EXPERIMENT NO 10

Title : Study and implement Views and Triggers in database system

Aim: Perform View and Trigger

Theory:

Views:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

Example:

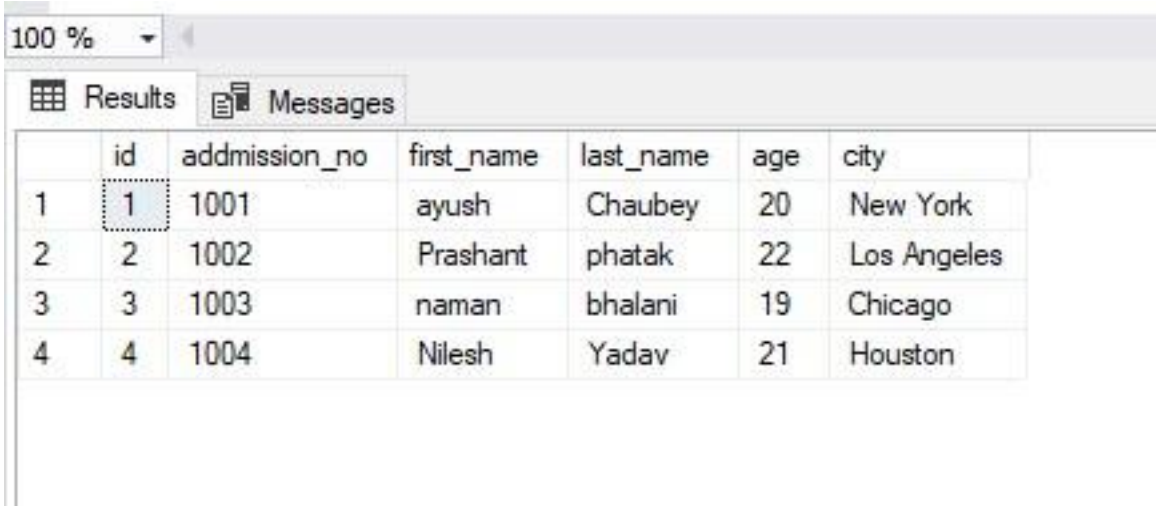
```
-- create the Student_details table CREATE
TABLE Student_details (
```

```
   id INTEGER,
addmission_no INTEGER,
first_name VARCHAR(10),
last_name VARCHAR(20),
age INTEGER,
   city VARCHAR(20)
);
```

-- insert some data into the Student_details table

```
INSERT INTO Student_details (id, addmission_no, first_name, last_name, age,
city)
VALUES
   (1, 1001, 'ayush', 'Chaubey', 20, 'New York'),
   (2, 1002, 'Prashant', 'phatak', 22, 'Los Angeles'),
   (3, 1003, 'naman', 'bhalani', 19, 'Chicago'),
   (4, 1004, 'Nilesh', 'Yadav', 21, 'Houston');

SELECT * FROM Student_details;
```


100 %

Results Messages

	id	addmission_no	first_name	last_name	age	city
1	1	1001	ayush	Chaubey	20	New York
2	2	1002	Prashant	phatak	22	Los Angeles
3	3	1003	naman	bhalani	19	Chicago
4	4	1004	Nilesh	Yadav	21	Houston

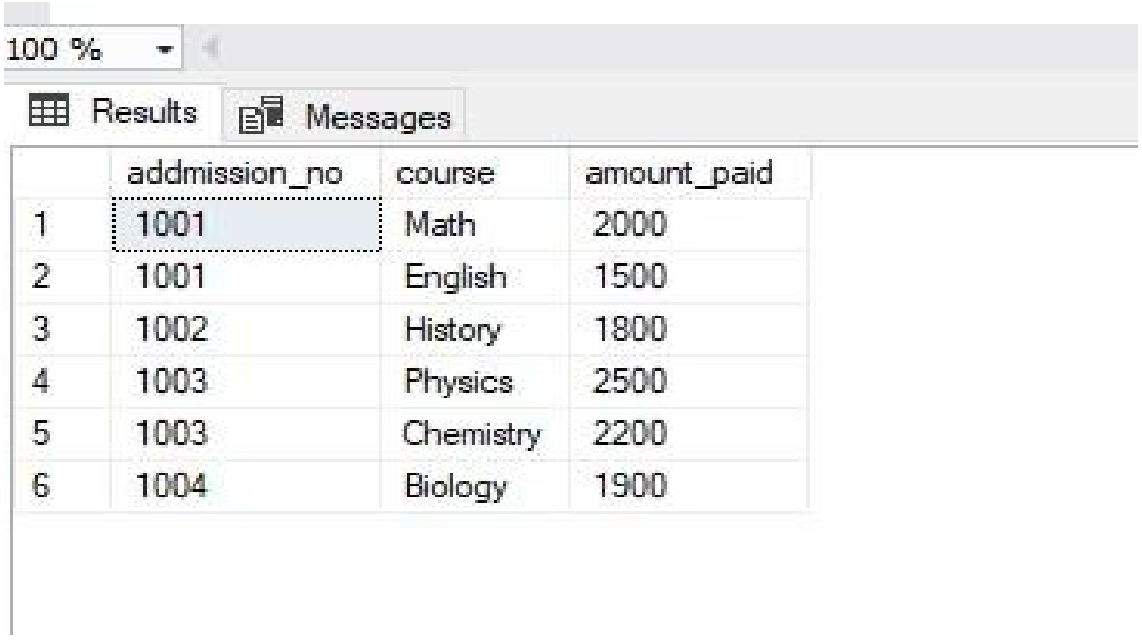
-- create the fees table

```
CREATE TABLE fees (
addmission_no INTEGER,     course
VARCHAR(20),
   amount_paid INTEGER
```

```
);

-- insert some data into the fees table
INSERT INTO fees (addmission_no, course, amount_paid)
VALUES
   (1001, 'Math', 2000),
   (1001, 'English', 1500),
   (1002, 'History', 1800),
   (1003, 'Physics', 2500),

   (1003, 'Chemistry', 2200),
   (1004, 'Biology', 1900);
SELECT * FROM fees;
```


100 %

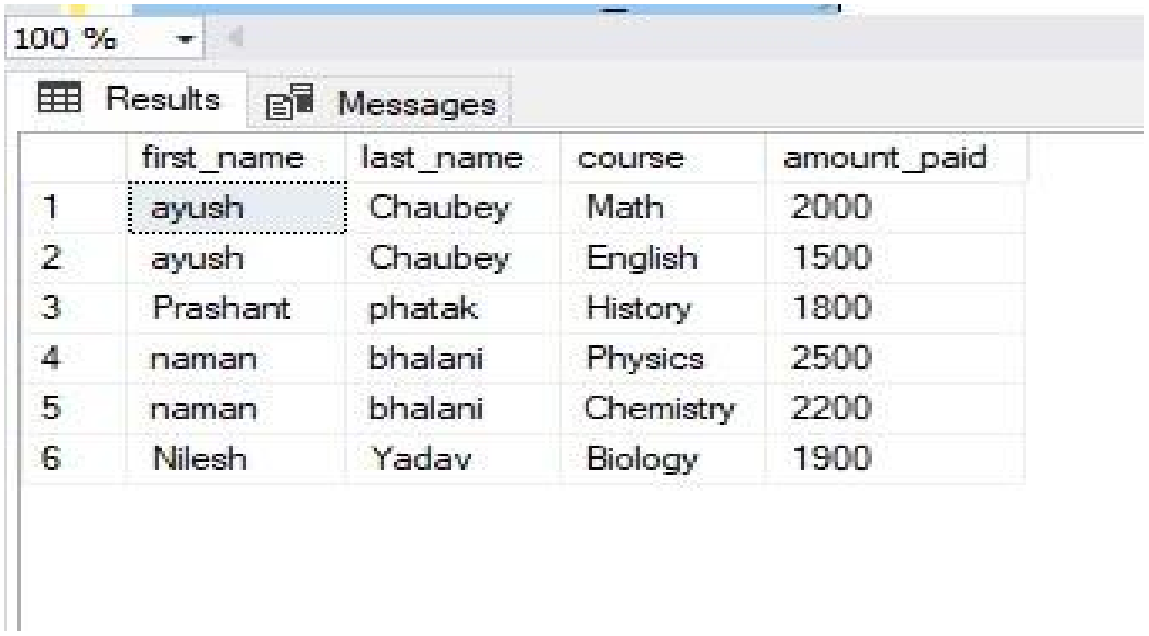
Results | Messages

	addmission_no	course	amount_paid
1	1001	Math	2000
2	1001	English	1500
3	1002	History	1800
4	1003	Physics	2500
5	1003	Chemistry	2200
6	1004	Biology	1900

```
-- create the course_enrolled view
CREATE VIEW course_enrolled
AS
SELECT first_name, last_name, course, amount_paid
FROM Student_details AS S
INNER JOIN fees AS F
```

```
ON S.addmission_no = F.addmission_no;


-- select all rows from the course_enrolled view
SELECT * FROM course_enrolled;
```



100 %

Results | Messages

	first_name	last_name	course	amount_paid
1	ayush	Chaubey	Math	2000
2	ayush	Chaubey	English	1500
3	Prashant	phatak	History	1800
4	naman	bhalani	Physics	2500
5	naman	bhalani	Chemistry	2200
6	Nilesh	Yadav	Biology	1900

TRIGGER:

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.

Advantages of Triggers

1. Triggers help us to enforce data integrity.

2. Triggers help us to validate data before inserted or updated.
3. Triggers help us to keep a log of records.

Disadvantages of Triggers

1. Triggers only allow using extended validations.
2. Triggers are invoked automatically, and their execution is invisible to the user
3. Triggers may increase the overhead of the database server.

Syntax of Trigger

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

Example of Trigger

```
CREATE TABLE Employee
(
  Id INT PRIMARY KEY,
  Name VARCHAR(45),
  Salary INT,
  Gender VARCHAR(12),
  DepartmentId INT
)

INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
```

(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2),
(5,'Eliana', 46000, 'Female', 3)

SELECT * FROM Employee;

Id	Name	Salary	Gender	DepartmentId
1	Steffan	82000	Male	3
2	Amelie	52000	Female	2
3	Antonio	25000	male	1
4	Marco	47000	Male	2
5	Eliana	46000	Female	3
6	Peter	62000	Male	3

also create another table named 'Employee_Audit_Test' to automatically store transaction records of each operation, such as INSERT, UPDATE, or DELETE on the Employee table:

```
CREATE TABLE Employee_Audit_Test
(
Id int IDENTITY,
Audit_Action text
)
```

Now create a trigger that stores transaction records of each insert operation on the Employee table into the Employee_Audit_Test table. Here we are going to create the insert trigger using the below statement:

```
CREATE TRIGGER trInsertEmployee
ON Employee
FOR INSERT
```

```
AS
BEGIN
  Declare @Id int
  SELECT @Id = Id from inserted
  INSERT INTO Employee_Audit_Test
  VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is
added at ' + CAST(Getdate() AS VARCHAR(22)))   END
```

Try to add new record in the table.

```
INSERT INTO Employee VALUES (6,'Peter', 62000, 'Male', 3)
```

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM

Now create another trigger to store transaction records of each delete operation on the Employee table into the Employee_Audit_Test table

```
CREATE TRIGGER trDeleteEmployee
ON Employee
FOR DELETE
AS
BEGIN
  Declare @Id int
  SELECT @Id = Id from deleted
  INSERT INTO Employee_Audit_Test
  VALUES ('An existing employee with Id = ' + CAST(@Id AS
VARCHAR(10)) + ' is deleted at ' + CAST(Getdate() AS VARCHAR(22)))
END
```

After creating a trigger, we will delete a record from the Employee table:

DELETE FROM Employee WHERE Id = 2;

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM
2	An existing employee with Id = 2 is deleted at Mar 25 2021 12:26PM

create database BHAVIN

```
CREATE TABLE Employee_new
(
Id INT PRIMARY KEY,
Name VARCHAR(45),
Salary INT,
Gender VARCHAR(12),
DepartmentId INT
) ;
INSERT INTO Employee_new VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2), (5,'Eliana',
46000, 'Female', 3)
select* from Employee_new;

CREATE TABLE Employee_Audit_Test1
(
Id int IDENTITY,
Audit_Action text
)
CREATE TRIGGER trInsertEmployee
ON Employee_new
FOR INSERT
```

```
AS
BEGIN
Declare @Id int
SELECT @Id = Id from inserted --show entry in different table
INSERT INTO Employee_Audit_Test1
VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is
added at ' +
CAST(Getdate() AS VARCHAR(22)))
END
INSERT INTO Employee_new VALUES (6,'Peter', 62000, 'Male', 3)


Delete from Employee_new

Where id = 3 ;

UPDATE Employee_new
SET Salary = 55000, Gender = 'Male'
WHERE Id = 2;

DROP TRIGGER trInsertEmployee;
DROP TABLE Employee_Audit_Test1;

select * from Employee_Audit_Test1
```

Conclusion: Hence in this way the View and trigger is implemented successfully.