SQL Interview Questions & Solutions

Q1.

Table: Employee

Column Name	Type
id	int
name	varchar
department	varchar
mentorId	int

id is the primary key column for this table. Each row of this table indicates the name of an employee, their department, and the id of their mentor. If mentorId is null, then the employee does not have a mentor. No employee will be the mentor of themself.
Write an SQL query to report the mentors with at least four direct reports. Return the result table in any order.

Sol1.

```
select name from employee
where id in
(select mentorId from Employee
group by mentorId
having count(mentorId)>=4);
```

Q2- 20

Table: EmpDetails

id	name	managerId	joiningDate	location
12	denis	14	04/03/2021	banglore
13	peter	46	11/08/2019	mumbai
14	kate	78	12/11/2019	Japan

Table: EmpSalaries

id	task	salary	bonus
12	t1	12000	300
13	t2	9000	1200
14	t1	10500	0

Q2. Write an SQL query to fetch all the Employee details from the EmpDetails table who joined in the Year 2019.

Sol

```
SELECT * FROM EmpDetails
WHERE joiningDate BETWEEN '2019/01/01'
AND '2019/12/31';
```

Q3. Write an SQL query to fetch the task-wise count of employees sorted by task's count in descending order.

Sol.

```
SELECT task, count(id) EmpTaskCount
FROM EmpSalaries
GROUP BY task
ORDER BY EmpTaskCount DESC;
```

Q4. Write an SQL query to fetch employee names having a salary greater than or equal to 4000 and less than or equal to 10000.

Sol.

```
SELECT name
FROM EmpDetails
WHERE id IN
(SELECT EmpId FROM EmpSalaries
```

```
WHERE Salary BETWEEN 4000 AND 10000);
```

Q5. Write an SQL query to fetch all the Employees who are also managers from the EmpDetails table.

Sol.

```
SELECT DISTINCT E.name
FROM EmpDetails E
INNER JOIN EmpDetails M
ON E.id = M.id;
```

Q6. Write an SQL query to fetch duplicate records from EmpDetails (without considering the primary key – id).

Sol.

```
SELECT name, managerId, joiningDate, location, COUNT(*)
FROM empDetails
GROUP BY name, managerId, joiningDate, location
HAVING COUNT(*) > 1;
```

Q7. Write an SQL query to fetch only odd rows from the table.

Sol.

```
SELECT * FROM EmpDetails
WHERE MOD (id, 2) <> 0;
```

Q8. Write an SQL query to fetch only even rows from the table.

Sol.

```
SELECT * FROM EmpDetails
WHERE MOD (id, 2) = 0;
```

Q9. Write an SQL query to fetch top 2 records.

Sol.

```
SELECT *
FROM EmpSalaries
ORDER BY salary DESC LIMIT 2;
```

Q10. Write an SQL query to find the 2nd highest salary from a table.

Sol.

```
SELECT salary
FROM Employee
ORDER BY salary DESC LIMIT 1,1;
```

Q11. Write an SQL query to fetch all employee records from the EmpDetails table who have a salary record in the EmpSalaries table.

Sol.

```
SELECT * FROM EmpDetails E
WHERE EXISTS
(SELECT * FROM EmpSalaries S
WHERE  E.id = S.id);
```

Q12. Find out all the employees who are not working on any task.

Sol.

```
SELECT id
FROM EmpSalaries
WHERE task IS NULL;
```

Q13. Write an SQL query to find the count of the total occurrences of a particular character – ‘n’ in the name field.

Sol.

```
SELECT name,
LENGTH(name) - LENGTH(REPLACE(name, 'n', ''))
FROM EmpDetails;
```

Q14. Write an SQL query to fetch the position of a given character(s) in the name field.

Sol.

```
SELECT INSTR(name, 'ter')
FROM EmpDetails;
```

Q15. Write an SQL query to fetch all the ids which are present in either of the tables – ‘EmpDetails’ and ‘EmpSalaries’.

Sol.

```
SELECT id FROM EmpDetails
UNION
SELECT id FROM EmpSalaries;
```

Q16. Write an SQL query to fetch the employees whose name begins with any two characters, followed by a text “te” and ends with any sequence of characters.

Sol.

```
SELECT name
FROM EmpDetails
WHERE name LIKE '__te%';
```

Q17. Write an SQL query to display the total salary of each employee adding the Salary with bonus value.

Sol.

```
SELECT id,
salary+bonus as TotalSalary
FROM EmpSalaries;
```

Q18. Write an SQL query to fetch all those employees who work on tasks other than t2.

Sol.

```
SELECT id
FROM EmpSalaries
WHERE task <> 't2';
```

Q19. Write an SQL query to fetch all the employees who either live in Banglore or work under a manager with managerId – 46.

Sol.

```
SELECT id, location, managerId
FROM EmpDetails
WHERE location='Banglore' OR managerId='46';
```

Q20. Write an SQL query to find the maximum, minimum, and average salary of the employees.

Sol.

```
SELECT Max(salary),
Min(salary),
AVG(salary)
FROM EmpSalaries;
```

Q21-23

Table Teacher

id	subjectId	fieldId
1	2	1
1	3	2
1	4	3
3	2	3
2	5	2
3	6	1
2	1	2

Q21. Write an SQL query to find the number of unique subjects each teacher teaches in the school.

Sol.

```
SELECT id,COUNT(DISTINCT subjectId) as cnt FROM Teacher
GROUP BY id;
```

Q22. find the number of teachers in the school

Sol.

```
SELECT COUNT(DISTINCT id) as cnt
FROM Teacher;
```

Q23. Find the number of subjects taught in the school

Sol.

```
SELECT COUNT(DISTINCT subjectId) as cnt
FROM Teacher;
```

Q24-25

Table Patients

patientId	patientName	gender	age	city	doctorId
1	rakesh	M	23	kerala	121
2	suman	F	25	sikar	139
3	riya	F	46	jaipur	121
4	karan	M	18	delhi	146

Table PatientsResult:

patientId	BP	weight	consultationFee
1	119/80	67	400
2	143/76	56	600
3	123/83	48	900
4	135/67	58	550

Q24. Find the 3rd highest consultationFee

Sol.

```sql
SELECT consultationFee
FROM PatientsResult
ORDER BY consultationFee DESC LIMIT 2,1;
```

Q25. Find the 3rd highest consultationFee without using Limit/Top keywords.

Sol.

```sql
SELECT consultationFee
FROM PatientsResult r1
WHERE 2 = (
      SELECT COUNT( DISTINCT ( r2.consultationFee ) )
      FROM PatientsResult r2
      WHERE r2.consultationFee > r1.consultationFee );
```

Department Table

Employees

employee_id	employee_name	salary	department_id
1	John Smith	5000.00	1
2	Mary Jhonson	5500.00	1
3	Robert Davis	6000.00	2
4	Jennifer Wilson	4800.00	2

5	Michael Thompson	7000.00	3
6	David Lee	4500.00	4
7	Sarah Clark	6200.00	4

Departments

Department_id	Department_name
1	Sales
2.	Marketing
3.	Finance
4.	Human Resources
5.	Operations

Q26 Write a query to retrieve the department names along with the total number of employees in each department.

```
SELECT department_name, COUNT(*) AS total_employees
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name;
```

Q.27 Write a query to find the department with the highest average salary among its employees.

```sql
SELECT department_name, AVG(salary) AS average_salary
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name
ORDER BY average_salary DESC
LIMIT 1;
```

Q.28 Write a query to calculate the total salary expense for each department.

```sql
SELECT department_name, SUM(salary) AS total_salary_expense
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name;
```

Q.29 Write a query to calculate the total salary expense for each department

```sql
SELECT department_name, COUNT(*) AS employee_count
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name
HAVING COUNT(*) = (
    SELECT COUNT(*)
    FROM employees
    GROUP BY department_id
    ORDER BY COUNT(*) DESC
    LIMIT 1
);
```

Q.30 Write a query to retrieve the department names along with the names of employees who have the highest salary in each department.

```sql
SELECT d.department_name, e.employee_name, e.salary
FROM departments d
JOIN employees e ON d.department_id = e.department_id
WHERE (e.department_id, e.salary) IN (
    SELECT department_id, MAX(salary)
    FROM employees
    GROUP BY department_id
);
```

Q31 Write a query to find the departments that have no employees assigned to them.

```sql
SELECT department_name
FROM departments
WHERE department_id NOT IN (
    SELECT DISTINCT department_id
    FROM employees
);
```

Q32 Write a query to calculate the average salary for employees in each department, including departments with no employees.

```sql
SELECT d.department_name, AVG(e.salary) AS average_salary
FROM departments d
LEFT JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

Q33 Write a query to find the department(s) with the highest employee count whose average salary is above a certain threshold.

```sql
SELECT department_name, COUNT(*) AS employee_count
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name
HAVING AVG(salary) > 5000
ORDER BY COUNT(*) DESC;
```

Q34 Write a query to retrieve the department names along with the total number of employees and the average salary in each department, sorted by the average salary in descending order.

```sql
SELECT d.department_name, COUNT(*) AS total_employees, AVG(e.salary) AS average_salary
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name
ORDER BY average_salary DESC;
```

Q35 Write a query to calculate the percentage of the total salary expense contributed by each department.

```sql
SELECT d.department_name, SUM(e.salary) / (SELECT SUM(salary) FROM employees) * 100 AS percentage_salary
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

Q36 Write a query to find the department(s) with the highest salary expense, excluding the department(s) that have fewer than 3 employees.

```sql
SELECT department_name, SUM(salary) AS total_salary_expense
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name
HAVING COUNT(*) >= 3
ORDER BY total_salary_expense DESC;
```

Q37 Write a query to retrieve the names of employees who have a salary higher than the average salary of their department.

```sql
SELECT employee_name
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = employees.department_id
);
```

Q38 Write a query to find the department(s) where the sum of the salaries of all employees is greater than the sum of the salaries of employees in any other department.

```sql
SELECT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name
HAVING SUM(e.salary) > ALL (
    SELECT SUM(salary)
    FROM employees
    GROUP BY department_id
    WHERE department_id != d.department_id
);
```

Q39 Write a query to calculate the top 3 departments with the highest salary expenses, considering the salaries of employees and their managers as well.

```sql
SELECT d.department_name, SUM(e.salary) AS total_salary_expense
FROM departments d
JOIN employees e ON d.department_id = e.department_id
LEFT JOIN employees m ON d.department_id = m.department_id AND m.employee_id = e.manager_id
GROUP BY d.department_name
ORDER BY total_salary_expense DESC
LIMIT 3;
```

Q40 Write a query to calculate the average salary difference between the highest-paid employee and the lowest-paid employee in each department

```sql
SELECT department_name, MAX(salary) - MIN(salary) AS average_salary_difference
FROM departments
JOIN employees ON departments.department_id = employees.department_id
GROUP BY department_name;
```

Q41 Write a query to find the department(s) with the highest average salary-to-price ratio of products belonging to that department.

```sql
SELECT d.department_name
FROM departments d
JOIN products p ON d.department_id = p.department_id
GROUP BY d.department_name
HAVING AVG(p.price / p.salary) = (
    SELECT MAX(avg_ratio)
    FROM (
        SELECT department_id, AVG(price / salary) AS avg_ratio
        FROM products
        GROUP BY department_id
    ) AS subquery
);
```

Q42 Write a query to retrieve the names of employees who have the same salary as their manager.

```sql
SELECT e.employee_name
FROM employees e
JOIN employees m ON e.department_id = m.department_id AND e.employee_id <> m.employee_id
WHERE e.salary = m.salary;
```

Q43 Write a query to find the department(s) with the highest percentage increase in salary compared to the previous year.

```sql
SELECT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
WHERE EXTRACT(YEAR FROM e.hire_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
GROUP BY d.department_name
HAVING AVG(e.salary) / (
    SELECT AVG(salary)
    FROM employees
    WHERE EXTRACT(YEAR FROM hire_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 2
) > ALL (
    SELECT AVG(e.salary) / AVG(salary)
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    WHERE EXTRACT(YEAR FROM e.hire_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
    GROUP BY d.department_name
    HAVING AVG(salary) <> 0
);
```

Q44 Write a query to calculate the total salary expense for each department, including the salary expenses of employees in its sub-departments.

```sql
WITH RECURSIVE subdepartments AS (
    SELECT department_id
    FROM departments
    WHERE department_id = 1 -- Starting department ID
    UNION ALL
    SELECT d.department_id
```

Q45 Write a query to find the employees who have worked in more than one

department throughout their career.

```sql
SELECT e.employee_name
FROM employees e
JOIN (
    SELECT employee_id
    FROM employees
    GROUP BY employee_id
    HAVING COUNT(DISTINCT department_id) > 1
) sub ON e.employee_id = sub.employee_id;
```

Q46 Write a query to calculate the median salary for each department.

```sql
SELECT department_name, AVG(salary) AS median_salary
FROM (
    SELECT department_name, salary, ROW_NUMBER() OVER (PARTITION BY department_name ORDER BY salary)
    AS rn, COUNT(*) OVER (PARTITION BY department_name) AS c
    FROM departments
    JOIN employees ON departments.department_id = employees.department_id
) sub
WHERE rn IN ((c + 1) / 2, (c + 2) / 2)
GROUP BY department_name;
```

Q47 Write a query to find the department(s) where the average employee salary is higher than the average salary of the employees across all departments.

```sql
SELECT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name
HAVING AVG(e.salary) > (SELECT AVG(salary) FROM employees);
```

Q48 Write a query to find the departments that have at least one employee

whose salary is higher than the maximum salary in any other department.

```sql
SELECT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
WHERE e.salary > ALL (
    SELECT MAX(salary)
    FROM employees
    WHERE department_id != d.department_id
    GROUP BY department_id
);
```

Q49 Write a query to find the departments where the salaries of all employees are within a certain range (e.g., $4000 to $6000).

```sql
SELECT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name
HAVING MIN(e.salary) >= 4000 AND MAX(e.salary) <= 6000;
```

Q50 Write a query to calculate the rank of each employee within their department based on their salary, considering ties.

```sql
SELECT department_name, employee_name, salary,
       DENSE_RANK() OVER (PARTITION BY department_name ORDER BY salary DESC) AS rank
FROM departments d
JOIN employees e ON d.department
```

Project

Id	Name	start_date	end_date	status	budget	department_id
1	A	2022-01-01	2022-06-30	In Progress	10000	1
2	B	2022-02-15	2022-12-31	In Progress	20000	1
3	C	2021-12-01	2022-04-30	Completed	15000	2
4	D	2022-03-10	2023-01-31	In Progress	30000	2
5	E	2022-05-01	2022-08-31	Completed	12000	3

Q51. Write a query to retrieve the project(s) with the longest duration.

```
SELECT project_name
FROM projects
WHERE end_date - start_date = (
    SELECT MAX(end_date - start_date)
    FROM projects
);
```

Q52. Write a query to calculate the total budget allocated to all projects.

```
SELECT SUM(budget) AS total_budget
FROM projects;
```

Q53 Write a query to find the department(s) with the highest total project budget

```
SELECT d.department_name, SUM(p.budget) AS total_budget
FROM departments d
JOIN projects p ON d.department_id = p.department_id
GROUP BY d.department_name
HAVING SUM(p.budget) = (
    SELECT MAX(total_budget)
    FROM (
        SELECT SUM(budget) AS total_budget
        FROM projects
        GROUP BY department_id
    ) AS subquery
);
```

Q54. Write a query to find the project(s) with the highest budget in each department.

```
SELECT d.department_name, p.project_name, p.budget
FROM departments d
JOIN projects p ON d.department_id = p.department_id
WHERE (p.department_id, p.budget) IN (
    SELECT department_id, MAX(budget)
    FROM projects
    GROUP BY department_id
);
```

Q55. Write a query to calculate the average budget of projects in each department.

```
SELECT d.department_name, AVG(p.budget) AS average_budget
FROM departments d
JOIN projects p ON d.department_id = p.department_id
```

```
GROUP BY d.department_name;
```

Q57 Write a query to find the project(s) with the highest budget-to-duration ratio.

```
SELECT project_name
FROM projects
ORDER BY budget / (end_date - start_date) DESC
LIMIT 1;
```

Q.58 Write a query to retrieve the department(s) with the lowest average project duration.

```
SELECT d.department_name
FROM departments d
JOIN projects p ON d.department_id = p.department_id
GROUP BY d.department_name
HAVING AVG(end_date - start_date) = (
    SELECT MIN(average_duration)
    FROM (
        SELECT AVG(end_date - start_date) AS average_duration
        FROM projects
        GROUP BY department_id
    ) AS subquery
);
```

Q.59 Write a query to find the project(s) that started more than 6 months ago and are still ongoing.

```
SELECT project_name
FROM projects
WHERE start_date < CURRENT_DATE - INTERVAL '6 months'
  AND status = 'In Progress';
```

Q.60 Write a query to calculate the total budget spent on completed projects.

```sql
SELECT SUM(budget) AS total_budget_spent
FROM projects
WHERE status = 'Completed';
```

Q.61 Write a query to find the project(s) that exceeded their allocated budget.

```sql
SELECT project_name
FROM projects
WHERE budget < (
    SELECT SUM(amount)
    FROM transactions
    WHERE project_id = projects.project_id
);
```

Q.62 Write a query to retrieve the department(s) with the highest average budget per project.

```sql
SELECT d.department_name
FROM departments d
JOIN (
    SELECT department_id, AVG(budget) AS avg_budget
    FROM projects
    GROUP BY department_id
) AS subquery ON d.department_id = subquery.department_id
GROUP BY d.department_name
HAVING AVG(budget) = (
    SELECT MAX(avg_budget)
    FROM (
```

```
        SELECT AVG(budget) AS avg_budget
        FROM projects
        GROUP BY department_id
    ) AS subquery
);
```

Q.63 Write a query to calculate the average budget of projects started in the last year.

```
SELECT AVG(budget) AS average_budget
FROM projects
WHERE start_date >= CURRENT_DATE - INTERVAL '1 year';
```

Q.64 Write a query to retrieve the project(s) that have a budget higher than the average budget of all projects.

```
SELECT project_name
FROM projects
WHERE budget > (
    SELECT AVG(budget)
    FROM projects
);
```

Q.65 Write a query to find the department(s) with the highest percentage of completed projects.

```
SELECT d.department_name
FROM departments d
JOIN (
    SELECT department_id, COUNT(*) AS total_projects,
           SUM(CASE WHEN status = 'Completed' THEN 1 ELSE 0
END) AS completed_projects
```

```
    FROM projects
    GROUP BY department_id
) AS subquery ON d.department_id = subquery.department_id
WHERE completed_projects = (
    SELECT MAX(completed_projects)
    FROM (
        SELECT COUNT(*) AS completed_projects
        FROM projects
        WHERE status = 'Completed'
        GROUP BY department_id
    ) AS subquery
);
```

Q.66 Write a query to calculate the total budget spent per year.

```
SELECT EXTRACT(YEAR FROM start_date) AS year, SUM(budget) AS
total_budget_spent
FROM projects
GROUP BY year
ORDER BY year;
```

Q.67 Write a query to retrieve the project(s) that have the earliest start date.

```
SELECT project_name
FROM projects
WHERE start_date = (
    SELECT MIN(start_date)
    FROM projects
);
```

Q.68 Write a query to find the project(s) with the highest cumulative budget across multiple years.

```sql
SELECT project_name
FROM (
    SELECT project_name, SUM(budget) AS cumulative_budget
    FROM projects
    GROUP BY project_name
) AS subquery
WHERE cumulative_budget = (
    SELECT MAX(cumulative_budget)
    FROM (
        SELECT SUM(budget) AS cumulative_budget
        FROM projects
        GROUP BY project_name
    ) AS subquery
);
```

Q69 Write a query to calculate the percentage of projects completed in each department.

```sql
SELECT d.department_name, COUNT(*) AS total_projects,
       (COUNT(*) FILTER (WHERE status = 'Completed')) * 100.0
/ COUNT(*) AS completion_percentage
FROM departments d
JOIN projects p ON d.department_id = p.department_id
GROUP BY d.department_name;
```

Q70 Write a query to find the project(s) that have the highest cumulative budget across multiple years, considering only projects that have been completed.

```sql
WITH recursive project_years AS (
  SELECT project_id, EXTRACT(YEAR FROM start_date) AS year,
budget
  FROM projects
```

```
  WHERE status = 'Completed'

  UNION ALL

  SELECT p.project_id, EXTRACT(YEAR FROM p.start_date) AS
year, p.budget + py.budget
  FROM projects p
  JOIN project_years py ON p.project_id = py.project_id AND
EXTRACT(YEAR FROM p.start_date) = py.year + 1
)

SELECT project_id, project_name, cumulative_budget
FROM (
  SELECT project_id, project_name, budget AS
cumulative_budget,
         ROW_NUMBER() OVER (ORDER BY budget DESC) AS rn
  FROM project_years
  WHERE year = (SELECT MIN(year) FROM project_years)
) AS subquery
WHERE rn = 1;
```

Sales Table:

ID	DATE	PRODUCT_ID	QUANTITY	PRICE_PER_UNIT
1	2022-01-01	1	5	10.99
2	2022-01-02	2	3	5.99
3	2022-01-03	1	2	15.99
4	2022-01-03	3	1	5.99

5.	2022-01-04	1	3	10.99

Q71 Write a query to calculate the total revenue generated from sales for each product, considering the quantity sold and the price per unit.

```
SELECT product_id, product_name, SUM(quantity * price) AS
total_revenue
FROM sales
JOIN products ON sales.product_id = products.product_id
GROUP BY product_id, product_name;
```

Q72 Write a query to find the top-selling product based on the total quantity sold.
Solution:

```
SELECT product_id, SUM(quantity) AS total_quantity_sold
FROM sales
GROUP BY product_id
ORDER BY total_quantity_sold DESC
LIMIT 1;
```

Q73 Write a query to retrieve the top 5 products based on the total revenue generated, considering both the quantity sold and price per unit
Solution:

```
SELECT p.product_id, p.product_name, SUM(s.quantity *
p.price) AS total_revenue
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_revenue DESC
LIMIT 5;
```

Q74 Write a query to calculate the average price per unit for each product, assuming each product has a quantity column representing the number of units.
Solution:

```
SELECT product_id, product_name, price / quantity AS
average_price_per_unit
FROM products;
```

Q75 Write a query to find the product(s) with the highest revenue.
Solution:

```
SELECT product_id, SUM(quantity * price_per_unit) AS
total_revenue
FROM sales
GROUP BY product_id
ORDER BY total_revenue DESC
LIMIT 1;
```

Q76 Write a query to retrieve the date(s) with the highest total sales amount
Solution:

```
SELECT sale_date, AVG(price_per_unit) AS
average_price_per_unit
FROM sales
GROUP BY sale_date
HAVING AVG(price_per_unit) = (
    SELECT MAX(average_price_per_unit)
    FROM (
        SELECT sale_date, AVG(price_per_unit) AS
average_price_per_unit
        FROM sales
```

```
        GROUP BY sale_date
    ) AS subquery
);
```

Q77 Write a query to find the product(s) that were sold on the most number of different dates.
Solution:

```
SELECT product_id, COUNT(DISTINCT sale_date) AS
unique_dates_sold
FROM sales
GROUP BY product_id
ORDER BY unique_dates_sold DESC
LIMIT 1;
```

Q78 Write a query to calculate the total revenue generated from sales for each quarter of the year.
Solution:

```
SELECT
    EXTRACT(QUARTER FROM sale_date) AS quarter,
    SUM(quantity * price) AS total_revenue
FROM
    sales
GROUP BY
    quarter;
```

Q79 Write a query to find the product(s) with the highest price per unit among products that have a unit quantity greater than 10.
Solution:

```
SELECT product_id, product_name, price / unit_quantity AS
price_per_unit
```

```
FROM products
WHERE unit_quantity > 10
ORDER BY price_per_unit DESC
LIMIT 1;
```

Q80 Write a query to retrieve the date(s) with the highest total sales amount
Solution:

```
SELECT sale_date
FROM sales
GROUP BY sale_date
HAVING SUM(quantity * price) = (
    SELECT MAX(total_sales)
    FROM (
        SELECT SUM(quantity * price) AS total_sales
        FROM sales
        GROUP BY sale_date
    ) AS subquery
);
```

Q81 Write a query to calculate the total revenue generated from sales for each product by multiplying the quantity sold by the price of the product
Solution:

```
SELECT p.product_id, p.product_name, SUM(s.quantity *
p.price) AS total_revenue
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_id, p.product_name;
```

Q82 Write a query to find the product(s) with the highest average revenue per sale.
Solution:

```sql
SELECT product_id, product_name, AVG(price) AS
average_revenue_per_sale
FROM sales
JOIN products ON sales.product_id = products.product_id
GROUP BY product_id, product_name
HAVING AVG(price) = (
    SELECT MAX(average_revenue_per_sale)
    FROM (
        SELECT product_id, AVG(price) AS
average_revenue_per_sale
        FROM sales
        GROUP BY product_id
    ) AS subquery
);
```

Q83 Write a query to retrieve the top 3 sales dates based on the total revenue generated.
Solution:

```sql
SELECT sale_date, SUM(quantity * price_per_unit) AS
total_revenue
FROM sales
GROUP BY sale_date
ORDER BY total_revenue DESC
LIMIT 3;
```

Q84 Write a query to find the product(s) with the highest average quantity sold per sale.
Solution:

```sql
SELECT product_id, AVG(quantity) AS
average_quantity_sold_per_sale
```

```
FROM sales
GROUP BY product_id
HAVING average_quantity_sold_per_sale = (
    SELECT MAX(average_quantity_sold_per_sale)
    FROM (
        SELECT product_id, AVG(quantity) AS
average_quantity_sold_per_sale
        FROM sales
        GROUP BY product_id
    ) AS subquery
);
```

Q85 Write a query to calculate the total revenue generated from sales for each year.
Solution:

```
SELECT EXTRACT(YEAR FROM sale_date) AS year, SUM(quantity *
price_per_unit) AS total_revenue
FROM sales
GROUP BY year;
```

Q86 Write a query to find the date(s) with the highest total sales quantity.
Solution:

```
SELECT sale_date, SUM(quantity) AS total_quantity_sold
FROM sales
GROUP BY sale_date
HAVING SUM(quantity) = (
    SELECT MAX(total_quantity_sold)
    FROM (
        SELECT sale_date, SUM(quantity) AS
total_quantity_sold
        FROM sales
        GROUP BY sale_date
```

```
    ) AS subquery
);
```

Q87 Write a query to retrieve the top 5 products based on the highest total sales revenue.
Solution:

```
SELECT p.product_id, p.product_name, p.price, SUM(s.quantity
* p.price) AS total_revenue
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_id, p.product_name, p.price
ORDER BY total_revenue DESC
LIMIT 5;
```

Q88 Write a query to calculate the total quantity sold for each product.
Solution:

```
SELECT product_id, SUM(quantity) AS total_quantity_sold
FROM sales
GROUP BY product_id;
```

Q89 Write a query to find the date(s) with the highest total sales revenue.
Solution:

```
SELECT sale_date, SUM(quantity * price_per_unit) AS
total_revenue
FROM sales
GROUP BY sale_date
HAVING SUM(quantity * price_per_unit) = (
    SELECT MAX(total_revenue)
    FROM (
        SELECT sale_date, SUM(quantity * price_per_unit) AS
total_revenue
```

```
        FROM sales
        GROUP BY sale_date
    ) AS subquery
);
```

Q90 Write a query to calculate the total revenue generated from sales for each product, including the product name and revenue, and sort the results in descending order of revenue.
Solution:

```
SELECT p.product_name, SUM(s.quantity * s.unit_price) AS
revenue
FROM products p
JOIN sales s ON p.product_id = s.product_id
GROUP BY p.product_id, p.product_name
ORDER BY revenue DESC;
```

Q91 Write a query to retrieve the date(s) with the highest average quantity sold per sale.
Solution:

```
SELECT sale_date, AVG(quantity) AS
average_quantity_sold_per_sale
FROM sales
GROUP BY sale_date
HAVING AVG(quantity) = (
    SELECT MAX(average_quantity_sold_per_sale)
    FROM (
        SELECT sale_date, AVG(quantity) AS
average_quantity_sold_per_sale
        FROM sales
        GROUP BY sale_date
    ) AS subquery
);
```

Q92 Write a query to find the product(s) with the highest total sales revenue.
Solution:

```
SELECT product_id, SUM(quantity * price_per_unit) AS
total_revenue
FROM sales
GROUP BY product_id
HAVING SUM(quantity * price_per_unit) = (
    SELECT MAX(total_revenue)
    FROM (
        SELECT product_id, SUM(quantity * price_per_unit) AS
total_revenue
        FROM sales
        GROUP BY product_id
    ) AS subquery
);
```

Q93 Write a query to calculate the total sales quantity for each month.
Solution:

```
SELECT EXTRACT(MONTH FROM sale_date) AS month, SUM(quantity)
AS total_quantity_sold
FROM sales
GROUP BY month;
```

Q.94 Write a query to retrieve the product(s) that were sold the most number of times within a specific date range.
Solution:

```
SELECT product_id, product_name, COUNT(*) AS sales_count
FROM sales
WHERE sale_date BETWEEN '2023-01-01' AND '2023-12-31'
```

```
GROUP BY product_id, product_name
ORDER BY sales_count DESC
LIMIT 1;
```

Q.95 Write a query to retrieve the product(s) that were sold on the most number of consecutive days.
Solution:

```
SELECT product_id, COUNT(DISTINCT sale_date) AS
consecutive_days
FROM (
    SELECT product_id, sale_date,
           ROW_NUMBER() OVER (PARTITION BY product_id ORDER
BY sale_date) AS rn
    FROM sales
) AS subquery
GROUP BY product_id, DATEDIFF(day, sale_date, rn)
HAVING consecutive_days = (
    SELECT MAX(consecutive_days)
    FROM (
        SELECT product_id, COUNT(DISTINCT sale_date) AS
consecutive_days
        FROM (
            SELECT product_id, sale_date,
                   ROW_NUMBER() OVER (PARTITION BY product_id
ORDER BY sale_date) AS rn
            FROM sales
        ) AS subquery
        GROUP BY product_id, DATEDIFF(day, sale_date, rn)
    ) AS subquery2
);
```

PRODUCTS TABLE

ID	NAME	CATEGORY	PRICE
1	Product A	Category X	9.99
2	Product B	Category Y	14.99
3	Product C	Category X	12.99
4	Product D	Category Z	13.00
5	Product E	Category Y	8.00
6	Product F	Category Z	6.99

Q.96 Write a query to retrieve the product(s) with the highest price.
Solution:

```sql
SELECT product_id, product_name, price
FROM products
WHERE price = (
    SELECT MAX(price)
    FROM products
);
```

Q.97 Write a query to calculate the total price for all products in each category and output in different table

Solution:

```sql
CREATE TABLE category_total_price AS
SELECT category, SUM(price) AS total_price
FROM products
GROUP BY category;
```

Q.98 Write a query to find the product(s) with the lowest price per category.

Solution:

```sql
SELECT product_id, product_name, category, price
FROM products
WHERE (category, price) IN (
    SELECT category, MIN(price)
    FROM products
    GROUP BY category
);
```

Q.99 Write a query to retrieve the top 3 categories with the highest total price for all products.
Solution:

```sql
SELECT category, SUM(price) AS total_price
FROM products
GROUP BY category
ORDER BY total_price DESC
LIMIT 3;
```

Q.100 Write a query to find the product(s) with the highest price per unit.
Solution:

```sql
 SELECT product_id, product_name, price
FROM products
WHERE price / (
    SELECT MAX(price)
    FROM products
) = 1;
```

Q.101 Write a query to retrieve the product(s) with the highest total price.
Solution:

```sql
SELECT product_id, product_name, price
FROM products
WHERE price * (
    SELECT MAX(price)
    FROM products ) = (
    SELECT MAX(price * (
        SELECT MAX(price)
        FROM products
    ))
    FROM products );
```

Q.102 Write a query to calculate the average price for each category.
Solution:

```sql
SELECT category, AVG(price) AS average_price
FROM products
GROUP BY category;
```

Q.103 Write a query to find the category(s) with the highest number of products.
Solution:

```sql
SELECT category, COUNT(*) AS product_count
FROM products
GROUP BY category
HAVING COUNT(*) = (
    SELECT MAX(product_count)
    FROM (
        SELECT category, COUNT(*) AS product_count
        FROM products
        GROUP BY category
    ) AS subquery
);
```

Q.104 Write a query to retrieve the product(s) with the highest price per category.
Solution:

```
SELECT product_id, product_name, category, price
FROM products
ORDER BY price DESC
LIMIT 5;
```

Q.105 Write a query to find the average price difference between products within the same category.
Solution:

```
SELECT category, AVG(price - (
    SELECT AVG(price)
    FROM products p2
    WHERE p1.category = p2.category
    GROUP BY category
)) AS average_price_difference
FROM products p1
GROUP BY category;
```

Q.106 Write a query to retrieve the top 5 products with the highest price relative to their category.
Solution:

```
SELECT product_id, product_name, category, price
FROM products p1
WHERE price - (
    SELECT AVG(price)
    FROM products p2
    WHERE p1.category = p2.category
    GROUP BY category
) = (
    SELECT MAX(price - (
```

```
        SELECT AVG(price)
        FROM products p2
        WHERE p1.category = p2.category
        GROUP BY category
    ))
    FROM products p3
    WHERE p1.category = p3.category
)
ORDER BY price DESC
LIMIT 5;
```

Q.107 Write a query to calculate the total price difference between the highest-priced and lowest-priced products in each category.
Solution:

```
SELECT category, MAX(price) - MIN(price) AS price_difference
FROM products
GROUP BY category;
```

Q.108 Write a query to retrieve the category(s) with the highest average price.
Solution:

```
SELECT category, AVG(price) AS average_price
FROM products
GROUP BY category
HAVING AVG(price) = (
    SELECT MAX(average_price)
    FROM (
        SELECT category, AVG(price) AS average_price
        FROM products
        GROUP BY category
    ) AS subquery
```

```
);
```

Q.109 Write a query to find the product(s) with the lowest price relative to their average category price.
Solution

```
SELECT product_id, product_name, category, price
FROM products p1
WHERE price - (
    SELECT AVG(price)
    FROM products p2
    WHERE p1.category = p2.category
    GROUP BY category
) = (
    SELECT MIN(price - (
        SELECT AVG(price)
        FROM products p2
        WHERE p1.category = p2.category
        GROUP BY category
    ))
    FROM products p3
    WHERE p1.category = p3.category
);
```

Q.110 Write a query to retrieve the product(s) that belong to the category with the lowest average price.
Solution:

```
SELECT product_id, product_name, category, price
FROM products
WHERE category = (
    SELECT category
    FROM products
    GROUP BY category
```

```
    HAVING AVG(price) = (
        SELECT MIN(average_price)
        FROM (
            SELECT category, AVG(price) AS average_price
            FROM products
            GROUP BY category
        ) AS subquery
    )
);
```