**-- 1. Retrieve the second-highest salary of an employee**

SELECT MAX(salary) AS second\_highest\_salary

FROM Employees

WHERE salary <

( SELECT MAX(salary)

FROM Employees

)

**-- 2. Get the nth highest salary (example: 3rd highest)**

SELECT salary

FROM (

SELECT salary, DENSE\_RANK() OVER (ORDER BY salary DESC) AS rnk

FROM Employees

) AS Ranked

WHERE rnk = 3;

**-- 3. Fetch all employees whose salary is greater than the average salary**

SELECT employee\_id, salary

FROM Employees

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

**-- 4. Display the current date and time**

SELECT CURRENT\_TIMESTAMP AS current\_date\_time;

**-- 5. Find duplicate records in a table (based on employee\_id)**

SELECT employee\_id, COUNT(\*) AS count

FROM Employees

GROUP BY employee\_id

HAVING COUNT(\*) > 1;

**-- 6. Delete duplicate rows (keep one)**

WITH Duplicates AS (

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY employee\_id ORDER BY (SELECT NULL)) AS rn

FROM Employees

)

DELETE FROM Duplicates WHERE rn > 1;

**-- 7. Get common records from two tables**

SELECT employee\_id

FROM Employees

INTERSECT

SELECT employee\_id

FROM Managers;

**-- 8. Retrieve the last 10 records from a table (based on hire\_date)**

SELECT TOP 10 \*

FROM Employees

ORDER BY hire\_date DESC;

**-- 9. Fetch the top 5 employees with the highest salaries**

SELECT TOP 5 employee\_id, salary

FROM Employees

ORDER BY salary DESC;

**-- 10. Calculate the total salary of all employees**

SELECT SUM(salary) AS total\_salary

FROM Employees;

**-- 11. Find all employees who joined in the year 2020**

SELECT employee\_id, hire\_date

FROM Employees

WHERE YEAR(hire\_date) = 2020;

**-- 12. Find employees whose name starts with 'A'**

SELECT employee\_id, first\_name

FROM Employees

WHERE first\_name LIKE 'A%';

**-- 13. Find employees who do not have a manager**

SELECT employee\_id

FROM Employees

WHERE manager\_id IS NULL;

**-- 14. Find the department with the highest number of employees**

SELECT TOP 1 department\_id, COUNT(\*) AS employee\_count

FROM Employees

GROUP BY department\_id

ORDER BY employee\_count DESC;

**-- 15. Get the count of employees in each department**

SELECT department\_id, COUNT(\*) AS employee\_count

FROM Employees

GROUP BY department\_id;

**-- 16. Employees with highest salary in each department**

SELECT department\_id, employee\_id, salary

FROM (

SELECT \*, RANK() OVER (PARTITION BY department\_id ORDER BY salary DESC) AS rk

FROM Employees

) AS Ranked

WHERE rk = 1;

**-- 17. Update the salary of all employees by 10%**

UPDATE Employees

SET salary = salary \* 1.10;

**-- 18. Find employees whose salary is between 50,000 and 100,000**

SELECT employee\_id, salary

FROM Employees

WHERE salary BETWEEN 50000 AND 100000;

**-- 19. Find the youngest employee**

SELECT TOP 1 employee\_id, birth\_date

FROM Employees

ORDER BY birth\_date DESC;

**-- 20. Fetch the first and last record from a table**

(SELECT TOP 1 \* FROM Employees ORDER BY hire\_date ASC)

UNION ALL

(SELECT TOP 1 \* FROM Employees ORDER BY hire\_date DESC);

**-- 21. Find employees reporting to a specific manager (e.g., manager\_id = 200)**

SELECT employee\_id, first\_name

FROM Employees

WHERE manager\_id = 200;

**-- 22. Find the total number of departments**

SELECT COUNT(DISTINCT department\_id) AS total\_departments

FROM Employees;

**-- 23. Department with the lowest average salary**

SELECT TOP 1 department\_id, AVG(salary) AS avg\_salary

FROM Employees

GROUP BY department\_id

ORDER BY avg\_salary ASC;

**-- 24. Delete all employees from a department (e.g., dept 101)**

DELETE FROM Employees

WHERE department\_id = 101;

**-- 25. Employees in the company for more than 5 years**

SELECT employee\_id, hire\_date

FROM Employees

WHERE DATEDIFF(YEAR, hire\_date, GETDATE()) > 5;

**-- 26. Second-largest value from a table (e.g., second-highest salary)**

SELECT MAX(salary) AS second\_highest

FROM Employees

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

**-- 27. Remove all records but keep the table structure**

DELETE FROM Employees;

**-- 28. Get all employee records in XML format**

SELECT \* FROM Employees FOR XML AUTO, ELEMENTS;

**-- 29. Get the current month’s name**

SELECT DATENAME(MONTH, GETDATE()) AS current\_month;

**-- 30. Convert a string to lowercase**

SELECT LOWER(first\_name) AS lower\_first\_name

FROM Employees;

**-- 31. Employees with no subordinates**

SELECT e.employee\_id

FROM Employees e

LEFT JOIN Employees m ON e.employee\_id = m.manager\_id

WHERE m.employee\_id IS NULL;

**-- 32. Total sales per customer (assumes Customer and Sales table)**

SELECT customer\_id, SUM(sales\_amount) AS total\_sales

FROM Sales

GROUP BY customer\_id;

**-- 33. Check if a table is empty**

SELECT CASE WHEN EXISTS (SELECT 1 FROM Employees)

THEN 'Table is not empty'

ELSE 'Table is empty' END AS table\_status;

**-- 34. Second highest salary per department**

SELECT department\_id, employee\_id, salary

FROM (

SELECT \*, RANK() OVER (PARTITION BY department\_id ORDER BY salary DESC) AS rk

FROM Employees

) AS Ranked

WHERE rk = 2;

**-- 35. Employees with salary a multiple of 10,000**

SELECT employee\_id, salary

FROM Employees

WHERE salary % 10000 = 0;

**-- 36. Fetch records where a column has null values (e.g., bonus)**

SELECT \*

FROM Employees

WHERE bonus IS NULL;

**-- 37. Count employees in each job title**

SELECT job\_title, COUNT(\*) AS employee\_count

FROM Employees

GROUP BY job\_title;

**-- 38. Fetch employees whose names end with ‘n’**

SELECT employee\_id, first\_name

FROM Employees

WHERE first\_name LIKE '%n';

**-- 39. Employees who work in both departments 101 and 102**

SELECT employee\_id

FROM Employees

WHERE department\_id IN (101, 102)

GROUP BY employee\_id

HAVING COUNT(DISTINCT department\_id) = 2;

**-- 40. Employees with the same salary**

SELECT employee\_id, salary

FROM Employees

WHERE salary IN (

SELECT salary

FROM Employees

GROUP BY salary

HAVING COUNT(\*) > 1

);

**-- 41. Update salaries based on department (e.g., 10% increase for dept 101)**

UPDATE Employees

SET salary = salary \* 1.10

WHERE department\_id = 101;

**-- 42. List employees without a department**

SELECT employee\_id

FROM Employees

WHERE department\_id IS NULL;

**-- 43. Max and min salary in each department**

SELECT department\_id, MAX(salary) AS max\_salary, MIN(salary) AS min\_salary

FROM Employees

GROUP BY department\_id;

**-- 44. Employees hired in last 6 months**

SELECT employee\_id, hire\_date

FROM Employees

WHERE hire\_date >= DATEADD(MONTH, -6, GETDATE());

**-- 45. Department-wise total and average salary**

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

FROM Employees

GROUP BY department\_id;

**-- 46. Employees who joined in the same month/year as their manager**

SELECT e.employee\_id, e.hire\_date, m.hire\_date AS manager\_hire\_date

FROM Employees e

JOIN Employees m ON e.manager\_id = m.employee\_id

WHERE MONTH(e.hire\_date) = MONTH(m.hire\_date) AND YEAR(e.hire\_date) = YEAR(m.hire\_date);

**-- 47. Count employees whose names start and end with the same letter**

SELECT COUNT(\*) AS matching\_names

FROM Employees

WHERE LEFT(LOWER(first\_name), 1) = RIGHT(LOWER(first\_name), 1);

**-- 48. Employee names and salaries in a single string**

SELECT first\_name + ' ' + last\_name + ' earns ' + CAST(salary AS VARCHAR) AS info

FROM Employees;

**-- 49. Employees earning more than their manager**

SELECT e.employee\_id, e.salary, e.manager\_id, m.salary AS manager\_salary

FROM Employees e

JOIN Employees m ON e.manager\_id = m.employee\_id

WHERE e.salary > m.salary;

**-- 50. Employees in departments with less than 3 people**

SELECT employee\_id, department\_id

FROM Employees

WHERE department\_id IN (

SELECT department\_id

FROM Employees

GROUP BY department\_id

HAVING COUNT(\*) < 3

);

**-- 51. Employees with the same first name**

SELECT first\_name, COUNT(\*) AS name\_count

FROM Employees

GROUP BY first\_name

HAVING COUNT(\*) > 1;

**-- 52. Delete employees in the company for over 15 years**

DELETE FROM Employees

WHERE DATEDIFF(YEAR, hire\_date, GETDATE()) > 15;

**-- 53. Employees under the same manager**

SELECT manager\_id, COUNT(\*) AS report\_count

FROM Employees

GROUP BY manager\_id

HAVING COUNT(\*) > 1;

**-- 54. Top 3 highest-paid employees in each department**

SELECT \*

FROM (

SELECT \*, RANK() OVER (PARTITION BY department\_id ORDER BY salary DESC) AS rk

FROM Employees

) AS Ranked

WHERE rk <= 3;

**-- 55. Employees with over 5 years of experience in each department**

SELECT employee\_id, department\_id, hire\_date

FROM Employees

WHERE DATEDIFF(YEAR, hire\_date, GETDATE()) > 5;

**-- 56. Employees in departments with no hires in the last 2 years**

SELECT \*

FROM Employees

WHERE department\_id IN (

SELECT department\_id

FROM Employees

GROUP BY department\_id

HAVING MAX(hire\_date) < DATEADD(YEAR, -2, GETDATE())

);

**-- 57. Employees earning more than the average of their department**

SELECT \*

FROM Employees e

WHERE salary > (

SELECT AVG(salary)

FROM Employees

WHERE department\_id = e.department\_id

);

**-- 58. Managers with more than 5 subordinates**

SELECT manager\_id, COUNT(\*) AS team\_size

FROM Employees

GROUP BY manager\_id

HAVING COUNT(\*) > 5;

**-- 59. Employee names and hire dates in "Name - MM/DD/YYYY" format**

SELECT first\_name + ' - ' + CONVERT(VARCHAR, hire\_date, 101) AS formatted\_output

FROM Employees;

**-- 60. Employees in the top 10% salary range**

SELECT \*

FROM Employees

WHERE salary >= (

SELECT PERCENTILE\_CONT(0.9) WITHIN GROUP (ORDER BY salary) OVER ()

);

**-- 61. Group employees by age brackets (20-30, 31-40, etc.)**

SELECT employee\_id,

CASE

WHEN age BETWEEN 20 AND 30 THEN '20-30'

WHEN age BETWEEN 31 AND 40 THEN '31-40'

WHEN age BETWEEN 41 AND 50 THEN '41-50'

ELSE '51+'

END AS age\_bracket

FROM (

SELECT employee\_id, DATEDIFF(YEAR, birth\_date, GETDATE()) AS age

FROM Employees

) AS AgeCalc;

**-- 62. Average salary of top 5 paid employees per department**

SELECT department\_id, AVG(salary) AS avg\_top\_5\_salary

FROM (

SELECT department\_id, salary, RANK() OVER (PARTITION BY department\_id ORDER BY salary DESC) AS rk

FROM Employees

) AS Ranked

WHERE rk <= 5

GROUP BY department\_id;

**-- 63. Percentage of employees in each department**

SELECT department\_id,

COUNT(\*) AS emp\_count,

CAST(COUNT(\*) \* 100.0 / SUM(COUNT(\*)) OVER () AS DECIMAL(5,2)) AS percentage

FROM Employees

GROUP BY department\_id;

**-- 64. Employees with emails from '@example.com'**

SELECT employee\_id, email

FROM Employees

WHERE email LIKE '%@example.com';

**-- 65. Year-to-date sales for each customer**

SELECT customer\_id, SUM(sales\_amount) AS ytd\_sales

FROM Sales

WHERE order\_date >= DATEFROMPARTS(YEAR(GETDATE()), 1, 1)

GROUP BY customer\_id;

**-- 66. Hire date and day of the week for each employee**

SELECT employee\_id, hire\_date, DATENAME(WEEKDAY, hire\_date) AS hire\_day

FROM Employees;

**-- 67. Employees older than 30 years**

SELECT employee\_id,

TRY\_CONVERT (Date, birth\_date,103) as Birthdate,

datediff (year, TRY\_CONVERT (Date, birth\_date,103), GETDATE()) as Age

FROM Employees

WHERE DATEDIFF(YEAR, TRY\_CONVERT (Date, birth\_date,103), GETDATE()) > 30;

**-- 68. Group employees by salary range (0–20K, 20K–50K, etc.)**

SELECT employee\_id,

salary,

CASE

WHEN salary BETWEEN 0 AND 50000 THEN '0-50K'

WHEN salary BETWEEN 50001 AND 75000 THEN '50K-75K'

ELSE '75K+'

END AS salary\_range

FROM Employees;

**-- 69. Employees without a bonus**

SELECT employee\_id

FROM Employees

WHERE bonus IS NULL;

**-- 70. Highest, lowest, and average salary for each job role**

SELECT job\_title,

MAX(salary) AS max\_salary,

MIN(salary) AS min\_salary,

AVG(salary) AS avg\_salary

FROM Employees

GROUP BY job\_title;