Section – A: Theory Questions

1. **What is an alias in SQL, and how is it used to rename tables or columns in a query?**
   It is used to assign a temporary name to a table or a column in a query. This temporary name, or alias, can be used throughout the query to reference the renamed table or column.
   Syntax:
   select table1.column1 as column from table1; -- column alias
   select t1.column1 from table1 as t1; -- table alias
   Note: Alias are made especially in the self-join query when the column name or table name is complex to write every time.

2. **What is the purpose of sub-queries in SQL, how do they differ from joins and what is their performance compared to joins?**
   A sub-query in SQL is a query nested inside another query. The purpose of sub-queries is to retrieve data from one or more tables and use this data as a condition to further limit the data retrieved by the main query.

   Joins and sub-queries are two different techniques used to combine data from multiple tables in a database. Joins are used to combine rows from two or more tables based on a related column between them, while sub-queries are used to retrieve data that will be used in the main query as a condition to further limit the data retrieved.

   Joins are generally faster for large data sets and when the relationships between the tables are simple. On the other hand, sub-queries may perform better when the data relationships are complex or when the result set is small.

3. **Explain the syntax of a sub-query in SQL using the keywords "SELECT", "WHERE", and "FROM".**
   Select:
   select column1, (select column2 from table1 where table1.column = table2.column) from table2;
   Where:
   select * from table1 where column in (select column from table2);
   From:
   select * from table1, (select column1, column2*value as column2 from table1) as table2 where table1.column1 = table2.column2

4. **What are the different types of sub-queries in SQL? Explain each type**.
   Single-row sub-query: Returns a single row as the result of the sub-query.
   Multi-row sub-query: Returns multiple rows as the result of the sub-query.
   Multi-column sub-query: Returns multiple columns as the result of the sub-query.
   Correlated sub-query: A sub-query that is dependent on the main query and can access values from the main query.
   Independent sub-query: A sub-query that is executed independently of the main query.
   Nested sub-query: A sub-query that contains another sub-query.

   Example:
   - **Single-row sub-query:** SELECT name, salary FROM employees WHERE salary = (SELECT max(salary) FROM employees); -- query returns the name and salary of the employee with the highest salary
   - **Multi-row sub-query:** SELECT name, department FROM employees WHERE department IN (SELECT department_name FROM departments WHERE location = 'Europe'); -- query returns the name and department of all employees who work in departments located in Europe
   - **Multi-column sub-query:** SELECT name, department, salary FROM employees WHERE (department, salary) = (SELECT department_name, max(salary) FROM employees GROUP BY department_name); -- query returns the name, department, and salary of employees who work in departments with the highest salaries
   - **Correlated sub-query:** SELECT name, department, salary FROM employees e1 WHERE salary > (SELECT AVG(salary) FROM employees e2 WHERE e2.department = e1.department); -- query returns the name, department, and salary of employees who have a salary higher than the average salary for their department
   - **Independent sub-query:** SELECT name, department FROM employees WHERE department IN (SELECT department_name FROM departments); -- query returns the name and department of all employees who work in departments that are listed in the department's table
   - **Nested sub-query:** SELECT name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM (SELECT salary FROM employees WHERE department = 'IT') sub); -- query returns the name and salary of employees who have a salary higher than the average salary for the employees in the IT department

5. **What are the functions used with sub-queries in SQL and how do they work?**
   ALL: returns true if all values in the sub-query meet the condition.
   ANY: returns true if any value in the sub-query meets the condition.
   EXISTS: returns true if any rows are returned by the sub-query.
   NOT EXISTS: returns true if no rows are returned by the sub-query.
   IN: returns true if the value in the main query is found in the result of the sub-query.
   NOT IN: returns true if the value in the main query is not found in the result of the sub-query.

Syntax:
SELECT * FROM orders WHERE total_amount > ALL (SELECT average_amount FROM sales);
SELECT * FROM orders WHERE total_amount > ANY (SELECT average_amount FROM sales);
SELECT * FROM orders WHERE EXISTS (SELECT * FROM sales WHERE sales.order_id = orders.order_id);
SELECT * FROM orders WHERE NOT EXISTS (SELECT * FROM sales WHERE sales.order_id = orders.order_id);
SELECT * FROM orders WHERE customer_id IN (SELECT customer_id FROM customers WHERE condition);
SELECT * FROM orders WHERE customer_id IN (SELECT customer_id FROM customers WHERE condition);

6. **What are the different operators used in SQL with sub-queries and how do they work?**

= (EQUAL): returns true if the value in the main query is equal to the value in the sub-query.
!=, <> (NOT EQUAL): returns true if the value in the main query is not equal to the value in the sub-query.
< (LESS THAN): returns true if the value in the main query is less than the value in the sub-query.
<= (LESS THAN OR EQUAL): returns true if the value in the main query is less than or equal to the value in the sub-query.
> (GREATER THAN): return true if the values in the main query are greater than the value in the sub-query
>= (GREATER THAN OR EQUAL): returns true if the value in the main query is greater than or equal to the value in the sub-query.

Syntax:
SELECT * FROM customers WHERE price = (SELECT max(price) FROM orders);
SELECT * FROM customers WHERE price != (SELECT min(price) FROM orders WHERE order_total = 500);
SELECT * FROM customers WHERE age < (SELECT AVG(age) FROM customers);
SELECT * FROM customers WHERE age <= (SELECT AVG(age) FROM customers);
SELECT * FROM customers WHERE age > (SELECT AVG(age) FROM customers);
SELECT * FROM customers WHERE age >= (SELECT AVG(age) FROM customers);

Section – B: Practice Exercises

-- --
# Dataset Used: employee_details.csv and Department_Details.csv
# Use subqueries to answer every question
-- --

1. **Retrieve employee_id, first_name, last_name, and salary details of those employees whose salary is greater than the average salary of all the employees.(11 Rows)**
   select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY from employee_details
   where SALARY > (select avg(salary) from employee_details);

2. **Display first_name , last_name and department_id of those employee where the location_id of their department is 1700(3 Rows)**
   select FIRST_NAME, LAST_NAME, DEPARTMENT_ID from employee_details
   where DEPARTMENT_ID in (select DEPARTMENT_ID from department_details where LOCATION_ID = 1700);

3. **From the table employees_details, extract the employee_id, first_name, last_name, job_id, and department_id who work in any of the departments of Shipping, Executive, and Finance. (9 Rows)**
   select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID from employee_details
   where DEPARTMENT_ID in (select DEPARTMENT_ID from department_details where DEPARTMENT_NAME in ('Shipping', 'Executive', 'Finance'));

4. **Extract employee_id, first_name, last_name, salary, phone_number, and email of the CLERKS who earn more than the salary of any IT_PROGRAMMER. (3 Rows) (not applicable using code eval)**
   select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, PHONE_NUMBER, EMAIL
   from (select * from employee_details where JOB_ID = 'ST_CLERK') as t
   where t.SALARY > any(select salary from employee_details where employee_details.JOB_ID = 'IT_PROG');

5. **Extract employee_id, first_name, last_name,salary, phone_number, email of the AC_ACCOUNTANTs who earn a salary more than all the AD_VPs.(2 Rows) (not applicable using code eval)**
   select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, PHONE_NUMBER, EMAIL from
   (select * from employee_details where JOB_ID = 'AC_ACCOUNTANT') as t
   where t.salary > all(select SALARY from employee_details where employee_details.JOB_ID = 'AD_VP');

6. **Write a Query to display the employee_id, first_name, last_name, and department_id of the employees who have been recruited after the middle(avg) hire_date. (10 Rows) (not applicable using code eval)**

```
set @mi = (select count(*) from employee_details);

with middle as (select round(if(mod(@mi,2) != 0, (@mi + 1) / 2, ((@mi/2)+((@mi/2)+1))/2))
as c),
    mhire as (select HIRE_DATE, row_number() over(order by HIRE_DATE) as r from
employee_details)

select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, DEPARTMENT_ID from
employee_details
where hire_date > (select HIRE_DATE from mhire, middle where middle.c = mhire.r);
```

7. **Extract employee_id, first_name, last_name, phone_number, salary and job_id of the employees belonging to the 'Contracting' department (3 Rows)**

```
select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, SALARY,
JOB_ID
from employee_details where DEPARTMENT_ID in (select DEPARTMENT_ID from
department_details where DEPARTMENT_NAME = 'Contracting');
```

8. **Extract employee_id, first_name, last_name, phone_number, salary and job_id of the employees who do not belong to the 'Contracting' department(18 Rows)**

```
select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, SALARY,
JOB_ID
from employee_details where DEPARTMENT_ID not in (select DEPARTMENT_ID from
department_details where DEPARTMENT_NAME = 'Contracting');
```

9. **Display the employee_id, first_name, last_name, job_id and department_id of the employees who were recruited first in the department(7 Rows)**

```
select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID, DEPARTMENT_ID from
employee_details
where EMPLOYEE_ID in (select FIRST_VALUE(EMPLOYEE_ID) over w as
first_person_hired
from employee_details
window w as (partition by JOB_ID order by hire_date desc));
```

10. **Display the employee_id, first_name, last_name, salary and job_id of the employees who earn maximum salary for every job.(7Rows)**

```
select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID from
employee_details
```

```
where salary in (select max(salary) over(partition by JOB_ID) as max_salary from
employee_details);
```