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:

INNER JOIN:

```
120     -- Inner Join
121 •   SELECT *
122     FROM customer
123     INNER JOIN account ON customer.customer id = account.customer id;
```

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

customer_id	fname	lname	phoneno	email	passworc	address	account_id	customer	type	cards	balance
3	Bob	Johnson	56667...	bob.johnso...	passw...	789 Oak St	103	3	Savings	Debit Card	3000
2	Alice	Smith	87654...	alice.smith...	passw...	456 Elm St	102	2	Checking	Credit Card	7000
1	John	Doe	12345...	john.doe@...	passw...	123 Main St	101	1	Savings	Debit Card	5000

LEFT JOIN:

```
133     -- Left Join
134 •   SELECT *
135     FROM customer
136     LEFT JOIN account ON customer.customer id = account.customer id;
```

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

customer_id	fname	lname	phoneno	email	passworc	address	account_id	customer	type	cards	balance
1	John	Doe	12345...	john.doe@...	passw...	123 Main St	101	1	Savings	Debit Card	5000
2	Alice	Smith	87654...	alice.smith...	passw...	456 Elm St	102	2	Checking	Credit Card	7000
3	Bob	Johnson	56667...	bob.johnso...	passw...	789 Oak St	103	3	Savings	Debit Card	3000
4	Emily	Johnson	51234...	emily.johns...	secure...	789 Oak St	NULL	NULL	NULL	NULL	NULL
5	Michael	Brown	98765...	michael.bro...	strong...	321 Pine St	NULL	NULL	NULL	NULL	NULL

RIGHT JOIN:

```
145     -- Right Join
146 •   SELECT *
147     FROM customer
148     RIGHT JOIN account ON customer.customer id = account.customer id;
```

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

customer_id	fname	lname	phoneno	email	passworc	address	account_id	customer	type	cards	balance
1	John	Doe	12345...	john.doe@...	passw...	123 Main St	101	1	Savings	Debit Card	5000
2	Alice	Smith	87654...	alice.smith...	passw...	456 Elm St	102	2	Checking	Credit Card	7000
3	Bob	Johnson	56667...	bob.johnso...	passw...	789 Oak St	103	3	Savings	Debit Card	3000

Conclusion:

Concluding the various join operations on a Social Media Management System (SMMS) database involves summarizing the effectiveness and implications of using different types of joins to retrieve data from multiple tables. Here's a possible conclusion:

In the realm of Social Media Management Systems, where data is distributed across multiple tables, the use of various join operations plays a pivotal role in amalgamating relevant information and providing comprehensive insights. Through our exploration of different join types—such as INNER JOIN, LEFT JOIN, and RIGHT JOIN—we've gained valuable insights into their functionality and applicability within the context of our SMMS database.

INNER JOINs have proven instrumental in retrieving data where there are matching records between tables, facilitating the extraction of cohesive datasets that meet specific criteria. This join type ensures that only the intersecting records are included in the result set, thereby focusing on commonalities shared across tables.

LEFT JOINs, on the other hand, have showcased their utility in scenarios where we aim to retrieve all records from the left table (the "first" table in the join) and matching records from the right table. This join type enables us to preserve all records from the left table, even if there are no corresponding matches in the right table, thereby providing a more inclusive view of the data.

Similarly, RIGHT JOINs offer a complementary approach by ensuring that all records from the right table are retained, with matching records from the left table. While less commonly used compared to LEFT JOINs, RIGHT JOINs are valuable in situations where the focus is on preserving all records from the right table, regardless of matching criteria.

Through these join operations, we've been able to harness the power of relational databases to consolidate disparate data elements and construct meaningful datasets that fuel analytics, decision-making, and user engagement strategies within our SMMS. By leveraging the appropriate join types based on our data retrieval requirements, we ensure accuracy, completeness, and relevance in the insights derived from our social media management endeavors.