

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.

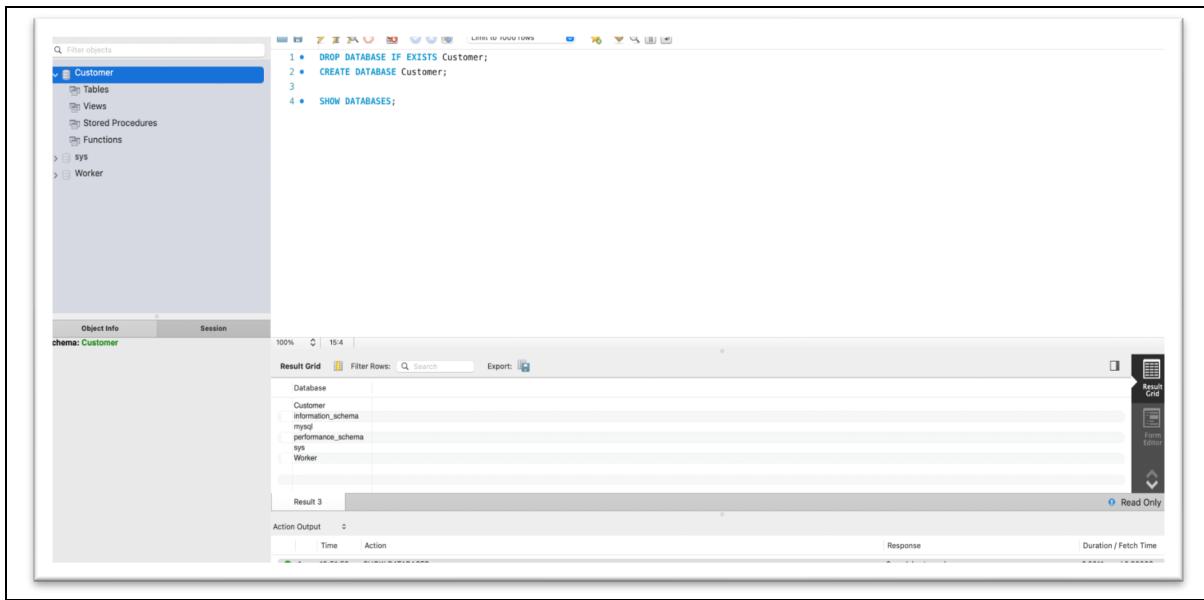
{ A } Using the MySQL Workbench, create a database called Customer. The database must be named “Customer”.

```
CREATE DATABASE Customer;
```

The screenshot shows the MySQL Workbench interface. In the left sidebar, there is a tree view labeled 'Filter objects' with nodes for 'Customer', 'sys', and 'Worker'. The main pane displays two SQL statements: 'DROP DATABASE IF EXISTS Customer;' and 'CREATE DATABASE Customer;'. Below the statements, the 'Action Output' section shows a single row of data: a yellow warning icon, the number '1', the time '18:49:41', the action 'DROP DATABASE IF EXISTS Customer', and the response '0 row(s) affected, 1 warning(s): 1008 Can't drop data... Duration / Fetch Time 0.00026 sec'. The status bar at the bottom indicates '100%' completion and '1/1' rows.

{ 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;
```

```

8 • USE Customer;
9
10 • CREATE TABLE Customer.CustomerChurn_Stage (
11     CustomerId INTEGER NOT NULL,
12     Surname VARCHAR(20) NOT NULL,
13     CreditScore INTEGER NOT NULL,
14     Geography VARCHAR(10) NOT NULL,
15     Gender VARCHAR(10) NOT NULL,
16     Age TINYINT NOT NULL,
17     Balance DECIMAL(10, 2) NOT NULL,
18     Exited BOOLEAN NOT NULL, -- Same as TINYINT(1)
19     PRIMARY KEY(CustomerId)
20 );
21
22 • DESCRIBE Customer.CustomerChurn_Stage;
23

```

100% 19:22

Object Info	Session
table: CustomerChurn_Stage	
columns:	
Customerid	int PK
Surname	varchar(20)
CreditScore	int
Geography	varchar(10)
Gender	varchar(10)
Age	tinyint
Balance	decimal(10,2)
Exited	tinyint(1)

Result 14

Action Output

Time	Action	Response	Duration / Fetch Time
1 19:32:52	CREATE TABLE Customer.CustomerChurn (CustomerId INTEGER NOT NULL, Surname VAR...	0 row(s) affected, 4 warning(s): 1681 Integer display width is deprecated and will be removed in a future release..	0.021 sec
2 19:35:12	DROP DATABASE IF EXISTS Customer	2 row(s) affected	0.041 sec
3 19:35:17	CREATE DATABASE Customer	1 row(s) affected	0.0038 sec
4 19:35:19	USE Customer	0 row(s) affected	0.00092 sec
5 19:35:27	CREATE TABLE Customer.CustomerChurn_Stage (CustomerId INTEGER NOT NULL, Surna...	0 row(s) affected	0.014 sec
6 19:35:33	DESCRIBE Customer.CustomerChurn_Stage	8 row(s) returned	0.0036 sec / 0.0000...

{ 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;
```

The screenshot shows the Object Explorer on the left with 'Customer' selected, and the 'Tables' node expanded to show 'CustomerChurn'. The 'CustomerChurn' table is selected. The middle pane displays the T-SQL code for creating the table:

```

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)
);

```

The 'DESCRIBE Customer.CustomerChurn;' command is also visible. The bottom pane shows the results of the 'CREATE TABLE' command, which was successful.

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;

```

The screenshot shows the Object Explorer on the left with 'Customer' selected, and the 'Tables' node expanded to show 'CustomerChurn' and 'CustomerChurn_Stage'. The 'CustomerChurn_Stage' table is selected. The middle pane contains the T-SQL command:

```

-- 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' .
-- WORKBENCH 8.0 has a bug where we need to set --local-infile while starting the connection.
-- CHANGE THE PATH BELOW ACCORDINGLY.
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;

```

The bottom pane shows the execution log for the command.

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)

The screenshot shows a database interface with the following details:

- SQL Query:** SELECT COUNT(*) FROM Customer.CustomerChurn_Stage
- Result Grid:** COUNT(*) | 100
- Action Output:**

Time	Action	Response	Duration / Fetch Time
22:54:14	LOAD DATA LOCAL INFILE "&Users\InGeorge\Documents\Untitled Folder\DBMS\week4\CustomerChurn1.csv" INTO TABLE Customer.CustomerChurn_Stage LIMIT 0,1000	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 1 row(s) returned	0.011 sec
22:59:14	SELECT COUNT(*) FROM Customer.CustomerChurn_Stage LIMIT 0,1000	1 row(s) returned	0.0017 sec / 0.00002...

	A	B	C	D	E	F	G	H
1	Customer	Surn	CreditScl	Geogra	Cent	Age	Bala	Exits
2	15634602	Hargrave	619	France	Female	42	0	1
3	15647311	Hill	608	Spain	Female	41	83807.9	0
4	15619304	Ohio	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	Gerashimov	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	Luciano	411	France	Male	29	596972	0
32	15589475	Azikiwe	591	Spain	Female	39	0	1
33	15706552	Ondakachukwu	533	France	Male	36	85311.7	0
34	15750181	Sander son	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	Buchcho	776	Germany	Female	37	103769	0
52	15616550	Chikilebele	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	Yo	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	758882	0
60	15623944	Tien	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	Chibango	735	Germany	Male	43	123180	0
70	15638424	Glaert	661	Germany	Female	35	150726	0
71	15755648	Pisan	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	Nduakaku	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	Florentini	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;

57 • `SELECT * FROM Customer.CustomerChurn_Stg ORDER BY CustomerId ASC;`

00% ◁ 68:57 |

Result Grid Filter Rows: Search: Edit: Export/Import:

Customer **Surname** **CreditScore** **Geography** **Gender** **Age** **Balance** **Exited**

15662065	Peterson	679	France	Female	30	0.00	0
15663708	Concord	777	France	Female	35	0.00	1
15675963	Capon	730	Spain	Male	42	0.00	0
15683553	O'Brien	788	France	Female	33	0.00	0
15684171	Bianchi	660	Spain	Female	61	155931.11	0
15687946	Osborne	556	France	Female	61	117419.35	0
15691483	Chin	549	France	Female	25	0.00	0
15693683	Yulie	814	Germany	Male	29	97086.40	0
15696801	Brownless	581	Germany	Female	34	101633.04	0
15699309	Gerasimov	510	Spain	Female	38	0.00	1
15699461	Fiorentini	515	Spain	Male	35	176273.95	0
15700774	Nebechi	571	France	Male	44	0.00	0
15701164	Onyeagwu	506	France	Female	34	90307.62	0
15701354	Boni	699	France	Female	39	0.00	0
15702014	Jeffrey	555	Spain	Male	33	56048.69	0
15702289	Parkhill	655	Germany	Male	41	125961.97	1
15703793	Konovalov	738	Germany	Male	58	133745.44	1
15706280	Buley	665	France	Female	34	96645.54	0
15706552	Malinika	533	France	Male	36	9311.70	0
15714236	Armstrong	850	France	Male	36	0.00	0
15725737	Mosman	669	France	Male	46	0.00	0
15726693	McWillis	574	Germany	Female	43	141349.43	0
15729599	Lorenzo	804	Spain	Male	33	76548.60	0
15731511	Ritchie	808	France	Male	45	118626.55	0
15732953	Clements	722	Spain	Female	29	0.00	0
15736816	Young	756	Germany	Male	36	136815.64	0
15737173	Andrews	497	Spain	Male	24	0.00	0
15737452	Romes	653	Germany	Male	58	132602.88	1
15737888	Mitchell	850	Spain	Female	43	125510.82	0
15738148	Clarke	465	France	Female	51	122522.32	1
15738191	Maclean	577	France	Male	25	0.00	0
15738721	Graham	773	Spain	Male	41	102827.44	0
15738751	Belt	493	France	Female	46	0.00	0
15750181	Sanderson	553	Germany	Male	41	110112.54	0
15751208	Pirozzi	684	Spain	Male	56	78707.16	0
15754849	Tyler	776	Germany	Female	32	109421.13	0
15755196	Leavine	834	France	Female	49	131394.56	1
15755640	Pisano	675	France	Female	21	96373.26	0

CustomerChurn_Stg 8

Action Output

Time	Action	Response	Duration / Fetch Time
1 23:15:57	SELECT * FROM Customer.CustomerChurn_Stg ORDER BY CustomerId ASC LIMIT 0, 1000	100 row(s) returned	0.0012 sec / 0.00004...

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).]]

The screenshot shows the SSMS interface with the following details:

- Object Explorer (Left):** Shows the database structure under the "Customer" schema, including tables (CustomerChurn, CustomerChurn_Stage), views, and stored procedures (PrCustomerChurn).
- Query Editor (Center):** Displays the script for the stored procedure `PrCustomerChurn`. The script includes comments explaining its purpose and how it interacts with the `CustomerChurn` and `CustomerChurn_Stage` tables.
- Status Bar (Bottom):** Shows performance metrics: 100% CPU usage at 8.55%, and a single action output entry: "Time Action" (1 12:33:15) followed by the command "DROP PROCEDURE IF EXISTS Customer.PrCustomerChurn".

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();
```

```

Administration Schemas Customer StoredProc_Template
schemas
Customer
Tables
CustomerChurn
CustomerChurn_Stage
Views
Stored Procedures
PrCustomerChurn
Functions
sys
Worker

167    INNER JOIN (
168        SELECT
169            ST.CustomerId
170        FROM
171            Customer.CustomerChurn_Stage AS ST
172        LEFT OUTER JOIN Customer.CustomerChurn AS TT ON ST.CustomerId = TT.CustomerId
173        WHERE
174            TT.CustomerId IS NULL
175        ) AS ChgdNew ON SrcTbl.CustomerId = ChgdNew.CustomerId;
176        --
177    END$$
178    DELIMITER ;
179
180 • SET SQL_SAFE_UPDATES = 0;
181 • CALL Customer.PrCustomerChurn();
182
183
184
100% 1:183
Action Output
Time Action Response Duration / Fetch Time
15:49:07 CALL 'Customer'.PrCustomerChu... 100 row(s) affected 0.012 sec

```

{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`.

```

Administration Schemas Customer StoredProc_Template
schemas
Customer
Tables
CustomerChurn
CustomerChurn_Stage
Views
Stored Procedures
PrCustomerChurn
Functions
sys
Worker

173
174 WHERE
175     TT.CustomerId IS NULL
176     ) AS ChgdNew ON SrcTbl.CustomerId = ChgdNew.CustomerId;
177     --
178    END$$
179    DELIMITER ;
180
181 • SET SQL_SAFE_UPDATES = 0;
182
183 • SELECT (SELECT COUNT(*) FROM Customer.CustomerChurn_Stage) as CustomerChurn_StageCount, (SELECT COUNT(*) FROM Customer.CustomerChurn) as CustomerChurnCount
100% 2:183
Result Grid
CustomerChurn_StageCount CustomerChurnCount
100 100
Result 39
Read Only
Action Output
Time Action Response Duration / Fetch Time
15:53:51 SELECT (SELECT COUNT(*) FROM... 1row(s) returned 0.0024 sec / 0.00002...

```

{ 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`.

185 • `SELECT * FROM Customer.CustomerChurn_Staged ORDER BY CustomerId;`

CustomerChurn_Staged 40

Action Output

Object Info	Session	Time	Action	Response	Duration / Fetch Time
o object selected		15:53:51	SELECT (SELECT COUNT(*)) FROM...	1 row(s) returned	0.0024 sec / 0.00002...
		16:09:34	SELECT * FROM Customer.Customer...	100 row(s) returned	0.0027 sec / 0.00006...

187 • `SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId;`

CustomerChurn 41

Action Output

Object Info	Session	Time	Action	Response	Duration / Fetch Time
o object selected		15:53:51	SELECT (SELECT COUNT(*)) FROM...	1 row(s) returned	0.0024 sec / 0.00002...
		16:09:34	SELECT * FROM Customer.Customer...	100 row(s) returned	0.0027 sec / 0.00006...
		16:10:58	SELECT * FROM Customer.Customer...	100 row(s) returned	0.0018 sec / 0.00007...

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_Staged via the stored procedure).

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

The screenshot shows the Object Explorer on the left with the path: Customer > Tables > CustomerChurn_Version1 selected. The script pane on the right contains the T-SQL code for creating the table:

```

175 ) AS ChgdNew ON SrcTbl.CustomerId = ChgdNew.CustomerId;
176
177 -- *****
178 END$ DELIMITER ;
179
180 • SET SQL_SAFE_UPDATES = 0;
181 • CALL Customer.PrCustomerChurn();
182
183 • SELECT (SELECT COUNT(*) FROM Customer.CustomerChurn_Stag... as CustomerChurn_StagCount, (SELECT COUNT(*) FROM Customer.CustomerChurn) as CustomerChurnCount;
184
185 • SELECT * FROM Customer.CustomerChurn_Stag ORDER BY CustomerId;
186
187 • SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId;
188
189 • CREATE TABLE Customer.CustomerChurn_Version1
190     AS (SELECT * FROM Customer.CustomerChurn);
191

```

The Action Output pane at the bottom shows the execution results:

Action	Time	Action	Response	Duration / Fetch Time
CREATE TABLE	17:01:11	Customer.CustomerChurn_Version1	100 row(s) affected Records: 100 Duplicates: 0 Warnings: 0	0.018 sec

The Object Info pane on the left displays the table definition:

```

table: CustomerChurn_Version1
columns:
Customerid int
Surname varchar(20)
CreditScore int
Geography varchar(10)
Gender enum('Male','Female')
Age tinyint
Balance decimal(10,2)
Extd tinyint(1)
SourceSystemNm varchar(20)
CreateAgentId varchar(20)
CreateDt datetime
ChangeAgentId varchar(20)
ChangeDt datetime

```

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

DESCRIBE Customer.CustomerChurn_Version1;

The screenshot shows the Object Explorer on the left with the path: Customer > Tables > CustomerChurn_Version1 selected. The script pane on the right contains the T-SQL code for creating the table and then executing the DESCRIBE command:

```

177 END$ DELIMITER ;
178
179
180 • SET SQL_SAFE_UPDATES = 0;
181 • CALL Customer.PrCustomerChurn();
182
183 • SELECT (SELECT COUNT(*) FROM Customer.CustomerChurn_Stag... as CustomerChurn_StagCount, (SELECT COUNT(*) FROM Customer.CustomerChurn) as CustomerChurnCount;
184
185 • SELECT * FROM Customer.CustomerChurn_Stag ORDER BY CustomerId;
186
187 • SELECT * FROM Customer.CustomerChurn ORDER BY CustomerId;
188
189 • CREATE TABLE Customer.CustomerChurn_Version1
190     AS (SELECT * FROM Customer.CustomerChurn);
191
192 • DESCRIBE Customer.CustomerChurn_Version1;
193
194

```

The Result Grid pane on the right shows the table structure:

Field	Type	Null	Key	Default	Extra
Customerid	int	NO			
Surname	varchar(20)	NO			
CreditScore	int	NO			
Geography	varchar(10)	NO			
Gender	enum('Male','Female')	NO			
Age	tinyint	NO			
Balance	decimal(10,2)	NO			
Extd	tinyint(1)	NO			
SourceSystemNm	varchar(20)	NO			
CreateAgentId	varchar(20)	NO			
CreateDt	datetime	NO			
ChangeAgentId	varchar(20)	NO			
ChangeDt	datetime	NO			

The Action Output pane at the bottom shows the execution results:

Action	Time	Action	Response	Duration / Fetch Time
CREATE TABLE	17:01:11	Customer.CustomerChurn_Version1	100 row(s) affected Records: 100 Duplicates: 0 Warnings: 0	0.018 sec
DESCRIBE	17:02:36	Customer.CustomerChurn_Version1	13 row(s) returned	0.0037 sec / 0.00001...

SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;

The screenshot shows the MySQL Workbench interface. On the left, the 'Customer' schema is expanded, showing tables like CustomerChurn, CustomerChurn_Stage, and CustomerChurn_Version1. A query editor window is open with the following SQL code:

```

193 •
194 • SELECT COUNT(*) FROM Customer.CustomerChurn_Version1;
195
196

```

The result grid shows a single row with the value '100'. Below the grid, the 'Action Output' section displays the execution history:

- 1 17:01:11 CREATE TABLE Customer.Customer... 100 rows(s) affected Records: 100 Duplicates: 0 Warnings: 0
- 2 17:02:36 DESCRIBE Customer.CustomerChur... 13 row(s) returned
- 3 18:04:24 SELECT COUNT(*) FROM Customer... 1 row(s) returned

Details at the bottom indicate a duration of 0.018 sec and a fetch time of 0.0037 sec / 0.00001...

{ 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;

The screenshot shows the MySQL Workbench interface with the same schema navigation on the left. The query editor contains the previous SELECT statement. The result grid displays numerous rows of customer data, ordered by CustomerId. The 'Action Output' section shows the execution history again:

- 1 17:01:11 CREATE TABLE Customer.Customer... 100 rows(s) affected Records: 100 Duplicates: 0 Warnings: 0
- 2 17:02:36 DESCRIBE Customer.CustomerChur... 13 row(s) returned
- 3 18:04:24 SELECT COUNT(*) FROM Customer... 1 row(s) returned
- 4 18:14:16 SELECT * FROM Customer.Customer... 100 rows(s) returned

Details at the bottom show a duration of 0.018 sec and a fetch time of 0.0037 sec / 0.00001...

{ 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/lngorge/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;
```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
202 -- (0) 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.
203
204 • TRUNCATE Customer.CustomerChurn_Stage;
-- WORKBENCH 8.0 has a bug where we need to set --local-infile=1 while starting the connection.
205
206 • CHANGE THE PATH BELOW ACCORDINGLY.
207 • LOAD DATA LOCAL INFILE "/Users/lngorge/Documents/Untitled Folder/DBMS/week4/CustomerChurn2.csv"
208 INTO TABLE Customer.CustomerChurn_Stage
209   COLUMNS TERMINATED BY ','
210   OPTIONALLY ENCLOSED BY ""
211   ESCAPED BY ''
212   LINES TERMINATED BY '\n'
213   IGNORE 1 LINES;
214
215 • SELECT COUNT(*) FROM Customer.CustomerChurn_Stage;
216
```

Result Grid:

COUNT(*)
101

Action Output:

Time	Action	Response	Duration / Fetch Time
2 17:02:36	DESCRIBE Customer.CustomerChur...	13 row(s) returned	0.0037 sec / 0.00001...
3 18:04:24	SELECT COUNT(*) FROM Customer...	1 row(s) returned	0.0014 sec / 0.00001...
4 18:14:16	SELECT * FROM Customer.Customer...	100 row(s) returned	0.0016 sec / 0.00006...
5 20:14:48	TRUNCATE Customer.CustomerChu...	0 row(s) affected	0.016 sec
6 20:14:51	LOAD DATA LOCAL INFILE "/Users/l...	101 row(s) affected Records: 101 Deleted: 0 Skipped: 0 Warnings: 0	0.0077 sec
7 20:14:55	SELECT COUNT(*) FROM Customer...	1 row(s) returned	0.0030 sec / 0.00002...

A	B	C	D	E	F	G	H
CustomerID	Surname	CreditScore	Geography	Gender	Age	Balance	Exited
1	15634602 Hargrave	619	France	Female	42	0	1
2	15647311 Hill	608	Spain	Female	41	83807.86	0
3	15619304 Onio	502	France	Female	42	159660.8	1
4	15701354 Boni	699	France	Female	39	0	0
5	15737888 Mitchell	850	Spain	Female	43	125510.82	0
6	15574012 Chu	645	Spain	Male	44	113755.78	1
7	15592531 Bartlett	822	France	Male	50	0	0
8	15656148 Obinna	376	Germany	Female	29	115046.74	1
9	15792365 He	501	France	Male	44	142051.07	0
10	15592389 Wnek	684	India	Male	27	134603.88	0
11	15767821 Sharon	528	Hong Kong	Male	31	102018.72	0
12	15737173 Andrews	497	Spain	Male	30	0	1
13	15632264 Kay	476	France	Female	34	0	0
14	15691483 Chin	549	France	Female	25	12345.5	0
15	15600888 Scott	635	Spain	Female	35	0	0
16	15643965 Goforth	616	Germany	Male	45	143129.41	0
17	15737452 Romeo	653	Germany	Male	58	132602.88	1
18	15788218 Henderson	549	Spain	Female	24	0	0
19	15661507 Muldowney	587	Spain	Male	45	0	0
20	15568982 Ha	726	France	Female	24	0	0
21	15577657 McDonald	732	France	Male	41	0	0
22	15597945 Deluccci	700	Spain	Female	32	654	0
23	15699309 Gerasimov	510	Spain	Female	38	0	1
24	15625047 Yen	846	France	Female	38	0	0
25	15738191 Maclean	577	France	Male	25	0	0
26	15736816 Young	756	Germany	Male	36	136815.64	0
27	15700772 Nebechi	571	France	Male	44	0	0
28	15728699 McWilliams	574	Germany	Female	43	141349.43	0
29	15656300 Lucciano	411	France	Male	29	59697.17	0
30	15589475 Azikwe	591	Spain	Female	39	0	1
31	15706552 Odinakachuk	533	France	Male	36	85311.7	0
32	15750181 Sanderson	553	Germany	Male	41	110112.54	0
33	15659428 Maggard	520	Spain	Female	42	0	0
34	15732963 Clements	722	Spain	Female	29	0	0
35	15794171 Lombardo	475	France	Female	45	134264.04	1
36	15788448 Watson	490	Spain	Male	31	145260.23	0
37	15729599 Lorenzo	804	Spain	Male	33	76548.6	0
38	15717426 Armstrong	850	France	Male	36	0	0
39	15585768 Cameron	582	Germany	Male	41	70349.48	0
40	15619360 Hsiao	472	Spain	Male	40	0	0
41	15738148 Clarke	465	France	Female	51	122522.32	1
42	15755196 Lavine	834	France	Female	49	131394.56	1
43	15684171 Bianchi	660	Spain	Female	61	155931.11	0
44	15754849 Tyler	776	Germany	Female	32	109421.13	0
45	15602280 Martin	829	Germany	Female	27	112045.67	1
46	15771573 Okagbare	637	Germany	Female	39	137843.8	1
47	15766205 Yin	550	Germany	Male	38	103391.38	0
48	15771873 Bucco	776	Germany	Female	37	103769.22	0
49	15616550 Chidiebele	698	Germany	Male	44	116363.37	0
50	15678193 Trevisani	585	Germany	Male	36	146050.97	0
51	15683553 Osman	788	USA	Male	22	75888.3	0
52	15702298 Parkhill	655	Germany	Male	41	125561.97	1
53	15656950 Yoo	601	Germany	Male	42	98495.72	1
54	15760861 Phillips	619	France	Male	43	125211.92	0
55	15630053 Tsao	656	France	Male	45	127864.4	0
56	15647091 Endrizzi	725	Germany	Male	19	75888.2	0
57	15623944 T'en	511	Spain	Female	66	0	1
58	15804771 Velazquez	614	France	Male	51	40685.92	0
59	15770811 Hunter	742	Germany	Male	35	136857	0
60	15773469 Clark	687	Germany	Female	27	152328.88	0
61	15702014 Jeffrey	555	Spain	Male	33	56084.69	0
62	15751209 Pirozzi	684	Spain	Male	56	78707.16	0
63	15592461 Jackson	603	Germany	Male	26	109166.37	0
64	15789484 Hammond	751	Germany	Female	36	169831.46	0
65	15696068 Brownless	581	Germany	Female	34	101633.04	0
66	15641582 Chibugo	735	Germany	Male	43	123180.01	0
67	15638424 Glauert	661	Germany	Female	35	150725.53	0
68	15703793 Konavalova	738	Germany	Male	58	133745.44	1
69	15620344 McKee	813	France	Male	29	0	0
70	16812518 Palermo	657	Spain	Female	37	163607.18	0
71	15779052 Ballard	604	Germany	Female	25	157780.84	0
72	15770811 Wallace	519	France	Male	36	0	0
73	15780961 Cavenagh	735	France	Female	21	178718.19	0
74	15614049 Hu	664	France	Male	55	0	0
75	15662085 Read	678	France	Female	32	0	0
76	15757185 Bushell	757	Spain	Male	33	77253.22	0
77	15803136 Postle	416	Germany	Female	41	122189.66	0
78	15706021 Buley	665	France	Female	34	96645.54	0
79	15663706 Leonard	777	France	Female	32	0	1
80	15641732 Mills	543	France	Female	36	0	0
81	15738751 Bell	493	France	Female	46	321	0
82	15805256 Ndukaku	652	Spain	Female	75	0	0
83	15762418 Gant	750	Spain	Male	22	121681.82	1
84	15625759 Rowley	729	France	Male	30	0	0
85	15622897 Sharpe	646	France	Female	46	0	1
86	15767954 Osborne	635	Germany	Female	28	81623.67	0
87	15757535 Heap	647	Spain	Female	44	0	1
88	15731511 Ritchie	808	France	Male	45	118626.55	0
89	15809248 Cole	524	France	Female	36	0	0
90	15640633 Capon	769	France	Male	29	0	0
91	15676966 Capon	730	Spain	Male	42	0	0
92	15699461 Fiorentini	515	Spain	Male	35	176273.95	0
93	15738721 Graham	773	Spain	Male	41	102827.44	0
94	15693683 Yuile	814	Germany	Male	29	97086.4	0
95	15633058 Fanucci	413	France	Male	34	0	0
96	15699392 Groves	756	Germany	Male	44	137452.09	0
97	15589975 Maclean	646	France	Female	73	97259.25	0
98	15726676 Marshall	616	Spain	Male	30	0	0
99	15771977 T'ao	730	France	Female	39	99010.67	0
100	15657566 Wieck	634	Germany	Male	24	103097.85	0
101	15727556 O'Donnell	744	Spain	Female	26	166297.89	0

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

The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays the Customer database with tables like CustomerChurn, CustomerChurn_Stake, and CustomerChurn_Version1. The main pane shows the SQL editor with the following code:

```

208    INTO TABLE Customer.CustomerChurn_Stake
209    COLUMNS TERMINATED BY ','
210    OPTIONALLY ENCLOSED BY '\"'
211    ESCAPED BY '\\'
212    LINES TERMINATED BY '\n'
213    IGNORE 3 LINES;
214
215    • SELECT COUNT(*) FROM Customer.CustomerChurn_Stake;
216
217    • SELECT * FROM Customer.CustomerChurn_Stake ORDER BY CustomerId ASC;
218
219

```

The Result Grid shows the following sample data:

CustomerID	Surname	CreditScore	Geography	Gender	Age	Balance	Exited
15760681	Phillips	619	France	Male	43	125211.92	0
15762418	Gant	750	Spain	Male	22	121681.82	1
15762420	Ward	540	Germany	Female	30	120000.00	0
15767821	Sharon	528	Hong Kong	Male	31	102016.72	0
15767854	Osborn	635	Germany	Female	28	81625.67	0
15768193	Trevianni	585	Germany	Male	36	8146050.97	0
15768200	Ward	500	France	Male	29	0.00	0
15771573	Odeighe	637	Germany	Female	39	137443.80	1
15771873	Buoho	776	Germany	Female	37	103769.22	0
15771977	Tiao	730	France	Female	39	6901.67	0
15772000	Ward	567	Germany	Female	30	120000.00	0
15779052	Ballard	654	Germany	Female	29	157760.84	0
15780961	Caverleigh	735	France	Female	21	178718.19	0
15788218	Henderson	549	Spain	Female	24	0.00	0
15789444	Harmont	649	Spain	Male	30	120000.23	0
15789444	Harmont	751	Germany	Female	36	169831.46	0
15792365	He	501	France	Male	44	124051.07	0
15794171	Lombardo	475	France	Female	45	134264.04	0
15800271	Ward	541	Germany	Female	37	121681.86	1
15804771	Valladares	614	France	Male	31	40684.67	0
15805254	Nakaku	652	Spain	Female	75	0.00	0
15809248	Cole	524	France	Female	36	0.00	0
15812518	Palermo	651	Spain	Female	37	169807.18	0

The Activity tab shows the following session history:

Time	Action	Response	Duration / Fetch Time
3 18:04:24	SELECT COUNT(*) FROM Customer...	1 row(s) returned	0.0014 sec / 0.00001...
4 18:14:16	SELECT * FROM Customer.Custo...	100 row(s) returned	0.0016 sec / 0.00008...
5 20:14:48	TRUNCATE Customer.CustomerCh...	0 row(s) affected	0.016 sec
6 20:14:51	LOAD DATA LOCAL INFILE 'User...	101 row(s) affected Records: 101 Deleted: 0 Skipped: 0 Warnings: 0	0.0077 sec
7 20:14:55	SELECT COUNT(*) FROM Customer...	1 row(s) returned	0.0030 sec / 0.00002...
8 20:54:53	SELECT * FROM Customer.Custo...	101 row(s) returned	0.0017 sec / 0.00004...

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;
```

```

219 -- 07. Execute the stored procedure, Customer.PrCustomerChurn, that was created in 04. After execution, the stored procedure should load data from the stage to the
220 persistent table: Customer.CustomerChurn. CALL 'customer'.'PrCustomerChurn'(); This time, the table will be refreshed via DELETE, UPDATE, and INSERT/SELECT
221 statements in the stored procedure. Show the row count results of both Customer.CustomerChurn_Version1 table [Data Source: CustomerChurn1.CSV] and the persistent
222 table: Customer.CustomerChurn. Compare the rows between the Customer.CustomerChurn_Version1 [Data Source: CustomerChurn1.CSV] table and the persistent table:
223 Customer.CustomerChurn [Data Source: CustomerChurn2.CSV]. Show the rows that are available in the Customer.CustomerChurn_Version1 table but not in the
224 Customer.CustomerChurn table (implementation of brand-new row DELETE statement of the stored procedure).
225
226 CALL Customer.PrCustomerChurn();
227
228 SELECT (SELECT COUNT(*) FROM Customer.CustomerChurn_Version1) as CustomerChurn_Version1Count, (SELECT COUNT(*) FROM Customer.CustomerChurn) as CustomerChurnCount;

```

Result Grid

CustomerChurn_Version1Count	CustomerChurnCount
100	101

Result 90

Action Output

	Time	Action	Response	Duration / Fetch Time
1	21:34:38	CALL Customer.PrCustomerChurn()	7 row(s) affected	0.014 sec
2	21:39:25	SELECT (SELECT COUNT(*) FROM Custo... 1 row(s) returned		0.0072 sec / 0.00002...

```

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, '*')
);

```

The screenshot shows a database interface with a query results window. The query retrieves data from two tables: Customer.CustomerChurn (TT) and Customer.CustomerChurn_Version1 (TTV1). The results are displayed in a table with the following columns:

CustomerID	SurnameOld	SurnameNew	CreditScoreOld	CreditScoreNew	GenderOld	GenderNew	AgeOld	AgeNew	BalanceOld	BalanceNew	ExitedOld	ExitedNew	SourceSystemNmOld	SourceSystemNmNew	CreateAgentIdOld	CreateAgentIdNew	CreateDtmOld	CreateDtmNew
15589975	Maclean	WILL	646	WILL	Female	WILL	73	WILL	97259.25	WILL	0	WILL	Kaggle-CSV					
15589289	Hivek	WILL	684	WILL	Male	Male	27	27	134003.88	134003.88	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15589280	Defucci	Domenick	655	TODD	Female	Female	23	23	100000.00	100000.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15561766	Wleck	WILL	634	WILL	Male	Male	24	24	103079.85	103079.85	0	0	Kaggle-CSV					
15683553	O'Brien	Osman	788	788	Female	Male	33	22	0.00	75888.30	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15691483	Chin	Chin	549	549	Female	Female	25	25	0.00	12945.50	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15589281	Conrad	WILL	758	WILL	Male	Male	44	44	150000.00	150000.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15729567	Marshall	WILL	616	WILL	Male	Male	30	WILL	0.00	WILL	0	WILL	Kaggle-CSV					
15729566	O'Donnell	WILL	744	WILL	Female	Male	26	WILL	169297.88	WILL	0	WILL	Kaggle-CSV					
15727173	Andrews	Andrews	497	497	Male	Male	24	30	0.00	0.00	0	1	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15727172	Don	Don	497	497	Female	Female	28	28	0.00	5261.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15767821	Bearce	Sharon	528	528	Male	Male	31	31	162016.72	162016.72	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-06-03 22:28	
15717977	Wao	Tao	730	WILL	Female	Male	39	WILL	99010.67	WILL	0	WILL	Kaggle-CSV					
16825158	Palermo	Palermo	657	WILL	Female	Female	57	WILL	163867.18	WILL	0	WILL	Kaggle-CSV					
15589280	Allred	Eric	710	WILL	Male	Male	25	25	0.00	0.00	0	0	Kaggle-CSV					
15667946	Ozborne	WILL	556	WILL	Female	Female	61	WILL	117419.35	WILL	0	WILL	Kaggle-CSV					
15701164	Oymenolu	WILL	506	WILL	Female	Female	34	WILL	90507.62	WILL	0	WILL	Kaggle-CSV					
15729573	Mosman	WILL	669	WILL	Male	Male	46	WILL	0.00	WILL	0	WILL	Kaggle-CSV					
15729572	Ward	Ward	671	WILL	Female	Female	45	WILL	202.28	WILL	0	WILL	Kaggle-CSV					
15825119	Palermo	WILL	657	WILL	Female	Male	37	WILL	163867.18	WILL	0	WILL	Kaggle-CSV					

At the bottom of the interface, there is a status bar showing the time (22:28:46), action (SELECT TT.CustomerId as CustomerId, TTV1.Surname...), number of rows returned (20 row(s) returned), duration (0.0028 sec / 0.00002), and a 'Read Only' indicator.

1. Here 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;
```

CustomerID	Surname	CreditScore	Geography	Gender	Age	Balance	Exited	SourceSystemNm	CreateAgentId	CreateDtm	ChangeAgentId	ChangeDtm
15604548	Allard	710	Spain	Male	22	0.00	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13
15607954	Osborn	2	France	Female	61	997419.38	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13
15725744	Thompson	506	France	Female	44	997419.62	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13
15725737	Mosman	669	France	Male	46	0.00	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13
15755648	Pisano	675	France	Female	21	98373.28	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13
15812518	Palermo	657	Spain	Female	37	1036071.18	0	Kaggle-CSV	root@localhost	2024-08-03 22:28:13	root@localhost	2024-08-03 22:28:13

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 ColumnsNames

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;

```

```

301     TTV1.CreateAgentId AS CreateAgentIdOld, TT.CreateAgentId AS CreateAgentIdNew,
302     TTV1.CreateDtm AS CreateDtmOld, TT.CreateDtm AS CreateDtmNew,
303     TTV1.ChangeAgentId AS ChangeAgentIdOld, TT.ChangeAgentId AS ChangeAgentIdNew,
304     TTV1.ChangeDtm AS ChangeDtmOld, TT.ChangeDtm AS ChangeDtmNew
305   FROM
306     Customer.CustomerChurn AS TT
307   INNER JOIN Customer.CustomerChurn_Version1 AS TTV1
308     ON TT.CustomerId = TTV1.CustomerId
309   WHERE
310     (
311       COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
312       OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
313       OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
314       OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
315       OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
316       OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
317       OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
318     ) ORDER BY TT.CustomerId;
319
320
321  ◇ 21:303

```

Result Grid Filter Rows: Search Export

Customerid	SurnameOld	SurnameNew	CreditScoreOld	CreditScoreNew	GenderOld	GenderNew	AgeOld	AgeNew	BalanceOld	BalanceNew	ExitedOld	ExitedNew	SourceSystemNmOld	SourceSystemNmNew	CreateAgentIdOld	CreateAgentIdNew	CreateDtmOld	CreateDtmNew	ChangeAgentIdOld	ChangeAgentIdNew
15692389	Hirsi	Vivek	684	684	Male	Male	27	27	134603.88	134603.88	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15697945	Dilello	Gabe	700	700	Female	Female	32	32	0.00	654.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15693353	O'Brien	Oman	788	788	Female	Male	33	22	0.00	75888.30	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15691483	Chin	Chin	549	549	Female	Female	25	25	0.00	12945.50	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15691483	Chin	Chin	549	549	Male	Male	25	25	0.00	12945.50	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15738751	Allen	Andrews	497	497	Male	Male	24	30	0.00	321.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15738751	Allen	Bell	493	493	Female	Female	46	46	0.00	321.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost
15767821	Bearce	Sharon	528	528	Male	Male	31	31	102016.72	102016.72	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost

Action Output

Time	Action	Response	Duration / Fetch Time
1	23:23:13	SELECT TT.CustomerId AS CustomerId, TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew, TT... 7 row(s) returned	0.0039 sec / 0.0000...
2	23:26:21	SELECT * FROM Customer.CustomerChurn AS TT INNER JOIN Customer.CustomerChurn_Version1 AS TT... 7 row(s) returned	0.0034 sec / 0.0000...
3	23:27:41	SELECT * FROM Customer.CustomerChurn AS TT INNER JOIN Customer.CustomerChurn_Version1 AS TT... 7 row(s) returned	0.0029 sec / 0.0000...
4	23:28:27	SELECT TT.CustomerId AS CustomerId, TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew, TT... 7 row(s) returned	0.0023 sec / 0.00003

```

301     TTV1.CreateAgentId AS CreateAgentIdOld, TT.CreateAgentId AS CreateAgentIdNew,
302     TTV1.CreateDtm AS CreateDtmOld, TT.CreateDtm AS CreateDtmNew,
303     TTV1.ChangeAgentId AS ChangeAgentIdOld, TT.ChangeAgentId AS ChangeAgentIdNew,
304     TTV1.ChangeDtm AS ChangeDtmOld, TT.ChangeDtm AS ChangeDtmNew
305   FROM
306     Customer.CustomerChurn AS TT
307   INNER JOIN Customer.CustomerChurn_Version1 AS TTV1
308     ON TT.CustomerId = TTV1.CustomerId
309   WHERE
310     (
311       COALESCE(TT.Surname, '*') <> COALESCE(TTV1.Surname, '*')
312       OR COALESCE(TT.CreditScore, '*') <> COALESCE(TTV1.CreditScore, '*')
313       OR COALESCE(TT.Geography, '*') <> COALESCE(TTV1.Geography, '*')
314       OR COALESCE(TT.Gender, '*') <> COALESCE(TTV1.Gender, '*')
315       OR COALESCE(TT.Age, '*') <> COALESCE(TTV1.Age, '*')
316       OR COALESCE(TT.Balance, '*') <> COALESCE(TTV1.Balance, '*')
317       OR COALESCE(TT.Exited, '*') <> COALESCE(TTV1.Exited, '*')
318     ) ORDER BY TT.CustomerId;
319
320
321  ◇ 21:303

```

Result Grid Filter Rows: Search Export

Old.CreditScoreNew	GenderOld	GenderNew	AgeOld	AgeNew	BalanceOld	BalanceNew	ExitedOld	ExitedNew	SourceSystemNmOld	SourceSystemNmNew	CreateAgentIdOld	CreateAgentIdNew	CreateDtmOld	CreateDtmNew	ChangeAgentIdOld	ChangeAgentIdNew	ChangeDtmOld	ChangeDtmNew
684	Male	Male	27	27	134603.88	134603.88	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
700	Female	Female	32	32	654.00	0.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
788	Female	Male	33	22	0.00	75888.30	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
549	Female	Female	25	25	0.00	12945.50	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
497	Male	Male	24	30	0.00	0.00	0	1	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
493	Female	Female	46	46	0.00	321.00	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13
528	Male	Male	31	31	102016.72	102016.72	0	0	Kaggle-CSV	Kaggle-CSV	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13	root@localhost	root@localhost	2024-08-03 22:28:13	2024-08-03 22:28:13

Action Output

Time	Action	Response	Duration / Fetch Time
1	23:23:13	SELECT TT.CustomerId AS CustomerId, TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew, TT... 7 row(s) returned	0.0039 sec / 0.0000...
2	23:26:21	SELECT * FROM Customer.CustomerChurn AS TT INNER JOIN Customer.CustomerChurn_Version1 AS TT... 7 row(s) returned	0.0034 sec / 0.0000...
3	23:27:41	SELECT * FROM Customer.CustomerChurn AS TT INNER JOIN Customer.CustomerChurn_Version1 AS TT... 7 row(s) returned	0.0029 sec / 0.0000...
4	23:28:27	SELECT TT.CustomerId AS CustomerId, TTV1.Surname AS SurnameOld, TT.Surname AS SurnameNew, TT... 7 row(s) returned	0.0023 sec / 0.00003

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;

```

CustomerID Surname CreditScore Geography Gender Age Balance Exited SourceSystemNm CreateAgentId CreateDtm ChangeAgentId ChangeDtm CustomerID Surname CreditScore Geography Gender Age Balance Exited SourceSystemNm CreateAgentId CreateDtm

15592389	Vwek	684	India	Male	27	134603.88	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15592389	H7	684	France	Male	27	134603.88	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15597945	Deluod	700	Spain	Female	32	654.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15597945	Deluod	636	Spain	Female	32	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15601496	O'Brien	603	Spain	Female	37	15601496.30	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15601496	O'Brien	768	France	Female	37	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15601483	Chin	149	France	Female	25	12345.50	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15601483	Chin	449	France	Female	25	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15737173	Andrews	497	Spain	Male	30	0.00	1	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15737173	Andrews	497	Spain	Male	24	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15738751	Bell	493	France	Female	46	321.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15738751	Bell	493	France	Female	46	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
15767821	Sharon	528	Hong Kong	Male	31	102018.72	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15767821	Bearce	528	France	Male	31	102018.72	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13

CustomerID Surname CreditScore Geography Gender Age Balance Exited SourceSystemNm CreateAgentId CreateDtm ChangeAgentId ChangeDtm CustomerID Surname CreditScore Geography Gender Age Balance Exited SourceSystemNm CreateAgentId CreateDtm

India	Male	27	134603.88	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15592389	H7	684	France	Male	27	134603.88	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
Spain	Female	32	654.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15597945	Deluod	636	Spain	Female	32	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
USA	Male	22	75888.30	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15601496	O'Brien	768	France	Female	33	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
France	Female	37	15601496.30	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15601483	Chin	449	France	Female	37	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
Spain	Male	50	0.00	1	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15737173	Andrews	497	France	Male	24	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
France	Female	46	321.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15738751	Bell	493	France	Female	46	0.00	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13
Hong Kong	Male	31	102018.72	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13	15767821	Bearce	528	France	Male	31	102018.72	0	Kaggle-CSV	root@localhost	2024-03-03 22:28:13

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.

-- 09. 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.														
CustomerID	Surname	CreditScore	Geography	Gender	Age	Balance	Exited	SourceSystemNm	CreateAgentId	CreateDtM	ChangeAgentId	ChangeDtM	Weight	RowNum
1571873	Bogacho	776	Germany	Female	37	103000	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	1
1571874	Tschauder	540	Germany	Male	30	103010.67	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	2
15773469	Clark	687	Germany	Female	27	152328.88	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	3
15779052	Ballard	604	Germany	Female	25	15780.84	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	4
15780996	Calveigh	735	France	Female	21	176718.18	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	5
15781000	Ward	492	Spain	Male	31	146260.23	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	6
15788448	Watson	490	Spain	Male	31	146260.23	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	7
15789445	Hammond	761	Germany	Female	36	169851.46	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	8
15792055	He	501	France	Male	44	142051.07	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	9
15792155	Alvarez	725	Spain	Male	31	122000.00	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	10
15803136	Potter	416	Germany	Female	41	122188.66	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	11
15847711	Velazquez	164	France	Male	51	40685.92	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	12
15850000	Yokuaku	524	Spain	Female	76	0.00	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	13
15852649	Reed	262	Spain	Male	30	0.00	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	14
16812510	Palermo	657	Spain	Female	37	169867.38	No	Kaggle-CSV	root@localhost	2024-03-22 28:13	root@localhost	2024-03-22 28:13	1	15
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														
<hr/>														

```
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;
```

```

341 -- 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.
342
343 • SELECT
344     TT.* 
345 FROM
346     Customer.CustomerChurn AS TT
347     LEFT OUTER JOIN Customer.CustomerChurn_Version1 AS TTV1 ON TTV1.CustomerId = TT.CustomerId
348 WHERE
349     TTV1.CustomerId IS NULL ORDER BY TTV1.CustomerId ASC;
350
351
100% 0 11:344
Result Grid Filter Rows Search Export:
CustomerID Surname CreditScore Geography Gender Age Balance Exited SourceSystemName CreateAgentId CreateDtM ChargeAgentId ChargeDtM
15569975 Marcan 646 France Female 73 9759.26 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
15687668 Week 634 Germany Male 24 10397.85 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
15669932 Groves 756 Germany Male 44 137452.09 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
15798679 Marshall 716 Spain Male 50 0.00 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
15725109 O'Donnell 744 Spain Female 29 16597.89 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
15719977 Tao 730 France Female 39 99010.67 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13
16872518 Palermo 657 Spain Female 37 163607.18 0 Kaggie-CSV root@localhost 2024-03-03 22:28:13 root@localhost 2024-03-03 22:28:13

```

Action Output

Time	Action	Response	Dura
23:45:22	SELECT TT* FROM Customer.CustomerChurn AS TT LEFT OUTER JOIN Customer.Custo...	7 row(s) returned	0.0024 sec / 0.00004

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 ;
```