

## 1 THEORY QUESTIONS (20 POINTS)

1. Describe the purpose of the following SQL commands, and write down the syntax for each command: SELECT, INSERT, UPDATE, DELETE, JOIN.

SELECT: Used to retrieve data from one or more tables.

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

INSERT: Used to insert new data into a table.

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

UPDATE: Used to modify existing data in a table.

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

DELETE: Used to delete data from a table.

DELETE FROM table\_name

WHERE condition;

JOIN: Used to combine rows from two or more tables based on a related column.

SELECT columns

FROM table1

JOIN table2

ON table1.column = table2.column;

2. Describe the differences between CSV and JSON data formats. When might you choose one format over the other?

- CSV (Comma-Separated Values):

- A data record is any line in the file that has one or more fields separated by commas.
- Ideal for data in tables.
- Simple text editors and spreadsheet applications can handle and interpret them more easily.
- Selected for uncomplicated, flat data structures devoid of hierarchical or nested connections.

- JSON (JavaScript Object Notation):
  - Text format that is simple for computers to understand and produce as well as for humans to read and write.
  - Key-value pairs are used to organize data, and nested and hierarchical data structures are supported.
  - Selected for intricate data structures, particularly with relation to online services or APIs.

## **2 EXPERIMENT (30 POINTS)**

-- Create the database

```
CREATE DATABASE company_db;
```

-- Use the database

```
USE company_db;
```

-- Create departments table

```
CREATE TABLE departments (  
    id INT PRIMARY KEY,  
    department_name VARCHAR(100) NOT NULL  
);
```

-- Create employees table

```
CREATE TABLE employees (  
    id INT PRIMARY KEY,  
    first_name VARCHAR(100) NOT NULL,  
    last_name VARCHAR(100) NOT NULL,  
    department_id INT,  
    salary DECIMAL(10, 2),  
    FOREIGN KEY (department_id) REFERENCES departments(id)  
);
```

-- Insert data into departments table

```
INSERT INTO departments (id, department_name) VALUES  
(1, 'Technology'),  
(2, 'Finance'),  
(3, 'Operations');
```

-- Insert data into employees table

```
INSERT INTO employees (id, first_name, last_name, department_id, salary) VALUES  
(123, 'Benny', 'Wang', 1, 50000.00),
```

(456, 'Quinn', 'Smith', 2, 60000.00),  
(789, 'Bert', 'Johnson', 3, 70000.00);

## --2.1.2 Basic SQL Queries

--Query to retrieve all employees from the employee table

SELECT \* FROM employees;

The screenshot shows a PostgreSQL IDE interface. On the left, the Object Explorer displays the database structure, including the 'employees' table under the 'public' schema. The main query editor contains the following SQL code:

```
--2.1.2 Basic SQL Queries
--Query to retrieve all employees from the employee table
SELECT * FROM employees;

--Insert a new employee into the employee table
INSERT INTO employees (id, first_name, last_name, department_id, salary)
VALUES (369, 'Jackson', 'Brown', 3, 65000.00);
SELECT * FROM employees;

--Update the salary of an employee with a specific ID
UPDATE employees
SET salary = 75000.00
WHERE id = 3;
SELECT * FROM employees;

--Delete an employee with a specific ID
DELETE FROM employees
WHERE id = 4;
SELECT * FROM employees;

--2.1.3 Advanced SQL Queries
```

Below the query editor, the Data Output tab shows the results of the first query, displaying three rows of employee data:

id	first_name	last_name	department_id	salary
123	Benny	Wang	1	50000.00
456	Quinn	Smith	2	60000.00
789	Bert	Johnson	3	70000.00

The status bar at the bottom indicates "Total rows: 3 of 3" and "Query complete 00:00:00.049".

```
--Insert a new employee into the employee table
INSERT INTO employees (id, first_name, last_name, department_id, salary)
VALUES (369, 'Jackson', 'Brown', 3, 65000.00);
SELECT * FROM employees;
```

The screenshot shows a database management tool interface with the following components:

- Object Explorer:** A tree view on the left showing the database structure. The 'company\_db' database is selected, and the 'Tables (2)' folder is expanded, showing 'employees' and 'departments'.
- Query Editor:** The main area for writing SQL queries. It contains the following SQL code:
 

```
--2.1.2 Basic SQL Queries
--Query to retrieve all employees from the employee table
SELECT * FROM employees;

--Insert a new employee into the employee table
INSERT INTO employees (id, first_name, last_name, department_id, salary)
VALUES (369, 'Jackson', 'Brown', 3, 65000.00);
SELECT * FROM employees;

--Update the salary of an employee with a specific ID
UPDATE employees
SET salary = 75000.00
WHERE id = 3;
SELECT * FROM employees;

--Delete an employee with a specific ID
DELETE FROM employees
WHERE id = 4;
SELECT * FROM employees;

--2.1.3 Advanced SQL Queries
```
- Data Output:** A table showing the results of the SQL queries. It has 5 columns: 'id', 'first\_name', 'last\_name', 'department\_id', and 'salary'. The data is as follows:
 

id	first_name	last_name	department_id	salary
1	Benny	Wang	1	50000.00
2	Quinn	Smith	2	60000.00
3	Bert	Johnson	3	70000.00
4	Jackson	Brown	3	65000.00
- Status Bar:** At the bottom, it shows 'Total rows: 4 of 4' and 'Query complete 00:00:00.068'.

--Update the salary of an employee with a specific ID

UPDATE employees

SET salary = 75000.00

WHERE id = 3;

SELECT \* FROM employees;

The screenshot shows a PostgreSQL IDE interface. On the left is the Object Explorer showing the database structure. The main window displays a SQL query with several comments and commands. The query is as follows:

```
--2.1.2 Basic SQL Queries
--Query to retrieve all employees from the employee table
SELECT * FROM employees;

--Insert a new employee into the employee table
INSERT INTO employees (id, first_name, last_name, department_id, salary)
VALUES (369, 'Jackson', 'Brown', 3, 65000.00);
SELECT * FROM employees;

--Update the salary of an employee with a specific ID
UPDATE employees
SET salary = 75000.00
WHERE id = 3;
SELECT * FROM employees;

--Delete an employee with a specific ID
DELETE FROM employees
WHERE id = 4;
SELECT * FROM employees;
```

Below the query window, the Data Output tab shows the result of the last query (SELECT \* FROM employees;). The table has 4 rows and 5 columns: id, first\_name, last\_name, department\_id, and salary.

id	first_name	last_name	department_id	salary
1	Benny	Wang	1	50000.00
2	Quinn	Smith	2	60000.00
3	Bert	Johnson	3	70000.00
4	Jackson	Brown	3	65000.00

At the bottom of the IDE, a status bar indicates: Total rows: 4 of 4, Query complete 00:00:00.046. A green message box at the bottom right says: Successfully run. Total query runtime: 46 msec. 4 rows affected.

--Delete an employee with a specific ID

DELETE FROM employees

WHERE id = 4;

SELECT \* FROM employees;

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays the database structure, including 'Servers (2)', 'HW2', 'Databases (2)', 'company\_db', and 'public' schema. The 'Tables (2)' under 'public' is selected. The main pane shows a SQL query editor with the following content:

```
--2.1.2 Basic SQL Queries
--Query to retrieve all employees from the employee table
SELECT * FROM employees;

--Insert a new employee into the employee table
INSERT INTO employees (id, first_name, last_name, department_id, salary)
VALUES (369, 'Jackson', 'Brown', 3, 65000.00);
SELECT * FROM employees;

--Update the salary of an employee with a specific ID
UPDATE employees
SET salary = 75000.00
WHERE id = 3;
SELECT * FROM employees;

--Delete an employee with a specific ID
DELETE FROM employees
WHERE id = 4;
SELECT * FROM employees;

--2.1.3 Advanced SQL Queries
```

Below the query editor, the 'Data Output' pane shows the results of the last query (SELECT \* FROM employees;):

	id [PK] integer	first_name character varying (100)	last_name character varying (100)	department_id integer	salary numeric (10,2)
1	123	Benny	Wang	1	50000.00
2	456	Quinn	Smith	2	60000.00
3	789	Bert	Johnson	3	70000.00
4	369	Jackson	Brown	3	65000.00

At the bottom, a status bar indicates 'Total rows: 4 of 4' and 'Query complete 00:00:00.048'. A green message box on the right says 'Successfully run. Total query runtime: 48 msec. 4 rows affected.'

### --2.1.3 Advanced SQL Queries

--Find the highest salary in each department

SELECT employees.department\_id, MAX(salary) as highest\_salary

FROM employees

GROUP BY employees.department\_id;

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays the database structure, including 'Servers (2)', 'HW2', 'Databases (2)', 'company\_db', and 'public' schema. The 'Tables (2)' folder is selected. The main pane shows a SQL query editor with the following code:

```
46 UPDATE employees
47 SET salary = 75000.00
48 WHERE id = 3;
49 SELECT * FROM employees;
50
51 --Delete an employee with a specific ID
52 DELETE FROM employees
53 WHERE id = 4;
54 SELECT * FROM employees;
55
56
57 --2.1.3 Advanced SQL Queries
58 --Find the highest salary in each department
59 SELECT employees.department_id, MAX(salary) as highest_salary
60 FROM employees
61 GROUP BY employees.department_id;
62
63 --List employees along with their department names using JOIN
64 SELECT employees.id, employees.first_name, employees.last_name, departments.department_name
65 FROM employees
66 JOIN departments ON employees.department_id = departments.id;
67
```

Below the query editor, the 'Data Output' pane shows the results of the query. The table has two columns: 'department\_id' (integer) and 'highest\_salary' (numeric). The results are as follows:

department_id	highest_salary
1	70000.00
2	60000.00
3	50000.00

The status bar at the bottom indicates 'Total rows: 3 of 3' and 'Query complete 00:00:00.046'. The bottom right corner shows 'Ln 59, Col 1'.

```
--List employees along with their department names using JOIN
SELECT employees.id, employees.first_name, employees.last_name,
departments.department_name, employees.salary
FROM employees
JOIN departments ON employees.department_id = departments.id;
```

The screenshot shows a PostgreSQL IDE interface. On the left is the Object Explorer showing the database structure. The main pane displays a SQL query and its results.

**Query:**

```
46 UPDATE employees
47 SET salary = 75000.00
48 WHERE id = 3;
49 SELECT * FROM employees;
50
51 --Delete an employee with a specific ID
52 DELETE FROM employees
53 WHERE id = 4;
54 SELECT * FROM employees;
55
56
57 --2.1.3 Advanced SQL Queries
58 --Find the highest salary in each department
59 SELECT employees.department_id, MAX(salary) as highest_salary
60 FROM employees
61 GROUP BY employees.department_id;
62
63 --List employees along with their department names using JOIN
64 SELECT employees.id, employees.first_name, employees.last_name, departments.department_name
65 FROM employees
66 JOIN departments ON employees.department_id = departments.id;
67
```

**Data Output:**

id	first_name	last_name	department_name	salary
1	Benny	Wang	Technology	50000.00
2	Quinn	Smith	Finance	60000.00
3	Bert	Johnson	Operations	70000.00
4	Jackson	Brown	Operations	65000.00

Successfully run. Total query runtime: 36 msec. 4 rows affected.

Total rows: 4 of 4 Query complete 00:00:00.036 Ln 64, Col 1