Experiment No.6
Implement various join operations
Date of Performance:
Date of Submission:

Aim :- Write simple query to implement join operations(equi join, natural join, inner join, outer joins).

Objective :- To apply different types of join to retrieve queries from the database management system.

Theory:

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table matching_column: Column

common to both the tables.

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table. table2: Second table

matching_column: Column common to both the

tables.

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table. table2: Second table

matching_column: Column common to both the

tables.

Implementation:

```
SELECT *
FROM customer
INNER JOIN payement ON customer.Cust_name = payement.payement_id
LIMIT 1000;
SELECT *
FROM customer
LEFT JOIN payement ON customer.Cust_name = payement.payement_id
LIMIT 1000;
SELECT *
FROM customer
RIGHT JOIN payement ON customer.Cust_name = payement.payement_id
LIMIT 1000;
SELECT *
FROM customer
FULL JOIN payement ON customer.Custermor_id = payement.payement_id
LIMIT 1000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Custermor_id	Cust_name	cu_address	phone_no	aadhar_no	DOB	room_id	payement_id	amount	payment_type
1	John Doe	123 Main St	1234567	123456789	1990-01-01	NULL	NULL	NULL	NULL
2	Jane Smith	456 Elm St	9876543	98765432	1985-05-15	NULL	NULL	NULL	NULL
3	Alice Johnson	789 Oak St	5551234	55512345	2000-10-20	NULL	NULL	NULL	NULL
4	Bob Brown	321 Pine St	9998887	99888877	1978-12-30	NULL	NULL	NULL	NULL
5	Emily Wilson	654 Birch St	1112222	11122233	1995-08-25	NULL	NULL	NULL	NULL

Custermor_id	Cust_name	cu_address	phone_no	aadhar_no	DOB	room_id	payement_id	amount	payment_type
NULL	NULL	NULL	NULL	NULL	NULL	NULL	1	500	Credit Card
NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	750	Debit Card
NULL	NULL	NULL	NULL	NULL	NULL	NULL	3	1000	Cash
NULL	NULL	NULL	NULL	NULL	NULL	NULL	4	300	Online Transfer
NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	900	Cheque

Conclusion:

1. Illustrate how to perform natural join for the joining attributes with different names with a suitable example.

   Performing a natural join with joining attributes having different names requires explicitly specifying the join condition. Here's a concise example:

   Example:

   ```
   SELECT *
   FROM Employees
   NATURAL JOIN Departments
   ON Employees.dept_id = Departments.department_id;
   ```

   In this example, Employees and Departments tables have different column names (dept_id and department_id). The ON clause specifies the common columns for the natural join.

2. Illustrate significant differences between natural join, equi-join and inner join.

Differences Between Natural Join, Equi Join, and Inner Join:

Natural Join: Automatically matches columns with the same name but can produce unexpected results.

Equi Join: Specifies join conditions explicitly, allowing joining attributes with different names.

Inner Join: Returns rows that satisfy the join condition specified in the ON clause, providing control over the join condition.