Prerequisite :

TRIGGERS:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Benefits of Triggers

Triggers can be written for the following purposes:

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name: Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
- [OF col_name]: This specifies the column name that would be updated.
- [ON table_name]: This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Q-1 Create a trigger to update history table when the price of product is updated in the product table.

```
SQL> select * from product16;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha         200
   102 B        Akshat          600
   103 C        Somya           400
   104 D        Abhilasha        400
SQL>  select * from Product16_History;
Empty set (0.06 sec)
SQL>
 CREATE OR REPLACE TRIGGER P_H_T BEFORE UPDATE OF UNIT_PRICE ON PRODUCT16 FOR EACH ROW
BEGIN INSERT INTO PRODUCT16_HISTORY   VALUES
(:old.Pid,:old.pname,:old.supplier_name,:old.unit_price);
End;
Trigger created.

SQL> update product16 set unit_price=700 where pid=101;

1 row updated.

SQL> select * from product16;
```

```
   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha         700
   102 B        Akshat          600
   103 C        Somya           400
   104 D        Abhilasha        400

SQL> select * from product16_history;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha         200
```

Q-2 Insert the product information into history table whenever a new product is entered in the product table.

```
SQL>  CREATE OR REPLACE TRIGGER P_H_T_I AFTER INSERT ON PRODUCT16 For each row BEGIN INSERT
INTO PRODUCT16_HISTORY VALUES (:new.Pid,:new.pname,:new.supplier_name,:new.unit_price);
End;  2
  3  /

Trigger created.

SQL> insert into product16 values(105,'E','Kartik',400);

1 row created.
SQL> select * from product16;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha         700
   102 B        Akshat          600
   103 C        Somya           400
   104 D        Abhilasha        400
   105 E        Kartik          400

SQL> select * from product16_history;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha         200
   105 E        Kartik          400
```

Q-3 Always copy the product information before removing it from the product table.

```
SQL>  CREATE OR REPLACE TRIGGER P_H_T_D Before delete ON PRODUCT16 For each row BEGIN INSERT
INTO PRODUCT16_HISTORY VALUES (:old.Pid,:old.pname,:old.supplier_name,:old.unit_price);
End;  2
Trigger created.
```

```
SQL> select * from product16;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha          700
   103 C        Somya           900
   104 D        Abhilasha         900
   105 E        Ankur           400

SQL> select * from product16_history;

   PID PNAME      SUPPLIER_NAME      UNIT_PRICE
---------- -------------------- -------------------- ----------
   101 A        Akansha          200
   105 E        Ankur           400
   102 B        Akshat           600
```