E-COMMERCE BIG DATA ANALYSIS

Every pixel, every line of code, every idea—stronger when shared

MEET OUR TEAM

We are a team of data engineers, analysts, and developers driven by a shared passion for distributed computing and scalable solutions.
Each member brings deep technical expertise and a collaborative spirit that fuels our innovation

→

AGENDA

INTRODUCTION

With the growth of e-commerce platforms, users often struggle to find relevant products. To solve this, personalized recommendation systems are used to suggest items based on user behavior. These systems require heavy computation, making platforms like Hadoop and Spark ideal for implementation.

We used a real-world e-commerce dataset containing transaction records like Invoice No, Product Code, Description, Quantity, Price, Customer ID, and Country. The original dataset had over 540,000 entries, which were reduced to 358,000 after cleaning with Pandas, including removal of null values and duplicates.

STATEMENT OF WORK

1. Analyze e-commerce sales data sourced from SQL systems and CSV files.

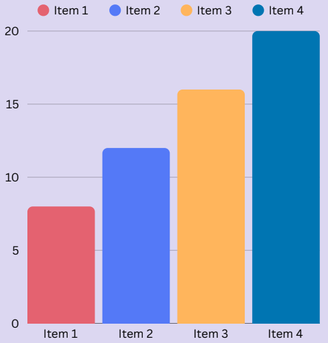
2. Use pandas to extract, clean, and transform the data for analysis.

3. Leverage Hive, Spark, and PySpark for large-scale data processing.

4. Visualize insights through Power BI dashboards for business reporting.

5. Focus on query performance and optimizations.

TECHNOLOGIES USED



HADOOP

Hadoop is Software used to analyze Data and HDFS Stores Data

1

SPARK

Used to optimize data workflows

2

HIVE

Hive is a data warehouse tool in Hadoop for querying and analyzing large datasets

3

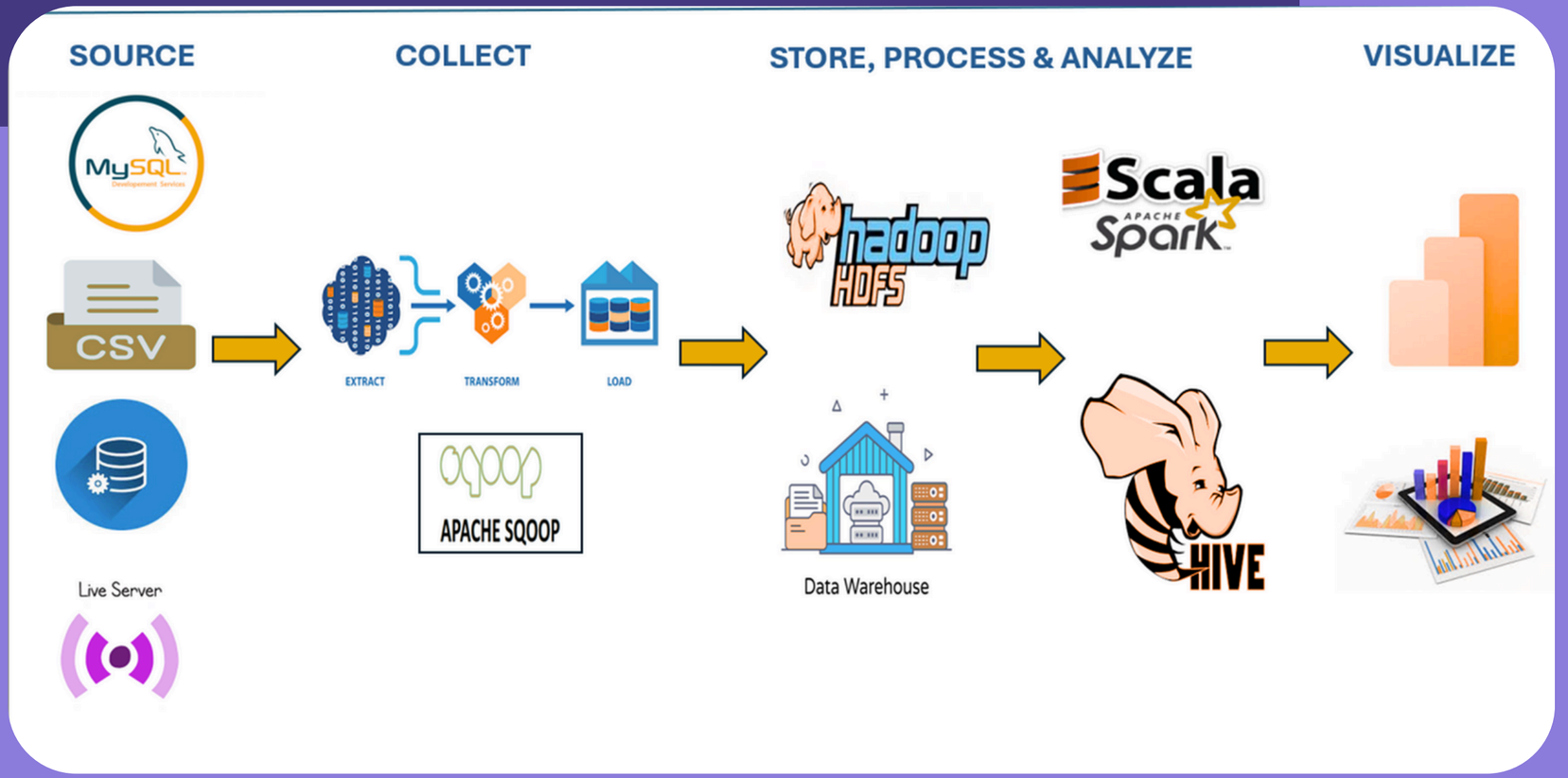
POWER BI

Reporting Tool used to creates Reports

4

ECOMMERCE - BIG DATA ECOSYSTEM

- Start by collecting data from multiple sources like sensors, logs, or APIs.
- Use tools like Sqoop or Flume to ingest and stream the data efficiently.
- Process and store it using Hadoop, Spark, or cloud data lakes.
- Analyze patterns through ML models or BI tools like Tableau or Power BI.
- Generate insights and create a feedback loop to refine future decisions.



CLEANED DATA.CSV FILE

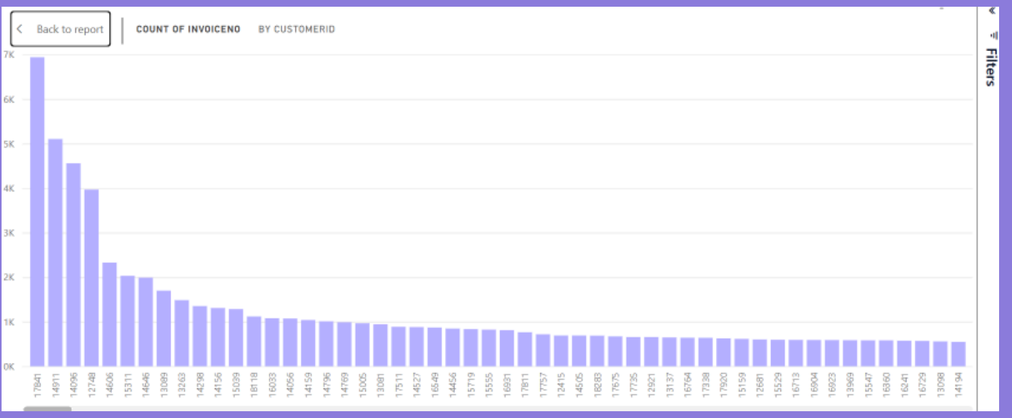
DATA DESCRIPTION

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	71053	WHITE METAL LANTERN	6	12-01-2010 08:26	3.39	17850	United Kingdo
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12-01-2010 08:26	7.65	17850	United Kingdo
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12-01-2010 08:26	4.25	17850	United Kingdo
536366	22633	HAND WARMER UNION JACK	6	12-01-2010 08:28	1.85	17850	United Kingdo
536366	22632	HAND WARMER RED POLKA DOT	6	12-01-2010 08:28	1.85	17850	United Kingdo
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12-01-2010 08:34	1.69	13047	United Kingdo
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12-01-2010 08:34	2.1	13047	United Kingdo
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12-01-2010 08:34	2.1	13047	United Kingdo
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12-01-2010 08:34	3.75	13047	United Kingdo
536367	22310	IVORY KNITTED MUG COSY	6	12-01-2010 08:34	1.65	13047	United Kingdo
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12-01-2010 08:34	4.25	13047	United Kingdo
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12-01-2010 08:34	4.95	13047	United Kingdo
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12-01-2010 08:34	9.95	13047	United Kingdo
536367	21754	HOME BUILDING BLOCK WORD	3	12-01-2010 08:34	5.95	13047	United Kingdo
536367	21755	LOVE BUILDING BLOCK WORD	3	12-01-2010 08:34	5.95	13047	United Kingdo
536367	21777	RECIPE BOX WITH METAL HEART	4	12-01-2010 08:34	7.95	13047	United Kingdo
536367	48187	DOORMAT NEW ENGLAND	4	12-01-2010 08:34	7.95	13047	United Kingdo
536368	22960	JAM MAKING SET WITH JARS	6	12-01-2010 08:34	4.25	13047	United Kingdo
536368	22913	RED COAT RACK PARIS FASHION	3	12-01-2010 08:34	4.95	13047	United Kingdo

Q1.

Find the no. of customers in each invoice number

```python
# Step 1: Import required modules
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count

# Step 2: Create or get Spark session
spark = SparkSession.builder.appName("InvoiceCustomerCount").getOrCreate()

# Step 3: Load the dataset
df = spark.read.csv( path: "C:/Users/saldubey/Downloads/cleaned_data 1.csv", header=True, inferSchema=True).repartition((4))

# Step 4: Group by InvoiceNo and count CustomerId
grouped_df = df.groupBy("InvoiceNo").agg(count("CustomerId").alias("CustomerCount"))

# Step 5: Rename columns and order by count descending
final_df = grouped_df.select(
    col("InvoiceNo").alias("Invoice_Number"),
    col("CustomerCount")
).orderBy( *cols: "CustomerCount", ascending=False)

# Step 6: Show result
final_df.show(10)

# Step 7: Check number of partitions
print("Number of partitions:", final_df.rdd.getNumPartitions())

# Step 8: Save result partitioned by Invoice_Number
final_df.write.partitionBy("Invoice_Number").mode("overwrite").parquet("output2/query_1_invoice_customer")
```



Name	Date modified	Type
Invoice_Number=536365	23-07-2025 15:16	File folder
Invoice_Number=536366	23-07-2025 15:16	File folder
Invoice_Number=536367	23-07-2025 15:16	File folder
Invoice_Number=536368	23-07-2025 15:16	File folder
Invoice_Number=536369	23-07-2025 15:16	File folder
Invoice_Number=536370	23-07-2025 15:16	File folder

Invoice_Number	CustomerCount
579196	485
576339	485
580727	472
578270	402
573576	385
567656	373
567183	365
575607	335
571441	323
570488	313

only showing top 10 rows

Q2.
filter out the column where customerID 15862

```scala
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object query5 {
  def main(args: Array[String]): Unit = {

    // Step 1: Create SparkSession
    val spark = SparkSession.builder().appName("Query5_CustomerID_15862").master("local[*]").getOrCreate()

    // Step 2: Load data (CSV without repartitioning)
    val df = spark.read
      .option("header", "true")
      .option("inferSchema", "true")
      .csv("C:/Users/amishr85/Downloads/cleaned_data 1.csv")

    // Step 3: Filter rows where CustomerID == 15862
    val filtered = df.filter(col("CustomerID") === 15862)

    // Step 4: Select only required columns
    val result = filtered.select(
      "InvoiceNo", "StockCode", "Description", "Quantity",
      "InvoiceDate", "UnitPrice", "CustomerID", "Country"
    )

    // Step 6: Show output
    result.show(10,false)

    // Step 7: Save as Parquet
    result.write
      .mode("overwrite")
      .parquet("output/query5_customer_15862")
  }
```

._SUCCESS.crc	23-07-2025 14:35	CRC File
.part-00000-6a755dad-b214-4984-87b0-0274...	23-07-2025 14:35	CRC File
.part-00002-6a755dad-b214-4984-87b0-0274...	23-07-2025 14:35	CRC File
.part-00003-6a755dad-b214-4984-87b0-0274...	23-07-2025 14:35	CRC File
.part-00006-6a755dad-b214-4984-87b0-0274...	23-07-2025 14:35	CRC File
_SUCCESS	23-07-2025 14:35	File
part-00000-6a755dad-b214-4984-87b0-02741...	23-07-2025 14:35	PARQUET File
part-00002-6a755dad-b214-4984-87b0-02741...	23-07-2025 14:35	PARQUET File
part-00003-6a755dad-b214-4984-87b0-02741...	23-07-2025 14:35	PARQUET File
part-00006-6a755dad-b214-4984-87b0-02741...	23-07-2025 14:35	PARQUET File

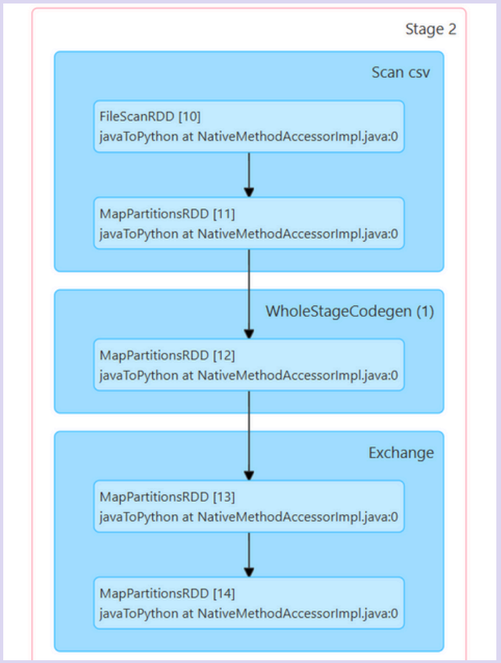
InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536401	22110	BIRD HOUSE HOT WATER BOTTLE	1	12/1/2010 11:21	2.55	15862.0	United Kingdom
536401	22098	BOUDOIR SQUARE TISSUE BOX	1	12/1/2010 11:21	1.25	15862.0	United Kingdom
536401	22100	SKULLS SQUARE TISSUE BOX	2	12/1/2010 11:21	1.25	15862.0	United Kingdom
536401	22766	PHOTO FRAME CORNICE	1	12/1/2010 11:21	2.95	15862.0	United Kingdom
536401	22451	SILK PURSE BABUSHKA RED	1	12/1/2010 11:21	3.35	15862.0	United Kingdom
536401	22549	PICTURE DOMINOES	1	12/1/2010 11:21	1.45	15862.0	United Kingdom
536401	84744	S/6 SEW ON CROCHET FLOWERS	1	12/1/2010 11:21	1.25	15862.0	United Kingdom
536401	21328	BALLOONS WRITING SET	1	12/1/2010 11:21	1.65	15862.0	United Kingdom
536401	22961	JAM MAKING SET PRINTED	4	12/1/2010 11:21	1.45	15862.0	United Kingdom
536401	22473	TV DINNER TRAY VINTAGE PAISLEY	1	12/1/2010 11:21	4.95	15862.0	United Kingdom

Q3.

Find the total number of people from france.

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

# Step 1: Start Spark Session
spark = SparkSession.builder \
    .appName("Query3_Count_Customers_France") \
    .getOrCreate()

# Step 2: Load cleaned CSV with partitioning
df = spark.read.csv( path: "C:/Users/saldubey/Downloads/cleaned_data 1.csv", header=True, inferSchema=True).repartition(4)

# Step 3: Select only necessary columns
df = df.select("CustomerID", "Country")

# Step 4: Filter records where Country is France
df_france = df.filter(col("Country") == "France")

# Step 5: Count distinct CustomerID (people from France)
result = df_france.select("CustomerID").distinct()

# Step 6: Show number of partitions before coalesce
print("Number of Partitions: ", result.rdd.getNumPartitions())

# Step 7: Show output
result.show(10)

# Step 8: Count and print final total
print("Total number of people from France:", result.count())

# Step 9: Save as parquet file
result.write.mode("overwrite").parquet("C:/Users/salodubey/IdeaProjects/ECOMMERCE PROJECT/output/query3_people_france")
```

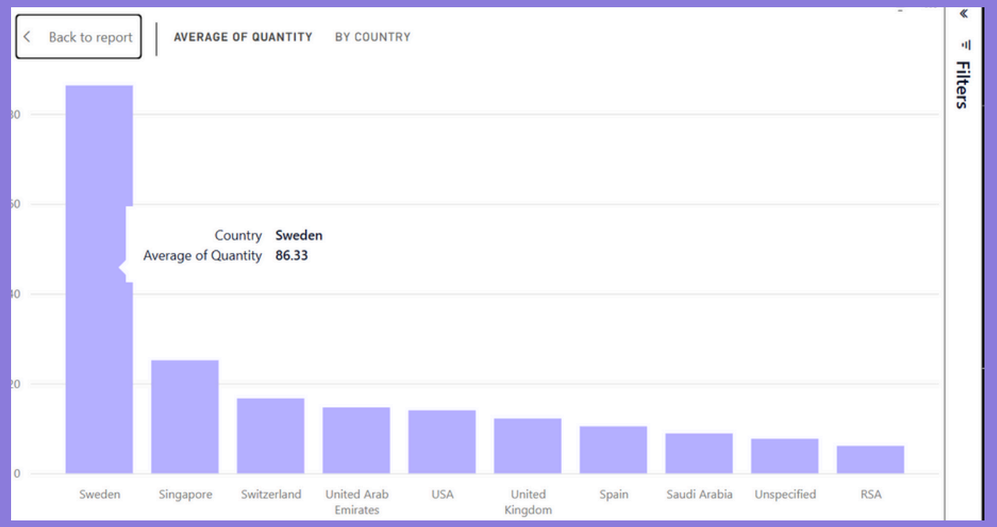
```
Number of Partitions:  1
+----------+
|CustomerID|
+----------+
|   12493.0|
|   12690.0|
|   12616.0|
|   12583.0|
|   12700.0|
|   12562.0|
|   12437.0|
|   12637.0|
|   12657.0|
|   12726.0|
+----------+
only showing top 10 rows

Total number of people from France: 87
```



Q4.

Display the average quantity of each country.

```python
# Step 1: Import required PySpark libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, round

# Step 2: Initialize SparkSession
spark = SparkSession.builder \
    .appName("Query 5 - Average Quantity by Country") \
    .config("spark.sql.shuffle.partitions", "4") \
    .getOrCreate()

# Step 3: Read data from CSV without cleaning
df = spark.read.csv( path: "C:/Users/saldubey/Downloads/cleaned_data 1.csv", header=True, inferSchema=True).repartition((4))

# Step 4: Select required columns
df_selected = df.select("Country", "Quantity")

# Step 5: Calculate average quantity per country (rounded)
avg_quantity_df = df_selected.groupBy("Country") \
    .agg(round(avg("Quantity"), scale: 2).alias("Average_Quantity"))

# Step 6: Print number of partitions
print("Number of partitions:", avg_quantity_df.rdd.getNumPartitions())

# Step 7: Show output
avg_quantity_df.show( n: 10,truncate=False)

# Step 8: Save output partitioned and coalesced
avg_quantity_df.write.partitionBy("Country") \
    .mode("overwrite") \
    .parquet("output2/query_5_avg_quantity_by_country")
```




```
Number of partitions: 1
+--------------------+----------------+
|Country             |Average_Quantity|
+--------------------+----------------+
|EIRE                |20.04           |
|Australia           |73.29           |
|Austria             |12.82           |
|Israel              |16.57           |
|Norway              |18.17           |
|United Arab Emirates|14.68           |
|Iceland             |13.19           |
|European Community  |8.42            |
|Sweden              |86.33           |
|Netherlands         |86.91           |
+--------------------+----------------+
only showing top 10 rows
```

Q5.

Display the top 5 countries with highest quantity in descending order

```
[training@localhost ~]$ hadoop fs -mkdir /user/cloudera/final_output
[training@localhost ~]$ hadoop fs -put /home/training/cleaned_data.csv /user/cloudera/final_outpu
t
[training@localhost ~]$
[training@localhost ~]$ hadoop fs -ls /user/cloudera/final_output
Found 1 items
-rw-r--r--   1 training supergroup   30887039 2025-07-23 22:25 /user/cloudera/final_output/cleane
d_data.csv
```

```
[training@localhost ~]$ hive
Hive history file=/tmp/training/hive_job_log_training_202507232230_15731751.txt
hive> CREATE TABLE sales_data_one (
    >   InvoiceNo STRING,
    >   StockCode STRING,
    >   Description STRING,
    >   Quantity INT,
    >   InvoiceDate STRING,
    >   UnitPrice INT,
    >   CustomerID STRING,
    >   Country STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 2.825 seconds
hive>
    > LOAD DATA INPATH '/user/cloudera/final_output/cleaned_data.csv'into table sales_data_one;
Loading data to table default.sales_data_one
OK
```

```
hive> create table temp_ans1 as SELECT Country, SUM(Quantity) AS to
ne GROUP BY Country
    > ORDER BY total_quantity DESC
    > LIMIT 5;
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data siz
```

```
hive> select * from temp_ans1 limit 1
OK
United Kingdom  3792217
Netherlands     192079
EIRE    130641
Germany 111151
France  106248
Time taken: 2.871 seconds
hive>
```

Q6.

find out the customerid who have count more than 500 InvoiceNo

```scala
import org.apache.spark.sql.{SparkSession}
import org.apache.spark.sql.functions._

object Query8 {
  def main(args: Array[String]): Unit = {
    // Step 1: Start Spark Session
    val spark = SparkSession.builder().appName("Query8_Customer_MoreThan500_Invoice").master("local[*]")
      .getOrCreate()

    import spark.implicits._

    // Step 2: Load the cleaned CSV with early repartitioning
    val df = spark.read.option("header", "true").option("inferSchema", "true")
      .csv("C:/Users/amishr85/Downloads/cleaned_data 1.csv")
      .repartition(4) // early partitioning

    // Step 3: Select only required columns to optimize
    val selectedDF = df.select("CustomerID", "InvoiceNo")

    // Step 4: Group by CustomerID and count InvoiceNo
    val result = selectedDF.groupBy("CustomerID").agg(count("InvoiceNo").alias("InvoiceCount"))
      .filter($"InvoiceCount" > 500) // filter after aggregation for optimization

    // Step 5: Reduce file count for output
    val output = result.coalesce(1)

    // Step 7: Show output
    output.show(10, false)

    // Step 6: Save to Parquet format
    output.write.mode("overwrite").parquet("output/query8_customers_gt_500_invoices")
```


```
+--------------+-------------+
|Invoice_Number|CustomerCount|
+--------------+-------------+
|        579196|          485|
|        576339|          485|
|        580727|          472|
|        578270|          402|
|        573576|          385|
|        567656|          373|
|        567183|          365|
|        575607|          335|
|        571441|          323|
|        570488|          313|
+--------------+-------------+
only showing top 10 rows

Number of partitions: 1
```

Summary

This project analyzes e-commerce sales data using an end-to-end analytics pipeline. Data is extracted from SQL sources, processed through ETL tools, analyzed with big data technologies like Hive and Spark, and visualized using Power BI to uncover key business insights.

LET’S TAKE A MOMENT TO THANK OUR TRAINER

VENKAT SIR

We are truly grateful for the incredible guidance and support you’ve provided throughout this training journey. Your ability to simplify complex concepts, share valuable real-world insights, and instill teamwork values has made a lasting impact on all of us.

THANK YOU

Thank you for your time and attention.
We look forward to any feedback or questions you may have.