Oracle SQL Assignment Questions

1. **Basic SELECT Query**

- Write an SQL query to retrieve all columns from the `employees` table.

```
SELECT * FROM employees;
```

2. **Filtering Data**

- Write an SQL query to find all employees who are working in the "Sales" department.

```
SELECT *
FROM employees
WHERE department = 'Sales';
```

3. **Sorting Data**

- Write an SQL query to get the names and salaries of employees in the "Marketing" department, sorted by their salaries in descending order.

```
SELECT name, salary
FROM employees
WHERE department = 'Marketing'
ORDER BY salary DESC;
```

4. **Using Aggregate Functions**

- Write an SQL query to calculate the average salary of employees in the "HR" department.

```
 SELECT AVG(salary) AS avg_salary
FROM employees
WHERE department = 'HR';
```

5. **Group By Clause**

- Write an SQL query to find the total number of employees in each department.

```
SELECT department, COUNT(*) AS total_employees
FROM employees
GROUP BY department;
```

6. **Using DISTINCT**

- Write an SQL query to list all unique job titles from the `employees` table.

```
SELECT DISTINCT job_title
FROM employees;
```

7. **Using LIKE Operator**

- Write an SQL query to retrieve all employees whose names start with the letter "J".

```
SELECT *
FROM employees
WHERE name LIKE 'J%';
```

8. **Using AND/OR Conditions**

- Write an SQL query to find employees who are either in the "IT" department or have a salary greater than $50,000.

```
SELECT *
FROM employees
WHERE department = 'IT' OR salary > 50000;
```

9. **Joining Tables (Inner Join)**

- Write an SQL query to display employee names along with their department names by joining the `employees` and `departments` tables.

```
SELECT e.name, d.department_name
```

```
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

10. **Joining Tables (Left Join)**

- Write an SQL query to display all employees and their department names, including those employees who are not assigned to any department.

```
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

11. **Subqueries**

- Write an SQL query to find employees whose salary is greater than the average salary in the `employees` table.

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

12. **Using IN Operator**

- Write an SQL query to list all employees who belong to the departments "Sales", "Marketing", or "HR".

```
SELECT *
FROM employees
WHERE department IN ('Sales', 'Marketing', 'HR');
```

13. **Using BETWEEN Operator**

- Write an SQL query to find employees whose salaries are between $40,000 and $60,000.

```
SELECT *
```

```
FROM employees
WHERE salary BETWEEN 40000 AND 60000;
```

14. **Using EXISTS**

- Write an SQL query to find departments that have at least one employee with a salary greater than $70,000.

```
SELECT *
FROM departments d
WHERE EXISTS (
    SELECT 1
    FROM employees e
    WHERE e.department_id = d.department_id
    AND e.salary > 70000
);
```

15. **Date Functions**

- Write an SQL query to find all employees who joined after January 1, 2020.

```
SELECT *
FROM employees
WHERE join_date > '2020-01-01';
```

16. **Updating Data**

- Write an SQL query to increase the salary of all employees in the "IT" department by 10%.

```
UPDATE employees
SET salary = salary * 1.10
WHERE department = 'IT';
```

17. **Deleting Data**

- Write an SQL query to delete all employees who are no longer with the company.

```
DELETE FROM employees
WHERE status = 'inactive';
```

18. **Creating a Table**

- Write an SQL query to create a table called `customers` with columns `customer_id`, `first_name`, `last_name`, `email`, and `phone_number`.

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(20)
);
```

19. **Modifying a Table (ALTER)**

- Write an SQL query to add a new column `hire_date` to the `employees` table.

```
ALTER TABLE employees
ADD hire_date DATE;
```

20. **Dropping a Table**

- Write an SQL query to drop the `temporary_employees` table if it exists.

```
DROP TABLE IF EXISTS temporary_employees;
```