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:

1. Illustrate how to perform natural join for the joining attributes with different names with a suitable example.
   Natural join is a type of join operation in relational databases that combines rows from two tables based on columns with the same name and datatype. When performing a natural join, the database system automatically matches and combines rows with identical values in the specified columns.
   Example:
   WE have two tables: customer and account. The customer table contains information about customers, and the account table contains information about their accounts.

   customer Table:
   | customer_id | fname | lname | address |

   account Table:
   | account_id | customer_id | balance |

Here, the customer table has a column named customer_id, and the account table also has a column named customer_id. These columns represent the same concept but have different names.
To perform a natural join on these tables, the database system will automatically match and combine rows with the same values in the customer_id column from the customer table and the customer_id column from the account table.
The result of the natural join will include columns from both tables, with the matching rows combined into a single row.

Result of Natural Join:
| customer_id | fname | lname | address | account_id | balance |

2. Illustrate significant differences between natural join equi join and inner join.
   Natural Join:
   Automatically matches columns with the same name in the joined tables.
   Combines rows from two tables based on columns with the same name and datatype.
   Eliminates duplicate columns that appear in the joined tables.
   Does not require specifying the join condition explicitly.
   May produce unexpected results if there are columns with the same name but different meanings in the joined tables.
   Equi Join:
   Requires explicitly specifying the join condition using equality comparisons (=) between columns in the joined tables.
   Combines rows from two tables based on matching values in the specified columns.
   Allows joining tables on columns with different names or datatypes.
   Can be used to perform natural joins by explicitly specifying the join condition on columns with the same name.
   Inner Join:
   Returns only the rows that have matching values in both tables based on the specified join condition.
   Requires explicitly specifying the join condition, which can include equality comparisons (=) or other comparison operators.
   Can be considered a more general form of equi join, as it allows specifying any join condition, not just equality comparisons.
   Can be used to perform natural joins by explicitly specifying the join condition on columns with the same name.