1. **Select all records from a table:**

sql

Copy code

SELECT \* FROM employees;

1. **Select specific columns from a table:**

sql

Copy code

SELECT first\_name, last\_name FROM employees;

1. **Select distinct values from a column:**

sql

Copy code

SELECT DISTINCT department FROM employees;

1. **Count the number of rows in a table:**

sql

Copy code

SELECT COUNT(\*) FROM employees;

1. **Count the number of distinct values in a column:**

sql

Copy code

SELECT COUNT(DISTINCT department) FROM employees;

1. **Select rows with a condition (WHERE clause):**

sql

Copy code

SELECT \* FROM employees WHERE salary > 50000;

1. **Select rows with multiple conditions (AND operator):**

sql

Copy code

SELECT \* FROM employees WHERE department = 'Sales' AND salary > 50000;

1. **Select rows with multiple conditions (OR operator):**

sql

Copy code

SELECT \* FROM employees WHERE department = 'Sales' OR salary > 50000;

1. **Select rows where a column value is in a list (IN operator):**

sql

Copy code

SELECT \* FROM employees WHERE department IN ('Sales', 'Marketing', 'IT');

1. **Select rows where a column value is not in a list (NOT IN operator):**

sql

Copy code

SELECT \* FROM employees WHERE department NOT IN ('Sales', 'Marketing', 'IT');

**2. Advanced Filtering**

1. **Select rows using the BETWEEN operator:**

sql

Copy code

SELECT \* FROM employees WHERE salary BETWEEN 40000 AND 60000;

1. **Select rows using the LIKE operator for pattern matching:**

sql

Copy code

SELECT \* FROM employees WHERE first\_name LIKE 'A%';

1. **Select rows using the NOT LIKE operator:**

sql

Copy code

SELECT \* FROM employees WHERE first\_name NOT LIKE 'A%';

1. **Select rows using the IS NULL operator:**

sql

Copy code

SELECT \* FROM employees WHERE manager\_id IS NULL;

1. **Select rows using the IS NOT NULL operator:**

sql

Copy code

SELECT \* FROM employees WHERE manager\_id IS NOT NULL;

**3. Sorting Results**

1. **Sort results in ascending order:**

sql

Copy code

SELECT \* FROM employees ORDER BY salary ASC;

1. **Sort results in descending order:**

sql

Copy code

SELECT \* FROM employees ORDER BY salary DESC;

1. **Sort results by multiple columns:**

sql

Copy code

SELECT \* FROM employees ORDER BY department ASC, salary DESC;

**4. Aggregate Functions**

1. **Calculate the sum of a column:**

sql

Copy code

SELECT SUM(salary) FROM employees;

1. **Calculate the average of a column:**

sql

Copy code

SELECT AVG(salary) FROM employees;

1. **Find the minimum value in a column:**

sql

Copy code

SELECT MIN(salary) FROM employees;

1. **Find the maximum value in a column:**

sql

Copy code

SELECT MAX(salary) FROM employees;

1. **Group rows and calculate the sum for each group:**

sql

Copy code

SELECT department, SUM(salary) FROM employees GROUP BY department;

1. **Group rows and calculate the average for each group:**

sql

Copy code

SELECT department, AVG(salary) FROM employees GROUP BY department;

1. **Group rows and find the maximum value for each group:**

sql

Copy code

SELECT department, MAX(salary) FROM employees GROUP BY department;

1. **Group rows and find the minimum value for each group:**

sql

Copy code

SELECT department, MIN(salary) FROM employees GROUP BY department;

1. **Filter groups using HAVING clause:**

sql

Copy code

SELECT department, AVG(salary) FROM employees GROUP BY department HAVING AVG(salary) > 50000;

**5. Joining Tables**

1. **Inner join between two tables:**

sql

Copy code

SELECT employees.first\_name, departments.department\_name

FROM employees

INNER JOIN departments ON employees.department\_id = departments.department\_id;

1. **Left join between two tables:**

sql

Copy code

SELECT employees.first\_name, departments.department\_name

FROM employees

LEFT JOIN departments ON employees.department\_id = departments.department\_id;

1. **Right join between two tables:**

sql

Copy code

SELECT employees.first\_name, departments.department\_name

FROM employees

RIGHT JOIN departments ON employees.department\_id = departments.department\_id;

1. **Full outer join between two tables:**

sql

Copy code

SELECT employees.first\_name, departments.department\_name

FROM employees

FULL OUTER JOIN departments ON employees.department\_id = departments.department\_id;

1. **Cross join between two tables:**

sql

Copy code

SELECT employees.first\_name, departments.department\_name

FROM employees

CROSS JOIN departments;

**6. Subqueries**

1. **Subquery in WHERE clause:**

sql

Copy code

SELECT \* FROM employees WHERE department\_id = (SELECT department\_id FROM departments WHERE department\_name = 'Sales');

1. **Subquery in SELECT clause:**

sql

Copy code

SELECT first\_name, last\_name, (SELECT department\_name FROM departments WHERE departments.department\_id = employees.department\_id) AS department\_name FROM employees;

1. **Subquery with IN operator:**

sql

Copy code

SELECT \* FROM employees WHERE department\_id IN (SELECT department\_id FROM departments WHERE location = 'New York');

1. **Subquery with EXISTS operator:**

sql

Copy code

SELECT \* FROM employees WHERE EXISTS (SELECT 1 FROM departments WHERE departments.department\_id = employees.department\_id AND location = 'New York');

**7. Common Table Expressions (CTEs)**

1. **Simple CTE usage:**

sql

Copy code

WITH DepartmentSalaries AS (

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

)

SELECT \* FROM DepartmentSalaries;

1. **CTE with multiple references:**

sql

Copy code

WITH DepartmentSalaries AS (

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

)

SELECT department\_id FROM DepartmentSalaries WHERE total\_salary > 100000;

**8. Data Manipulation**

1. **Insert a new record into a table:**

sql

Copy code

INSERT INTO employees (first\_name, last\_name, department\_id, salary) VALUES ('John', 'Doe', 3, 55000);

1. **Insert multiple records into a table:**

sql

Copy code

INSERT INTO employees (first\_name, last\_name, department\_id, salary)

VALUES ('Jane', 'Smith', 4, 60000), ('Mark', 'Johnson', 5, 70000);

1. **Update a record in a table:**

sql

Copy code

UPDATE employees SET salary = 60000 WHERE first\_name = 'John' AND last\_name = 'Doe';

1. **Update multiple records in a table:**

sql

Copy code

UPDATE employees SET salary = salary \* 1.1 WHERE department\_id = 4;

1. **Delete a record from a table:**

sql

Copy code

DELETE FROM employees WHERE first\_name = 'John' AND last\_name = 'Doe';

1. **Delete multiple records from a table:**

sql

Copy code

DELETE FROM employees WHERE department\_id = 4;

**9. Advanced SQL Functions**

1. **Use of COALESCE to handle NULL values:**

sql

Copy code

SELECT first\_name, COALESCE(manager\_id, 'No Manager') FROM employees;

1. **Use of CASE statement in SELECT:**

sql

Copy code

SELECT first\_name, last\_name,

CASE

WHEN salary < 40000 THEN 'Low'

WHEN salary BETWEEN 40000 AND 60000 THEN 'Medium'

ELSE 'High'

END AS salary\_range

FROM employees;

1. **Use of ROUND function to format numbers:**

sql

Copy code

SELECT first\_name, last\_name, ROUND(salary, 2) FROM employees;

1. **Use of DATE functions (GETDATE, DATEADD):**

sql

Copy code

SELECT first\_name, last\_name, GETDATE() AS current\_date, DATEADD(year, 1, hire\_date) AS next\_year\_anniversary FROM employees;

**10. SQL for Data Analysis**

1. **Calculate the total number of employees per department:**

sql

Copy code

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

1. **Find departments with more than 5 employees:**

sql

Copy code

SELECT department\_id FROM employees GROUP BY department\_id HAVING COUNT(\*) > 5;

1. **Calculate the average salary per job title:**

sql

Copy code

SELECT job\_title, AVG(salary) AS avg\_salary FROM employees GROUP BY job\_title;

1. **Find the top 5 highest-paid employees:**

sql

Copy code

SELECT first\_name, last\_name, salary FROM employees ORDER BY salary DESC LIMIT 5;

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

sql

Copy code

SELECT SUM(salary) OVER () AS total\_salary FROM employees;

1. **Calculate the running total of salaries:**

sql

Copy code

SELECT first\_name, last\_name, salary, SUM(salary) OVER (ORDER BY salary) AS running\_total FROM employees;

1. **Find the rank of employees based on salary:**

sql

Copy code

SELECT first\_name, last\_name, salary, RANK() OVER (ORDER BY salary DESC) AS salary\_rank FROM employees;

1. **Calculate the dense rank of employees based on salary:**

sql

Copy code

SELECT first\_name, last\_name, salary, DENSE\_RANK() OVER (ORDER BY salary DESC) AS dense\_rank FROM employees;

1. **Calculate the difference between current and previous salaries:**

sql

Copy code

SELECT first\_name, last\_name, salary, LAG(salary) OVER (ORDER BY hire\_date) AS previous\_salary, salary - LAG(salary) OVER (ORDER BY hire\_date) AS salary\_difference FROM employees;

1. **Find the lead salary of employees:**

sql

Copy code

SELECT first\_name, last\_name, salary, LEAD(salary) OVER (ORDER BY hire\_date) AS next\_salary FROM employees;

1. **Calculate the percentile rank of employees based on salary:**

sql

Copy code

SELECT first\_name, last\_name, salary, PERCENT\_RANK() OVER (ORDER BY salary DESC) AS percentile\_rank FROM employees;

1. **Find employees who earn above the median salary:**

sql

Copy code

WITH MedianSalary AS (

SELECT salary, PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY salary) OVER () AS median\_salary FROM employees

)

SELECT \* FROM MedianSalary WHERE salary > median\_salary;

**11. More Advanced SQL Queries**

1. **Perform a self-join to find pairs of employees in the same department:**

sql

Copy code

SELECT e1.first\_name, e1.last\_name, e2.first\_name, e2.last\_name

FROM employees e1

JOIN employees e2 ON e1.department\_id = e2.department\_id AND e1.employee\_id != e2.employee\_id;

1. **Find the longest tenured employee in each department:**

sql

Copy code

SELECT department\_id, first\_name, last\_name, hire\_date

FROM (

SELECT department\_id, first\_name, last\_name, hire\_date,

ROW\_NUMBER() OVER (PARTITION BY department\_id ORDER BY hire\_date) AS row\_num

FROM employees

) AS ranked\_employees

WHERE row\_num = 1;

1. **Calculate the year-over-year salary growth of each employee:**

sql

Copy code

SELECT first\_name, last\_name, salary, LAG(salary) OVER (PARTITION BY employee\_id ORDER BY hire\_date) AS previous\_salary,

(salary - LAG(salary) OVER (PARTITION BY employee\_id ORDER BY hire\_date)) / LAG(salary) OVER (PARTITION BY employee\_id ORDER BY hire\_date) \* 100 AS salary\_growth\_percentage

FROM employees;

1. **Find employees with salaries above the department average:**

sql

Copy code

SELECT first\_name, last\_name, salary, department\_id

FROM employees e

WHERE salary > (SELECT AVG(salary) FROM employees WHERE department\_id = e.department\_id);

1. **Find employees with salaries above the company average:**

sql

Copy code

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

1. **Find employees with more than the average salary in their department:**

sql

Copy code

SELECT first\_name, last\_name, salary

FROM employees e

WHERE salary > (SELECT AVG(salary) FROM employees WHERE department\_id = e.department\_id);

1. **Identify employees who joined in the last year:**

sql

Copy code

SELECT \* FROM employees WHERE hire\_date >= DATEADD(year, -1, GETDATE());

1. **Identify departments with a combined salary budget greater than $1,000,000:**

sql

Copy code

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

HAVING SUM(salary) > 1000000;

1. **List employees who have not been assigned a manager:**

sql

Copy code

SELECT \* FROM employees WHERE manager\_id IS NULL;

1. **List employees who are managers:**

sql

Copy code

SELECT \* FROM employees WHERE employee\_id IN (SELECT DISTINCT manager\_id FROM employees WHERE manager\_id IS NOT NULL);

1. **List departments with no employees:**

sql

Copy code

SELECT \* FROM departments WHERE department\_id NOT IN (SELECT DISTINCT department\_id FROM employees);

1. **Find duplicate records in a table:**

sql

Copy code

SELECT first\_name, last\_name, COUNT(\*)

FROM employees

GROUP BY first\_name, last\_name

HAVING COUNT(\*) > 1;

1. **Find the second highest salary in the employees table:**

sql

Copy code

SELECT MAX(salary) AS second\_highest\_salary

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees);

1. **List all employees hired after the most recent employee:**

sql

Copy code

SELECT \* FROM employees WHERE hire\_date > (SELECT MAX(hire\_date) FROM employees);

1. **Identify employees whose salary is within 10% of the highest salary:**

sql

Copy code

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

1. **Find employees who have never been assigned a department:**

sql

Copy code

SELECT \* FROM employees WHERE department\_id IS NULL;

1. **List employees who work in the same department as a specific employee (e.g., 'John Doe'):**

sql

Copy code

SELECT \* FROM employees WHERE department\_id = (SELECT department\_id FROM employees WHERE first\_name = 'John' AND last\_name = 'Doe');

1. **Find the third highest salary using a subquery:**

sql

Copy code

SELECT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 2;

1. **List employees by their job title and the number of employees with that title:**

sql

Copy code

SELECT job\_title, COUNT(\*) AS num\_employees

FROM employees

GROUP BY job\_title;

1. **Calculate the percentage of employees in each department:**

sql

Copy code

SELECT department\_id, COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM employees) AS percentage

FROM employees

GROUP BY department\_id;

1. **Find employees who have been with the company for more than 10 years:**

sql

Copy code

SELECT \* FROM employees WHERE DATEDIFF(year, hire\_date, GETDATE()) > 10;

1. **Identify departments with a salary variance above a certain threshold:**

sql

Copy code

SELECT department\_id, VARIANCE(salary) AS salary\_variance

FROM employees

GROUP BY department\_id

HAVING VARIANCE(salary) > 10000;

1. **List all employees with the same job title as the highest-paid employee:**

sql

Copy code

SELECT \* FROM employees WHERE job\_title = (SELECT job\_title FROM employees ORDER BY salary DESC LIMIT 1);

1. **Find the number of employees in each department and sort by the number of employees descending:**

sql

Copy code

SELECT department\_id, COUNT(\*) AS num\_employees

FROM employees

GROUP BY department\_id

ORDER BY num\_employees DESC;

1. **Find the employees who earn the minimum salary in their department:**

sql

Copy code

SELECT first\_name, last\_name, salary

FROM employees e

WHERE salary = (SELECT MIN(salary) FROM employees WHERE department\_id = e.department\_id);

1. **Find the employee with the most direct reports:**

sql

Copy code

SELECT manager\_id, COUNT(\*) AS num\_reports

FROM employees

GROUP BY manager\_id

ORDER BY num\_reports DESC

LIMIT 1;

1. **List all employees who share the same hire date:**

sql

Copy code

SELECT hire\_date, GROUP\_CONCAT(first\_name || ' ' || last\_name) AS employees

FROM employees

GROUP BY hire\_date

HAVING COUNT(\*) > 1;

1. **Find the average salary by department and round to two decimal places:**

sql

Copy code

SELECT department\_id, ROUND(AVG(salary), 2) AS avg\_salary

FROM employees

GROUP BY department\_id;

1. **List the job titles with the number of employees in each, sorted by the number of employees descending:**

sql

Copy code

SELECT job\_title, COUNT(\*) AS num\_employees

FROM employees

GROUP BY job\_title

ORDER BY num\_employees DESC;