Experiment No. 6

Aim:	Applying SQL Joins (INNER & OUTER) on a created database.
Objective:	To understand & implement various SQL JOIN operations - including INNER JOIN, LEFT JOIN, RIGHT JOIN & FULL OUTER JOIN. - in order to retrieve & analyze data from multiple related tables in a relational database system.

Aim:	Applying SQL joins (INNER & OUTER) on a created Database
Objective:	To understand & implement various SQL JOIN operations - including INNER JOIN, LEFT JOIN, RIGHT JOIN & FULL OUTER JOIN - in order to retrieve & analyze data from multiple related tables in a relational database system.
About the SQL Joins:	SQL JOIN is used to retrieve data from two or more tables based on a related column between them. It allows us to merge rows that have a logical relationship, making it easier to gather information spreaded across different tables.
Types:	i) INNER JOIN ii) LEFT JOIN iii) RIGHT JOIN iv) FULL OUTER JOIN.
I)	INNER JOIN : Returns records that have matching values in both tables. Excludes rows without matches.

Example :

customer_id	customer_name	city
1	Alice	NY
2	Charle	SF
3	Bob	LA

order_id	customer_id	order_amount
101	1	500
102	2	300
103	1	700
104	3	250

Query :

```
SELECT customers.customer_name, Orders.order_id,
Order.order_amount
FROM Customers INNER JOIN Orders ON Customers.
customer_id = Orders.customer_id;
```

output :

customer_name	order_id	order_amount
Alice	101	500
Bob	102	300
Alice	103	700
Charlie	104	250

II) LEFT JOIN (LEFT OUTER JOIN) :

Returns all rows from the left table, & matched rows from the right table.

Non matching rows from the right side return NULL.

Example :

customer_id	customer_name	city
1	Alice	NY
2	Bob	LA
3	Charlie	SF
4	David	TX

order_id	customer_id	order_amount
101	1	500
102	2	300
103	1	700
104	3	250

Query:
```
SELECT Customers.customer_name, Orders.order_id,
Orders.order_amount
FROM Customers LEFT JOIN Orders ON Customers.
customer_id = Orders.customer_id;
```

Output:

customer_name	order_id	order_amount
Alice	101	500
Alice	103	700
Bob	102	300
Charlie	104	250
David	NULL	NULL

III> RIGHT JOIN (RIGHT OUTER JOIN):
Returns all rows from the right table, & the matched rows from the left table.
Non matching rows from the left side return NULL.

Example:

customer_id	customer_name	city
1	Alice	NY
2	Bob	LA
3	Charlie	SF

order_id	customer_id	order_amount
101	1	500
102	2	300
103	1	700
104	3	250

Query:
```
SELECT Customers.customer_name, Orders.order_id,
Orders.order_amount
FROM Customers RIGHT JOIN Orders ON Customers.customer_id
=Orders.customer_id;
```

Output :

customer_name	order_id	order_amount
Alice	101	500
Bob	102	300
Alice	103	700
Charlie	104	950

IV) FULL JOIN (FULL OUTER JOIN) :

Returns all records from both tables with NULL's in places where no a match doesn't exists

Example :

customer_id	customer_name	city
1	Alice	NY
2	Bob	LA
3	Charlie	SF
4	David	TX

order_id	customer_id	order_amount
101	1	500
102	2	300
103	1	700
104	3	250
105	5	150

Query :

```
SELECT Customers.customer_name, Orders.order_id,
Orders.order_amount
FROM Customers FULL JOIN Orders ON Customers.
customer_id = Orders.customer_id ;
```

plus →

Conclusion: This experiment demonstrated the use of various SQL JOIN operations to query data spread across multiple tables. Understanding how each JOIN TYPE work is essential in relational database management system.

1. SQL JOIN

The JOIN operation in SQL is used to combine rows from two or more tables based on a related column. It helps in fetching data that is stored across multiple tables.

Types of JOINs in SQL

1. INNER JOIN (Default JOIN)

- Returns only the records where there is a match in both tables.
- If there is no match, rows from both tables are excluded.

☑ **Example:**
Consider the following tables:

Customers Table

customer_id	customer_name	city
1	Alice	NY
2	Bob	LA
3	Charlie	SF

Orders Table

order_id	customer_id	order_amount
101	1	500
102	2	300
103	1	700
104	3	250

Query (Using INNER JOIN):

```
SELECT Customers.customer_name, Orders.order_id, Orders.order_amount
FROM Customers
INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

Output:

customer_name	order_id	order_amount
Alice	101	500
Alice	103	700
Bob	102	300
Charlie	104	250

---

2. LEFT JOIN (LEFT OUTER JOIN)

- Returns **all rows** from the left table and **only matching** rows from the right table.
- If there is no match, the right table returns NULL.

☑ **Query (Using LEFT JOIN):**

```
SELECT Customers.customer_name, Orders.order_id, Orders.order_amount
FROM Customers
LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

Output:

customer_name	order_id	order_amount
Alice	101	500
Alice	103	700
Bob	102	300
Charlie	104	250
David	NULL	NULL

(If there was a customer in Customers who didn't place an order, their order fields would be NULL.)

---

3. RIGHT JOIN (RIGHT OUTER JOIN)

- Returns **all rows** from the right table and **only matching** rows from the left table.
- If there is no match, the left table returns NULL.

☑ **Query (Using RIGHT JOIN):**

```
SELECT Customers.customer_name, Orders.order_id, Orders.order_amount
FROM Customers
RIGHT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

---

4. FULL JOIN (FULL OUTER JOIN)

- Returns all rows when there is a match in either table.
- If there is no match, NULL values are returned for missing matches.

☑ **Query (Using FULL JOIN):**

```
SELECT Customers.customer_name, Orders.order_id, Orders.order_amount
FROM Customers
FULL JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

---

5. CROSS JOIN

- Returns the Cartesian product (all possible combinations of rows) between two tables.

☑ **Query (Using CROSS JOIN):**

```
SELECT Customers.customer_name, Orders.order_id, Orders.order_amount
FROM Customers
CROSS JOIN Orders;
```

---

2. SQL UNION

The UNION operation in SQL is used to combine the results of two or more SELECT statements. Unlike JOIN, which combines columns from multiple tables, UNION stacks rows from multiple queries into a single result set.

Rules for UNION

- The number of columns in all queries must be the same.
- The data types of corresponding columns must be compatible.
- UNION automatically removes duplicate rows unless UNION ALL is used.

1. UNION (Removes Duplicates)

☑ **Example:**

Employees_DeptA Table

emp_id	emp_name	department
1	John	IT
2	Alice	IT

Employees_DeptB Table

emp_id	emp_name	department
3	Bob	HR
4	Charlie	HR
1	John	IT

☑ **Query (Using UNION):**

```
SELECT emp_id, emp_name, department FROM Employees_DeptA
UNION
SELECT emp_id, emp_name, department FROM Employees_DeptB;
```

Output:

emp_id	emp_name	department
1	John	IT
2	Alice	IT
3	Bob	HR
4	Charlie	HR

Explanation:

- The duplicate "John" from Employees_DeptB is removed.

---

2. UNION ALL (Includes Duplicates)

☑ **Query (Using UNION ALL):**

```
SELECT emp_id, emp_name, department FROM Employees_DeptA
UNION ALL
SELECT emp_id, emp_name, department FROM Employees_DeptB;
```

Output:

emp_id	emp_name	department
1	John	IT
2	Alice	IT
3	Bob	HR
4	Charlie	HR
1	John	IT

Key Differences Between JOIN and UNION

Feature	JOIN	UNION
Purpose	Combines columns from multiple tables based on a condition.	Combines rows from multiple queries into a single result set.
Number of Columns	Can include different columns from different tables.	All queries must have the same number of columns.
Matching Condition	Uses ON condition to match rows.	No matching condition is required.
Duplicate Handling	Keeps all matching rows.	UNION removes duplicates, UNION ALL keeps them.
Example Use Case	Fetching related data from multiple tables (e.g., customers & orders).	Merging similar datasets (e.g., employees from different departments).

Output :

customer_name	order_id	order_amount
Alice	101	500
Bob	102	300
Alice	103	700
Charlie	104	250
David	NULL	NULL
NULL	105	150

Procedure :

i) Create two tables (eg customers & Orders) with a common key (customer_id)

ii) Insert sample records for each table.

iii) Execute the SQL queries for INNER JOIN, LEFT JOIN, RIGHT JOIN, & FULL OUTER JOIN.

iv) Observe & records the results.

v) Analyze how different types of JOIN's affect the result set.

Conclusion : This experiment demonstrated the use of various SQL JOIN operations to query data spread across multiple tables. Understanding how each JOIN TYPE works is essential in relational databases management system.

A+ Rubrics
24/4/25