SQL Practice questions

1. **Retrieve all columns from a table named "employees":**

sql

SELECT \* FROM employees;

1. **Retrieve the names and ages of employees who are older than 30 years:**

sql

SELECT name, age FROM employees WHERE age > 30;

1. **Calculate the average salary of all employees:**

sql

SELECT AVG(salary) AS average\_salary FROM employees;

1. **Find the highest salary from the "employees" table:**

sql

SELECT MAX(salary) AS highest\_salary FROM employees;

1. **Retrieve the top 5 highest-paid employees along with their names and salaries:**

sql

SELECT name, salary FROM employees ORDER BY salary DESC LIMIT 5;

1. **Count the number of employees in each department and display the department name along with the count:**

sql

SELECT department, COUNT(\*) AS num\_employees FROM employees GROUP BY department;

1. **Display the names of employees who have the same age, and sort them in ascending order:**

sql

SELECT name, age FROM employees

WHERE age IN (SELECT age FROM employees GROUP BY age HAVING COUNT(\*) > 1)

ORDER BY age, name;

1. **Retrieve the names of employees whose names start with "John" and have salaries greater than 50000:**

sql

SELECT name FROM employees WHERE name LIKE 'John%' AND salary > 50000;

1. **Find the total salary expenditure for each department and display the department name along with the total salary:**

sql

SELECT department, SUM(salary) AS total\_salary FROM employees GROUP BY department;

1. **Retrieve the names of employees who do not belong to any department:**

sql

SELECT name FROM employees WHERE department IS NULL;

1. **Calculate the average salary of employees for each age group (<= 30 and > 30) and display the age group along with the average salary:**

sql

SELECT

CASE

WHEN age <= 30 THEN '<= 30'

ELSE '> 30'

END AS age\_group,

AVG(salary) AS average\_salary

FROM employees

GROUP BY age\_group;

1. **List the employees whose salaries are within the top 10% of the highest salary:**

sql

SELECT name, salary FROM employees

WHERE salary >= (SELECT 0.9 \* MAX(salary) FROM employees);

1. Consider a table named "orders" with the following columns:

* order\_id (integer)
* customer\_id (integer)
* order\_date (date)
* total\_amount (decimal)

Write a SQL query to find the total sales amount for each customer, along with the customer's name and the number of orders they made. Assume that there is another table named "customers" with the following columns:

* customer\_id (integer)
* customer\_name (varchar)

Your query should join the two tables using the "customer\_id" column.

Here's the SQL query to achieve this:

sql

SELECT

c.customer\_name,

COUNT(o.order\_id) AS num\_orders,

SUM(o.total\_amount) AS total\_sales\_amount

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

GROUP BY

c.customer\_id, c.customer\_name;

In this query, we use the GROUP BY clause to group the results by "customer\_id" and "customer\_name" from the "customers" table. The JOIN operation combines the "orders" table using the "customer\_id" column. The COUNT function is used to calculate the number of orders each customer made, and the SUM function is used to calculate the total sales amount for each customer. The result will show the customer name, the number of orders they made, and the total sales amount for each customer.

1. Consider two tables, "orders" and "customers." The "orders" table has columns "order\_id," "customer\_id," and "order\_date," and the "customers" table has columns "customer\_id" and "customer\_name."

**Example: Retrieve the names of customers who have placed orders.**

SELECT customer\_name

FROM customers

WHERE customer\_id IN (SELECT DISTINCT customer\_id FROM orders);

1. ADD column using case

ALTER TABLE products

ADD category VARCHAR(50) DEFAULT 'Unknown'

GENERATED ALWAYS AS (

CASE

WHEN price <= 50 THEN 'Low Price'

WHEN price > 50 AND price <= 100 THEN 'Medium Price'

ELSE 'High Price'

END

);

In this example, the "category" column will be automatically populated based on the "price" column values. If the price is less than or equal to 50, the category will be set as "Low Price," if the price is between 50 (exclusive) and 100 (inclusive), the category will be "Medium Price," and for any other values of price, the category will be set as "High Price." The default value is set as "Unknown" in case none of the conditions are met.