A.Y. 2020-2021
Class: SE-ITA/B, Semester: III

Subject: **Structured Query Lab**

Experiment – 5: Perform joins and views on the chosen system.

Aim: To Perform joins and views on the chosen system.

Objective:
After performing the experiment, the students will be able to formulate and use various join operation to manipulate database and retrieve data
Use views to have a different view of data from the database

Outcome: L303.4: To Write queries in SQL to retrieve any type of information from a database.

Prerequisite: Understanding of various SQL JOIN operations with notations and terminologies along with sample syntax.

Requirements: PC, Oracle 11g/SQL Server 2008 R2, Microsoft Word, Internet

Pre-Experiment Exercise:
Brief Theory :(To be hand written)
Explain what Joins are.
A JOIN clause is used to combine rows from two or more tables, based on a related column between them.A join condition defines the way two tables are related in a query by:

- Specifying the column from each table to be used for the join. A typical join condition specifies a foreign key from one table and its associated key in the other table.
- Specifying a logical operator (for example, = or <>,) to be used in comparing values from the columns.

Following are type of SQL joins
INNER JOIN
LEFT [OUTER] JOIN
RIGHT [OUTER] JOIN
FULL [OUTER] JOIN
CROSS JOIN

Implement examples of:

Inner join

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Left outer join

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Right outer join

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

Full outer join

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Cross join

A cross join returns the Cartesian product of rows from the rowsets in the join. In other words, it will combine each row from the first rowset with each row from the second rowset.

```
SELECT e.EmpName, d.DepName
    FROM employees AS e
         CROSS JOIN departments AS d
    WHERE d.DepName == "Engineering";
```

Explain what Views are.

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 functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Laboratory Exercise

Procedure:

Open SQL server 2008 using below login credentials:
Username: sa, Password: Lab301a
Use existing database created by you or
Construct your own database
Construct tables for any two to three entities from your chosen case study
Insert at least 8 to 10 records for each tables
Execute below queries:

Use INNER JOIN example

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

LEFT OUTER JOIN example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

RIGHT OUTER JOIN example

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

FULL OUTER JOIN example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SELF JOIN Example

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2,
A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

VIEWS
CREATE View
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = "Brazil";

SELECT * FROM [Products Above Average Price];

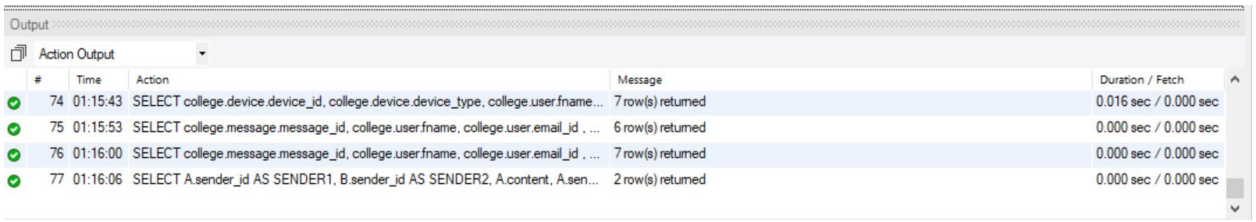
vi)Write/Print output for each query

Result/Observation/Program code: Attach all queries executed code with proper output

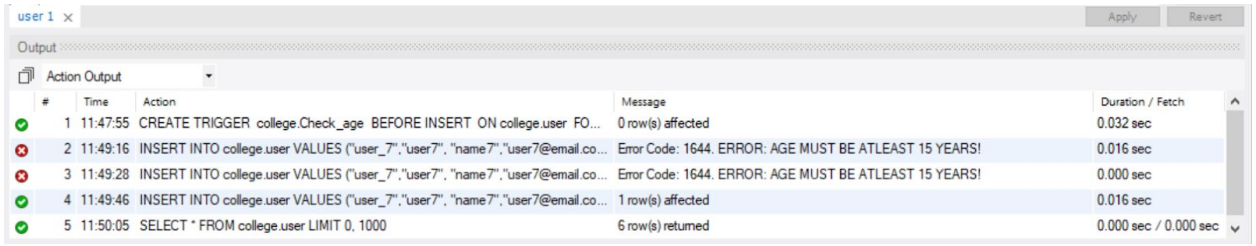
```
SELECT college.device.device_id, college.device.device_type,
college.user.fname, college.user.email_id
FROM college.device
INNER JOIN college.user ON college.device.user_id = college.user.user_name
ORDER BY college.user.fname;

SELECT college.message.message_id, college.user.fname,
college.user.email_id , college.message.reciever_id,
college.message.send_date, college.message.content
FROM college.message
LEFT JOIN college.user ON college.message.sender_id =
college.user.user_name
ORDER BY college.user.fname;

SELECT college.message.message_id, college.user.fname,
college.user.email_id , college.message.reciever_id,
college.message.send_date, college.message.content
FROM college.message
RIGHT JOIN college.user ON college.message.sender_id =
college.user.user_name
ORDER BY college.user.fname;

SELECT A.sender_id AS SENDER1, B.sender_id AS SENDER2, A.content,
A.send_date
FROM College.message A, College.message B
WHERE A.sender_id <> B.sender_id
```

```
AND A.reciever_id = B.reciever_id
ORDER BY A.send_date;

delimiter $$
CREATE TRIGGER  college.Check_age  BEFORE INSERT
ON college.user
FOR EACH ROW
BEGIN
IF NEW.age < 15 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR: AGE MUST BE ATLEAST 15 YEARS!';
END IF;
END; $$
delimiter;

INSERT INTO college.user VALUES ("user_7","user7",
"name7","user7@email.com","user1234",9035709124,true,14,'2006-09-11');
INSERT INTO college.user VALUES ("user_7","user7",
"name7","user7@email.com","user1234",9035709124,true,20,'2000-09-11');

CREATE VIEW ACTIVE_USERS AS
SELECT user_name, email_id
FROM College.user
WHERE activity_stats = 1;

SELECT * FROM ACTIVE_USERS;

CREATE VIEW PC_USERS AS
SELECT device_id, user_id
FROM College.device
WHERE device_type = "PC-Windows";

SELECT * FROM PC_USERS;
```

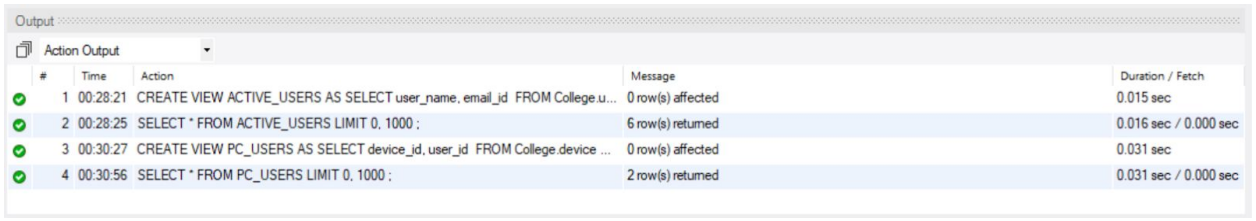
Join Output



Trigger Output



View Output



Post Experimental Exercise-

Questions:

1. What is the general syntax of creating a view?

The general syntax of creating a view is

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

2. Give the syntax for deleting a view. Etc.

```
DROP VIEW view_name;
```

B. Conclusion:

In this experiment we have written SQL scripts to implement Different types of SQL joins like inner join, left outer join, right outer join, self join and Triggers and Views.

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Joins help retrieving data from two or more database tables. The tables are mutually related using primary and foreign keys.

Triggers can be used to implement certain integrity constraints that cannot be specified

using the constraint mechanism of SQL. Triggers are also useful mechanisms for alerting humans or for starting certain tasks automatically when certain conditions are met.

A view in SQL terminology is a single table that is derived from other tables.6 These other tables can be base tables or previously defined views. A view does not necessarily exist in physical form; it is considered to be a virtual table, in contrast to base tables, whose tuples are always physically stored in the database. This limits the possible update operations that can be applied to views, but it does not provide any limitations on querying a view.

C. References:

[1] Elmasri and Navathe, “Fundamentals of Database Systems”, 5th Edition, PEARSON Education.
[2] Korth, Silberchatz, Sudarshan, “Database System Concepts”, 6th Edition, McGraw – Hill
[3] https://www.w3schools.com/sql/sql_default.asp