1.from pyspark.sql import SparkSession

# Create a SparkSession

spark = SparkSession.builder.appName("Read CSV").getOrCreate()

# Read the CSV file into a DataFrame

df = spark.read.csv("employees.csv", header=True, inferSchema=True)

# Display the top 10 records

df.show(10)

2.from pyspark.sql import SparkSession

from pyspark.sql.functions import sum, desc

# Create a SparkSession

spark = SparkSession.builder.appName("Calculate Total Revenue").getOrCreate()

# Calculate the total revenue for each product

total\_revenue\_df = sales\_data.groupBy("product\_name").agg(sum("revenue").alias("total\_revenue"))

# Sort the DataFrame in descending order of total revenue

sorted\_df = total\_revenue\_df.orderBy(desc("total\_revenue"))

# Display the result

sorted\_df.show()

3.from pyspark.sql import SparkSession

from pyspark.sql.functions import col

# Create a SparkSession

spark = SparkSession.builder.appName("Read JSON").getOrCreate()

# Read the JSON file into a DataFrame

df = spark.read.json("students.json")

# Filter the DataFrame to include only students whose age is greater than 18

filtered\_df = df.filter(col("age") > 18)

# Display the filtered DataFrame

filtered\_df.show()

4.from pyspark.sql import SparkSession

from pyspark.sql.functions import avg

# Create a SparkSession

spark = SparkSession.builder.appName("Calculate Average Transaction Amount").getOrCreate()

# Calculate the average transaction amount for each user

avg\_amount\_df = transactions.groupBy("user\_id").agg(avg("amount").alias("average\_amount"))

# Display the result

avg\_amount\_df.show()

5. from pyspark.sql import SparkSession

from pyspark.sql.functions import hour

from pyspark.sql.types import TimestampType

# Create a SparkSession

spark = SparkSession.builder.appName("Count Events per Hour").getOrCreate()

# Convert the "timestamp" column to a timestamp type if it's not already

logs = logs.withColumn("timestamp", logs["timestamp"].cast(TimestampType()))

# Extract the hour from the timestamp column

logs = logs.withColumn("hour", hour("timestamp"))

# Count the number of events that occurred in each hour

event\_counts = logs.groupBy("hour").count().orderBy("hour")

# Display the result

event\_counts.show()

6.from pyspark.sql import SparkSession

# Create a SparkSession

spark = SparkSession.builder.appName("Retrieve Customers").getOrCreate()

# Register the "customers\_df" DataFrame as a temporary view

customers\_df.createOrReplaceTempView("Customers")

# Use SQL query to retrieve the desired customers

result\_df = spark.sql("""

SELECT \*

FROM Customers

WHERE age > 25

AND customer\_id IN (

SELECT customer\_id

FROM Purchases

)

""")

# Display the result

result\_df.show()

7.from pyspark.sql import SparkSession

from pyspark.sql.functions import count

from pyspark.sql.window import Window

from pyspark.sql.functions import desc

# Create a SparkSession

spark = SparkSession.builder.appName("Count Orders").getOrCreate()

# Calculate the total number of orders

order\_counts\_df = orders\_df.groupBy("customer\_id").agg(count("order\_id").alias("order\_count"))

# Sort the DataFrame by the number of orders in descending order

sorted\_df = order\_counts\_df.orderBy(desc("order\_count"))

# Display the result

sorted\_df.show()

8.from pyspark.sql import SparkSession

# Create a SparkSession

spark = SparkSession.builder.appName("Retrieve Out of Stock Products").getOrCreate()

# Filter the DataFrame to include only out-of-stock products

out\_of\_stock\_df = products\_df.filter(products\_df["stock"] == 0)

# Select the names of the out-of-stock products

out\_of\_stock\_names = out\_of\_stock\_df.select("name")

# Display the result

out\_of\_stock\_names.show()

9.from pyspark.sql import SparkSession

from pyspark.sql.functions import avg

# Create a SparkSession

spark = SparkSession.builder.appName("Calculate Average Price by Category").getOrCreate()

# Calculate the average price of products

average\_price\_df = products\_df.groupBy("category").agg(avg("price").alias("average\_price"))

# Display the result

average\_price\_df.show()

10.from pyspark.sql import SparkSession

from pyspark.sql.functions import sum, desc

# Create a SparkSession

spark = SparkSession.builder.appName("Retrieve Top Customers").getOrCreate()

# Calculate the total amount spent by each customer

total\_amount\_df = purchases\_df.groupBy("customer\_id").agg(sum("amount").alias("total\_amount"))

# Sort the DataFrame by the total amount in descending order

sorted\_df = total\_amount\_df.orderBy(desc("total\_amount"))

# Retrieve the top 5 customers

top\_customers\_df = sorted\_df.limit(5)

# Display the result

top\_customers\_df.show()