

....ASSIGNMENT : SQL JOINS....

Note : Create the following dummy tables in MySQL Workbench using CREATE FUNCTION-

Table 1: Customers

CUSTOMER ID	CUSTOMER NAME	CITY
1	John Smith	New York
2	Mary Johnson	Chicago
3	Peter Adams	Los Angeles
4	Nancy Miller	Houston
5	Robert White	Miami

Create database customersdataset;

Use customers dataset;

```
CREATE TABLE Customers (
```

```
    CustomerID INT PRIMARY KEY,
```

```
    CustomerName VARCHAR(50),
```

```
    City VARCHAR(50));
```

```
INSERT INTO Customers VALUES
```

```
(1, 'John Smith', 'New York'),
```

```
(2, 'Mary Johnson', 'Chicago'),
```

```
(3, 'Peter Adams', 'Los Angeles'),
```

```
(4, 'Robert White', 'Houston'),
```

```
(5, 'Nancy Miller', 'Miami');
```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window with the following SQL code:

```
19
);
20 •   SELECT * FROM CUSTOMERS;
--
```

The line 20 is highlighted in blue, indicating it is the current statement being executed. Below the code editor is a result grid titled "Result Grid". The grid has three columns: "CustomerID", "CustomerName", and "City". The data matches the table provided in the assignment:

CustomerID	CustomerName	City
1	John Smith	New York
2	Mary Johnson	Chicago
3	Peter Adams	Los Angeles
4	Robert White	Houston
5	Nancy Miller	Miami
NULL	NULL	NULL

TABLE 2:ORDERS

ORDER ID	CUSTOMER ID	ORDER DATE	AMOUNT
101	1	2024-10-01	250
102	2	2024-10-05	300
103	1	2024-10-07	150
104	3	2024-10-10	450
105	6	2024-10-12	400

CREATE DATABASE ORDERSDATASET;

USE ORDERSDATASET;

CREATE TABLE Orders (

 OrderID INT PRIMARY KEY,

 CustomerID INT,

 OrderDate DATE,

 Amount INT);

INSERT INTO Orders VALUES

(101, 1, '2024-10-01', 250),

(102, 2, '2024-10-05', 300),

(103, 1, '2024-10-07', 150),

(104, 3, '2024-10-10', 450),

(105, 6, '2024-10-12', 400);

The screenshot shows a database interface with a code editor and a result grid. The code editor contains SQL commands to create a database, a table, and insert data into it. The result grid displays the inserted data.

```
32
33     (104, 3, '2024-10-10', 450),
34     (105, 6, '2024-10-12', 400);
35 •   SELECT * FROM ORDERS;
```

	OrderID	CustomerID	OrderDate	Amount
▶	101	1	2024-10-01	250
	102	2	2024-10-05	300
	103	1	2024-10-07	150
	104	3	2024-10-10	450
	105	6	2024-10-12	400
*	NULL	NULL	NULL	NULL

Table 3: payments

Payment id	Customer id	Payment date	AMOUNT
P001	1	2024-10-02	250
P002	2	2024-10-06	300
P003	3	2024-10-11	450
P004	4	2024-10-15	200

Create database paymentsdataset;

Use paymentsdataset;

```
CREATE TABLE Payments (
```

```
    PaymentID VARCHAR(10) PRIMARY KEY,
```

```
    CustomerID INT,
```

```
    PaymentDate DATE,
```

```
    Amount INT
```

```
);
```

```
INSERT INTO Payments VALUES
```

```
('P001', 1, '2024-10-02', 250),
```

```
('P002', 2, '2024-10-06', 300),
```

```
('P003', 3, '2024-10-11', 450),
```

```
('P004', 4, '2024-10-15', 200);
```

The screenshot shows a database interface with a 'Result Grid' tab selected. The grid displays the data from the 'Payments' table. The columns are labeled: PaymentID, CustomerID, PaymentDate, and Amount. The data rows correspond to the entries in Table 3: P001 (1, 2024-10-02, 250), P002 (2, 2024-10-06, 300), P003 (3, 2024-10-11, 450), and P004 (4, 2024-10-15, 200). There is also a row with all NULL values.

	PaymentID	CustomerID	PaymentDate	Amount
▶	P001	1	2024-10-02	250
	P002	2	2024-10-06	300
	P003	3	2024-10-11	450
	P004	4	2024-10-15	200
	NULL	NULL	NULL	NULL

TABLE 4: EMPLOYEES

EMPLOYEE ID	EMPLOYEE NAME	MANAGER ID
1	Alex Green	NULL
2	Brian Lee	1
3	Carol Ray	1
4	David Kim	2
5	Eva Smith	2

CREATE DATABASE EMPLOYEESDATASET;

USE EMPLOYEESDATASET;

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(50),

ManagerID INT

);

INSERT INTO Employees VALUES

(1, 'Alex Green', NULL),

(2, 'Brian Lee', 1),

(3, 'Carol Ray', 1),

(4, 'David Kim', 2),

(5, 'Eva Smith', 2);

Result Grid | Filter Rows:

	EmployeeID	EmployeeName	ManagerID
▶	1	Alex Green	NULL
	2	Brian Lee	1
	3	Carol Ray	1
	4	David Kim	2
	5	Eva Smith	2
	NULL	NULL	NULL

Q NO 1: Retrieve all customers who have placed at least one order.

SOL: SELECT DISTINCT c.*

```
FROM Customers c
INNER JOIN Orders o
ON c.CustomerID = o.CustomerID;
```

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

	CustomerID	CustomerName	City
▶	1	John Smith	New York
	2	Mary Johnson	Chicago
	3	Peter Adams	Los Angeles

Q NO 2: Retrieve all customers and their orders, including customers who have not placed any orders.

SOL: SELECT c.CustomerName, o.OrderID, o.Amount

```
FROM Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID;
```

Result Grid		Filter Rows:	Export:		
	CustomerID	CustomerName	OrderID	OrderDate	Amount
▶	1	John Smith	103	2024-10-07	150
	1	John Smith	101	2024-10-01	250
	2	Mary Johnson	102	2024-10-05	300
	3	Peter Adams	104	2024-10-10	450
	4	Robert White	NULL	NULL	NULL
	5	Nancy Miller	NULL	NULL	NULL

Q NO 3: Retrieve all orders and their corresponding customers, including orders placed by unknown customers.

SOL: SELECT o.OrderID, c.CustomerName, o.Amount

FROM Orders o

LEFT JOIN Customers c

ON o.CustomerID = c.CustomerID;

ON o.CustomerID = c.CustomerID;							
Result Grid		Filter Rows:	Export:			Wrap Cell Contents	
	OrderID	OrderDate	Amount	CustomerID	CustomerName	City	
▶	101	2024-10-01	250	1	John Smith	New York	
	102	2024-10-05	300	2	Mary Johnson	Chicago	
	103	2024-10-07	150	1	John Smith	New York	
	104	2024-10-10	450	3	Peter Adams	Los Angeles	
	105	2024-10-12	400	NULL	NULL	NULL	

Q NO 4:Display all customers and orders, whether matched or not.

SOL: SELECT c.CustomerName, o.OrderID

FROM Customers c

LEFT JOIN Orders o

ON c.CustomerID = o.CustomerID

UNION

SELECT c.CustomerName, o.OrderID

FROM Customers c

RIGHT JOIN Orders o

ON c.CustomerID = o.CustomerID;

The screenshot shows a database query result grid titled "Result Grid". The columns are CustomerID, CustomerName, OrderID, and Amount. The data includes rows for John Smith, Mary Johnson, Peter Adams, Robert White, Nancy Miller, and a row with NULL values.

	CustomerID	CustomerName	OrderID	Amount
▶	1	John Smith	103	150
	1	John Smith	101	250
	2	Mary Johnson	102	300
	3	Peter Adams	104	450
	4	Robert White	NULL	NULL
	5	Nancy Miller	NULL	NULL
	NULL	NULL	105	400

Q NO 5: Find customers who have not placed any orders.

SOL: SELECT

```
c.CustomerID,  
c.CustomerName,  
c.City  
FROM Customers c  
LEFT JOIN Orders o  
ON c.CustomerID = o.CustomerID  
WHERE o.OrderID IS NULL;
```

The screenshot shows a database query result grid titled "Result Grid". The column is CustomerName, displaying Robert White and Nancy Miller.

	CustomerName
▶	Robert White
	Nancy Miller

Q NO 6: Retrieve customers who made payments but did not place any orders.

SOL: SELECT DISTINCT

```
c.CustomerID,  
c.CustomerName,
```

```

c.City
FROM Customers c
INNER JOIN Payments p
ON c.CustomerID = p.CustomerID
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
WHERE o.OrderID IS NULL;

```

The screenshot shows a MySQL command-line interface. The query entered is:

```

150 FROM Customers c
151 INNER JOIN Payments p
152 ON c.CustomerID = p.CustomerID
153 LEFT JOIN Orders o
154 ON c.CustomerID = o.CustomerID
155 WHERE o.OrderID IS NULL;
156

```

The result grid displays the following data:

CustomerName
Robert White

Result 11 ×

Q NO 7: Generate a list of all possible combinations between Customers and Orders.

SOL: SELECT

```

c.CustomerID,
c.CustomerName,
o.OrderID,
o.Amount
FROM Customers c
CROSS JOIN Orders o;

```

The image shows two separate database result grids, likely from a SQL query execution environment. Both grids have columns: CustomerID, CustomerName, OrderID, and Amount.

Result Grid 1 (Top):

	CustomerID	CustomerName	OrderID	Amount
▶	5	Nancy Miller	101	250
	4	Robert White	101	250
	3	Peter Adams	101	250
	2	Mary Johnson	101	250
	1	John Smith	101	250
	5	Nancy Miller	102	300
	4	Robert White	102	300
	3	Peter Adams	102	300
	2	Mary Johnson	102	300
	1	John Smith	102	300
	5	Nancy Miller	103	150
	4	Robert White	103	150
	3	Peter Adams	103	150

Result Grid 2 (Bottom):

	CustomerID	CustomerName	OrderID	Amount
	3	Peter Adams	103	150
	2	Mary Johnson	103	150
	1	John Smith	103	150
	5	Nancy Miller	104	450
	4	Robert White	104	450
	3	Peter Adams	104	450
	2	Mary Johnson	104	450
	1	John Smith	104	450
	5	Nancy Miller	105	400
	4	Robert White	105	400
	3	Peter Adams	105	400
	2	Mary Johnson	105	400
	1	John Smith	105	400

Q NO 8: Show all customers along with order and payment amounts in one table.

SOL: SELECT

```

c.CustomerName,
o.Amount AS OrderAmount,
p.Amount AS PaymentAmount
FROM Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
LEFT JOIN Payments p
ON c.CustomerID = p.CustomerID;

```

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

	CustomerName	OrderAmount	PaymentAmount
▶	John Smith	150	250
	John Smith	250	250
	Mary Johnson	300	300
	Peter Adams	450	450
	Robert White	NULL	200
	Nancy Miller	NULL	NULL

Result 13 ×

Output

Action Output

Time Action

Q NO 9: Retrieve all customers who have both placed orders and made payments.

SOL: SELECT DISTINCT c.CustomerName

FROM Customers c

INNER JOIN Orders o

ON c.CustomerID = o.CustomerID

INNER JOIN Payments p

ON c.CustomerID = p.CustomerID;

177 ON c.CustomerID = p.CustomerID;

178 ---

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

	CustomerName
▶	John Smith
	Mary Johnson
	Peter Adams

Result 14 ×