2. SQL challenge queries 1 to 6 with spark dataframe.

package org.itc.com  
import org.apache.spark.SparkConf  
import org.apache.spark.SparkContext  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.\_  
import org.apache.spark.sql.types.\_  
import org.apache.spark.sql.expressions.Window  
  
object task2\_apr15 extends App{  
 val *spark* = SparkSession.*builder*()  
 .appName("ReadTextFile").master("local[1]")  
 .getOrCreate()  
  
 val *orderSchemasales* = "customer\_id String,order\_date Date,product\_id Int"  
 val *orderSchemamenu* = "product\_id Int,product\_name String,price Int"  
 val *orderSchemamembers* = "customer\_id String,join\_date Date"  
  
 val *salesDF* = *spark*.read.option("header", "true").schema(*orderSchemasales*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\sales.csv")  
 *salesDF*.show(5)  
  
 val *menuDF* = *spark*.read.option("header", "true").schema(*orderSchemamenu*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\menu.csv")  
 *menuDF*.show()  
 val *membersDF* = *spark*.read.option("header", "true").schema(*orderSchemamembers*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\members.csv")  
 *membersDF*.show()  
  
 val *new\_salesDF* = *salesDF*.join(*menuDF*, "product\_id")

//Q1.What is the total amount each customer spent at the restaurant?  
 val *totalAmountSpentDF* = *new\_salesDF* .groupBy("customer\_id")  
 .agg(*sum*(*col*("price")).alias("total\_amount\_spent"))  
  
 *totalAmountSpentDF*.show()  
//Q2. How many days has each customer visited the restaurant?  
val *customerVisitsDF* = *salesDF* .select("customer\_id", "order\_date")  
 .distinct()  
 .groupBy("customer\_id")  
 .agg(*count*("order\_date").alias("visits"))  
  
*customerVisitsDF*.show()

//Q3.What was the first item from the menu purchased by each customer?  
  
val *firstPurchaseDateDF* = *new\_salesDF* .groupBy("customer\_id")  
 .agg(*min*(*col*("order\_date")).alias("first\_purchase\_date"))  
  
val *firstItemDF* = *new\_salesDF* .join(*firstPurchaseDateDF*, Seq("customer\_id"), "inner")  
 .filter(*col*("order\_date") === *col*("first\_purchase\_date"))  
 .select("customer\_id", "product\_name")  
  
*firstItemDF*.show()

//Q4.What is the most purchased item on the menu and how many times was it purchased by all customers?  
 val *mostPurchasedItemDF* = *new\_salesDF* .groupBy("product\_name")  
 .agg(*count*("\*").alias("total\_purchases"))  
 .orderBy(*desc*("total\_purchases"))  
 .limit(1)  
  
 *mostPurchasedItemDF*.show()

//Q5. Which item was the most popular for each customer?  
  
val *windowSpec* = Window.*partitionBy*("customer\_id").orderBy(*desc*("total\_purchases"))  
  
val *rankedItemsDF* = *new\_salesDF* .groupBy("customer\_id", "product\_name")  
 .agg(*count*("\*").alias("total\_purchases"))  
 .withColumn("rank", *rank*().over(*windowSpec*))  
 .filter(*col*("rank") === 1)  
 .drop("rank")  
 .orderBy(*desc*("total\_purchases"))  
  
*rankedItemsDF*.show()

//Q6. Which item was purchased first by the customer after they became a member?  
val *memJoinDF* = *new\_salesDF*.join(*membersDF*, "customer\_id")  
  
val *firstPurchaseAfterJoinDF* = *memJoinDF* .filter(*col*("order\_date") >= *col*("join\_date"))  
 .groupBy("customer\_id")  
 .agg(*min*(*col*("order\_date")).alias("first\_purchase\_after\_join\_date"))  
  
val *firstItemAfterJoinDF* = *memJoinDF* .join(*firstPurchaseAfterJoinDF*, "customer\_id")  
 .filter(*col*("order\_date") === *col*("first\_purchase\_after\_join\_date"))  
 .select("customer\_id", "product\_name")  
  
*firstItemAfterJoinDF*.show()  
}

**Q1 Output:**



**Q2 Output:**



**Q3 Output:**



**Q4 Output:**



**Q5 Output:**



**Q6 Output:**



///////////////////////////////////////////////////////////////

package org.itc.com  
  
import org.apache.spark.SparkConf  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.\_  
import org.apache.spark.sql.functions.*broadcast*import org.apache.spark.sql.types.\_  
import org.apache.spark.sql.expressions.Window  
  
object taskapr16 extends App{  
 val *spark* = SparkSession.*builder*()  
 .appName("ReadTextFile").master("local[1]")  
 .getOrCreate()  
 import *spark*.implicits.\_  
  
 val *orderSchemasales* = "customer\_id String,order\_date Date,product\_id Int"  
 val *orderSchemamenu* = "product\_id Int,product\_name String,price Int"  
 val *orderSchemamembers* = "customer\_id String,join\_date Date"  
  
 val *salesDF* = *spark*.read.option("header", "true").schema(*orderSchemasales*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\sales.csv")  
 *salesDF*.show(5)  
  
 val *menuDF* = *spark*.read.option("header", "true").schema(*orderSchemamenu*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\menu.csv")  
 *menuDF*.show()  
 val *membersDF* = *spark*.read.option("header", "true").schema(*orderSchemamembers*)  
 .csv("D:\\spark\_code\\sparkdemo\\input\\members.csv")  
 *membersDF*.show()  
 *salesDF*.createOrReplaceTempView("sales")  
 *menuDF*.createOrReplaceTempView("menu")  
 *membersDF*.createOrReplaceTempView("members")  
  
 *spark*.sql("select \* from sales").show()  
 *spark*.sql("select \* from menu").show()  
 *spark*.sql("select \* from members").show()

// Q7.Which item was purchased just before the customer became a member?  
  
 *spark*.sql(  
 """SELECT s.customer\_id, m.product\_name AS last\_purchase\_before\_membership  
 |FROM (  
 | SELECT s.customer\_id, MAX(s.order\_date) AS last\_purchase\_date  
 | FROM sales s  
 | JOIN members m ON s.customer\_id = m.customer\_id  
 | WHERE s.order\_date < m.join\_date  
 | GROUP BY s.customer\_id  
 |) last\_purchase  
 |JOIN sales s ON s.customer\_id = last\_purchase.customer\_id AND s.order\_date = last\_purchase.last\_purchase\_date  
 |JOIN menu m ON s.product\_id = m.product\_id  
 |""".stripMargin  
 ).show()

//Q8. What is the total items and amount spent for each member before they became a member?  
*spark*.sql(  
 """SELECT m.customer\_id,  
 | COUNT(s.product\_id) AS total\_items\_purchased,  
 | SUM(m2.price) AS total\_amount\_spent  
 |FROM sales s  
 |JOIN menu m2 ON s.product\_id = m2.product\_id  
 |JOIN members m ON s.customer\_id = m.customer\_id  
 |WHERE s.order\_date < m.join\_date  
 |GROUP BY m.customer\_id  
 |""".stripMargin  
).show()

//Q9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier -  
// how many points would each customer have?  
*spark*.sql(  
 """SELECT  
 | s.customer\_id,  
 | SUM(  
 | CASE  
 | WHEN m.product\_name = 'sushi' THEN 20 \* m.price  
 | ELSE 10 \* m.price  
 | END  
 | ) AS total\_points  
 |FROM sales s  
 |JOIN menu m ON s.product\_id = m.product\_id  
 |GROUP BY s.customer\_id  
 |""".stripMargin  
).show()

//Q-- 10. In the first week after a customer joins the program (including their join date) they earn  
 // 2x points on all items,not just sushi - how many points do customer A and B have at the end of January?  
 *spark*.sql(  
 """SELECT  
 | m.customer\_id,  
 | SUM(  
 | CASE  
 | WHEN s.order\_date >= m.join\_date THEN 20 \* m2.price  
 | ELSE 10 \* m2.price  
 | END  
 | ) AS total\_points  
 |FROM sales s  
 |JOIN menu m2 ON s.product\_id = m2.product\_id  
 |JOIN members m ON s.customer\_id = m.customer\_id  
 |WHERE m.customer\_id IN ('A', 'B')  
 | AND s.order\_date <= '2021-01-31'  
 |GROUP BY m.customer\_id  
 |""".stripMargin  
 ).show()  
 }  
**Q7. OUTPUT:**



**Q8. OUTPUT:**



**Q9. OUTPUT:**



**Q10. OUTPUT:**

