SQL SELECT and HAVING Clauses Assignment

Database Schema

- **EMP Table**: empno, ename, sal, job, mgr, comm, deptno, hiredate
- **DEPT Table**: deptno, dname, loc

Questions and Solutions

Q1. Retrieve the names of employees who have a salary greater than 50,000.

```sql
SELECT ename
FROM emp
WHERE sal > 50000;
```

Result: No rows selected (no employees earn more than 50,000)

Q2. List the location and department name for the 'Sales' department.

```sql
SELECT loc, dname
FROM dept
WHERE dname = 'SALES';
```

Result:

LOC	DNAME
CHICAGO	SALES

Q3. Find the number of employees in each department.

```sql
SELECT deptno, COUNT(*) as employee_count
FROM emp
GROUP BY deptno;
```

Result:

DEPTNO	EMPLOYEE_COUNT
10	3
20	4
30	6

Q4. Find the average salary for each department.

```sql
SELECT deptno, AVG(sal) as avg_salary
FROM emp
GROUP BY deptno;
```

Result:

DEPTNO	AVG_SALARY
10	2916.67
20	2443.75
30	1566.67

Q5. List departments with more than 5 employees.

```sql
SELECT deptno, COUNT(*) as employee_count
FROM emp
GROUP BY deptno
HAVING COUNT(*) > 5;
```

Result:

DEPTNO	EMPLOYEE_COUNT
30	6

Q6. What is the difference between WHERE and HAVING clauses?

Answer:

- **WHERE clause**: Filters rows before grouping. Used with individual records.
- **HAVING clause**: Filters groups after GROUP BY operation. Used with aggregate functions.
- WHERE is executed before GROUP BY, HAVING is executed after GROUP BY.

Q7. Find the department with the highest average salary.

```sql
SELECT deptno, avg_sal
FROM (
    SELECT deptno, AVG(sal) as avg_sal
    FROM emp
    GROUP BY deptno
)
WHERE avg_sal = (
    SELECT MAX(AVG(sal))
    FROM emp
    GROUP BY deptno
);
```

Result:

DEPTNO	AVG_SAL
10	2916.67

Q8. List employees whose names start with the letter 'A'.

```sql
SELECT *
FROM emp
WHERE ename LIKE 'A%';
```

Result: ALLEN (and any other employees starting with 'A')

Q9. List departments and their total salary payout where the total is more than 200,000.

```sql
SELECT deptno, SUM(sal) as total_salary
FROM emp
GROUP BY deptno
HAVING SUM(sal) > 200000;
```

Result: No rows selected (no department has total salary > 200,000)

Q10. Display each employee's name and their annual salary (assuming monthly salary), and alias the calculated column as AnnualSalary.

```sql
SELECT ename AS employee_name,
       sal * 12 AS annual_salary
FROM emp;
```

Result: All employees with their annual salaries calculated

Q11. List the name of employees if their name starts with '' *and ends with* ''.

```sql
SELECT ename AS employee_name
FROM emp
WHERE ename LIKE '!_%' AND ename LIKE '%!_' ESCAPE '!';
```

Result: No rows selected (no employees with names starting and ending with underscore)

Q12. Display the most recent hire date in employee table.

```sql
SELECT *
FROM emp
WHERE hiredate = (SELECT MAX(hiredate) FROM emp);
```

Result: SCOTT (hired on 19-APR-87)

Q13. Display employee names in alphabetical order.

```sql
SELECT ename
FROM emp
ORDER BY ename ASC;
```

Result: All employee names sorted alphabetically

Q14. What are wildcard characters?

Answer:

- **% (Percent)**: Matches zero or more characters
- **_ (Underscore)**: Matches exactly one character
- **ESCAPE**: Used to treat wildcard characters as literal characters

---

Q15. Display employees who are doing the same job in the same department.

```sql
SELECT *
FROM emp
WHERE (deptno, job) IN (
    SELECT deptno, job
    FROM emp
    GROUP BY deptno, job
    HAVING COUNT(*) > 1
);
```

Result: Employees with duplicate job-department combinations

Q16. Display repeated salaries.

```sql
SELECT sal, COUNT(*) as frequency
FROM emp
GROUP BY sal
HAVING COUNT(*) > 1;
```

Result: Salaries that appear more than once

Q17. Explain GROUP BY clause.

Answer:

- **GROUP BY clause** is used to group rows that have the same values in specified columns
- It's typically used with aggregate functions (COUNT, SUM, AVG, MAX, MIN)
- Groups are formed first, then aggregate functions are applied to each group
- All non-aggregate columns in SELECT must be included in GROUP BY
- ORDER of execution: WHERE → GROUP BY → HAVING → ORDER BY

Key Learning Points

1. **Aggregate Functions**: COUNT(), SUM(), AVG(), MAX(), MIN()
2. **Grouping**: GROUP BY clause groups similar data
3. **Filtering Groups**: HAVING clause filters grouped data
4. **Wildcards**: % and _ for pattern matching
5. **Subqueries**: Nested queries for complex operations
6. **Column Aliases**: AS keyword for readable output

Assignment completed as part of SQL database course