AIM:	To implement triggers.

Theory :

Triggers in SQL Server

A trigger is a set of SQL statements that reside in system memory with unique names. It is a specialized category of stored procedure that is called automatically when a database server event occurs. Each trigger is always associated with a table.

A **trigger is called a special procedure** because it cannot be called directly like a stored procedure. The key distinction between the trigger and procedure is that a trigger is called automatically when a data modification event occurs against a table. A stored procedure, on the other hand, must be invoked directly.

The following are the main characteristics that distinguish triggers from stored procedures:

- We cannot manually execute/invoked triggers.
- Triggers have no chance of receiving parameters.
- A transaction cannot be committed or rolled back inside a trigger.

Syntax of Trigger

We can create a trigger in SQL Server by using the **CREATE TRIGGER** statement as follows:

```sql
CREATE TRIGGER schema.trigger_name
ON table_name
AFTER  {INSERT, UPDATE, DELETE}
[NOT FOR REPLICATION]
AS
{SQL_Statements}
```

The parameter descriptions of this syntax illustrate below:

schema: It is an optional parameter that defines which schema the new trigger belongs to.

trigger_name: It is a required parameter that defines the name for the new trigger.

table_name: It is a required parameter that defines the table name to which the trigger applies. Next to the table name, we need to write the AFTER clause where any events like INSERT, UPDATE, or DELETE could be listed.

NOT FOR REPLICATION: This option tells that SQL Server does not execute the trigger when data is modified as part of a replication process.

SQL_Statements: It contains one or more SQL statements that are used to perform actions in response to an event that occurs.

When we use triggers?

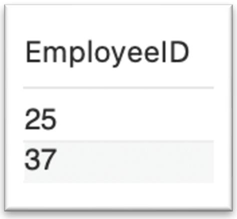
Triggers will be helpful when we need to execute some events automatically on certain desirable scenarios. **For example**, we have a constantly changing table and need to know the occurrences of changes and when these changes happen. If the primary table made any changes in such scenarios, we could create a trigger to insert the desired data into a separate table.

Queries

1. **Syntax:**

```
create view  view1 as
Select EmployeeID
From Orders
where EmployeeID in
(Select EmployeeID
from DeliveryPerson
where Rating>3)
group by EmployeeID;
```

Result:

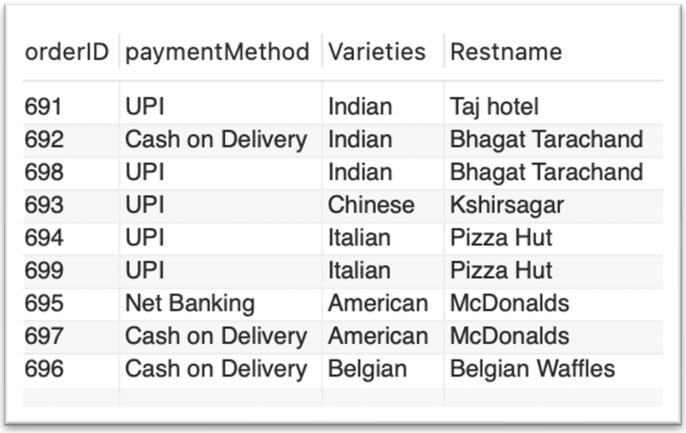


EmployeeID
25
37

2. **Syntax:**

```
create view  view2 as
select o.orderID, o.paymentMethod, r.Varieties, r.Restname
from orders as o
right join Restaurants as r
on r.ResturID=o.ResturID;
```

Result:

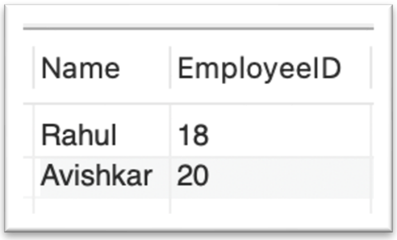


orderID	paymentMethod	Varieties	Restname
691	UPI	Indian	Taj hotel
692	Cash on Delivery	Indian	Bhagat Tarachand
698	UPI	Indian	Bhagat Tarachand
693	UPI	Chinese	Kshirsagar
694	UPI	Italian	Pizza Hut
699	UPI	Italian	Pizza Hut
695	Net Banking	American	McDonalds
697	Cash on Delivery	American	McDonalds
696	Cash on Delivery	Belgian	Belgian Waffles

3. **Syntax:**

```
create view view3 as
Select DelName as Name,EmployeeID
From DeliveryPerson
where EmployeeID in
(Select EmployeeID
from Orders
where RestName in ("Bhagat Tarachand","Taj hotel"));
```

Result:

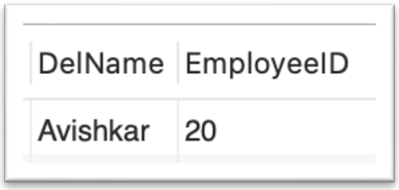


Name	EmployeeID
Rahul	18
Avishkar	20

4. **Syntax:**

```
create view view4 as
Select DelName,EmployeeID
From DeliveryPerson
where EmployeeID in
(Select EmployeeID
from Orders
where CustomerID=21458);
```

Result:

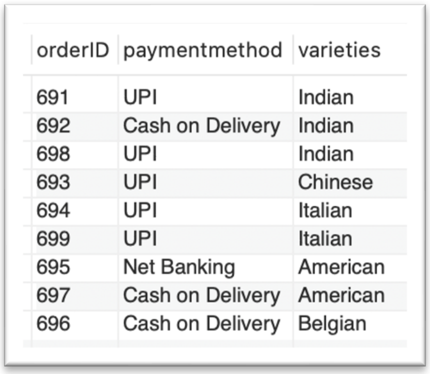


DelName	EmployeeID
Avishkar	20

5. **Syntax:**

```
create view view5 as
select o.orderID, o.paymentmethod, r.varieties
from orders as o
join Restaurants as r
on r.ResturID=o.ResturID;
```

Result:

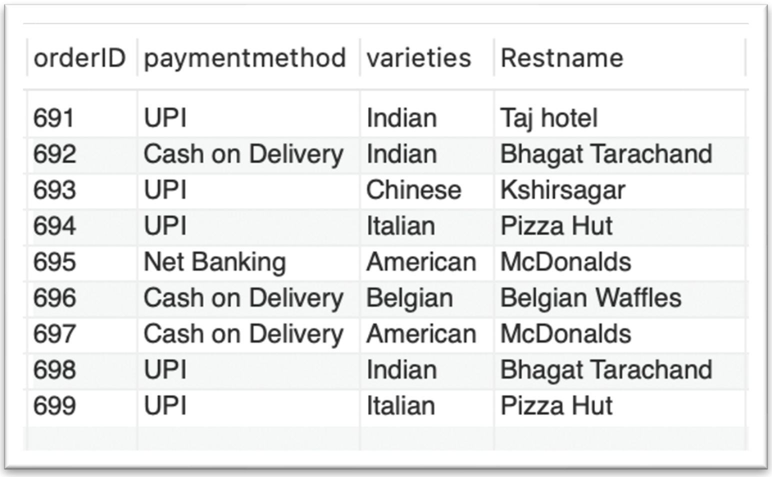


orderID	paymentmethod	varieties
691	UPI	Indian
692	Cash on Delivery	Indian
698	UPI	Indian
693	UPI	Chinese
694	UPI	Italian
699	UPI	Italian
695	Net Banking	American
697	Cash on Delivery	American
696	Cash on Delivery	Belgian

6. **Syntax:**

```
create view view6 as
select o.orderID, o.paymentmethod, r.varieties, r.Restname
from orders as o
LEFT join Restaurants as r
on r.ResturID=o.ResturID
union
select o.orderID, o.paymentmethod, r.varieties, r.Restname
from orders as o
right join Restaurants as r
on r.ResturID=o.ResturID;
```

Result:



orderID	paymentmethod	varieties	Restname
691	UPI	Indian	Taj hotel
692	Cash on Delivery	Indian	Bhagat Tarachand
693	UPI	Chinese	Kshirsagar
694	UPI	Italian	Pizza Hut
695	Net Banking	American	McDonalds
696	Cash on Delivery	Belgian	Belgian Waffles
697	Cash on Delivery	American	McDonalds
698	UPI	Indian	Bhagat Tarachand
699	UPI	Italian	Pizza Hut

Conclusion

From this experiment I learned to create tiggers, update data using triggers and delete data using triggers.