Spark-shell – To start the spark application

Table 1:- (ListOfOrders)

```
val df=spark.read.option("header","true").csv("C:/dataset/ListOfOrders.csv")
df.createOrReplaceTempView("ListOfOrders")
val res22=spark.sql("select * from ListOfOrders")
res22.show()
res22.write.option("header","true").csv("C:/dataset/ListOfOrders.csv")
```

Table 2 :- (Order Details)

```
val df2=spark.read.option("header","true").csv("C:/dataset/Order Details.csv")
df2.createOrReplaceTempView("OrderDetails")
val res2=spark.sql("select * from OrderDetails")
res2.show()
res2.write.option("header","true").csv("C:/dataset/OrderDetails.csv")
```

Table 3:- (SalesTarget)

```
val df3 = spark.read.option("header","true").csv("C:/dataset/Sales target.csv")
df3.createOrReplaceTempView("SalesTarget")
val res3 = spark.sql("SELECT * FROM SalesTarget")
res3.show()
res3.write.option("header", "true").csv("C:/dataset/SalesTarget.csv")
```

1.Write a SQL query to count orders by state

```
val res23=spark.sql("SELECT State, COUNT(*) as TotalOrders FROM ListOfOrders
GROUP BY State")
res23.show()
```

2. Write a SQL query to find total money that was spent by the customers for each sub category in category

```
val joinComdition1="ListOfOrders.`Order ID`=OrderDetails.`Order ID`"
val joincond=spark.sql(
        s"""
        SELECT *
         FROM ListOfOrders
       INNER JOIN OrderDetails
```

```
      ON $joinComdition1
    """)
joincond.createOrReplaceTempView("list_of_order_details")
val query = """
    SELECT CustomerName, Category, `Sub-Category`, SUM(Amount) AS TotalAmountSpend
    FROM list_of_order_details
    GROUP BY CustomerName, Category, `Sub-Category`
    ORDER BY TotalAmountSpend
  """
val resultDF = spark.sql(query)
resultDF.show()
```

3. Write a SQL query to retrieve information about orders, order details, and sales targets for a specific category and month.

```
val df=spark.read.option("header","true").csv("C:/dataset/ListOfOrders.csv")
df.createOrReplaceTempView("ListOfOrders")
val df2=spark.read.option("header","true").csv("C:/dataset/Order Details.csv")
df2.createOrReplaceTempView("OrderDetails")
val df3 = spark.read.option("header","true").csv("C:/dataset/Sales target.csv")
df3.createOrReplaceTempView("SalesTarget")
import org.apache.spark.sql.functions._
val dfWithFormattedDate = df.withColumn("Order Date", to_date(col("Order Date"), "dd-MM-yyyy"))
val df333 = df3.withColumn("Month of Order Date", to_date(last_day(to_date(col("Month of Order Date"), "MMM-yy"))))
df333.createOrReplaceTempView("NewSalesTarget")
spark.sql(
    """
    SELECT
      LO.`Order ID`,
      LO.`Order Date`,
      LO.CustomerName,
```

```
LO.State,
LO.City,
OD.Amount,
OD.Profit,
OD.Quantity,
OD.Category AS OrderCategory,
OD.`Sub-Category`,
ST.`Month Of Order Date`,
ST.Target
FROM  df1 LO
LEFT JOIN OrderDetails OD ON LO.`Order ID` = OD.`Order ID`
LEFT JOIN NewSalesTarget ST ON MONTH(LO.`Order Date`) = MONTH(ST.`Month Of Order Date`) AND OD.Category = ST.Category
WHERE OD.Category = 'Electronics' AND MONTH(LO.`Order Date`) = 1
"""
).show(Int.MaxValue, false)
```

4. Write an SQL query to find the top 3 customers with the highest total profit for each category and month, including the details of their orders.

A)
```
val Query =
"""
WITH RankedProfits AS (
SELECT
LO.CustomerName,
OD.Category,
MONTH(LO.`Order Date`) AS Month,
OD.`Order ID`,
OD.Profit,
ROW_NUMBER() OVER (PARTITION BY OD.Category, MONTH(LO.`Order Date`) ORDER BY OD.Profit DESC) AS Rank
FROM
df1 LO
```

```
    JOIN
      OrderDetails OD
    ON
      LO.`Order ID` = OD.`Order ID`
  )
  SELECT
    RP.CustomerName,
    RP.Category,
    RP.Month,
    RP.`Order ID`,
    RP.Profit
  FROM
    RankedProfits RP
  WHERE
    RP.Rank <= 3
""".stripMargin
val result = spark.sql(Query)
result.show()
```

6. Write an SQL to calculate the total profit per customer for orders placed in a specific month. create a user-defined function that takes an OrderID as a parameter and returns the profit for that order. Use this function in a query to calculate the total profit per customer for orders placed in a specific month.

A)

```
import org.apache.spark.sql.functions.udf
val decimalDf = df2.withColumn("Profit", col("Profit").cast(DecimalType(38, 18)))
val decimalDf1 = df2.withColumn("Amount", col("Amount").cast(DecimalType(38, 18)))
val decimalDf = decimalDf1.withColumn("Profit", col("Profit").cast(DecimalType(38, 18)))
dfWithFormattedDate.createOrReplaceTempView("finallistoforders")
decimalDf.createOrReplaceTempView("FinalOrderDetails")


val calculateProfitUDF = udf((profit: Double) => {
```

```
    if (profit >= 0) {
      profit // Return positive profits as is
    } else {
      0 + profit
    }
  })
val resultD = spark.sql("""
        SELECT O.CustomerName, SUM(calculateProfit(OD.Profit)) AS TotalProfit
        FROM finallistoforders O
        JOIN FinalOrderDetails OD ON O.`Order ID` = OD.`Order ID`
        WHERE MONTH(O.`Order Date`) = 1
        GROUP BY O.CustomerName
      """)
```