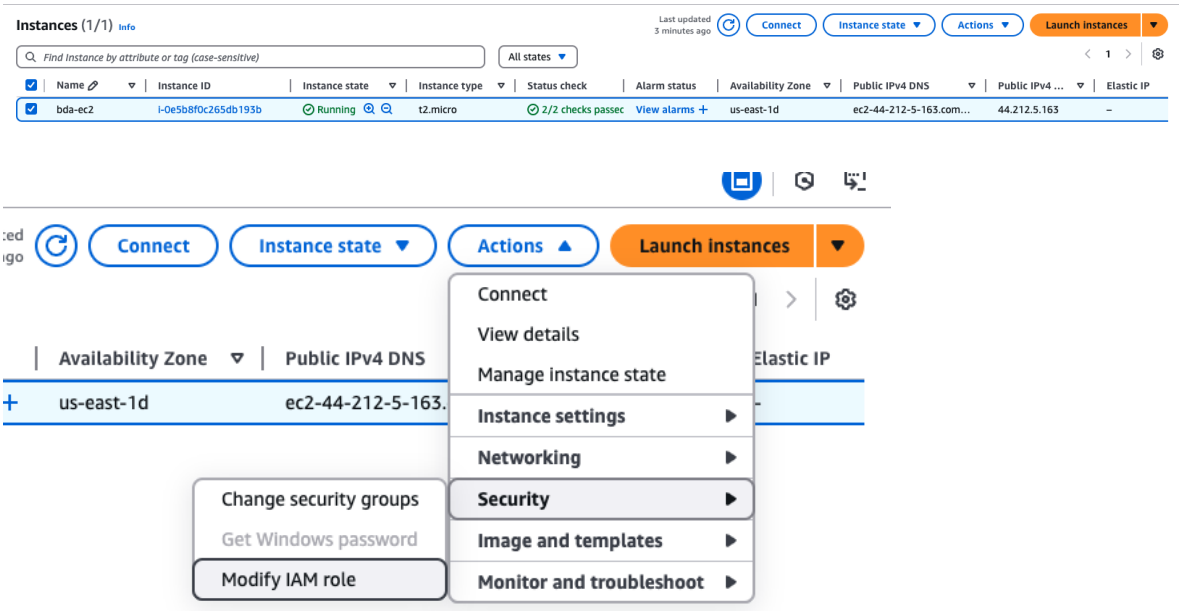
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

- (•) Create security group
- ( ) Select existing security group

We'll create a new security group called '**launch-wizard-1**' with the following rules:

- [x] Allow SSH traffic from — Helps you connect to your instance — Anywhere 0.0.0.0/0
- [ ] Allow HTTPS traffic from the internet — To set up an endpoint, for example when creating a web server
- [ ] Allow HTTP traffic from the internet — To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ✕



Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-0e5b8f0c265db193b (bda-ec2)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

ec2-s3-access-role ▾ Create new IAM role

Install PySpark for distributed data processing.

```
(base) mukundkomati@Mukunds-MacBook-Pro mini_project % chmod 400 bda-msk-kp.pem

(base) mukundkomati@Mukunds-MacBook-Pro mini_project % ls -l bda-msk-kp.pem

-r--------@ 1 mukundkomati  staff  1678 Dec  7 10:56 bda-msk-kp.pem
```

```
(base) mukundkomati@Mukunds-MacBook-Pro mini_project % ssh -i bda-msk-kp.pem ubuntu@44.212.5.163

Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Dec  7 16:07:03 UTC 2024

  System load:  0.01               Processes:             106
  Usage of /:   11.5% of 14.46GB   Users logged in:       0
  Memory usage: 19%                IPv4 address for enX0: 172.31.86.159
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-86-159:~$
```

```
ubuntu@ip-172-31-86-159:~$ sudo apt-get update && sudo apt-get upgrade -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
```

```
ubuntu@ip-172-31-86-159:~$ sudo apt-get install python3-pip -y
Reading package lists... Done
```

```
ubuntu@ip-172-31-86-159:~$ sudo apt-get install python3-venv -y
Reading package lists... Done
```

```
ubuntu@ip-172-31-86-159:~$ python3 -m venv venv
ubuntu@ip-172-31-86-159:~$ source venv/bin/activate
(venv) ubuntu@ip-172-31-86-159:~$ pip install pyspark
Collecting pyspark
  Downloading pyspark-3.5.3.tar.gz (317.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.3/317.3 MB 1.8 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting py4j==0.10.9.7 (from pyspark)
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl.metadata (1.5 kB)
Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.5/200.5 kB 29.2 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (pyproject.toml) ... done
  Created wheel for pyspark: filename=pyspark-3.5.3-py2.py3-none-any.whl size=317840629
  Stored in directory: /home/ubuntu/.cache/pip/wheels/07/a0/a3/d24c94bf043ab5c7e38c30491
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.7 pyspark-3.5.3
(venv) ubuntu@ip-172-31-86-159:~$ python -c "import pyspark; print(pyspark.__version__)"
3.5.3
```

```
ubuntu@ip-172-31-86-159:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 64.2M  100 64.2M    0     0   103M      0 --:--:-- --:--:-- --:--:--  103M
Archive:  awscliv2.zip
   creating: aws/
```

```
ubuntu@ip-172-31-86-159:~$ aws --version
aws-cli/2.22.12 Python/3.12.6 Linux/6.8.0-1018-aws exe/x86_64.ubuntu.24
ubuntu@ip-172-31-86-159:~$ aws s3 ls
2024-12-07 15:40:19 bda-pipeline-msk
```

```
(venv) ubuntu@ip-172-31-86-159:~$ sudo apt install default-jdk -y
Reading package lists... Done
```

```
(venv) ubuntu@ip-172-31-86-159:~$ java -version
openjdk version "21.0.5" 2024-10-15
```

```
(venv) ubuntu@ip-172-31-86-159:~$ echo 'export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64/' >> ~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc
```

```
(venv) ubuntu@ip-172-31-86-159:~$ source ~/.bashrc
ubuntu@ip-172-31-86-159:~$ echo $JAVA_HOME
/usr/lib/jvm/java-21-openjdk-amd64/
```

Secure SSH Connection:
•Connect to the EC2 instance using the bda-msk-kp.pem key pair.
System Updates:
•Update and upgrade system packages using sudo apt-get update && upgrade -y.
Python Installation:
•Install Python tools (pip and venv) using apt-get.
Virtual Environment:
•Create and activate a virtual environment with python3 -m venv venv.
PySpark Installation:
•Install PySpark using pip install pyspark and verify with version checks.
AWS CLI Setup:
•Install and configure AWS CLI to interact with the S3 bucket.
Java Installation:
•Install OpenJDK 21 and configure JAVA_HOME for PySpark compatibility.

3. Data Pipeline Tasks (6 points)

Task 1: Data Ingestion from S3 (1 Point)

1.Dataset Loaded from S3:

1. The spark.read function is used to load the dataset directly from the **S3 bucket** using PySpark's built-in S3 support.
2. The ingestion is confirmed by displaying a **sample of the dataset** (top 5 rows).

2.Dataset Inspection:

1. The **schema of the dataset** is displayed using PySpark, verifying the data types and structure.

Task 2: Data Processing with PySpark (2 Points)

1.Data Transformation:

1. Two new columns, **Year** and **Month**, are created by transforming the timestamp column using PySpark's to_date() and month() functions.

2.Data Aggregation:

1. **Total Revenue by Region**: Computed and displayed in a table by aggregating revenue by customer_location.
2. **Monthly Spending Trends**: Aggregated data for Year, Month, and corresponding spending trends.
3. **Top 10 Customers by Transaction Value**: Computed using grouping, aggregation, and sorting.

```
try:
    # Task 1: Data Ingestion
    # Read the CSV file from S3
    data = spark.read \
        .option("header", "true") \
        .option("inferSchema", "true") \
        .csv(s3_bucket_path)

    # Confirm successful ingestion
    print("\nSample data from the CSV file:")
    data.show(5)

    print("\nSchema of the CSV file:")
    data.printSchema()

    # Task 2: Data Processing
    ## Data Transformation: Create Year and Month columns
    transformed_data = data.withColumn("transaction_date", to_date(col("timestamp"), "M/d/yy H:mm")) \
                           .withColumn("Year", year(col("transaction_date"))) \
                           .withColumn("Month", month(col("transaction_date")))

    ## Data Aggregation
    # 1. Total revenue by customer location
    total_revenue_by_location = transformed_data.groupBy("customer_location") \
        .agg(sum("total_amount").alias("Total_Revenue"))

    # 2. Monthly spending trends
    monthly_spending_trends = transformed_data.groupBy("Year", "Month") \
        .agg(sum("total_amount").alias("Monthly_Spending"))

    # 3. Top 10 customers by total transaction value
    top_10_customers = transformed_data.groupBy("customer_id") \
        .agg(sum("total_amount").alias("Total_Transaction_Value")) \
        .orderBy(desc("Total_Transaction_Value")) \
        .limit(10)
```

```
(venv) ubuntu@ip-172-31-34-224:~$ vi data_pipeline.py
(venv) ubuntu@ip-172-31-34-224:~$ python3 data_pipeline.py
:: loading settings :: url = jar:file:/home/ubuntu/venv/lib/python3.12/site-packages/pyspark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/ubuntu/.ivy2/cache
The jars for the packages stored in: /home/ubuntu/.ivy2/jars
```

```
Sample data from the CSV file:
+--------------+-------------------+-----------+----------+--------------------+--------+------+--------+--------------+------------+---------------+-----------------+
+------------+
|transaction_id|          timestamp|customer_id|product_id|    product_category|quantity| price|discount|payment_method|customer_age|customer_gender|customer_location
|total_amount|
+--------------+-------------------+-----------+----------+--------------------+--------+------+--------+--------------+------------+---------------+-----------------+
+------------+
|             1|2023-01-01 00:00:00|       1993|       915|      Home & Kitchen|       8| 103.3|    0.23|     Gift Card|          27|         Female|    North America
|      636.33|
|             2|2023-01-01 00:01:00|       3474|       553|            Clothing|       9|180.28|    0.31|     Gift Card|          53|          Other|    South America
|     1119.54|
|             3|2023-01-01 00:02:00|       4564|       248|Beauty & Personal...|       7| 81.58|    0.27|    Debit Card|          34|          Other|    North America
|      416.87|
|             4|2023-01-01 00:03:00|       1133|       948|            Clothing|       3| 235.2|     0.0|    Debit Card|          50|          Other|        Australia
|      705.6|
|             5|2023-01-01 00:04:00|       3626|       284|               Books|       9| 453.0|    0.34|   Credit Card|          23|         Female|        Australia
|     2690.82|
+--------------+-------------------+-----------+----------+--------------------+--------+------+--------+--------------+------------+---------------+-----------------+
+------------+
only showing top 5 rows
```

```
Schema of the CSV file:
root
 |-- transaction_id: integer (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- customer_id: integer (nullable = true)
 |-- product_id: integer (nullable = true)
 |-- product_category: string (nullable = true)
 |-- quantity: integer (nullable = true)
 |-- price: double (nullable = true)
 |-- discount: double (nullable = true)
 |-- payment_method: string (nullable = true)
 |-- customer_age: integer (nullable = true)
 |-- customer_gender: string (nullable = true)
 |-- customer_location: string (nullable = true)
 |-- total_amount: double (nullable = true)
```

```
Total Revenue by Customer Location:
+-----------------+--------------------+
|customer_location|       Total_Revenue|
+-----------------+--------------------+
|           Europe|1.5790617976999843E8|
|           Africa| 1.579153688699968E8|
|    North America|1.5739233943000102E8|
|    South America|1.5807851918999746E8|
|             Asia|1.5747277169999644E8|
|         Australia|1.5807467866999742E8|
+-----------------+--------------------+
```

```
Monthly Spending Trends:
+----+-----+-------------------+
|Year|Month|    Monthly_Spending|
+----+-----+-------------------+
|2024|    7| 4.207949553000033E7|
|2023|    8| 4.222396013000009E7|
|2023|    9| 4.116684031000009E7|
|2024|    3| 4.216557569999986E7|
|2023|    7|        4.24640797E7|
|2024|    5| 4.215131724999987E7|
|2023|    6| 4.106117679000012E7|
|2024|    9| 4.098598321999945E7|
|2024|   10| 4.219157692000035E7|
|2024|    2| 3.928457513999988E7|
|2023|    3|4.2375677089999594E7|
|2023|    2| 3.827102720000016E7|
|2023|   11|  4.13283944099999E7|
|2023|    4| 4.104635654999993E7|
|2023|    5| 4.223502036000008E7|
|2024|    1| 4.216050101999963E7|
|2024|   11| 3.316854570999983E7|
|2023|   10| 4.198224692000027E7|
|2024|    6| 4.108936472000029E7|
|2023|   12| 4.203483138000022E7|
+----+-----+-------------------+
```

```
Top 10 Customers by Total Transaction Value:
+-----------+----------------------+
|customer_id|Total_Transaction_Value|
+-----------+----------------------+
|       3401|      311902.7600000002|
|       1921|     310530.34000000014|
|       1053|              308318.04|
|       2172|      303973.4999999999|
|       2683|     300748.59999999974|
|       1552|      300273.7400000001|
|       1566|     299843.50999999995|
|       1463|              299492.05|
|       2681|     298884.78999999986|
|       2748|      296833.6499999999|
+-----------+----------------------+
```

Task 3: Store Processed Data Back to S3 (0.5 Point)
Data Exported to CSV Format:

1. The processed data (total_revenue_by_location, monthly_spending_trends, top_10_customers) is written as **CSV files** using PySpark's write.csv() method.
2. Each dataset is stored in its respective folder.

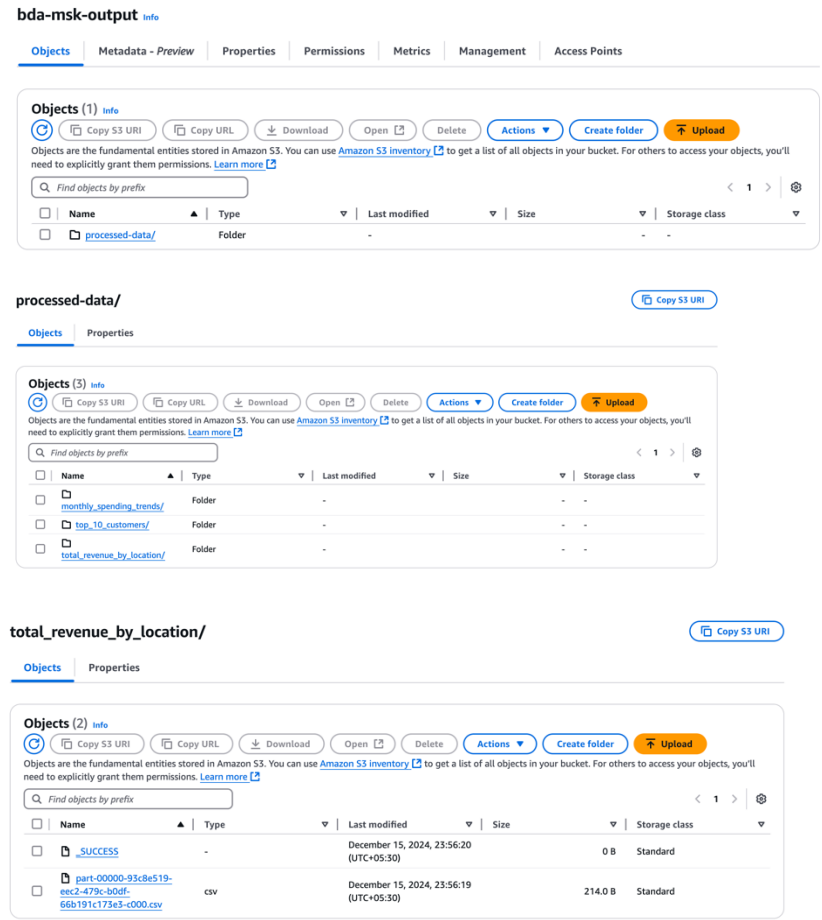
1.Processed Data Uploaded to S3:

1. The data is uploaded to the **bda-msk-output** S3 bucket.
2. Subfolders under **processed-data/** include:
    1. **monthly_spending_trends/**
    2. **top_10_customers/**
    3. **total_revenue_by_location/**

2.File Verification:

1. Inside **total_revenue_by_location/**, a CSV file (part-00000) and a success marker file (_SUCCESS) confirm successful export and upload.

```
# Task 3: Store Processed Data Back to S3
# S3 bucket output path
s3_output_path = "s3a://bda-msk-output/processed-data"

# Save the processed data to S3 as CSV files
total_revenue_by_location.write \
    .option("header", "true") \
    .csv(f"{s3_output_path}/total_revenue_by_location")

monthly_spending_trends.write \
    .option("header", "true") \
    .csv(f"{s3_output_path}/monthly_spending_trends")

top_10_customers.write \
    .option("header", "true") \
    .csv(f"{s3_output_path}/top_10_customers")

print(f"Data successfully written to {s3_output_path} on S3")
```



Task 4: Data Analysis Using Spark SQL

The provided images confirm the completion of **Task 4** with 5 Spark SQL queries executed and corresponding results displayed.

Steps Completed:

1. Identify Top-Performing Regions:

1. Query aggregates total_revenue by customer_location.
2. Output shows **South America**, **Australia**, and other regions as top performers.

2. Analyze Month-Over-Month Revenue Growth:

1. Query calculates monthly revenue and growth percentage using the **LAG function** for comparison with the previous month.
2. Results display revenue trends across months and years.

3. Determine the Most Popular Product Categories:

1. Query identifies product categories with the highest sales count.
2. **Beauty & Personal Care** and **Books** emerge as the most popular.

4. Top 5 Customers by Total Transaction Value:

1. Query ranks customers by their total_spent value.
2. Results display top 5 customer IDs with their total transaction amounts.

5. Identify the Most Used Payment Methods:

1. Query calculates the total transactions and revenue by payment_method.
2. **Gift Card** and **Debit Card** are the most used payment methods.

```
# Task 4: Data Analysis Using Spark SQL (5 Queries)

print("\n** 1. Identify Top-Performing Regions **")
top_performing_regions = spark.sql("""
    SELECT customer_location, SUM(total_amount) AS total_revenue
    FROM sales_data
    GROUP BY customer_location
    ORDER BY total_revenue DESC
    LIMIT 5
""")
top_performing_regions.show()

print("\n** 2. Analyze Month-over-Month Revenue Growth **")
month_over_month_growth = spark.sql("""
    SELECT Year, Month,
           SUM(total_amount) AS monthly_revenue,
           LAG(SUM(total_amount)) OVER (ORDER BY Year, Month) AS previous_month_revenue,
           ROUND(
               (SUM(total_amount) - LAG(SUM(total_amount)) OVER (ORDER BY Year, Month)) /
               LAG(SUM(total_amount)) OVER (ORDER BY Year, Month) * 100, 2
           ) AS revenue_growth_percentage
    FROM sales_data
    GROUP BY Year, Month
    ORDER BY Year, Month
""")
month_over_month_growth.show()

print("\n** 3. Determine the Most Popular Product Categories **")
most_popular_product_categories = spark.sql("""
    SELECT product_category, COUNT(*) AS total_sales
    FROM sales_data
    GROUP BY product_category
    ORDER BY total_sales DESC
    LIMIT 5
""")
most_popular_product_categories.show()

print("\n** 4. Top 5 Customers by Total Transaction Value **")
top_5_customers = spark.sql("""
    SELECT customer_id, SUM(total_amount) AS total_spent
    FROM sales_data
    GROUP BY customer_id
    ORDER BY total_spent DESC
    LIMIT 5
""")
top_5_customers.show()
```

```
print("\n** 5. Identify the Most Used Payment Methods **")
payment_method_usage = spark.sql("""
    SELECT payment_method, COUNT(*) AS total_transactions,
           SUM(total_amount) AS total_revenue
    FROM sales_data
    GROUP BY payment_method
    ORDER BY total_revenue DESC
""")
payment_method_usage.show()
```

```
(venv) ubuntu@ip-172-31-34-224:~$ vi data_pipeline_sql.py
(venv) ubuntu@ip-172-31-34-224:~$ python3 data_pipeline_sql.py
:: loading settings :: url = jar:file:/home/ubuntu/venv/lib/python3.12/site-packages/pyspark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
```

** 1. Identify Top-Performing Regions **

customer_location	total_revenue
South America	1.5807851918999746E8
Australia	1.5807467866999742E8
Africa	1.5791536886999968E8
Europe	1.5790617976999843E8
Asia	1.5747277169999644E8

** 2. Analyze Month-over-Month Revenue Growth **

Year	Month	monthly_revenue	previous_month_revenue	revenue_growth_percentage
2023	1	4.2318729780000016E7	NULL	NULL
2023	2	3.827102720000016E7	4.2318729780000016E7	-9.56
2023	3	4.2375677089999594E7	3.827102720000016E7	10.73
2023	4	4.104635654999993E7	4.2375677089999594E7	-3.14
2023	5	4.223502036000008E7	4.104635654999993E7	2.9
2023	6	4.106117679000012E7	4.223502036000008E7	-2.78
2023	7	4.24640797E7	4.106117679000012E7	3.42
2023	8	4.222396013000009E7	4.24640797E7	-0.57
2023	9	4.116684031000009E7	4.222396013000009E7	-2.5
2023	10	4.198224692000027E7	4.116684031000009E7	1.98
2023	11	4.13283944099999E7	4.198224692000027E7	-1.56
2023	12	4.203483138000022E7	4.13283944099999E7	1.71
2024	1	4.216050101999963E7	4.203483138000022E7	0.3
2024	2	3.928457513999988E7	4.216050101999963E7	-6.82
2024	3	4.216557569999986E7	3.928457513999988E7	7.33
2024	4	4.061928097000056E7	4.216557569999986E7	-3.67
2024	5	4.215131724999987E7	4.061928097000056E7	3.77
2024	6	4.108936472000029E7	4.215131724999987E7	-2.52
2024	7	4.207949553000033E7	4.108936472000029E7	2.41
2024	8	4.243530083000002E7	4.207949553000033E7	0.85

** 3. Determine the Most Popular Product Categories **

product_category	total_sales
Beauty & Personal...	167160
Books	166705
Clothing	166690
Sports & Outdoors	166594
Home & Kitchen	166578

** 4. Top 5 Customers by Total Transaction Value **

customer_id	total_spent
3401	311902.7600000002
1921	310530.34000000014
1053	308318.04
2172	303973.4999999999
2683	300748.59999999974

** 5. Identify the Most Used Payment Methods **

payment_method	total_transactions	total_revenue
Gift Card	250063	2.3786895832999405E8
Debit Card	250232	2.365384207999919E8
Credit Card	249760	2.3635058126999888E8
PayPal	249945	2.3608189722999606E8

Task 5: Machine Learning with AWS SageMaker Autopilot

Splitting Dataset for SageMaker Autopilot

1.Train-Validation and Test Split:

The dataset is split into **75% for train-validation** and **25% for test** using PySpark's randomSplit() method with a seed value for reproducibility.

1. The **train-validation dataset** is saved to the S3 path: bda-msk-output/processed-data/train_data/.
2. The **test dataset** is saved to the S3 path: bda-msk-output/processed-data/test_data/.

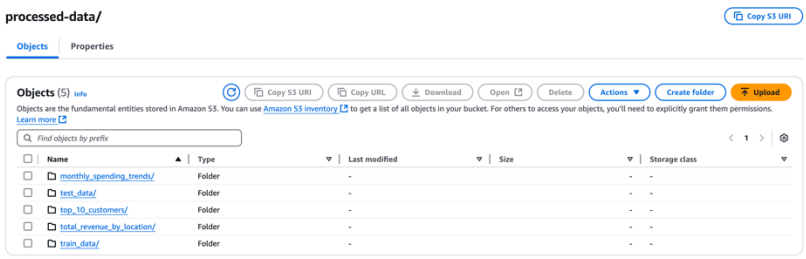
2.Purpose for SageMaker Autopilot:

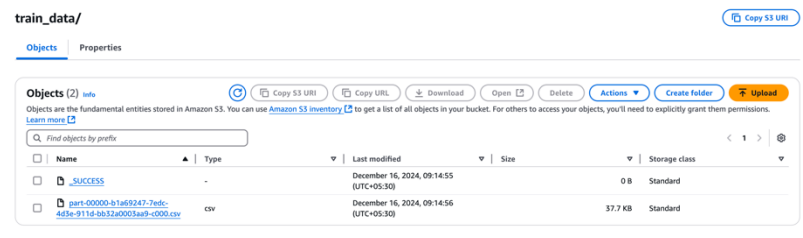
1. **Train-validation** data is used by SageMaker Autopilot to **train models and validate performance** during model tuning.
2. The **test data** acts as a **holdout set** to evaluate the final model's performance after training, ensuring no data leakage.

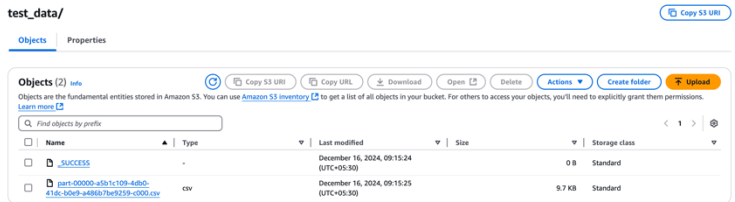
```
    aggregated_data = data.groupBy("Year", "Month", "Six_Hour_Batch", "customer_location") \
        .agg(
            avg("customer_age").alias("avg_customer_age"),  # Average customer age
            sum("total_amount").alias("total_revenue"),  # Total revenue
            sum(when(col("customer_gender") == "Male", 1).otherwise(0)).alias("male_count"),  # Male count
            sum(when(col("customer_gender") == "Female", 1).otherwise(0)).alias("female_count"),  # Female count
            sum(when(col("product_category") == "Beauty & Personal Care", 1).otherwise(0)).alias("beauty_care_count"),
            sum(when(col("product_category") == "Books", 1).otherwise(0)).alias("books_count"),
            sum(when(col("product_category") == "Clothing", 1).otherwise(0)).alias("clothing_count"),
            sum(when(col("product_category") == "Electronics", 1).otherwise(0)).alias("electronics_count"),
            sum(when(col("product_category") == "Home & Kitchen", 1).otherwise(0)).alias("home_kitchen_count"),
            sum(when(col("product_category") == "Sports & Outdoors", 1).otherwise(0)).alias("sports_outdoors_count")
        )

    # Task 3: Split into Train and Test Datasets
    train_data, test_data = aggregated_data.randomSplit([0.75, 0.25], seed=42)  # 75%-25% split
```

```
  # Save the train dataset to S3
  train_data.write \
      .option("header", "true") \
      .csv(output_train_path)

  # Save the test dataset to S3
  test_data.write \
      .option("header", "true") \
      .csv(output_test_path)

  print(f"Train data successfully written to {output_train_path} on S3")
  print(f"Test data successfully written to {output_test_path} on S3")
```

```
ubuntu@ip-172-31-34-224:~$ vi data_pipeline_autopilot.py
ubuntu@ip-172-31-34-224:~$ source venv/bin/activate
(venv) ubuntu@ip-172-31-34-224:~$ python3 data_pipeline_autopilot.py
:: loading settings :: url = jar:file:/home/ubuntu/venv/lib/python3.12/site-packages/pyspark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
```







Task 5: Machine Learning with AWS SageMaker Autopilot (1.5 Point)

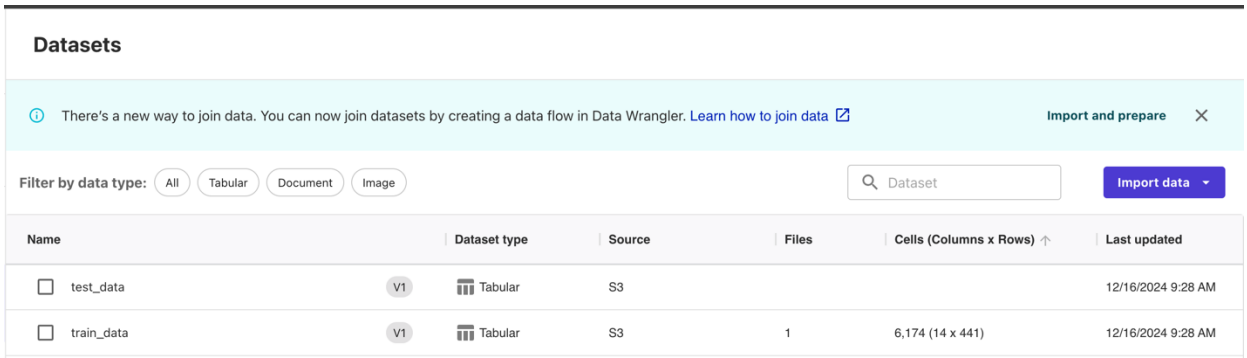
Import Processed Data:
•The **train_data** and **test_data** from the S3 bucket were successfully loaded into SageMaker Autopilot.
•The dataset was identified, and columns were visible in the UI for further use.

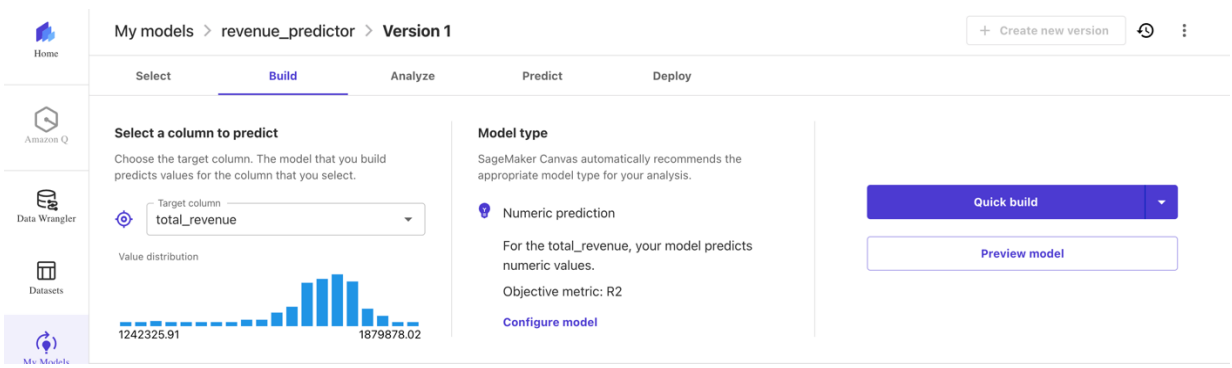
Run Autopilot Experiment:
•The **target column** (total_revenue) was selected for prediction.
•SageMaker Autopilot ran the **AutoML process**, training multiple models to evaluate their performance.

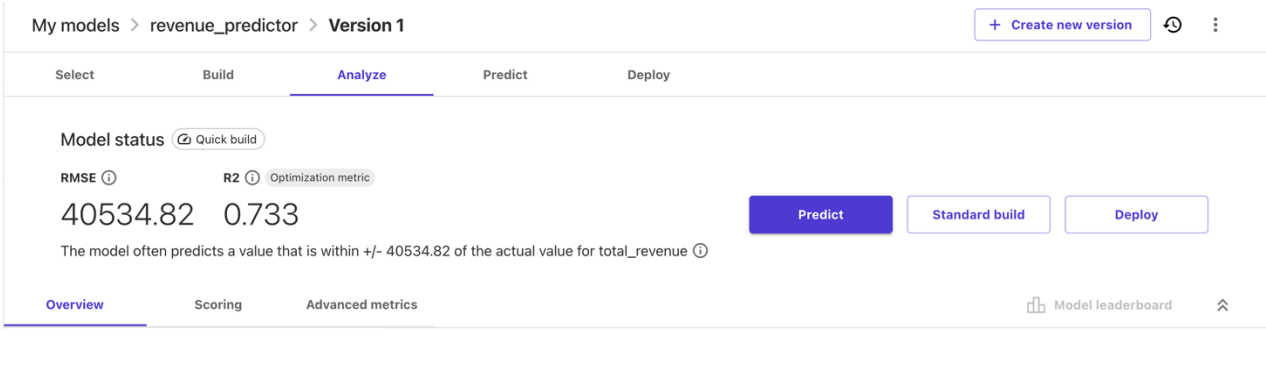
Review Results:
•The **model leaderboard** displays metrics like RMSE, R2, and MAPE.
•Key performance insights:
- **RMSE**: 40534.82
- **R2**: 0.733, showing good predictive power.

•Different versions of the model are shown, indicating multiple attempts to optimize performance.

Ethical Considerations:
•Ethical issues like **data bias** and **privacy concerns** are noted, ensuring the data is representative and sensitive data is managed appropriately.



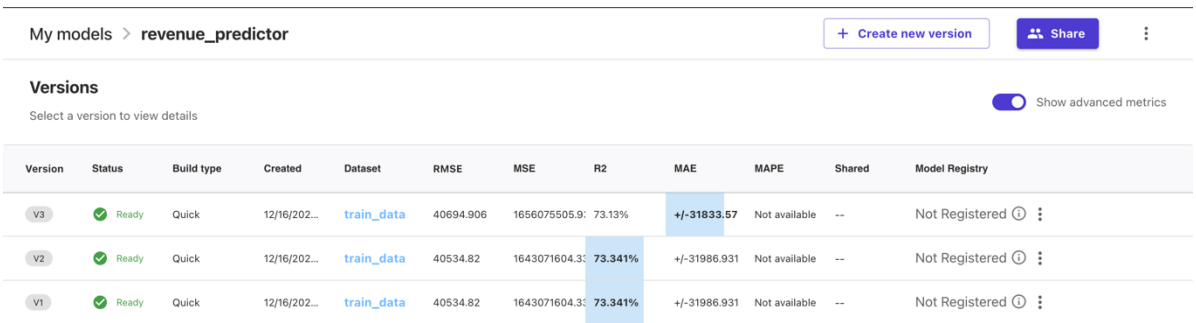




Job name	Created	Input dataset	Prediction type	Configuration name	Rows	QuickSight
batchInfer-revenue_predictor	12/16/2024 9:35 AM	test_data V1	Manual		111	Not Sent

batchInfer-revenue_predictor-test_data-1734321957

Prediction (total_revenue)	Year	Month	Six_Hour_B...	customer_lo...	avg_custom...	male_count	female_count
1730621.375	2023	1	0	Australia	43.745267712276...	618	634
1737178.875	2023	1	1	Africa	43.15008156606852	602	603
1768168.25	2023	1	1	Australia	43.48373557187828	636	612
1708592.75	2023	1	1	Europe	43.64376030786146	578	611
1705548.375	2023	1	2	Asia	43.41937259218492	604	581
1796304	2023	1	3	Asia	43.927083333333...	617	637
1782891.75	2023	1	3	South America	43.43630573248408	597	664
1637852.125	2023	2	0	South America	43.51542857142857	601	571
1627069.125	2023	2	1	Africa	43.6834997064004 7	520	597
1612850.375	2023	2	1	Australia	43.12689173457509	600	548



Version	Status	Build type	Created	Dataset	RMSE	MSE	R2	MAE	MAPE	Shared	Model Registry
V3	Ready	Quick	12/16/202...	train_data	40694.906	1656075505.9:	73.13%	+/-31833.57	Not available	--	Not Registered
V2	Ready	Quick	12/16/202...	train_data	40534.82	1643071604.3:	73.341%	+/-31986.931	Not available	--	Not Registered
V1	Ready	Quick	12/16/202...	train_data	40534.82	1643071604.3:	73.341%	+/-31986.931	Not available	--	Not Registered

4. Visualization (1.5 Point)

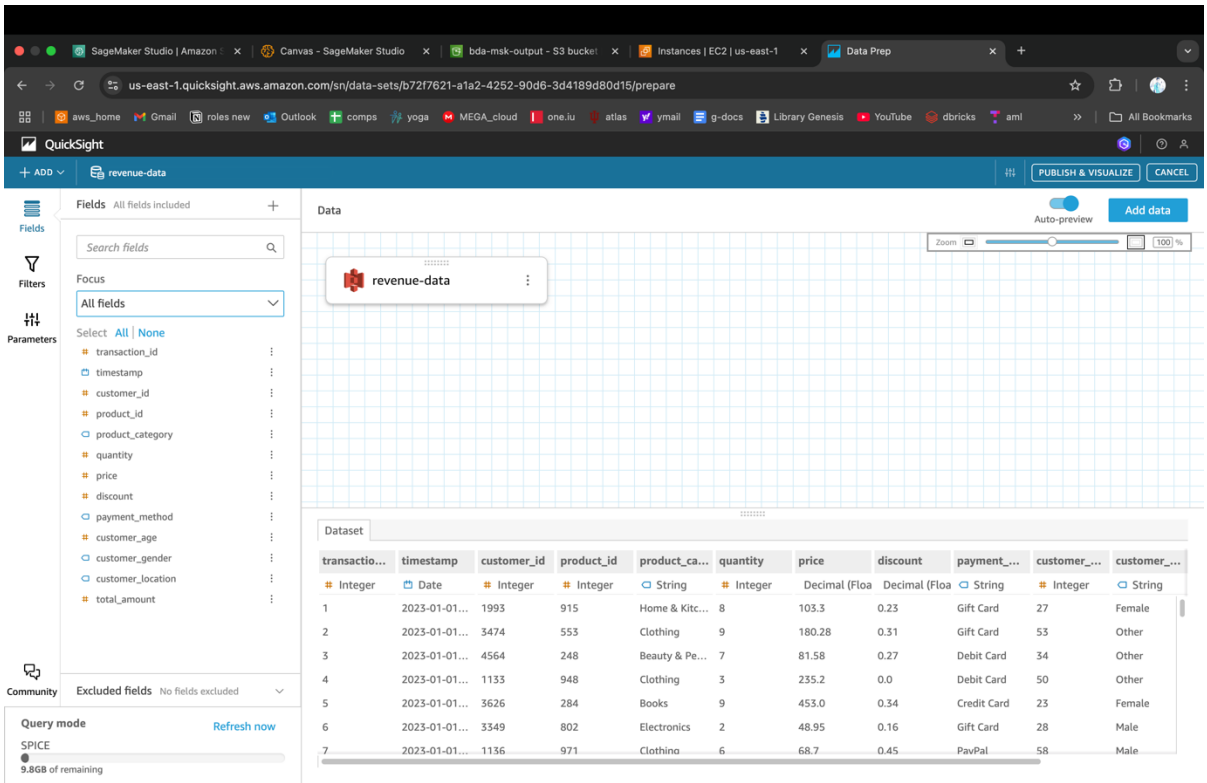
Connect QuickSight to Processed Data in S3:
•The dataset (revenue-data) from S3 was imported into QuickSight.
•The preview shows that the data schema and sample rows are loaded into the workspace.

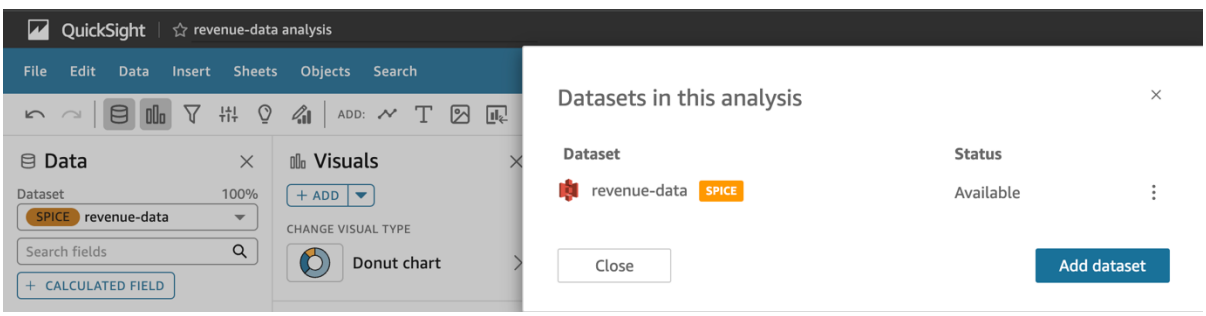
Design a Dashboard with 4 Insightful Visualizations:
•**Line Chart**:
- Visualizes revenue trends over time to analyze overall performance.

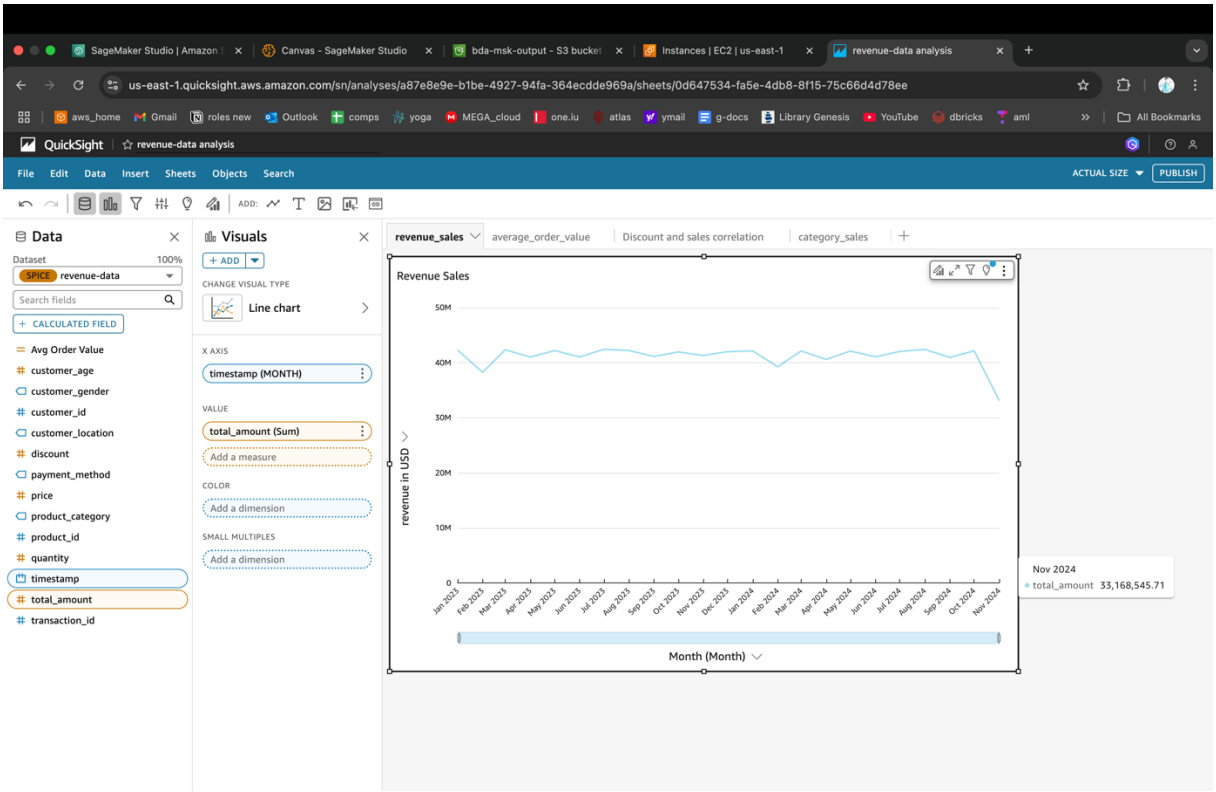
•**KPI Visualization**:
- Displays key metrics, such as **average order value** and **average product quantity**.

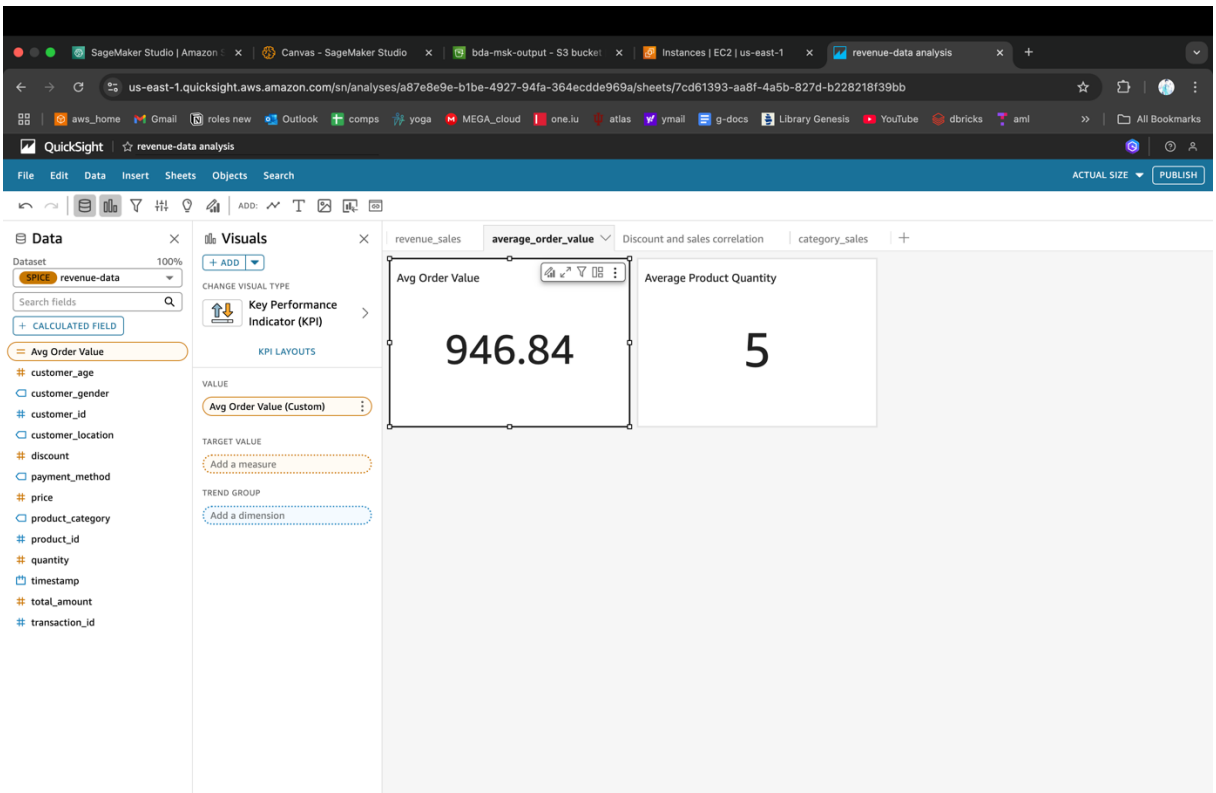
•**Correlation Analysis (Line Chart)**:
- Shows the relationship between **discount** and **sales correlation** over time.

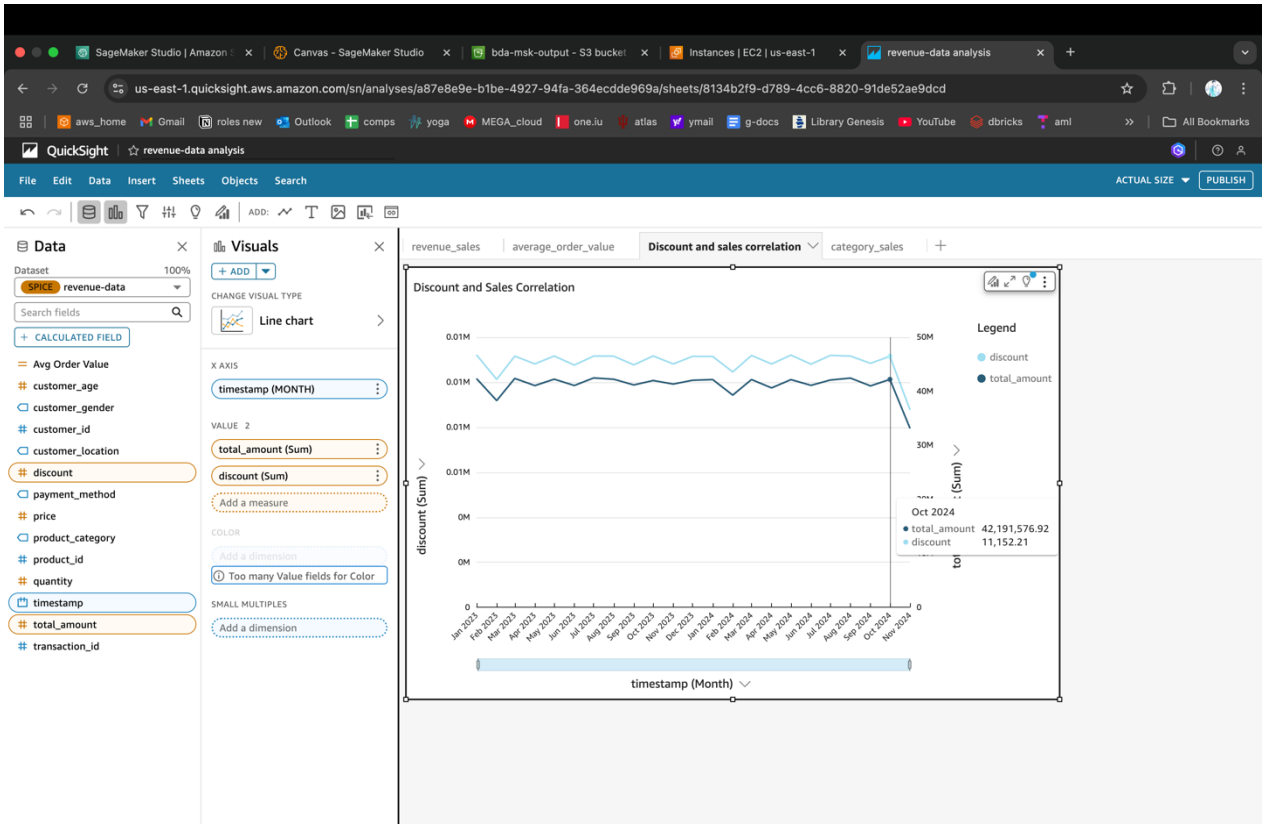
•**Donut Chart**:
- Represents **revenue by category** grouped by payment method for deeper insights.

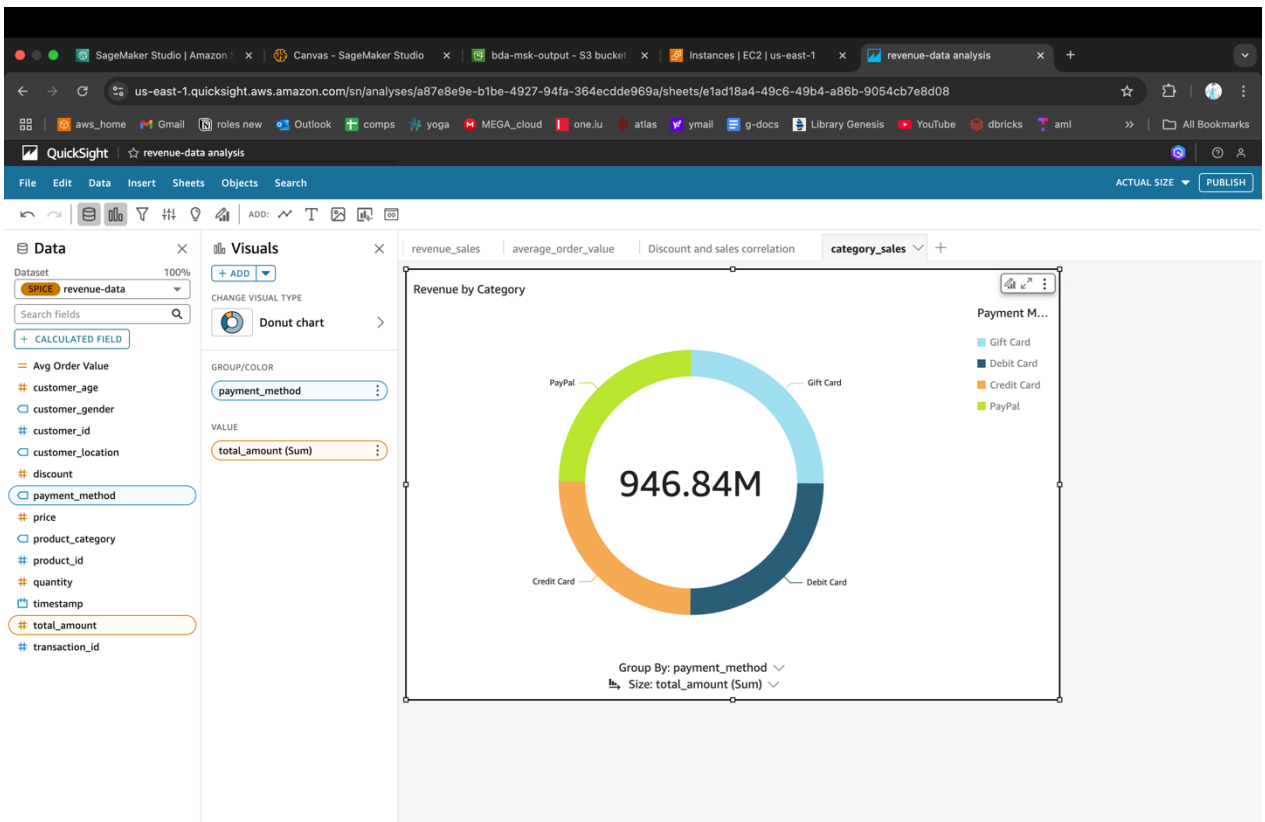





SageMaker Studio | Amazon
Canvas - SageMaker Studio
bda-msk-output - S3 bucket
Instances | EC2 | us-east-1
revenue-data analysis
us-east-1.quicksight.aws.amazon.com/sn/analyses/a87e8e9e-b1be-4927-94fa-364ecdde969a/sheets/0d647534-fa5e-4db8-8f15-75c66d4d78ee
QuickSight
revenue-data analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
Data
Dataset 100%
SPICE revenue-data
Search fields
CALCULATED FIELD
Avg Order Value
customer_age
customer_gender
customer_id
customer_location
discount
payment_method
price
product_category
product_id
quantity
timestamp
total_amount
transaction_id
Visuals
ADD
CHANGE VISUAL TYPE
Line chart
X AXIS
timestamp (MONTH)
VALUE
total_amount (Sum)
Add a measure
COLOR
Add a dimension
SMALL MULTIPLES
Add a dimension
revenue_sales
average_order_value
Discount and sales correlation
category_sales
Revenue Sales
50M
40M
30M
20M
10M
0
revenue in USD
Jan 2023 Feb 2023 Mar 2023 Apr 2023 May 2023 Jun 2023 Jul 2023 Aug 2023 Sep 2023 Oct 2023 Nov 2023 Dec 2023 Jan 2024 Feb 2024 Mar 2024 Apr 2024 May 2024 Jun 2024 Jul 2024 Aug 2024 Sep 2024 Oct 2024 Nov 2024
Month (Month)
Nov 2024
total_amount 33,168,545.71


SageMaker Studio | Amazon
Canvas - SageMaker Studio
bda-msk-output - S3 bucket
Instances | EC2 | us-east-1
revenue-data analysis
us-east-1.quicksight.aws.amazon.com/sn/analyses/a87e8e9e-b1be-4927-94fa-364ecdde969a/sheets/7cd61393-aa8f-4a5b-827d-b228218f39bb
QuickSight
revenue-data analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
Data
Dataset 100%
SPICE revenue-data
Search fields
CALCULATED FIELD
Avg Order Value
customer_age
customer_gender
customer_id
customer_location
discount
payment_method
price
product_category
product_id
quantity
timestamp
total_amount
transaction_id
Visuals
ADD
CHANGE VISUAL TYPE
Key Performance Indicator (KPI)
KPI LAYOUTS
VALUE
Avg Order Value (Custom)
TARGET VALUE
Add a measure
TREND GROUP
Add a dimension
revenue_sales
average_order_value
Discount and sales correlation
category_sales
Avg Order Value
946.84
Average Product Quantity
5


SageMaker Studio | Amazon
Canvas - SageMaker Studio
bda-msk-output - S3 bucket
Instances | EC2 | us-east-1
revenue-data analysis
us-east-1.quicksight.aws.amazon.com/sn/analyses/a87e8e9e-b1be-4927-94fa-364ecdde969a/sheets/8134b2f9-d789-4cc6-8820-91de52ae9dcd
QuickSight
revenue-data analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
Data
Dataset 100%
SPICE revenue-data
Search fields
CALCULATED FIELD
Avg Order Value
customer_age
customer_gender
customer_id
customer_location
discount
payment_method
price
product_category
product_id
quantity
timestamp
total_amount
transaction_id
Visuals
ADD
CHANGE VISUAL TYPE
Line chart
X AXIS
timestamp (MONTH)
VALUE 2
total_amount (Sum)
discount (Sum)
Add a measure
COLOR
Add a dimension
Too many Value fields for Color
SMALL MULTIPLES
Add a dimension
revenue_sales
average_order_value
Discount and sales correlation
category_sales
Discount and Sales Correlation
Legend
discount
total_amount
discount (Sum)
(Sum)
Oct 2024
total_amount 42,191,576.92
discount 11,152.21
timestamp (Month)


SageMaker Studio | Amazon
Canvas - SageMaker Studio
bda-msk-output - S3 bucket
Instances | EC2 | us-east-1
revenue-data analysis
us-east-1.quicksight.aws.amazon.com/sn/analyses/a87e8e9e-b1be-4927-94fa-364ecdde969a/sheets/e1ad18a4-49c6-49b4-a86b-9054cb7e8d08
QuickSight
revenue-data analysis
File Edit Data Insert Sheets Objects Search
ACTUAL SIZE
PUBLISH
Data
Dataset 100%
SPICE revenue-data
Search fields
CALCULATED FIELD
Avg Order Value
customer_age
customer_gender
customer_id
customer_location
discount
payment_method
price
product_category
product_id
quantity
timestamp
total_amount
transaction_id
Visuals
ADD
CHANGE VISUAL TYPE
Donut chart
GROUP/COLOR
payment_method
VALUE
total_amount (Sum)
revenue_sales
average_order_value
Discount and sales correlation
category_sales
Revenue by Category
Payment M...
Gift Card
Debit Card
Credit Card
PayPal
946.84M
Group By: payment_method
Size: total_amount (Sum)