Q1. Write a PySpark code to read a CSV file named "employees.csv" containing the following columns: "employee\_id", "name", "age", "department". Display the top 10 records from the DataFrame.

Ans. from pyspark.sql import SparkSession

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

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

df.show(10)

Q2. Given a PySpark DataFrame named "sales\_data" with columns "product\_name" and "revenue", write a code to calculate the total revenue for each product and display the result in descending order.

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import sum

from pyspark.sql.window import Window

from pyspark.sql.functions import desc

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

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

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

total\_revenue\_df.show()

Q3.  Write a PySpark code to read a JSON file named "students.json" containing student records with the following schema: "name" (string), "age" (integer), "grade" (string). Filter the DataFrame to include only students whose age is greater than 18.

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import col

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

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

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

filtered\_df.show()

Q4. Consider a PySpark DataFrame named "transactions" with columns "transaction\_id", "user\_id", and "amount". Write a code to calculate the average transaction amount for each user and display the result

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import avg

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

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

average\_amount\_df.show()

Q5.  Given a PySpark DataFrame named "logs" with columns "timestamp" (timestamp) and "event" (string), write a code to count the number of events that occurred in each hour and display the result sorted by the hour

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import hour

from pyspark.sql.functions import count

spark = SparkSession.builder.appName("Event Count by Hour").getOrCreate()

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

event\_count\_df = logs.groupBy("hour").agg(count("event").alias("event\_count"))

sorted\_df = event\_count\_df.orderBy("hour")

sorted\_df.show()

Q6. Retrieve all the customers from the "Customers" table whose age is greater than 25 and have made at least one purchase.

Ans. from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Customer Analysis").getOrCreate()

customers\_df.createOrReplaceTempView("Customers")

query = """

SELECT \*

FROM Customers

WHERE age > 25 AND customer\_id IN (

SELECT DISTINCT customer\_id

FROM Purchases

)

"""

result\_df = spark.sql(query)

result\_df.show()

Q7. Find the total number of orders placed by each customer and display the results in descending order of the number of orders

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import count

from pyspark.sql.window import Window

from pyspark.sql.functions import desc

spark = SparkSession.builder.appName("Order Analysis").getOrCreate()

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

ordered\_df = order\_count\_df.orderBy(desc("order\_count"))

ordered\_df.show()

Q8. Retrieve the names of all products that are currently out of stock from the "Products" table.

Ans. from pyspark.sql import SparkSession

# Create a SparkSession

spark = SparkSession.builder.appName("Product Analysis").getOrCreate()

products\_df.createOrReplaceTempView("Products")

query = """

SELECT product\_name

FROM Products

WHERE quantity = 0

"""

out\_of\_stock\_products\_df = spark.sql(query)

out\_of\_stock\_products\_df.show()

Q9. Calculate the average price of all products in each category and display the results along with the category name.

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import avg

spark = SparkSession.builder.appName("Product Analysis").getOrCreate()

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

average\_price\_df.show()

Q10.  Retrieve the top 5 customers who have spent the highest total amount on purchases.

Ans. from pyspark.sql import SparkSession

from pyspark.sql.functions import avg

spark = SparkSession.builder.appName("Product Analysis").getOrCreate()

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

average\_price\_df.show()