**Analytics Avenue for Research and Development**

-In the journey of empowering the digital minds

**Worksheet-3**

**JOINS**

**1. Inner Join**

**Problem**: You have two tables:

* Customers(customer\_id, name, email)
* Orders(order\_id, customer\_id, order\_date, total\_amount)

Write an SQL query to fetch all customers who have placed an order, along with the details of their orders.

ANSWER:

select

c.customer\_id,

c.name,

c.email,

o.order\_id,

o.order\_date,

o.total\_amount

from

customers c

inner join

orders o on c.customer\_id = o.customer\_id;

**2. Left Join**

**Problem**: You have two tables:

* Employees(employee\_id, name, department\_id)
* Departments(department\_id, department\_name)

Write an SQL query to fetch all employees, along with their department names. If an employee is not assigned to a department, return NULL for the department name.

ANSWER:

SELECT

e.employee\_id,

e.name,

d.department\_name

FROM

Employees e

LEFT JOIN

Departments d ON e.department\_id = d.department\_id;

**3.** **Right Join**

**Problem**: You have two tables:

* Products(product\_id, product\_name)
* Orders(order\_id, product\_id, quantity)

Write an SQL query to fetch all products and the total quantity ordered for each product. Include products even if they have never been ordered (with NULL values for quantity).

ANSWER:

SELECT

p.product\_id,

p.product\_name,

SUM(o.quantity) AS total\_quantity

FROM

Orders o

RIGHT JOIN

Products p ON p.product\_id = o.product\_id

GROUP BY

p.product\_id, p.product\_name;

**4. Full Outer Join**

**Problem**: You have two tables:

* Students(student\_id, student\_name)
* Courses(course\_id, course\_name)

Write an SQL query to fetch all students and the courses they have registered for. Include students who haven't registered for any courses, as well as courses that no student has registered for.

ANSWER:

SELECT

s.student\_id,

s.student\_name,

c.course\_id,

c.course\_name

FROM

Students s

FULL OUTER JOIN

Courses c ON s.student\_id = c.course\_id;

**5.** **Cross Join**

**Problem**: You have two tables:

* Categories(category\_id, category\_name)
* Products(product\_id, product\_name)

Write an SQL query to display all possible combinations of categories and products.

ANSWER:

SELECT

c.category\_id,

c.category\_name,

p.product\_id,

p.product\_name

FROM

Categories c

CROSS JOIN

Products p;

**6. Union**

**Problem**: You have two tables:

* Employees(employee\_id, name)
* Contractors(contractor\_id, name)

Write an SQL query to list all individuals (both employees and contractors) with unique names.

ANSWER:

SELECT name

FROM Employees

UNION

SELECT name

FROM Contractors;

**7. Union All**

**Problem**: You have two tables:

* OnlineOrders(order\_id, customer\_name)
* InStoreOrders(order\_id, customer\_name)

Write an SQL query to list all customer names who placed orders, including possible duplicates from both OnlineOrders and InStoreOrders.

ANSWER:

SELECT customer\_name

FROM OnlineOrders

UNION ALL

SELECT customer\_name

FROM InStoreOrders;

**Date Time Functions**

**1. Extracting Date Parts (YEAR, MONTH, DAY, WEEK, HOUR, etc.)**

**Problem**: You have a table Sales(order\_id, order\_date, total\_amount).

Write an SQL query to find the total sales (total\_amount) for each month in the year 2023.

* **Hint**: Use the YEAR() and MONTH() functions.

**ANSWER**:

SELECT

YEAR(order\_date) AS order\_year,

MONTH(order\_date) AS order\_month,

SUM(total\_amount) AS total\_sales

FROM

Sales

WHERE

YEAR(order\_date) = 2023

GROUP BY

YEAR(order\_date), MONTH(order\_date)

ORDER BY

order\_month;

**2. Date Difference (DATEDIFF, TIMESTAMPDIFF)**

**Problem**: You have a table Employees(employee\_id, name, hire\_date).

Write an SQL query to find all employees who have been working for more than 5 years.

* **Hint**: Use the DATEDIFF() or TIMESTAMPDIFF() function.

**ANSWER**:

SELECT employee\_id, name, hire\_date

FROM Employees

WHERE DATEDIFF(CURDATE(), hire\_date) > 1825;

**3. Current Date and Time (NOW(), CURRENT\_DATE, CURRENT\_TIME)**

**Problem**: You have a table Subscriptions(subscription\_id, user\_id, start\_date).

Write an SQL query to find all subscriptions that started in the last 30 days.

* **Hint**: Use NOW() or CURRENT\_DATE() with date calculations.

**ANSWER**:

SELECT subscription\_id, user\_id, start\_date

FROM Subscriptions

WHERE start\_date >= CURDATE() - INTERVAL 30 DAY;

**4. Date Arithmetic (DATE\_ADD, DATE\_SUB)**

**Problem**: You have a table Projects(project\_id, start\_date, deadline).

Write an SQL query to find all projects whose deadlines are within the next 7 days from today.

* **Hint**: Use the DATE\_ADD() or DATE\_SUB() functions.

**ANSWER**:

SELECT project\_id, start\_date, deadline

FROM Projects

WHERE deadline BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 7 DAY);

**5. Formatting Dates (DATE\_FORMAT)**

**Problem**: You have a table Orders(order\_id, order\_date).

Write an SQL query to display the order dates in the format MM-DD-YYYY.

* **Hint**: Use the DATE\_FORMAT() function.

ANSWER:

SELECT order\_id, DATE\_FORMAT(order\_date, '%m-%d-%Y') AS formatted\_order\_date

FROM Orders;

**6. Date Truncation (DATE\_TRUNC, TRUNCATE())**

**Problem**: You have a table LogEntries(log\_id, log\_date, event).

Write an SQL query to count how many logs were recorded for each week.

* **Hint**: Use the DATE\_TRUNC('week', log\_date) function or equivalent in your SQL flavor.

**ANSWER**:

**SQL Query using DATE\_TRUNC() (for PostgreSQL):**

SELECT DATE\_TRUNC('week', log\_date) AS week\_start, COUNT(\*) AS log\_count

FROM LogEntries

GROUP BY DATE\_TRUNC('week', log\_date)

ORDER BY week\_start;

**7. Working with Time (HOUR, MINUTE, SECOND)**

**Problem**: You have a table ServerLogs(log\_id, log\_time).

Write an SQL query to find all logs that were recorded between 2 PM and 5 PM.

* **Hint**: Use the HOUR() function to extract the hour part from the log\_time column.

**ANSWER**:

SELECT log\_id, log\_time

FROM ServerLogs

WHERE HOUR(log\_time) BETWEEN 14 AND 17;

**8. Finding Day of the Week (DAYOFWEEK, DAYNAME)**

**Problem**: You have a table Deliveries(delivery\_id, delivery\_date).

Write an SQL query to find how many deliveries were made on weekends (Saturday and Sunday).

* **Hint**: Use the DAYOFWEEK() or DAYNAME() function.

**ANSWER**:

SELECT COUNT(\*) AS weekend\_deliveries

FROM Deliveries

WHERE DAYOFWEEK(delivery\_date) IN (1, 7);

**9. Finding First and Last Day of the Month (LAST\_DAY, DATE\_ADD, DATE\_SUB)**

**Problem**: You have a table Invoices(invoice\_id, invoice\_date).

Write an SQL query to find all invoices issued in the last day of any month.

* **Hint**: Use the LAST\_DAY() function.

**ANSWER**:

**SQL Query using LAST\_DAY():**

SELECT invoice\_id, invoice\_date

FROM Invoices

WHERE invoice\_date = LAST\_DAY(invoice\_date);

**10. Timestamp Comparison (TIMESTAMP(), UNIX\_TIMESTAMP())**

**Problem**: You have a table Bookings(booking\_id, start\_time, end\_time).

Write an SQL query to find all bookings that lasted more than 3 hours.

* **Hint**: Use TIMESTAMPDIFF(HOUR, start\_time, end\_time).

**ANSWER**:

SELECT booking\_id, start\_time, end\_time

FROM Bookings

WHERE TIMESTAMPDIFF(HOUR, start\_time, end\_time) > 3;

**Case when, Rollup, Group\_concat**

**1. CASE WHEN**

**Problem**: You have a table Employees(employee\_id, name, salary, department).

Write an SQL query that categorizes employees into three salary ranges:

* Low salary: Less than $3000
* Medium salary: Between $3000 and $6000
* High salary: More than $6000

Return the employee name, department, and their salary category ("Low", "Medium", or "High").

* **Hint**: Use CASE WHEN to create the salary ranges.

**ANSWER**:

SELECT name, department, salary,

CASE

WHEN salary < 3000 THEN 'Low'

WHEN salary BETWEEN 3000 AND 6000 THEN 'Medium'

WHEN salary > 6000 THEN 'High'

END AS salary\_category

FROM Employees;

**2. CASE WHEN with Aggregation**

**Problem**: You have a table Orders(order\_id, customer\_id, order\_date, total\_amount).

Write an SQL query that returns the total number of orders, the total number of orders placed in 2023, and the total number of orders placed in 2022.

* **Hint**: Use CASE WHEN inside an aggregate function like COUNT() to filter by year.

**ANSWER**:

SELECT

COUNT(\*) AS total\_orders,

COUNT(CASE WHEN YEAR(order\_date) = 2023 THEN 1 END) AS orders\_2023,

COUNT(CASE WHEN YEAR(order\_date) = 2022 THEN 1 END) AS orders\_2022

FROM Orders;

**3. WITH ROLLUP**

**Problem**: You have a table Sales(sale\_id, product\_id, product\_category, sale\_amount).

Write an SQL query to display the total sales amount for each product category. Also, include an additional row that shows the overall total sales across all product categories.

* **Hint**: Use GROUP BY product\_category WITH ROLLUP.

**ANSWER**:

SELECT product\_category,

SUM(sale\_amount) AS total\_sales

FROM Sales

GROUP BY product\_category WITH ROLLUP;

**4. WITH ROLLUP and CASE WHEN**

**Problem**: You have a table Sales(sale\_id, product\_id, region, sale\_amount).

Write an SQL query to display the total sales amount for each region, as well as the overall total sales. If the region is NULL (generated by ROLLUP), display it as "Overall Total".

* **Hint**: Use WITH ROLLUP and CASE WHEN to replace NULL with "Overall Total".

ANSWER:

SELECT

CASE

WHEN region IS NULL THEN 'Overall Total'

ELSE region

END AS region,

SUM(sale\_amount) AS total\_sales

FROM Sales

GROUP BY region WITH ROLLUP;

**5. GROUP\_CONCAT**

**Problem**: You have a table Courses(course\_id, course\_name, student\_id) where each course has multiple students.

Write an SQL query to list each course, along with the names of all students enrolled in that course, as a single comma-separated string.

* **Hint**: Use GROUP\_CONCAT() to concatenate the student names.

**ANSWER**:

SELECT course\_name,

GROUP\_CONCAT(student\_id ORDER BY student\_id) AS enrolled\_students

FROM Courses

GROUP BY course\_name;

**6. GROUP\_CONCAT with Distinct**

**Problem**: You have a table Orders(order\_id, customer\_name, product\_id) where each order can include multiple products and customers may have multiple orders.

Write an SQL query to list each customer, along with a distinct list of all products they have ever ordered.

* **Hint**: Use GROUP\_CONCAT(DISTINCT product\_id) to get a unique list of products for each customer.

**ANSWER**:

SELECT customer\_name,

GROUP\_CONCAT(DISTINCT product\_id ORDER BY product\_id) AS products\_ordered

FROM Orders

GROUP BY customer\_name;

**7. GROUP\_CONCAT with Order**

**Problem**: You have a table Projects(project\_id, team\_member, start\_date).

Write an SQL query to list each project and its team members in alphabetical order, separated by commas.

* **Hint**: Use GROUP\_CONCAT() with ORDER BY inside the function.

**ANSWER**:

SELECT project\_id,

GROUP\_CONCAT(team\_member ORDER BY team\_member ASC) AS team\_members

FROM Projects

GROUP BY project\_id;