Assignment - Week-4 Data Warehousing and Analytics

Student's Full Name: Anand Ramaswamy Jayshree

Course Title: INFO531

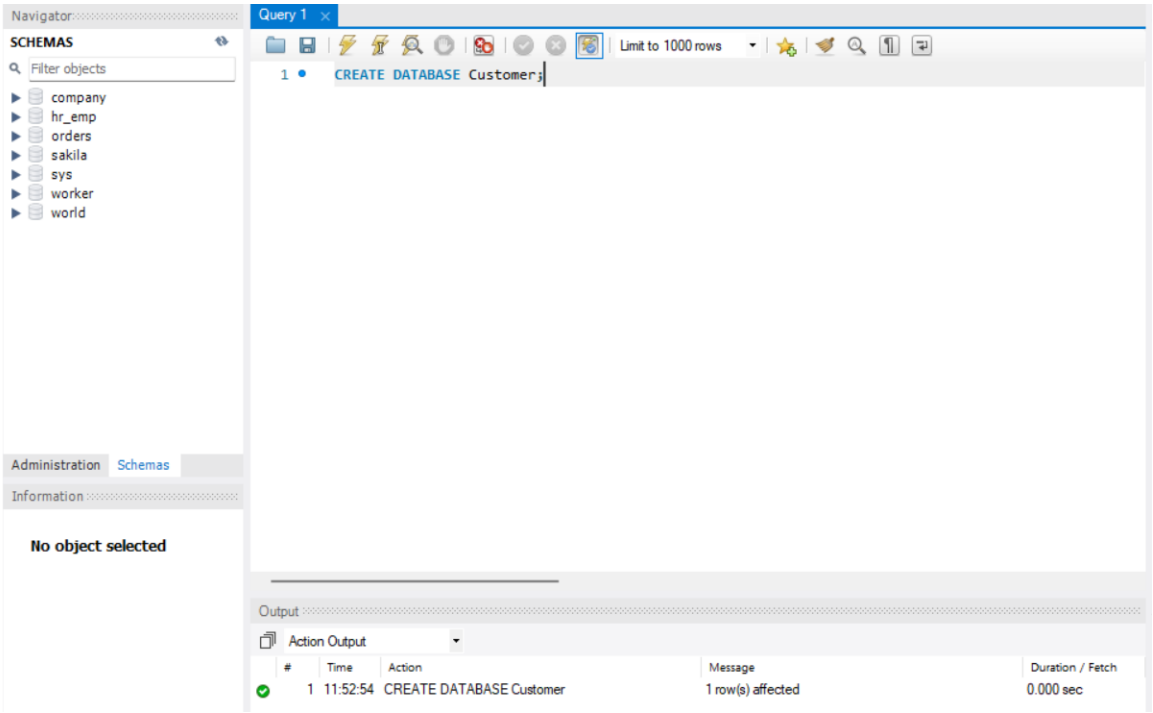
Term name and year: Summer 2024

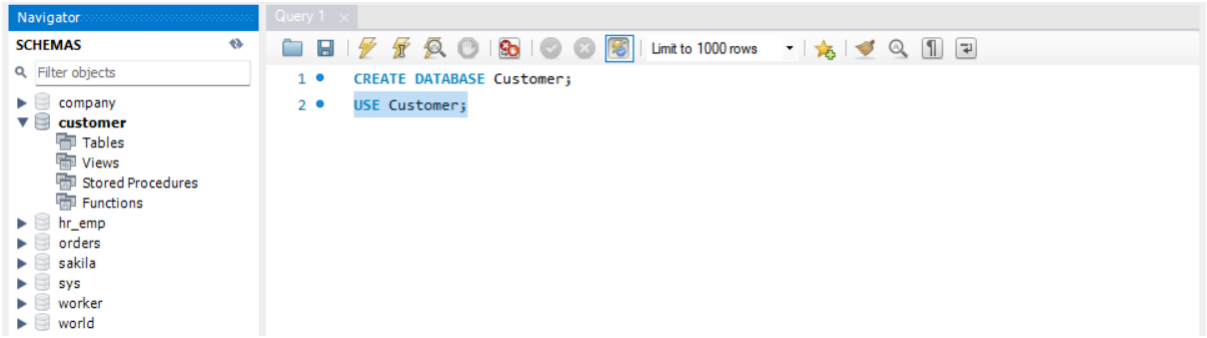
Submission Week: Week 4 - Assignment 3

Instructor's Name: Dr.Nayem Rahman

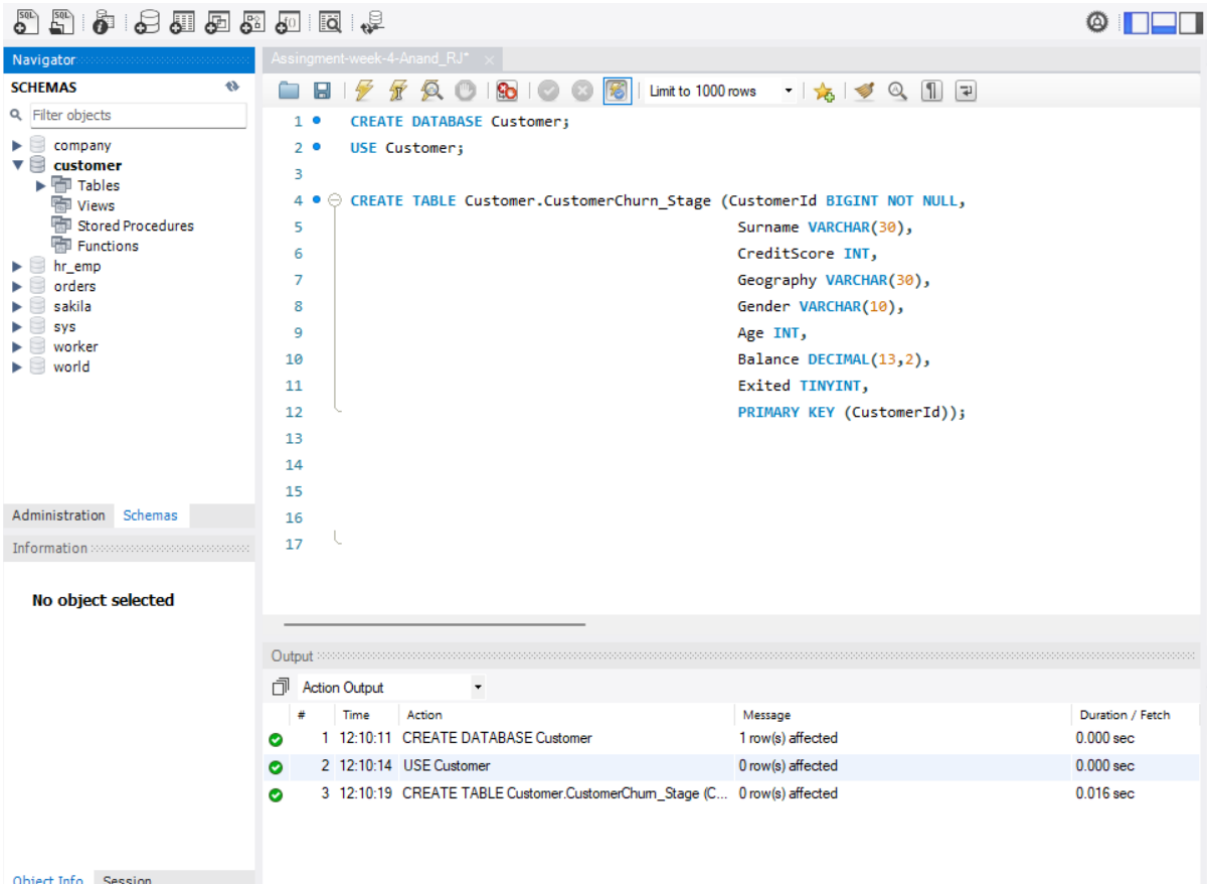
Date of Submission: August 5, 2024

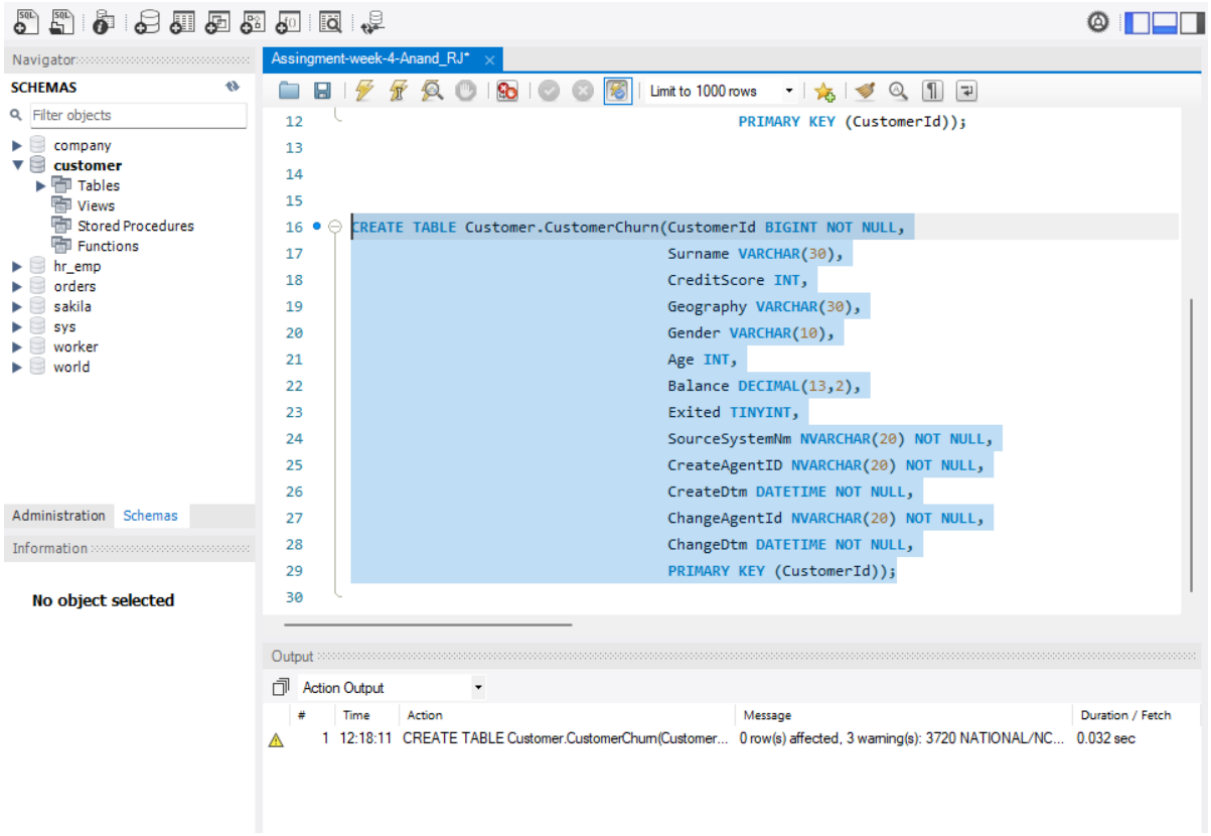
Q1. { A } Using the MySQL Workbench, create a database called Customer. The database must be named "Customer". { B } Check if the database was created and use the same for further questions

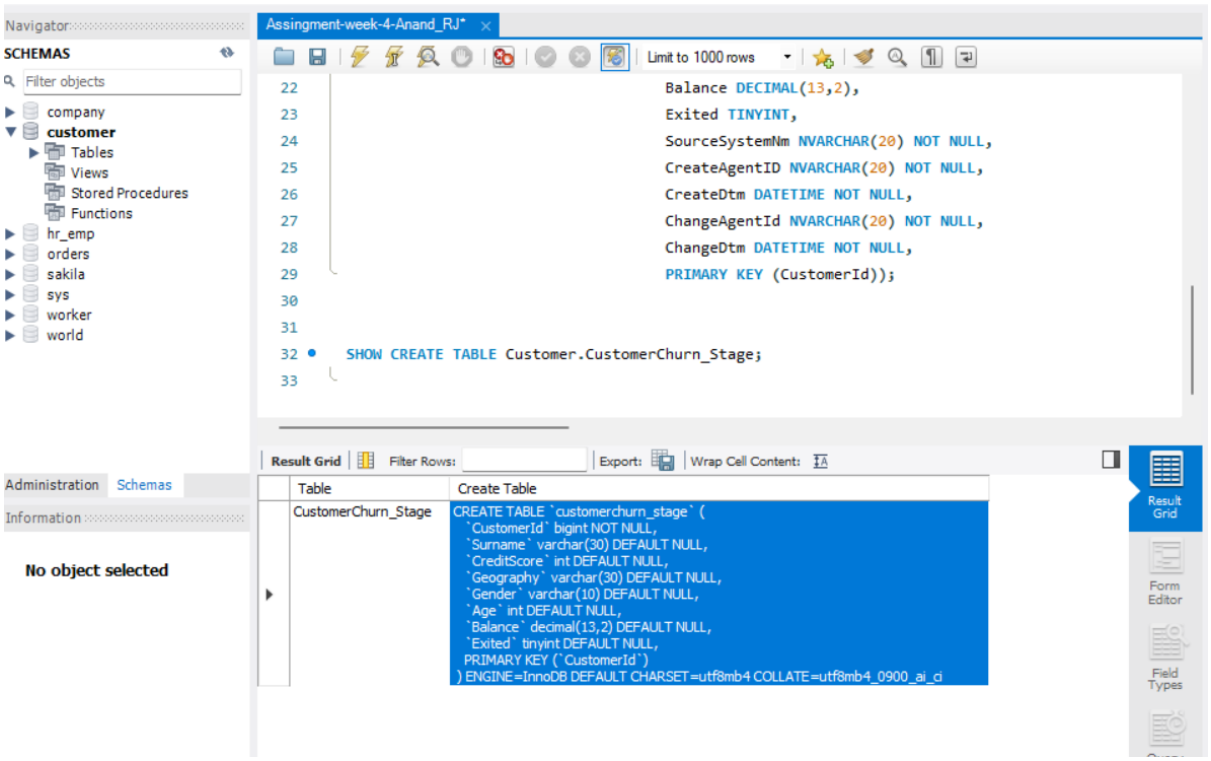




Q2. { A } Create a staging table, ** Customer.CustomerChurn_Stage **, in a database system, with the column list provided in the CSV file. Define the ' CustomerId ' as the Primary Key (PK). Get the table definition (DDL) from the database system and capture it in a Word document for submission. { B } 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

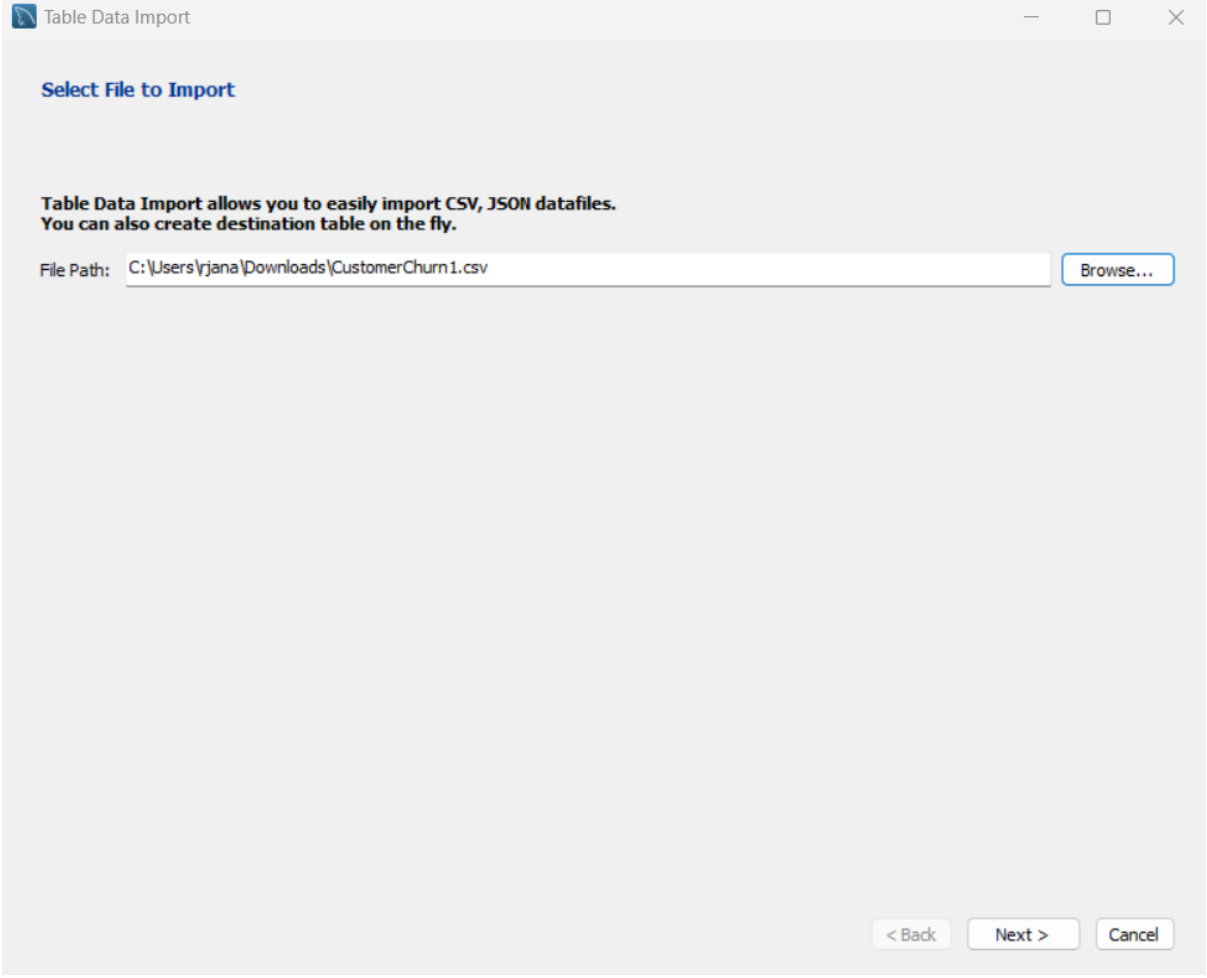






Q3. { A } Load the staging table, ** Customer.CustomerChurn_Stage **, with data from the CSV file, CustomerChurn1.csv . { B } Verify data by comparing the row counts between the CSV file and the staging table, ** Customer.CustomerChurn_Stage [Data Source: CustomerChurn1.CSV] **. Provide the screenshot of last few rows using the ' SELECT * ' .

Make sure the output shows all column values. The SELECT statement must use the ORDER BY ' CustomerId '



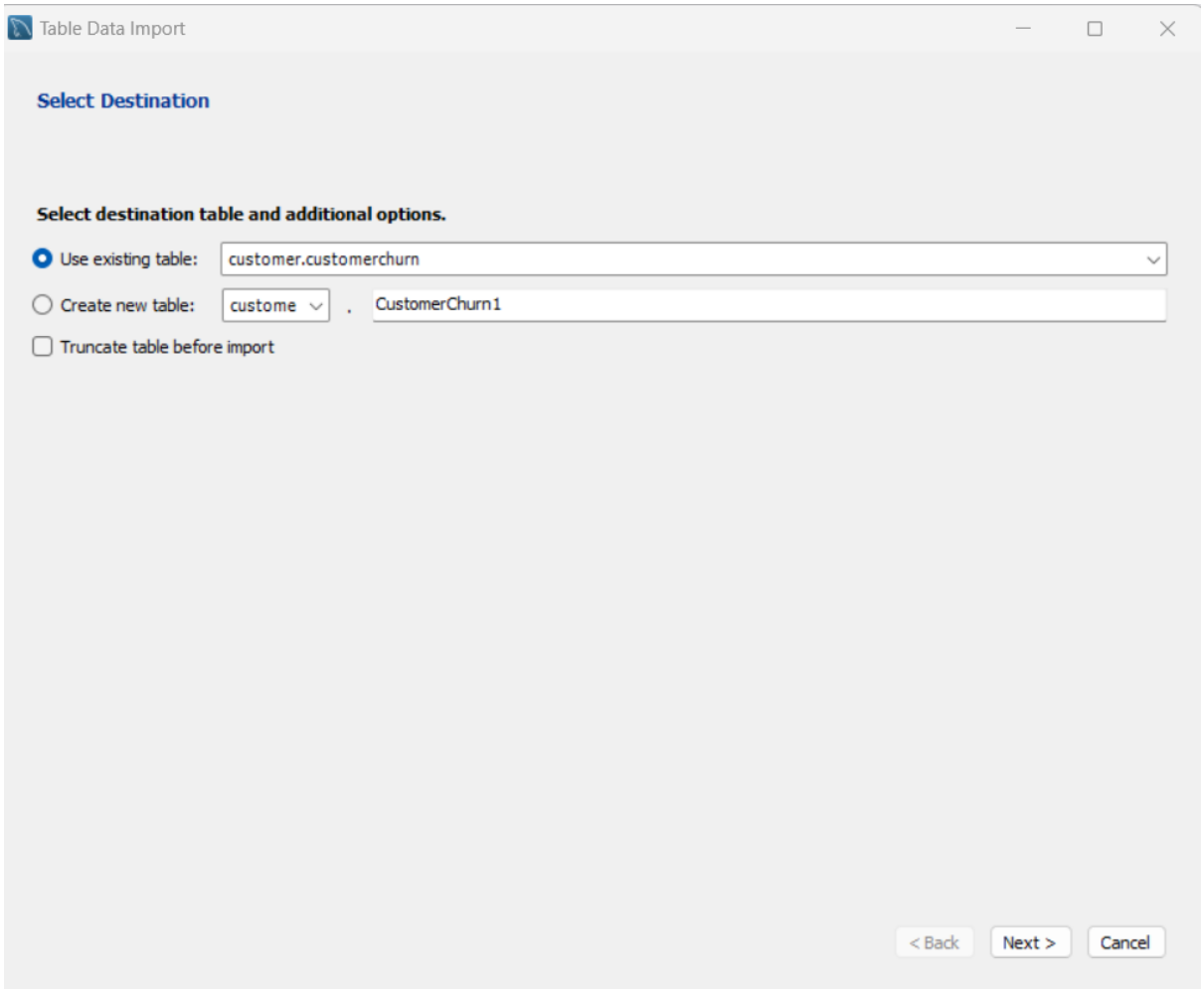
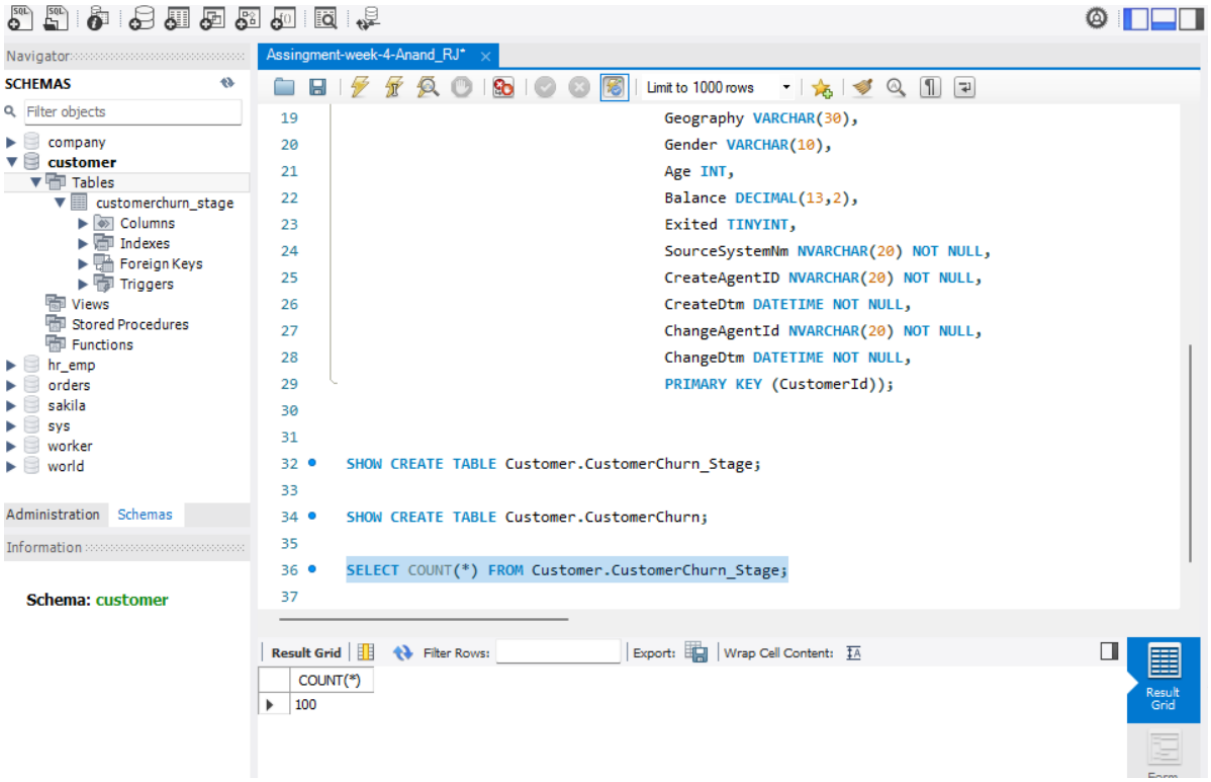
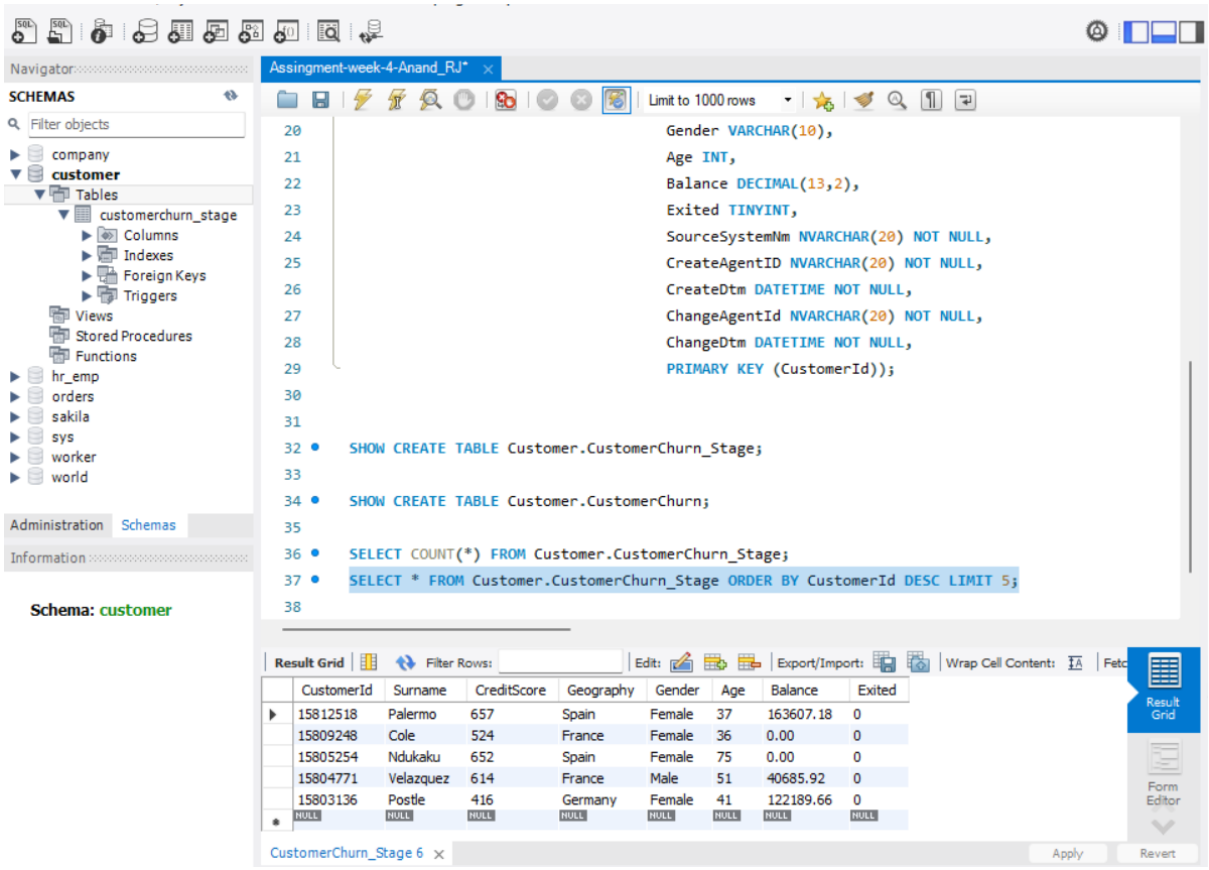


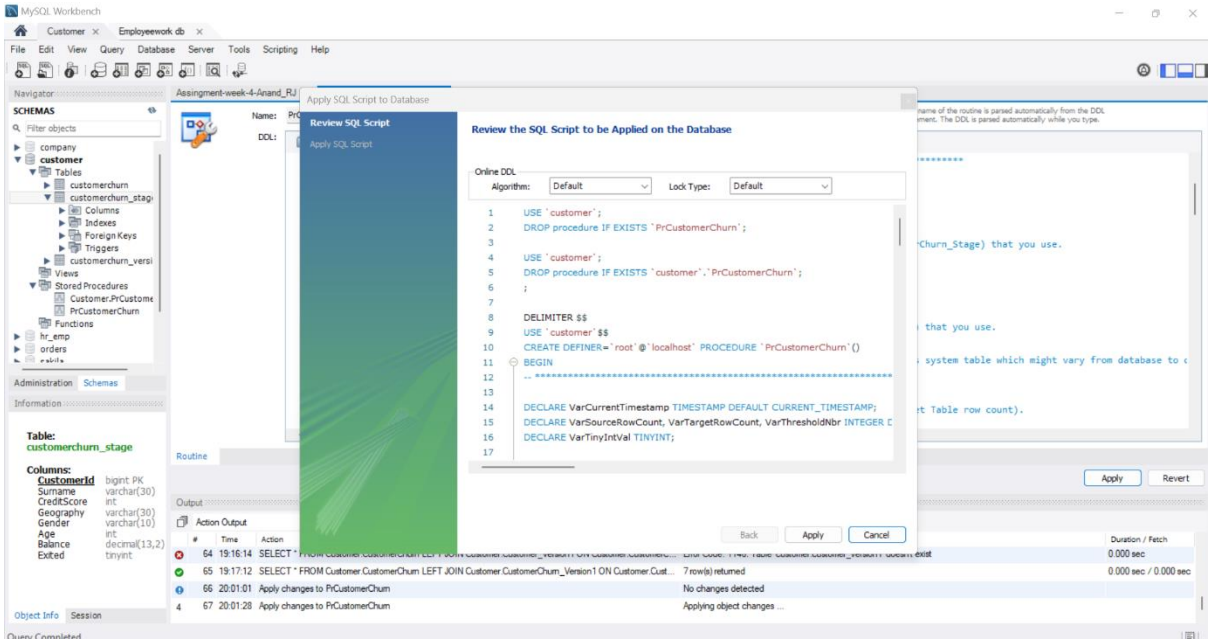
Table Data Import
Select Destination
Select destination table and additional options.
Use existing table:
customer.customerchurn
Create new table:
custome
CustomerChurn1
Truncate table before import
< Back
Next >
Cancel

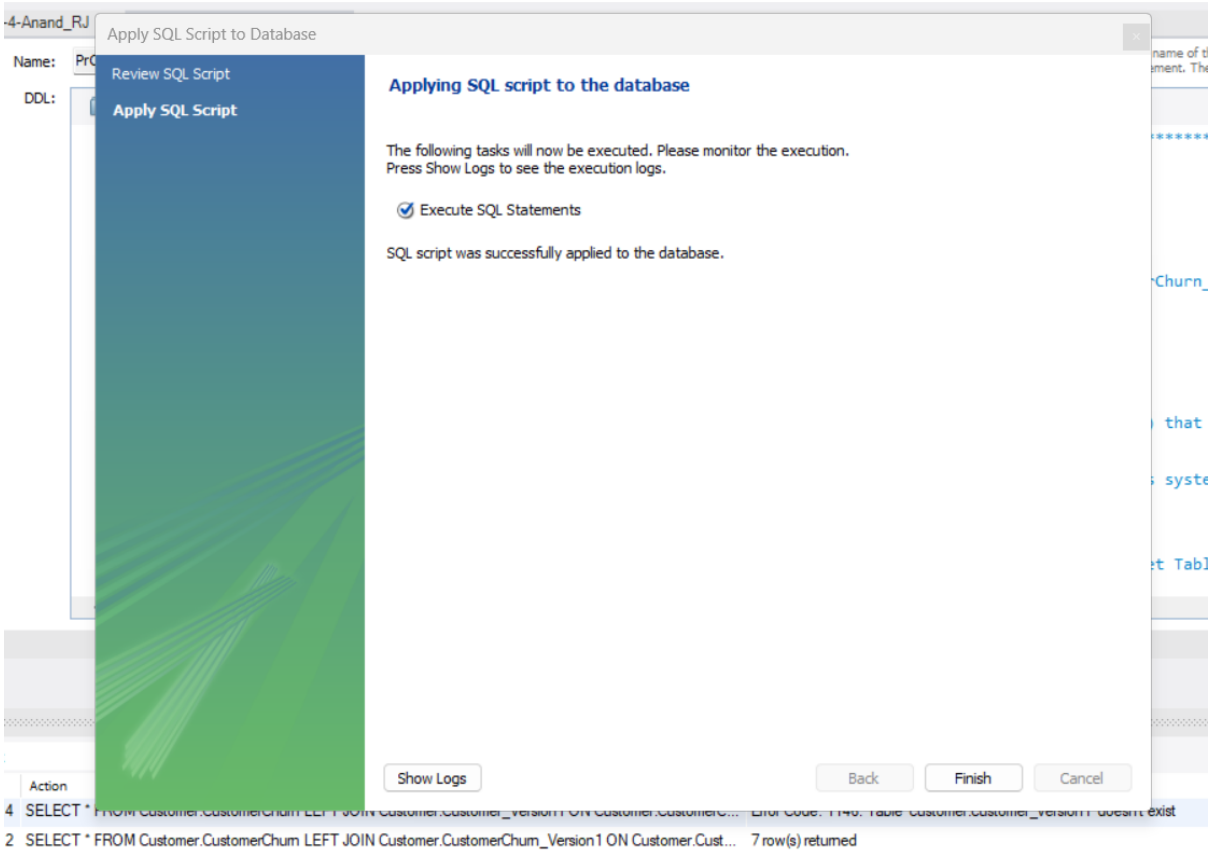

Navigator
SCHEMAS
Filter objects
company
customer
Tables
customerchurn_stage
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
hr_emp
orders
sakila
sys
worker
world
Administration
Schemas
Information
Schema: customer
Assingment-week-4-Anand_RJ*
Limit to 1000 rows
Geography VARCHAR(30),
Gender VARCHAR(10),
Age INT,
Balance DECIMAL(13,2),
Exited TINYINT,
SourceSystemNm NVARCHAR(20) NOT NULL,
CreateAgentID NVARCHAR(20) NOT NULL,
CreateDtm DATETIME NOT NULL,
ChangeAgentId NVARCHAR(20) NOT NULL,
ChangeDtm DATETIME NOT NULL,
PRIMARY KEY (CustomerId));
SHOW CREATE TABLE Customer.CustomerChurn_Stage;
SHOW CREATE TABLE Customer.CustomerChurn;
SELECT COUNT(*) FROM Customer.CustomerChurn_Stage;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
COUNT(*)
100
Result Grid



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).]]

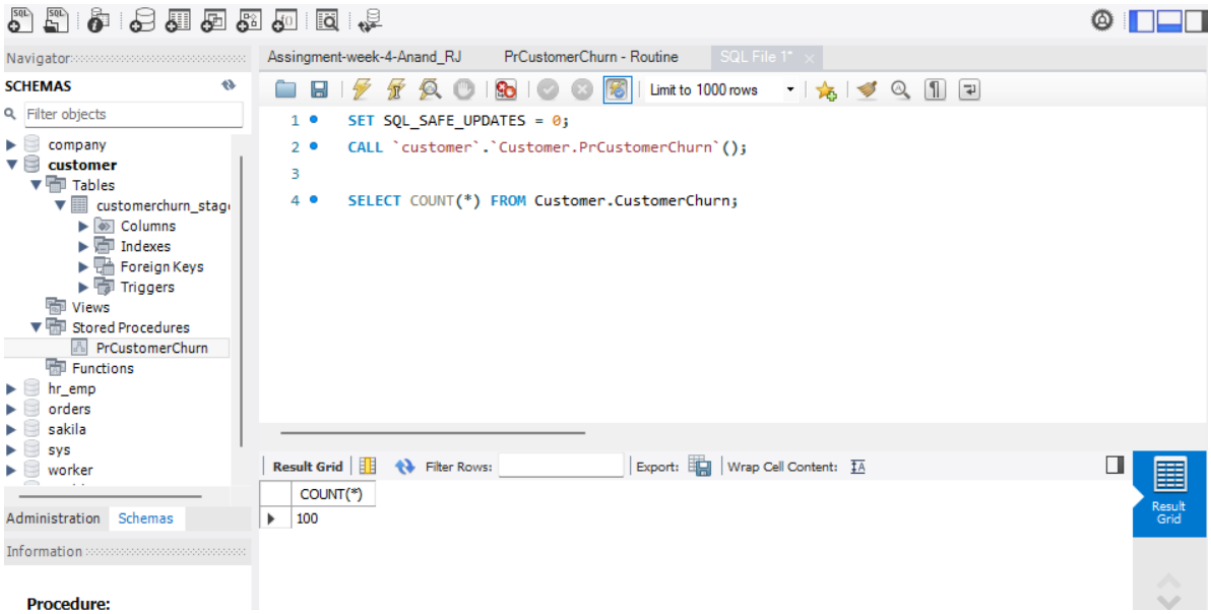
I have Attached the SQL Document in the Submission

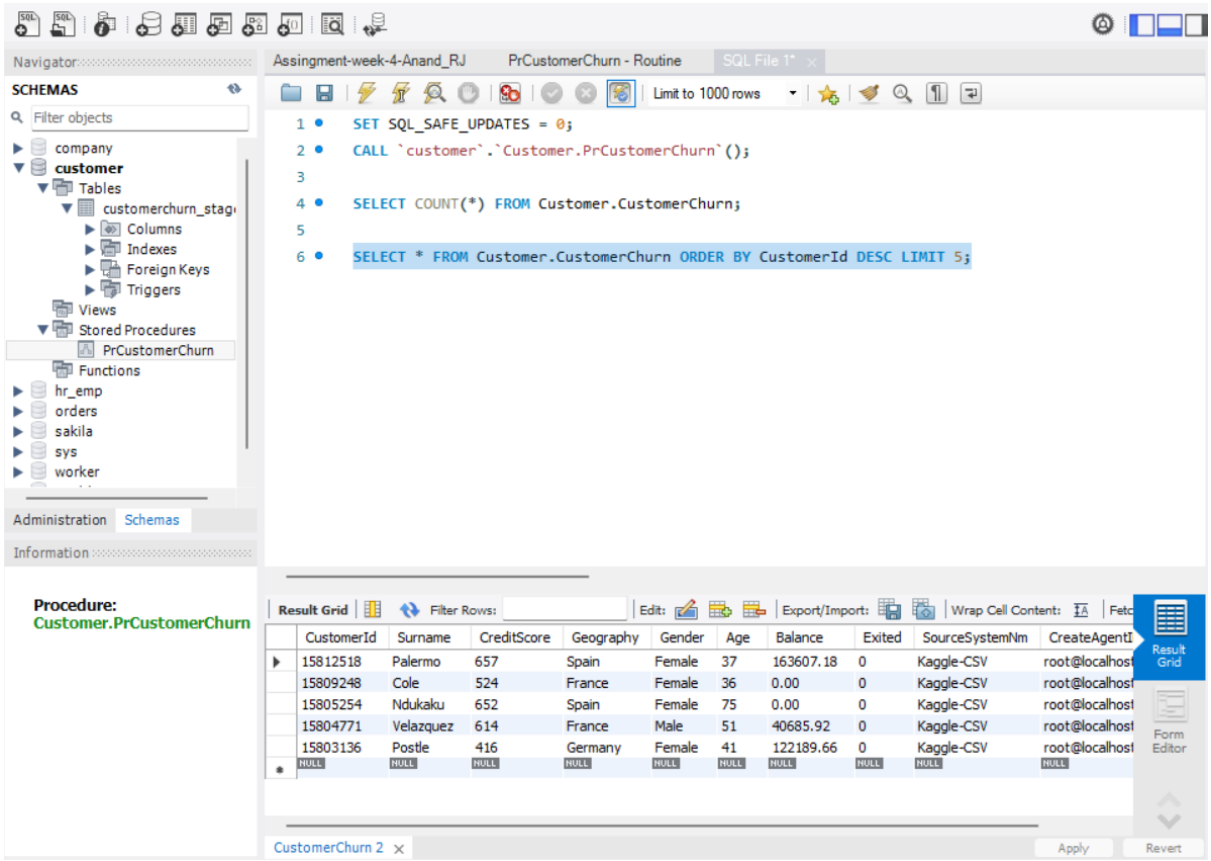




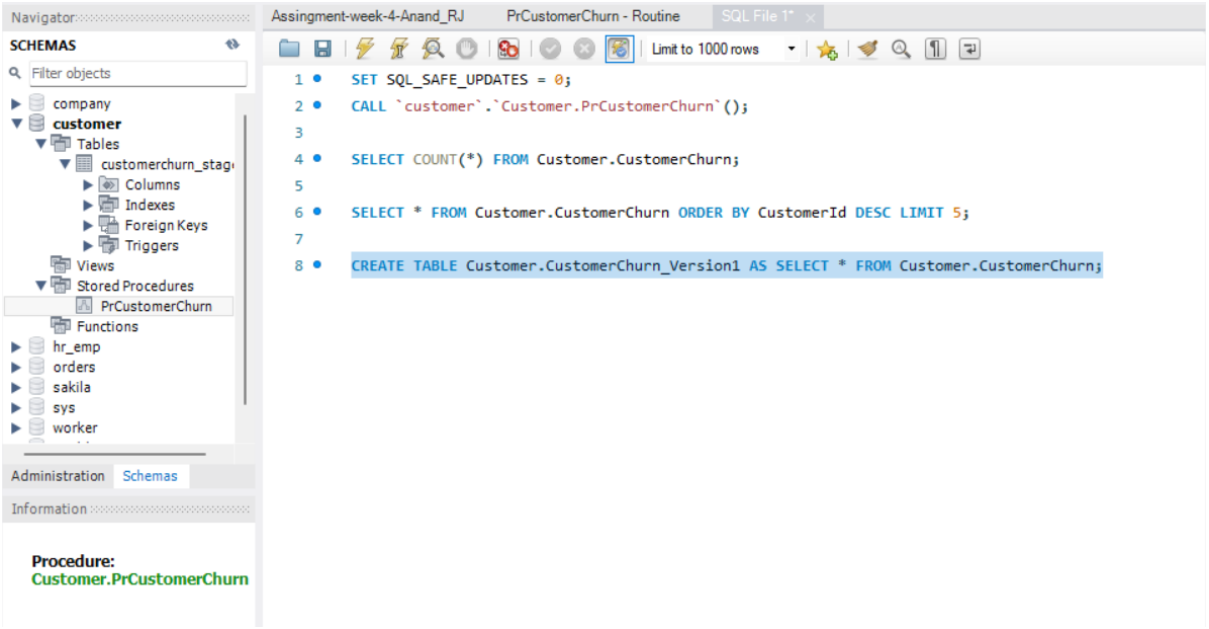
Q5. Execute the stored procedure, ** Customer.PrCustomerChurn **, that was created in Q4. After execution, the stored procedure should load data from the stage to the persistent table: ** Customer.CustomerChurn **. {A} Verify data by comparing the row counts between the staging table, ** Customer.CustomerChurn_Stage [Data Source:

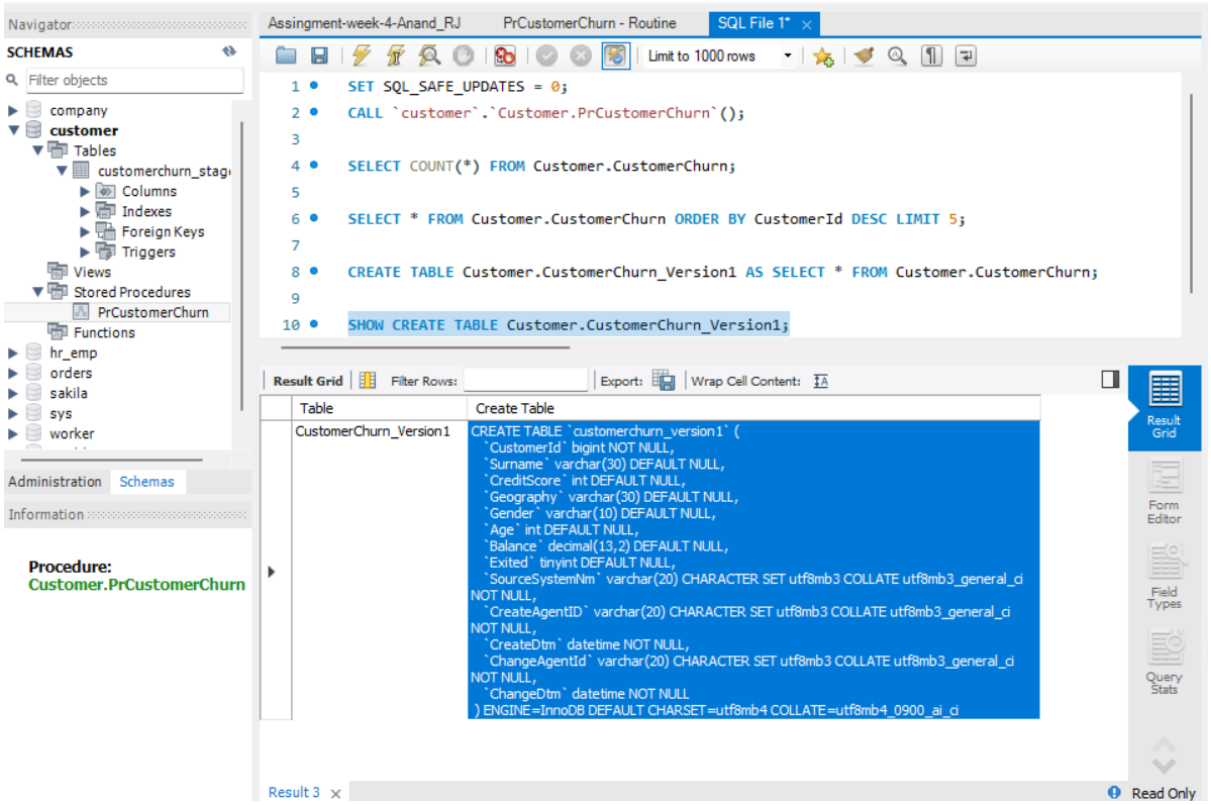
CustomerChurn1.CSV] ** and the persistent table: ** Customer.CustomerChurn **. { B } Provide the screenshot of last few rows using the SELECT *. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId.

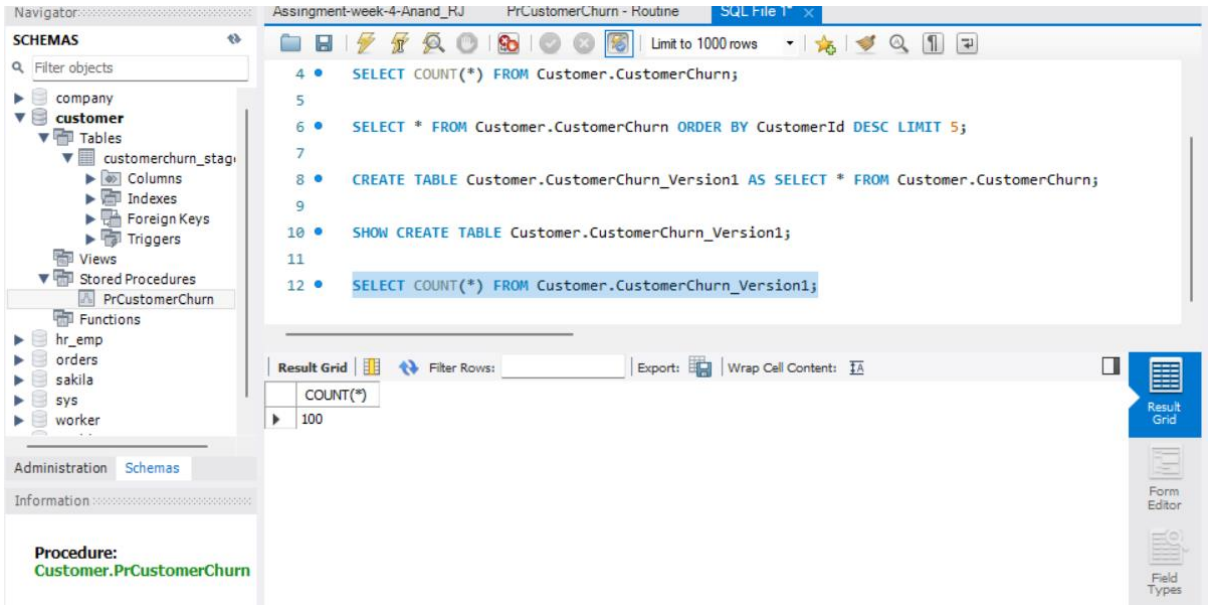


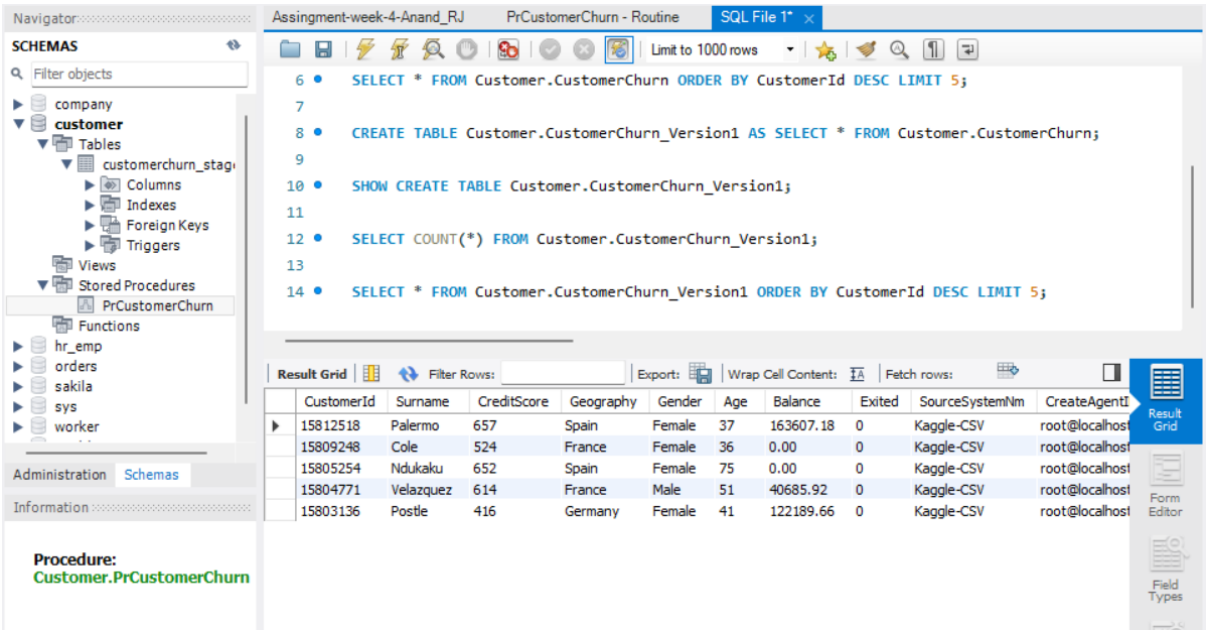


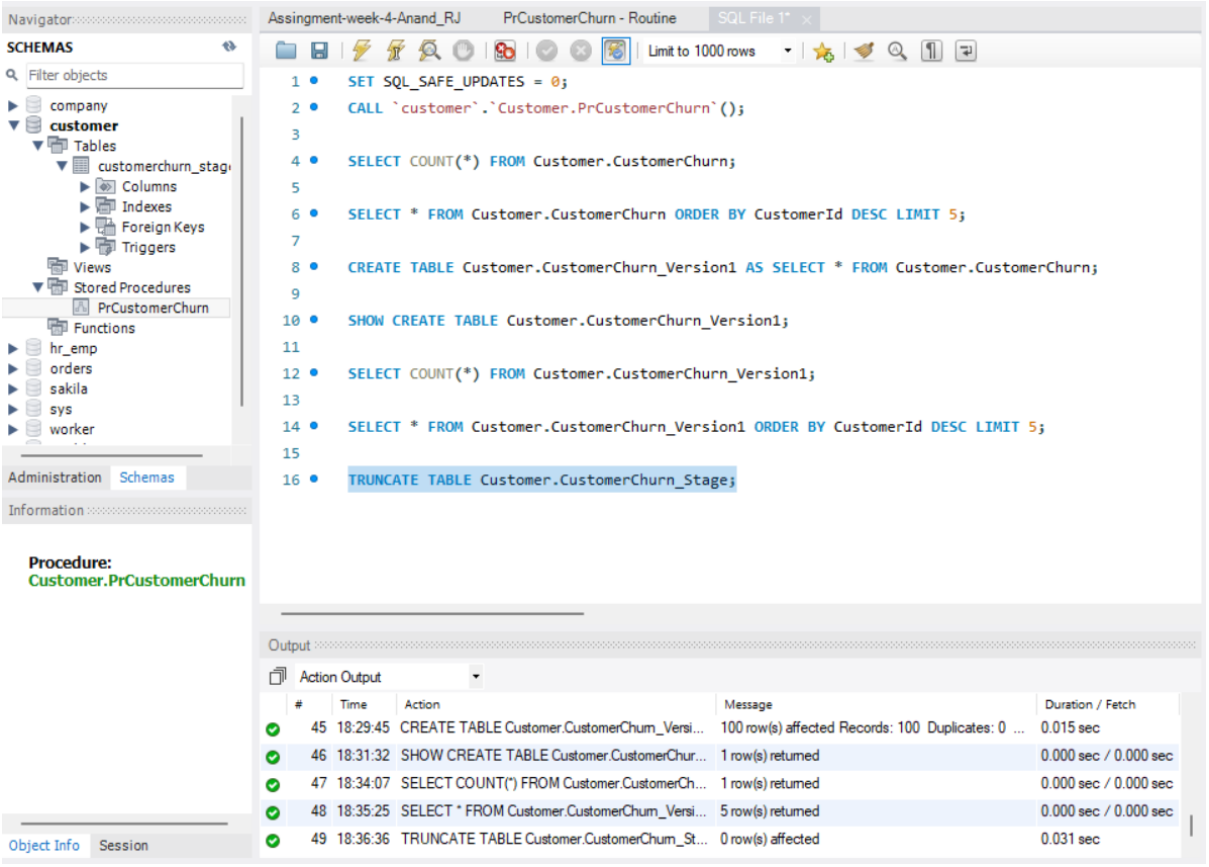
Q6. After data verification is completed, in Q5 , { A } create table, ** Customer.CustomerChurn_Version1 **, with data from ** Customer.CustomerChurn ** (that was already loaded from Customer.CustomerChurn_Stage via the stored procedure). { B } Show table definition of Customer.CustomerChurn_Version1 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.




Navigator:
SCHEMAS
Filter objects
company
customer
Tables
customerchurn_stag
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
PrCustomerChurn
Functions
hr_emp
orders
sakila
sys
worker
Administration
Schemas
Information
Procedure:
Customer.PrCustomerChurn
Assingment-week-4-Anand_RJ
PrCustomerChurn - Routine
SQL File 1*
Limit to 1000 rows
1 SET SQL_SAFE_UPDATES = 0;
2 CALL `customer`.`Customer.PrCustomerChurn`();
3
4 SELECT COUNT(*) FROM Customer.CustomerChurn;
5
6 SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId DESC LIMIT 5;
7
8 CREATE TABLE Customer.CustomerChurn_Version1 AS SELECT * FROM Customer.CustomerChurn;
9
10 SHOW CREATE TABLE Customer.CustomerChurn_Version1;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Table
Create Table
CustomerChurn_Version1
CREATE TABLE `customerchurn_version1` (
`CustomerId` bigint NOT NULL,
`Surname` varchar(30) DEFAULT NULL,
`CreditScore` int DEFAULT NULL,
`Geography` varchar(30) DEFAULT NULL,
`Gender` varchar(10) DEFAULT NULL,
`Age` int DEFAULT NULL,
`Balance` decimal(13,2) DEFAULT NULL,
`Exited` tinyint DEFAULT NULL,
`SourceSystemNm` varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
`CreateAgentID` varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
`CreateDtm` datetime NOT NULL,
`ChangeAgentId` varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
`ChangeDtm` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
Result Grid
Form Editor
Field Types
Query Stats
Result 3
Read Only


Navigator:
SCHEMAS
Filter objects
company
customer
Tables
customerchurn_stag
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
PrCustomerChurn
Functions
hr_emp
orders
sakila
sys
worker
Administration
Schemas
Information
Procedure:
Customer.PrCustomerChurn
Assingment-week-4-Anand_RJ
PrCustomerChurn - Routine
SQL File 1*
Limit to 1000 rows
4 SELECT COUNT(*) FROM Customer.CustomerChurn;
5
6 SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId DESC LIMIT 5;
7
8 CREATE TABLE Customer.CustomerChurn_Version1 AS SELECT * FROM Customer.CustomerChurn;
9
10 SHOW CREATE TABLE Customer.CustomerChurn_Version1;
11
12 SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
COUNT(*)
100
Result Grid
Form Editor
Field Types


Navigator
SCHEMAS
Filter objects
company
customer
Tables
customerchurn_stag
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
PrCustomerChurn
Functions
hr_emp
orders
sakila
sys
worker
Administration
Schemas
Information
Procedure:
Customer.PrCustomerChurn
Assingment-week-4-Anand_RJ
PrCustomerChurn - Routine
SQL File 1*
Limit to 1000 rows
6 SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId DESC LIMIT 5;
8 CREATE TABLE Customer.CustomerChurn_Version1 AS SELECT * FROM Customer.CustomerChurn;
10 SHOW CREATE TABLE Customer.CustomerChurn_Version1;
12 SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;
14 SELECT * FROM Customer.CustomerChurn_Version1 ORDER BY CustomerId DESC LIMIT 5;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
CustomerId Surname CreditScore Geography Gender Age Balance Exited SourceSystemNm CreateAgentI
15812518 Palermo 657 Spain Female 37 163607.18 0 Kaggle-CSV root@localhost
15809248 Cole 524 France Female 36 0.00 0 Kaggle-CSV root@localhost
15805254 Ndukaku 652 Spain Female 75 0.00 0 Kaggle-CSV root@localhost
15804771 Velazquez 614 France Male 51 40685.92 0 Kaggle-CSV root@localhost
15803136 Postle 416 Germany Female 41 122189.66 0 Kaggle-CSV root@localhost
Result Grid
Form Editor
Field Types


Navigator
SCHEMAS
Filter objects
company
customer
Tables
customerchurn_stag
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
PrCustomerChurn
Functions
hr_emp
orders
sakila
sys
worker
Administration
Schemas
Information
Procedure:
Customer.PrCustomerChurn
Object Info
Session
Assingment-week-4-Anand_RJ
PrCustomerChurn - Routine
SQL File 1*
Limit to 1000 rows
1 SET SQL_SAFE_UPDATES = 0;
2 CALL `customer`.`Customer.PrCustomerChurn`();
4 SELECT COUNT(*) FROM Customer.CustomerChurn;
6 SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId DESC LIMIT 5;
8 CREATE TABLE Customer.CustomerChurn_Version1 AS SELECT * FROM Customer.CustomerChurn;
10 SHOW CREATE TABLE Customer.CustomerChurn_Version1;
12 SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;
14 SELECT * FROM Customer.CustomerChurn_Version1 ORDER BY CustomerId DESC LIMIT 5;
16 TRUNCATE TABLE Customer.CustomerChurn_Stage;
Output
Action Output
Time Action Message Duration / Fetch
45 18:29:45 CREATE TABLE Customer.CustomerChurn_Versi... 100 row(s) affected Records: 100 Duplicates: 0 ... 0.015 sec
46 18:31:32 SHOW CREATE TABLE Customer.CustomerChur... 1 row(s) returned 0.000 sec / 0.000 sec
47 18:34:07 SELECT COUNT(*) FROM Customer.CustomerCh... 1 row(s) returned 0.000 sec / 0.000 sec
48 18:35:25 SELECT * FROM Customer.CustomerChurn_Versi... 5 row(s) returned 0.000 sec / 0.000 sec
49 18:36:36 TRUNCATE TABLE Customer.CustomerChurn_St... 0 row(s) affected 0.031 sec

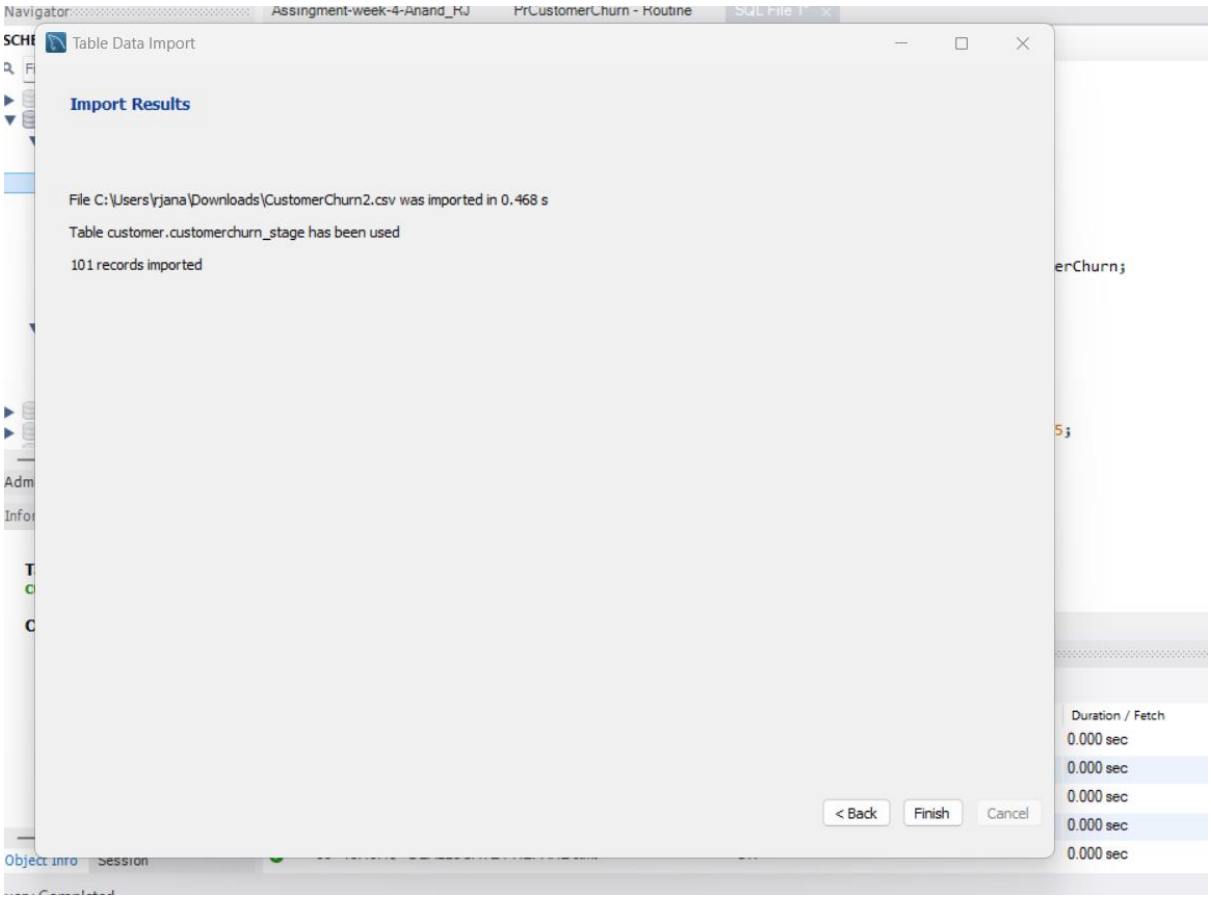
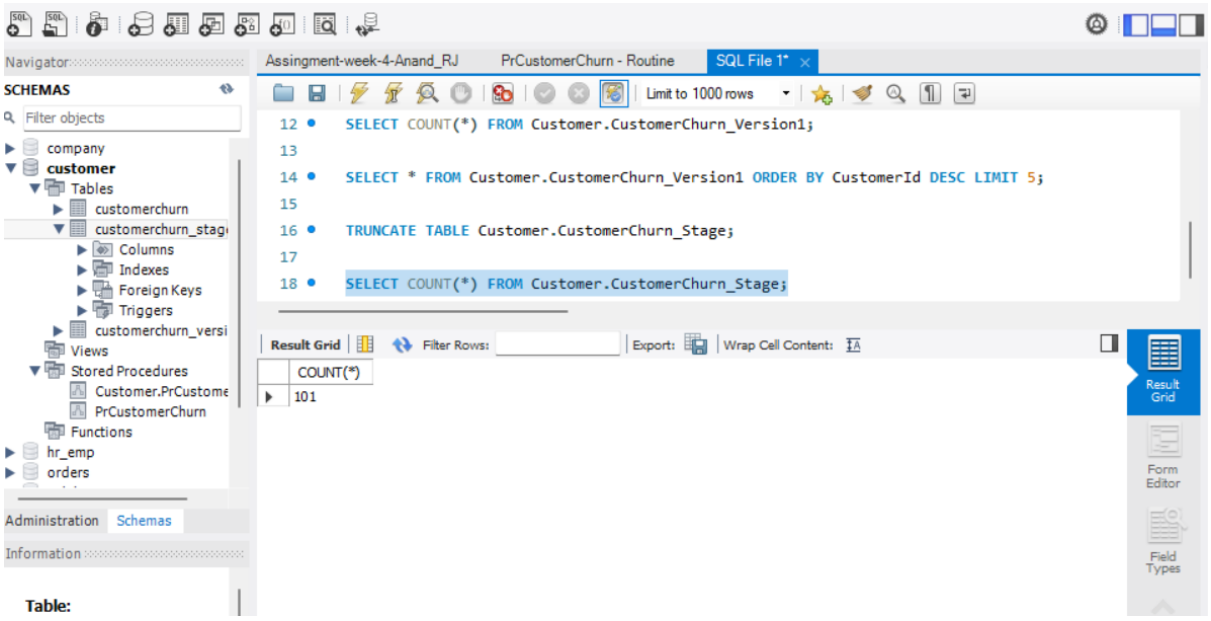
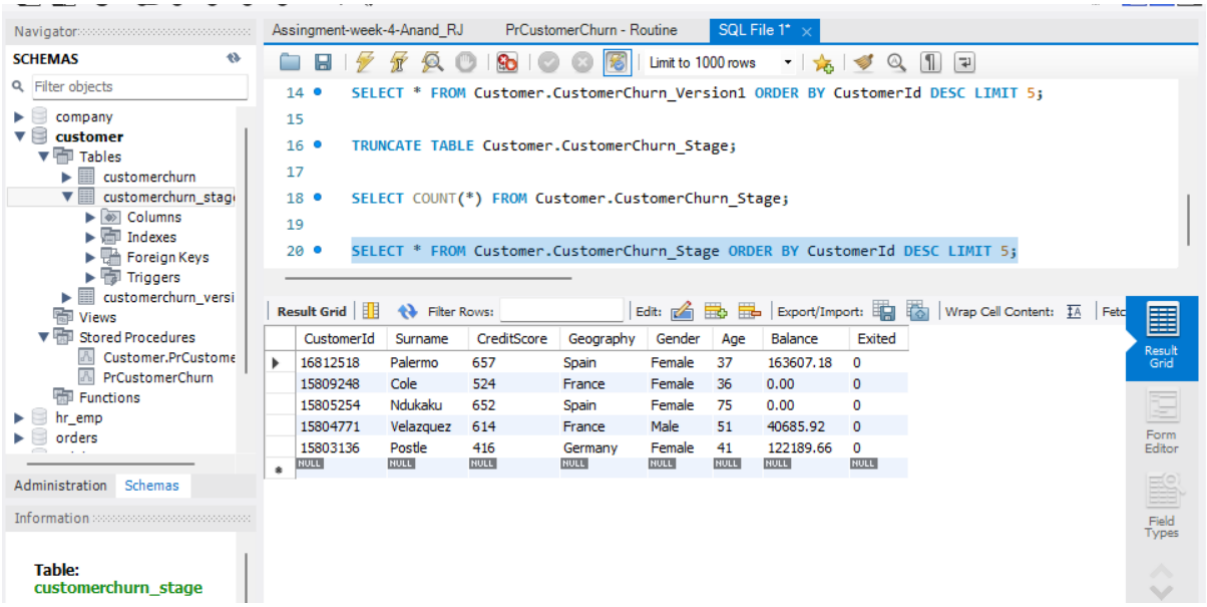
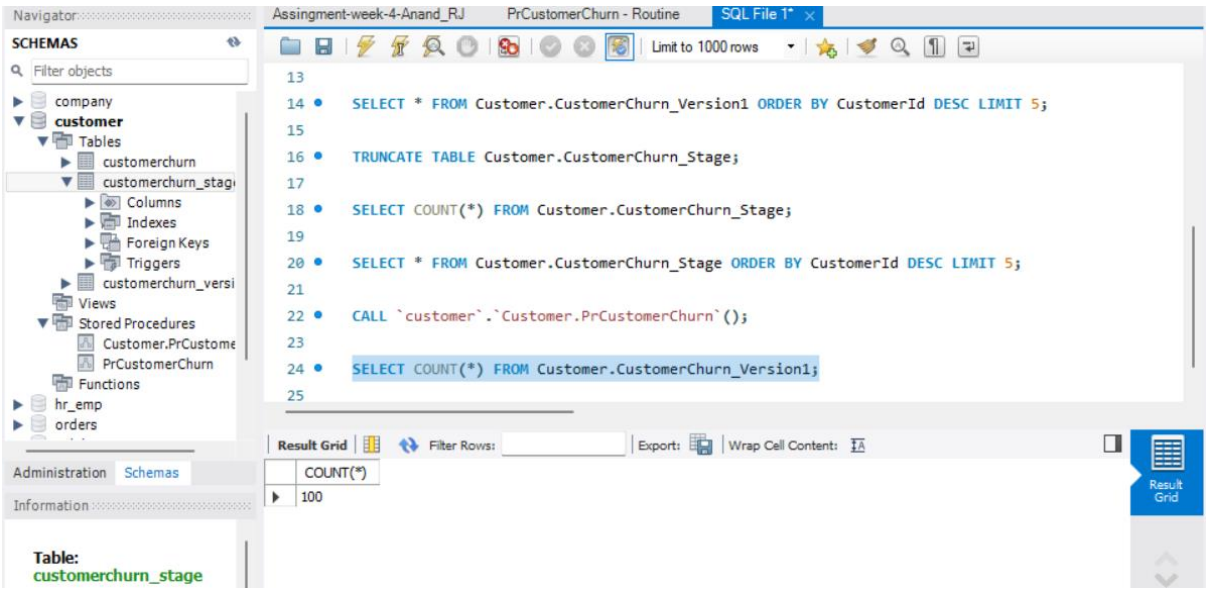


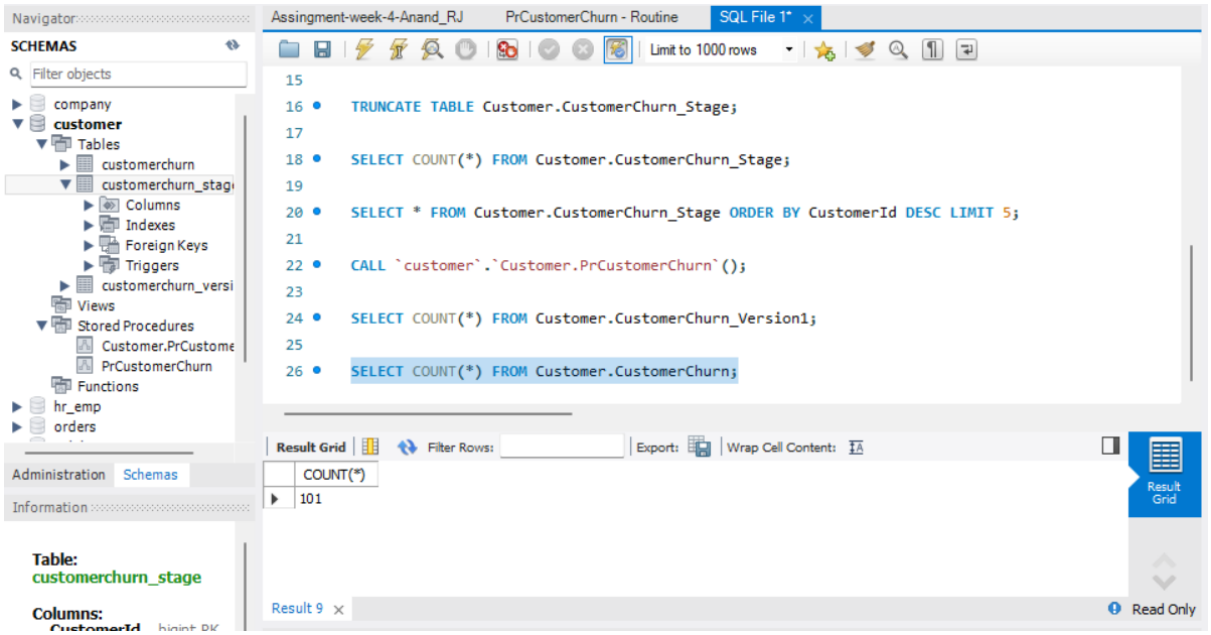
Table Data Import
Import Results
File C:\Users\rjana\Downloads\CustomerChurn2.csv was imported in 0.468 s
Table customer.customerchurn_stage has been used
101 records imported
< Back
Finish
Cancel
Duration / Fetch
0.000 sec
0.000 sec
0.000 sec
0.000 sec
0.000 sec

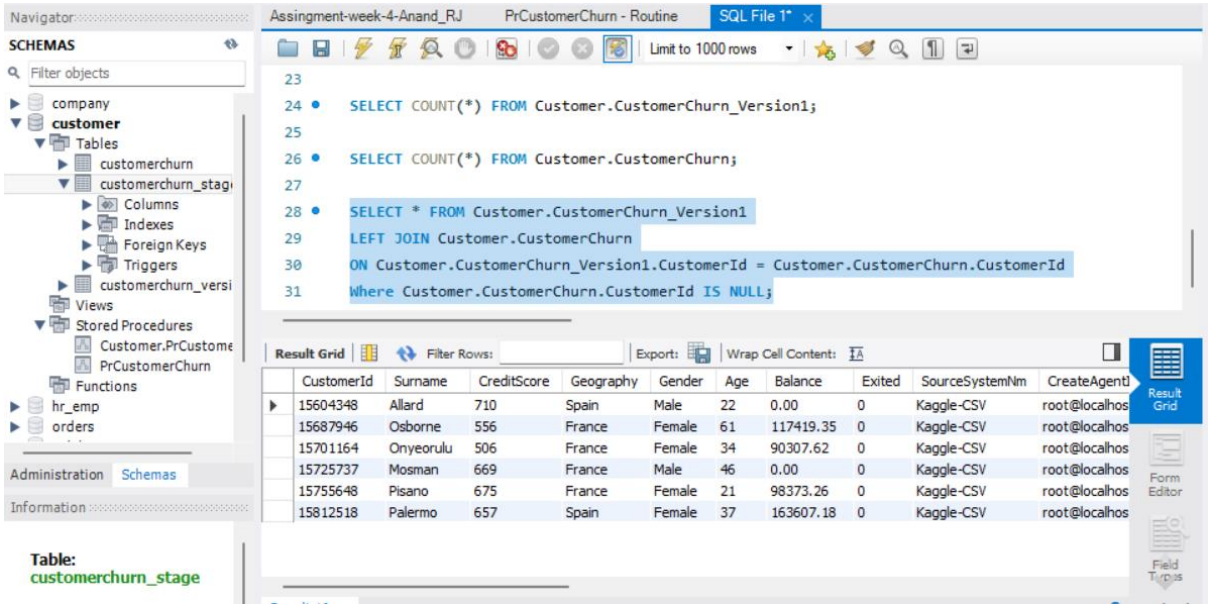

Navigator
SCHEMAS
Filter objects
company
customer
Tables
customerchurn
customerchurn_stage
Columns
Indexes
Foreign Keys
Triggers
customerchurn_versi
Views
Stored Procedures
Customer.PrCustome
PrCustomerChurn
Functions
hr_emp
orders
Administration
Schemas
Information
Table:
Assingment-week-4-Anand_RJ
PrCustomerChurn - Routine
SQL File 1*
Limit to 1000 rows
12 SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;
13
14 SELECT * FROM Customer.CustomerChurn_Version1 ORDER BY CustomerId DESC LIMIT 5;
15
16 TRUNCATE TABLE Customer.CustomerChurn_Stage;
17
18 SELECT COUNT(*) FROM Customer.CustomerChurn_Stage;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
COUNT(*)
101
Result Grid
Form Editor
Field Types



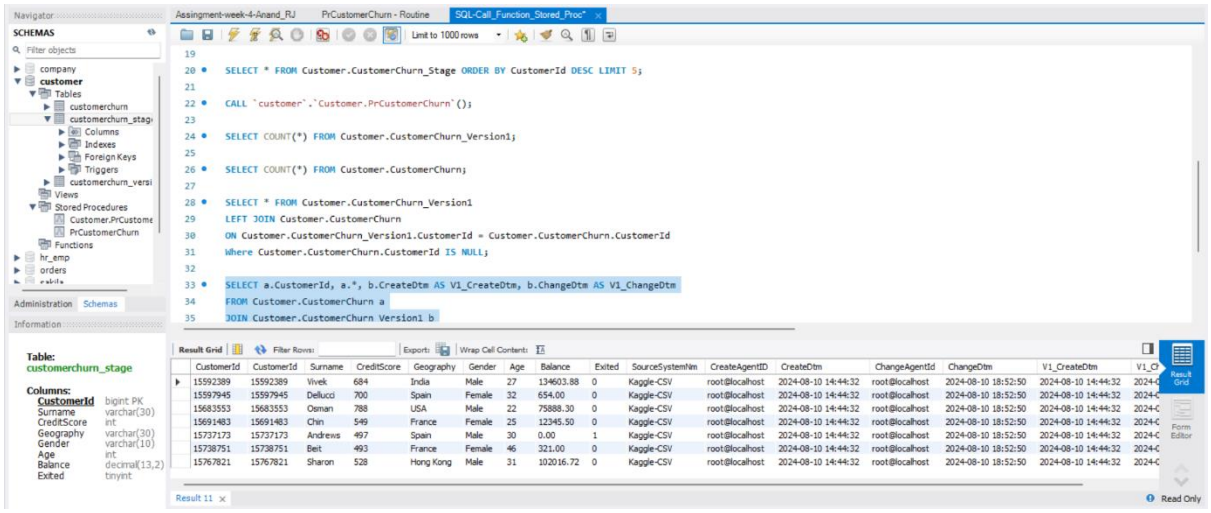
Q7. Execute the stored procedure, Customer.PrCustomerChurn, that was created in Q4. After execution, the stored procedure should load data from the stage to the persistent table: Customer.CustomerChurn. CALL `customer`.`PrCustomerChurn`(); This time, the table will be refreshed via DELETE, UPDATE, and 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).



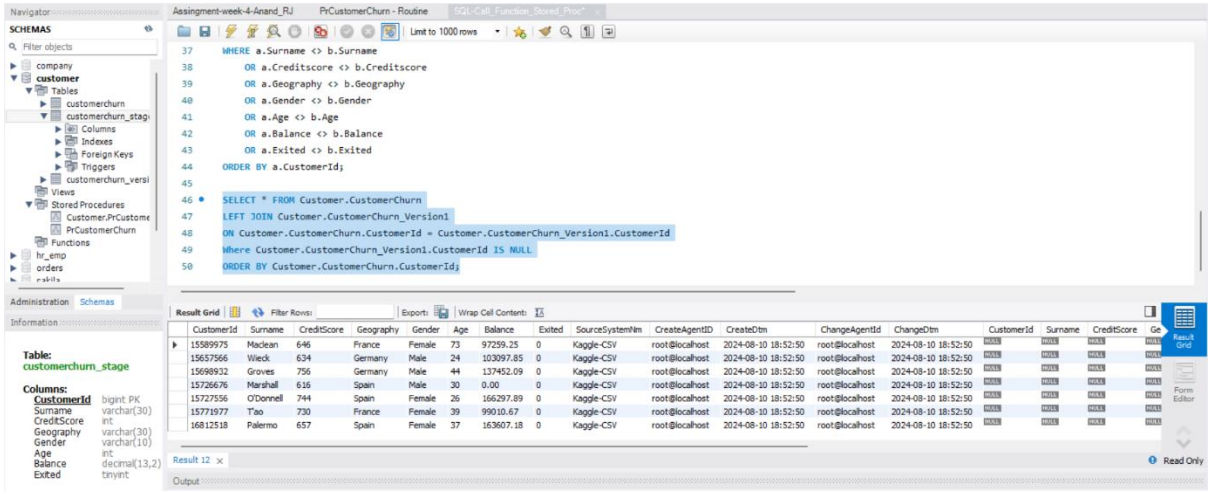




Q8. Show the rows (SELECT *) that changed (one or many non-Primary Key columns), in the Customer.CustomerChurn table (implementation of UPDATE statement of the stored procedure). You need to perform a comparison between Customer.CustomerChurn table [Data Source: CustomerChurn2.CSV] and Customer.CustomerChurn_Version1 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.



Q10. Show the final version of the stored procedure code that was used to load the persistent table, Customer.CustomerChurn. Submit it as a *.TXT file. Submit your work (QUESTIONS 1-9) in a single Word Document/PDF file.

Attached the SQL-Call_Function_Stored_Proc.txt file