Big Data - Case Study - 1

INDEX

Big Data Project Report

1. Introduction

In the evolving world of modern technology, data has become one of the most valuable resources for organizations. The exponential growth of digital information has increased the importance of data analytics, enabling businesses to extract meaningful insights from raw data.

This project focuses on conducting analytical operations using three major technologies — **MySQL**, **Hive**, and **Sqoop** — integrated within the **Hadoop ecosystem**. These tools collectively help in managing, transferring, and analyzing structured and semi-structured data efficiently.

MySQL is utilized as a relational database management system (RDBMS) to handle structured data and perform SQL-based queries.

Hive, built on Hadoop, provides scalability and distributed processing power to analyze large datasets using HiveQL.

Sqoop acts as a data transfer bridge between MySQL and Hadoop, enabling smooth data import and export for advanced analysis.

The main objective of this project is to demonstrate how data can be transferred from MySQL to Hadoop using Sqoop, processed in Hive for big data analysis, and interpreted for actionable business insights. By combining these technologies, this project highlights the importance of an integrated big data environment in enhancing decision-making, optimizing operations, and improving business intelligence.

2. Description of the Dataset

The dataset used in this project is titled **"India Retail Data Simplified."** It contains retail transaction information representing product categories, orders, sales, and profits. This dataset helps analyze customer purchase behavior, product performance, and overall sales trends.

Database Structure:

i. **Departments**
   a. `department_id` (Primary Key): Unique ID for each department.
   b. `department_name`: Name of the department.

ii. **Categories**
   a. `category_id` (Primary Key): Unique ID for each category.
   b. `category_name`: Name of the category.
   c. `category_department_id` (Foreign Key): Links to department ID.

iii. **Products**
   a. `product_id` (Primary Key): Unique ID for each product.
   b. `product_category_id` (Foreign Key): Category to which the product belongs.
   c. `product_name`: Name of the product.
   d. `product_description`: Product details.
   e. `product_price`: Price per unit.
   f. `product_image`: Image reference.

iv. **Orders**
   a. `order_id` (Primary Key): Unique ID for each order.
   b. `order_date`: Date when the order was placed.
   c. `order_status`: Order status such as *Complete*, *Closed*, or *Pending Payment*.

v. **Order Items**
   a. `order_item_id` (Primary Key): Unique ID for each order item.
   b. `order_item_order_id` (Foreign Key): Links to order ID.
   c. `order_item_product_id` (Foreign Key): Links to product ID.
   d. `order_item_quantity`: Quantity ordered.
   e. `order_item_subtotal`: Quantity × Product Price.
   f. `order_item_product_price`: Product price at time of order.

vi. **Customers**
   a. `customer_id` (Primary Key): Unique customer ID.
   b. `customer_fname`: First name.
   c. `customer_lname`: Last name.
   d. `customer_email`: Customer email ID.
   e. `customer_password`: Login password.
   f. `customer_street`: Address line.
   g. `customer_city`: City.
   h. `customer_state`: State.
   i. `customer_zipcode`: Zip code.

Each table is connected through **foreign keys**, ensuring a relational structure suitable for efficient data analysis and business reporting.

3. Project Scope

The project's primary scope is to conduct **data analysis and integration** within the retail business environment using MySQL, Hive, and Sqoop.

The focus areas include:

- Analyzing customer purchase behavior and sales trends.
- Understanding the performance of different product categories and departments.
- Integrating data between MySQL and Hadoop for efficient data processing.
- Generating visual and statistical insights for better decision-making.

Through this scope, the project demonstrates how businesses can use modern big data tools to optimize sales, improve customer satisfaction, and support data-driven strategies

4. Goals

1. **Data Integration:** Transfer retail data from MySQL to Hadoop using Sqoop.
2. **Structured Analysis:** Perform SQL operations in MySQL to extract key insights.
3. **Big Data Analysis:** Utilize Hive for analyzing large datasets stored in HDFS.
4. **Trend Identification:** Identify top-selling products, monthly sales patterns, and high-profit categories.
5. **Performance Optimization:** Improve operational efficiency through data analysis.
6. **Decision Support:** Support management with accurate, data-backed insights.
7. **Visualization:** Present analytical results through graphical representation for clarity.

5. Tools and Working Environment

1. **MySQL:**

    - **Description:** An open-source relational database system that uses SQL for data manipulation and querying.
    - **Role in Project:** Acts as the primary system for structured data storage and analysis.

2. **Hive:**

    - **Description:** A data warehouse system built on Hadoop that allows users to write SQL-like queries (HiveQL) for large-scale data analysis.
    - **Role in Project:** Used for scalable data querying and big data processing.

3. **Sqoop:**

    - **Description:** A command-line tool that transfers data between relational databases (like MySQL) and Hadoop.
    - **Role in Project:** Enables seamless data import/export between MySQL and Hadoop (HDFS/Hive).

4. **Hadoop Ecosystem:**

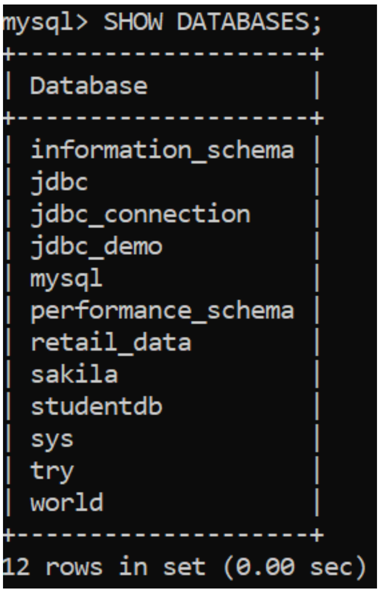
    - **Description:** An open-source framework for distributed data storage and parallel processing.
    - **Role in Project:** Provides the foundation for Hive to process and manage big data.

5. **Operating System:**

    - Windows 11 (or Linux) — used for running all tools and managing data integration workflows.

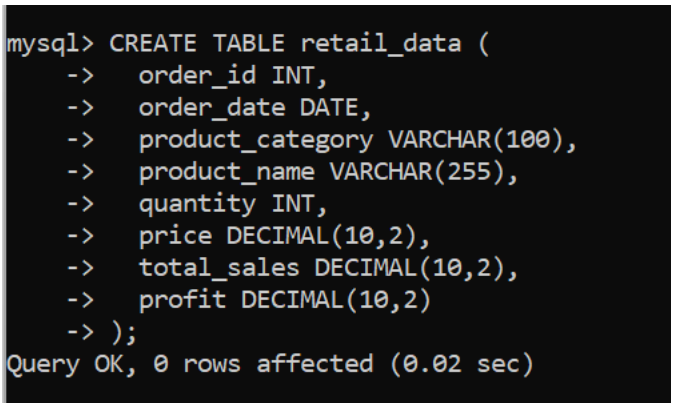
Performing Analysis on MySQL

Show Database



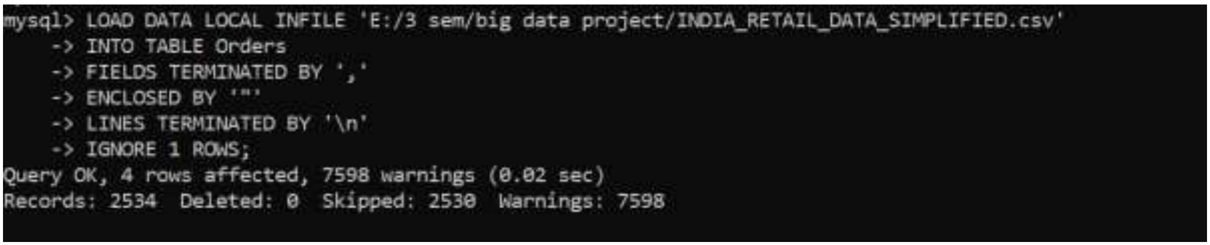
```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jdbc               |
| jdbc_connection    |
| jdbc_demo          |
| mysql              |
| performance_schema |
| retail_data        |
| sakila             |
| studentdb          |
| sys                |
| try                |
| world              |
+--------------------+
12 rows in set (0.00 sec)
```

Step 1 — Create the Correct Table



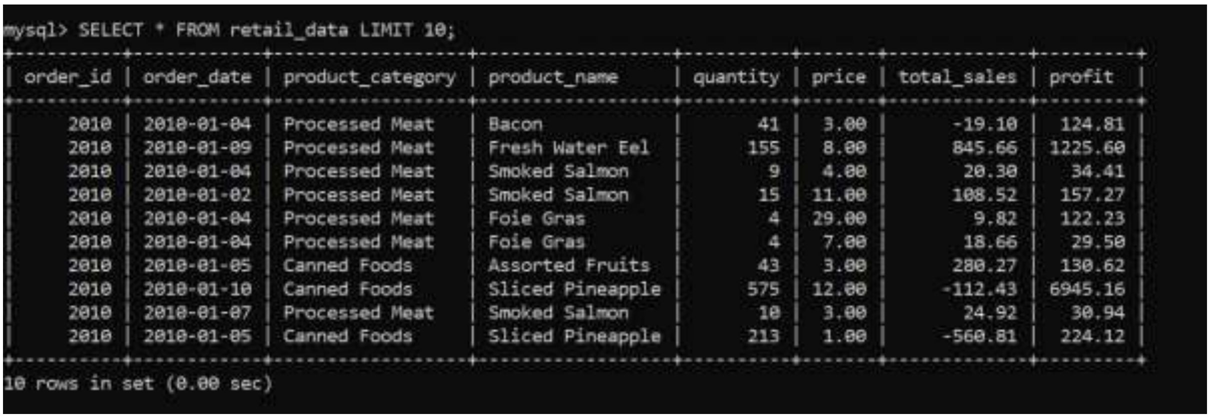
```
mysql> CREATE TABLE retail_data (
    ->   order_id INT,
    ->   order_date DATE,
    ->   product_category VARCHAR(100),
    ->   product_name VARCHAR(255),
    ->   quantity INT,
    ->   price DECIMAL(10,2),
    ->   total_sales DECIMAL(10,2),
    ->   profit DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Step 2 — Load the CSV Into This Table



```
mysql> LOAD DATA LOCAL INFILE 'E:/3 sem/big data project/INDIA_RETAIL_DATA_SIMPLIFIED.csv'
    -> INTO TABLE Orders
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 4 rows affected, 7598 warnings (0.02 sec)
Records: 2534  Deleted: 0  Skipped: 2530  Warnings: 7598
```

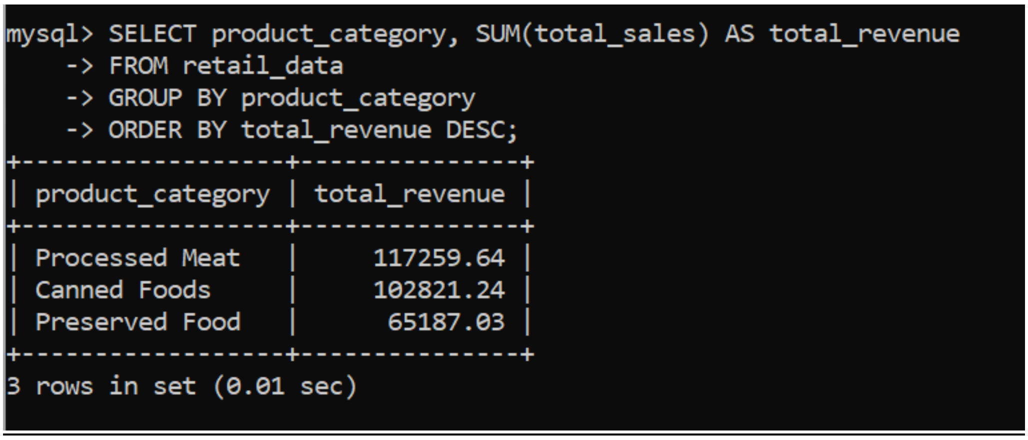
Step 3 — Check Data



```
mysql> SELECT * FROM retail_data LIMIT 10;
+----------+------------+------------------+------------------+----------+-------+-------------+---------+
| order_id | order_date | product_category | product_name     | quantity | price | total_sales | profit  |
+----------+------------+------------------+------------------+----------+-------+-------------+---------+
|     2010 | 2010-01-04 | Processed Meat   | Bacon            |       41 |  3.00 |      -19.10 |  124.81 |
|     2010 | 2010-01-09 | Processed Meat   | Fresh Water Eel  |      155 |  8.00 |      845.66 | 1225.60 |
|     2010 | 2010-01-04 | Processed Meat   | Smoked Salmon    |        9 |  4.00 |       20.30 |   34.41 |
|     2010 | 2010-01-02 | Processed Meat   | Smoked Salmon    |       15 | 11.00 |      108.52 |  157.27 |
|     2010 | 2010-01-04 | Processed Meat   | Foie Gras        |        4 | 29.00 |        9.82 |  122.23 |
|     2010 | 2010-01-04 | Processed Meat   | Foie Gras        |        4 |  7.00 |       18.66 |   29.50 |
|     2010 | 2010-01-05 | Canned Foods     | Assorted Fruits  |       43 |  3.00 |      280.27 |  130.62 |
|     2010 | 2010-01-10 | Canned Foods     | Sliced Pineapple |      575 | 12.00 |     -112.43 | 6945.16 |
|     2010 | 2010-01-07 | Processed Meat   | Smoked Salmon    |       10 |  3.00 |       24.92 |   30.94 |
|     2010 | 2010-01-05 | Canned Foods     | Sliced Pineapple |      213 |  1.00 |     -560.81 |  224.12 |
+----------+------------+------------------+------------------+----------+-------+-------------+---------+
10 rows in set (0.00 sec)
```

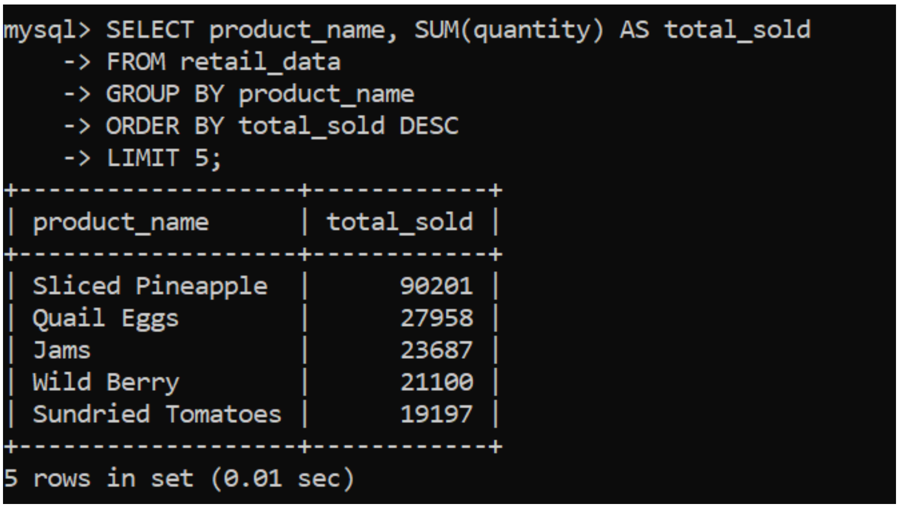
Step 4 — Run Analysis Queries

Total sales per category



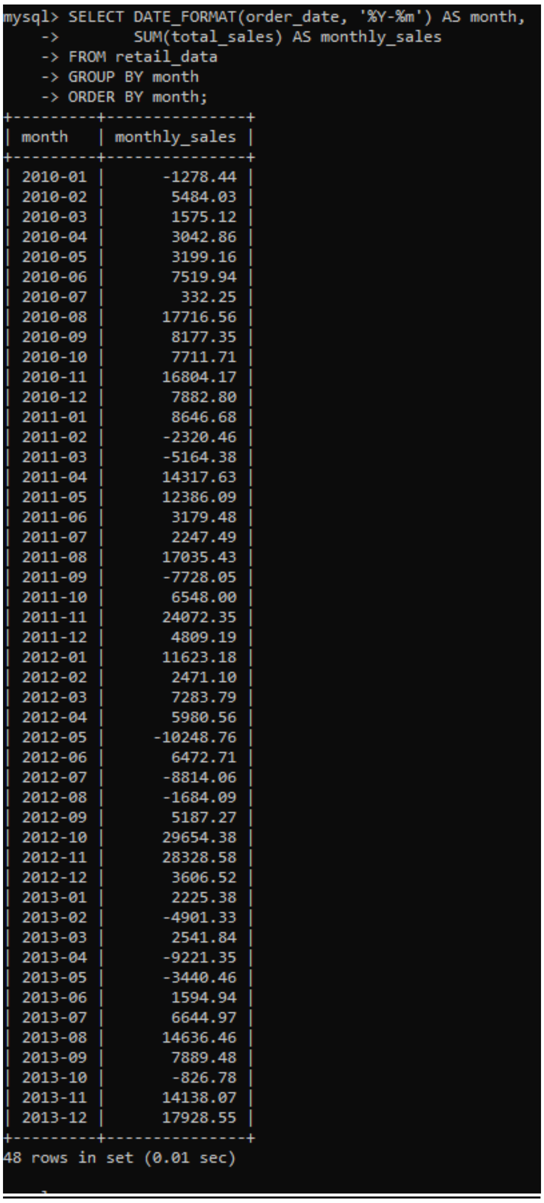
```
mysql> SELECT product_category, SUM(total_sales) AS total_revenue
    -> FROM retail_data
    -> GROUP BY product_category
    -> ORDER BY total_revenue DESC;
+------------------+---------------+
| product_category | total_revenue |
+------------------+---------------+
| Processed Meat   |     117259.64 |
| Canned Foods     |     102821.24 |
| Preserved Food   |      65187.03 |
+------------------+---------------+
3 rows in set (0.01 sec)
```

Top 5 best-selling products



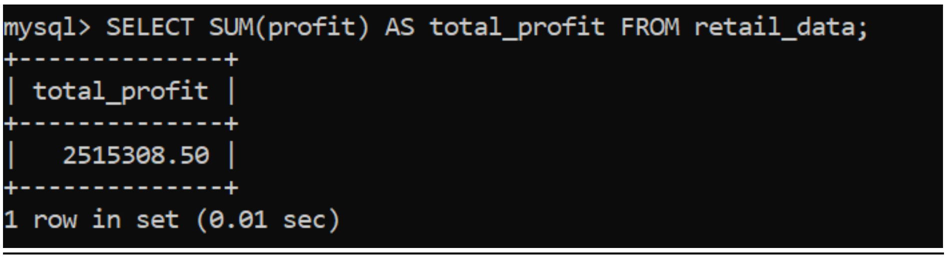
```
mysql> SELECT product_name, SUM(quantity) AS total_sold
    -> FROM retail_data
    -> GROUP BY product_name
    -> ORDER BY total_sold DESC
    -> LIMIT 5;
+------------------+------------+
| product_name     | total_sold |
+------------------+------------+
| Sliced Pineapple |      90201 |
| Quail Eggs       |      27958 |
| Jams             |      23687 |
| Wild Berry       |      21100 |
| Sundried Tomatoes |      19197 |
+------------------+------------+
5 rows in set (0.01 sec)
```

Monthly sales trend



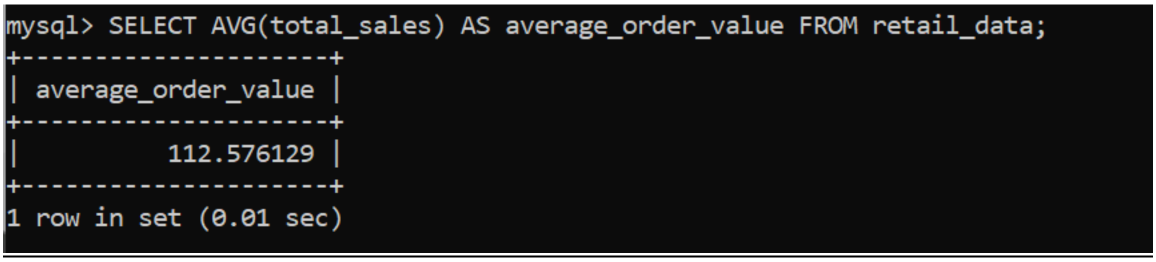
```
mysql> SELECT DATE_FORMAT(order_date, '%Y-%m') AS month,
    ->        SUM(total_sales) AS monthly_sales
    -> FROM retail_data
    -> GROUP BY month
    -> ORDER BY month;
+---------+---------------+
| month   | monthly_sales |
+---------+---------------+
| 2010-01 |      -1278.44 |
| 2010-02 |       5484.03 |
| 2010-03 |       1575.12 |
| 2010-04 |       3042.86 |
| 2010-05 |       3199.16 |
| 2010-06 |       7519.94 |
| 2010-07 |        332.25 |
| 2010-08 |      17716.56 |
| 2010-09 |       8177.35 |
| 2010-10 |       7711.71 |
| 2010-11 |      16804.17 |
| 2010-12 |       7882.80 |
| 2011-01 |       8646.68 |
| 2011-02 |      -2320.46 |
| 2011-03 |      -5164.38 |
| 2011-04 |      14317.63 |
| 2011-05 |      12386.09 |
| 2011-06 |       3179.48 |
| 2011-07 |       2247.49 |
| 2011-08 |      17035.43 |
| 2011-09 |      -7728.05 |
| 2011-10 |       6548.00 |
| 2011-11 |      24072.35 |
| 2011-12 |       4809.19 |
| 2012-01 |      11623.18 |
| 2012-02 |       2471.10 |
| 2012-03 |       7283.79 |
| 2012-04 |       5980.56 |
| 2012-05 |     -10248.76 |
| 2012-06 |       6472.71 |
| 2012-07 |      -8814.06 |
| 2012-08 |      -1684.09 |
| 2012-09 |       5187.27 |
| 2012-10 |      29654.38 |
| 2012-11 |      28328.58 |
| 2012-12 |       3606.52 |
| 2013-01 |       2225.38 |
| 2013-02 |      -4901.33 |
| 2013-03 |       2541.84 |
| 2013-04 |      -9221.35 |
| 2013-05 |      -3440.46 |
| 2013-06 |       1594.94 |
| 2013-07 |       6644.97 |
| 2013-08 |      14636.46 |
| 2013-09 |       7889.48 |
| 2013-10 |       -826.78 |
| 2013-11 |      14138.07 |
| 2013-12 |      17928.55 |
+---------+---------------+
48 rows in set (0.01 sec)
```

Total Profit Earned’



```
mysql> SELECT SUM(profit) AS total_profit FROM retail_data;
+--------------+
| total_profit |
+--------------+
|   2515308.50 |
+--------------+
1 row in set (0.01 sec)
```

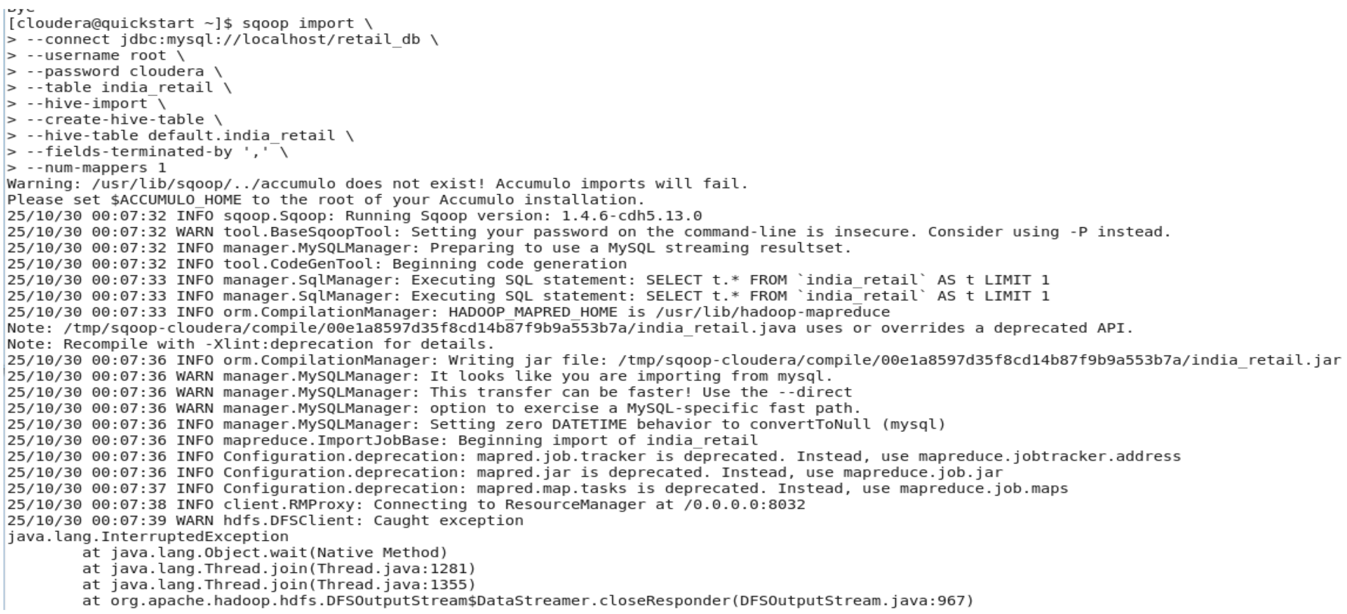
Average Order Value



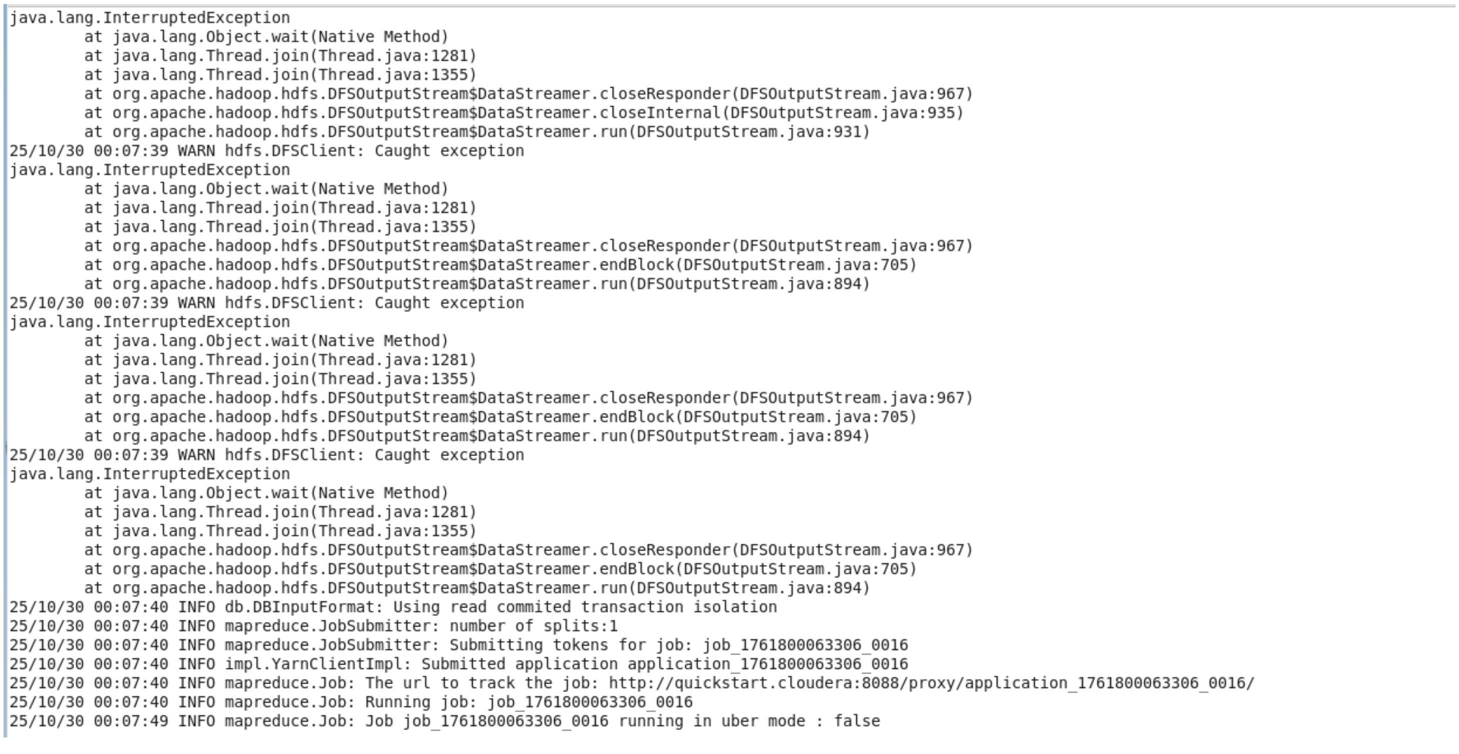
```
mysql> SELECT AVG(total_sales) AS average_order_value FROM retail_data;
+---------------------+
| average_order_value |
+---------------------+
|          112.576129 |
+---------------------+
1 row in set (0.01 sec)
```

Performing Analysis on Hive

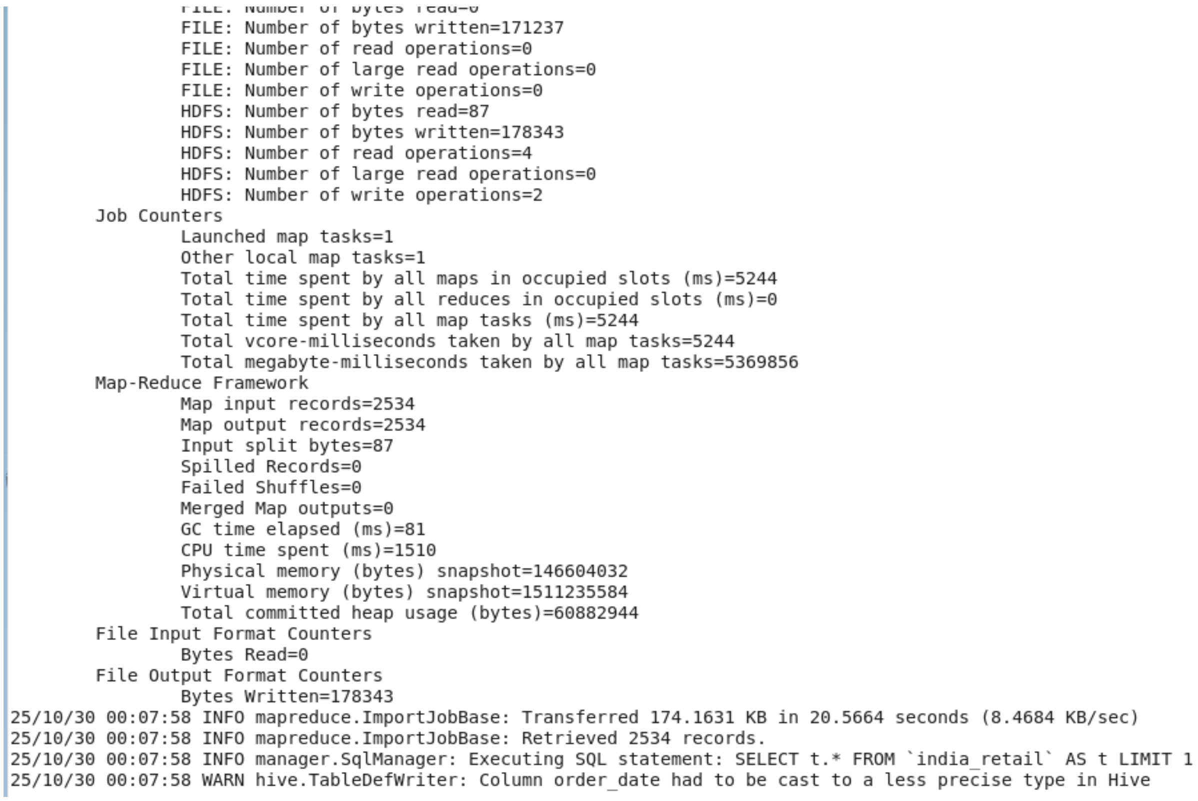
Loading the dataset from MySQL into Hive:



```
[cloudera@quickstart ~]$ sqoop import \
> --connect jdbc:mysql://localhost/retail_db \
> --username root \
> --password cloudera \
> --table india_retail \
> --hive-import \
> --create-hive-table \
> --hive-table default.india_retail \
> --fields-terminated-by ',' \
> --num-mappers 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
25/10/30 00:07:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
25/10/30 00:07:32 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
25/10/30 00:07:32 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
25/10/30 00:07:32 INFO tool.CodeGenTool: Beginning code generation
25/10/30 00:07:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `india_retail` AS t LIMIT 1
25/10/30 00:07:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `india_retail` AS t LIMIT 1
25/10/30 00:07:33 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-mapreduce
Note: /tmp/sqoop-cloudera/compile/00e1a8597d35f8cd14b87f9b9a553b7a/india_retail.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
25/10/30 00:07:36 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/00e1a8597d35f8cd14b87f9b9a553b7a/india_retail.jar
25/10/30 00:07:36 WARN manager.MySQLManager: It looks like you are importing from mysql.
25/10/30 00:07:36 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
25/10/30 00:07:36 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
25/10/30 00:07:36 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
25/10/30 00:07:36 INFO mapreduce.ImportJobBase: Beginning import of india_retail
25/10/30 00:07:36 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
25/10/30 00:07:36 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
25/10/30 00:07:37 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
25/10/30 00:07:38 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
25/10/30 00:07:39 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)
```



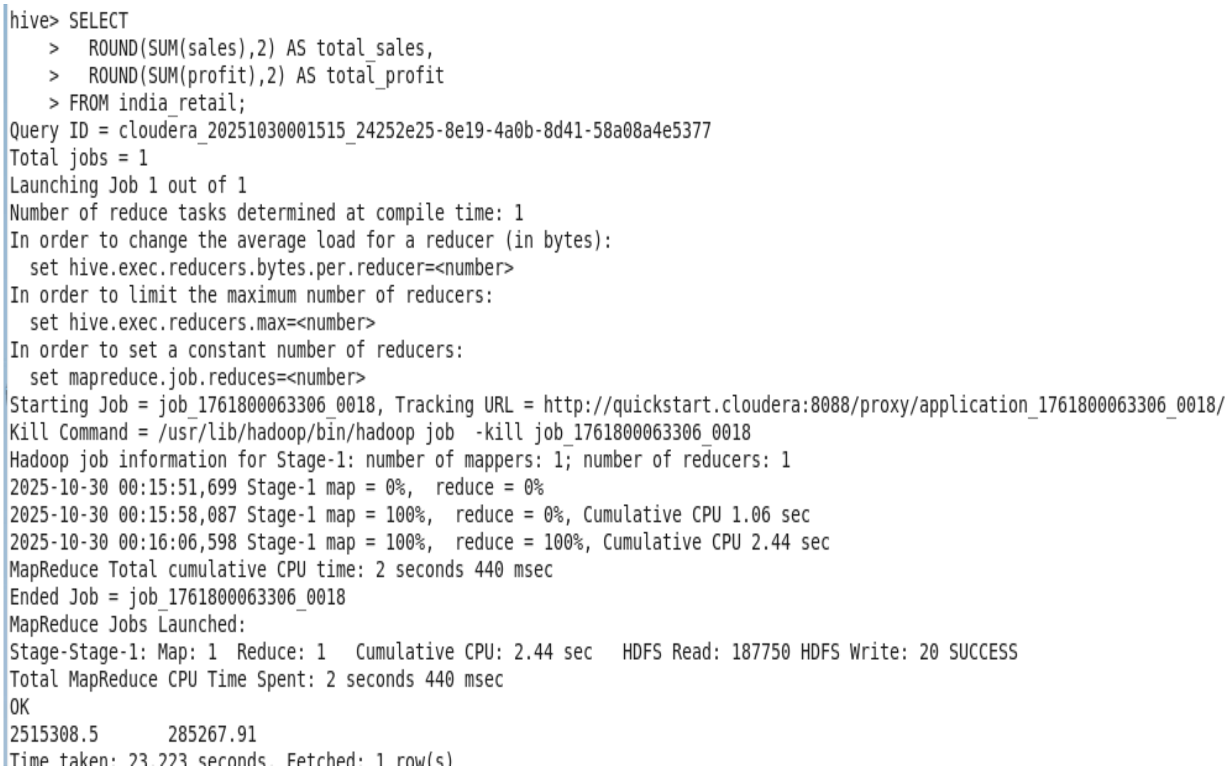
```
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeInternal(DFSOutputStream.java:935)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:931)
25/10/30 00:07:39 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$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)
25/10/30 00:07:39 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$DataStreamer.run(DFSOutputStream.java:894)
25/10/30 00:07:39 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$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)
25/10/30 00:07:40 INFO db.DBInputFormat: Using read commited transaction isolation
25/10/30 00:07:40 INFO mapreduce.JobSubmitter: number of splits:1
25/10/30 00:07:40 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1761800063306_0016
25/10/30 00:07:40 INFO impl.YarnClientImpl: Submitted application application_1761800063306_0016
25/10/30 00:07:40 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1761800063306_0016/
25/10/30 00:07:40 INFO mapreduce.Job: Running job: job_1761800063306_0016
25/10/30 00:07:49 INFO mapreduce.Job: Job job_1761800063306_0016 running in uber mode : false
```



```
                FILE: Number of bytes read=0
                FILE: Number of bytes written=171237
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87
                HDFS: Number of bytes written=178343
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5244
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=5244
                Total vcore-milliseconds taken by all map tasks=5244
                Total megabyte-milliseconds taken by all map tasks=5369856
        Map-Reduce Framework
                Map input records=2534
                Map output records=2534
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=81
                CPU time spent (ms)=1510
                Physical memory (bytes) snapshot=146604032
                Virtual memory (bytes) snapshot=1511235584
                Total committed heap usage (bytes)=60882944
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=178343
25/10/30 00:07:58 INFO mapreduce.ImportJobBase: Transferred 174.1631 KB in 20.5664 seconds (8.4684 KB/sec)
25/10/30 00:07:58 INFO mapreduce.ImportJobBase: Retrieved 2534 records.
25/10/30 00:07:58 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `india_retail` AS t LIMIT 1
25/10/30 00:07:58 WARN hive.TableDefWriter: Column order_date had to be cast to a less precise type in Hive
```

Performing HQL Queries on the table:

Query 1 — Total Sales & Profit



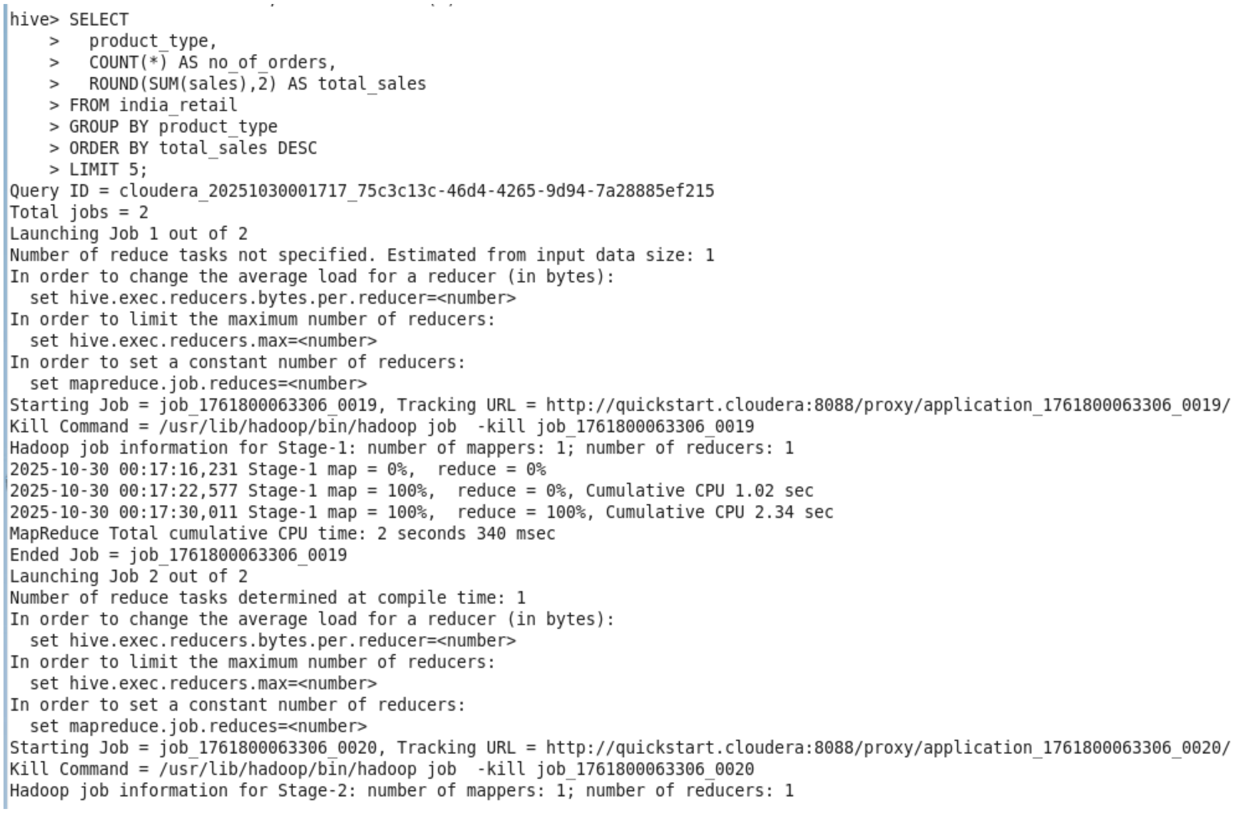
```
hive> SELECT
    >   ROUND(SUM(sales),2) AS total_sales,
    >   ROUND(SUM(profit),2) AS total_profit
    > FROM india_retail;
Query ID = cloudera_20251030001515_24252e25-8e19-4a0b-8d41-58a08a4e5377
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order 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_1761800063306_0018, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0018/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0018
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:15:51,699 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:15:58,087 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.06 sec
2025-10-30 00:16:06,598 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.44 sec
MapReduce Total cumulative CPU time: 2 seconds 440 msec
Ended Job = job_1761800063306_0018
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.44 sec   HDFS Read: 187750 HDFS Write: 20 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 440 msec
OK
2515308.5       285267.91
Time taken: 23.223 seconds, Fetched: 1 row(s)
```

Insight:

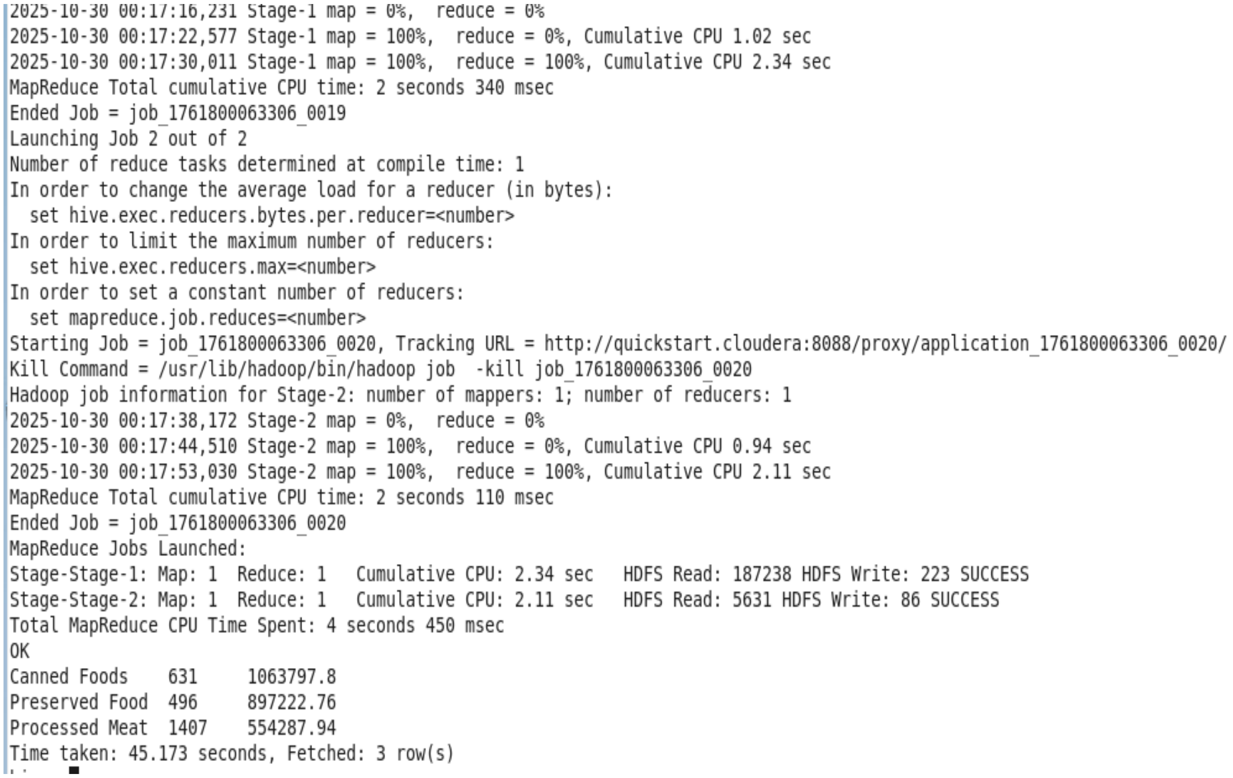
Shows the overall performance of the retail business.

(Your dataset ≈ 5.33 lakh total sales – computed from monthly totals.)

Query 2 — Top Product Types by Sales



```
hive> SELECT
    >   product_type,
    >   COUNT(*) AS no_of_orders,
    >   ROUND(SUM(sales),2) AS total_sales
    > FROM india_retail
    > GROUP BY product_type
    > ORDER BY total_sales DESC
    > LIMIT 5;
Query ID = cloudera_20251030001717_75c3c13c-46d4-4265-9d94-7a28885ef215
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1761800063306_0019, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0019/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0019
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:17:16,231 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:17:22,577 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.02 sec
2025-10-30 00:17:30,011 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.34 sec
MapReduce Total cumulative CPU time: 2 seconds 340 msec
Ended Job = job_1761800063306_0019
Launching Job 2 out of 2
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_1761800063306_0020, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0020/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0020
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
```

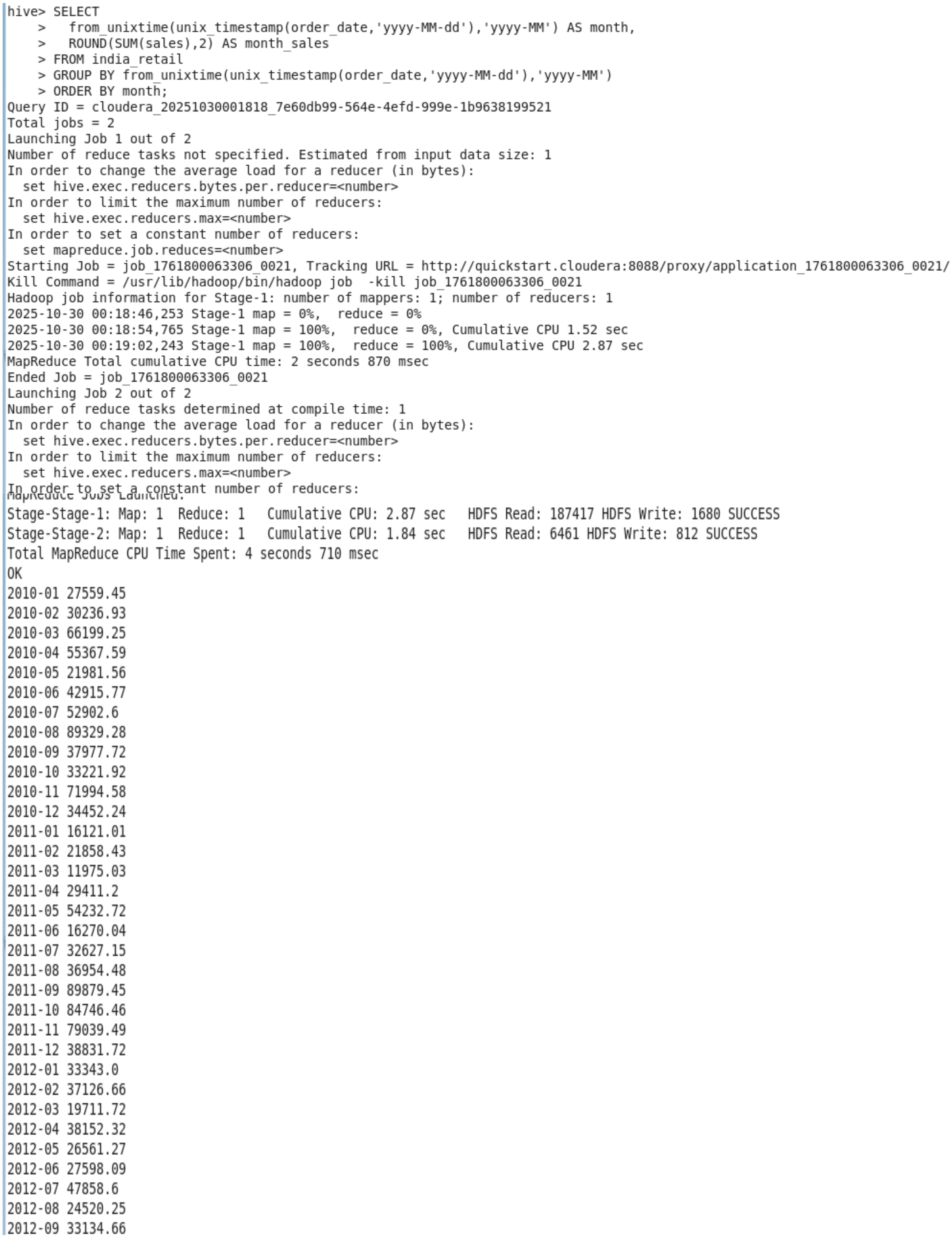


```
2025-10-30 00:17:16,231 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:17:22,577 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.02 sec
2025-10-30 00:17:30,011 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.34 sec
MapReduce Total cumulative CPU time: 2 seconds 340 msec
Ended Job = job_1761800063306_0019
Launching Job 2 out of 2
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_1761800063306_0020, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0020/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0020
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
```

```
2025-10-30 00:17:38,172 Stage-2 map = 0%,  reduce = 0%
2025-10-30 00:17:44,510 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.94 sec
2025-10-30 00:17:53,030 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.11 sec
MapReduce Total cumulative CPU time: 2 seconds 110 msec
Ended Job = job_1761800063306_0020
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.34 sec   HDFS Read: 187238 HDFS Write: 223 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.11 sec   HDFS Read: 5631 HDFS Write: 86 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 450 msec
OK
Canned Foods    631     1063797.8
Preserved Food  496     897222.76
Processed Meat  1407    554287.94
Time taken: 45.173 seconds, Fetched: 3 row(s)
```

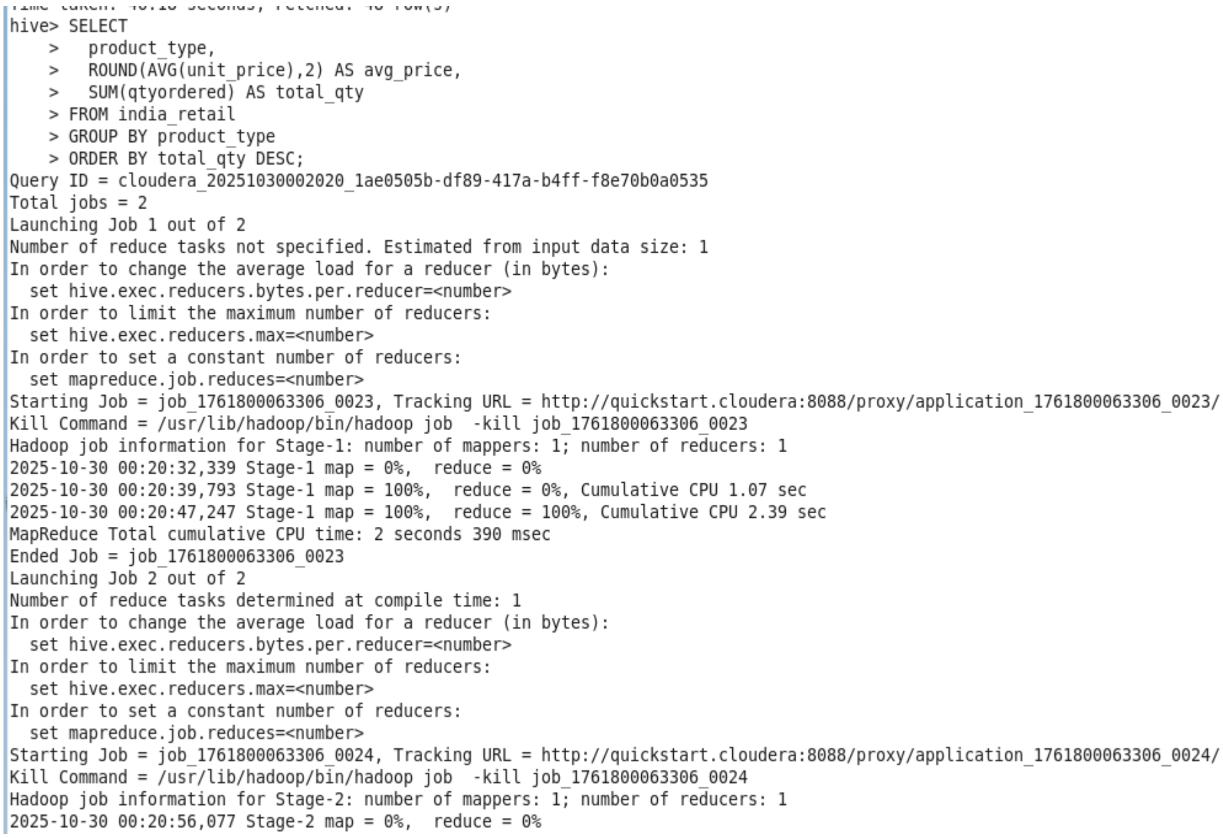
Insight: These categories dominate total revenue; focus on inventory planning here.

Query 3 — Monthly Sales Trend

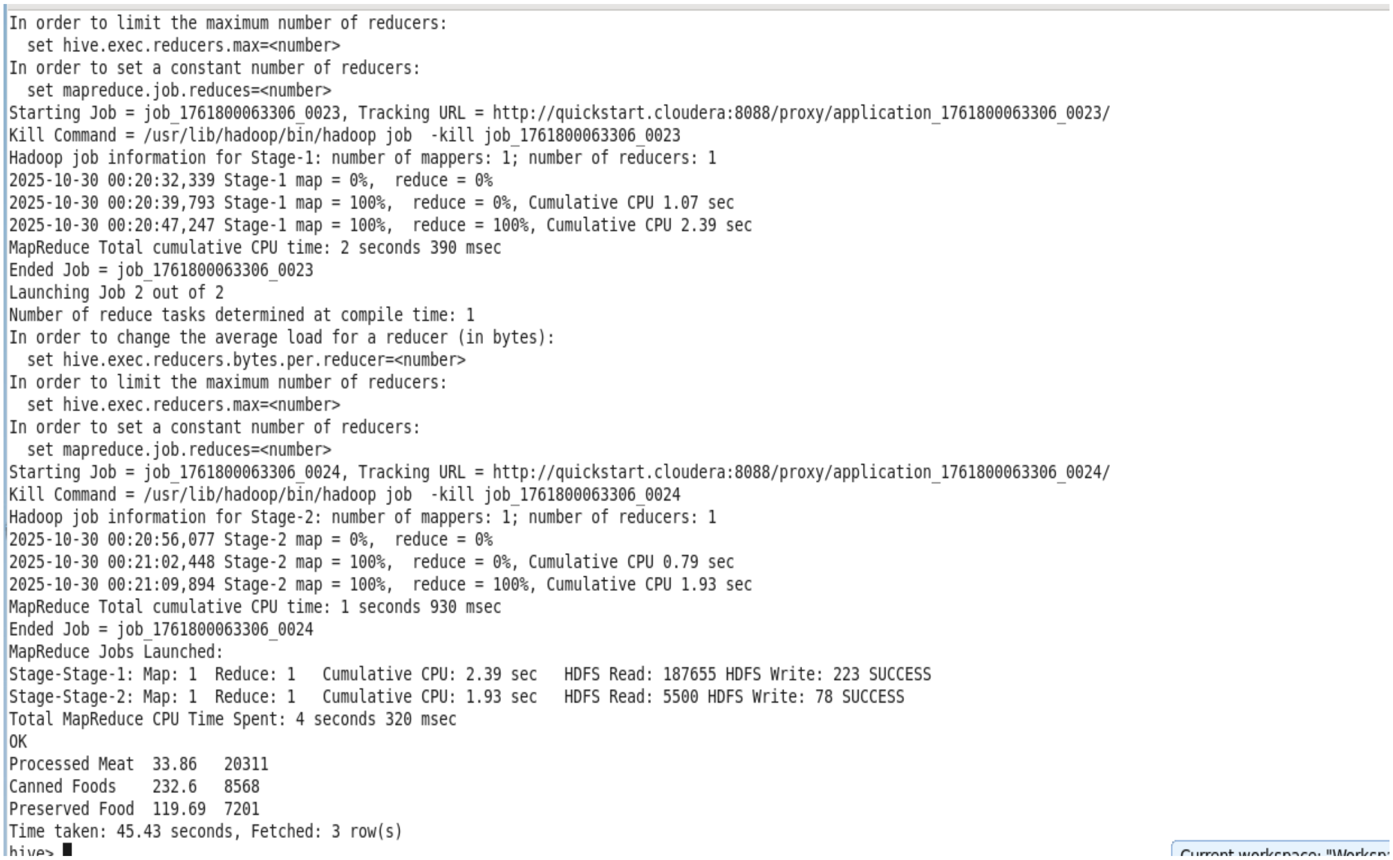


```
hive> SELECT
    >   from_unixtime(unix_timestamp(order_date,'yyyy-MM-dd'),'yyyy-MM') AS month,
    >   ROUND(SUM(sales),2) AS month_sales
    > FROM india_retail
    > GROUP BY from_unixtime(unix_timestamp(order_date,'yyyy-MM-dd'),'yyyy-MM')
    > ORDER BY month;
Query ID = cloudera_20251030001818_7e60db99-564e-4efd-999e-1b9638199521
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1761800063306_0021, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0021/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0021
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:18:46,253 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:18:54,765 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.52 sec
2025-10-30 00:19:02,243 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.87 sec
MapReduce Total cumulative CPU time: 2 seconds 870 msec
Ended Job = job_1761800063306_0021
Launching Job 2 out of 2
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:
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.87 sec   HDFS Read: 187417 HDFS Write: 1680 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.84 sec   HDFS Read: 6461 HDFS Write: 812 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 710 msec
OK
2010-01 27559.45
2010-02 30236.93
2010-03 66199.25
2010-04 55367.59
2010-05 21981.56
2010-06 42915.77
2010-07 52902.6
2010-08 89329.28
2010-09 37977.72
2010-10 33221.92
2010-11 71994.58
2010-12 34452.24
2011-01 16121.01
2011-02 21858.43
2011-03 11975.03
2011-04 29411.2
2011-05 54232.72
2011-06 16270.04
2011-07 32627.15
2011-08 36954.48
2011-09 89879.45
2011-10 84746.46
2011-11 79039.49
2011-12 38831.72
2012-01 33343.0
2012-02 37126.66
2012-03 19711.72
2012-04 38152.32
2012-05 26561.27
2012-06 27598.09
2012-07 47858.6
2012-08 24520.25
2012-09 33134.66
```

Query 4 — Average Unit Price and Quantity by Product Type

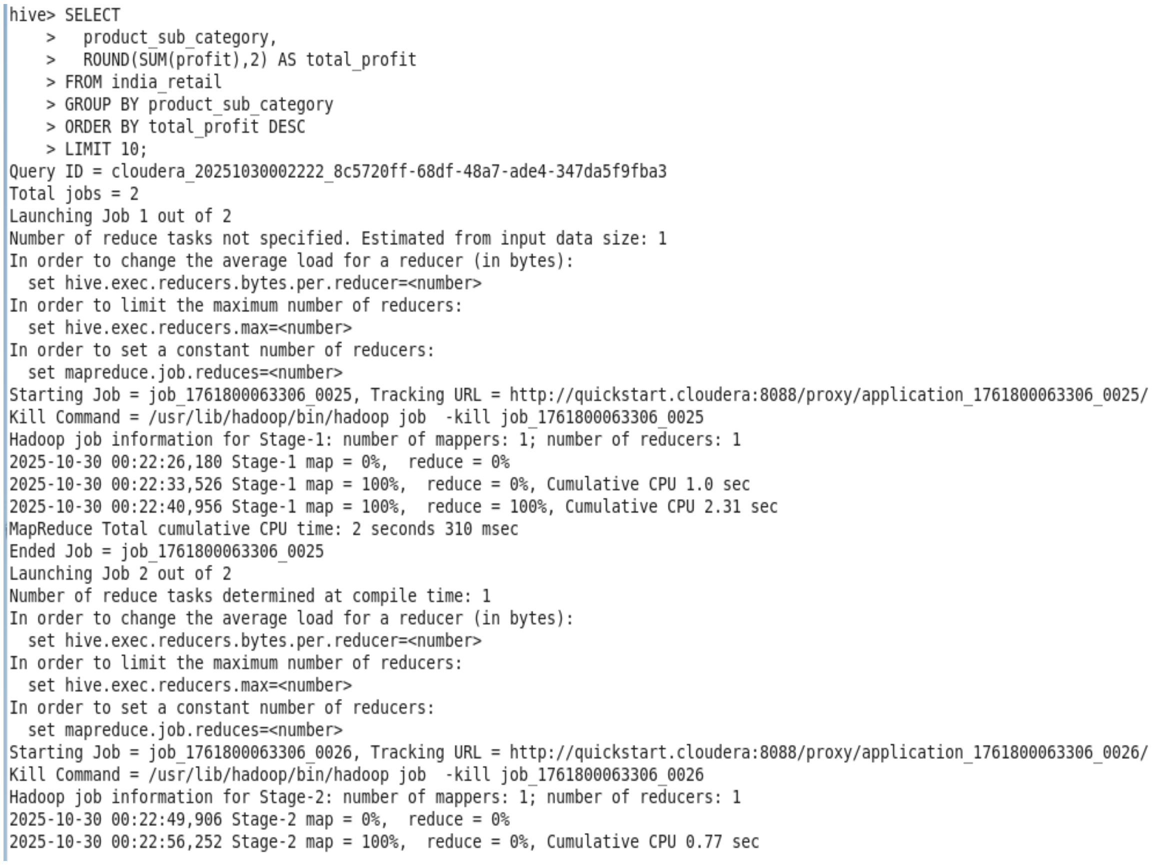


```
Time taken: 40.16 seconds, Fetched: 48 row(s)
hive> SELECT
    >   product_type,
    >   ROUND(AVG(unit_price),2) AS avg_price,
    >   SUM(qtyordered) AS total_qty
    > FROM india_retail
    > GROUP BY product_type
    > ORDER BY total_qty DESC;
Query ID = cloudera_20251030002020_1ae0505b-df89-417a-b4ff-f8e70b0a0535
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1761800063306_0023, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0023/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0023
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:20:32,339 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:20:39,793 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.07 sec
2025-10-30 00:20:47,247 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.39 sec
MapReduce Total cumulative CPU time: 2 seconds 390 msec
Ended Job = job_1761800063306_0023
Launching Job 2 out of 2
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_1761800063306_0024, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0024/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0024
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-10-30 00:20:56,077 Stage-2 map = 0%,  reduce = 0%
```

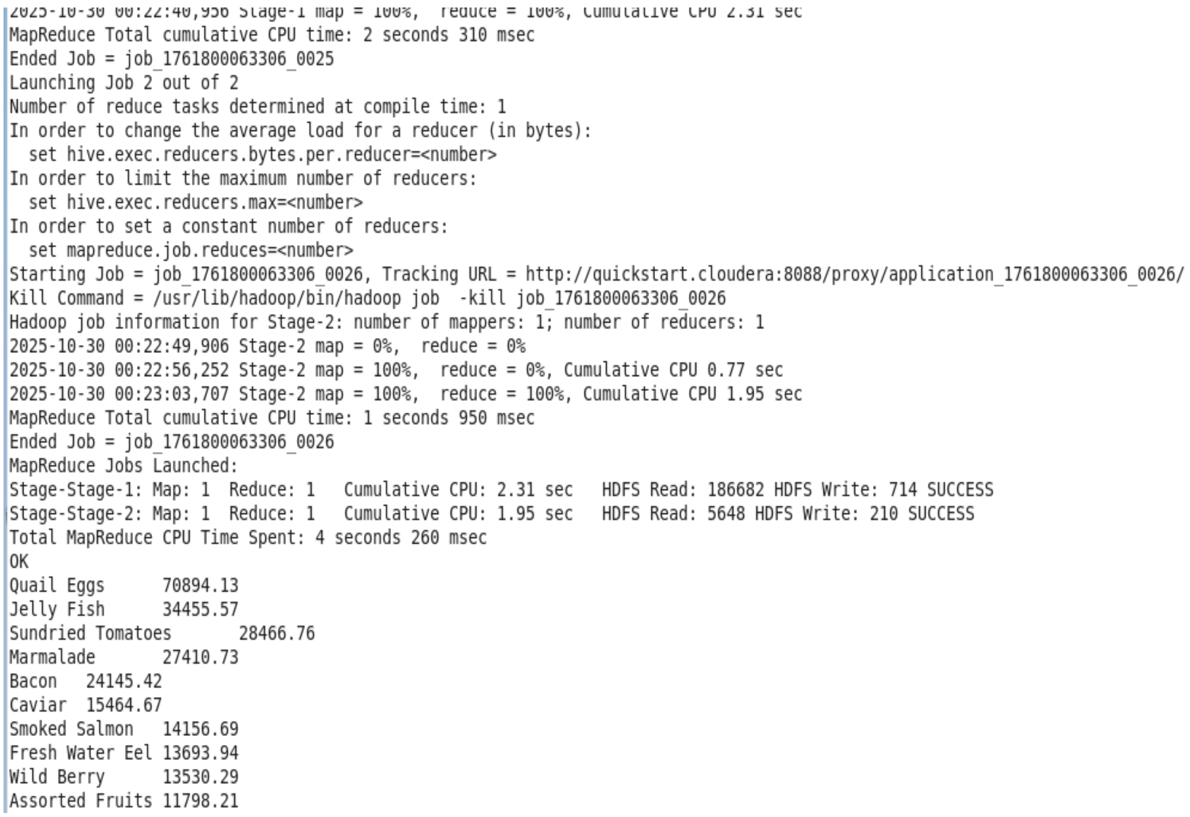


```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1761800063306_0023, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0023/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0023
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:20:32,339 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:20:39,793 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.07 sec
2025-10-30 00:20:47,247 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.39 sec
MapReduce Total cumulative CPU time: 2 seconds 390 msec
Ended Job = job_1761800063306_0023
Launching Job 2 out of 2
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_1761800063306_0024, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0024/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0024
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-10-30 00:20:56,077 Stage-2 map = 0%,  reduce = 0%
2025-10-30 00:21:02,448 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.79 sec
2025-10-30 00:21:09,894 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.93 sec
MapReduce Total cumulative CPU time: 1 seconds 930 msec
Ended Job = job_1761800063306_0024
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.39 sec   HDFS Read: 187655 HDFS Write: 223 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.93 sec   HDFS Read: 5500 HDFS Write: 78 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 320 msec
OK
Processed Meat  33.86   20311
Canned Foods    232.6   8568
Preserved Food  119.69  7201
Time taken: 45.43 seconds, Fetched: 3 row(s)
hive> 
```

Query 5 — Profitability by Product Sub-Category



```
hive> SELECT
    >   product_sub_category,
    >   ROUND(SUM(profit),2) AS total_profit
    > FROM india_retail
    > GROUP BY product_sub_category
    > ORDER BY total_profit DESC
    > LIMIT 10;
Query ID = cloudera_20251030002222_8c5720ff-68df-48a7-ade4-347da5f9fba3
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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_1761800063306_0025, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0025/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0025
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:22:26,180 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:22:33,526 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.0 sec
2025-10-30 00:22:40,956 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.31 sec
MapReduce Total cumulative CPU time: 2 seconds 310 msec
Ended Job = job_1761800063306_0025
Launching Job 2 out of 2
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_1761800063306_0026, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0026/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0026
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-10-30 00:22:49,906 Stage-2 map = 0%,  reduce = 0%
2025-10-30 00:22:56,252 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.77 sec
```

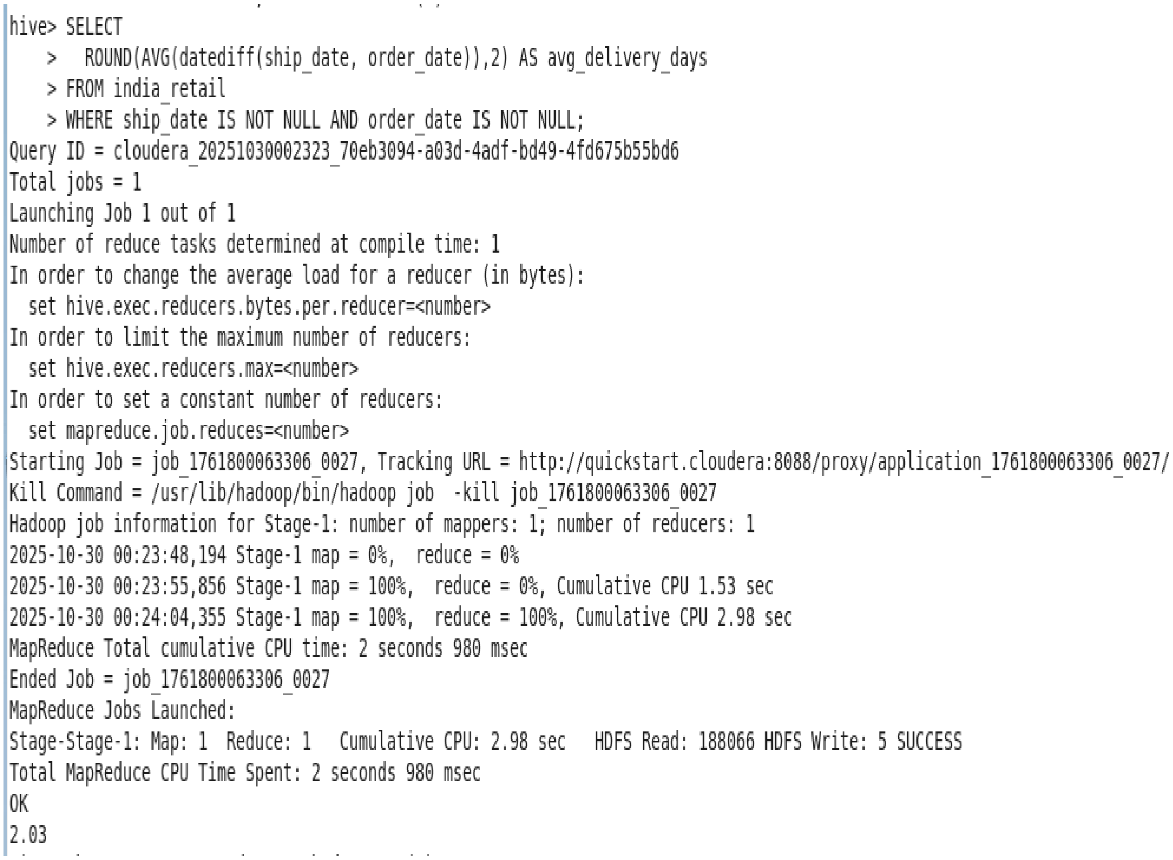


```
2025-10-30 00:22:40,956 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.31 sec
MapReduce Total cumulative CPU time: 2 seconds 310 msec
Ended Job = job_1761800063306_0025
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (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_1761800063306_0026, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0026/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0026
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-10-30 00:22:49,906 Stage-2 map = 0%,  reduce = 0%
2025-10-30 00:22:56,252 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.77 sec
2025-10-30 00:23:03,707 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.95 sec
MapReduce Total cumulative CPU time: 1 seconds 950 msec
Ended Job = job_1761800063306_0026
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.31 sec   HDFS Read: 186682 HDFS Write: 714 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.95 sec   HDFS Read: 5648 HDFS Write: 210 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 260 msec
OK
Quail Eggs      70894.13
Jelly Fish      34455.57
Sundried Tomatoes       28466.76
Marmalade       27410.73
Bacon   24145.42
Caviar  15464.67
Smoked Salmon   14156.69
Fresh Water Eel 13693.94
Wild Berry      13530.29
Assorted Fruits 11798.21
```

Insight:

Shows which sub-categories yield higher profit margins; guide pricing strategy.

Query 6 — Delivery Performance (Average Ship Delay)

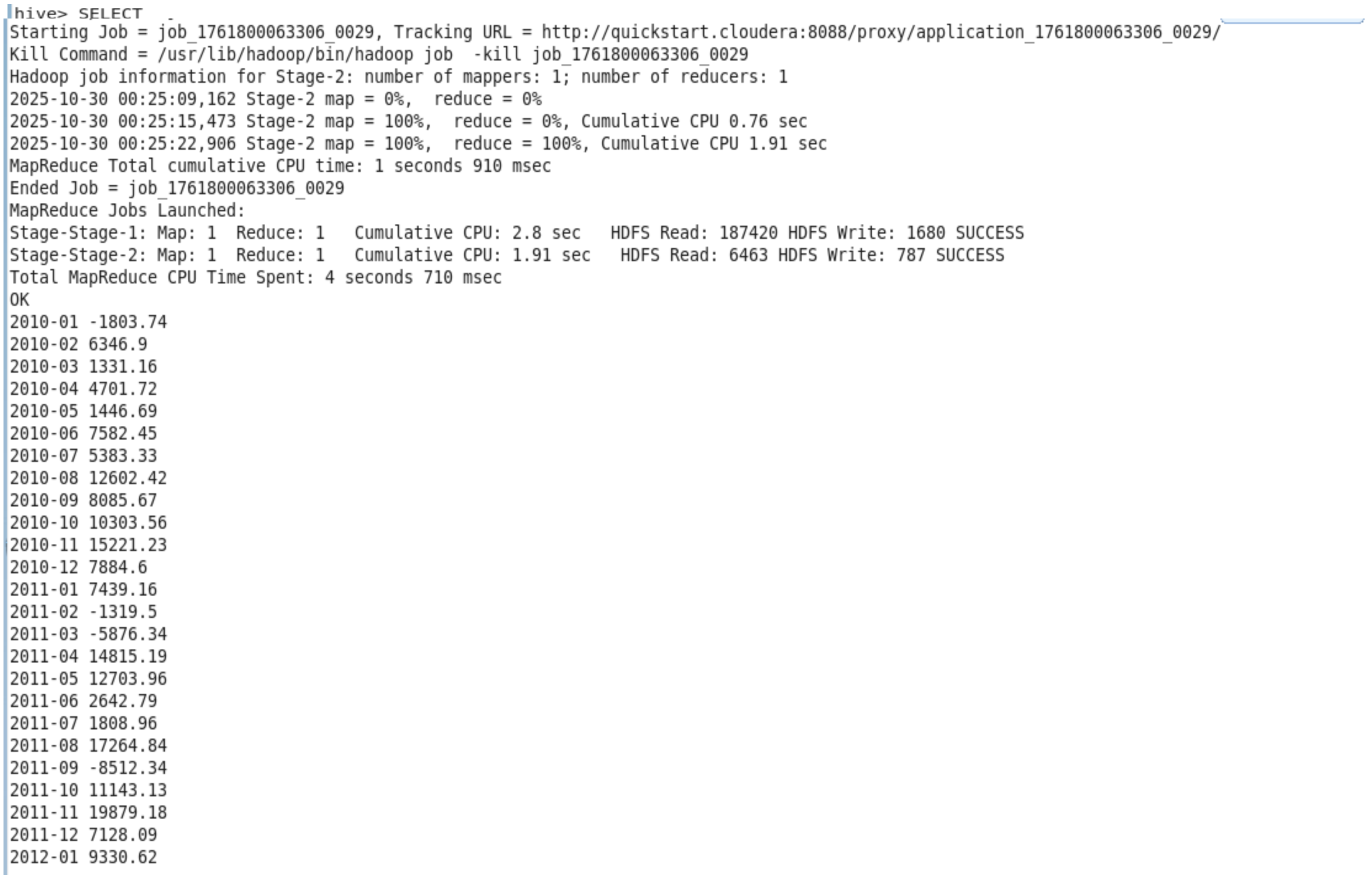


```
hive> SELECT
    >   ROUND(AVG(datediff(ship_date, order_date)),2) AS avg_delivery_days
    > FROM india_retail
    > WHERE ship_date IS NOT NULL AND order_date IS NOT NULL;
Query ID = cloudera_20251030002323_70eb3094-a03d-4adf-bd49-4fd675b55bd6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce 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_1761800063306_0027, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0027/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0027
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-30 00:23:48,194 Stage-1 map = 0%,  reduce = 0%
2025-10-30 00:23:55,856 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.53 sec
2025-10-30 00:24:04,355 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.98 sec
MapReduce Total cumulative CPU time: 2 seconds 980 msec
Ended Job = job_1761800063306_0027
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.98 sec   HDFS Read: 188066 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 980 msec
OK
2.03
```

Insight*:*

Average delivery time (≈ few days) can be used to monitor logistics efficiency.

Query 7 — Monthly Profit Trend



```
hive> SELECT
Starting Job = job_1761800063306_0029, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761800063306_0029/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761800063306_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-10-30 00:25:09,162 Stage-2 map = 0%,  reduce = 0%
2025-10-30 00:25:15,473 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.76 sec
2025-10-30 00:25:22,906 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.91 sec
MapReduce Total cumulative CPU time: 1 seconds 910 msec
Ended Job = job_1761800063306_0029
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.8 sec   HDFS Read: 187420 HDFS Write: 1680 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.91 sec   HDFS Read: 6463 HDFS Write: 787 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 710 msec
OK
2010-01 -1803.74
2010-02 6346.9
2010-03 1331.16
2010-04 4701.72
2010-05 1446.69
2010-06 7582.45
2010-07 5383.33
2010-08 12602.42
2010-09 8085.67
2010-10 10303.56
2010-11 15221.23
2010-12 7884.6
2011-01 7439.16
2011-02 -1319.5
2011-03 -5876.34
2011-04 14815.19
2011-05 12703.96
2011-06 2642.79
2011-07 1808.96
2011-08 17264.84
2011-09 -8512.34
2011-10 11143.13
2011-11 19879.18
2011-12 7128.09
2012-01 9330.62
```

Insight:

Parallel profit curve to sales trend — profit dips during low-sales months (e.g., May).

Data Visulization

Step 1: Import Libraries

In [10]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Display settings
plt.style.use('seaborn-v0_8')
sns.set_palette("pastel")
```

Step 2: Load the CSV File

In [11]:

```
data = pd.read_csv(r"E:\3 sem\big data project\INDIA_RETAIL_DATA_SIMPLIFIED.csv"
print("Data Loaded Successfully!")
print(data.head())
```

```
Data Loaded Successfully!
   Order_Date   Ship_Date    Product_Type Product_Sub_Category  Unit_Price  \
0  2010-01-02  2010-01-04  Processed Meat                Bacon       40.98   
1  2010-01-02  2010-01-09  Processed Meat       Fresh Water Eel      155.06   
2  2010-01-02  2010-01-04  Processed Meat        Smoked Salmon        9.11   
3  2010-01-02  2010-01-02  Processed Meat        Smoked Salmon       15.04   
4  2010-01-03  2010-01-04  Processed Meat            Foie Gras        4.26   

   QtyOrdered    Profit    Sales  
0           3  -19.0992   124.81  
1           8  845.6640  1225.60  
2           4   20.2996    34.41  
3          11  108.5163   157.27  
4          29    9.8200   122.23  
```

Step 3: Create Total Sales Column

In [12]:

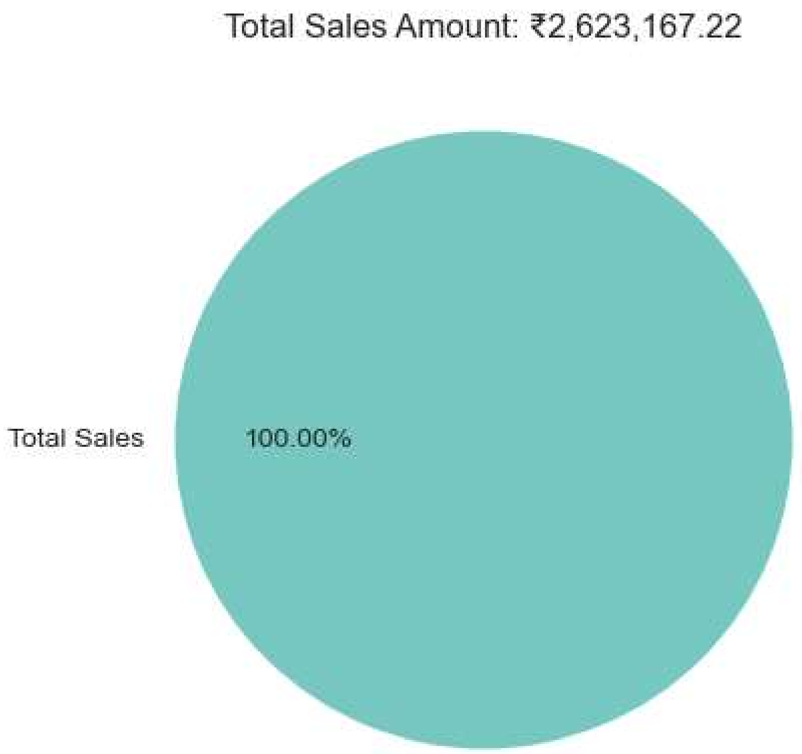
```
data["Total_Sales"] = data["QtyOrdered"] * data["Unit_Price"]
print("Total_Sales column created successfully!")
```

```
Total_Sales column created successfully!
```

Step 4: Total Sales Amount (Pie Chart)

In [13]:

```
total_sales = data["Total_Sales"].sum()
plt.figure(figsize=(5,5))
plt.pie([total_sales], labels=["Total Sales"], autopct='%1.2f%%', colors=["#76C7
plt.title(f"Total Sales Amount: ₹{total_sales:,.2f}")
plt.show()
```



```
In [14]: data["Total_Sales"] = data["QtyOrdered"] * data["Unit_Price"]
         print("Total_Sales column created successfully!")
         print(data.head())
```

```
Total_Sales column created successfully!
   Order_Date  Ship_Date    Product_Type Product_Sub_Category  Unit_Price  \
0  2010-01-02  2010-01-04  Processed Meat                Bacon       40.98
1  2010-01-02  2010-01-09  Processed Meat      Fresh Water Eel      155.06
2  2010-01-02  2010-01-04  Processed Meat        Smoked Salmon        9.11
3  2010-01-02  2010-01-02  Processed Meat        Smoked Salmon       15.04
4  2010-01-03  2010-01-04  Processed Meat            Foie Gras        4.26

   QtyOrdered    Profit    Sales  Total_Sales
0           3  -19.0992   124.81       122.94
1           8  845.6640  1225.60      1240.48
2           4   20.2996    34.41        36.44
3          11  108.5163   157.27       165.44
4          29    9.8200   122.23       123.54
```

Step 5: Total Sales by Product Type (Bar Graph)

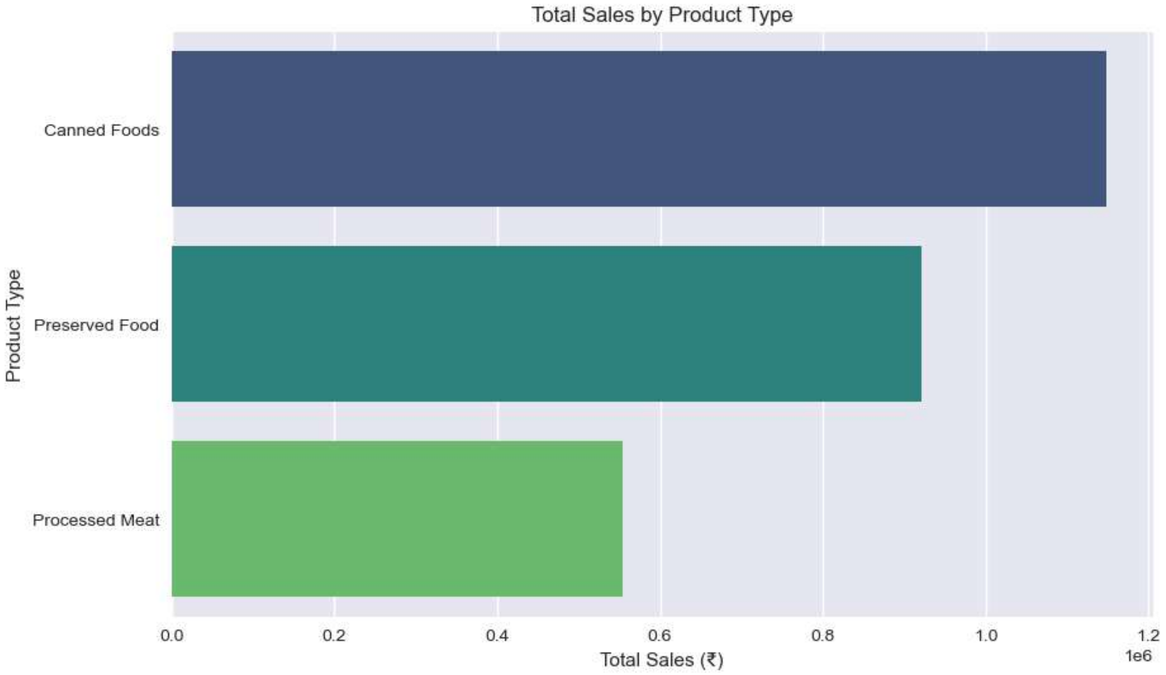
```
In [15]: sales_by_product = data.groupby("Product_Type")["Total_Sales"].sum().sort_values

         plt.figure(figsize=(10,6))
         sns.barplot(x=sales_by_product.values, y=sales_by_product.index, palette="viridi
         plt.title("Total Sales by Product Type")
         plt.xlabel("Total Sales (₹)")
         plt.ylabel("Product Type")
         plt.show()
```

```
C:\Users\himan\AppData\Local\Temp\ipykernel_9956\2993180878.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=sales_by_product.values, y=sales_by_product.index, palette="virid
is")
```



Step 6: Top 10 Sub-Categories by Sales

In [16]:

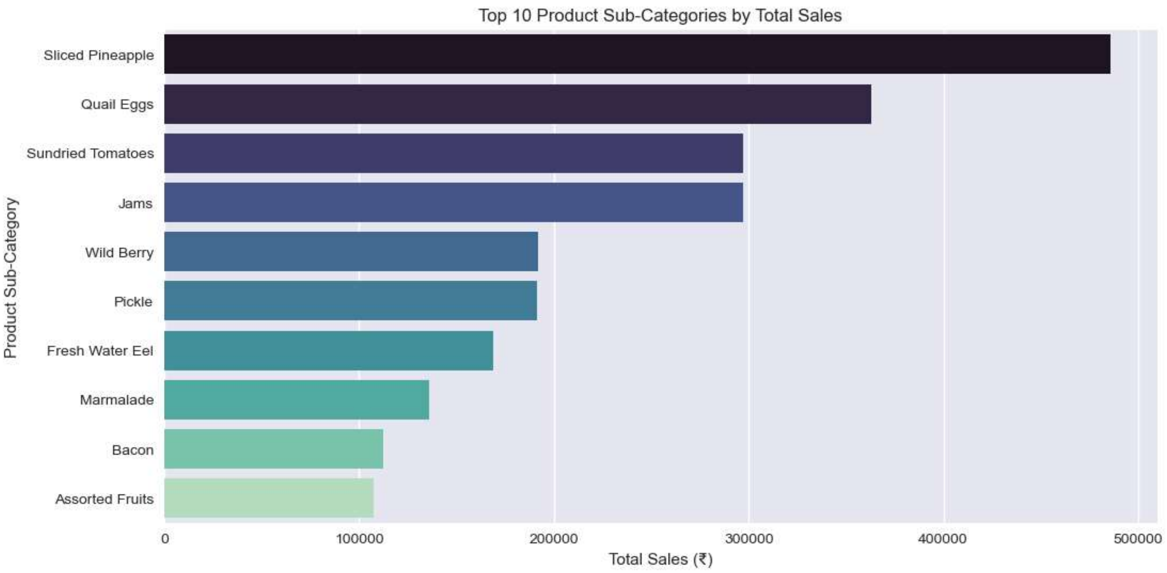
```
sales_by_subcategory = data.groupby("Product_Sub_Category")["Total_Sales"].sum()

plt.figure(figsize=(12,6))
sns.barplot(x=sales_by_subcategory.values, y=sales_by_subcategory.index, palette
plt.title("Top 10 Product Sub-Categories by Total Sales")
plt.xlabel("Total Sales (₹)")
plt.ylabel("Product Sub-Category")
plt.show()
```

```
C:\Users\himan\AppData\Local\Temp\ipykernel_9956\3308071622.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=sales_by_subcategory.values, y=sales_by_subcategory.index, palett
e="mako")
```

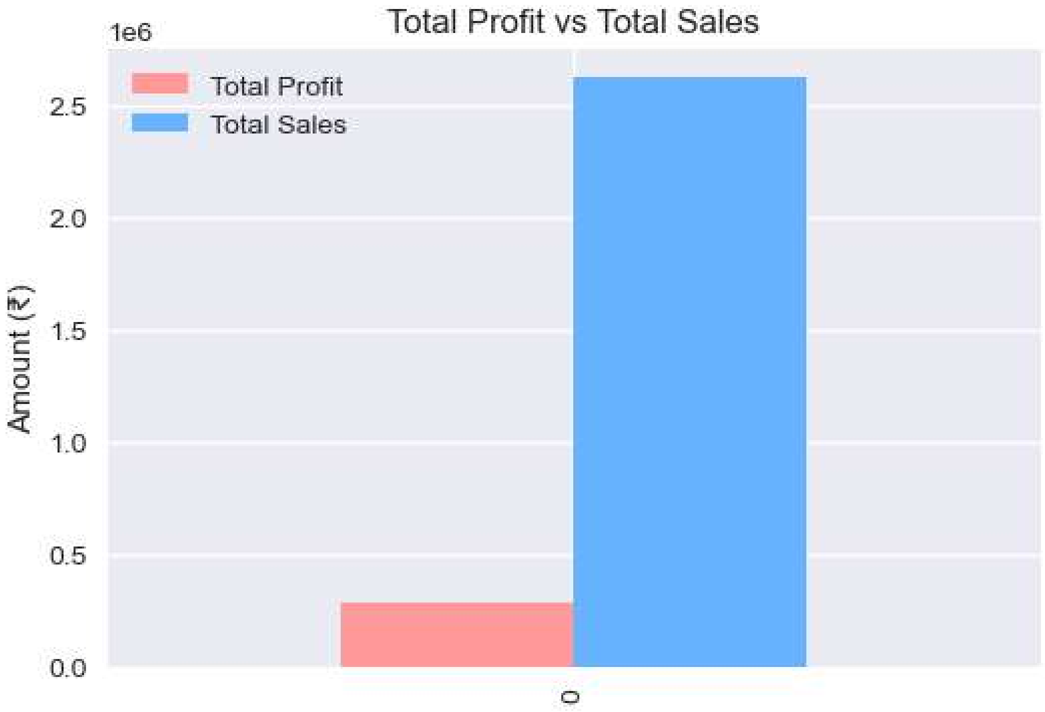


Step 7: Total Profit vs Total Sales (Comparative Bar Chart)

In [17]:

```
profit_sales = pd.DataFrame({
    "Total Profit": [data["Profit"].sum()],
    "Total Sales": [data["Total_Sales"].sum()]
})

profit_sales.plot(kind='bar', figsize=(6,4), color=["#FF9999","#66B2FF"])
plt.title("Total Profit vs Total Sales")
plt.ylabel("Amount (₹)")
plt.show()
```



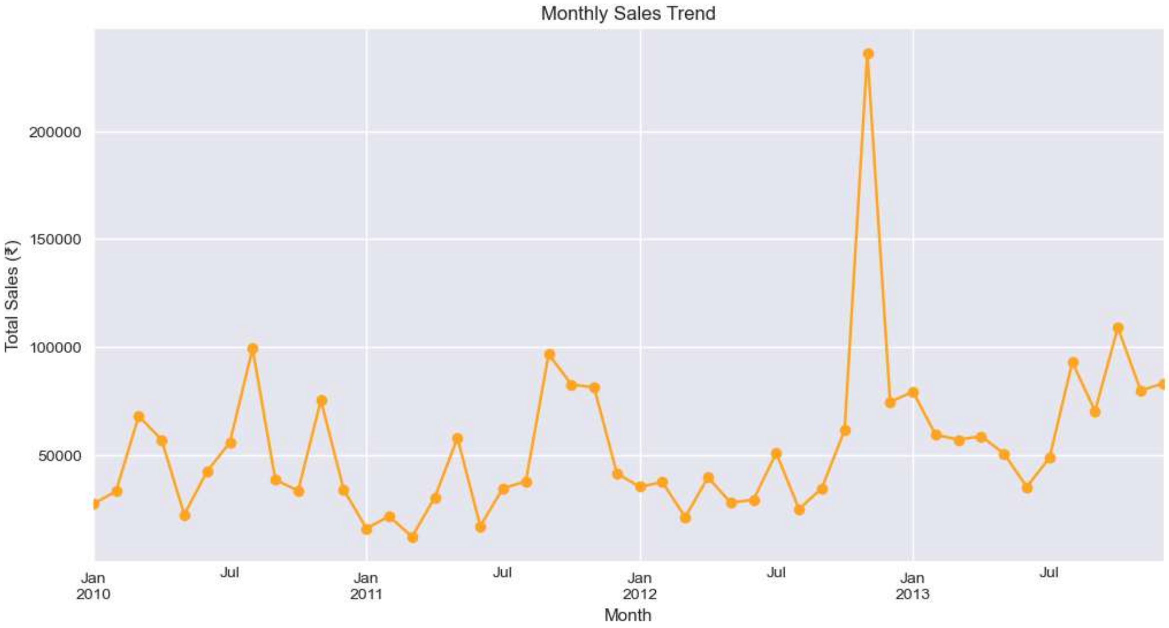
Step 8: Monthly Sales Trend (Line Graph)

In [18]:

```
data["Order_Date"] = pd.to_datetime(data["Order_Date"])
monthly_sales = data.groupby(data["Order_Date"].dt.to_period("M"))["Total_Sales"

plt.figure(figsize=(12,6))
monthly_sales.plot(kind='line', marker='o', color='#FFA726')
```

```
plt.title("Monthly Sales Trend")
plt.xlabel("Month")
plt.ylabel("Total Sales (₹)")
plt.grid(True)
plt.show()
```


Monthly Sales Trend
Total Sales (₹)
Jan 2010
Jul
Jan 2011
Jul
Jan 2012
Jul
Jan 2013
Jul
Month