



EXPERIMENT - 8

DATABASE MANAGEMENT SYSTEMS LAB

Aim

Write the SQL queries to implement the joins.

Syeda Reeha Quasar

14114802719

4C7

EXPERIMENT – 9

Aim:

Write the SQL queries to implement the joins.

Tools Used:

MariaDB

Procedure/ Queries:

SQL joins are used to fetch / retrieve data from two or more data tables, based on join condition. A join condition is a relationship among some columns in the data tables that take part in SQL join. Basically data tables are related to each other with keys. We use keys relationship in SQL joins.

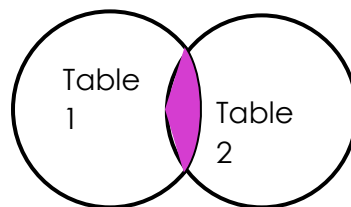
Types of Joins: in SQL Server we have only three types of joins. Using these joins we fetched the data from multiple tables based on condition.

1. INNER JOIN: Inner join returns only those records/rows that match/exists in both the tables. Syntax for inner join is as:

```
Select * from table_1 as t1
```

```
Inner join table_2 as t2
```

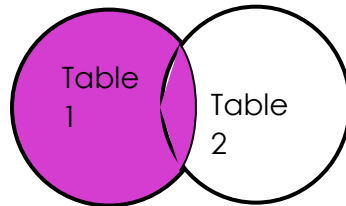
```
On t1.IDcol=t2.IDccol
```



2. OUTER JOIN: We have three types of outer joins:

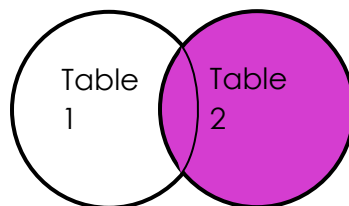
a) Left Outer Join: Left outer join returns all records/rows from left table and from right table returns only matched records. If there are no columns matched in the right table, it returns NULL values. Syntax for Left outer Join is as:

Select * from table1 as t1
Left outer join table2 as t2
On t1.IDcol=t2.IDcol



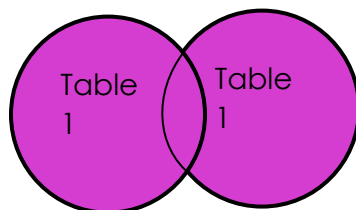
b) Right Outer Join: Right outer join returns all records/rows from right table and from left table returns only matched records. If there are no columns matched in the right table, it returns NULL values. Syntax for Left outer Join is as:

Select * from table1 as t1
Right outer join table2 as t2
On t1.IDcol=t2.IDcol



c) Full outer Join: full outer join combines left outer join and right outer join. This joint returns all records/rows from both the tables. If there are no columns matching in both the tables, it returns NULL values. Syntax for full outer join is as:

Select * from table1 as t1
Full outer join table2 as t2
On t1.IDcol=t2.IDcol



3. **CROSS JOIN:** cross join is the cartesian product of both tables. This joint does not need any condition to join two tables. This joint returns records/rows that are multiplication of records number from both the tables means each row on left table will be related to each row of right table. Syntax for cross join is:

Select*from table1

Cross join table2

4. **SELF JOIN:** Self join is used to join a data base table to itself, particularly when the table has a foreign key that references it's own primary key. Basically we have only three types of joins: inner join, outer join and cross join. We use any of these three joints to join a table to itself. Hence self join is not a type of SQL join.

INNER JOIN & LEFT JOIN :

```
MySQL Client (MariaDB 10.5 (x64)) - "C:\Program Files\MariaDB 10.5\bin\mysql.exe" "--defaults-file=C:\Program Files\MariaDB 10.5\data\my.ini" -uroot -p
| 2 | Java | 7 | 2 |
| 3 | Python | 8 | 3 |
| 4 | Kotlin | 5 | 4 |
+-----+-----+
4 rows in set (1.073 sec)

MariaDB [1k]> select student.stu_name,course.c_name from student inner join course on course.stu_id = student.stu_id;
+-----+-----+
| stu_name | c_name |
+-----+-----+
| Aman | C++ |
| Raghav | Java |
| Rohan | Python |
| Sarthak | Kotlin |
+-----+-----+
4 rows in set (0.263 sec)

MariaDB [1k]> select student.stu_name,course.c_name from student left join course on course.stu_id = student.stu_id;
+-----+-----+
| stu_name | c_name |
+-----+-----+
| Aman | C++ |
| Raghav | Java |
| Rohan | Python |
| Sarthak | Kotlin |
+-----+-----+
4 rows in set (0.015 sec)
```

2. RIGHT JOIN:

```
MariaDB [(none)]> use lk;
Database changed
MariaDB [lk]> select student.stu_name ,course.c_name
-> from student RIGHT JOIN course on course.stu_id = student.stu_id;
+-----+-----+
| stu_name | c_name |
+-----+-----+
| Aman     | C++    |
| Raghav   | Java   |
| Rohan    | Python |
| Sarthak  | Kotlin |
+-----+-----+
4 rows in set (0.133 sec)
```

QUERIES:

Find out the products which have been sold to 'Ivan Bayross'.

Input Query:

```
select c.name,p.production,p. description,s.orderno
from client_master c,product_master1 p,sales_order s,sales_order_details so
where c.name="Ivan" and c.clientno=s.clientno and s.orderno=so.orderno
and so.productno=p.production;
```

OUTPUT:

```
1 select c.name,p.production,p. description,s.orderno
2 from client_master c,product_master1 p,sales_order s,sales_order_details so
3 where c.name="Ivan" and c.clientno=s.clientno and s.orderno=so.orderno
4 and so.productno=p.production;
5
```

Result #1 (4r x 4c)			
name	production	description	orderno
Ivan	P00001	T-shirts	O19001
Ivan	P07965	DenimShirts	O19001
Ivan	P07885	PullOvers	O19001
Ivan	P06734	CottonJeans	O19003

Find out the products and their quantities that we have to deliver in the current month.

Input Query:

```
SELECT SOD.PRODUCTNO, PM.DESRIPTION, SUM(SOD.QTYORDERED)
FROM SALES_ORDER_DETAILS SOD, SALES_ORDER SO, product_master1 PM
WHERE PM.PRODUCTION = SOD.PRODUCTNO AND SO.ORDERNO = SOD.ORDERNO
AND monthname(SO.OrderDate)=monthname(curdate())
GROUP BY SOD.PRODUCTNO, PM.DESRIPTION;
```

OUTPUT:

```

1 SELECT SOD.PRODUCTNO, PM.DESRIPTION, SUM(SOD.QTYORDERED)
2 FROM SALES_ORDER_DETAILS SOD, SALES_ORDER SO, product_master1 PM
3 WHERE PM.PRODUCTION = SOD.PRODUCTNO AND SO.ORDERNO = SOD.ORDERNO
4 AND monthname(SO.OrderDate)=monthname(curdate())
5 GROUP BY SOD.PRODUCTNO, PM.DESRIPTION;
6

```

SALES_ORDER_DETAILS (0r x 3c)

PRODUCTNO	DESCRIPTION	SUM(SOD.QTYORDERED)
-----------	-------------	---------------------

list the product number and the description of the constantly sold(i.e rapidly moving) products.

Input Query:

```

select s.productno,p.description
from sales_order_details s,product_master1 p
where s.productno=p.production
and ProductRate=(select max(ProductRate) from sales_order_details);

```

OUTPUT:

```

1 select s.productno,p.description
2 from sales_order_details s,product_master1 p
3 where s.productno=p.production
4 and ProductRate=(select max(ProductRate) from sales_order_details);
5

```

Result #1 (1r x 2c)

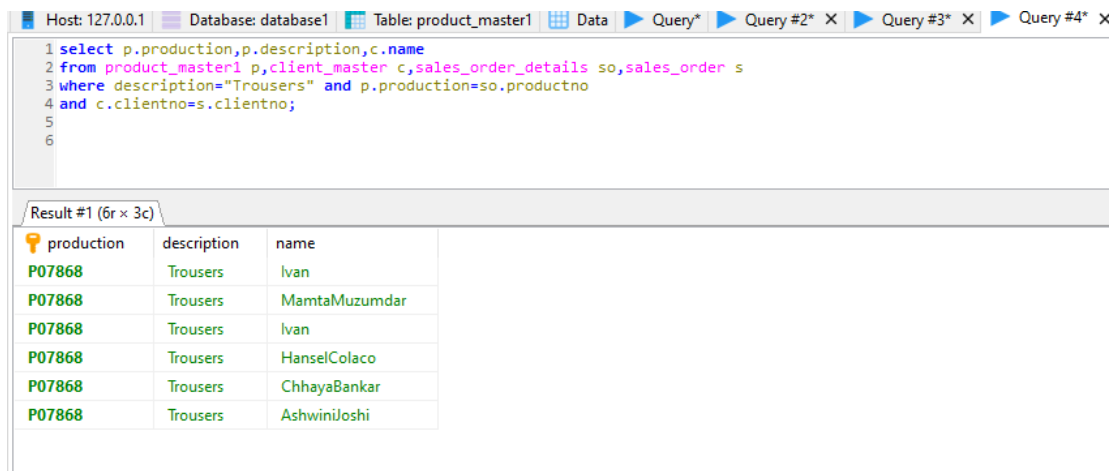
productno	description
P06734	CottonJeans

find the names of clients who have purchased 'Trousers'.

Input Query:

```
select p.production,p.description,c.name
from product_master1 p,client_master c,sales_order_details so,sales_order s
where description="Trousers" and p.production=so.productno
and c.clientno=s.clientno;
```

OUTPUT:



The screenshot shows a database query interface with a toolbar at the top containing icons for Host, Database, Table, Data, and several Query tabs. The SQL query is entered in a text area and is as follows:

```
1 select p.production,p.description,c.name
2 from product_master1 p,client_master c,sales_order_details so,sales_order s
3 where description="Trousers" and p.production=so.productno
4 and c.clientno=s.clientno;
5
6
```

Below the query editor, the results are displayed in a table titled "Result #1 (6r x 3c)". The table has three columns: production, description, and name. It contains six rows of data, all with the same production number (P07868) and description (Trousers), but with different client names.

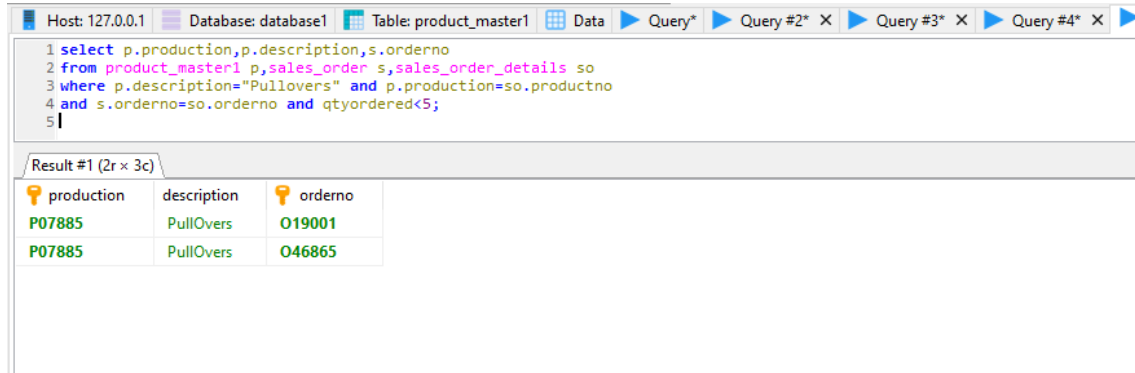
production	description	name
P07868	Trousers	Ivan
P07868	Trousers	MamtaMuzumdar
P07868	Trousers	Ivan
P07868	Trousers	HanselColaco
P07868	Trousers	ChhayaBankar
P07868	Trousers	AshwiniJoshi

list of products and orders from customers who have ordered less than five units of 'Pullovers'.

Input Query:

```
select p.production,p.description,s.orderno
from product_master1 p,sales_order s,sales_order_details so
where p.description="Pullovers" and p.production=so.productno
and s.orderno=so.orderno and qtyordered<5;
```


OUTPUT:



The screenshot shows a database query interface with a toolbar at the top containing icons for Host, Database, Table, Data, and several Query tabs. The SQL query is displayed in a text area, and the results are shown in a table below it.

```
1 select p.production,p.description,s.orderno
2 from product_master1 p,sales_order s,sales_order_details so
3 where p.description="Pullovers" and p.production=so.productno
4 and s.orderno=so.orderno and qtyordered<5;
5 |
```

Result #1 (2r x 3c)

production	description	orderno
P07885	PullOvers	O19001
P07885	PullOvers	O46865

Find the products and their quantities for the orders placed by 'Ivan Bayross' and 'Mamta Muzumdar'.

Input Query:

```
select p.production,p.description,sum(qtyordered)"QUANTITY ORDERED"
from product_master1 p,sales_order s,sales_order_details so,client_master c
where s.orderno=so.orderno and p.production=so.productno
and c.clientno=s.clientno and(c.name="Ivan" or c.name="Mamta Mazumdar")
group by so.productno,p.description;
```

OUTPUT:

Host: 127.0.0.1 Database: database1 Table: product_master1 Data Query* Query #2* × Query #3* × Query #4* ×			
<pre> 1 select p.production,p.description,sum(qtyordered)"QUANTITY ORDERED" 2 from product_master1 p,sales_order s,sales_order_details so,client_master c 3 where s.orderno=s.orderno and p.production=so.productno 4 and c.clientno=s.clientno and(c.name="Ivan" or c.name="Mamta Mazumdar")group by so.productno,p.description; 5 </pre>			
product_master1 (4r × 3c)			
production	description	QUANTITY ORDERED	
P00001	T-shirts	4	
P06734	CottonJeans	1	
P07885	PullOvers	2	
P07965	DenimShirts	2	

Find the products and their quantities for the orders placed by client number 'C00001' and 'C00002'.

Input Query:

SELECT SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRPTION, **SUM**(qTYORDERED)
UNITSORDERED

FROM SALES_ORDER SO, SALES_ORDER_DETAILS SOD, product_master1 PM,
CLIENT_MASTER CM

WHERE SO.ORDERNO = SOD.ORDERNO **AND** SOD.PRODUCTNO = PM.PRODUCTION **AND**
SO.CLIENTNO = CM.CLIENTNO

GROUP BY SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRPTION

HAVING SO.CLIENTNO = 'C00001' **OR** SO.CLIENTNO = 'C00002';

OUTPUT:

Host: 127.0.0.1 Database: database1 Table: product_master1 Data Query* Query #3* X Query #4* X Query #5* X Q

```

1 SELECT SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRPTION, SUM(qTYORDERED) UNITSORDERED
2 FROM SALES_ORDER SO, SALES_ORDER_DETAILS SOD, product_master1 PM, CLIENT_MASTER CM
3 WHERE SO.ORDERNO = SOD.ORDERNO AND SOD.PRODUCTNO = PM.PRODUCTION AND SO.CLIENTNO = CM.CLIENTNO
4 GROUP BY SO.CLIENTNO, SOD.PRODUCTNO, PM.DESCRPTION
5 HAVING SO.CLIENTNO = 'C00001' OR SO.CLIENTNO = 'C00002';
6

```

SALES_ORDER (5r x 4c)

CLIENTNO	PRODUCTNO	DESCRIPTION	UNITSORDERED
C00001	P00001	T-shirts	4
C00001	P06734	CottonJeans	1
C00001	P07885	PullOvers	2
C00001	P07965	DenimShirts	2
C00002	P00001	T-shirts	10

```

MariaDB [(none)]> create database students;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use students;
Database changed
MariaDB [students]> create table info(student_id int, student_name varchar(20), student_address varchar(50));
Query OK, 0 rows affected (0.040 sec)

MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'values(100, 'Tushar', '9/54 bagichi gali')' at line 1
MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
Query OK, 1 row affected (0.026 sec)

MariaDB [students]> insert into info values(101, 'Aayush', '8/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(102, 'Somu', '7/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(103, 'vinay', '6/54 bagichi gali');
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into info values(104, 'nimit', '5/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> create table marks(student_id int, total_marks int);
Query OK, 0 rows affected (0.029 sec)

MariaDB [students]> insert into marks values(100,90);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(101,99);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(102,98);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,97);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(104,96);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,95);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(106,89);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(107,85);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(108,84);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(109,81);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> select info.student_name, marks.total_marks from info
-> inner join marks on info.student_id = marks.student_id;
+-----+-----+

```

```

MariaDB [(none)]> create database students;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use students;
Database changed
MariaDB [students]> create table info(student_id int, student_name varchar(20), student_address varchar(50));
Query OK, 0 rows affected (0.048 sec)

MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'vlues(100, 'Tushar', '9/54 bagichi gali')' at 1
line 1
MariaDB [students]> insert into info values(100, 'Tushar', '9/54 bagichi gali');
Query OK, 1 row affected (0.026 sec)

MariaDB [students]> insert into info values(101, 'Aayush', '8/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(102, 'Sonu', '7/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into info values(103, 'vinay', '6/54 bagichi gali');
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into info values(104, 'nimit', '5/54 bagichi gali');
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> create table marks(student_id int, total_marks int);
Query OK, 0 rows affected (0.029 sec)

MariaDB [students]> insert into marks values(100,90);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(101,99);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(102,98);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,97);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(104,96);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(103,95);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(106,89);
Query OK, 1 row affected (0.002 sec)

MariaDB [students]> insert into marks values(107,85);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(108,84);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> insert into marks values(109,81);
Query OK, 1 row affected (0.001 sec)

MariaDB [students]> select info.student_name, marks.total_marks from info
-> inner join marks on info.student_id = marks.student_id;
+-----+-----+

```

VIVA QUESTIONS:

Q.1: What is a JOIN Operation?

Ans:

A join is an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables. A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by \bowtie .

Q.2: What is the difference between JOIN and PRODUCT operations?

Ans:

<u>S.no</u>	<u>JOIN Operation</u>	<u>PRODUCT Operations</u>
1.	Natural join is denoted by ' \bowtie '. It is applicable only when there is atleast one attribute common between 2 relations.	Product operation is denoted by $X * Y$ and returns a relation on tuples, whose schema contains all fields of X followed by all fields of Y.
2.	If you will use join, you will have keys which will be able to join rows from first table with rows of second table and depending on your relation, number of rows may vary.	Cartesian product means you will have rows with each record from one table matched with all rows of second table.
3.	JOIN operation is applied only with a WHERE clause.	In product operations, WHERE clause is not required.

Q.3: What are different types of JOIN operations?

Ans:

There are mainly 3 types of joins in SQL: Inner Join, Outer Join and Cross Join.

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table.
- **CROSS JOIN:** This join returns records/rows that are multiplication of record number from both the tables means each row on the left table will relate to each row of right table.

Q.4: Difference between RIGHT OUTER JOIN and LEFT OUTER JOIN?

Ans:

<u>S.NO</u>	<u>RIGHT OUTER JOIN</u>	<u>LEFT OUTER JOIN</u>
1.	Fetches all the rows from the table on the right, regardless of whether there are any matching columns in the "left" table.	Fetches all the rows from the table on the left, regardless of whether there are any matching columns in the "right" table.
2.	The result of Right Outer Join can be seen as: Inner Join +all the unmatched rows from the right table.	The result of Left Outer Join can be seen as: Inner Join +all the unmatched rows from the left table.
3.	Unmatched data of the left table is lost.	Unmatched data of the right table is lost.
4.	If left table doesn't have the matching record then for such records left table column will have NULL value in the result.	If right table doesn't have the matching record then for such records right table column will have NULL value in the result.

Q.5: What is an INNER Join?

Ans:

Inner Join returns only those records/rows that match/exists in both the tables. Inner Join clause in SQL creates a new table (not physical) by combining rows that have matching values in two or more tables. This join is based on a logical relationship (or a common field) between the tables and is used to retrieve data that appears in both tables. Syntax for Inner Join is as follows:

```
Select* from table_1 as t1
inner join table_2 as t2
on t1.IDcol=t2.IDcol;
```