Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL402**	**Course Name:**	**Database Management System Lab**

Name of Student:	**Shruti Gauchandra**
Roll No. :	**16**
Experiment No.:	**6**
Title of the Experiment:	**Implement various joins and set operations.**
Date of Performance:	**06/02/25**
Date of Submission:	**13/02/25**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Neha Raut

Signature :

Date:

Experiment No 6

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:

For INNER JOIN

Code:

```
SELECT
C.Customer_ID,
C.F_name,
C.L_Name,
C.Email_id,
C.Mobile_no,
C.Gender,
T.Ticket_ID,
T.Movie_Name,
T.Ticket_Price,
T.Show_Date,
TH.t_name AS Theatre_Name,
TH.t_location AS Theatre_Location
FROM Customer C
INNER JOIN Ticket T ON C.Customer_ID = T.Customer_ID
INNER JOIN Theatre TH ON T.t_id = TH.t_id;
```

Output:

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

Customer_ID	F_name	L_Name	Email_id	Mobile_no	Gender	Ticket_ID	Movie_Name	Ticket_Pric
1	John	Doe	johndoe@example.com	9876543210	Male	1	Inception	250.00
2	Jane	Smith	janesmith@example.com	9123456789	Female	2	Avengers	300.00
3	Michael	Johnson	michaelj@example.com	9234567890	Male	3	Interstellar	275.00
4	Emma	Williams	emmaw@example.com	9345678901	Female	4	Titanic	200.00
5	Daniel	Brown	danielb@example.com	9456789012	Male	5	Joker	350.00

For LEFT JOIN

Code:

```
SELECT
C.Customer_ID,
C.F_name,
```

C.L_Name,

C.Email_id,

C.Mobile_no,

C.Gender,

T.Ticket_ID,

T.Movie_Name,

T.Ticket_Price,

T.Show_Date,

TH.t_name AS Theatre_Name,

TH.t_location AS Theatre_Location

FROM Customer C

LEFT JOIN Ticket T ON C.Customer_ID = T.Customer_ID

LEFT JOIN Theatre TH ON T.t_id = TH.t_id;

Output:

Customer_ID	F_name	L_Name	Email_id	Mobile_no	Gender	Ticket_ID	Movie_Name	Ticket_
2	Jane	Smith	janesmith@example.com	9123456789	Female	2	Avengers	300.00
3	Michael	Johnson	michaelj@example.com	9234567890	Male	3	Interstellar	275.00
4	Emma	Williams	emmaw@example.com	9345678901	Female	4	Titanic	200.00
5	Daniel	Brown	danielb@example.com	9456789012	Male	5	Joker	350.00
6	Olivia	Davis	oliviad@example.com	9567890123	Female	NULL	NULL	NULL
7	Liam	Miller	liamm@example.com	9678901234	Male	NULL	NULL	NULL
8	Sophia	Wilson	sophiaw@example.com	9789012345	Female	NULL	NULL	NULL
9	James	Anderson	jamesa@example.com	9890123456	Male	NULL	NULL	NULL
10	Isabella	Thomas	isabellat@example.com	9901234567	Female	NULL	NULL	NULL

For RIGHT JOIN
Code:

SELECT

C.Customer_ID,

C.F_name,

C.L_Name,

C.Email_id,

C.Mobile_no,

C.Gender,

T.Ticket_ID,

T.Movie_Name,

T.Ticket_Price,

T.Show_Date,

TH.t_name AS Theatre_Name,

TH.t_location AS Theatre_Location

FROM Customer C

RIGHT JOIN Ticket T ON C.Customer_ID = T.Customer_ID

RIGHT JOIN Theatre TH ON T.t_id = TH.t_id;

Output:

Customer_ID	F_name	L_Name	Email_id	Mobile_no	Gender	Ticket_ID	Movie_Name	Ticket_
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	Jane	Smith	janesmith@example.com	9123456789	Female	2	Avengers	300.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	Michael	Johnson	michaelj@example.com	9234567890	Male	3	Interstellar	275.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	Emma	Williams	emmaw@example.com	9345678901	Female	4	Titanic	200.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	Daniel	Brown	danielb@example.com	9456789012	Male	5	Joker	350.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

For FULL JOIN

Code:

SELECT

C.Customer_ID,

C.F_name,

C.L_Name,

C.Email_id,

C.Mobile_no,

C.Gender,

T.Ticket_ID,

T.Movie_Name,

T.Ticket_Price,

T.Show_Date,

TH.t_name AS Theatre_Name,

```
TH.t_location AS Theatre_Location
FROM Customer C
LEFT JOIN Ticket T ON C.Customer_ID = T.Customer_ID
LEFT JOIN Theatre TH ON T.t_id = TH.t_id;
UNION
SELECT
C.Customer_ID,
C.F_name,
C.L_Name,
C.Email_id,
C.Mobile_no,
C.Gender,
T.Ticket_ID,
T.Movie_Name,
T.Ticket_Price,
T.Show_Date,
TH.t_name AS Theatre_Name,
TH.t_location AS Theatre_Location
FROM Customer C
RIGHT JOIN Ticket T ON C.Customer_ID = T.Customer_ID
RIGHT JOIN Theatre TH ON T.t_id = TH.t_id;
```

Output:

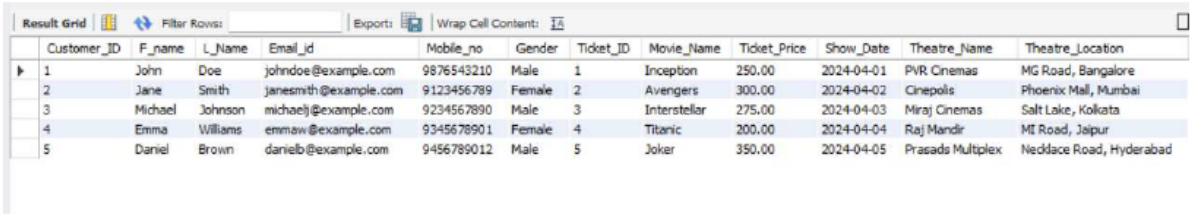
Customer_ID	F_name	L_Name	Email_id	Mobile_no	Gender	Ticket_ID	Movie_Name	Ticket_Price	Show_Date	Theatre_Name
1	John	Doe	johndoe@example.com	9876543210	Male	1	Inception	250.00	2024-04-01	PVR Cinemas
2	Jane	Smith	janesmith@example.com	9123456789	Female	2	Avengers	300.00	2024-04-02	Cinepolis
3	Michael	Johnson	michaelj@example.com	9234567890	Male	3	Interstellar	275.00	2024-04-03	Miraj Cinemas
4	Emma	Williams	emmaw@example.com	9345678901	Female	4	Titanic	200.00	2024-04-04	Raj Mandir
5	Daniel	Brown	danielb@example.com	9456789012	Male	5	Joker	350.00	2024-04-05	Prasads Multiplex
6	Olivia	Davis	oliviad@example.com	9567890123	Female	NULL	NULL	NULL	NULL	NULL
7	Liam	Miller	liamm@example.com	9678901234	Male	NULL	NULL	NULL	NULL	NULL
8	Sophia	Wilson	sophiaw@example.com	9789012345	Female	NULL	NULL	NULL	NULL	NULL
9	James	Anderson	jamesa@example.com	9890123456	Male	NULL	NULL	NULL	NULL	NULL
10	Isabella	Thomas	isabellat@example.com	9901234567	Female	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	INOX Movies
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Carnival Cinemas
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Galaxy Cinema
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Sathyam Cinemas
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Lulu PVR

For EQUI JOIN

Code:

```
SELECT
C.Customer_ID,
C.F_name,
C.L_Name,
C.Email_id,
C.Mobile_no,
C.Gender,
T.Ticket_ID,
T.Movie_Name,
T.Ticket_Price,
T.Show_Date,
TH.t_name AS Theatre_Name,
TH.t_location AS Theatre_Location
FROM Customer C
JOIN Ticket T ON C.Customer_ID = T.Customer_ID
JOIN Theatre TH ON T.t_id = TH.t_id;
```

Output:



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

Customer_ID	F_name	L_Name	Email_id	Mobile_no	Gender	Ticket_ID	Movie_Name	Ticket_Price	Show_Date	Theatre_Name	Theatre_Location
1	John	Doe	johndoe@example.com	9876543210	Male	1	Inception	250.00	2024-04-01	PVR Cinemas	MG Road, Bangalore
2	Jane	Smith	janesmith@example.com	9123456789	Female	2	Avengers	300.00	2024-04-02	Cinepolis	Phoenix Mall, Mumbai
3	Michael	Johnson	michaelj@example.com	9234567890	Male	3	Interstellar	275.00	2024-04-03	Miraj Cinemas	Salt Lake, Kolkata
4	Emma	Williams	emmaw@example.com	9345678901	Female	4	Titanic	200.00	2024-04-04	Raj Mandir	MI Road, Jaipur
5	Daniel	Brown	danielb@example.com	9456789012	Male	5	Joker	350.00	2024-04-05	Prasads Multiplex	Necklace Road, Hyderabad

Conclusion:

In this experiment, we implemented various joins (like INNER, LEFT, RIGHT, FULL) and set operations (such as UNION, INTERSECT, and MINUS). This helped us understand how to combine data from multiple tables and perform advanced queries, enhancing our ability to retrieve meaningful information from relational databases.

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

Ans. **A Natural Join automatically joins tables based on columns with the same name. However, when the joining attributes have different names, we need to use aliases and explicit conditions in the ON clause. Since MySQL does not directly support NATURAL JOIN for differently named columns, we manually specify the condition using JOIN ... ON.**

For example, suppose we have a Customer table where the primary key is Customer_ID, and a Booking table where the foreign key is named Cust_ID. Since the column names are different, we cannot use a direct NATURAL JOIN but instead perform an INNER JOIN with an explicit condition.

Example:

```
CREATE TABLE Customer (
   Customer_ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Email VARCHAR(100)
);
CREATE TABLE Booking (
   Booking_ID INT PRIMARY KEY,
   Cust_ID INT,
   Movie_Name VARCHAR(100),
   Ticket_Price DECIMAL(10,2)
);
FOREIGN KEY (Cust_ID) REFERENCES Customer(Customer_ID);
INSERT INTO Customer (Customer_ID, Name, Email) VALUES
(1, 'John Doe', 'johndoe@example.com'),
(2, 'Jane Smith', 'janesmith@example.com');
INSERT INTO Booking (Booking_ID, Cust_ID, Movie_Name, Ticket_Price) VALUES
(101, 1, 'Inception', 250.00),
(102, 2, 'Titanic', 200.00);
SELECT C.Customer_ID, C.Name, C.Email, B.Booking_ID, B.Movie_Name, B.Ticket_Price
FROM Customer C
JOIN Booking B ON C.Customer_ID = B.Cust_ID;
```

B) Illustrate significant differences between natural join equi join and inner join.

Ans. A Natural Join automatically joins tables based on common column names without requiring an explicit ON condition. It eliminates duplicate columns in the result set, making it more concise, but it can only be used if the tables share at least one column with the same name. If the column names do not match, a natural join cannot be performed.

An Equi Join, on the other hand, explicitly matches rows from two tables using the equality (=) operator in the ON clause. Unlike a natural join, it does not remove duplicate columns and provides more control by allowing joins even when the column names are different. Equi joins can be applied as either Inner Joins or Outer Joins (Left or Right).

An Inner Join retrieves only the records that have matching values in both tables based on a specified condition. It is functionally similar to an equi join but allows additional conditions using operators other than = such as <, >, or BETWEEN. Unlike a natural join, an inner join does not automatically match columns; instead, it requires explicitly specifying the condition for joining.

The key differences among them are that a natural join relies on common column names and removes duplicates, an equi join explicitly specifies the join condition but retains duplicate columns, and an inner join provides even more flexibility by allowing various conditions beyond simple equality.