DBMS LAB EXPERIMENTS

EXPERIMENT 1:

AIM: Creation, altering and dropping of tables and inserting rows into a table using constraints and SELECT command.

Step 1: CREATE TABLE (with Constraints)

Syntax:

```
CREATE TABLE table_name (
  column_name datatype [constraint],
  column_name datatype [constraint]
);
```

Program:

```
SQL> CREATE TABLE student (
  2    sid INT PRIMARY KEY,
  3    name VARCHAR2(20) NOT NULL,
  4    marks INT CHECK (marks BETWEEN 0 AND 100),
  5    email VARCHAR2(30) UNIQUE
  6 );
```

Output:

```
Table student created.
```

Check table structure:

Syntax:

```
DESCRIBE table_name;
```

Program:

```
SQL> DESCRIBE student;
```

Output:

```
Name    Null?    Type
SID     NOT NULL NUMBER(10)
NAME    NOT NULL VARCHAR2(20)
MARKS            NUMBER(10)
EMAIL            VARCHAR2(30)
```

Step 2: INSERT VALUES INTO TABLE

Syntax:

```
INSERT INTO table_name VALUES (value1, value2, ...);
```

Program:

```
SQL> INSERT INTO student VALUES (1, 'Ravi', 85, 'ravi@gmail.com');
SQL> INSERT INTO student VALUES (2, 'Sita', 92, 'sita@gmail.com');
SQL> INSERT INTO student VALUES (3, 'Amit', 78, 'amit@gmail.com');
SQL> INSERT INTO student VALUES (4, 'Neha', 88, 'neha@gmail.com');
SQL> INSERT INTO student VALUES (5, 'Kiran', 95, 'kiran@gmail.com');
```

Output:

```
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
```

Step 3: DISPLAY TABLE DATA (SELECT)

Syntax:

```
SELECT * FROM table_name;
```

Program:

```
SQL> SELECT * FROM student;
```

Output:

```
SID   NAME   MARKS   EMAIL
1     Ravi    85     ravi@gmail.com
2     Sita    92     sita@gmail.com
```

ALTER Command:

ALTER is a DDL (Data Definition Language) command used to modify the structure of an existing table without losing data. It is mainly used to add, modify, or delete columns and constraints of a table.

Step 4: ALTER TABLE (ADD COLUMN)

Syntax:

```
ALTER TABLE table_name ADD column_name datatype;
```

Program:

```
SQL> ALTER TABLE student ADD dept VARCHAR2(20);
```

Output:

```
Table student altered.
```

Check table structure:

```
SQL> DESCRIBE student;
```

Output:

```
Name    Null?    Type
SID     NOT NULL NUMBER(10)
NAME    NOT NULL VARCHAR2(20)
MARKS            NUMBER(10)
EMAIL            VARCHAR2(30)

DEPT             VARCHAR2(20)
```

Step 5: ALTER TABLE (MODIFY COLUMN)

Syntax:

```
ALTER TABLE table_name MODIFY column_name datatype;
```

Program:

```
SQL> ALTER TABLE student MODIFY name VARCHAR2(30);
```

Output:

```
Table student altered.
```

Step 6: ALTER TABLE (DROP COLUMN)

Syntax:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Program:

```
SQL> ALTER TABLE student DROP COLUMN dept;
```

Output:

```
Table student altered.
```

Step 7: ALTER TABLE (RENAME COLUMN)

Syntax:

```
ALTER TABLE table_name RENAME COLUMN old_name TO new_name;
```

Program:

```
SQL> ALTER TABLE student RENAME COLUMN marks TO score;
```

Output:

```
Table student altered.
```

Step 8: ALTER TABLE (RENAME TABLE)

Syntax:

```
ALTER TABLE old_table_name RENAME TO new_table_name;
```

Program:

```
SQL>ALTER TABLE student RENAME TO student_details;
```

Step 9: ALTER TABLE (ADD CONSTRAINT)

Constraint names are required to uniquely identify constraints in a table. If a constraint name is not provided, Oracle generates a system-defined name, which is difficult to remember and manage. Providing a constraint name makes it easy to modify or drop constraints later.

Method 1: ADD constraint WITHOUT constraint name

Syntax:

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
ALTER TABLE table_name ADD UNIQUE (column_name);
```

Program:

```
SQL> ALTER TABLE student ADD PRIMARY KEY (sid);
SQL> ALTER TABLE student ADD UNIQUE (email);
```

Output:

```
Table student altered.
```

Method 2: ADD constraint WITH constraint name

Syntax:

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name constraint_type
(column_name);
```

Program:

```
SQL> ALTER TABLE student ADD CONSTRAINT chk_score CHECK (score >= 0);
SQL> ALTER TABLE student ADD CONSTRAINT unique_email UNIQUE (email);
```

Output:

```
Table student altered.
Table student altered.
```

Step 10: ALTER TABLE (DROP CONSTRAINT)

Method 1: DROP PRIMARY KEY

Syntax:

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

Program:

```
SQL> ALTER TABLE student DROP PRIMARY KEY;
```

Output:

```
Table student altered.
```

Method 2: DROP constraint using constraint name

Syntax:

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Program:

```
SQL> ALTER TABLE student DROP CONSTRAINT chk_score;
```

Output:

```
Table student altered.
```

Step 11: FOREIGN KEY

11a: Inline Foreign Key

Syntax:

```
column_name datatype REFERENCES parent_table(parent_column);
```

Program:

```
SQL> CREATE TABLE dept (
  2    deptid INT PRIMARY KEY,
  3    dname VARCHAR2(20)
  4  );
SQL> INSERT INTO dept VALUES (101, 'CSE');
SQL> INSERT INTO dept VALUES (102, 'ECE');
SQL> CREATE TABLE student (
  2    sid INT PRIMARY KEY,
  3    name VARCHAR2(30) NOT NULL,
  4    score INT,
  5    email VARCHAR2(30) UNIQUE,
  6    deptid INT REFERENCES dept(deptid)  -- inline FK
  7  );
SQL> INSERT INTO student VALUES (1, 'Ravi', 85, 'ravi@gmail.com', 101);
SQL> INSERT INTO student VALUES (2, 'Sita', 92, 'sita@gmail.com', 102);
```

Output:

```
Table dept created.
1 row inserted.
1 row inserted.
```

Check referential integrity:

```
SQL> INSERT INTO student VALUES (3, 'Anita', 88, 'anita@gmail.com', 103);
```

Output:

```
ORA-02291: integrity constraint (SYSTEM.STUDENT_DEPTID_FK) violated - parent
key not found
```

***11b**: Table-Level Foreign Key*

Syntax:

```
CONSTRAINT constraint_name FOREIGN KEY (child_column) REFERENCES
parent_table(parent_column);
```

Program:

```
SQL> CREATE TABLE student (
  2    sid INT PRIMARY KEY,
  3    name VARCHAR2(30) NOT NULL,
  4    score INT,
  5    email VARCHAR2(30) UNIQUE,
  6    deptid INT,
  7    CONSTRAINT fk_dept FOREIGN KEY (deptid) REFERENCES dept(deptid)
  8  );
SQL> INSERT INTO student VALUES (1, 'Ravi', 85, 'ravi@gmail.com', 101);
SQL> INSERT INTO student VALUES (2, 'Sita', 92, 'sita@gmail.com', 102);
SQL> INSERT INTO student VALUES (3, 'Anita', 88, 'anita@gmail.com', 101);
```

Output:

```
Table student created.
1 row inserted.
1 row inserted.
1 row inserted.
```

Check referential integrity:

```
SQL> INSERT INTO student VALUES (4, 'Rohan', 75, 'rohan@gmail.com', 103);
```

Output:

```
ORA-02291: integrity constraint (FK_DEPT) violated - parent key not found
```

Step 12: DROP TABLES

Syntax:

```
DROP TABLE table_name;
```

Program:

```
SQL> DROP TABLE student;
SQL> DROP TABLE dept;
```

Output:

```
Table student dropped.
Table dept dropped.
```
