Lab- 09

Topic: STORED PROCEDURE, Triggers

Stored Procedures:

Stored procedures (SPs) in SQL Server are just like procedures/routines in other programming languages. Each procedure has one or more statements. In our case, these are SQL statements. So, you can write a procedure that will – insert new data, update or delete existing, retrieve data using the SELECT statement.

Example:1

```
DROP PROCEDURE IF EXISTS p_customer_all;
GO
CREATE PROCEDURE p_customer_all
-- procedure returns all rows from the customer table
AS BEGIN
  SELECT *
  FROM customer;
END;
```

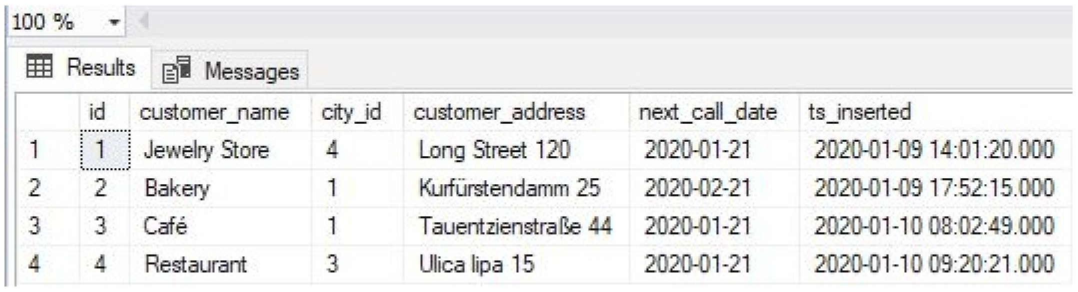
Explanation:

- We've used the DROP PROCEDURE IF EXISTS p_customer_all; statement in the first line. This is nice practice, especially when you're creating scripts you want to work always, no matter the state of the database. The command DROP PROCEDURE p_customer_all; would delete the procedure with the given name. Still, if the procedure wasn't already created in the database, this would result in an error. Therefore, adding IF EXISTS prevents this from happening. This row generally says – I will delete this procedure if it's on the server, and if it is not present, OK, do nothing.
- The word GO is inserted between two SQL statements in situations like this one.
- The name of our procedure is p_customer_all.
- The body of the procedure is just a simple select statement returning all rows from this table

To do this, we'll use the syntax: EXEC procedure_name <parameters if any>;. So, our statement is:

```
EXEC p_customer_all;
```

Result will be shown as below:


100 %
Results | Messages

	id	customer_name	city_id	customer_address	next_call_date	ts_inserted
1	1	Jewelry Store	4	Long Street 120	2020-01-21	2020-01-09 14:01:20.000
2	2	Bakery	1	Kurfürstendamm 25	2020-02-21	2020-01-09 17:52:15.000
3	3	Café	1	Tauentzienstraße 44	2020-01-21	2020-01-10 08:02:49.000
4	4	Restaurant	3	Ulica lipa 15	2020-01-21	2020-01-10 09:20:21.000

Example:2
Procedure that will return only one row based on the id given as input by user:

```
DROP PROCEDURE IF EXISTS p_customer;
GO
CREATE PROCEDURE p_customer (@id INT)
-- procedure returns the entire row for the given id
AS BEGIN
  SELECT *
  FROM customer
  WHERE id = @id;
END;
```

Execute this procuder using this syntex:

```
EXEC p_customer 4;
```

The result is, as expected, all details for the customer with id = 4.

Example:3
Procedure that will insert a new customer in the table.

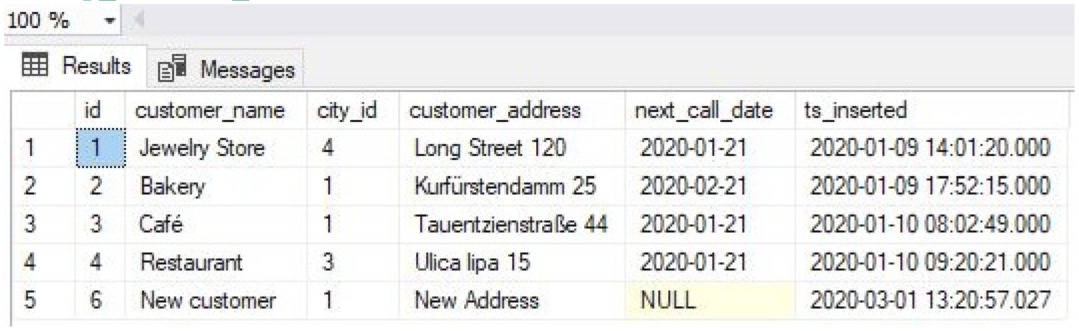
```
DROP PROCEDURE IF EXISTS p_customer_insert;
GO
CREATE PROCEDURE p_customer_insert (@customer_name VARCHAR(255), @city_id INT,
@customer_address VARCHAR(255), @next_call_date DATE)
-- procedure inserts a new customer
AS BEGIN
  INSERT INTO customer (customer_name, city_id, customer_address, next_call_date,
ts_inserted)
  VALUES (@customer_name, @city_id, @customer_address, @next_call_date,
SYSDATETIME());
END;
```

After executing the procedure, using the statement:

```
EXEC p_customer_insert "New customer", 1, "New Address", NULL;
```

the new row was added. We’ll check what is in the table by calling the first procedure we’ve created:

```
EXEC p_customer_all;
```



	id	customer_name	city_id	customer_address	next_call_date	ts_inserted
1	1	Jewelry Store	4	Long Street 120	2020-01-21	2020-01-09 14:01:20.000
2	2	Bakery	1	Kurfürstendamm 25	2020-02-21	2020-01-09 17:52:15.000
3	3	Café	1	Tauentzienstraße 44	2020-01-21	2020-01-10 08:02:49.000
4	4	Restaurant	3	Ulica lipa 15	2020-01-21	2020-01-10 09:20:21.000
5	6	New customer	1	New Address	NULL	2020-03-01 13:20:57.027

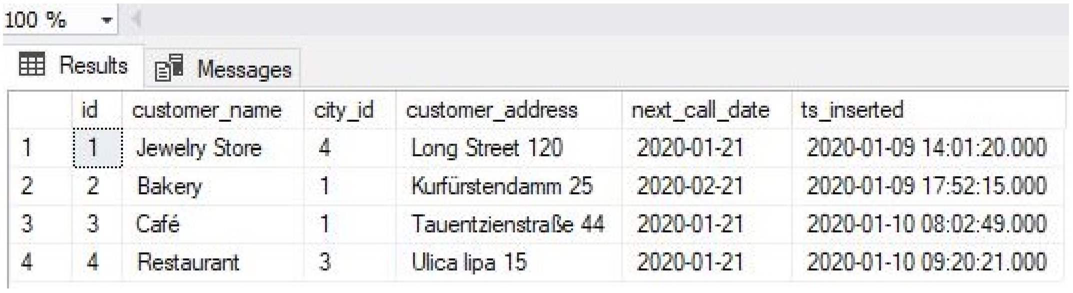
Example:4

<u>Procedure to delete a row using the id passed as parameter.</u>

```
DROP PROCEDURE IF EXISTS p_customer_delete;
GO
CREATE PROCEDURE p_customer_delete (@id INT)
-- procedure deletes the row for the given id
AS BEGIN
  DELETE
  FROM customer
  WHERE id = @id;
END;
```

We pass only 1 parameter and that is the id of the row to delete. Let’s call the procedure now: Once again, we’ve followed the same naming convention when giving the name to our procedure. We pass only 1 parameter and that is the id of the row to delete. Let’s call the procedure now:

```
EXEC p_customer_delete 6;
```



	id	customer_name	city_id	customer_address	next_call_date	ts_inserted
1	1	Jewelry Store	4	Long Street 120	2020-01-21	2020-01-09 14:01:20.000
2	2	Bakery	1	Kurfürstendamm 25	2020-02-21	2020-01-09 17:52:15.000
3	3	Café	1	Tauentzienstraße 44	2020-01-21	2020-01-10 08:02:49.000
4	4	Restaurant	3	Ulica lipa 15	2020-01-21	2020-01-10 09:20:21.000

Triggers

What is a Trigger

i. A SQL trigger is special stored procedure that runs when specific actions occur within a database. Most database triggers are defined to run when changes are made to a table's data. Triggers can be defined to run *instead of* or *after* DML (Data Manipulation Language) actions such as INSERT, UPDATE, and DELETE.

ii. Triggers help the database designer ensure certain actions, such as maintaining an audit file, are completed regardless of which program or user makes changes to the data.

iii. The programs are called triggers since an event, such as adding a record to a table, fires their execution.

iv. Triggers and their implementations are specific to database vendors. The concepts are the same or similar in Microsoft SQL server, Oracle and MySQL.

i) After Triggers

These triggers run after an insert, update or delete on a table. They are **not supported for views.** AFTER TRIGGERS can be classified further into three types as:

a. AFTER INSERT Trigger
b. AFTER UPDATE Trigger
c. AFTER DELETE Trigger

Create Triggers

Example:

```
CREATE TRIGGER insert_trigger
ON citizen
AFTER INSERT
AS
BEGIN
      PRINT 'New data added or updated.';
END;
```

Audit Trigger:

Example:

```
CREATE TRIGGER audit_trigger
ON citizen
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   IF EXISTS(SELECT * FROM inserted)
      PRINT 'New data added or updated.';
   ELSE IF EXISTS(SELECT * FROM deleted)
      PRINT 'Data deleted.';
END;
```

Triggers:

Use the Northwind schema for this purpose.

Use customer Table:

The audit table will have **CustomerNumber, CustomerName, City** and **Country.**

Question:1

Write a Trigger on the Customer Table that will invoke whenever a new entry is being inserted into the table.

Question:2

Write a Trigger on the Customer Table that will invoke whenever a new entry is being updated in the table.

Question:3

Write a Trigger on the Customer Table that will invoke whenever any entry is being deleted from the table.

Use Employee Table:

The audit table will have **FirstName, LastName, Email**, **Title, Hire Date**

Question:4

Write a Trigger on the Employee Table that will invoke whenever a new entry is being inserted into the table.

Question:5

Write a Trigger on the Employee Table that will invoke whenever a new entry is being updated in the table.

Question:6

Write a Trigger on the Employee Table that will invoke whenever any entry is being deleted from the table.

Stored Procedures:

Question-07: Create and display a procedure that displays all the employees**.**

Question-08: Create and display a procedure that displays all the records of the employees based on the Extension number given by the user.

Question-09: Create and displays a procedure that displays the count of the employees associated with the city “London”.

Question-10: Create and displays a procedure that inserts employee’s ID=10, Firstname=Ali, Lastname=Khan, Title= Sales Manager.