



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

Batch: A2

Roll No.: 1911027

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, Silberchatz, 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



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

- 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;



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

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;
```



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

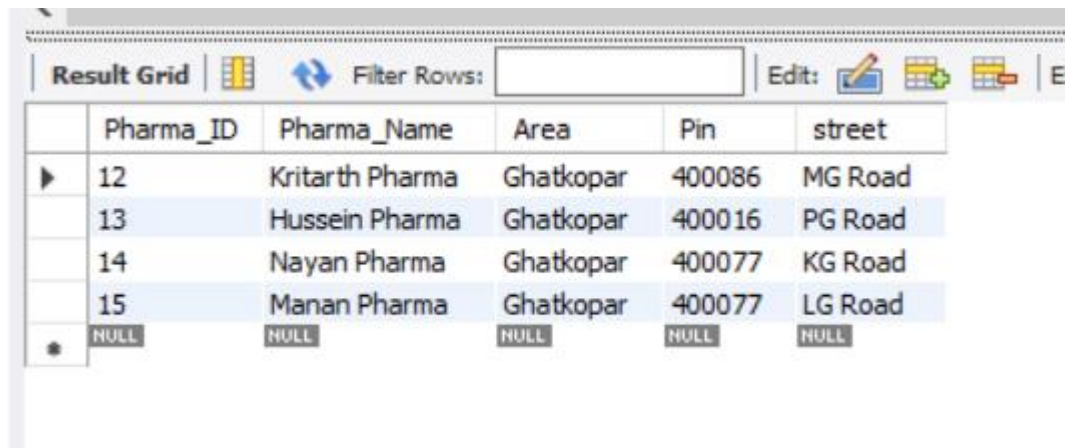
Dropping Views

Syntax: DROP VIEW view_name;

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

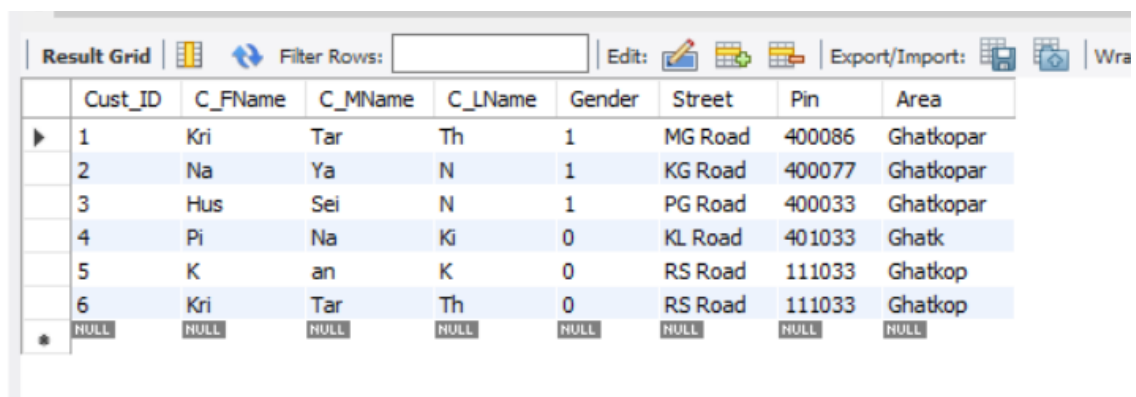
Problem Statement: Online pharmacy management system where user can order medicines from a particular pharmacy by comparing prices from different pharmacies available. And also checking availability of medicines in the pharmacy.

Pharmacy table:



| | Pharma_ID | Pharma_Name | Area | Pin | street |
|---|-----------|-----------------|-----------|--------|---------|
| ▶ | 12 | Kritarth Pharma | Ghatkopar | 400086 | MG Road |
| | 13 | Hussein Pharma | Ghatkopar | 400016 | PG Road |
| | 14 | Nayan Pharma | Ghatkopar | 400077 | KG Road |
| | 15 | Manan Pharma | Ghatkopar | 400077 | LG Road |
| * | NULL | NULL | NULL | NULL | NULL |

Customer table:



| | Cust_ID | C_FName | C_MName | C_LName | Gender | Street | Pin | Area |
|---|---------|---------|---------|---------|--------|---------|--------|-----------|
| ▶ | 1 | Kri | Tar | Th | 1 | MG Road | 400086 | Ghatkopar |
| | 2 | Na | Ya | N | 1 | KG Road | 400077 | Ghatkopar |
| | 3 | Hus | Sei | N | 1 | PG Road | 400033 | Ghatkopar |
| | 4 | Pi | Na | Ki | 0 | KL Road | 401033 | Ghatk |
| | 5 | K | an | K | 0 | RS Road | 111033 | Ghatkop |
| | 6 | Kri | Tar | Th | 0 | RS Road | 111033 | Ghatkop |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |



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

Transactions table:

| Trans_ID | C_ID | P_ID | M_ID | Quantity | Total_Cost | Phar_Name | Pres_ID | D_FName | D_LName |
|----------|------|------|------|----------|------------|------------|---------|----------|----------|
| 164 | 1 | 12 | 1 | 3 | 100 | AbcPhrama | 33 | Nayan | Mandliya |
| 166 | 2 | 13 | 2 | 1 | 200 | XyzPhrama | 34 | Hussein | Motiwala |
| 168 | 3 | 14 | 1 | 1 | 500 | NonePhrama | 35 | Kritarth | Jain |
| 200 | 3 | 14 | 1 | 1 | 5000 | NonePhrama | 35 | Kritarth | Jain |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Medicine table:

| Med_ID | Med_Name | Med_Company | Quantity | Price | Med_Type | Pharma_Name | Exp_Date | Phar_ID | Cus_ID |
|--------|-------------|-------------|----------|-------|------------|-------------|------------|---------|--------|
| 1 | Crocine | intel | 200 | 15 | headache | AbcPharma | 2022-12-12 | 12 | 1 |
| 2 | Paracetamol | AMD | 30 | 100 | Antibiotic | XyzPharma | 2023-12-12 | 13 | 4 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Join operation:

Query: select Cust_ID,C_ID,Trans_ID,C_FName,total_cost from customer join transactions on customer.Cust_id=transactions.C_ID;

Screenshot:

| Cust_ID | C_ID | Trans_ID | C_FName | total_cost |
|---------|------|----------|---------|------------|
| 1 | 1 | 164 | Kri | 100 |
| 2 | 2 | 166 | Na | 200 |
| 3 | 3 | 168 | Hus | 500 |
| 3 | 3 | 200 | Hus | 5000 |



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

Cross Join operation:

Query: select Cust_ID,C_ID,Trans_ID,C_FName,total_cost from customer cross join transactions;

Screenshot:

The screenshot shows a database query result grid with the following columns: Cust_ID, C_ID, Trans_ID, C_FName, and total_cost. The results are a cross join of the customer and transactions tables, resulting in 20 rows. The first row is highlighted with a mouse cursor.

| | Cust_ID | C_ID | Trans_ID | C_FName | total_cost |
|---|---------|------|----------|---------|------------|
| ▶ | 1 | 3 | 200 | Kri | 5000 |
| | 1 | 3 | 168 | Kri | 500 |
| | 1 | 2 | 166 | Kri | 200 |
| | 1 | 1 | 164 | Kri | 100 |
| | 2 | 3 | 200 | Na | 5000 |
| | 2 | 3 | 168 | Na | 500 |
| | 2 | 2 | 166 | Na | 200 |
| | 2 | 1 | 164 | Na | 100 |
| | 3 | 3 | 200 | Hus | 5000 |
| | 3 | 3 | 168 | Hus | 500 |
| | 3 | 2 | 166 | Hus | 200 |
| | 3 | 1 | 164 | Hus | 100 |
| | 4 | 3 | 200 | Pi | 5000 |
| | 4 | 3 | 168 | Pi | 500 |
| | 4 | 2 | 166 | Pi | 200 |
| | 4 | 1 | 164 | Pi | 100 |
| | 5 | 3 | 200 | K | 5000 |
| | 5 | 3 | 168 | K | 500 |
| | 5 | 2 | 166 | K | 200 |
| | 5 | 1 | 164 | K | 100 |
| | 6 | 3 | 200 | Kri | 5000 |
| | 6 | 3 | 168 | Kri | 500 |
| | 6 | 2 | 166 | Kri | 200 |
| | 6 | 1 | 164 | Kri | 100 |

Result 9 x



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

Inner Join operation:

Query: select Cust_ID,Trans_ID,C_FName,total_cost from customer inner join transactions on customer.Cust_ID=transactions.C_ID;

Screenshot:

| | Cust_ID | Trans_ID | C_FName | total_cost |
|---|---------|----------|---------|------------|
| ▶ | 1 | 164 | Kri | 100 |
| | 2 | 166 | Na | 200 |
| | 3 | 168 | Hus | 500 |
| | 3 | 200 | Hus | 5000 |

Left outer Join operation:

Query: select Cust_ID,C_FName,C_ID,Trans_ID,total_cost from customer left outer join transactions on customer.Cust_ID=transactions.C_ID;

Screenshot:

| | Cust_ID | C_FName | C_ID | Trans_ID | total_cost |
|---|---------|---------|------|----------|------------|
| ▶ | 1 | Kri | 1 | 164 | 100 |
| | 2 | Na | 2 | 166 | 200 |
| | 3 | Hus | 3 | 200 | 5000 |
| | 3 | Hus | 3 | 168 | 500 |
| | 4 | Pi | NULL | NULL | NULL |
| | 5 | K | NULL | NULL | NULL |
| | 6 | Kri | NULL | NULL | NULL |

Result 19 x



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

Right outer Join operation:

Query: select * from medicine right outer join pharmacy on Pharma_ID=Phar_ID;

Screenshot:

| Med_ID | Med_Name | Med_Company | Quantity | Price | Med_Type | Pharma_Name | Exp_Date | Phar_ID | Cus_ID | Pharma_ID | Pharma_Name | Area | Pin | street |
|--------|-------------|-------------|----------|-------|------------|-------------|------------|---------|--------|-----------|-----------------|-----------|--------|---------|
| 1 | Crocin | intel | 200 | 15 | headache | AbcPharma | 2022-12-12 | 12 | 1 | 12 | Kritarth Pharma | Ghatkopar | 400086 | MG Road |
| 2 | Paracetamol | AMD | 30 | 100 | Antibiotic | XyzPharma | 2023-12-12 | 13 | 4 | 13 | Hussein Pharma | Ghatkopar | 400016 | PG Road |
| | | | | | | | | | | 14 | Nayan Pharma | Ghatkopar | 400077 | KG Road |

Full outer Join operation:

Query: select * from pharmacy left outer join medicine on Pharma_ID=Phar_ID

UNION

select * from pharmacy right outer join medicine on Pharma_ID=Phar_ID

Screenshot:

| Pharma_ID | Pharma_Name | Area | Pin | street | Med_ID | Med_Name | Med_Company | Quantity | Price | Med_Type | Pharma_Name | Exp_Date | Phar_ID | Cus_ID |
|-----------|-----------------|-----------|--------|---------|--------|-------------|-------------|----------|-------|------------|-------------|------------|---------|--------|
| 12 | Kritarth Pharma | Ghatkopar | 400086 | MG Road | 1 | Crocin | intel | 200 | 15 | headache | AbcPharma | 2022-12-12 | 12 | 1 |
| 13 | Hussein Pharma | Ghatkopar | 400016 | PG Road | 2 | Paracetamol | AMD | 30 | 100 | Antibiotic | XyzPharma | 2023-12-12 | 13 | 4 |
| 14 | Nayan Pharma | Ghatkopar | 400077 | KG Road | | | | | | | | | | |
| 15 | Manan Pharma | Ghatkopar | 400077 | LG Road | | | | | | | | | | |



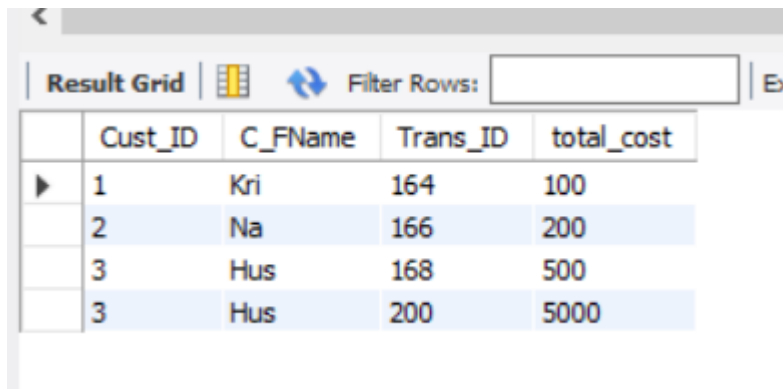
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Views:

Query: create view cus_trans as select Cust_ID,C_FName,Trans_ID,total_cost from customer join transactions on customer.Cust_ID=transactions.C_ID;

select * from cus_trans;

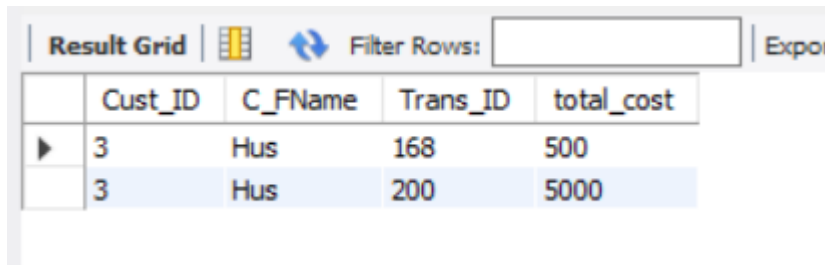
Screenshot:



| | Cust_ID | C_FName | Trans_ID | total_cost |
|---|---------|---------|----------|------------|
| ▶ | 1 | Kri | 164 | 100 |
| | 2 | Na | 166 | 200 |
| | 3 | Hus | 168 | 500 |
| | 3 | Hus | 200 | 5000 |

Query: select * from cus_trans where C_FName="Hus";

Screenshot:



| | Cust_ID | C_FName | Trans_ID | total_cost |
|---|---------|---------|----------|------------|
| ▶ | 3 | Hus | 168 | 500 |
| | 3 | Hus | 200 | 5000 |

Conclusion: By performing this experiment we understood the concept of joins in relational database management systems. Implemented different types of joins on our project database in MySQL. We also understood how views can be created and how we can perform different operations on views.



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

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

ANS) 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

ANS) 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:



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

| 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 |

Query :

select * from class left outer join class_info on class.ID=class_info.ID

UNION

select * from class right outer join class_info on class.ID=class_info.ID;