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;

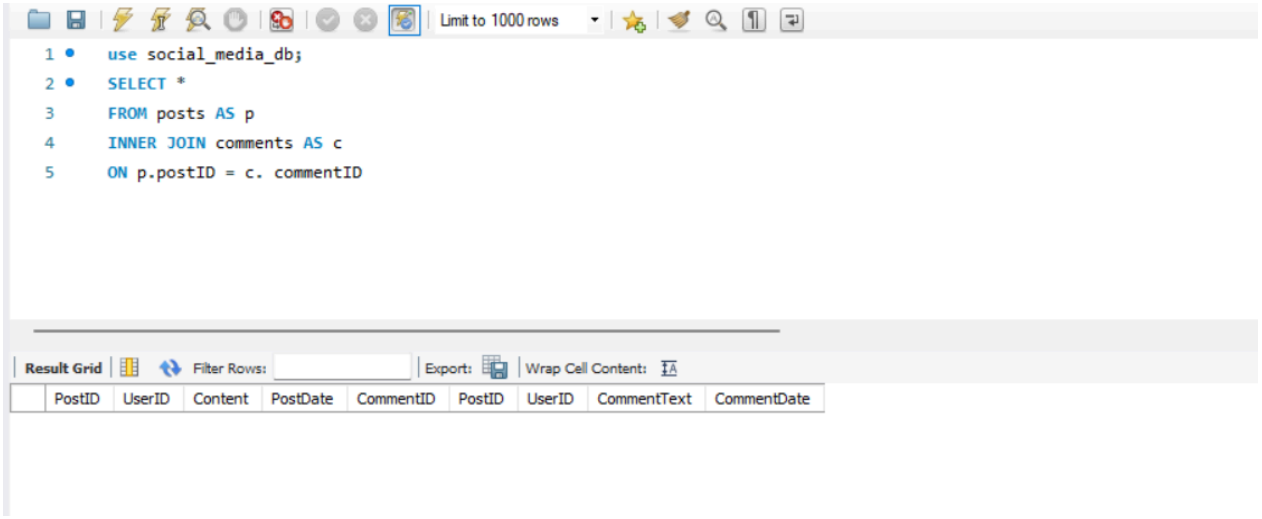
table1: First table.

table2: Second table

matching_column: Column common to both the tables.

Implementation

INNER JOIN:-

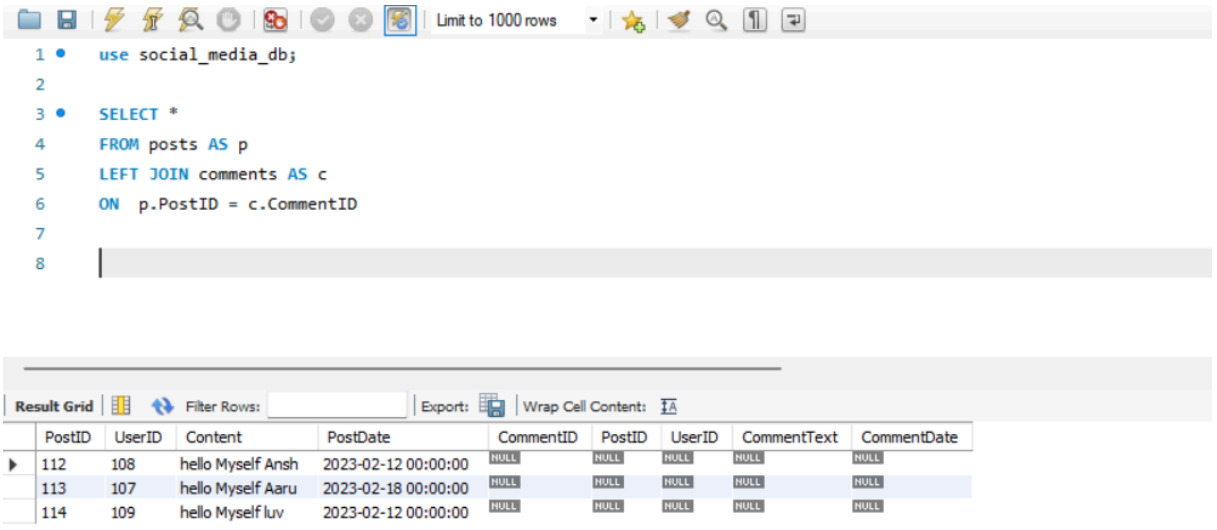


```
use social_media_db;
SELECT *
FROM posts AS p
INNER JOIN comments AS c
ON p.postID = c. commentID
```

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

PostID	UserID	Content	PostDate	CommentID	PostID	UserID	CommentText	CommentDate

LEFT JOIN:-



```
use social_media_db;

SELECT *
FROM posts AS p
LEFT JOIN comments AS c
ON  p.PostID = c.CommentID
```

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

PostID	UserID	Content	PostDate	CommentID	PostID	UserID	CommentText	CommentDate
112	108	hello Myself Ansh	2023-02-12 00:00:00	NULL	NULL	NULL	NULL	NULL
113	107	hello Myself Aaru	2023-02-18 00:00:00	NULL	NULL	NULL	NULL	NULL
114	109	hello Myself luv	2023-02-12 00:00:00	NULL	NULL	NULL	NULL	NULL

RIGHT JOIN:-

```
1 •  use social_media_db;
2
3 •  SELECT *
4    FROM posts AS p
5    RIGHT JOIN comments AS c
6    ON  p.PostID = c.CommentID
7
8
```

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

PostID	UserID	Content	PostDate	CommentID	PostID	UserID	CommentText	CommentDate
NULL	NULL	NULL	NULL	102	113	107	hello Myself Aaru	2023-02-18 00:00:00
NULL	NULL	NULL	NULL	103	114	109	hello Myself luv	2023-02-12 00:00:00
NULL	NULL	NULL	NULL	106	113	107	hello Myself Aaru	2023-02-18 00:00:00

Conclusion:

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

Certainly! In a natural join, columns with the same name in the tables being joined are used as the join criteria. However, when the joining attributes have different names, you can still perform a natural join by explicitly specifying the column names using the USING clause. Here's an example:

Consider two tables: `Employees` and `Departments`:

Employees:

| EmployeeID | Name | DepartmentID |

|---|---|---|

| 1 | Alice | 101 |

| 2 | Bob | 102 |

| 3 | Charlie | 101 |

| 4 | David | 103

| Departments:

| DeptID | DeptName |

|---|---|

| 101 | IT |

| 102 | HR |

| 103 | Finance |

To perform a natural join on these tables, even though the joining attribute names are different (`DepartmentID` in `Employees` and `DeptID` in `Departments`), you can use the `USING` clause to specify the column to join on. Here's the SQL query:

SELECT * FROM Employees

NATURAL JOIN Departments USING (DepartmentID);

| EmployeeID | Name | DepartmentID | DeptName |

|---|---|---|---|

| 1 | Alice | 101 | IT |

| 2 | Bob | 102 | HR |

| 3 | Charlie | 101 | IT |

| 4 | David | 103 | Finance |

In this example, the `USING` clause explicitly specifies that the join should be performed on the `DepartmentID` column in `Employees` and the `DeptID` column in `Departments`, allowing the natural join to occur despite the different column names.

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

Certainly! Here are the significant differences between natural join, equi join, and inner join: Natural Join:
Automatically joins tables based on columns with the same name.
Eliminates duplicate columns from the result set.
It's a type of inner join but uses the common columns as the join condition implicitly.

Equi Join:
Joins tables based on a specified equality condition between columns.
Uses the = operator to match values in columns from both tables.
It's a specific type of inner join where the condition is explicitly defined.

Inner Join:
Joins tables based on a specified condition.
The condition can be any logical expression, not just equality.
Returns only the rows where the condition evaluates to true.

Can be performed using different join conditions like equality (=), inequality (<, >), or other logical conditions (AND, OR, etc.).