Vidyavardhini's College of Engineering and Technology, Vasai
Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL402**	**Course Name:**	**DBMS Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**6**
Title of the Experiment:	**Implement various joins and set operations**
Date of Performance:	**12/03/2025**
Date of Submission:	**19/03/2025**

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:

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.

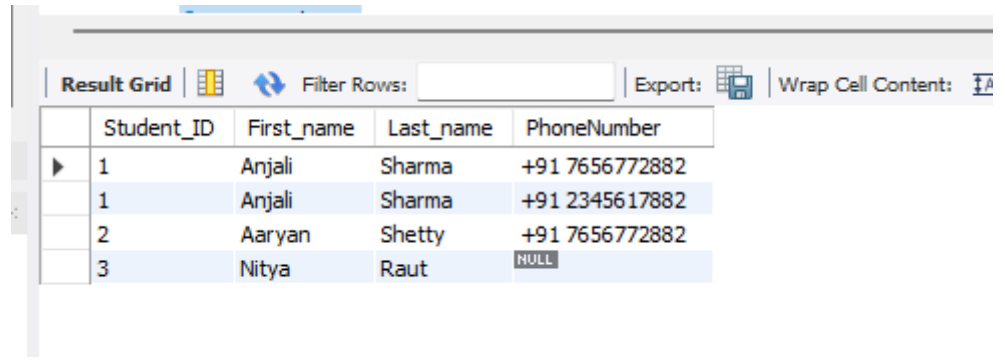
table2: Second table

matching_column: Column common to both the tables.

Implementation:

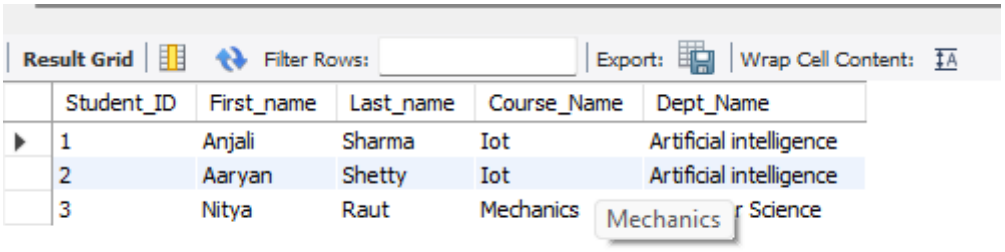
```
select Student.Student_ID, Student.First_name, Student.Last_name,
Student_Marks.Marks_Obtained, Student_marks.Grade

from Student

join Student_Marks on Student.Student_ID = Student_Marks.Student_ID;


select Student.Student_ID, Student.First_name, Student.Last_name, Course.Course_Id,
Course.Course_Name

from Student

natural join Course;


select Student.Student_ID, Student.First_name, Student.Last_name,
Student_PhoneNumber.PhoneNumber

from Student

inner join Student_PhoneNumber on Student.Student_ID =
Student_PhoneNumber.Student_ID;


select Student.Student_ID, Student.First_name, Student.Last_name,
Student_PhoneNumber.PhoneNumber

from Student

left join Student_PhoneNumber on Student.Student_ID =
Student_PhoneNumber.Student_ID;


select Student.Student_ID, Student.First_name, Student.Last_name,
Student_PhoneNumber.PhoneNumber

from Student

right join Student_PhoneNumber on Student.Student_ID =
Student_PhoneNumber.Student_ID;


select Student.Student_ID, Student.First_name, Student.Last_name,

    Course.Course_Name, Department.Dept_Name

from Student

left outer join Course on Student.Course_Id = Course.Course_Id
```

```
left outer join Department on Course.Dept_ID = Department.Dept_ID;

select Student.Student_ID, Student.First_name, Student.Last_name,
Student_PhoneNumber.PhoneNumber

from Student

left join Student_PhoneNumber on Student.Student_ID = Student_PhoneNumber.Student_ID

union

select Student.Student_ID, Student.First_name, Student.Last_name,
Student_PhoneNumber.PhoneNumber

from Student

right join Student_PhoneNumber on Student.Student_ID =
Student_PhoneNumber.Student_ID;
```



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

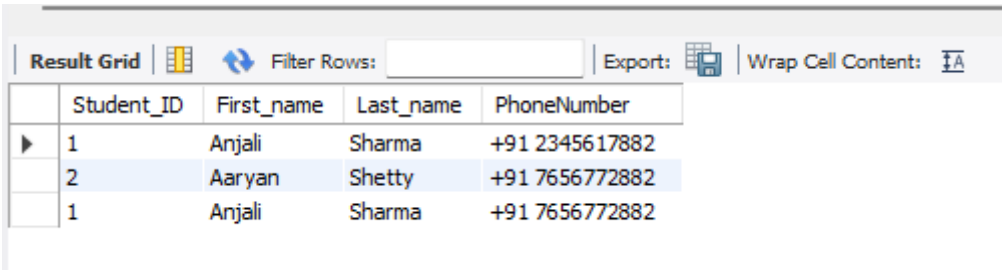
Student_ID	First_name	Last_name	PhoneNumber
1	Anjali	Sharma	+91 7656772882
1	Anjali	Sharma	+91 2345617882
2	Aaryan	Shetty	+91 7656772882
3	Nitya	Raut	NULL



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

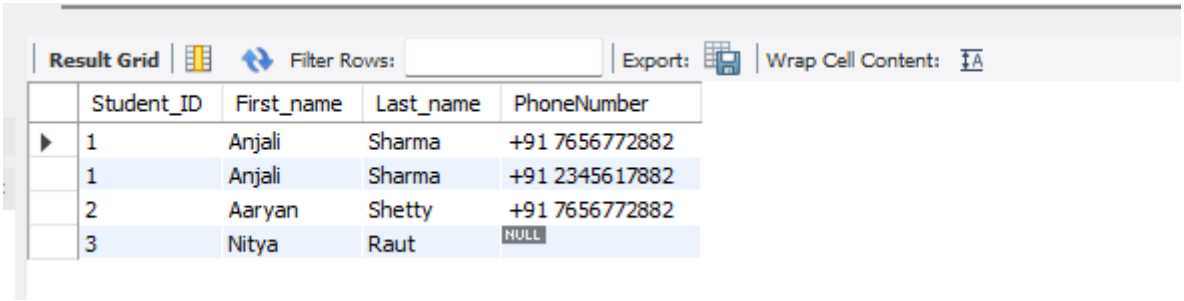
Student_ID	First_name	Last_name	Course_Name	Dept_Name
1	Anjali	Sharma	Iot	Artificial intelligence
2	Aaryan	Shetty	Iot	Artificial intelligence
3	Nitya	Raut	Mechanics	r Science

Mechanics



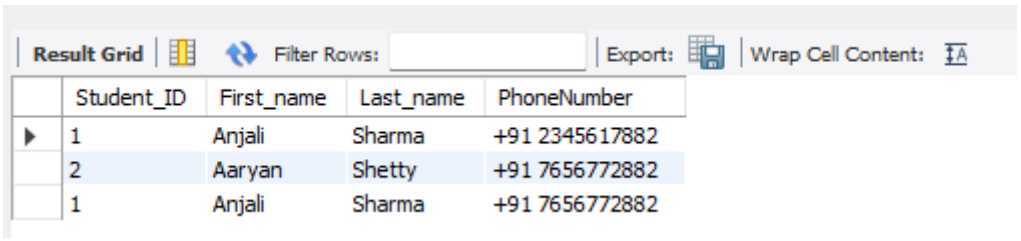
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_ID	First_name	Last_name	PhoneNumber
1	Anjali	Sharma	+91 2345617882
2	Aaryan	Shetty	+91 7656772882
1	Anjali	Sharma	+91 7656772882



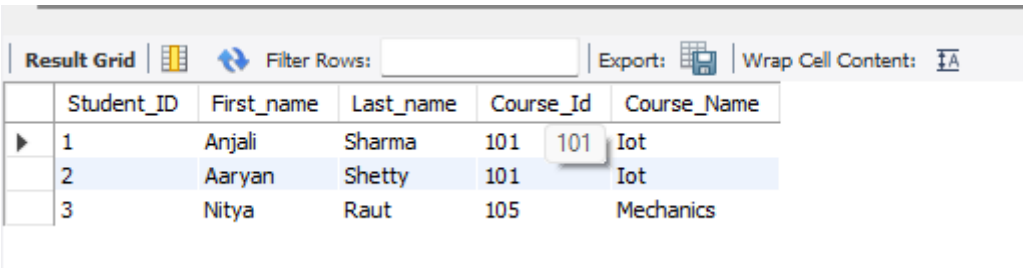
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_ID	First_name	Last_name	PhoneNumber
1	Anjali	Sharma	+91 7656772882
1	Anjali	Sharma	+91 2345617882
2	Aaryan	Shetty	+91 7656772882
3	Nitya	Raut	NULL



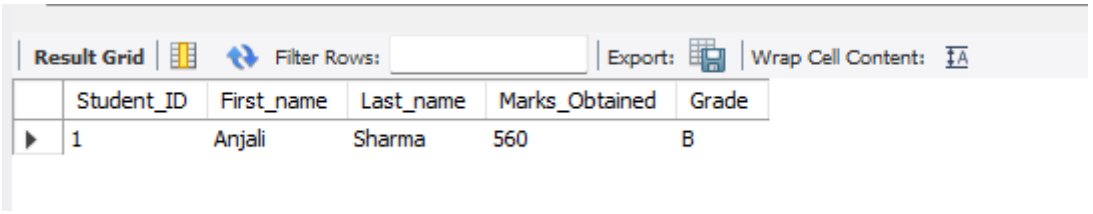
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_ID	First_name	Last_name	PhoneNumber
1	Anjali	Sharma	+91 2345617882
2	Aaryan	Shetty	+91 7656772882
1	Anjali	Sharma	+91 7656772882



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

Student_ID	First_name	Last_name	Course_Id	Course_Name
1	Anjali	Sharma	101	Iot
2	Aaryan	Shetty	101	Iot
3	Nitya	Raut	105	Mechanics



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

Student_ID	First_name	Last_name	Marks_Obtained	Grade
1	Anjali	Sharma	560	B

Conclusion:

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

A **Natural Join** is used to join tables based on common columns, typically having the same name and data type. However, when the column names are different, you cannot directly use a natural join. Instead, you can achieve the same result using an **INNER JOIN** with an explicit condition.

Example

Consider the following tables:

Employee Table

Emp_ID	Emp_Name	Dept_ID
1	Alice	D01
2	Bob	D02
3	Charlie	D01

Department Table

Department_ID	Department_Name
D01	HR
D02	IT

Performing Natural Join with Different Names

```
SELECT Employee.Emp_ID, Employee.Emp_Name, Department.Department_Name
FROM Employee
INNER JOIN Department
ON Employee.Dept_ID = Department.Department_ID;
```

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

Differences Between Natural Join, Equi Join, and Inner Join

Aspect	Natural Join	Equi Join	Inner Join
Definition	Joins tables using all columns with the same name and data type.	Joins tables using a specified equality condition.	Joins tables based on a specified condition, not limited to equality.
Column Selection	Automatically removes duplicate columns.	Retains both columns used in the join.	Retains both columns unless specified otherwise.
Condition Type	Implicit (no need to specify join condition explicitly).	Uses ON or WHERE with = operator.	Uses ON or WHERE with any logical condition.
Flexibility	Limited (only works when column names are the same).	More flexible (column names can be different).	Most flexible (any condition, including inequality or comparison).