Module/Week4: Assignment 3

Topic: Incremental Refresh in Data Warehousing

Student's Full Name: LIBIN NAITHELLOOR GEORGE

Course Title: Data Warehousing and Analytics in the Cloud

Term name and year: Fall 2023

Submission Week: [example, Week 3- Assignment 2]

Instructor's Name: Dr.Nayem Rahman

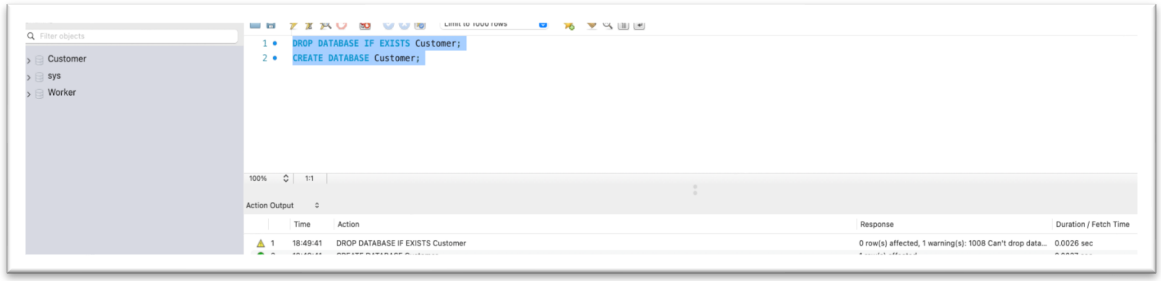
Date of Submission: 27 July 2024

PURPOSE:

This assignment introduces you to the incremental refresh techniques in a data warehouse. As part of this assignment, you demonstrate your understanding of the database system, basic SQL, and incremental refresh techniques via database stored procedures. In real-world business, data warehouses are refreshed with data from operational database systems. In data warehouse refreshments, incremental techniques are used. This is a very important task that a data engineer does.

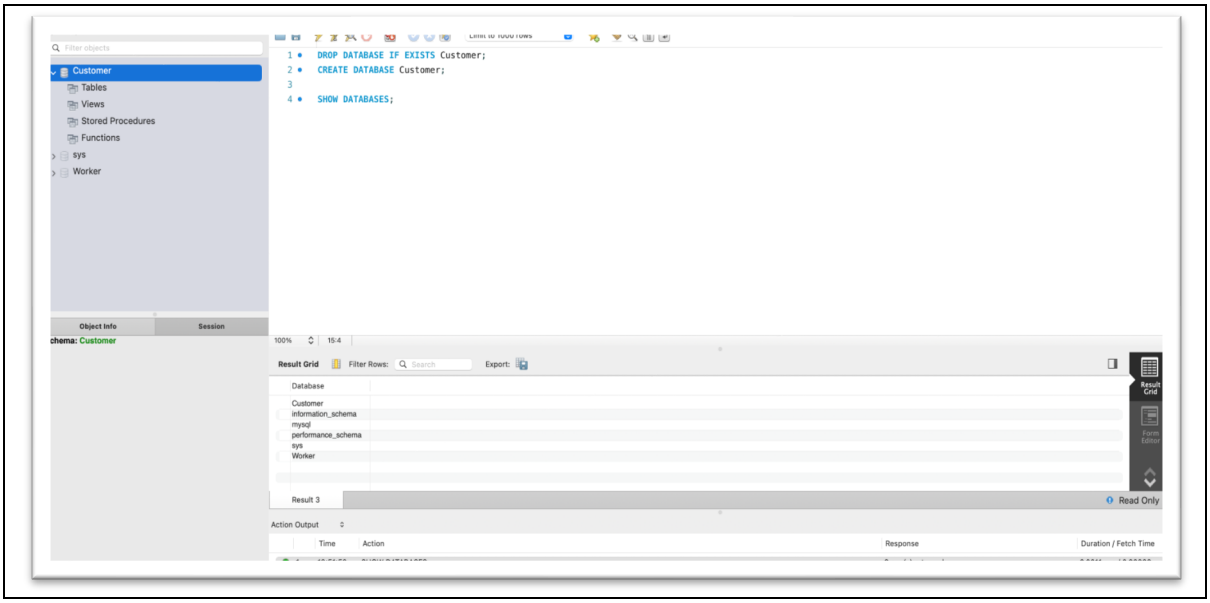
Q1.
{ A } Using the MySQL Workbench, create a database called Customer. The database must be named "Customer".

```
CREATE DATABASE Customer;
```



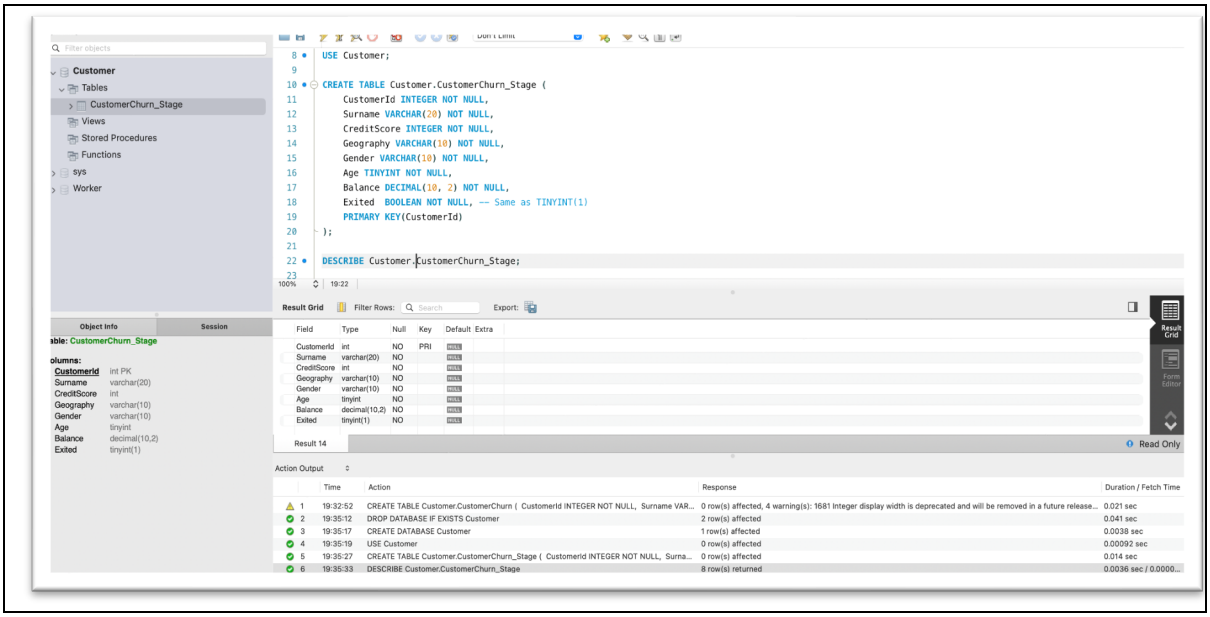
{ B } Check if the database was created and use the same for further questions.

```
SHOW DATABASES;
```



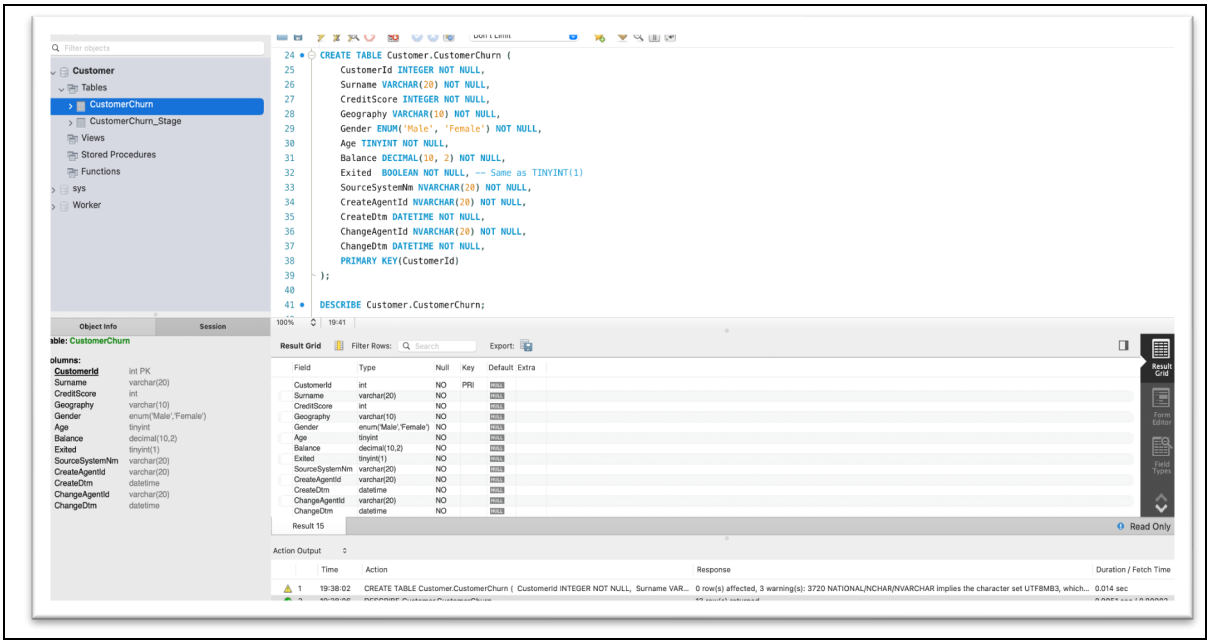
Q2. { A } Create a staging table, ** Customer.CustomerChurn_Stage **, in a database system, with the column list provided in the CSV file. Define the ' CustomerId ' as the Primary Key (PK). Get the table definition (DDL) from the database system and capture it in a Word document for submission.

```
CREATE TABLE Customer.CustomerChurn_Stage (
    CustomerId INTEGER NOT NULL,
   Surname VARCHAR(20) NOT NULL,
   CreditScore INTEGER NOT NULL,
   Geography VARCHAR(10) NOT NULL,
   Gender VARCHAR(10) NOT NULL,
   Age TINYINT NOT NULL,
   Balance DECIMAL(10, 2) NOT NULL,
   Exited  BOOLEAN NOT NULL,
   PRIMARY KEY(CustomerId)
);

DESCRIBE Customer.CustomerChurn_Stage;
```



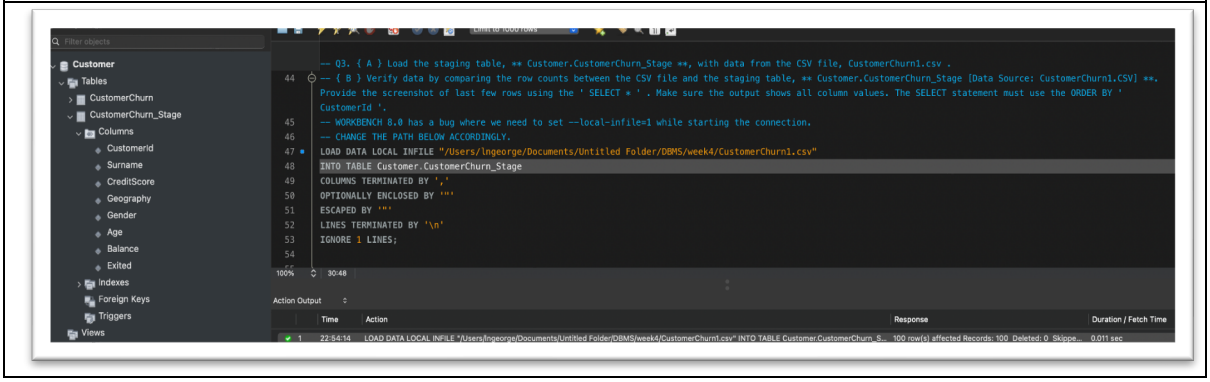
{ B } Create a persistent table, ** Customer.CustomerChurn **, with the column list provided in the CSV file + following 5 columns : << SourceSystemNm NVARCHAR(20) NOT NULL , CreateAgentId NVARCHAR(20) NOT NULL , CreateDtm DATETIME NOT NULL, ChangeAgentId NVARCHAR(20) NOT NULL , ChangeDtm DATETIME NOT 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.

```
CREATE TABLE Customer.CustomerChurn (
   CustomerId INTEGER NOT NULL,
   Surname VARCHAR(20) NOT NULL,
   CreditScore INTEGER NOT NULL,
   Geography VARCHAR(10) NOT NULL,
   Gender ENUM('Male', 'Female') NOT NULL,
   Age TINYINT NOT NULL,
   Balance DECIMAL(10, 2) NOT NULL,
   Exited  BOOLEAN NOT NULL, -- Same as TINYINT(1)
   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)
);

DESCRIBE Customer.CustomerChurn;
```



Q3. { A } Load the staging table, ** Customer.CustomerChurn_Stage **, with data from the CSV file, CustomerChurn1.csv .

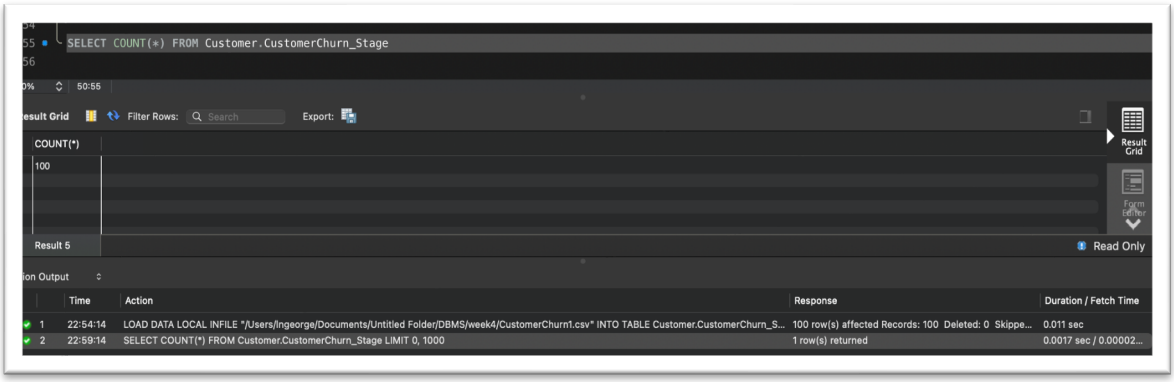
```
LOAD DATA LOCAL INFILE "/Users/lngeorge/Documents/Untitled
Folder/DBMS/week4/CustomerChurn1.csv"
INTO TABLE Customer.CustomerChurn_Stage
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```



Another option would be to use import wizard to import from CSV file.

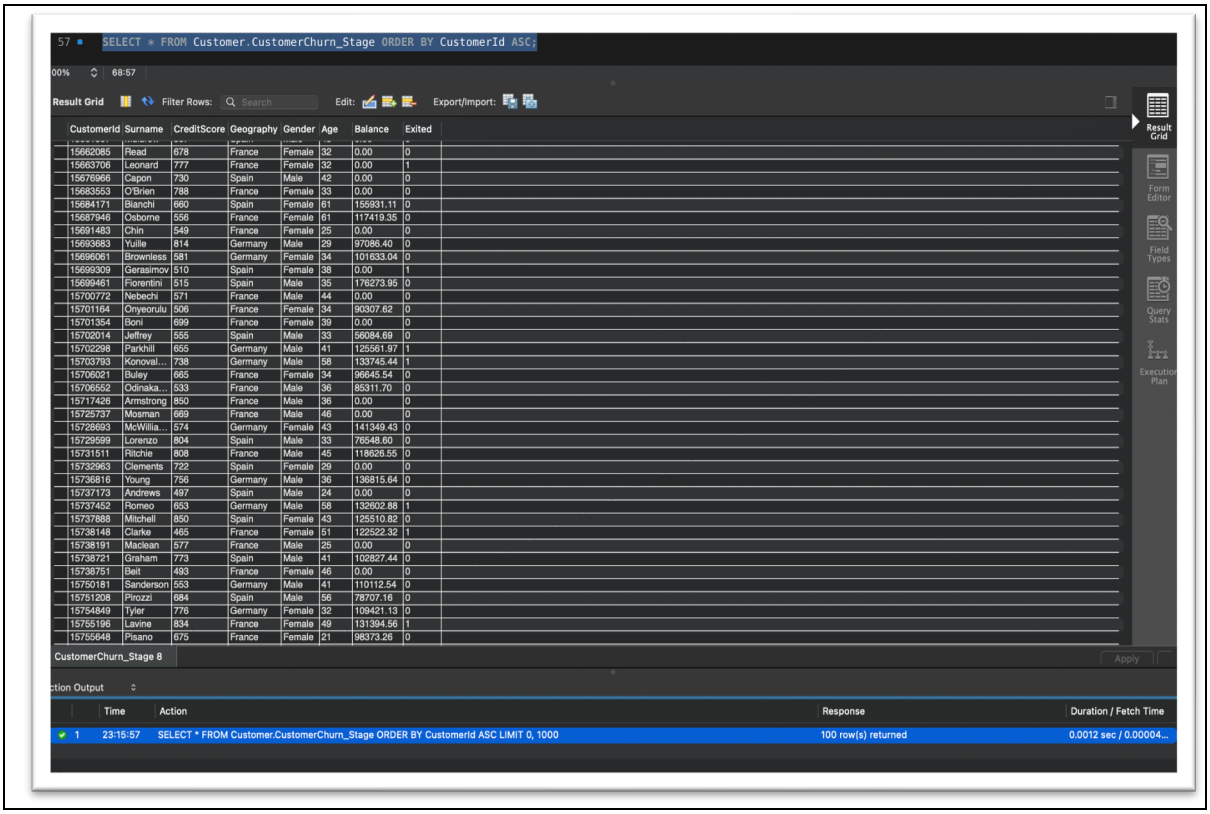
{ B } Verify data by comparing the row counts between the CSV file and the staging table, ** Customer.CustomerChurn_Stage [Data Source: CustomerChurn1.CSV] **. Provide the screenshot of last few rows using the ' SELECT * ' . Make sure the output shows all column values. The SELECT statement must use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

Both Table and CSV has same amount of rows (Count: 100)

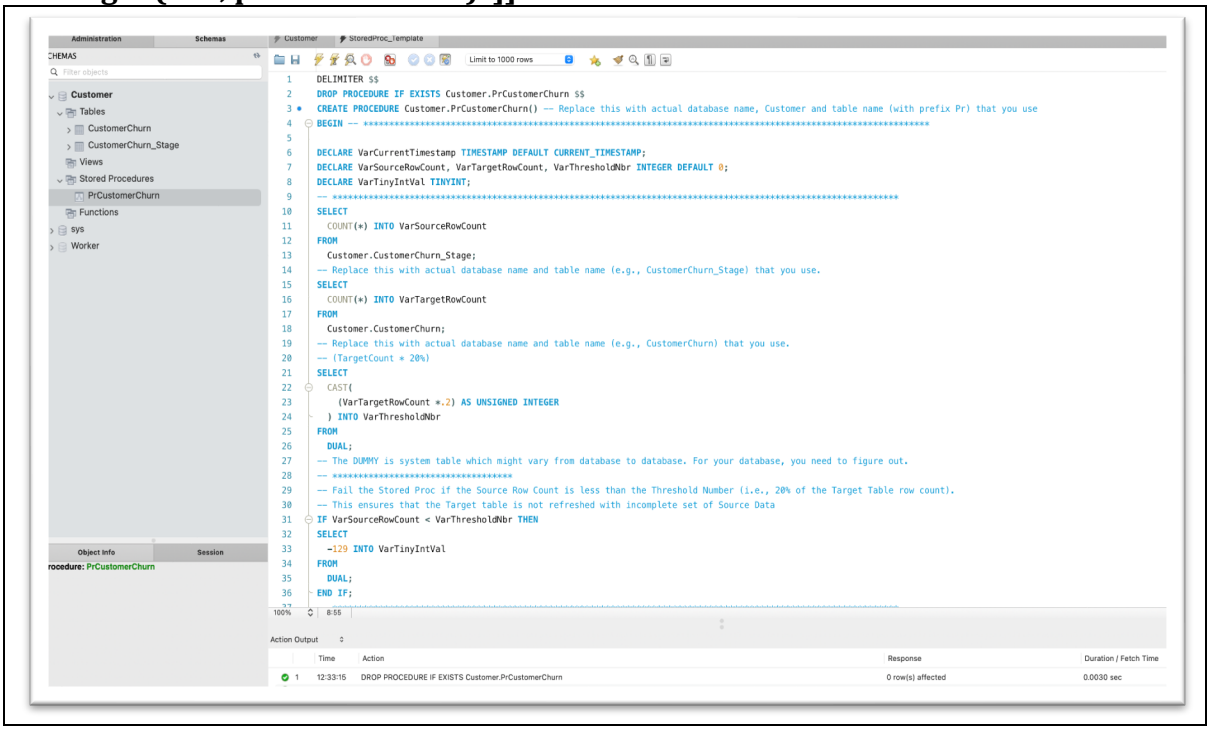


	A	B	C	D	E	F	G	H
1	Custom	Surn	CreditSc	Geogra	Gen	Age	Bala	Exite
2	15634602	Hargrave	619	France	Female	42	0	1
3	15647311	Hill	608	Spain	Female	41	83807.9	0
4	15619304	Onio	502	France	Female	42	159661	1
5	15701354	Boni	699	France	Female	39	0	0
6	15737888	Mitchell	850	Spain	Female	43	125511	0
7	15574012	Chu	645	Spain	Male	44	113756	1
8	15592531	Bartlett	822	France	Male	50	0	0
9	15656148	Obinna	376	Germany	Female	29	115047	1
10	15792365	He	501	France	Male	44	142051	0
11	15592389	H?	684	France	Male	27	134604	0
12	15767821	Bearce	528	France	Male	31	102017	0
13	15737173	Andrews	497	Spain	Male	24	0	0
14	15632264	Kay	476	France	Female	34	0	0
15	15691483	Chin	549	France	Female	25	0	0
16	15600882	Scott	635	Spain	Female	35	0	0
17	15643966	Goforth	616	Germany	Male	45	143129	0
18	15737452	Romeo	653	Germany	Male	58	132603	1
19	15788218	Henderson	549	Spain	Female	24	0	0
20	15661507	Muldrow	587	Spain	Male	45	0	0
21	15568982	Hao	726	France	Female	24	0	0
22	15577657	McDonald	732	France	Male	41	0	0
23	15597945	Dellucci	636	Spain	Female	32	0	0
24	15699309	Gerasimov	510	Spain	Female	38	0	1
25	15725737	Mosman	669	France	Male	46	0	0
26	15625047	Yen	846	France	Female	38	0	0
27	15738191	Maclean	577	France	Male	25	0	0
28	15736816	Young	756	Germany	Male	36	136816	0
29	15700772	Nebechi	571	France	Male	44	0	0
30	15728693	McWilliams	574	Germany	Female	43	141349	0
31	15656300	Lucciano	411	France	Male	29	59697.2	0
32	15589475	Azikiwe	591	Spain	Female	39	0	1
33	15706552	Odinakachuk	533	France	Male	36	85311.7	0
34	15750181	Sanderson	553	Germany	Male	41	110113	0
35	15659428	Maggard	520	Spain	Female	42	0	0
36	15732963	Clements	722	Spain	Female	29	0	0
37	15794171	Lombardo	475	France	Female	45	134264	1
38	15788448	Watson	490	Spain	Male	31	145260	0
39	15729599	Lorenzo	804	Spain	Male	33	76548.6	0
40	15717426	Armstrong	850	France	Male	36	0	0
41	15585768	Cameron	582	Germany	Male	41	70349.5	0
42	15619360	Hsiao	472	Spain	Male	40	0	0
43	15738148	Clarke	465	France	Female	51	122522	1
44	15687946	Osborne	556	France	Female	61	117419	0
45	15755196	Lavine	834	France	Female	49	131395	1
46	15684171	Bianchi	660	Spain	Female	61	155931	0
47	15754849	Tyler	776	Germany	Female	32	109421	0
48	15602280	Martin	829	Germany	Female	27	112046	1
49	15771573	Okagbue	637	Germany	Female	39	137844	1
50	15766205	Yin	550	Germany	Male	38	103391	0
51	15771873	Buccho	776	Germany	Female	37	103769	0
52	15616550	Chidiebele	698	Germany	Male	44	116363	0
53	15768193	Trevisani	585	Germany	Male	36	146051	0
54	15683553	O'Brien	788	France	Female	33	0	0
55	15702298	Parkhill	655	Germany	Male	41	125562	1
56	15569590	Yoo	601	Germany	Male	42	98495.7	1
57	15760861	Phillipps	619	France	Male	43	125212	0
58	15630053	Tsao	656	France	Male	45	127864	0
59	15647091	Endrizzi	725	Germany	Male	19	75888.2	0
60	15623944	T'ien	511	Spain	Female	66	0	1
61	15804771	Velazquez	614	France	Male	51	40685.9	0
62	15651280	Hunter	742	Germany	Male	35	136857	0
63	15773469	Clark	687	Germany	Female	27	152329	0
64	15702014	Jeffrey	555	Spain	Male	33	56084.7	0
65	15751208	Pirozzi	684	Spain	Male	56	78707.2	0
66	15592461	Jackson	603	Germany	Male	26	109166	0
67	15789484	Hammond	751	Germany	Female	36	169831	0
68	15696061	Brownless	581	Germany	Female	34	101633	0
69	15641582	Chibugo	735	Germany	Male	43	123180	0
70	15638424	Glauert	661	Germany	Female	35	150726	0
71	15755648	Pisano	675	France	Female	21	98373.3	0
72	15703793	Konovalova	738	Germany	Male	58	133745	1
73	15620344	McKee	813	France	Male	29	0	0
74	15812518	Palermo	657	Spain	Female	37	163607	0
75	15779052	Ballard	604	Germany	Female	25	157781	0
76	15770811	Wallace	519	France	Male	36	0	0
77	15780961	Cavenagh	735	France	Female	21	178718	0
78	15614049	Hu	664	France	Male	55	0	0
79	15662085	Read	678	France	Female	32	0	0
80	15575185	Bushell	757	Spain	Male	33	77253.2	0
81	15803136	Postle	416	Germany	Female	41	122190	0
82	15706021	Buley	665	France	Female	34	96645.5	0
83	15663706	Leonard	777	France	Female	32	0	1
84	15641732	Mills	543	France	Female	36	0	0
85	15701164	Onyeorulu	506	France	Female	34	90307.6	0
86	15738751	Beit	493	France	Female	46	0	0
87	15805254	Ndukaku	652	Spain	Female	75	0	0
88	15762418	Gant	750	Spain	Male	22	121682	1
89	15625759	Rowley	729	France	Male	30	0	0
90	15622897	Sharpe	646	France	Female	46	0	1
91	15767954	Osborne	635	Germany	Female	28	81623.7	0
92	15757535	Heap	647	Spain	Female	44	0	1
93	15731511	Ritchie	808	France	Male	45	118627	0
94	15809248	Cole	524	France	Female	36	0	0
95	15640635	Capon	769	France	Male	29	0	0
96	15676966	Capon	730	Spain	Male	42	0	0
97	15699461	Fiorentini	515	Spain	Male	35	176274	0
98	15738721	Graham	773	Spain	Male	41	102827	0
99	15693683	Yuille	814	Germany	Male	29	97086.4	0
100	15604348	Allard	710	Spain	Male	22	0	0
101	15633059	Fanucci	413	France	Male	34	0	0

```
SELECT * FROM Customer.CustomerChurn_Stage ORDER BY CustomerId ASC;
```

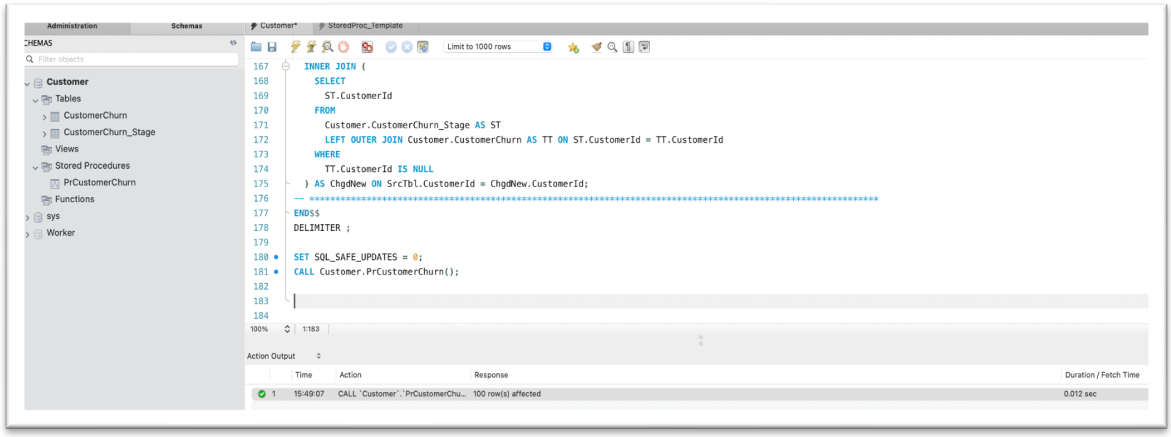


Q4. Create a database stored procedure based on the template provided along with this assignment << StoredProc_Template.txt >>. Name the stored procedure name this: ** Customer.PrCustomerChurn ** . [[NOTE : This stored procedure will use the table, ** Customer.CustomerChurn_Stage ** , as the source (aka,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

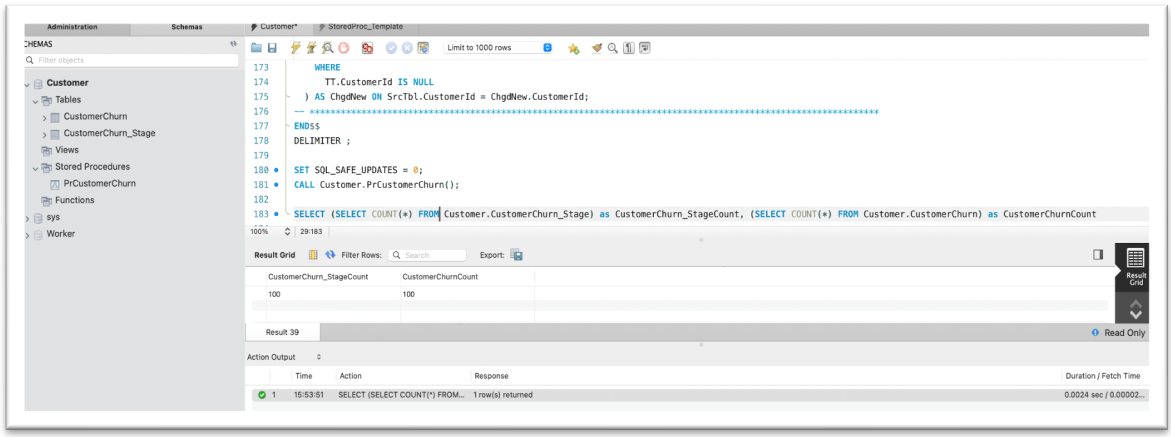


Q5. Execute the stored procedure, ** Customer.PrCustomerChurn **, that was created in Q4. After execution, the stored procedure should load data from the stage to the persistent table: ** Customer.CustomerChurn **.

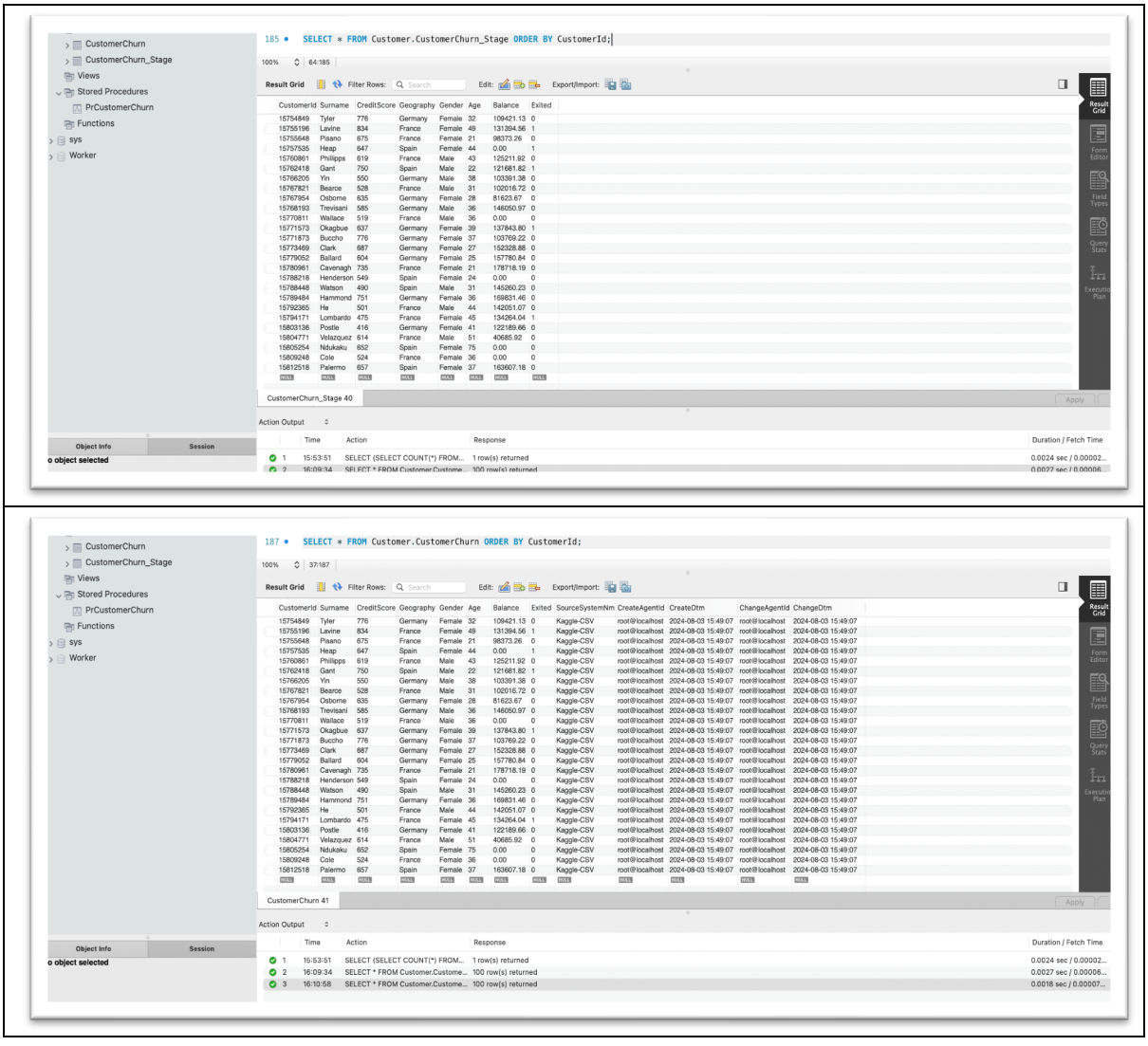
```
CALL Customer.PrCustomerChurn();
```



{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 **.



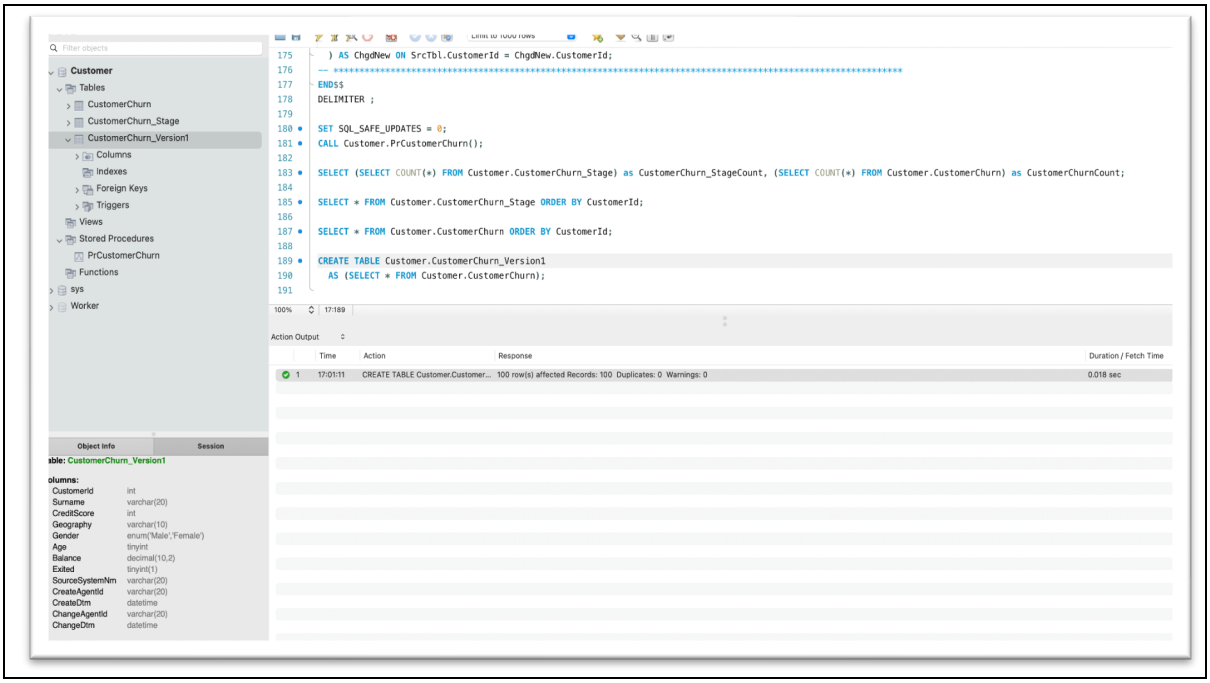
{ B } Provide the screenshot of last few rows using the SELECT *. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId.



Q6. After data verification is completed, in Q5 ,

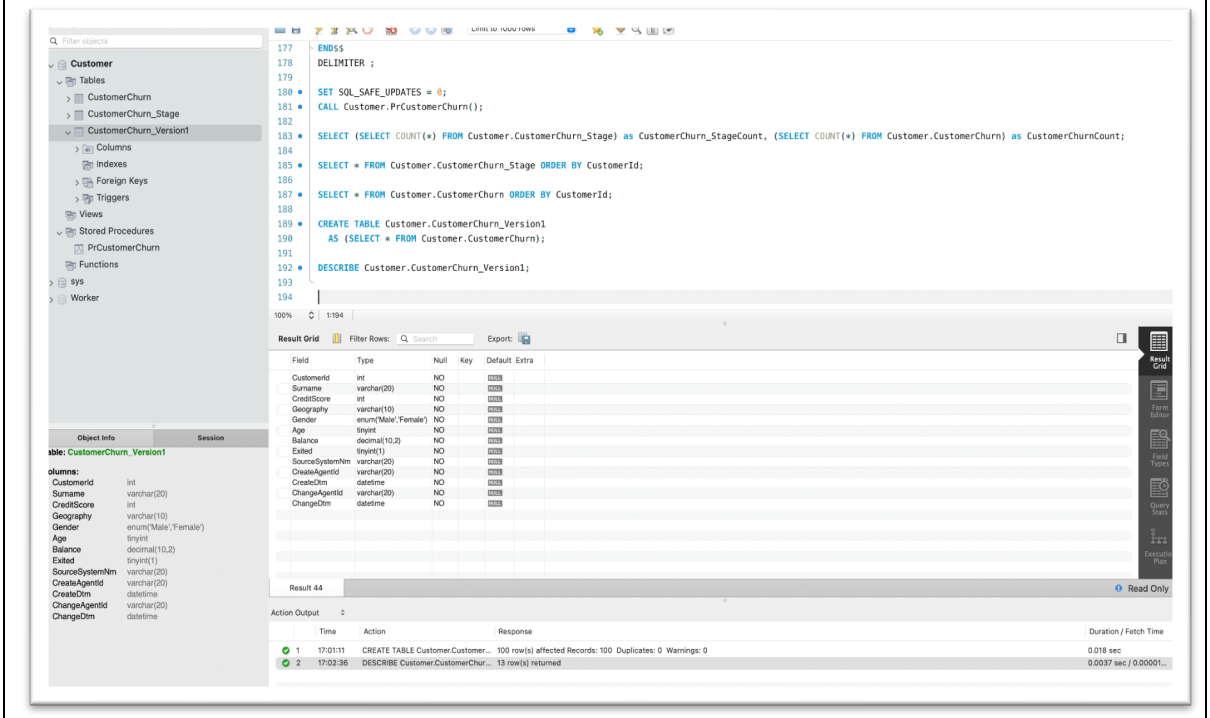
{ A } create table, ** Customer.CustomerChurn_Version1 **, with data from ** Customer.CustomerChurn ** (that was already loaded from Customer.CustomerChurn_Stage via the stored procedure).

```
CREATE TABLE Customer.CustomerChurn_Version1
  AS (SELECT * FROM Customer.CustomerChurn);
```

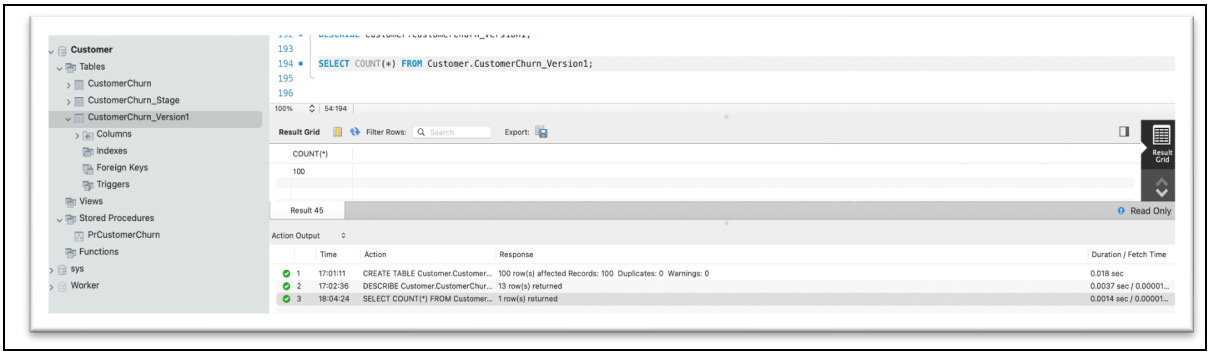


{ B } Show table definition of Customer.CustomerChurn_Version1 and show the row count of the table, ** Customer.CustomerChurn_Version1 **:

DESCRIBE Customer.CustomerChurn_Version1;

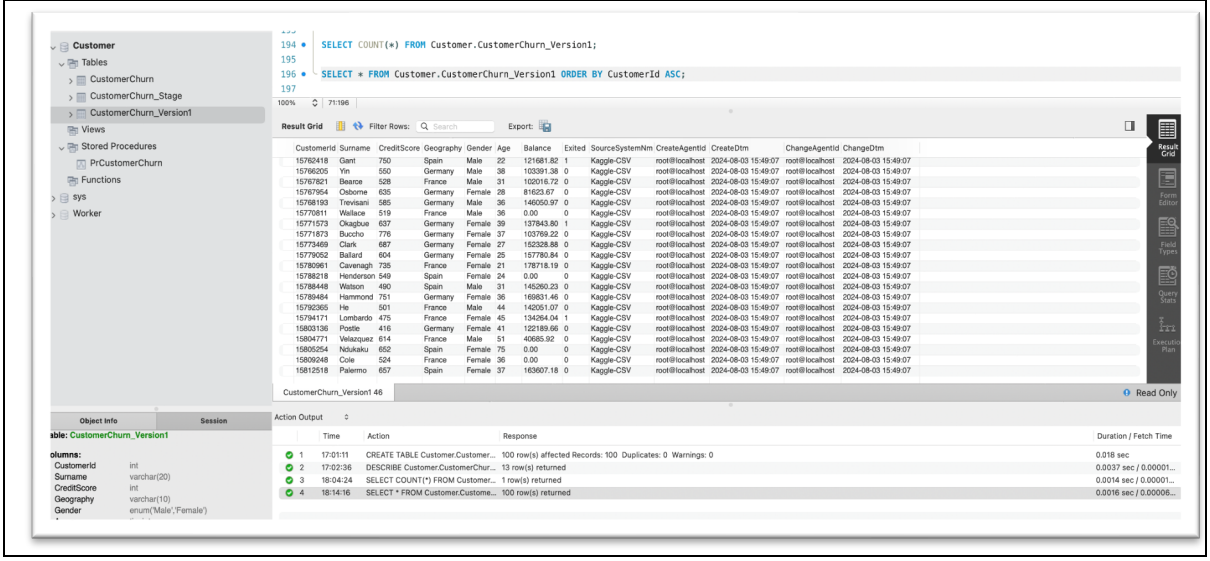


SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;



{ 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.

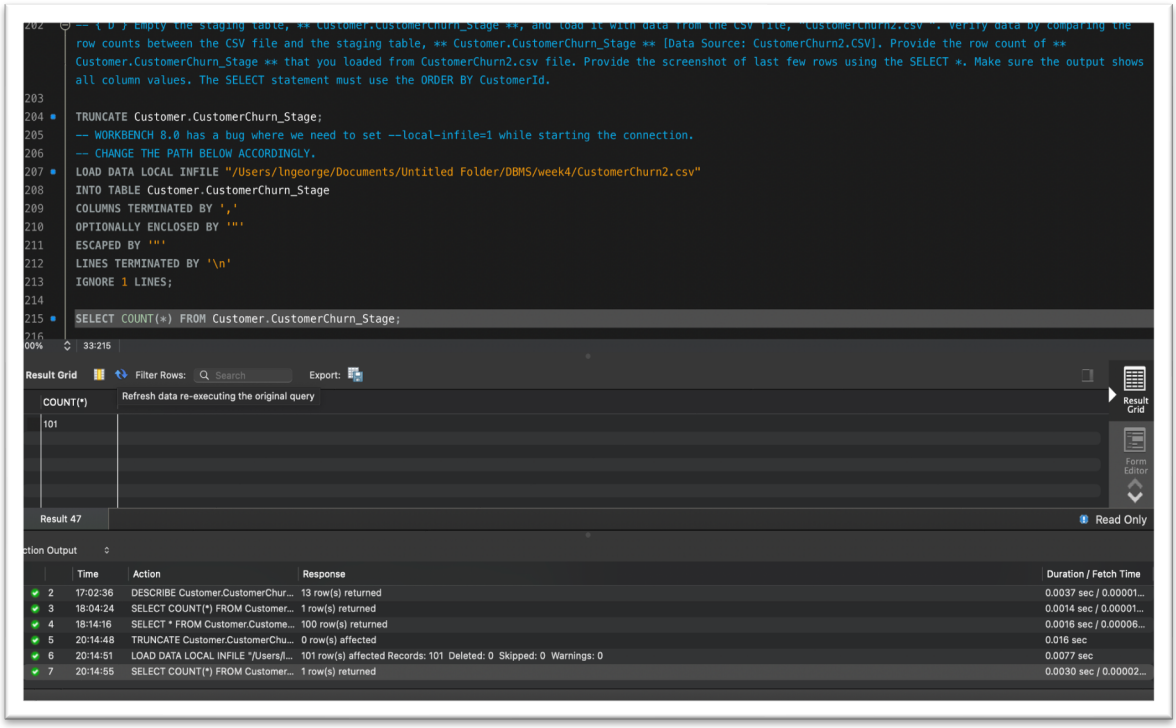
```
SELECT * FROM Customer.CustomerChurn_Version1 ORDER BY CustomerId ASC;
```



{ 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.

```
TRUNCATE Customer.CustomerChurn_Stage;
-- WORKBENCH 8.0 has a bug where we need to set --local-infile=1 while starting the connection.
-- CHANGE THE PATH BELOW ACCORDINGLY.
LOAD DATA LOCAL INFILE "/Users/lngeorge/Documents/Untitled Folder/DBMS/week4/CustomerChurn2.csv"
INTO TABLE Customer.CustomerChurn_Stage
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
```

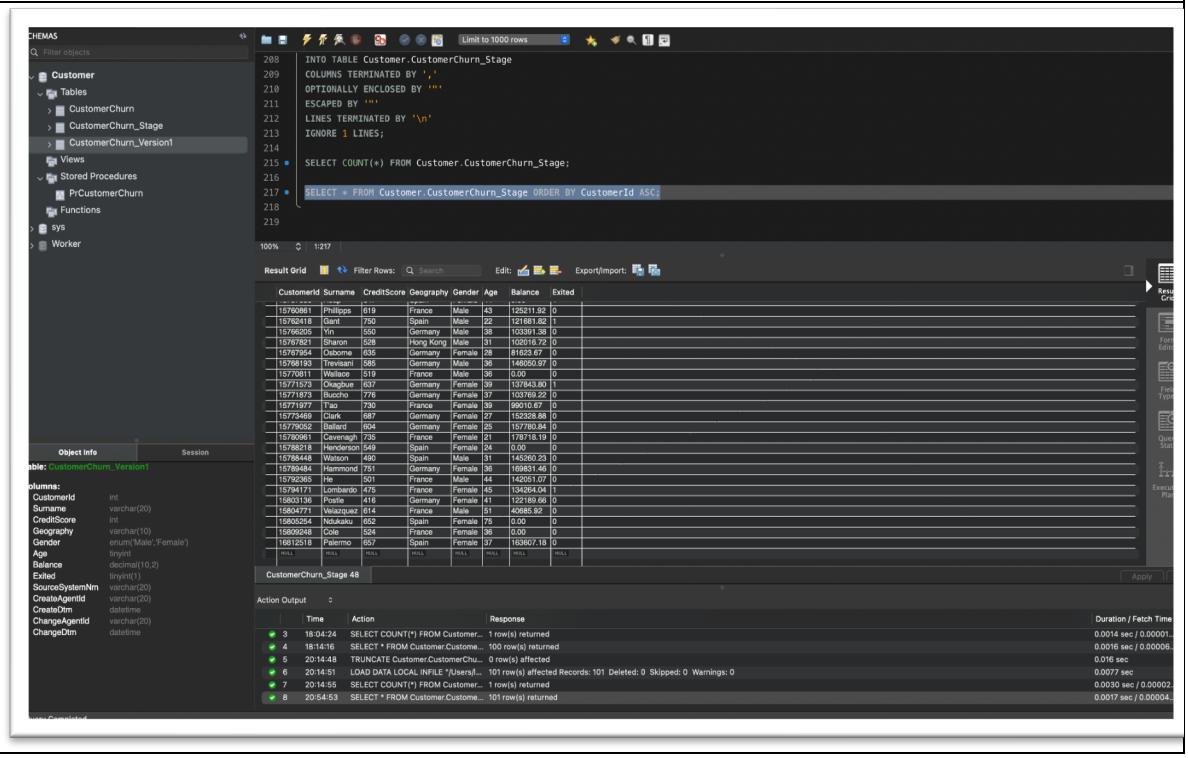
```
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

SELECT COUNT(*) FROM Customer.CustomerChurn_Stage;

SELECT * FROM Customer.CustomerChurn_Stage ORDER BY CustomerId ASC;
```



	A	B	C	D	E	F	G	H
1	CustomerId	Surname	CreditScore	Geography	Gender	Age	Balance	Exited
2	15634602	Hargrave	619	France	Female	42	0	1
3	15647311	Hill	608	Spain	Female	41	83807.86	0
4	15619304	Onio	502	France	Female	42	159660.8	1
5	15701354	Boni	699	France	Female	39	0	0
6	15737888	Mitchell	850	Spain	Female	43	125510.82	0
7	15574012	Chu	645	Spain	Male	44	113755.78	1
8	15592531	Bartlett	822	France	Male	50	0	0
9	15656148	Obinna	376	Germany	Female	29	115046.74	1
10	15792365	He	501	France	Male	44	142051.07	0
11	15592389	Vivek	684	India	Male	27	134603.88	0
12	15767821	Sharon	528	Hong Kong	Male	31	102016.72	0
13	15737173	Andrews	497	Spain	Male	30	0	1
14	15632264	Kay	476	France	Female	34	0	0
15	15691483	Chin	549	France	Female	25	12345.5	0
16	15600882	Scott	635	Spain	Female	35	0	0
17	15643966	Goforth	616	Germany	Male	45	143129.41	0
18	15737452	Romeo	653	Germany	Male	58	132602.88	1
19	15788218	Henderson	549	Spain	Female	24	0	0
20	15661507	Muldrow	587	Spain	Male	45	0	0
21	15568982	Hao	726	France	Female	24	0	0
22	15577657	McDonald	732	France	Male	41	0	0
23	15597945	Dellucci	700	Spain	Female	32	654	0
24	15699309	Gerasimov	510	Spain	Female	38	0	1
25	15625047	Yen	846	France	Female	38	0	0
26	15738191	Maclean	577	France	Male	25	0	0
27	15736816	Young	756	Germany	Male	36	136815.64	0
28	15700772	Nebechi	571	France	Male	44	0	0
29	15728693	McWilliams	574	Germany	Female	43	141349.43	0
30	15656300	Lucciano	411	France	Male	29	59697.17	0
31	15589475	Azikiwe	591	Spain	Female	39	0	1
32	15706552	Odinakachuk	533	France	Male	36	85311.7	0
33	15750181	Sanderson	553	Germany	Male	41	110112.54	0
34	15659428	Maggard	520	Spain	Female	42	0	0
35	15732963	Clements	722	Spain	Female	29	0	0
36	15794171	Lombardo	475	France	Female	45	134264.04	1
37	15788448	Watson	490	Spain	Male	31	145260.23	0
38	15729599	Lorenzo	804	Spain	Male	33	76548.6	0
39	15717426	Armstrong	850	France	Male	36	0	0
40	15585768	Cameron	582	Germany	Male	41	70349.48	0
41	15619360	Hsiao	472	Spain	Male	40	0	0
42	15738148	Clarke	465	France	Female	51	122522.32	1
43	15755196	Lavine	834	France	Female	49	131394.56	1
44	15684171	Bianchi	660	Spain	Female	61	155931.11	0
45	15754849	Tyler	776	Germany	Female	32	109421.13	0
46	15602280	Martin	829	Germany	Female	27	112045.67	1
47	15771573	Okagbue	637	Germany	Female	39	137843.8	1
48	15766205	Yin	550	Germany	Male	38	103391.38	0
49	15771873	Buccho	776	Germany	Female	37	103769.22	0
50	15616550	Chidiebele	698	Germany	Male	44	116363.37	0
51	15768193	Trevisani	585	Germany	Male	36	146050.97	0
52	15683553	Osman	788	USA	Male	22	75888.3	0
53	15702298	Parkhill	655	Germany	Male	41	125561.97	1
54	15569590	Yoo	601	Germany	Male	42	98495.72	1
55	15760861	Phillipps	619	France	Male	43	125211.92	0
56	15630053	Tsao	656	France	Male	45	127864.4	0
57	15647091	Endrizzi	725	Germany	Male	19	75888.2	0
58	15623944	T'ien	511	Spain	Female	66	0	1
59	15804771	Velazquez	614	France	Male	51	40685.92	0
60	15651280	Hunter	742	Germany	Male	35	136857	0
61	15773469	Clark	687	Germany	Female	27	152328.88	0
62	15702014	Jeffrey	555	Spain	Male	33	56084.69	0
63	15751208	Pirozzi	684	Spain	Male	56	78707.16	0
64	15592461	Jackson	603	Germany	Male	26	109166.37	0
65	15789484	Hammond	751	Germany	Female	36	169831.46	0
66	15696061	Brownless	581	Germany	Female	34	101633.04	0
67	15641582	Chibugo	735	Germany	Male	43	123180.01	0
68	15638424	Glauert	661	Germany	Female	35	150725.53	0
69	15703793	Konovalova	738	Germany	Male	58	133745.44	1
70	15620344	McKee	813	France	Male	29	0	0
71	16812518	Palermo	657	Spain	Female	37	163607.18	0
72	15779052	Ballard	604	Germany	Female	25	157780.84	0
73	15770811	Wallace	519	France	Male	36	0	0
74	15780961	Cavenagh	735	France	Female	21	178718.19	0
75	15614049	Hu	664	France	Male	55	0	0
76	15662085	Read	678	France	Female	32	0	0
77	15575185	Bushell	757	Spain	Male	33	77253.22	0
78	15803136	Postle	416	Germany	Female	41	122189.66	0
79	15706021	Buley	665	France	Female	34	96645.54	0
80	15663706	Leonard	777	France	Female	32	0	1
81	15641732	Mills	543	France	Female	36	0	0
82	15738751	Beit	493	France	Female	46	321	0
83	15805254	Ndukaku	652	Spain	Female	75	0	0
84	15762418	Gant	750	Spain	Male	22	121681.82	1
85	15625759	Rowley	729	France	Male	30	0	0
86	15622897	Sharpe	646	France	Female	46	0	1
87	15767954	Osborne	635	Germany	Female	28	81623.67	0
88	15757535	Heap	647	Spain	Female	44	0	1
89	15731511	Ritchie	808	France	Male	45	118626.55	0
90	15809248	Cole	524	France	Female	36	0	0
91	15640635	Capon	769	France	Male	29	0	0
92	15676966	Capon	730	Spain	Male	42	0	0
93	15699461	Fiorentini	515	Spain	Male	35	176273.95	0
94	15738721	Graham	773	Spain	Male	41	102827.44	0
95	15693683	Yuille	814	Germany	Male	29	97086.4	0
96	15633059	Fanucci	413	France	Male	34	0	0
97	15698932	Groves	756	Germany	Male	44	137452.09	0
98	15589975	Maclean	646	France	Female	73	97259.25	0
99	15726676	Marshall	616	Spain	Male	30	0	0
100	15771977	T'ao	730	France	Female	39	99010.67	0
101	15657566	Wieck	634	Germany	Male	24	103097.85	0
102	15727556	O'Donnell	744	Spain	Female	26	166297.89	0

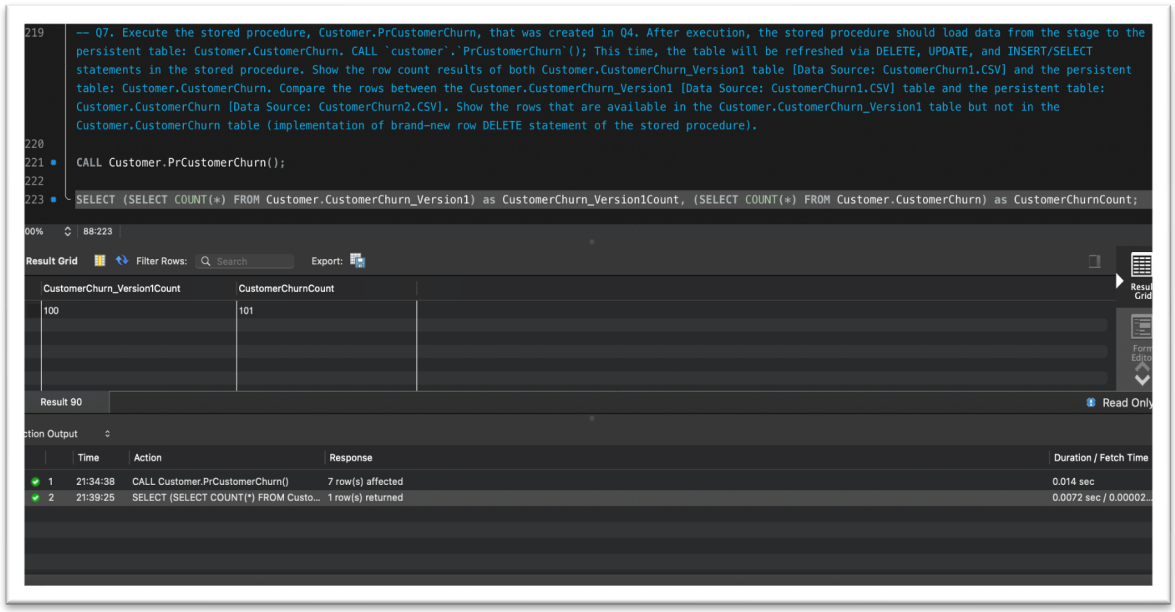
CustomerChurn2

SELECT * FROM Customer.CustomerChurn_Stage ORDER BY CustomerId ASC;



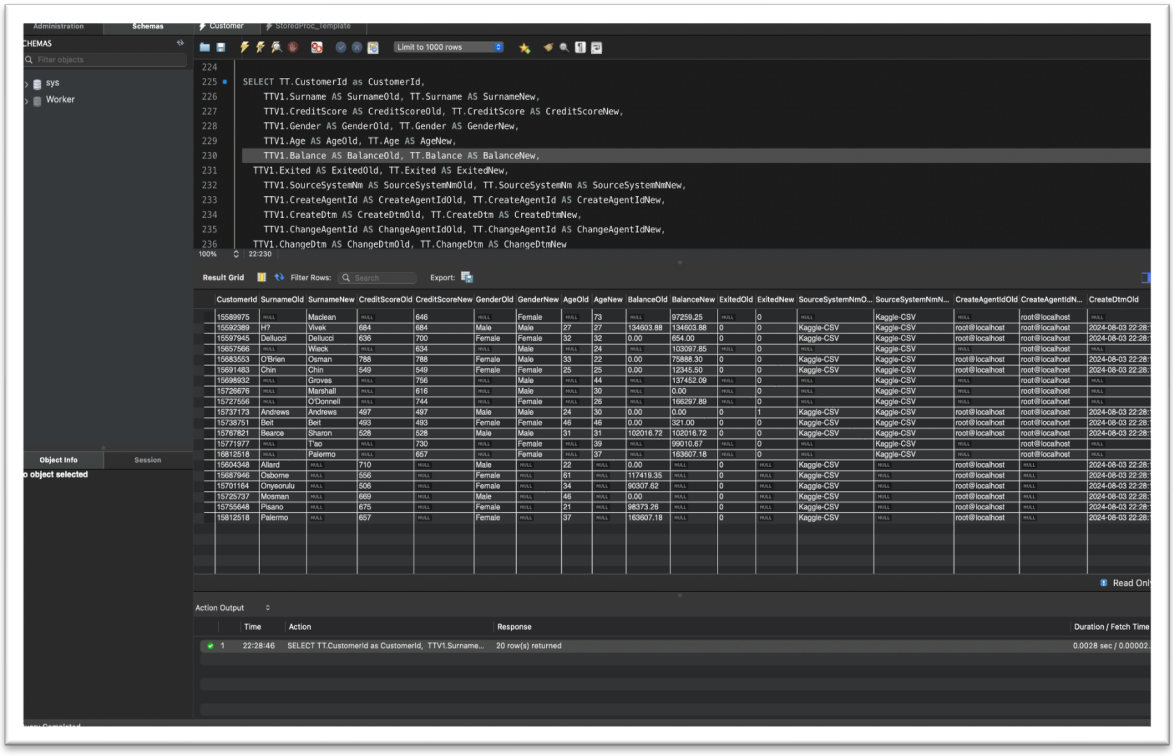
Q7. Execute the stored procedure, Customer.PrCustomerChurn, that was created in **Q4. After execution, the stored procedure should load data from the stage to the persistent table: Customer.CustomerChurn. CALL `customer`.`PrCustomerChurn`(); This time, the table will be refreshed via DELETE, UPDATE,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

```
CALL `customer`.`PrCustomerChurn`();
SELECT (SELECT COUNT(*) FROM Customer.CustomerChurn_Version1) as
CustomerChurn_Version1Count, (SELECT COUNT(*) FROM
Customer.CustomerChurn) as CustomerChurnCount;
```




```
SELECT TT.CustomerId as CustomerId,
        TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew,
        TTV1.CreditScore AS CreditScoreOld, TT.CreditScore AS CreditScoreNew,
        TTV1.Gender AS GenderOld, TT.Gender AS GenderNew,
        TTV1.Age AS AgeOld, TT.Age AS AgeNew,
        TTV1.Balance AS BalanceOld, TT.Balance AS BalanceNew,
 TTV1.Exited AS ExitedOld, TT.Exited AS ExitedNew,
        TTV1.SourceSystemNm AS SourceSystemNmOld, TT.SourceSystemNm AS
SourceSystemNmNew,
        TTV1.CreateAgentId AS CreateAgentIdOld, TT.CreateAgentId AS
CreateAgentIdNew,
        TTV1.CreateDtm AS CreateDtmOld, TT.CreateDtm AS CreateDtmNew,
        TTV1.ChangeAgentId AS ChangeAgentIdOld, TT.ChangeAgentId AS
ChangeAgentIdNew,
 TTV1.ChangeDtm AS ChangeDtmOld, TT.ChangeDtm AS ChangeDtmNew
FROM
 Customer.CustomerChurn AS TT
 LEFT OUTER JOIN Customer.CustomerChurn_Version1 AS TTV1
 ON TT.CustomerId = TTV1.CustomerId
WHERE
 (
  COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
  OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
  OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
  OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
  OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
  OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
  OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
 )
UNION
SELECT TTV1.CustomerId AS CustomerId,
```

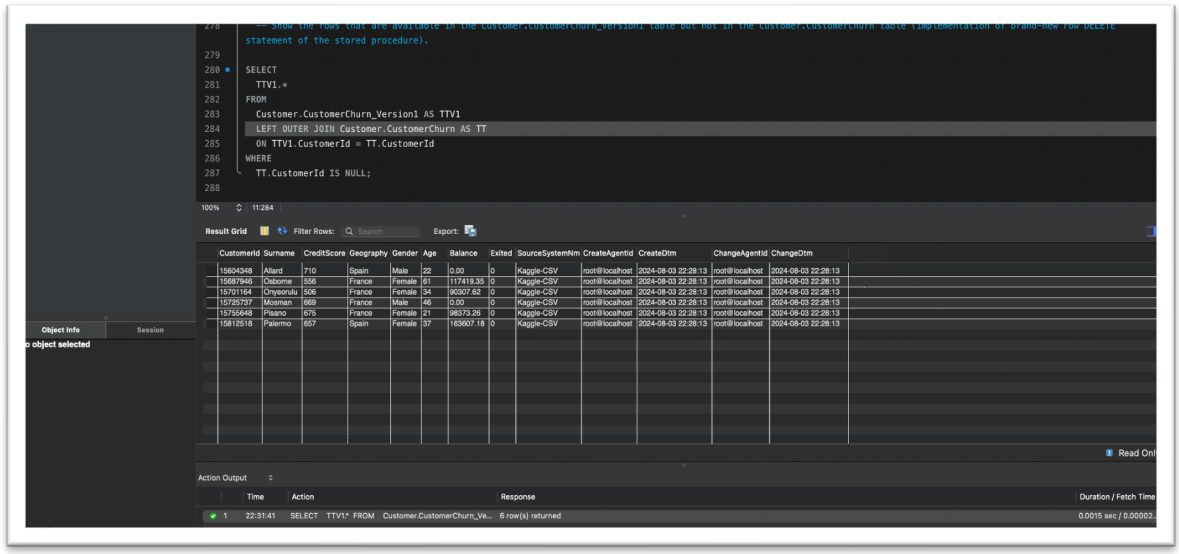
```
        TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew,
        TTV1.CreditScore AS CreditScoreOld, TT.CreditScore AS CreditScoreNew,
        TTV1.Gender AS GenderOld, TT.Gender AS GenderNew,
        TTV1.Age AS AgeOld, TT.Age AS AgeNew,
        TTV1.Balance AS BalanceOld, TT.Balance AS BalanceNew,
 TTV1.Exited AS ExitedOld, TT.Exited AS ExitedNew,
        TTV1.SourceSystemNm AS SourceSystemNmOld, TT.SourceSystemNm AS
SourceSystemNmNew,
        TTV1.CreateAgentId AS CreateAgentIdOld, TT.CreateAgentId AS
CreateAgentIdNew,
        TTV1.CreateDtm AS CreateDtmOld, TT.CreateDtm AS CreateDtmNew,
        TTV1.ChangeAgentId AS ChangeAgentIdOld, TT.ChangeAgentId AS
ChangeAgentIdNew,
 TTV1.ChangeDtm AS ChangeDtmOld, TT.ChangeDtm AS ChangeDtmNew
FROM
 Customer.CustomerChurn AS TT
 RIGHT OUTER JOIN Customer.CustomerChurn_Version1 AS TTV1
 ON TT.CustomerId = TTV1.CustomerId
WHERE
 (
  COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
  OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
  OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
  OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
  OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
  OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
  OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
 );
```



1. Here the we can see the updated columns as having both old and new values.
2. The new columns lack any old data.
3. The deleted columns lack any new data.

-- 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).

```
SELECT
  TTV1.*
FROM
  Customer.CustomerChurn_Version1 AS TTV1
  LEFT OUTER JOIN Customer.CustomerChurn AS TT
  ON TTV1.CustomerId = TT.CustomerId
WHERE
  TT.CustomerId IS NULL;
```

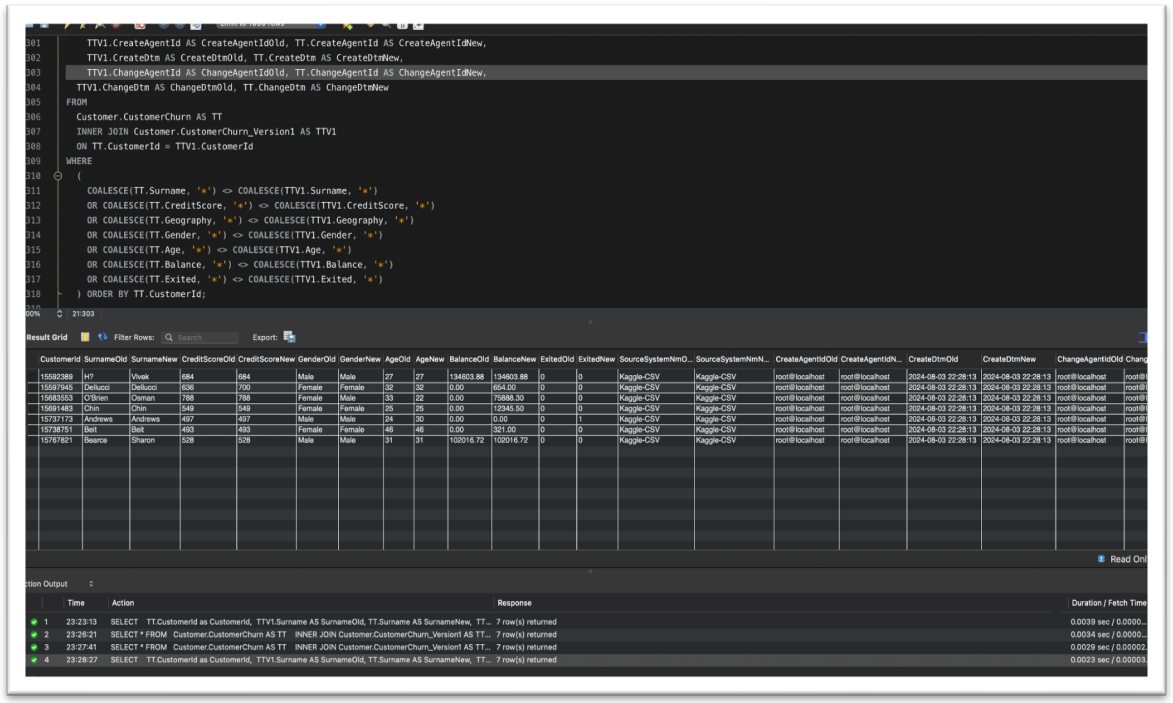


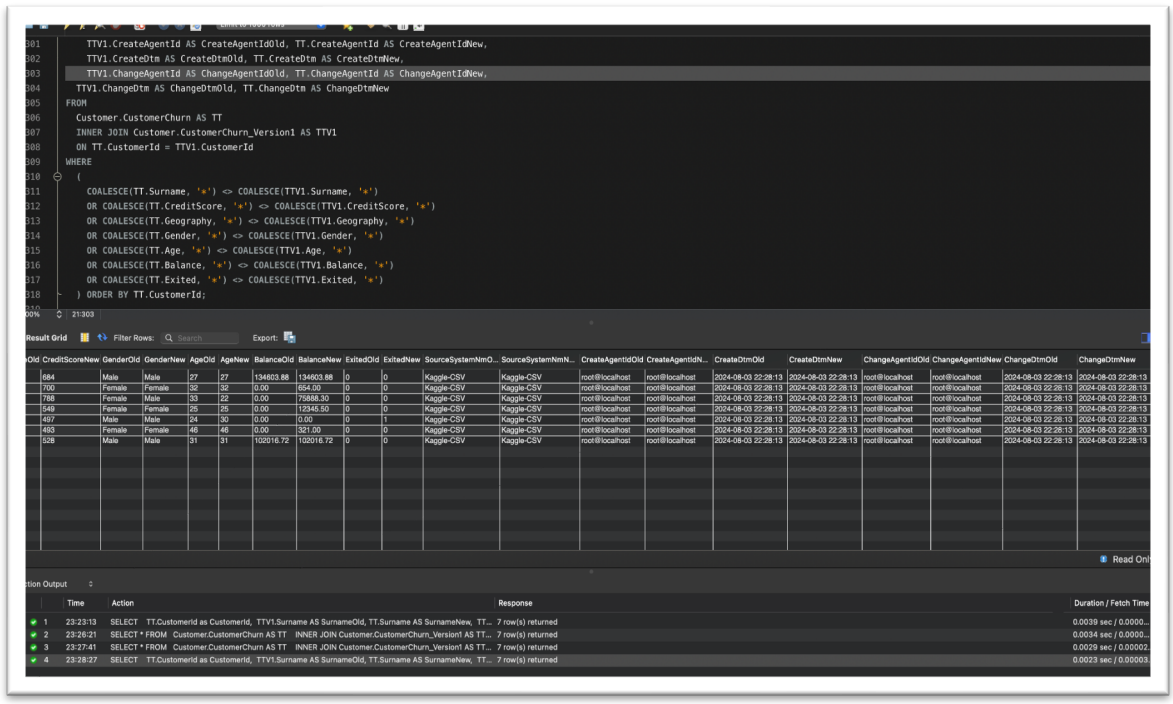
Q8. Show the rows (SELECT *) that changed (one or many non-Primary Key columns), in the Customer.CustomerChurn table (implementation of UPDATE statement of the stored procedure). You need to perform a comparison between Customer.CustomerChurn table [Data Source: CustomerChurn2.CSV] and Customer.CustomerChurn_Version1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

I have added two cases. One with select * and one with select ColumsNames

CASE 1: Using Select with column names makes it easier to compare.

```
SELECT
 TT.CustomerId as CustomerId,
        TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew,
        TTV1.CreditScore AS CreditScoreOld, TT.CreditScore AS CreditScoreNew,
        TTV1.Gender AS GenderOld, TT.Gender AS GenderNew,
        TTV1.Age AS AgeOld, TT.Age AS AgeNew,
        TTV1.Balance AS BalanceOld, TT.Balance AS BalanceNew,
 TTV1.Exited AS ExitedOld, TT.Exited AS ExitedNew,
        TTV1.SourceSystemNm AS SourceSystemNmOld, TT.SourceSystemNm AS
SourceSystemNmNew,
        TTV1.CreateAgentId AS CreateAgentIdOld, TT.CreateAgentId AS
CreateAgentIdNew,
        TTV1.CreateDtm AS CreateDtmOld, TT.CreateDtm AS CreateDtmNew,
        TTV1.ChangeAgentId AS ChangeAgentIdOld, TT.ChangeAgentId AS
ChangeAgentIdNew,
 TTV1.ChangeDtm AS ChangeDtmOld, TT.ChangeDtm AS ChangeDtmNew
FROM
 Customer.CustomerChurn AS TT
 INNER JOIN Customer.CustomerChurn_Version1 AS TTV1
 ON TT.CustomerId = TTV1.CustomerId
WHERE
 (
  COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
  OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
  OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
  OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
  OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
  OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
  OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
 ) ORDER BY TT.CustomerId;
```

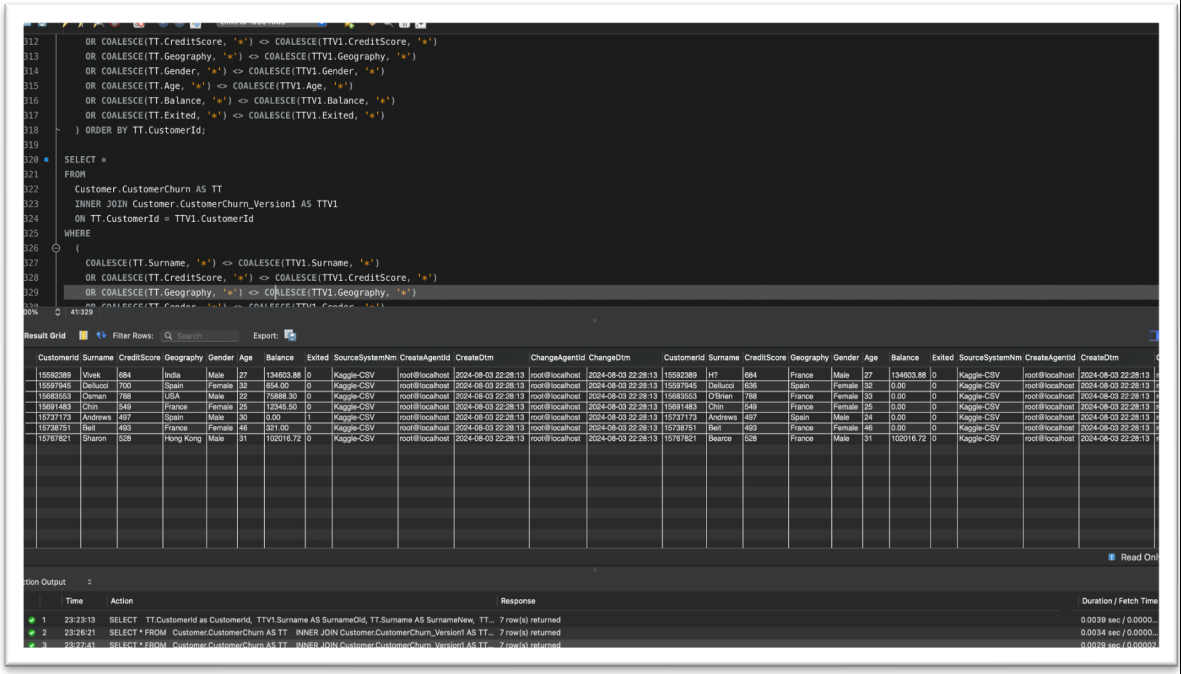


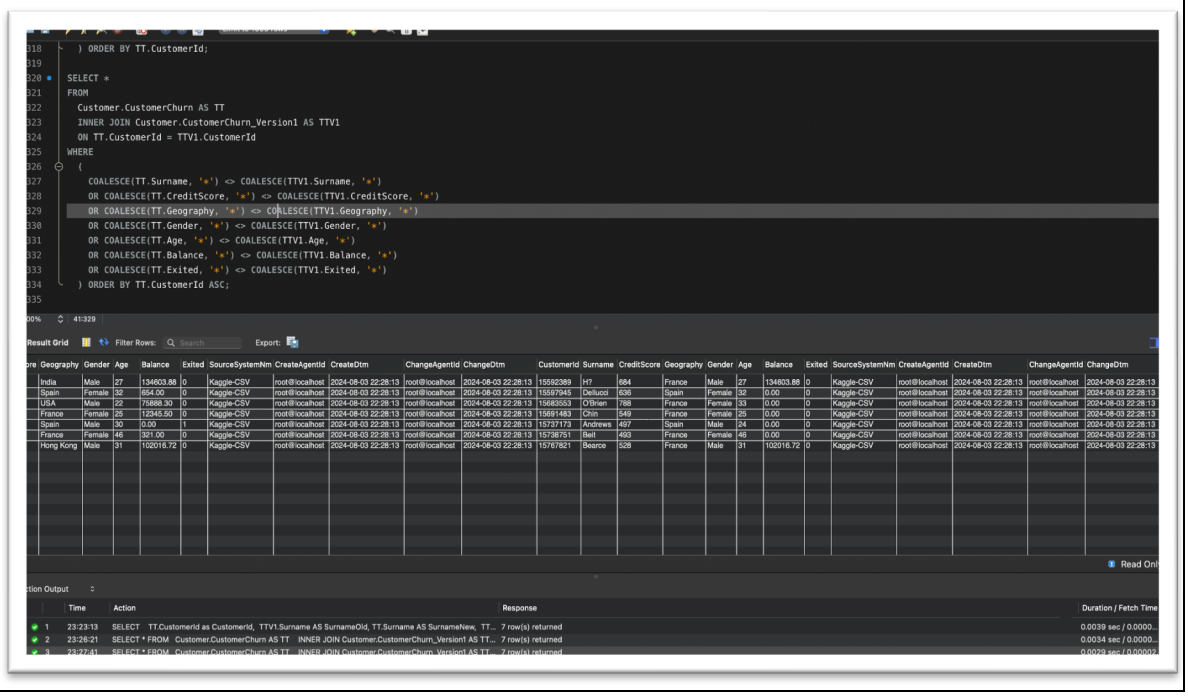


CASE 2: Using Select *.

```
SELECT *
FROM
  Customer.CustomerChurn AS TT
  INNER JOIN Customer.CustomerChurn_Version1 AS TTV1
  ON TT.CustomerId = TTV1.CustomerId
WHERE
  (
    COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
```

```
  OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
  OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
  OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
  OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
  OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
  OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
) ORDER BY TT.CustomerId ASC;
```

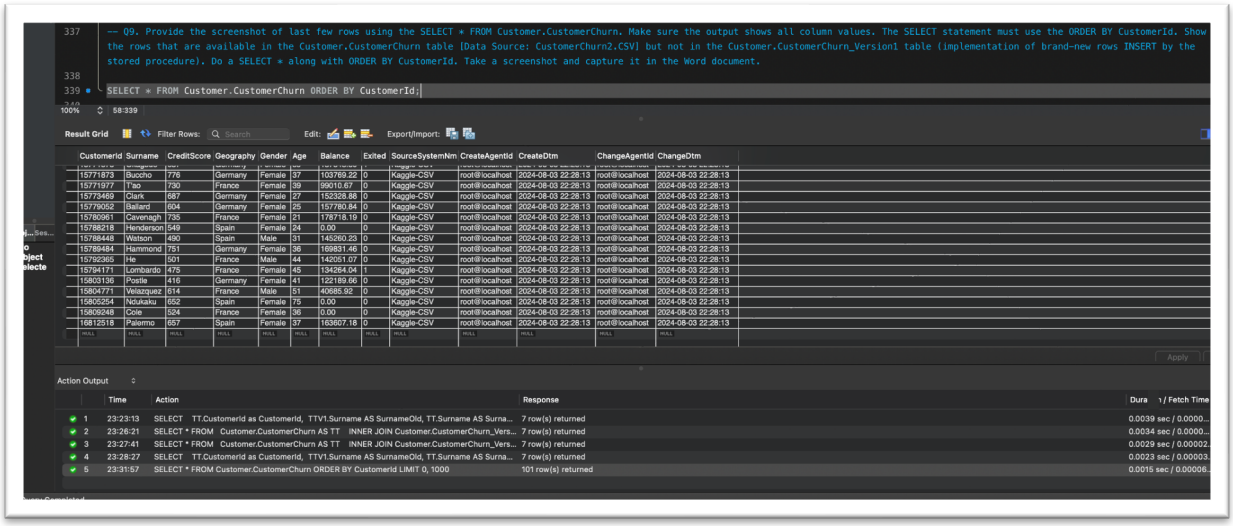




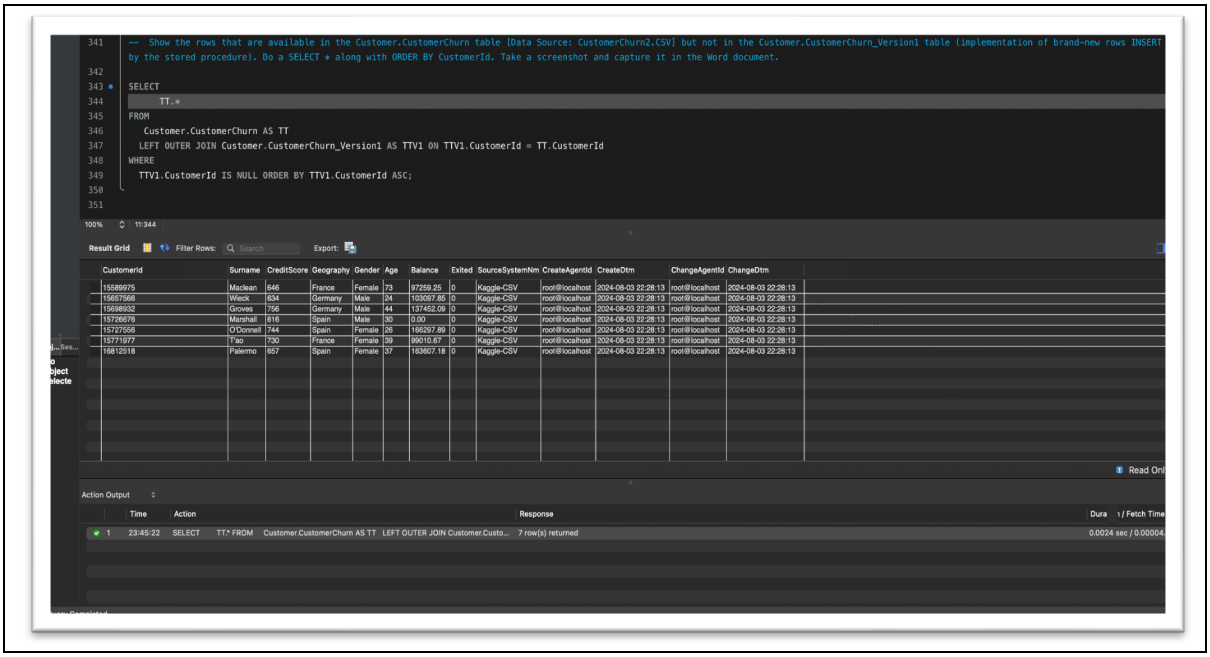
Q9. Provide the screenshot of last few rows using the SELECT * FROM Customer.CustomerChurn. Make sure the output shows all column values. The SELECT statement must use the ORDER BY CustomerId. Show the rows that are available in the Customer.CustomerChurn table [Data Source:

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.

First part asks for the screenshot of last few rows using the SELECT * FROM Customer.CustomerChurn.




```
SELECT
    TT.*
FROM
  Customer.CustomerChurn AS TT
 LEFT OUTER JOIN Customer.CustomerChurn_Version1 AS TTV1 ON
TTV1.CustomerId = TT.CustomerId
WHERE
 TTV1.CustomerId IS NULL ORDER BY TTV1.CustomerId ASC;
```



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.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS Customer.PrCustomerChurn $$
CREATE PROCEDURE Customer.PrCustomerChurn() -- Replace this with actual
database name, Customer and table name (with prefix Pr) that you use
BEGIN --
********************************************************************************
******************************

DECLARE VarCurrentTimestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
DECLARE VarSourceRowCount, VarTargetRowCount, VarThresholdNbr INTEGER
DEFAULT 0;
DECLARE VarTinyIntVal TINYINT;
--
********************************************************************************
******************************
SELECT
  COUNT(*) INTO VarSourceRowCount
FROM
  Customer.CustomerChurn_Stage;
-- Replace this with actual database name and table name (e.g.,
CustomerChurn_Stage) that you use.
SELECT
  COUNT(*) INTO VarTargetRowCount
FROM
  Customer.CustomerChurn;
```

```
-- Replace this with actual database name and table name (e.g., CustomerChurn)
that you use.
-- (TargetCount * 20%)
SELECT
  CAST(
    (VarTargetRowCount *.2) AS UNSIGNED INTEGER
  ) INTO VarThresholdNbr
FROM
  DUAL;
-- The DUMMY is system table which might vary from database to database. For your
database, you need to figure out.
-- **********************************
-- Fail the Stored Proc if the Source Row Count is less than the Threshold Number
(i.e., 20% of the Target Table row count).
-- This ensures that the Target table is not refreshed with incomplete set of Source
Data
IF VarSourceRowCount < VarThresholdNbr THEN
SELECT
  -129 INTO VarTinyIntVal
FROM
  DUAL;
END IF;
--
********************************************************************************
******************************
-- DELETE target table rows which are no longer available in source database table.
DELETE FROM
  Customer.CustomerChurn AS TrgtTbl
WHERE
  EXISTS (
    SELECT
      *
    FROM
      (
        SELECT
          TT.CustomerID -- Primary Key Column(s)
        FROM
          Customer.CustomerChurn AS TT -- Example table name: CustomerChurn
          LEFT OUTER JOIN Customer.CustomerChurn_Stage AS ST -- Example table
name: CustomerChurn_Stage
          ON TT.CustomerId = ST.CustomerId
        WHERE
          ST.CustomerId IS NULL
      ) AS SrcTbl
    WHERE
      TrgtTbl.CustomerId = SrcTbl.CustomerId
  );
```

```
-- **************************UPDATE ROWS THAT CHANGED IN SOURCE*****************************************
-- Update the rows for which new version of rows have arrived as part of delta/incremental feed (i.e., change to non-key values).
UPDATE
  Customer.CustomerChurn AS TrgtTbl
  INNER JOIN Customer.CustomerChurn_Stage AS SrcTbl
  ON TrgtTbl.CustomerId = SrcTbl.CustomerId
SET
  TrgtTbl.Surname = SrcTbl.Surname,
  TrgtTbl.CreditScore = SrcTbl.CreditScore ,
  TrgtTbl.Geography = SrcTbl.Geography ,
  TrgtTbl.Gender = SrcTbl.Gender ,
  TrgtTbl.Age = SrcTbl.Age ,
  TrgtTbl.Balance = SrcTbl.Balance ,
  TrgtTbl.Exited = SrcTbl.Exited ,
  TrgtTbl.ChangeDtm = VarCurrentTimestamp
WHERE
  (
    COALESCE(TrgtTbl.Surname, '*') <> COALESCE(SrcTbl.Surname, '*')
    OR COALESCE(TrgtTbl.CreditScore, '*') <> COALESCE(SrcTbl.CreditScore, '*')
    OR COALESCE(TrgtTbl.Geography, '*') <> COALESCE(SrcTbl.Geography, '*')
    OR COALESCE(TrgtTbl.Gender, '*') <> COALESCE(SrcTbl.Gender, '*')
    OR COALESCE(TrgtTbl.Age, '*') <> COALESCE(SrcTbl.Age, '*')
    OR COALESCE(TrgtTbl.Balance, '*') <> COALESCE(SrcTbl.Balance, '*')
    OR COALESCE(TrgtTbl.Exited, '*') <> COALESCE(SrcTbl.Exited, '*')
  );
-- ****************************INSERT BRAND NEW ROWS INTO TARGET**************************************
-- Identify brand new rows in source table and load into target table.
INSERT INTO Customer.CustomerChurn (
  CustomerId, Surname, CreditScore,
  Geography, Gender, Age, Balance, Exited,
  SourceSystemNm, CreateAgentId, CreateDtm,
  ChangeAgentId, ChangeDtm
)
SELECT
  SrcTbl.CustomerId,
  SrcTbl.Surname,
  SrcTbl.CreditScore,
  SrcTbl.Geography,
  SrcTbl.Gender,
  SrcTbl.Age,
  SrcTbl.Balance,
  SrcTbl.Exited,
  'Kaggle-CSV' AS SourceSystemNm,
  current_user() AS CreateAgentId,
```

```
  VarCurrentTimestamp AS CreateDtm,
  current_user() AS ChangeAgentId,
  VarCurrentTimestamp AS ChangeDtm
FROM
  Customer.CustomerChurn_Stage AS SrcTbl
  INNER JOIN (
    SELECT
      ST.CustomerId
    FROM
      Customer.CustomerChurn_Stage AS ST
      LEFT OUTER JOIN Customer.CustomerChurn AS TT ON ST.CustomerId =
TT.CustomerId
    WHERE
      TT.CustomerId IS NULL
  ) AS ChgdNew ON SrcTbl.CustomerId = ChgdNew.CustomerId;
--
********************************************************************************
*****************************
END$$
DELIMITER ;
```