Big Data - Case Study

INDEX

Introduction

In the dynamic landscape of modern business, the influx of vast and diverse datasets has propelled the significance of data analytics. This project embarks on a journey to conduct insightful analyses using three key technologies—MySQL, Hive, and Sqoop. As we navigate through the intricacies of data management and analysis, we'll explore the power of relational databases, the scalability of Hadoop-based frameworks, and the seamless data transfer capabilities offered by Sqoop.

Our project focuses on harnessing the strengths of MySQL, a reliable relational database system, for structured data analysis. MySQL provides a solid foundation for managing and querying structured data, offering stability in scenarios where relational integrity is crucial.

To address the need for processing semi-structured and unstructured data, we incorporate Hive into our toolkit. Hive, built on top of Hadoop, offers a SQL-like query language and data warehousing capabilities, enabling scalable analysis of large datasets.

Additionally, Sqoop plays a pivotal role in our project, facilitating the seamless transfer of data between MySQL and Hadoop. This tool acts as a bridge, ensuring that data stored in relational databases can be efficiently integrated into the Hadoop ecosystem, allowing for a more comprehensive and diverse range of analyses.

Our primary objectives include the integration of data from MySQL to Hadoop using Sqoop, leveraging MySQL for structured data analysis, and harnessing the scalability of Hive for processing large volumes of semi-structured and unstructured data.

In the pursuit of meaningful insights, the amalgamation of MySQL, Hive, and Sqoop underscores the adaptability and potency of contemporary data analytics tools. This project seeks to navigate the complexities of big data, empowering businesses with actionable insights and fostering a data-driven approach to decision-making in an era defined by information abundance.

Description of the Dataset

Database Description

The retail database is a valuable tool for businesses as it provides a structured and organized way to store and manage essential information related to their operations. By categorizing data into tables such as Departments, Categories, Products, Orders, Order Items, and Customers, the database enables businesses to track and analyse various aspects of their retail activities. For instance, businesses can use the database to monitor sales trends, manage inventory, understand customer behaviour, and track order fulfilment. The relational structure of the database allows for efficient querying, reporting, and analysis, providing valuable insights that can inform strategic decision-making. Moreover, the database facilitates the management of customer relationships by storing detailed customer information. Overall, the retail database is a powerful tool that aids businesses in enhancing operational efficiency, optimizing inventory, and improving customer satisfaction through informed decision-making.

1. Departments:

- Attributes:
  - `department_id` (Primary Key): Unique identifier for each department.
  - `department_name`: Name of the department.

2. Categories:

- Attributes:
  - `category_id` (Primary Key): Unique identifier for each category.
  - `category_name`: Name of the category.
  - `category_department_id` (Foreign Key referencing `department_id` in Departments table): Indicates the department to which the category belongs.

3. Products:

- Attributes:
  - `product_id` (Primary Key): Unique identifier for each product.

- `product_category_id` (Foreign Key referencing `category_id` in Categories table): Indicates the category to which the product belongs.

- `product_name`: Name of the product.

- `product_description`: Description of the product.

- `product_price`: Price of the product.

- `product_image`: Image or reference to the product image.

4. Order Items:

- Attributes:

- `order_item_id` (Primary Key): Unique identifier for each order item.

- `order_item_product_id` (Foreign Key referencing `product_id` in Products table): Indicates the product included in the order item.

- `order_item_order_id` (Foreign Key referencing `order_id` in Orders table): Indicates the order to which the order item belongs.

- `order_item_quantity`: Quantity of the product in the order item.

- `order_item_subtotal`: Subtotal for the order item (quantity * product price).

- `order_item_product_price`: Price of the product at the time of the order.

5. Orders:

- Attributes:

- `order_id` (Primary Key): Unique identifier for each order.

- `order_date`: Date when the order was placed.

- `order_status`: Status of the order, which can be "COMPLETE," "CLOSED," or "PAYMENT_PENDING."

6. Customers:

- Attributes:

- `customer_id` (Primary Key): Unique identifier for each customer.

- `customer_fname`: First name of the customer.
- `customer_lname`: Last name of the customer.
- `customer_email`: Email address of the customer.
- `customer_password`: Password associated with the customer's account.
- `customer_street`: Street address of the customer.
- `customer_city`: City of the customer.
- `customer_state`: State or region of the customer.
- `customer_zipcode`: Zip code of the customer.

Each table is designed to store specific information related to the retail business, and the relationships between tables are established through foreign keys. For example, the `category_department_id` in the Categories table links to the `department_id` in the Departments table, creating a relationship between departments and categories. Similarly, foreign keys in the Order Items table link to products and orders, connecting the order items to the products and orders they belong to.

Project Scope

The project's extensive scope involves a thorough investigation into customer purchasing patterns, utilizing data analytics and insights generation within retail establishments. Leveraging MySQL, Hive, and Sqoop in the Hadoop ecosystem, the project aims to analyze and extract valuable insights from diverse data sources to uncover nuanced aspects of customer buying behaviors. The primary focus is on gaining a comprehensive understanding of the intricate dynamics within retail stores, encompassing different departments and preferred payment methods. The overarching objective is to employ advanced, data-driven methods to improve decision-making processes, providing valuable and actionable insights to optimize various facets of retail operations. Through the effective application of these technologies, the ultimate goal is to enhance customer experiences, ensuring a seamless and personalized shopping journey.

Goals

1. Customer Behaviour Analysis: Gain deep insights into customer buying habits through comprehensive data analytics, identifying patterns and trends.

2. Data Integration and Processing: Utilize MySQL, Hive, and Sqoop in the Hadoop ecosystem to integrate and process data from diverse sources, ensuring a holistic view of customer interactions.

3. Retail Store Dynamics Understanding: Investigate and understand the dynamics within retail stores, including the analysis of various departments and the identification of preferred payment methods.

4. Insights Generation: Derive meaningful insights from the analysed data to provide a clearer understanding of customer purchasing behaviours and retail store dynamics.

5. Decision-Making Enhancement: Implement data-driven approaches to enhance decision-making processes, offering valuable perspectives for optimizing retail operations.

6. Operational Optimization: Optimize various facets of retail operations based on insights gained, improving efficiency, inventory management, and overall operational effectiveness.

7. Technology Utilization: Effectively leverage MySQL, Hive, and Sqoop to harness the power of the Hadoop ecosystem for efficient data analysis and insights generation.

8. Customer Experience Enhancement: Apply insights to enhance customer experiences by tailoring services, promotions, and interactions to meet specific customer preferences and needs.

Tools and Working Environment

1. MySQL:

- Description: MySQL is an open-source relational database management system (RDBMS) known for its reliability and ease of use. It uses a structured query language (SQL) to manage and manipulate relational databases.

- Working Environment: In the context of the project, MySQL serves as the primary tool for structured data analysis. It provides a familiar SQL interface for querying and managing structured datasets.

2. Hive:

- Description: Hive is a data warehousing and SQL-like query language built on top of Hadoop. It facilitates the processing and analysis of large volumes of data, especially in a distributed computing environment.

- Working Environment: Hive is employed for scalable processing of semi-structured and unstructured data. It allows users to write queries in a SQL-like language (HiveQL) and translates them into MapReduce jobs, enabling the analysis of diverse datasets stored in Hadoop Distributed File System (HDFS).

3. Sqoop:

- Description: Sqoop is a data transfer tool designed to efficiently move data between Apache Hadoop and relational databases. It supports the import and export of data, bridging the gap between Hadoop's distributed environment and traditional databases.

- Working Environment: Sqoop plays a crucial role in the project by facilitating the seamless integration of data between MySQL and Hadoop. It allows for the import of data from MySQL into Hadoop, enabling a more comprehensive range of analyses across different types of data.

4. Hadoop Ecosystem:

- Description: Hadoop is an open-source framework for distributed storage and processing of large datasets. It includes components like HDFS (Hadoop Distributed File System) for storage and MapReduce for parallel processing.

- Working Environment: The Hadoop ecosystem serves as the underlying framework for handling large-scale data processing. It provides the infrastructure needed for storing and analyzing data across a distributed network.

Performing Analysis on MySQL

Using database retail_db:

```
[cloudera@quickstart Desktop]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use retail_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

Displaying names of tables in retail_db:

```
Database changed
mysql> show tables;
+----------------------+
| Tables_in_retail_db |
+----------------------+
| categories           |
| customers            |
| departments          |
| order_items          |
| orders               |
| products             |
+----------------------+
6 rows in set (0.00 sec)
```

Displaying tables:

```
mysql> select * from order_items limit 5;
+---------------+--------------------+-----------------------+---------------------+-----------------------+--------------------------+
| order_item_id | order_item_order_id | order_item_product_id | order_item_quantity | order_item_subtotal | order_item_product_price |
+---------------+--------------------+-----------------------+---------------------+-----------------------+--------------------------+
|             1 |                  1 |                   957 |                   1 |                299.98 |                   299.98 |
|             2 |                  2 |                  1073 |                   1 |                199.99 |                   199.99 |
|             3 |                  2 |                   502 |                   5 |                   250 |                       50 |
|             4 |                  2 |                   403 |                   1 |                129.99 |                   129.99 |
|             5 |                  4 |                   897 |                   2 |                 49.98 |                    24.99 |
+---------------+--------------------+-----------------------+---------------------+-----------------------+--------------------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from orders limit 5;
+----------+---------------------+-------------------+--------------+
| order_id | order_date          | order_customer_id | order_status |
+----------+---------------------+-------------------+--------------+
|        1 | 2013-07-25 00:00:00 |             11599 | CLOSED       |
|        2 | 2013-07-25 00:00:00 |               256 | CLOSED       |
|        3 | 2013-07-25 00:00:00 |             12111 | COMPLETE     |
|        4 | 2013-07-25 00:00:00 |              8827 | CLOSED       |
|        5 | 2013-07-25 00:00:00 |             11318 | COMPLETE     |
+----------+---------------------+-------------------+--------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from categories limit 5;
+-------------+-------------------------+---------------------+
| category_id | category_department_id | category_name       |
+-------------+-------------------------+---------------------+
|           1 |                       2 | Football            |
|           2 |                       2 | Soccer              |
|           3 |                       2 | Baseball & Softball |
|           4 |                       2 | Basketball          |
|           5 |                       2 | Lacrosse            |
+-------------+-------------------------+---------------------+
5 rows in set (0.00 sec)
```

Performing SQL queries on the table:

```
mysql> select count(order_status) from orders where order_status = 'PENDING_PAYMENT';
+---------------------+
| count(order_status) |
+---------------------+
|               15030 |
+---------------------+
1 row in set (0.07 sec)
```

```
mysql> select * from customers limit 5;
+-------------+---------------+---------------+----------------+------------------+------------------------+---------------+----------------+-----------------+
| customer_id | customer_fname | customer_lname | customer_email | customer_password | customer_street        | customer_city | customer_state | customer_zipcode |
+-------------+---------------+---------------+----------------+------------------+------------------------+---------------+----------------+-----------------+
|           1 | Richard       | Hernandez     | XXXXXXXXX      | XXXXXXXXX        | 6303 Heather Plaza     | Brownsville   | TX             | 78521           |
|           2 | Mary          | Barrett       | XXXXXXXXX      | XXXXXXXXX        | 9526 Noble Embers Ridge | Littleton     | CO             | 80126           |
|           3 | Ann           | Smith         | XXXXXXXXX      | XXXXXXXXX        | 3422 Blue Pioneer Bend | Caguas        | PR             | 00725           |
|           4 | Mary          | Jones         | XXXXXXXXX      | XXXXXXXXX        | 8324 Little Common     | San Marcos    | CA             | 92069           |
|           5 | Robert        | Hudson        | XXXXXXXXX      | XXXXXXXXX        | 10 Crystal River Mall  | Caguas        | PR             | 00725           |
+-------------+---------------+---------------+----------------+------------------+------------------------+---------------+----------------+-----------------+
5 rows in set (0.00 sec)
```

```
mysql> select count(*) from customers  where customer_city = 'Brownsville';
+----------+
| count(*) |
+----------+
|       16 |
+----------+
1 row in set (0.03 sec)
```

```
mysql> select count(*) from order_items  where order_item_product_price>100;
+----------+
| count(*) |
+----------+
|    70388 |
+----------+
1 row in set (0.13 sec)
```

```
mysql> select count(*) from products where product_price > 100;
+----------+
| count(*) |
+----------+
|      484 |
+----------+
1 row in set (0.02 sec)
```

```
mysql> select count(*)  from products where product_price > 200 and product_price < 800;
+----------+
| count(*) |
+----------+
|      179 |
+----------+
1 row in set (0.00 sec)
```

-

Performing Analysis on Hive

Loading the dataset from MySQL into Hive:

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost/retail_db --username root --password cloudera --table order_items --hive-import --hive-table order_i --create-hive-table --hive-overwrite;
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/11/21 02:02:08 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
23/11/21 02:02:08 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
23/11/21 02:02:08 INFO tool.BaseSqoopTool: Using Hive-specific delimiters for output. You can override
23/11/21 02:02:08 INFO tool.BaseSqoopTool: delimiters with --fields-terminated-by, etc.
23/11/21 02:02:09 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
23/11/21 02:02:09 INFO tool.CodeGenTool: Beginning code generation
23/11/21 02:02:10 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `order_items` AS t LIMIT 1
23/11/21 02:02:11 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `order_items` AS t LIMIT 1
23/11/21 02:02:11 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-cloudera/compile/a8236c6f078f411aa0ebbafbe3e3f46f/order_items.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
23/11/21 02:02:20 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/a8236c6f078f411aa0ebbafbe3e3f46f/order_items.jar
23/11/21 02:02:20 WARN manager.MySQLManager: It looks like you are importing from mysql.
23/11/21 02:02:20 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
23/11/21 02:02:20 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
23/11/21 02:02:20 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
23/11/21 02:02:20 INFO mapreduce.ImportJobBase: Beginning import of order_items
23/11/21 02:02:20 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
23/11/21 02:02:22 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
23/11/21 02:02:25 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
23/11/21 02:02:25 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/11/21 02:02:28 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
23/11/21 02:02:28 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
23/11/21 02:02:28 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
23/11/21 02:02:28 WARN hdfs.DFSClient: Caught exception
cloudera@quickstart:...  [cloudera@quickstart:...  [cloudera@quickstart:...  cloudera@quickstart:~...
```

```
File  Edit  View  Search  Terminal  Help
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=685688
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=512
                HDFS: Number of bytes written=5408880
                HDFS: Number of read operations=16
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=8
        Job Counters
                Launched map tasks=4
                Other local map tasks=4
                Total time spent by all maps in occupied slots (ms)=350148
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=350148
                Total vcore-milliseconds taken by all map tasks=350148
                Total megabyte-milliseconds taken by all map tasks=358551552
        Map-Reduce Framework
                Map input records=172198
                Map output records=172198
                Input split bytes=512
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=5016
                CPU time spent (ms)=29270
                Physical memory (bytes) snapshot=500170752
                Virtual memory (bytes) snapshot=6045212672
                Total committed heap usage (bytes)=243531776
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=5408880
23/11/21 02:04:27 INFO mapreduce.ImportJobBase: Transferred 5.1583 MB in 121.5941 seconds (43.4405 KB/sec)
23/11/21 02:04:27 INFO mapreduce.ImportJobBase: Retrieved 172198 records.
23/11/21 02:04:27 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `order_items` AS t LIMIT 1
23/11/21 02:04:27 INFO hive.HiveImport: Loading uploaded data into Hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
OK
Time taken: 1.434 seconds
Loading data to table default.order_i
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/order_i': User does not belong to supergroup
Table default.order_i stats: [numFiles=4, numRows=0, totalSize=5408880, rawDataSize=0]
OK
Time taken: 0.42 seconds
[cloudera@quickstart Desktop]$
```

Performing HQL Queries on the table:

```
hive> select count(*) from order_i where order_item_product_id = 403;
Query ID = cloudera_20231121021616_6e7eb9dd-f2d3-4624-90b4-0e296be5b6fe
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1700531075270_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1700531075270_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1700531075270_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-21 02:16:34,387 Stage-1 map = 0%,  reduce = 0%
2023-11-21 02:16:39,621 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.06 sec
2023-11-21 02:16:45,836 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.83 sec
MapReduce Total cumulative CPU time: 1 seconds 830 msec
Ended Job = job_1700531075270_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.83 sec   HDFS Read: 5418924 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 830 msec
OK
22246

hive> select count(*) from order_i where order_item_product_id = 957;
Query ID = cloudera_20231121021313_94899c16-ad00-4672-a379-ce316051adb5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1700531075270_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1700531075270_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1700531075270_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-21 02:13:14,036 Stage-1 map = 0%,  reduce = 0%
2023-11-21 02:13:19,276 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.09 sec
2023-11-21 02:13:24,451 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.87 sec
MapReduce Total cumulative CPU time: 1 seconds 870 msec
Ended Job = job_1700531075270_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.87 sec   HDFS Read: 5418931 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 870 msec
OK
13729

hive> select avg(order_item_quantity) from order_i;
Query ID = cloudera_20231121024545_b5758646-8409-45cc-8a88-9f6e69d47b42
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1700531075270_0018, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1700531075270_0018/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1700531075270_0018
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-21 02:45:55,557 Stage-1 map = 0%,  reduce = 0%
2023-11-21 02:46:15,025 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.49 sec
2023-11-21 02:46:35,964 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.15 sec
MapReduce Total cumulative CPU time: 8 seconds 150 msec
Ended Job = job_1700531075270_0018
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.15 sec   HDFS Read: 5418262 HDFS Write: 19 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 150 msec
OK
2.1821275508426345
Time taken: 83.856 seconds, Fetched: 1 row(s)
hive>
```

```
Time taken: 62.641 seconds, Fetched: 1 row(s)
hive> select min(order_item_product_price) from order_i;
Query ID = cloudera_20231121025555_3dc5e254-111e-4dee-a801-919052a9350d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1700531075270_0022, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1700531075270_0022/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1700531075270_0022
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-21 02:55:52,943 Stage-1 map = 0%,  reduce = 0%
2023-11-21 02:56:12,460 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.85 sec
2023-11-21 02:56:32,763 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.45 sec
MapReduce Total cumulative CPU time: 8 seconds 450 msec
Ended Job = job_1700531075270_0022
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.45 sec   HDFS Read: 5417993 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 450 msec
OK
9.99
Time taken: 61.895 seconds, Fetched: 1 row(s)

hive> select max(order_item_subtotal) from order_i;
Query ID = cloudera_20231121022020_78fbfa93-b96c-4fef-8b80-e20bbe9ef033
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1700531075270_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1700531075270_0014/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1700531075270_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-21 02:20:25,209 Stage-1 map = 0%,  reduce = 0%
2023-11-21 02:20:30,412 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.02 sec
2023-11-21 02:20:36,661 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.77 sec
MapReduce Total cumulative CPU time: 1 seconds 770 msec
Ended Job = job_1700531075270_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.77 sec   HDFS Read: 5417997 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 770 msec
OK
1999.99
Time taken: 17.295 seconds, Fetched: 1 row(s)
hive>
```

Data Visualisation

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [3]:
```
data = pd.read_csv('/kaggle/input/retail-sales-dataset/retail_sales_dataset.csv')
```

In [4]:
```
data.head()
```

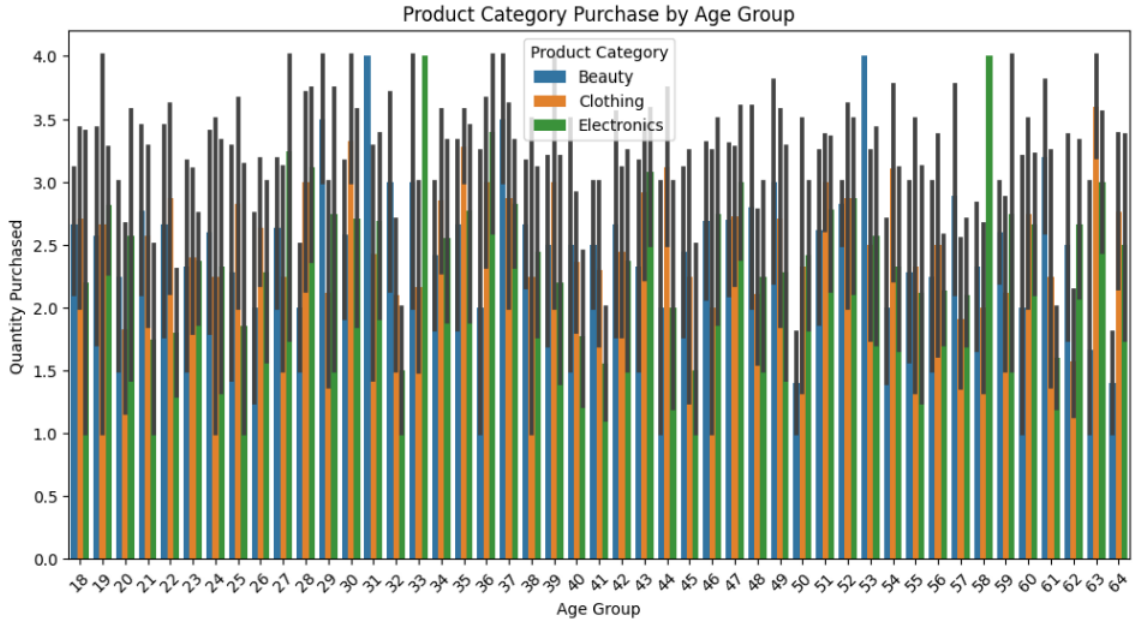
Out[4]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100

In [21]:
```
data = data.drop(['Date'], axis = 1)
```

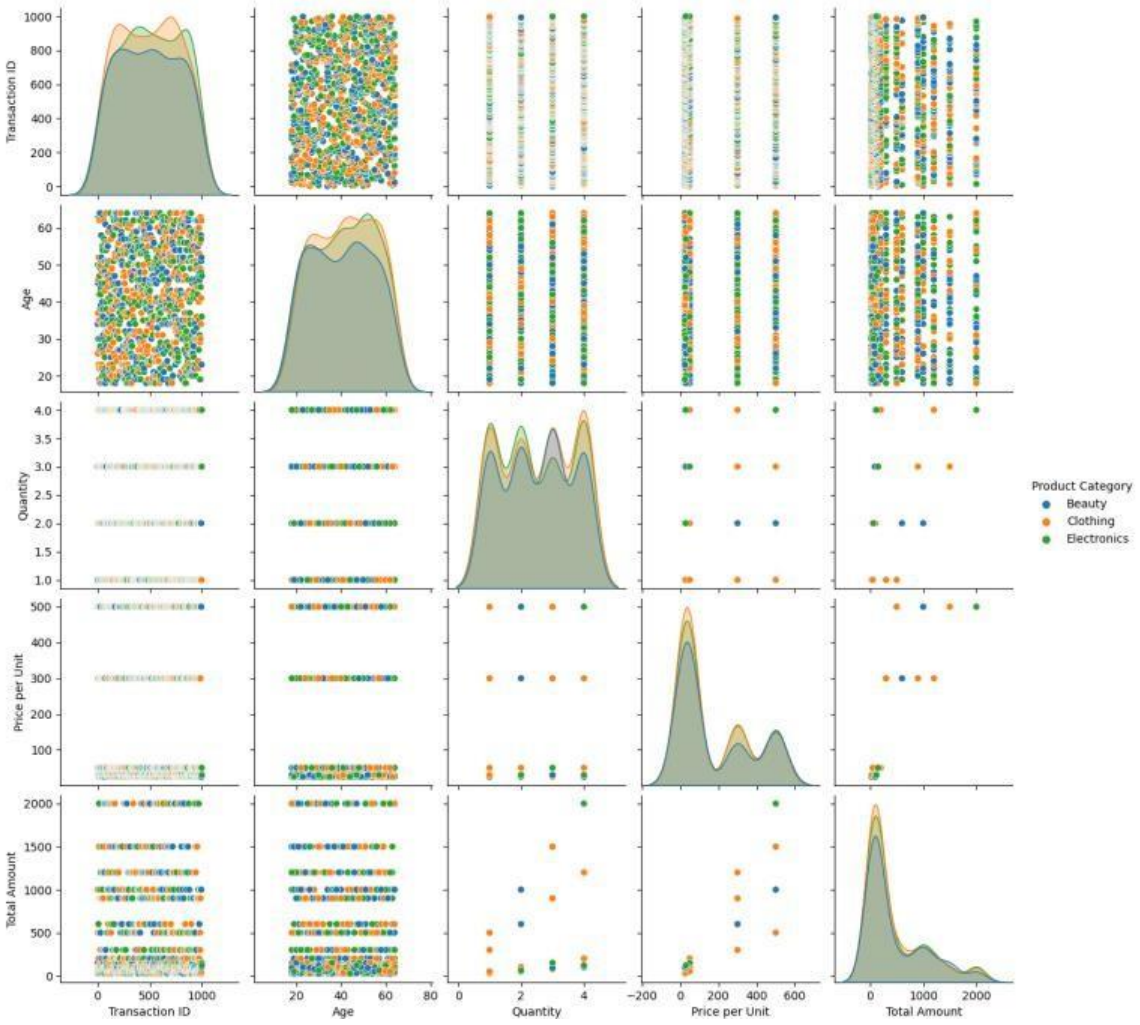
In [5]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='Age', y='Quantity', hue='Product Category', data=data)

plt.title('Product Category Purchase by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Quantity Purchased')
plt.xticks(rotation=45)
plt.figure(figsize = [10, 10])
plt.show()
```



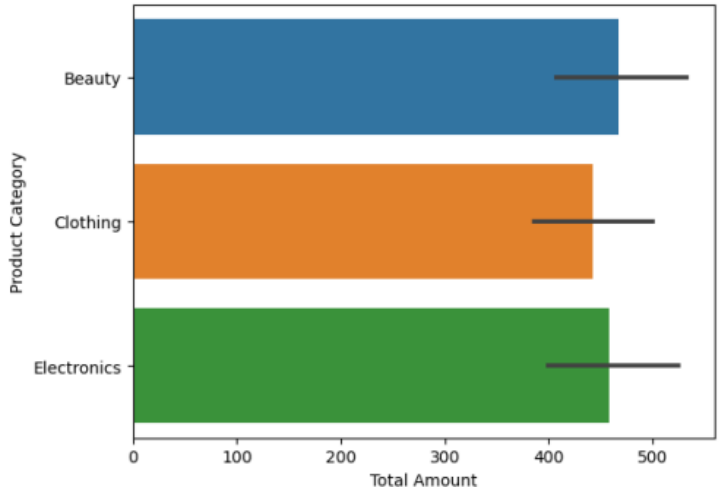

```
<Figure size 1000x1000 with 0 Axes>
```

In [7]:
```
sns.pairplot(data, hue='Product Category')
# Show the plot
plt.show()
```



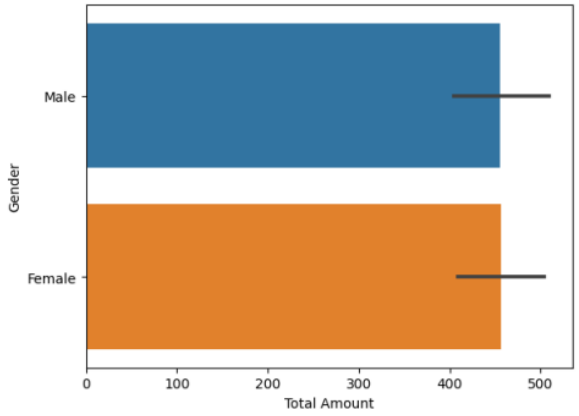
In [8]:
```
sns.barplot(x= data['Total Amount'], y=data['Product Category'])
```

Out[8]: <Axes: xlabel='Total Amount', ylabel='Product Category'>



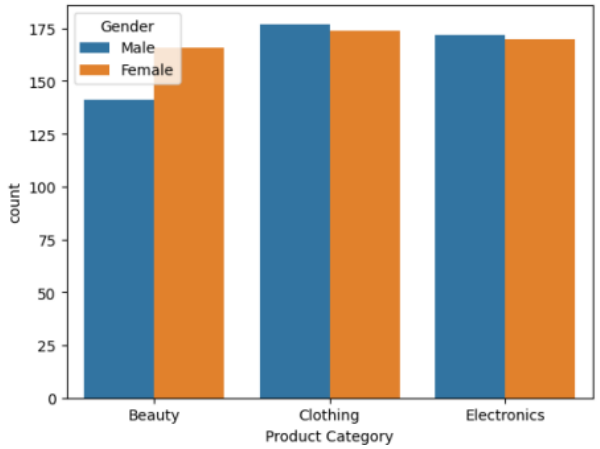
In [11]:
```
sns.barplot(x= data['Total Amount'], y=data['Gender'])
```

Out[11]: <Axes: xlabel='Total Amount', ylabel='Gender'>



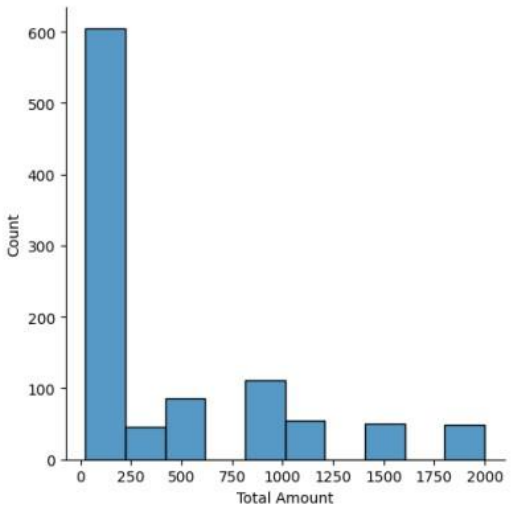
In [14]:
```
sns.countplot(data, x = data['Product Category'], hue = 'Gender')
```

Out[14]: <Axes: xlabel='Product Category', ylabel='count'>



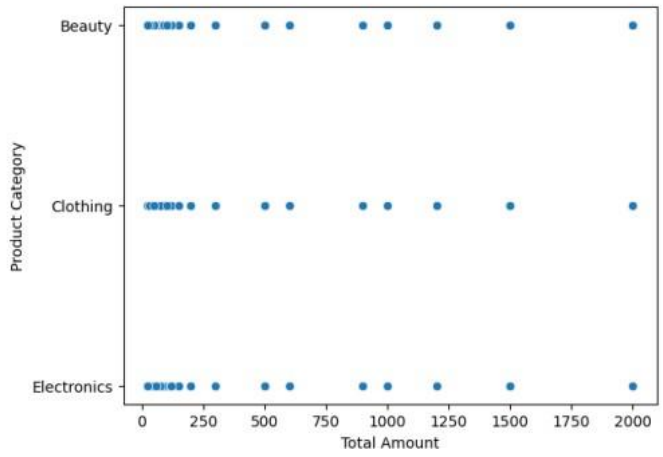
In [18]:
```
sns.displot(data['Total Amount'], bins = 10)
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x7e3a67517730>



In [24]:
```
sns.scatterplot(x='Total Amount', y='Product Category', data= data)
```

Out[24]:
```
<Axes: xlabel='Total Amount', ylabel='Product Category'>
```



In [32]:
```
sns.boxplot(x='Total Amount', y='Product Category', data=data)
```

Out[32]:
```
<Axes: xlabel='Total Amount', ylabel='Product Category'>
```