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;
```