K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch: A3 Roll No.: 1911034

Experiment / assignment / tutorial No. 5

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Queries based on Joins, and Views

Objective: To be able to use SQL JOIN clause to extract data from 2 (or more) tables, we need a relationship between certain columns in these tables.

Expected Outcome of Experiment:
CO 3 : Use SQL for Relational database creation, maintenance and query processing
CO 4 : Applying normalization to design database

Books/ Journals/ Websites referred:

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : “Database Systems Concept”, 5th Edition , McGraw Hill
4. Elmasri and Navathe,”Fundamentals of database Systems”, 4th Edition,PEARSON Education.

Resources used: Postgresql

Theory

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied. Or JOINS are used to retrieve data from multiple tables. A JOIN is performed whenever two or more tables are joined in a SQL statement.
There are different types of Joins:

- The CROSS JOIN
- The INNER JOIN

- The LEFT OUTER JOIN
- The RIGHT OUTER JOIN
- The FULL OUTER JOIN

A **CROSS JOIN** matches every row of the first table with every row of the second table. If the input tables have x and y columns, respectively, the resulting table will have x+y columns. Because CROSS JOINs have the potential to generate extremely large tables, care must be taken to use them only when appropriate.
Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY CROSS JOIN DEPARTMENT;

A **INNER JOIN** creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows, which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of table1 and table2 are combined into a result row.

Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The **OUTER JOIN** is an extension of the INNER JOIN. SQL standard defines three types of OUTER JOINs: LEFT, RIGHT, and FULL and PostgreSQL supports all of these.

In case of **LEFT OUTER JOIN**, an inner join is performed first. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. Thus, the joined table always has at least one row for each row in T1.

Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The RIGHT OUTER JOIN

First, an inner join is performed. Then, for each row in table T2 that does not satisfy the join condition with any row in table T1, a joined row is added with null values in columns of T1. This is the converse of a left join; the result table will always have a row for each row in T2.
Ex. SELECT EMP_ID, NAME, DEPT FROM COMPANY RIGHT OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

The **FULL OUTER JOIN**

First, an inner join is performed. Then, for each row in table T1 that does not satisfy the join condition with any row in table T2, a joined row is added with null values in columns of T2. In addition, for each row of T2 that does not satisfy the join condition with any row in T1, a joined row with null values in the columns of T1 is added.

SELECT EMP_ID, NAME, DEPT FROM COMPANY FULL OUTER JOIN DEPARTMENT ON COMPANY.ID = DEPARTMENT.EMP_ID;

Views are pseudo-tables. That is, they are not real tables; nevertheless appear as ordinary tables to SELECT. A view can represent a subset of a real table, selecting certain columns or certain rows from an ordinary table. A view can even represent joined tables. Because views are assigned separate permissions, you can use them to restrict table access so that the users see only specific rows or columns of a table.

A view can contain all rows of a table or selected rows from one or more tables. A view can be created from one or many tables, which depends on the written PostgreSQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can only see limited data instead of complete table.
- Summarize data from various tables, which can be used to generate reports.

Since views are not ordinary tables, you may not be able to execute a DELETE, INSERT, or UPDATE statement on a view. However, you can create a RULE to correct this problem of using DELETE, INSERT or UPDATE on a view.

Syntax

```
CREATE [TEMP | TEMPORARY] VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];
```

Ex

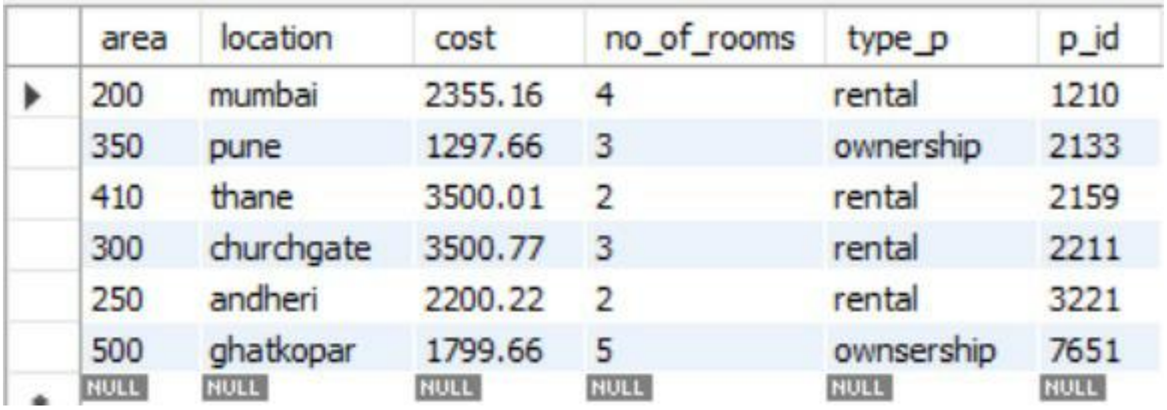
```
CREATE VIEW COMPANY_VIEW AS

SELECT ID, NAME, AGE

FROM  COMPANY;
```

Dropping Views

Syntax: DROP VIEW view_name;

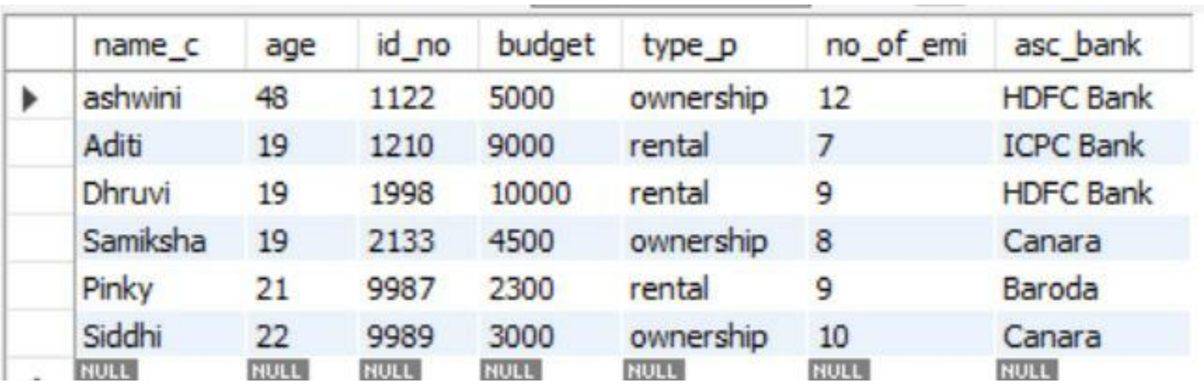
Implementation Screenshots (Problem Statement, Query and Screenshots of Results):

Original property table:-



area	location	cost	no_of_rooms	type_p	p_id
200	mumbai	2355.16	4	rental	1210
350	pune	1297.66	3	ownership	2133
410	thane	3500.01	2	rental	2159
300	churchgate	3500.77	3	rental	2211
250	andheri	2200.22	2	rental	3221
500	ghatkopar	1799.66	5	ownsership	7651
NULL	NULL	NULL	NULL	NULL	NULL

Original Customer table:-

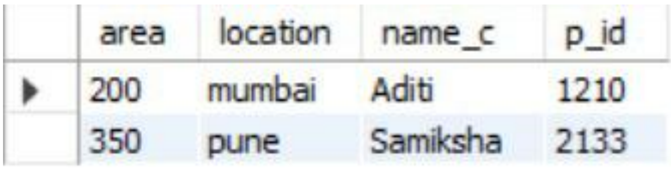


name_c	age	id_no	budget	type_p	no_of_emi	asc_bank
ashwini	48	1122	5000	ownership	12	HDFC Bank
Aditi	19	1210	9000	rental	7	ICPC Bank
Dhruvi	19	1998	10000	rental	9	HDFC Bank
Samiksha	19	2133	4500	ownership	8	Canara
Pinky	21	9987	2300	rental	9	Baroda
Siddhi	22	9989	3000	ownership	10	Canara
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Inner join:-

use propsys;

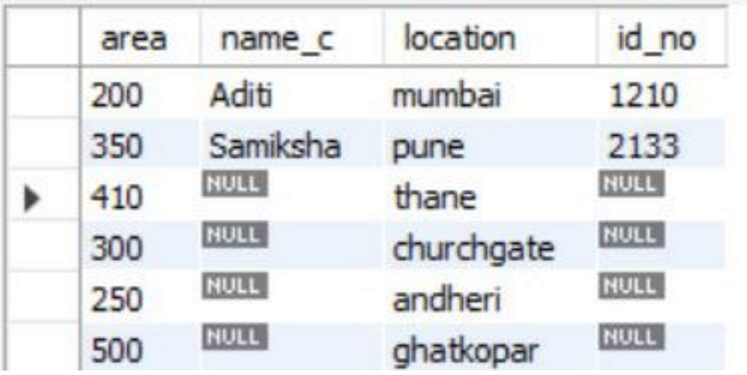
SELECT area, location, name_c, p_id FROM Property INNER JOIN customer ON Customer.id_no = property.p_id;



	area	location	name_c	p_id
▶	200	mumbai	Aditi	1210
	350	pune	Samiksha	2133

Left Outer Join:-

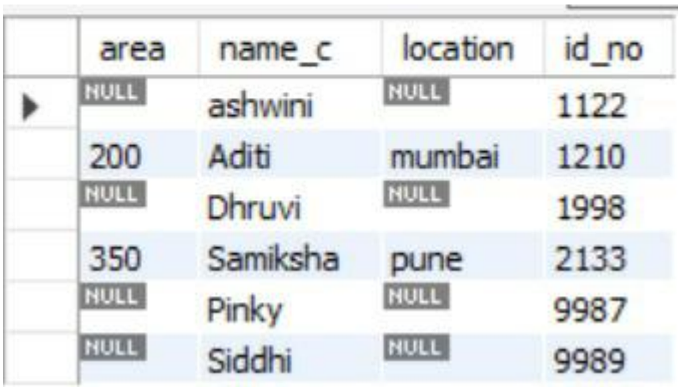
SELECT area, name_c,location,id_no FROM property LEFT OUTER JOIN customer ON customer.id_no = property.p_id;



	area	name_c	location	id_no
	200	Aditi	mumbai	1210
	350	Samiksha	pune	2133
▶	410	NULL	thane	NULL
	300	NULL	churchgate	NULL
	250	NULL	andheri	NULL
	500	NULL	ghatkopar	NULL

Right Outer join:-

SELECT area, name_c,location,id_no FROM property RIGHT OUTER JOIN customer ON customer.id_no = property.p_id;



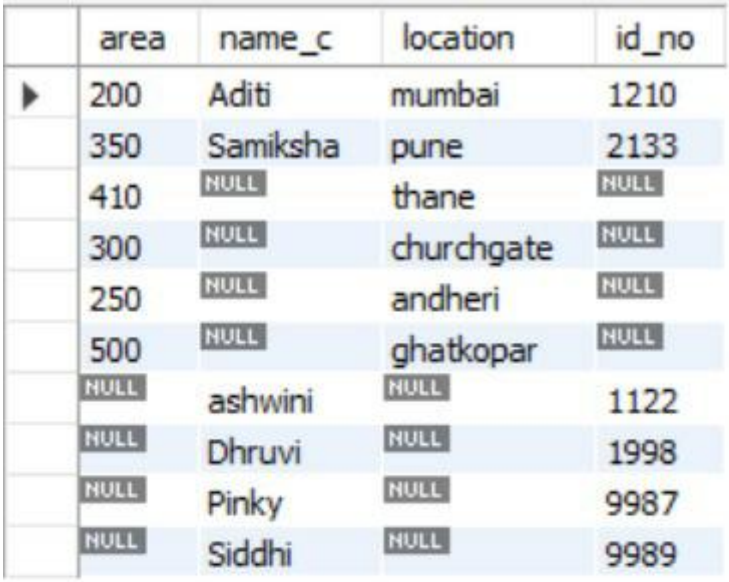
	area	name_c	location	id_no
▶	NULL	ashwini	NULL	1122
	200	Aditi	mumbai	1210
	NULL	Dhruvi	NULL	1998
	350	Samiksha	pune	2133
	NULL	Pinky	NULL	9987
	NULL	Siddhi	NULL	9989

Full Outer join:-

SELECT area, name_c,location,id_no FROM property LEFT OUTER JOIN customer ON customer.id_no = property.p_id

UNION

SELECT area, name_c,location,id_no FROM property RIGHT OUTER JOIN customer ON customer.id_no = property.p_id;

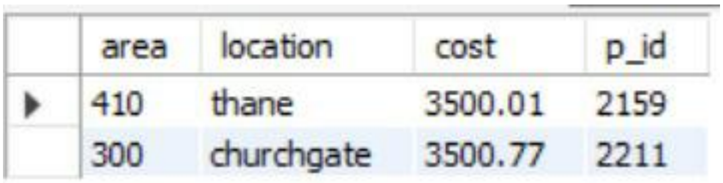


	area	name_c	location	id_no
▶	200	Aditi	mumbai	1210
	350	Samiksha	pune	2133
	410	NULL	thane	NULL
	300	NULL	churchgate	NULL
	250	NULL	andheri	NULL
	500	NULL	ghatkopar	NULL
	NULL	ashwini	NULL	1122
	NULL	Dhruvi	NULL	1998
	NULL	Pinky	NULL	9987
	NULL	Siddhi	NULL	9989

Views:-

create VIEW proprice AS SELECT area , location , cost , p_id FROM Property WHERE cost>2400;

select * from proprice;



	area	location	cost	p_id
▶	410	thane	3500.01	2159
	300	churchgate	3500.77	2211

Conclusion: Join operations were performed and views were created successfully.

Post Lab Questions:

1. **What is a view?**

   a) A view is a special stored procedure executed when certain event occurs

b) A view is a virtual table which results of executing a pre-compiled query
c) A view is a database diagram
d) None of the Mentioned

Answer : b) A view is a virtual table which results of executing a pre-compiled query

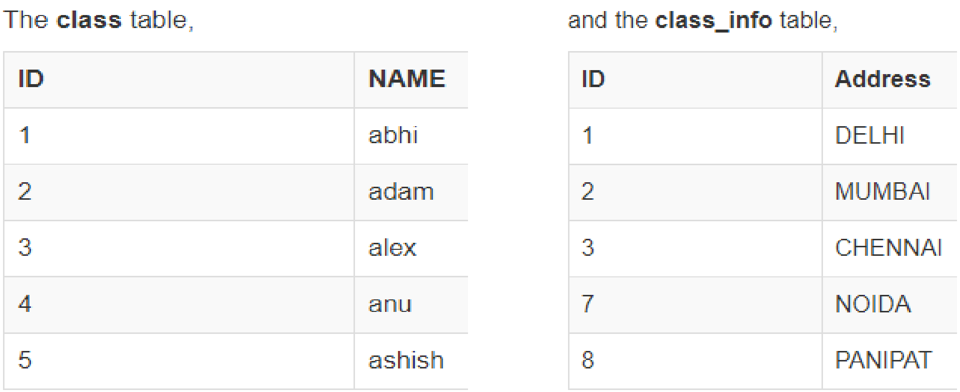
2. **What type of join is needed when you wish to include rows that do not have matching values?**

A. Equi-join
B. Natural join
C. Outer join
D. All of the mentioned

Answer : C. Outer join

3. **Write SQL query including join operator to get following output:**

Input Tables:



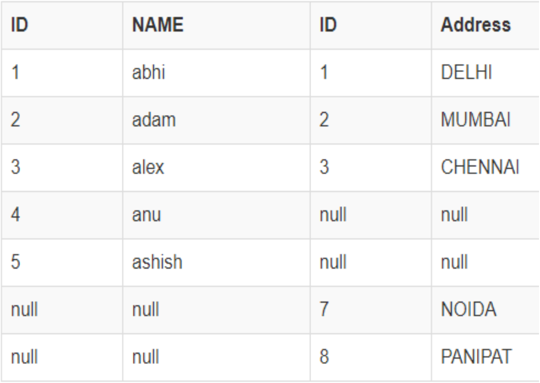
The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Output Table:



ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

```
CREATE database prop;

Use prop;

create table class(ID int, Name varchar(15));

INSERT INTO class(ID,Name)VALUES('1','abhi');

INSERT INTO class(ID,Name)VALUES('2',’adam');

INSERT INTO class(ID,Name)VALUES('3','alex');

INSERT INTO class(ID,Name)VALUES('4','anu');

 INSERT INTO class(ID,Name)VALUES('5','ashish');

select * from class;

create table class_info(Id int, Address varchar(15));

INSERT INTO class_info(Id,Address)VALUES('1','DELHI');

INSERT INTO class_info(Id,Address)VALUES('2','MUMBAI');

INSERT INTO class_info(Id,Address)VALUES('3','CHENNAI');

INSERT INTO class_info(Id,Address)VALUES('7','NOIDA');

INSERT INTO class_info(Id,Address)VALUES('8','PANIPAT');

Select * from class_info;
```

```
 SELECT class.ID,Name,class_info.Id,Address FROM class LEFT OUTER JOIN class_info
ON class.ID = class_info.Id UNION

 SELECT class.ID,Name,class_info.Id,Address FROM class RIGHT OUTER JOIN class_info
ON class.ID = class_info.Id;
```