

all major sql query assignment with solution

Published: September 19, 2026 | Index ID: #-

All Major SQL Query Assignment With Solution: A Complete Guide

SQL (Structured Query Language) is the lingua franca of data professionals. Whether you're a student tackling your first assignment or an experienced developer optimizing complex reports, the ability to write clear, efficient queries is essential. In this article, we'll walk through the most common SQL assignment types, provide detailed solutions, and share best practice tips that will help you master the language and impress instructors or managers alike.

Why “All Major SQL Query Assignment With Solution” Matters

Assignments in database courses are designed to test your understanding of core concepts—data retrieval, manipulation, and transformation. By studying a curated set of “all major SQL query assignments with solution,” you can:

- Build a solid foundation in relational algebra.
- Learn how to translate business questions into SQL.
- Identify patterns that recur across different problems.
- Practice writing clean, maintainable code.
- Prepare for real world scenarios such as reporting, analytics, and ETL.

Below, we'll explore each major query category, present sample assignment questions, and provide step by step solutions. By the end, you'll have a toolbox of techniques that will serve you throughout your career.

Table of Contents

1. Basic SELECT Statements
2. Filtering with WHERE, LIKE, IN, and BETWEEN
3. Joins: INNER, LEFT, RIGHT, FULL, and CROSS
4. Aggregations and GROUP BY
5. Subqueries and Correlated Subqueries
6. Set Operators: UNION, INTERSECT, and EXCEPT
7. Common Table Expressions (CTEs)
8. Window Functions
9. Data Manipulation: INSERT, UPDATE, DELETE
10. Advanced Topics: PIVOT, UNPIVOT, and Recursive Queries
11. Query Optimization Tips
12. Conclusion

Basic SELECT Statements

Assignment Example 1: Retrieve All Columns

Task: Return every column from the employees table.

```
SELECT * FROM employees;
```

While using * is quick, it's often better to specify columns explicitly in production code to avoid fetching unnecessary data.

Assignment Example 2: Select Specific Columns

Task: Retrieve the employee's ID, first name, and last name.

```
SELECT employee_id, first_name, last_name  
FROM employees;
```

Filtering with WHERE, LIKE, IN, and BETWEEN

Assignment Example 3: Simple WHERE Clause

Task: Find all employees in the "Sales" department.

```
SELECT employee_id, first_name, last_name  
FROM employees  
WHERE department = 'Sales';
```

Assignment Example 4: LIKE Operator

Task: Retrieve employees whose last name starts with "S".

```
SELECT employee_id, first_name, last_name  
FROM employees  
WHERE last_name LIKE 'S%';
```

Assignment Example 5: IN Clause

Task: Find employees in departments 10, 20, or 30.

```
SELECT employee_id, first_name, last_name, department  
FROM employees  
WHERE department IN (10, 20, 30);
```

Assignment Example 6: BETWEEN Operator

Task: List employees hired between 2015 and 2020.

```
SELECT employee_id, first_name, last_name, hire_date  
FROM employees  
WHERE hire_date BETWEEN '2015-01-01' AND '2020-12-31';
```

Joins: INNER, LEFT, RIGHT, FULL, and CROSS

Assignment Example 7: INNER JOIN

Task: Retrieve employee names and their department names.

```
SELECT e.first_name, e.last_name, d.department_name  
FROM employees e  
INNER JOIN departments d
```

ON e.department_id = d.department_id;

Assignment Example 8: LEFT JOIN

Task: List all departments and any employees they contain, including departments without employees.

```
SELECT d.department_name, e.first_name, e.last_name
FROM departments d
LEFT JOIN employees e
ON d.department_id = e.department_id
ORDER BY d.department_name, e.employee_id;
```

Assignment Example 9: RIGHT JOIN

Task: Show all employees and any associated project assignments, including employees with no projects.

```
SELECT e.first_name, e.last_name, p.project_name
FROM employees e
RIGHT JOIN employee_projects ep
ON e.employee_id = ep.employee_id
LEFT JOIN projects p
ON ep.project_id = p.project_id;
```

Assignment Example 10: FULL OUTER JOIN

Task: Combine customer and order data, ensuring all customers and orders are represented.

```
SELECT c.customer_id, c.customer_name, o.order_id
FROM customers c
FULL OUTER JOIN orders o
ON c.customer_id = o.customer_id
ORDER BY c.customer_id, o.order_id;
```

Assignment Example 11: CROSS JOIN

Task: Generate all possible pairs of products and suppliers.

```
SELECT p.product_name, s.supplier_name
FROM products p
CROSS JOIN suppliers s;
```

Aggregations and GROUP BY

Assignment Example 12: Basic Aggregation

Task: Count the total number of employees.

```
SELECT COUNT(*) AS total_employees
FROM employees;
```

Assignment Example 13: GROUP BY

Task: Determine the average salary per department.

```
SELECT department_id,  
       AVG(salary) AS avg_salary,  
       MIN(salary) AS min_salary,  
       MAX(salary) AS max_salary  
FROM employees  
GROUP BY department_id  
ORDER BY avg_salary DESC;
```

Assignment Example 14: HAVING Clause

Task: Find departments with more than 10 employees.

```
SELECT department_id, COUNT(*) AS employee_count  
FROM employees  
GROUP BY department_id  
HAVING COUNT(*) > 10  
ORDER BY employee_count DESC;
```

Subqueries and Correlated Subqueries

Assignment Example 15: Subquery in SELECT

Task: For each employee, display the number of projects they are assigned to.

```
SELECT e.employee_id,  
       e.first_name,  
       e.last_name,  
       (SELECT COUNT(*) FROM employee_projects ep  
        WHERE ep.employee_id = e.employee_id) AS project_count  
FROM employees e;
```

Assignment Example 16: Correlated Subquery

Task: List employees who earn more than the average salary of their department.

```
SELECT e.employee_id, e.first_name, e.last_name, e.salary, e.department_id  
FROM employees e  
WHERE e.salary > (  
    SELECT AVG(salary)  
    FROM employees  
    WHERE department_id = e.department_id  
);
```

Assignment Example 17: NOT EXISTS

Task: Find customers who have never placed an order.

```
SELECT c.customer_id, c.customer_name
```

```
FROM customers c
WHERE NOT EXISTS (
  SELECT 1 FROM orders o
  WHERE o.customer_id = c.customer_id
);
```

Set Operators: UNION, INTERSECT, and EXCEPT

Assignment Example 18: UNION

Task: Combine lists of product IDs from two suppliers.

```
SELECT product_id FROM supplier_a_products
UNION
SELECT product_id FROM supplier_b_products;
```

Assignment Example 19: UNION ALL

Task: Include duplicates when combining two product lists.

```
SELECT product_id FROM supplier_a_products
UNION ALL
SELECT product_id FROM supplier_b_products;
```

Assignment Example 20: INTERSECT

Task: Find product IDs present in both supplier lists.

```
SELECT product_id FROM supplier_a_products
INTERSECT
SELECT product_id FROM supplier_b_products;
```

Assignment Example 21: EXCEPT

Task: List product IDs from Supplier A that are not supplied by Supplier B.

```
SELECT product_id FROM supplier_a_products
EXCEPT
SELECT product_id FROM supplier_b_products;
```

Common Table Expressions (CTEs)

Assignment Example 22: Recursive CTE for Hierarchies

Task: Retrieve the management chain for a given employee.

```
WITH RECURSIVE ManagementChain AS (
  SELECT employee_id, manager_id, first_name, last_name
  FROM employees
  WHERE employee_id = 101 -- starting employee
```

UNION ALL

```
SELECT e.employee_id, e.manager_id, e.first_name, e.last_name
FROM employees e
INNER JOIN ManagementChain mc
ON e.employee_id = mc.manager_id
)
SELECT * FROM ManagementChain;
```

Assignment Example 23: CTE for Subquery Simplification

Task: Compute the total sales per product and then find products with sales above the average.

```
WITH ProductSales AS (
    SELECT product_id,
           SUM(quantity * unit_price) AS total_sales
    FROM order_items
    GROUP BY product_id
),
AverageSales AS (
    SELECT AVG(total_sales) AS avg_sales
    FROM ProductSales
)
SELECT ps.product_id, ps.total_sales
FROM ProductSales ps
CROSS JOIN AverageSales as
WHERE ps.total_sales > as.avg_sales
ORDER BY ps.total_sales DESC;
```

Window Functions

Assignment Example 24: Row Numbering

Task: Rank employees by salary within each department.

```
SELECT employee_id,
       first_name,
       last_name,
       department_id,
       salary,
       ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary DESC) AS salary_rank
FROM employees
ORDER BY department_id, salary_rank;
```

Assignment Example 25: Running Total

Baca Juga (Artikel Terkait)

- [alko case study solutions maktabore](http://www.aussizzlegal.com/alko-case-study-solutions-maktabore.pdf)

URL: <http://www.aussizzlegal.com/alko-case-study-solutions-maktabore.pdf>

- [alkaloids alkaloids plants tarek ismail kakhia](http://www.aussizzlegal.com/alkaloids-alkaloids-plants-tarek-ismail-kakhia.pdf)

URL: <http://www.aussizzlegal.com/alkaloids-alkaloids-plants-tarek-ismail-kakhia.pdf>

- [alkalize or die pdf](http://www.aussizzlegal.com/alkalize-or-die-pdf.pdf)

URL: <http://www.aussizzlegal.com/alkalize-or-die-pdf.pdf>

- [alkaline material safety data sheet batteries](http://www.aussizzlegal.com/alkaline-material-safety-data-sheet-batteries.pdf)

URL: <http://www.aussizzlegal.com/alkaline-material-safety-data-sheet-batteries.pdf>

Tags: #major #query #assignment #with #solution

