Student’s Full Name: Surajit Pal

Course Title: Data Warehousing and Analytics in the Cloud

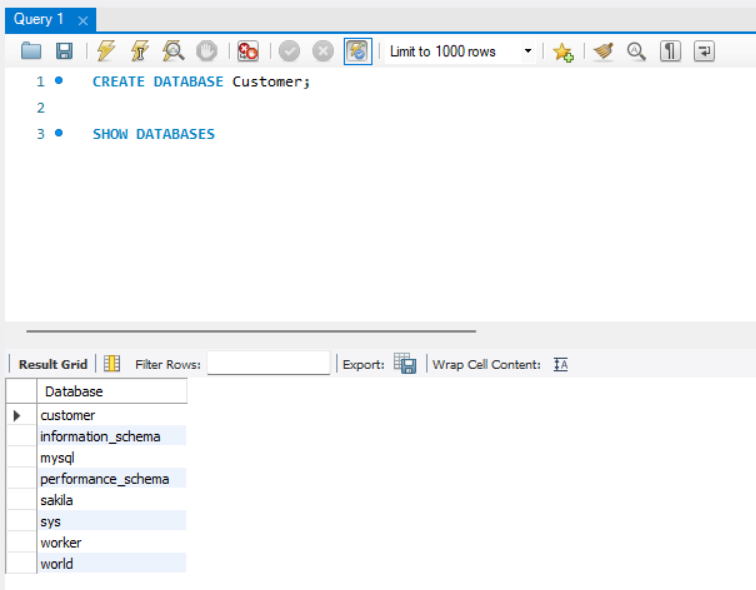
Term name and year: Spring 2023

Submission Week: Week 4 - Assignment 3

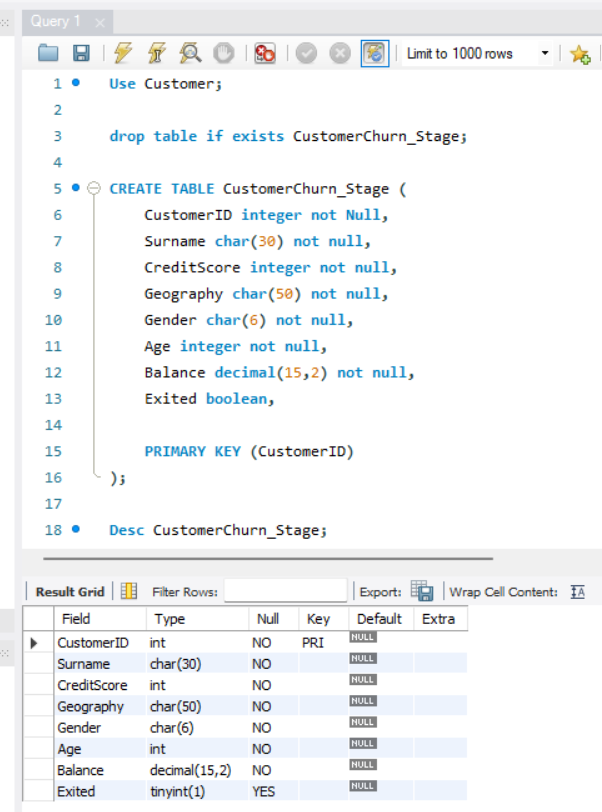
Instructor’s Name: Dr.Nayem Rahman

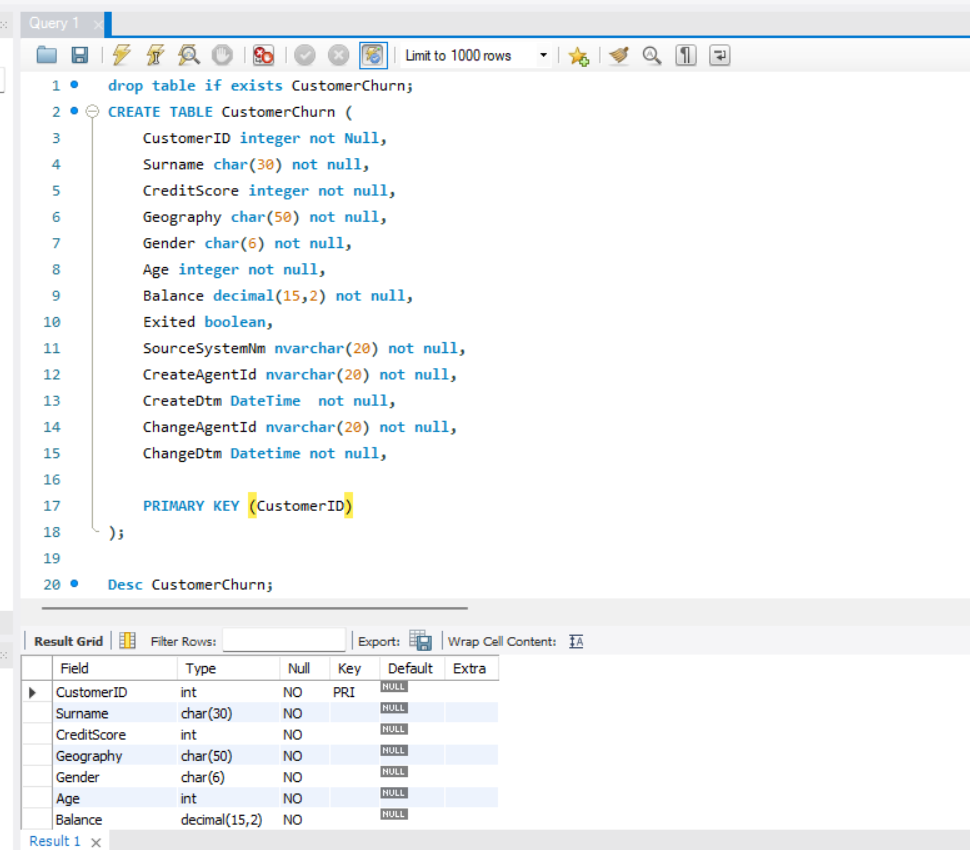
Date of Submission: 22nd Nov 2023

**Q1. { A } Using the MySQL Workbench, create a database called Customer. The database must be named “Customer”. { B } Check if the database was created and use the same for further questions.**

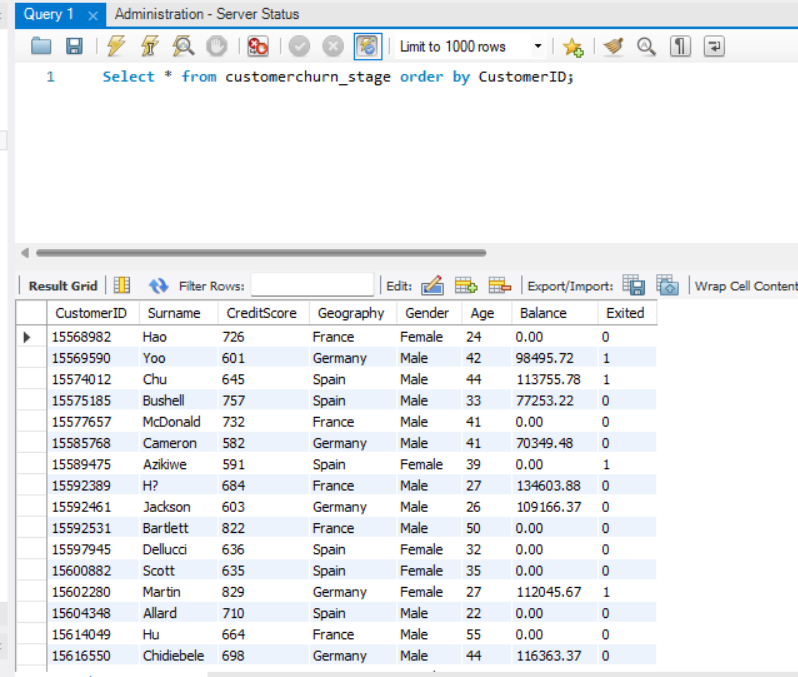


**Q2. { A } Create a staging table, \*\* Customer.CustomerChurn\_Stage \*\*, in a database system, with the column list provided in the CSV file. Define the ' CustomerId ' as the Primary Key (PK). Get the table definition (DDL) from the database system and capture it in a Word document for submission. { B } Create a persistent table, \*\* Customer.CustomerChurn \*\*, with the column list provided in the CSV file + following 5 columns : << SourceSystemNm NVARCHAR(20) NOT NULL , CreateAgentId NVARCHAR(20) NOT NULL , CreateDtm DATETIME NOT NULL, ChangeAgentId NVARCHAR(20) NOT NULL , ChangeDtm DATETIME NOT NULL >> Define the ' CustomerId ' as the Primary Key (PK). Get the table definition (DDL) from the database system and capture it in a Word document for submission.**

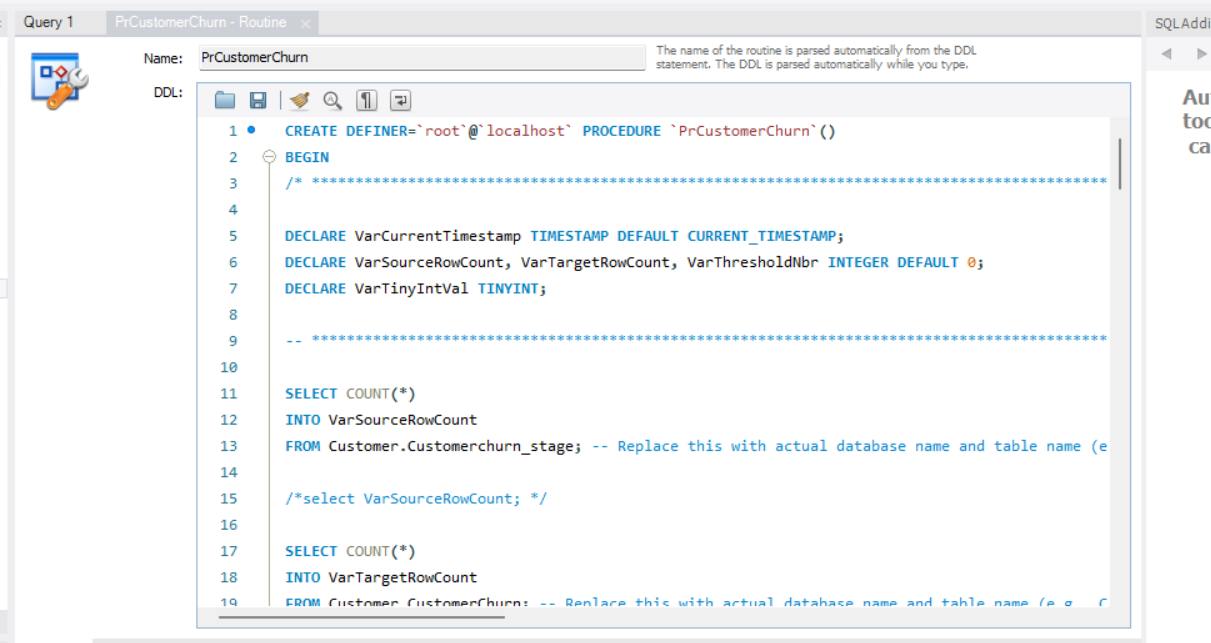




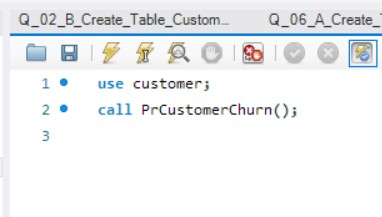
**Q3. { A } Load the staging table, \*\* Customer.CustomerChurn\_Stage \*\*, with data from the CSV file, CustomerChurn1.csv . { B } Verify data by comparing the row counts between the CSV file and the staging table, \*\* Customer.CustomerChurn\_Stage [Data Source: CustomerChurn1.CSV] \*\*. Provide the screenshot of last few rows using the ' SELECT \* ' . Make sure the output shows all column values. The SELECT statement must use the ORDER BY ' CustomerId '.**

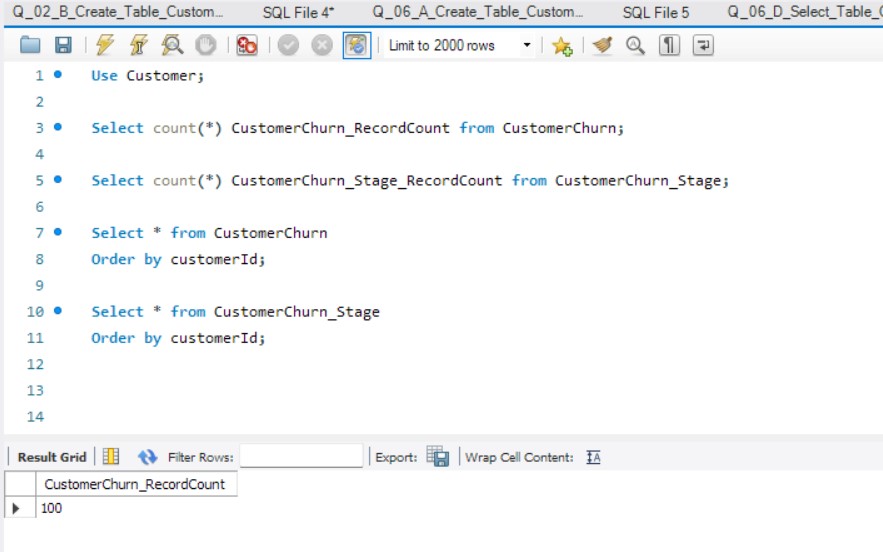


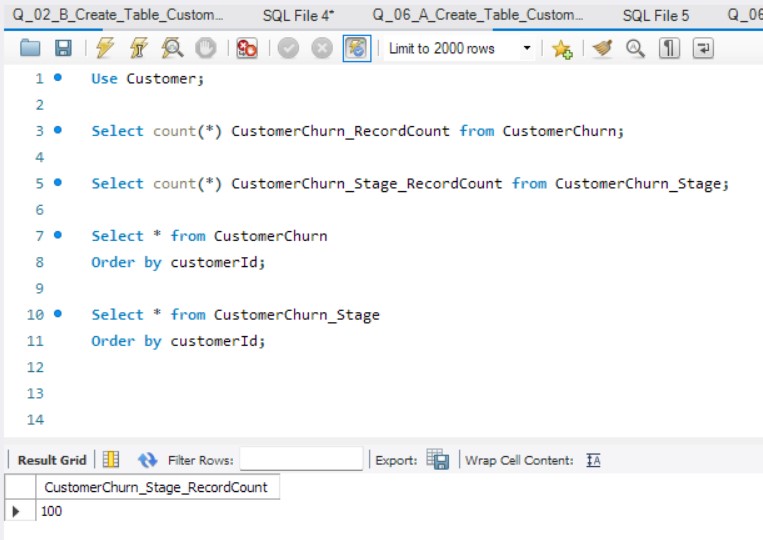
**Q4. Create a database stored procedure based on the template provided along with this assignment << StoredProc\_Template.txt >>. Name the stored procedure name this: \*\* Customer.PrCustomerChurn \*\* . [[ NOTE : This stored procedure will use the table, \*\* Customer.CustomerChurn\_Stage \*\* , as the source (aka, staging table). This stored procedure will use the table, \*\* Customer.CustomerChurn \*\*, as the target (aka, persistent table). ]]**

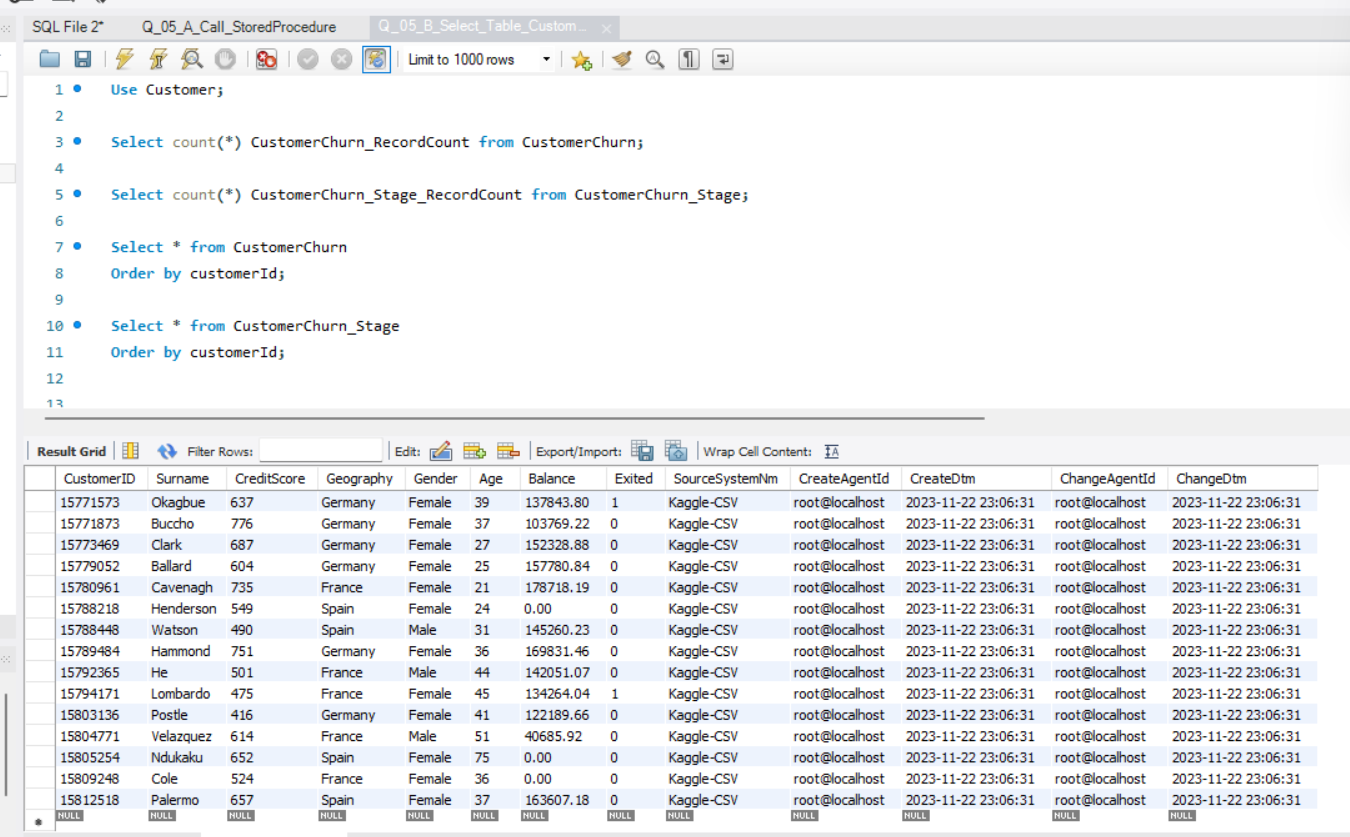


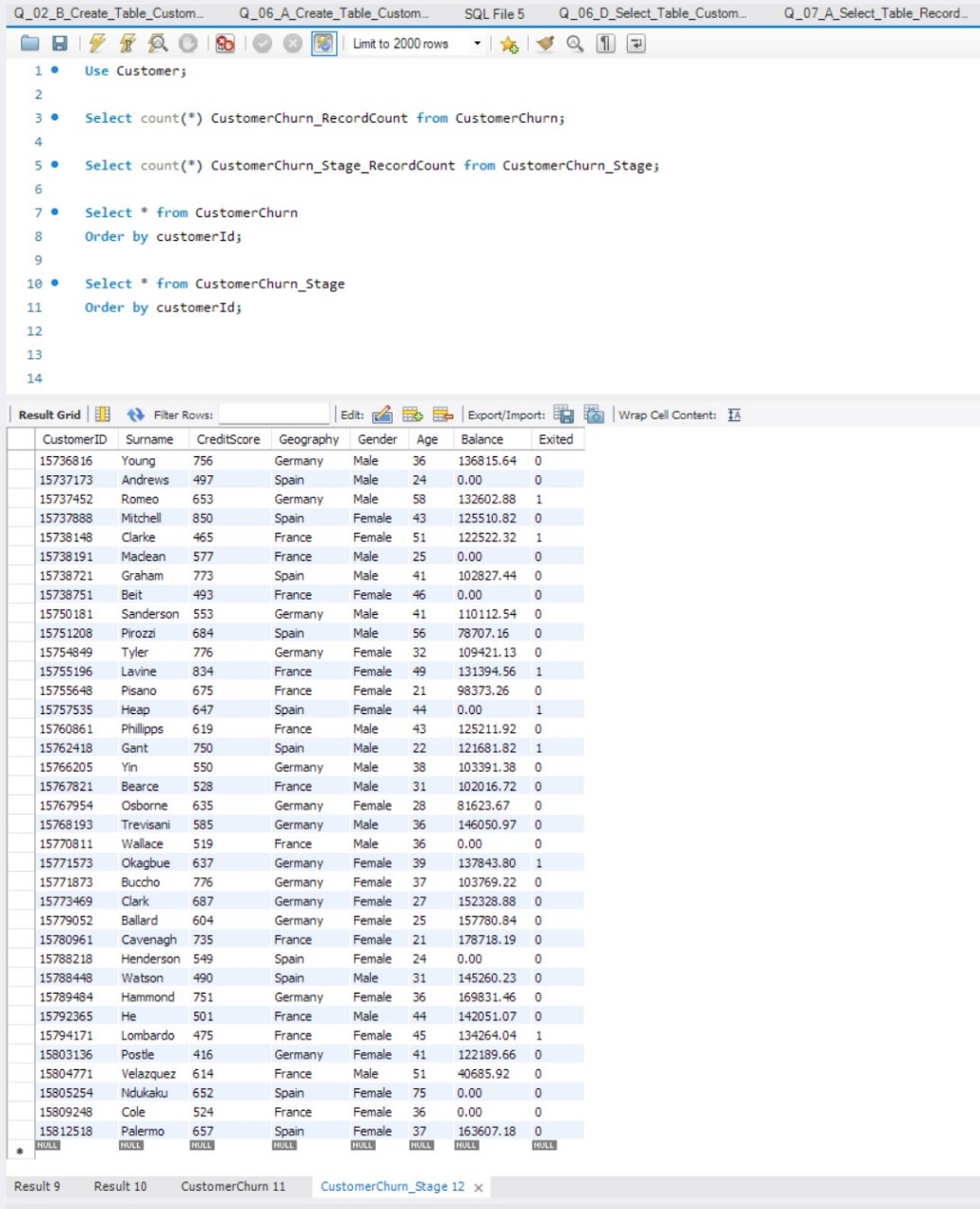
**Q5. Execute the stored procedure, \*\* Customer.PrCustomerChurn \*\*, that was created in Q4. After execution, the stored procedure should load data from the stage to the persistent table: \*\* Customer.CustomerChurn \*\*. {A} Verify data by comparing the row counts between the staging table, \*\* Customer.CustomerChurn\_Stage [Data Source: CustomerChurn1.CSV] \*\* and the persistent table: \*\* Customer.CustomerChurn \*\*. { B } Provide the screenshot of last few rows using the SELECT \*. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId.**



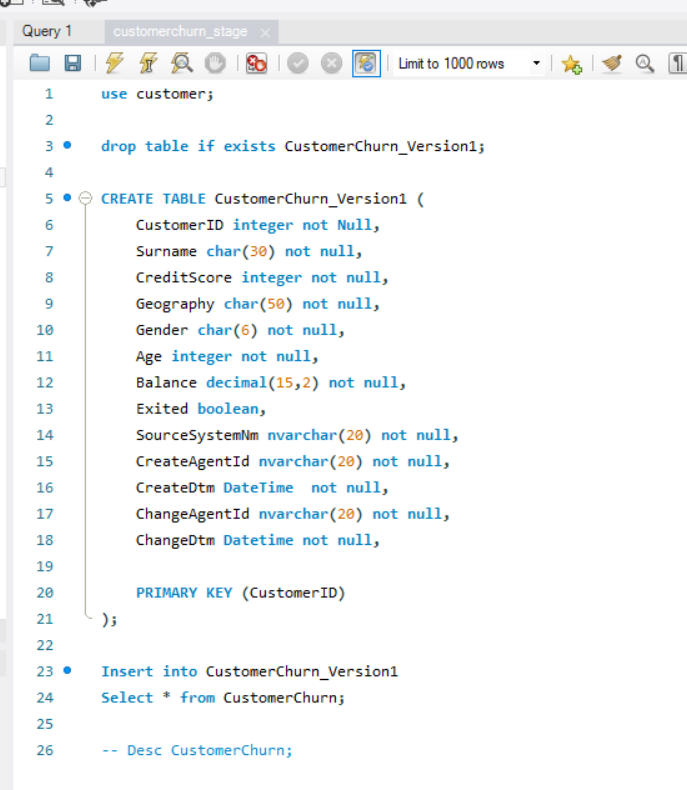


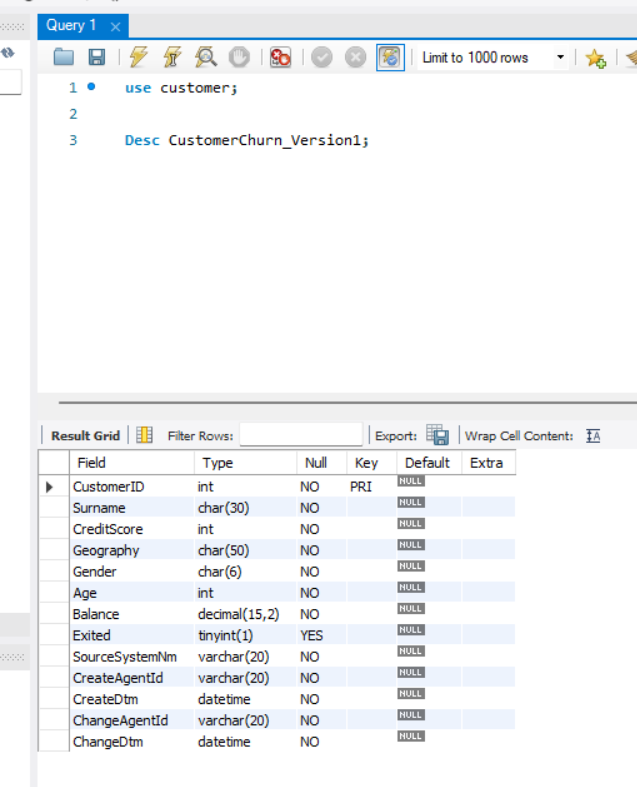


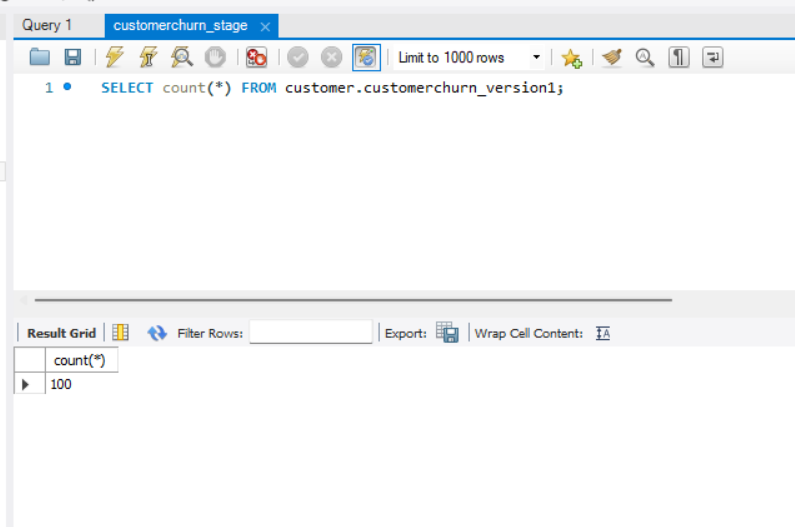
****

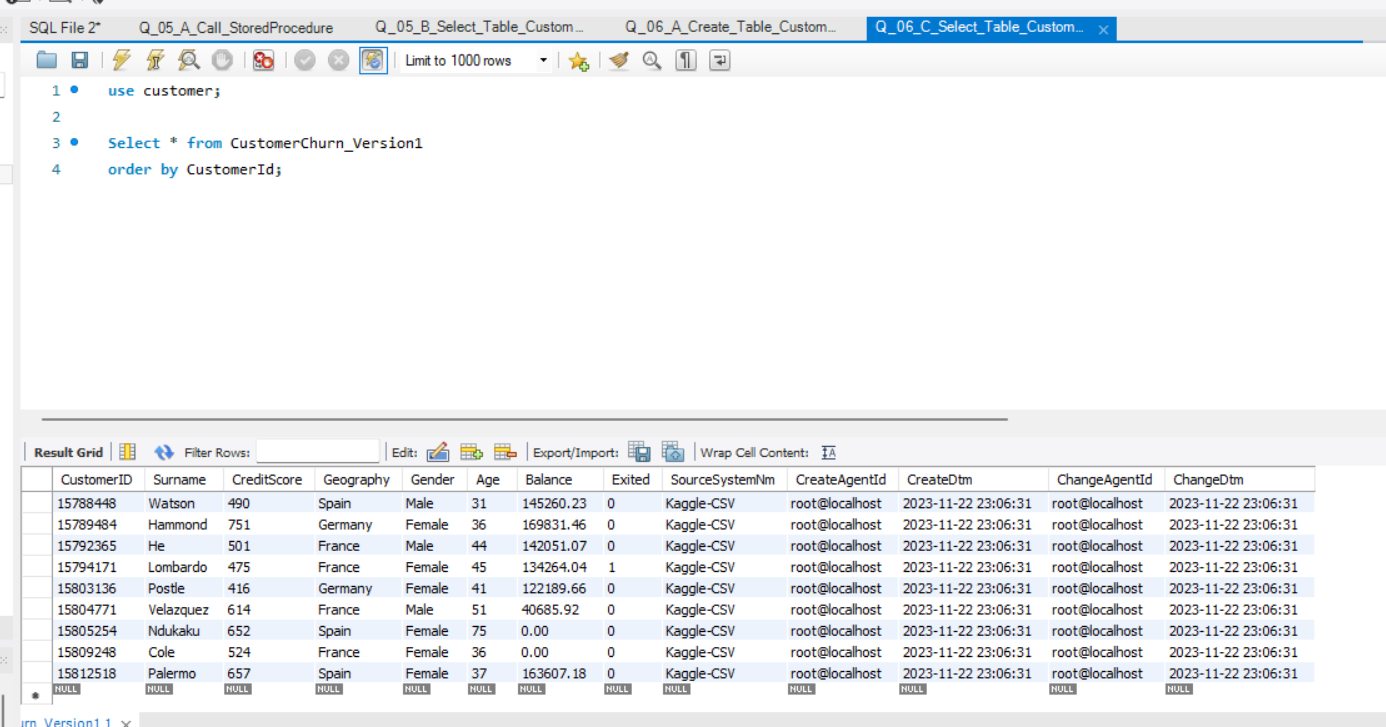


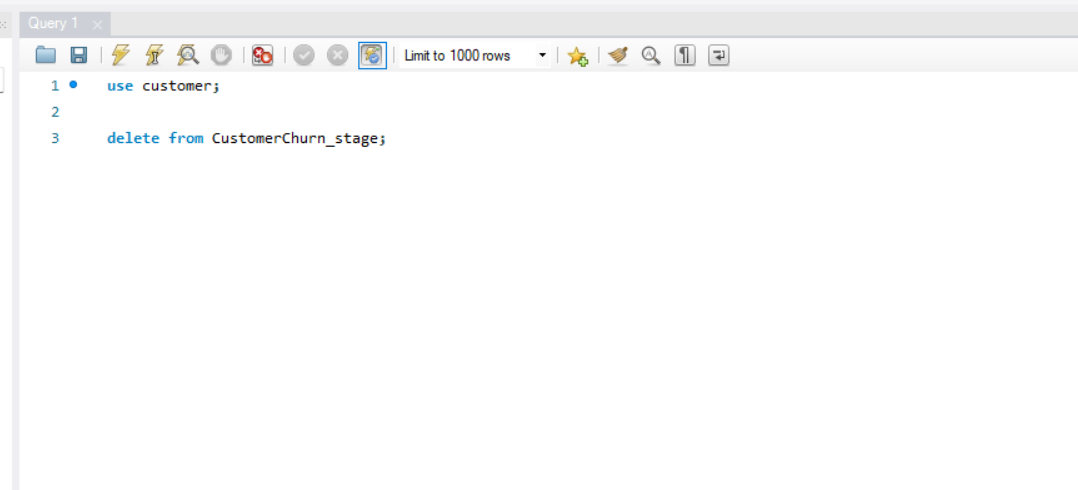
**Q6. After data verification is completed, in Q5 , { A } create table, \*\* Customer.CustomerChurn\_Version1 \*\*, with data from \*\* Customer.CustomerChurn \*\* (that was already loaded from Customer.CustomerChurn\_Stage via the stored procedure). { B } Show table definition of Customer.CustomerChurn\_Version1 and show the row count of the table, \*\* Customer.CustomerChurn\_Version1 \*\*: { C } Provide the screenshot of last few rows for \*\* Customer.CustomerChurn\_Version1 \*\* [Originally data came from: CustomerChurn1.CSV]. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId. { D } Empty the staging table, \*\* Customer.CustomerChurn\_Stage \*\*, and load it with data from the CSV file, "CustomerChurn2.csv ". Verify data by comparing the row counts between the CSV file and the staging table, \*\* Customer.CustomerChurn\_Stage \*\* [Data Source: CustomerChurn2.CSV]. Provide the row count of \*\* Customer.CustomerChurn\_Stage \*\* that you loaded from CustomerChurn2.csv file. Provide the screenshot of last few rows using the SELECT \*. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId.**

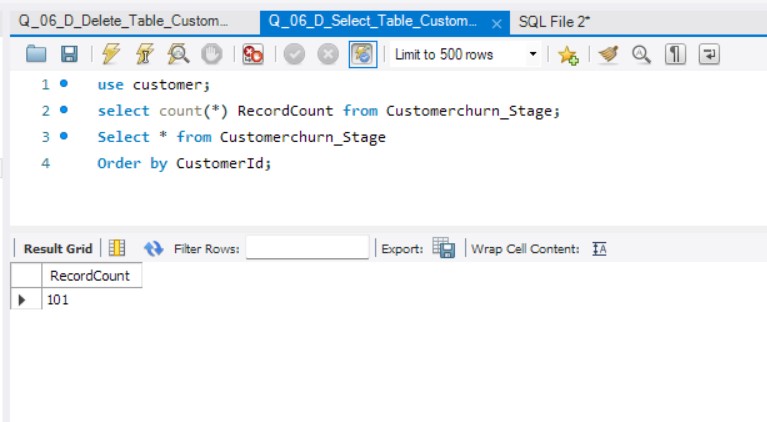
****

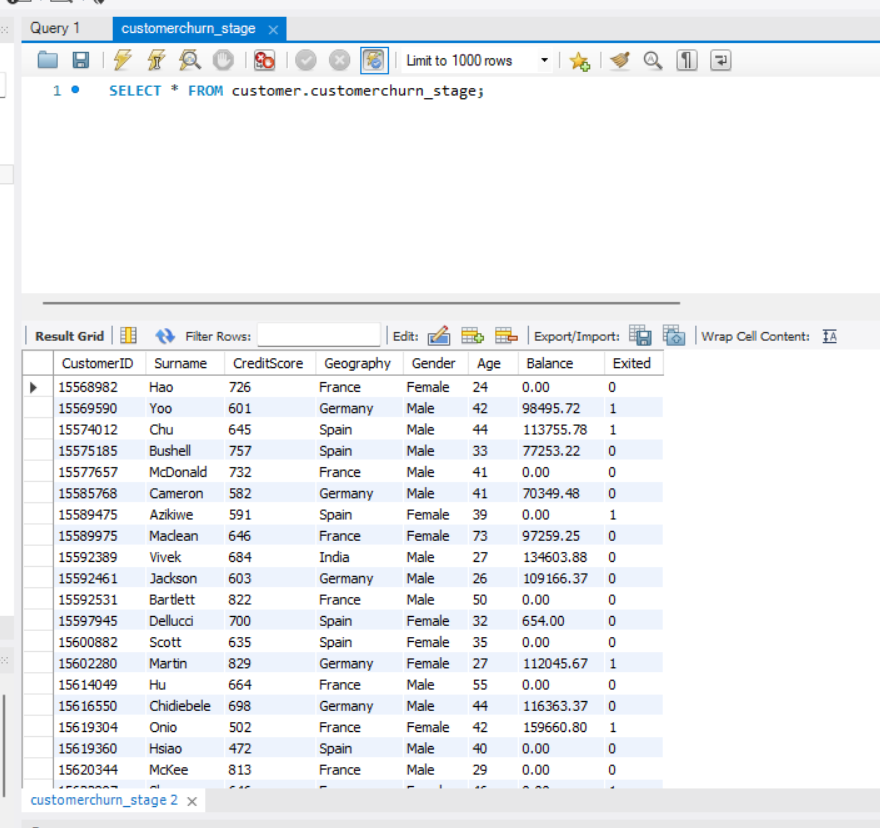
****

****

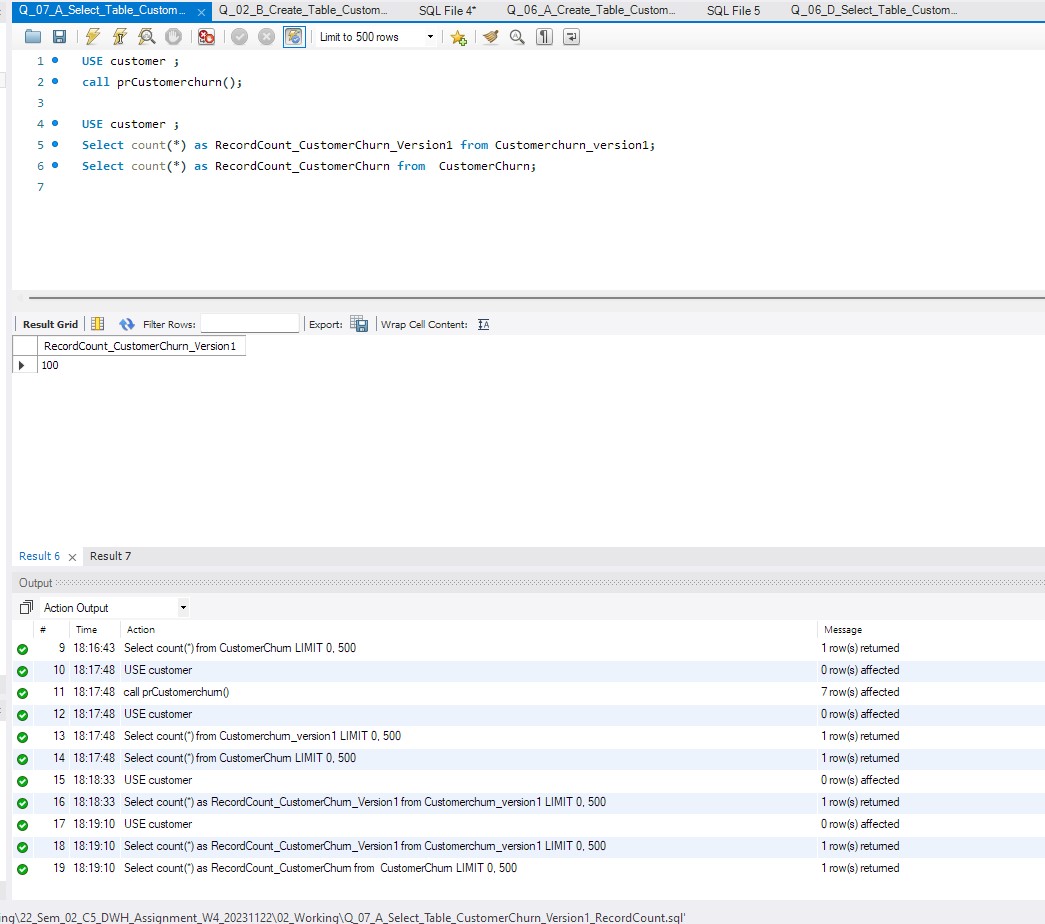


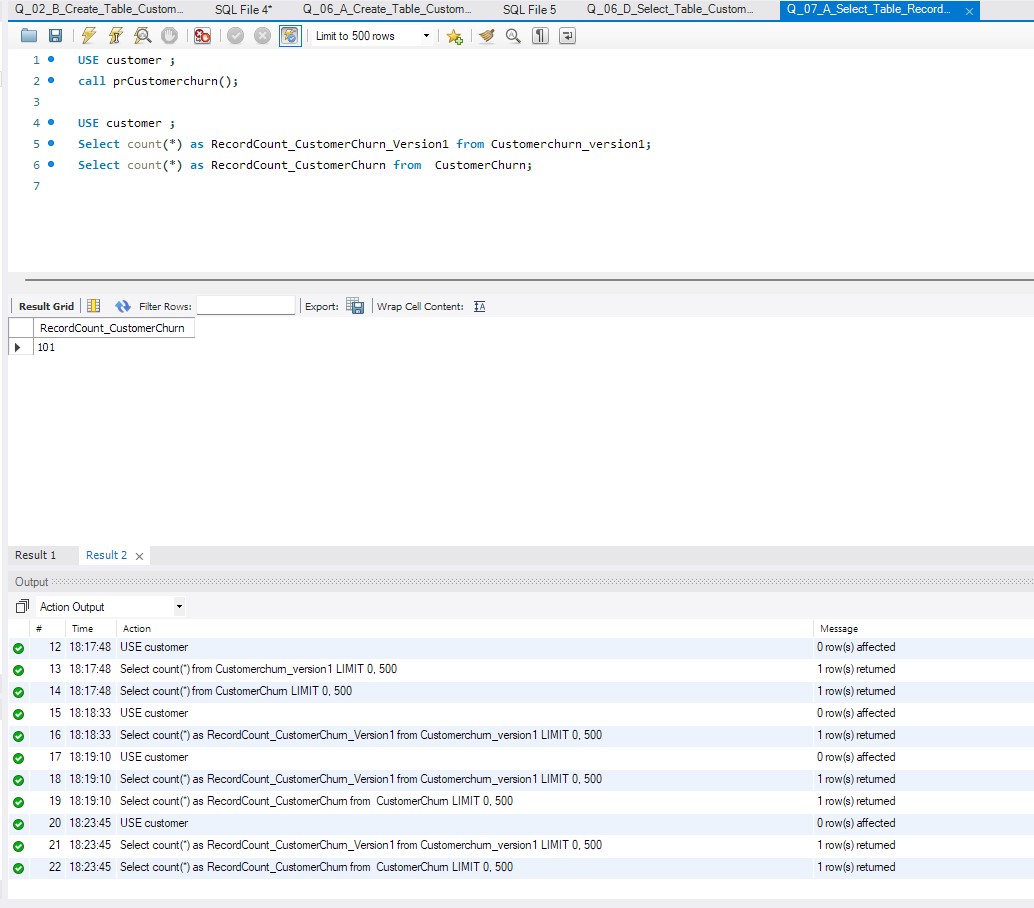


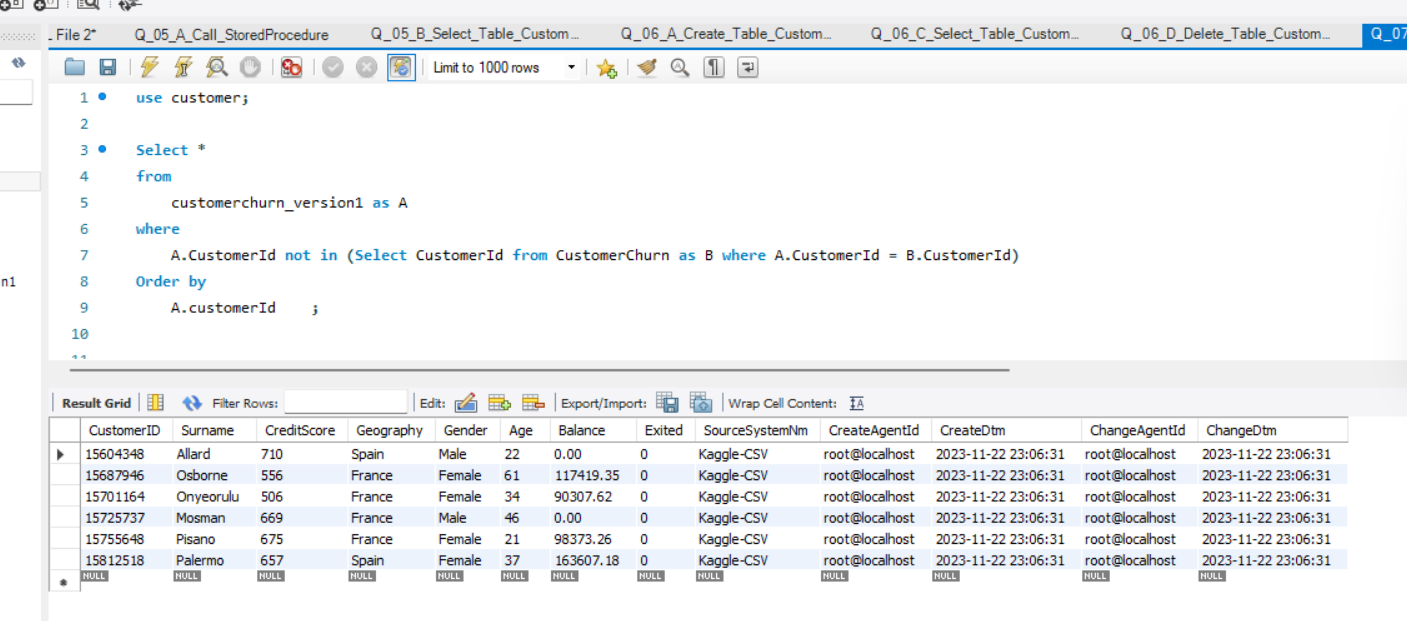




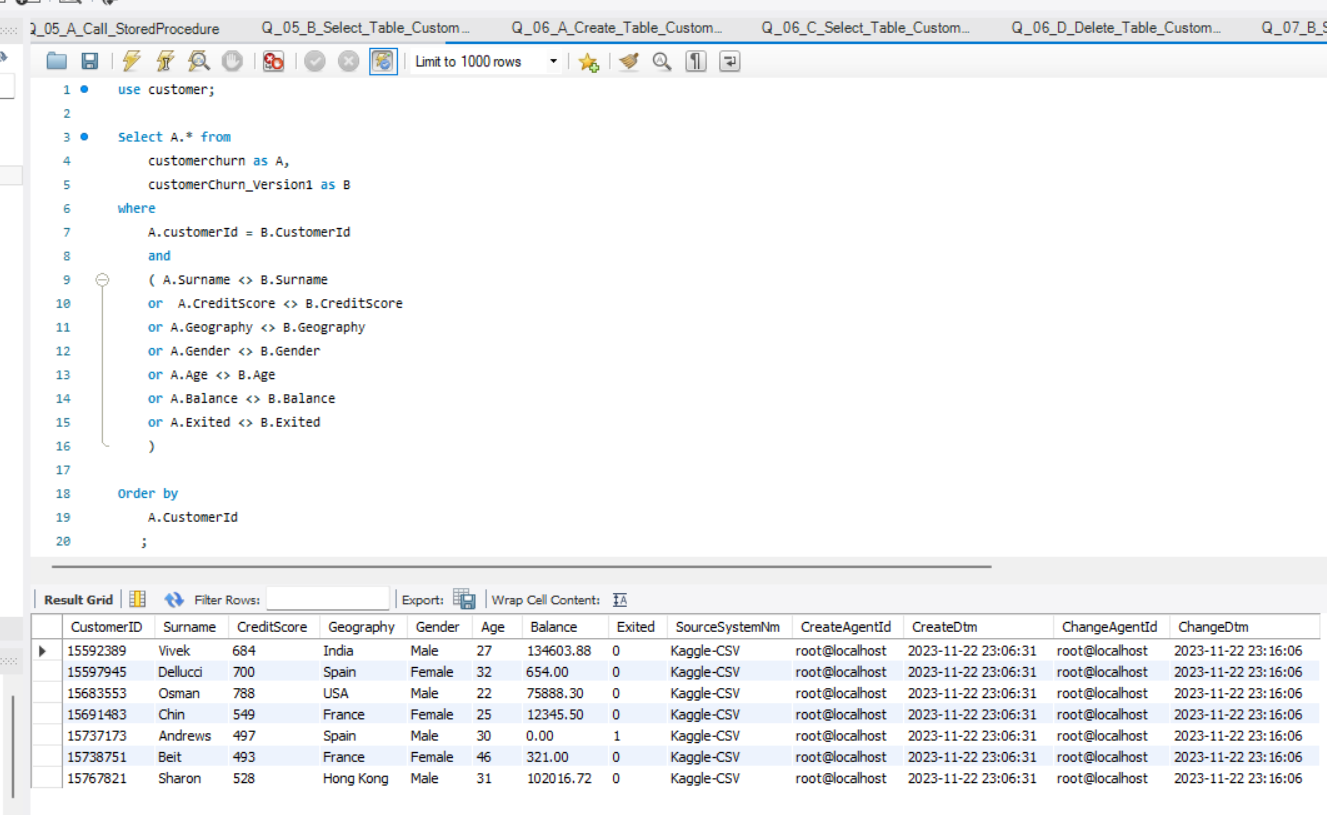
**Q7. Execute the stored procedure, Customer.PrCustomerChurn, that was created in Q4. After execution, the stored procedure should load data from the stage to the persistent table: Customer.CustomerChurn. CALL `customer`.`PrCustomerChurn`(); This time, the table will be refreshed via DELETE, UPDATE, and INSERT/SELECT statements in the stored procedure. Show the row count results of both Customer.CustomerChurn\_Version1 table [Data Source: CustomerChurn1.CSV] and the persistent table: Customer.CustomerChurn. Compare the rows between the Customer.CustomerChurn\_Version1 [Data Source: CustomerChurn1.CSV] table and the persistent table: Customer.CustomerChurn [Data Source: CustomerChurn2.CSV]. Show the rows that are available in the Customer.CustomerChurn\_Version1 table but not in the Customer.CustomerChurn table (implementation of brand-new row DELETE statement of the stored procedure).**







**Q8. Show the rows (SELECT \*) that changed (one or many non-Primary Key columns), in the Customer.CustomerChurn table (implementation of UPDATE statement of the stored procedure). You need to perform a comparison between Customer.CustomerChurn table [Data Source: CustomerChurn2.CSV] and Customer.CustomerChurn\_Version1 table [Data Source: CustomerChurn1.CSV] in terms of non-PK columns (Excluds: SourceSystemNm, CreateAgentId, CreateDtm, ChangeAgentId, ChangeDtm), and with a join condition using the PK column(s). You must do ORDER BY CustomerId. The output of this query should show different values for the CreateDtm and ChangeDtm columns in Customer.CustomerChurn table for the changed rows. Take a screenshot and capture it in the Word document. Make sure all columns including CreateDtm and ChangeDtm of CustomerChurn table are displayed.**



**Q9. Provide the screenshot of last few rows using the SELECT \* FROM Customer.CustomerChurn. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId. Show the rows that are available in the Customer.CustomerChurn table [Data Source: CustomerChurn2.CSV] but not in the Customer.CustomerChurn\_Version1 table (implementation of brand-new rows INSERT by the stored procedure). Do a SELECT \* along with ORDER BY CustomerId. Take a screenshot and capture it in the Word document.**

