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:**

