1. **INNER JOIN Question**:
    - **Task**: Write a query to retrieve the first and last names of employees along with their corresponding department names.
    - **Explanation**: An INNER JOIN retrieves records that have matching values in both tables. In this case, we will join the employees table with the departments table based on the department_id to get the names of employees and their departments.
    - **Sample Query**:

```
SELECT employees.first_name, employees.last_name, departments.department_name
FROM employees
INNER JOIN departments ON employees.department_id = departments.department_id;
```

2. **LEFT JOIN Question**:
    - **Task**: Create a query that lists all employees and their total sales amounts. Include employees who do not have any sales records.
    - **Explanation**: A LEFT JOIN returns all records from the left table (in this case, employees) and the matched records from the right table (sales). If there is no match, the result is NULL for the sales fields. This helps us see which employees have sales and which do not.
    - **Sample Query**:

```
SELECT employees.first_name, employees.last_name, sales.sales
FROM employees
LEFT JOIN sales ON employees.department_id = sales.department_id;
```

3. **RIGHT JOIN Question**:
    - **Task**: Write a query to display all departments and their employees. Include departments that do not have any employees.
    - **Explanation**: A RIGHT JOIN returns all records from the right table (departments) and matched records from the left table (employees). If a department has no employees, the employee fields will show NULL, allowing us to see all departments regardless of employee presence.
    - **Sample Query**:

```
SELECT departments.department_name, employees.first_name, employees.last_name
FROM employees
RIGHT JOIN departments ON employees.department_id = departments.department_id;
```

4. **FULL OUTER JOIN Question**:

- **Task**: Write a query using FULL OUTER JOIN to find all employees and departments, ensuring that you include employees without departments and departments without employees.
- **Explanation**: A FULL OUTER JOIN combines the results of both LEFT JOIN and RIGHT JOIN, returning all records from both tables and filling in NULLs where there are no matches. This allows us to see every employee and every department, even if they are unmatched.
- **Sample Query**:

```
SELECT employees.first_name, employees.last_name, departments.department_name
FROM employees
FULL OUTER JOIN departments ON employees.department_id = departments.department_id;
```

5. **Subquery Question**:
   - **Task**: Create a query to find all employees whose salaries are above the average salary of all employees in the company.
   - **Explanation**: A subquery is a query nested inside another query. Here, we first calculate the average salary from the employees table, and then use that result to filter employees whose salaries exceed the average.
   - **Sample Query**:

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

6. **Correlated Subquery Question**:
   - **Task**: Write a query that lists employees who earn more than the average salary of their respective departments.
   - **Explanation**: A correlated subquery refers to the outer query in its execution. In this example, we will calculate the average salary for each department within the subquery and compare each employee's salary to that average.
   - **Sample Query**:

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

7. **NATURAL JOIN Question**:
    - **Task**: Create a query that uses a NATURAL JOIN to combine the employees and departments tables and displays the first name, last name, and department name.
    - **Explanation**: A NATURAL JOIN automatically joins tables based on columns with the same name. In this case, it will match based on department_id. It simplifies queries when you know that the tables share common columns.
    - **Sample Query**:

```
SELECT *
FROM employees
NATURAL JOIN departments;
```

8. **Name Conflict Question**:
    - **Task**: Write a query to retrieve the first name and last name of employees along with their department names. Ensure that you handle any potential name conflicts using aliases.
    - **Explanation**: When columns have the same name in both tables (like department_id), we need to use aliases to avoid ambiguity. This makes it clear which table each column is coming from.
    - **Sample Query**:

```
SELECT e.first_name AS EmployeeFirstName, e.last_name AS EmployeeLastName,
d.department_name AS DepartmentName
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

Exercise:

- **Task**: Choose one of the questions above and write the SQL query in your environment. Experiment by adding WHERE clauses or modifying the JOIN types to see how the results change.