These questions involve a range of SQL and PL/SQL concepts, including table creation, modification, queries, constraints, PL/SQL programming, and ER modeling. Below are structured answers to the questions:

---

**1. EMPLOYEE Table Creation and Modifications**

**a. Add a new column HIREDATE:**

ALTER TABLE EMPLOYEE ADD HIREDATE DATE;

**b. Change datatype of JOB_ID from CHAR to VARCHAR2:**

ALTER TABLE EMPLOYEE MODIFY JOB_ID VARCHAR2(20);

**c. Rename the column Emp_no to E_no:**

ALTER TABLE EMPLOYEE RENAME COLUMN Emp_no TO E_no;

**d. Modify column width of the job field:**

ALTER TABLE EMPLOYEE MODIFY JOB_ID VARCHAR2(30);

**e. Differences between Nested and Correlated Nested Queries:**

- **Nested Query**: Executed independently. Example:
- SELECT * FROM EMPLOYEE WHERE Dept_no IN (SELECT Dept_no FROM DEPARTMENT WHERE Dept_name='SALES');
- **Correlated Nested Query**: Dependent on the outer query. Example:
- SELECT E_name FROM EMPLOYEE e WHERE EXISTS (SELECT 1 FROM DEPARTMENT d WHERE d.Dept_no = e.Dept_no AND d.Dept_name='SALES');

---

**2. EMPLOYEE Table Queries**

**a. Insert 5 rows:**

INSERT INTO EMPLOYEE (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name, Job_id, Salary)

VALUES (1, 'John', 'NY', '1234567890', 10, 'HR', 'HR01', 50000);

-- Repeat for 5 records

**b. Display all information:**

SELECT * FROM EMPLOYEE;

**c. Update city for Emp_no = 12:**

UPDATE EMPLOYEE SET E_address = 'Nagpur' WHERE Emp_no = 12;

**d. List out SQL commands and their structure:**

- **DDL**: CREATE, ALTER, DROP (for schema changes)

- **DML**: SELECT, INSERT, UPDATE, DELETE (for data manipulation)
- **TCL**: COMMIT, ROLLBACK
- **DCL**: GRANT, REVOKE

---

**3. EMPLOYEE Queries**

**a. Employees in MECH:**

SELECT * FROM EMPLOYEE WHERE Dept_name = 'MECH';

**b. Delete email_id of James:**

UPDATE EMPLOYEE SET E_ph_no = NULL WHERE E_name = 'James';

**c. Records in SALES:**

SELECT * FROM EMPLOYEE WHERE Dept_name = 'SALES';

**d. Differences:**

- **Primary Key**: Unique, non-null.
- **Unique Key**: Unique, allows one null.
- **Not Null**: Ensures column always has a value.

---

**4. EMPLOYEE Count and Age Queries**

**a. Count Employee Names:**

SELECT COUNT(E_name) FROM EMPLOYEE;

**b. Maximum age:**

SELECT MAX(Salary) FROM EMPLOYEE;

**c. Minimum age:**

SELECT MIN(Salary) FROM EMPLOYEE;

**d. Types of Notations in ER Diagrams:**

- **Entity**: Rectangle
- **Relationship**: Diamond
- **Attribute**: Oval
- **Key Attribute**: Underlined

---

**5. Grouped and Ordered Salaries**

**a. Grouped Salaries:**

SELECT Dept_name, SUM(Salary) FROM EMPLOYEE GROUP BY Dept_name;

**b. Salaries Ascending:**

SELECT Salary FROM EMPLOYEE ORDER BY Salary ASC;

**c. Salaries Descending:**

SELECT Salary FROM EMPLOYEE ORDER BY Salary DESC;

**6. EMPLOYEE Table Constraints and PL/SQL Concepts**

**a. Insert 3 records and check:**

INSERT INTO EMPLOYEE (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name, Job_id, Salary)

VALUES (101, 'Alice', 'Pune', '9876543210', 5, 'IT', 'DEV01', 70000),

(102, 'Bob', 'Delhi', '8765432109', 6, 'HR', 'HR01', 65000),

(103, 'Charlie', 'Mumbai', '7654321098', 7, 'SALES', 'SLS01', 55000);

**b. Add primary key and not null constraints:**

ALTER TABLE EMPLOYEE ADD CONSTRAINT PK_Emp PRIMARY KEY (Emp_no);

ALTER TABLE EMPLOYEE MODIFY E_name VARCHAR2(20) NOT NULL;

**c. Insert null values and verify:**

INSERT INTO EMPLOYEE (Emp_no, E_name) VALUES (104, NULL); -- This will fail due to NOT NULL constraint

**d. Attribute Definitions:**

- **Derived Attribute**: Value derived from other attributes (e.g., Age from DOB).
- **Composite Attribute**: Made of multiple components (e.g., Full Name = First Name + Last Name).
- **Strong Entity**: Independent entity with its own key.

---

**7. Sailor, Reserves, Boats Table Queries**

**a. Names of sailors who reserved both red and green boats (INTERSECT):**

SELECT S_name FROM Sailors WHERE Boat_color = 'Red'

INTERSECT

SELECT S_name FROM Sailors WHERE Boat_color = 'Green';

**b. Names of sailors who reserved red and green boats (UNION ALL):**

SELECT S_name FROM Sailors WHERE Boat_color = 'Red'

UNION ALL

SELECT S_name FROM Sailors WHERE Boat_color = 'Green';

**c. Names of sailors who reserved boat 103 (EXISTS):**

SELECT S_name FROM Sailors s WHERE EXISTS

(SELECT 1 FROM Reserves r WHERE r.S_id = s.S_id AND r.Boat_id = 103);

**d. Aggregate Functions:**

- **SUM()**: Adds values
- **AVG()**: Averages values
- **MAX()/MIN()**: Finds max/min value
- **COUNT()**: Counts rows Example:
- SELECT MAX(Salary), COUNT(*) FROM EMPLOYEE;

---

**8. Sailors Table Queries with Ratings**

**a. Average age of sailors with rating 10:**

SELECT AVG(S_age) FROM Sailors WHERE Rating = 10;

**b. Name and age of oldest sailor:**

SELECT S_name, S_age FROM Sailors WHERE S_age = (SELECT MAX(S_age) FROM Sailors);

**c. Youngest sailor per rating:**

SELECT Rating, MIN(S_age) AS Youngest_Age FROM Sailors GROUP BY Rating;

**d. Average age per rating with at least two sailors:**

SELECT Rating, AVG(S_age)

FROM Sailors

GROUP BY Rating

HAVING COUNT(*) >= 2;

**e. Normalization (1NF, 2NF):**

- **1NF**: Remove multivalued attributes. Example:
- Name Courses
- Alice Math, Science --> Convert to
- Alice Math
- Alice Science
- **2NF**: Remove partial dependencies. Example: If a table's non-key attribute depends only on part of a composite key, decompose it.

---

**9. Customer and Order Table Joins**

**a. INNER JOIN:**

SELECT c.Customer_name, o.Order_id

FROM Customer c INNER JOIN Orders o

ON c.Customer_id = o.Customer_id;

**b. LEFT OUTER JOIN:**

SELECT c.Customer_name, o.Order_id

FROM Customer c LEFT OUTER JOIN Orders o

ON c.Customer_id = o.Customer_id;

**c. RIGHT OUTER JOIN:**

SELECT c.Customer_name, o.Order_id

FROM Customer c RIGHT OUTER JOIN Orders o

ON c.Customer_id = o.Customer_id;

**d. FULL OUTER JOIN:**

SELECT c.Customer_name, o.Order_id

FROM Customer c FULL OUTER JOIN Orders o

ON c.Customer_id = o.Customer_id;

**e. Triggers:**

- **Definition**: Automatically invoked SQL blocks on specific events like INSERT, UPDATE, DELETE.
- **Example**:
- CREATE OR REPLACE TRIGGER after_update_order
- AFTER UPDATE ON Orders
- FOR EACH ROW
- BEGIN
- INSERT INTO Order_audit (Order_id, Update_date) VALUES (:NEW.Order_id, SYSDATE);
- END;

---

**10. Sailors and Nested Queries**

**a. Names of sailors who reserved red boat (Nested Query):**

SELECT S_name FROM Sailors WHERE S_id IN

(SELECT S_id FROM Reserves WHERE Boat_id IN

(SELECT Boat_id FROM Boats WHERE Color = 'Red'));

**b. Reserved boat 103 (Correlated Nested Query):**

SELECT S_name FROM Sailors s WHERE EXISTS

(SELECT 1 FROM Reserves r WHERE r.S_id = s.S_id AND r.Boat_id = 103);

**c. Oldest sailor:**

SELECT S_name, S_age FROM Sailors WHERE S_age = (SELECT MAX(S_age) FROM Sailors);

**d. Difference: COUNT() vs COUNT(*):**

- **COUNT(Column)**: Counts non-null values.
- **COUNT(*)**: Counts all rows.

**e. Difference: DROP vs DELETE:**

- **DROP**: Removes table schema and data.
- DROP TABLE EMPLOYEE;
- **DELETE**: Removes data but keeps schema.
- DELETE FROM EMPLOYEE WHERE Dept_name = 'HR';

**11. Sailors Table Queries and Constraints**

**a. Sailors with a better rating than Horatio (ANY):**

SELECT S_name FROM Sailors WHERE Rating >

(SELECT Rating FROM Sailors WHERE S_name = 'Horatio');

**b. Sailors with the highest rating (ALL):**

SELECT S_name FROM Sailors WHERE Rating >=

ALL (SELECT Rating FROM Sailors);

**c. Sailors reserved red and green boats (UNION):**

SELECT S_name FROM Sailors WHERE Boat_color = 'Red'

UNION

SELECT S_name FROM Sailors WHERE Boat_color = 'Green';

**d. Delete, Truncate, Drop Commands:**

- **DELETE**: Deletes specific rows:
- DELETE FROM Sailors WHERE S_name = 'John';
- **TRUNCATE**: Deletes all rows:
- TRUNCATE TABLE Sailors;

- **DROP**: Deletes table schema and data:
- DROP TABLE Sailors;

**e. Primary Key vs Unique Key:**

- **Primary Key**: Uniquely identifies each row. Only one allowed per table.
- **Unique Key**: Allows null values, multiple unique keys allowed.

---

**12. Triggers in PL/SQL**

**a. Trigger to insert data:**

```
CREATE OR REPLACE TRIGGER trg_insert_employee
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
  INSERT INTO Audit (Emp_no, Action_date, Action_type)
  VALUES (:NEW.Emp_no, SYSDATE, 'INSERT');
END;
```

**b. Trigger to update data:**

```
CREATE OR REPLACE TRIGGER trg_update_employee
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
  INSERT INTO Audit (Emp_no, Action_date, Action_type)
  VALUES (:NEW.Emp_no, SYSDATE, 'UPDATE');
END;
```

**c. Trigger to delete data:**

```
CREATE OR REPLACE TRIGGER trg_delete_employee
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
  INSERT INTO Audit (Emp_no, Action_date, Action_type)
  VALUES (:OLD.Emp_no, SYSDATE, 'DELETE');
END;
```

**d. Attribute Definitions:**

- **Derived Attribute**: Calculated attribute (e.g., Age derived from DOB).
- **Composite Attribute**: Multi-part attribute (e.g., Full Address = Street + City + State).
- **Strong Entity**: Independent entity with a unique key.

---

**13. Procedures in PL/SQL**

**a. Procedure to view specific columns:**

```
CREATE OR REPLACE PROCEDURE view_employee_columns
IS
BEGIN
  SELECT Emp_no, E_name FROM Employee;
END;
```

**b. Modify procedure:**

```
CREATE OR REPLACE PROCEDURE modify_employee_salary (e_no IN NUMBER, new_salary IN NUMBER)
IS
BEGIN
  UPDATE Employee SET Salary = new_salary WHERE Emp_no = e_no;
END;
```

**c. Primary Key vs Unique Key:**

- **Primary Key**: No nulls, single unique identifier.
- **Unique Key**: Allows nulls, ensures data uniqueness.

**d. ER Diagram Notations:**

- **Entity**: Rectangle
- **Attribute**: Oval
- **Relationship**: Diamond

---

**14. PL/SQL Programs**

**a. Cursor Example:**

```
DECLARE
  CURSOR emp_cursor IS SELECT Emp_no, E_name FROM Employee;
```

```
 emp_record emp_cursor%ROWTYPE;
BEGIN
 OPEN emp_cursor;
 LOOP
  FETCH emp_cursor INTO emp_record;
  EXIT WHEN emp_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE(emp_record.Emp_no || ' ' || emp_record.E_name);
 END LOOP;
 CLOSE emp_cursor;
END;
```

**b. Multiplication Program:**

```
DECLARE
 num NUMBER := 5;
 result NUMBER := 1;
BEGIN
 FOR i IN 1..10 LOOP
  result := num * i;
  DBMS_OUTPUT.PUT_LINE(num || ' x ' || i || ' = ' || result);
 END LOOP;
END;
```

---

**15. Additional PL/SQL Programs**

**a. Modify Procedure to View Columns:**

```
CREATE OR REPLACE PROCEDURE view_specific_columns (cols IN VARCHAR2)
IS
BEGIN
 EXECUTE IMMEDIATE 'SELECT ' || cols || ' FROM Employee';
END;
```

**b. Factorial Program:**

```
DECLARE
 num NUMBER := 5;
```

```
 fact NUMBER := 1;
BEGIN
 FOR i IN 1..num LOOP
  fact := fact * i;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is ' || fact);
END;
```

**c. Trigger to Update:**

```
CREATE OR REPLACE TRIGGER trg_update_audit
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
 INSERT INTO Audit (Emp_no, Action_type, Action_date)
 VALUES (:NEW.Emp_no, 'UPDATED', SYSDATE);
END;
```

---

**16. ER Model Conversion**

**a. Converting ER to Relational:**

- **Professors**: Professor(SSN, Name, Age, Rank, Specialty)
- **Projects**: Project(P_no, Sponsor, Start_date, End_date, Budget)
- **Relationships**:
    - Manages(P_no, SSN)
    - Works_on(P_no, SSN)

**17. Converting ER Model to Relational Model**

**Representing Entities and Relationships:**

1. **Professors Table:**
2. CREATE TABLE Professors (
3. SSN VARCHAR2(11) PRIMARY KEY,
4. Name VARCHAR2(50),
5. Age NUMBER(3),
6. Rank VARCHAR2(20),

7. Specialty VARCHAR2(50)
8. );
9. **Projects Table:**
10. CREATE TABLE Projects (
11. P_no NUMBER PRIMARY KEY,
12. Sponsor VARCHAR2(50),
13. Start_date DATE,
14. End_date DATE,
15. Budget NUMBER(12, 2)
16. );
17. **Relationships:**
    - **Manages:**
    - CREATE TABLE Manages (
    - P_no NUMBER,
    - SSN VARCHAR2(11),
    - PRIMARY KEY (P_no, SSN),
    - FOREIGN KEY (P_no) REFERENCES Projects(P_no),
    - FOREIGN KEY (SSN) REFERENCES Professors(SSN)
    - );
    - **Works_On:**
    - CREATE TABLE Works_On (
    - P_no NUMBER,
    - SSN VARCHAR2(11),
    - PRIMARY KEY (P_no, SSN),
    - FOREIGN KEY (P_no) REFERENCES Projects(P_no),
    - FOREIGN KEY (SSN) REFERENCES Professors(SSN)
    - );

---

**18. SQL Queries for Students, Faculty, and Courses**

**a. Drop a column in Students Table:**

ALTER TABLE Students DROP COLUMN login;

**b. Rename Students Table:**

ALTER TABLE Students RENAME TO STUDENT;

**c. Insert 3 Rows into Each Table:**

INSERT INTO Students VALUES ('S001', 'Alice', 'alice@uni.edu', 20, 3.5);

INSERT INTO Students VALUES ('S002', 'Bob', 'bob@uni.edu', 22, 3.7);

INSERT INTO Students VALUES ('S003', 'Charlie', 'charlie@uni.edu', 21, 3.6);

INSERT INTO Faculty VALUES ('F001', 'Dr. Smith', 70000);

INSERT INTO Faculty VALUES ('F002', 'Dr. Taylor', 75000);

INSERT INTO Faculty VALUES ('F003', 'Dr. Brown', 72000);

INSERT INTO Courses VALUES ('C001', 'Database Systems', 3);

INSERT INTO Courses VALUES ('C002', 'Operating Systems', 4);

INSERT INTO Courses VALUES ('C003', 'Networks', 3);

**d. Delete, Truncate, Drop Commands:**

- **Delete**:
- DELETE FROM Students WHERE Age > 21;
- **Truncate**:
- TRUNCATE TABLE Faculty;
- **Drop**:
- DROP TABLE Courses;

**e. Difference Between Primary Key and Unique Key:**

- **Primary Key**: Ensures data uniqueness; no NULLs.
- **Unique Key**: Allows NULL values.

---

**19. Constraints Examples**

**Difference Between Primary Key, Unique Key, and Not Null:**

- **Primary Key**: Combines Unique and Not Null:
- CREATE TABLE Example (
- ID NUMBER PRIMARY KEY
- );

- **Unique Key**: Ensures unique values:
- CREATE TABLE Example (
- Email VARCHAR2(50) UNIQUE
- );
- **Not Null**: Disallows null values:
- CREATE TABLE Example (
- Name VARCHAR2(50) NOT NULL
- );

---

**20. Employee and Department Tables**

**Create Employee Table:**

```
CREATE TABLE EMP (
 EMPNO NUMBER(6) PRIMARY KEY,
 ENAME VARCHAR2(20),
 JOB VARCHAR2(10),
 DEPTNO NUMBER(3),
 SAL NUMBER(7, 2)
);
```

**Create Department Table:**

```
CREATE TABLE DEPT (
 DEPTNO NUMBER(2) PRIMARY KEY,
 DNAME VARCHAR2(10),
 LOC VARCHAR2(10)
);
```

**Queries:**

1. **Insert Record into Dept Table:**
2. INSERT INTO DEPT VALUES (10, 'HR', 'New York');
3. **Display Specific Columns:**
4. SELECT ENAME, JOB FROM EMP;
5. **Delete Lecturer Data:**
6. DELETE FROM EMP WHERE JOB = 'Lecturer';

7. **List Records by Salary in Ascending Order:**
8. SELECT * FROM EMP ORDER BY SAL ASC;
9. **Update Salary for Managers:**
10. UPDATE EMP SET SAL = 14000 WHERE JOB = 'Manager';

---

**21. Modifications in Employee Table**

**Add Experience Column:**

ALTER TABLE EMP ADD EXPERIENCE NUMBER(2);

**Modify Column Width:**

ALTER TABLE EMP MODIFY JOB VARCHAR2(15);

**Create EMP1 Table with Constraints:**

```
CREATE TABLE EMP1 (
  ENAME VARCHAR2(20),
  EMPNO NUMBER(6) CHECK (EMPNO > 100)
);
```

**Drop a Column:**

ALTER TABLE EMP DROP COLUMN EXPERIENCE;

**Rename Column in Dept Table:**

ALTER TABLE DEPT RENAME COLUMN LOC TO LOCATION;

**SQL Command Structures:**

- **SELECT**:
- SELECT column_name FROM table_name WHERE condition;
- **INSERT**:
- INSERT INTO table_name (column1, column2) VALUES (value1, value2);
- **UPDATE**:
- UPDATE table_name SET column1 = value1 WHERE condition;

---

**22. University Database ER to Relational Conversion**

**Entities and Tables:**

1. **Professors**:
2. CREATE TABLE Professors (

3. SSN VARCHAR2(11) PRIMARY KEY,
4. Name VARCHAR2(50),
5. Age NUMBER(3),
6. Rank VARCHAR2(20),
7. Specialty VARCHAR2(50)
8. );
9. **Projects**:
10. CREATE TABLE Projects (
11. P_no NUMBER PRIMARY KEY,
12. Sponsor VARCHAR2(50),
13. Start_date DATE,
14. End_date DATE,
15. Budget NUMBER(12, 2)
16. );

**i. Delete, Truncate, Drop Commands:**

- See **18.d**.

**ii. Primary Key vs Unique Key:**

- See **18.e**.

**23. Attributes of Entities and Relationships**

**Entities and Attributes:**

1. **BUS**:
    - Attributes: Bus_ID (Primary Key), Bus_Number, Capacity, Type, Operator.
2. **TICKET**:
    - Attributes: Ticket_ID (Primary Key), Date_of_Journey, Seat_Number, Price, Bus_ID (Foreign Key).
3. **PASSENGER**:
    - Attributes: Passenger_ID (Primary Key), Name, Contact, Age, Gender.

**Relationships:**

1. **Reservation**:
    - Attributes: Reservation_ID (Primary Key), Passenger_ID (Foreign Key), Ticket_ID (Foreign Key), Date, Status.
2. **Cancellation**:

- Attributes: Cancellation_ID (Primary Key), Reservation_ID (Foreign Key), Date, Reason.

**Keys:**

- **Candidate Key**: A subset of attributes uniquely identifying a record (e.g., Ticket_ID in the TICKET table).
- **Partial Key**: A unique attribute within a weak entity set that requires a foreign key (e.g., Reservation_ID in the RESERVATION table).

---

**b. PL/SQL Program Using Cursor Operations:**

```
DECLARE
  CURSOR reservation_cursor IS
    SELECT Passenger_ID, Ticket_ID FROM Reservation;
  passenger_id Reservation.Passenger_ID%TYPE;
  ticket_id Reservation.Ticket_ID%TYPE;
BEGIN
  OPEN reservation_cursor;
  LOOP
    FETCH reservation_cursor INTO passenger_id, ticket_id;
    EXIT WHEN reservation_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Passenger ID: ' || passenger_id || ', Ticket ID: ' || ticket_id);
  END LOOP;
  CLOSE reservation_cursor;
END;
/
```

---

**24. EMPLOYEE123 Table**

**Table Creation:**

```
CREATE TABLE EMPLOYEE123 (
  EmpID VARCHAR2(10) PRIMARY KEY,
  Name VARCHAR2(15) UNIQUE,
  Job VARCHAR2(10),
  Address VARCHAR2(35),
```

Salary NUMBER(10, 2),

DOJ DATE

);

**a. Insert Records:**

INSERT INTO EMPLOYEE123 VALUES ('E001', 'Alice', 'Manager', 'New York', 75000, '2020-01-01');

INSERT INTO EMPLOYEE123 VALUES ('E002', 'Bob', 'Engineer', 'Los Angeles', 60000, '2019-03-15');

INSERT INTO EMPLOYEE123 VALUES ('E003', 'Charlie', 'Analyst', 'Chicago', 55000, '2021-07-10');

INSERT INTO EMPLOYEE123 VALUES ('E004', 'David', 'Manager', 'Houston', 80000, '2018-05-20');

INSERT INTO EMPLOYEE123 VALUES ('E005', 'Eva', 'HR', 'Seattle', 70000, '2020-09-25');

**b. Update and Delete:**

- **Update**:
- UPDATE EMPLOYEE123 SET Salary = 78000 WHERE EmpID = 'E002';
- **Delete**:
- DELETE FROM EMPLOYEE123 WHERE Job = 'Analyst';

**c. Aggregate Functions:**

- Example:
- SELECT COUNT(*), AVG(Salary), MAX(Salary), MIN(Salary) FROM EMPLOYEE123;

**d. Display Names Starting with 'P':**

SELECT Name FROM EMPLOYEE123 WHERE Name LIKE 'P%';

**e. Cursor Operations:**

Refer to **23.b**.

---

**25. MERCHANT100 Table**

**Table Creation:**

CREATE TABLE MERCHANT100 (

Mer_ID VARCHAR2(10),

Name VARCHAR2(15) UNIQUE,

Age INTEGER,

Budget NUMBER(12, 2) CHECK (Budget >= 100000)

);

**a. Add Address Column:**

ALTER TABLE MERCHANT100 ADD Address VARCHAR2(15);

**b. Modify Address Size:**

ALTER TABLE MERCHANT100 MODIFY Address VARCHAR2(35);

**c. Drop Age Column:**

ALTER TABLE MERCHANT100 DROP COLUMN Age;

**d. Add Primary Key:**

ALTER TABLE MERCHANT100 ADD CONSTRAINT PK_Mer_ID PRIMARY KEY (Mer_ID);

**e. Insert Records:**

INSERT INTO MERCHANT100 VALUES ('M001', 'Alice', 120000, 'New York');

INSERT INTO MERCHANT100 VALUES ('M002', 'Bob', 150000, 'Los Angeles');

INSERT INTO MERCHANT100 VALUES ('M003', 'Charlie', 200000, 'Chicago');

INSERT INTO MERCHANT100 VALUES ('M004', 'David', 175000, 'Houston');

INSERT INTO MERCHANT100 VALUES ('M005', 'Eva', 180000, 'Seattle');

**f. List SQL Commands and Structures:**

Refer to **18.d**.