DMS - Practical - QB - Solutions - By - Th3_

1. Create a table Employee with Emp(EMP_NO, NAME, DEPARTMENT_NO, DEPARTMENT_NAME, JOB_ID, SALARY)

```
CREATE TABLE Employee (
    EMP_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    DEPARTMENT_NO NUMBER(5),
    DEPARTMENT_NAME VARCHAR2(50),
    JOB_ID VARCHAR2(10),
    SALARY NUMBER(10, 2)
);
– Insert values
INSERT INTO Employee VALUES (101, 'John Smith', 10, 'Finance', 'FIN001', 75000.00);

– Display table
SELECT * FROM Employee;
```

2. Create a table for employee, use ALTER and RENAME commands

```
-- Create table
CREATE TABLE Employee (
    EMP_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    DEPARTMENT_NO NUMBER(5),
    DEPARTMENT_NAME VARCHAR2(50),
    SALARY NUMBER(10, 2)
);

– Insert values
INSERT INTO Employee VALUES (101, 'John Smith', 10, 'Finance', 'FIN001', 75000.00);

– Display table
SELECT * FROM Employee;

-- Alter table to add a new column
ALTER TABLE Employee ADD JOIN_DATE DATE;

-- Rename the table
RENAME Employee TO Employee_Details;

– Display renamed table
SELECT * FROM Employee_Details;
```

DMS - Practical - QB - Solutions - By - Th3_

3. Create a table Student, insert values into it

```
CREATE TABLE Student (
    ROLL_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    SUBJECT VARCHAR2(50),
    DEPARTMENT_NAME VARCHAR2(50),
    MARKS NUMBER(5)
);

-- Insert values
INSERT INTO Student VALUES (1, 'John Doe', 'Mathematics', 'Science', 85);
INSERT INTO Student VALUES (2, 'Jane Smith', 'Physics', 'Science', 90);

– Display table
SELECT * FROM Student;
```

4. Create a table for employee, use INSERT and UPDATE commands

```
CREATE TABLE Employee (
    EMP_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    DEPARTMENT_NO NUMBER(5),
    DEPARTMENT_NAME VARCHAR2(50),
    JOB_ID VARCHAR2(10),
    SALARY NUMBER(10, 2)
);

-- Insert values
INSERT INTO Employee VALUES (101, 'Alice', 1, 'HR', 'HR01', 50000);
INSERT INTO Employee VALUES (102, 'Bob', 2, 'Finance', 'FIN01', 60000);

– Display table
SELECT * FROM Employee;

-- Update command
UPDATE Employee SET SALARY = SALARY + 5000 WHERE EMP_NO = 101;
```

DMS - Practical - QB - Solutions - By - Th3_

5. Create a table Employee, insert values, display employees whose salary is between 20000 and 50000

```
CREATE TABLE Employee (
    EMP_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    DEPARTMENT_NO NUMBER(5),
    DEPARTMENT_NAME VARCHAR2(50),
    CITY VARCHAR2(50),
    SALARY NUMBER(10, 2)
);

-- Insert values
INSERT INTO Employee VALUES (1, 'Alice', 1, 'HR', 'New York', 30000);
INSERT INTO Employee VALUES (2, 'Bob', 2, 'Finance', 'Chicago', 25000);
INSERT INTO Employee VALUES (3, 'Clark', 3, 'IT', 'California', 60000);

-- Display employees with salary between 20000 and 50000
SELECT * FROM Employee WHERE SALARY BETWEEN 20000 AND 50000;
```

6. Create a table Student, insert values, display students with marks above 15

```
CREATE TABLE Student (
    Stu_NO NUMBER(5) PRIMARY KEY,
    NAME VARCHAR2(50),
    SUBJECT VARCHAR2(50),
    DEPARTMENT_NAME VARCHAR2(50),
    MARKS NUMBER(5)
);

-- Insert values
INSERT INTO Student VALUES (1, 'Alice', 'Mathematics', 'Science', 20);
INSERT INTO Student VALUES (2, 'Jake', 'Physics', 'Science', 10);

-- Display students with marks above 15
SELECT * FROM Student WHERE MARKS > 15;
```

DMS - Practical - QB - Solutions - By - Th3_

7. Create a table Person, apply INITCAP, LTRIM, and UPPER functions

```
CREATE TABLE Person (
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    ADDRESS VARCHAR2(100),
    CITY VARCHAR2(50)
);

-- Insert values
INSERT INTO Person VALUES ('Alex', 'Zander', ' 123 Main St', 'new york');

-- Apply functions
SELECT
    INITCAP(FIRST_NAME) AS Capitalized_First_Name,
    LTRIM(ADDRESS) AS Trimmed_Address,
    UPPER(CITY) AS Uppercase_City
FROM Person;
```

8. Create a table Person, apply 3 aggregate functions

```
CREATE TABLE Person (
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    SALARY NUMBER(10, 2),
    CITY VARCHAR2(50)
);

-- Insert rows into the updated Person table
INSERT INTO Person VALUES ('Alice', 'Smith', 60000.00, 'Los Angeles');
INSERT INTO Person VALUES ('Bob', 'Brown', 75000.00, 'Chicago');
INSERT INTO Person VALUES ('Clark', 'Williams', 50000.00, 'California');

-- Apply aggregate functions
SELECT
    COUNT(*) AS Total_People,
    MIN(SALARY) AS Minimum_Salary,
    MAX(SALARY) AS Maximum_Salary,
    AVG(SALARY) AS Average_Salary
FROM Person;
```

DMS - Practical - QB - Solutions - By - Th3_

9. PL/SQL code to print the largest of three numbers

```
DECLARE
   num1 NUMBER := 10;
   num2 NUMBER := 20;
   num3 NUMBER := 15;
   largest NUMBER;
BEGIN
   IF num1 > num2 AND num1 > num3 THEN
      largest := num1;
   ELSIF num2 > num3 THEN
      largest := num2;
   ELSE
      largest := num3;
   END IF;
   DBMS_OUTPUT.PUT_LINE('The largest number is: ' || largest);
END;
```

10. PL/SQL code to print the sum of `n` odd numbers using a `FOR` loop

```
DECLARE
   n NUMBER := 5; -- Change this to any number
   sum_odd NUMBER;
BEGIN
   sum_odd := n * n; -- Using the formula
   DBMS_OUTPUT.PUT_LINE('The sum of first ' || n || ' odd numbers is: ' || sum_odd);
END;
```

11. Create a table for Airways, use `ALTER` command

```
CREATE TABLE Airways (
   FLIGHT_NO NUMBER(5) PRIMARY KEY,
   AIRLINE_NAME VARCHAR2(50),
   DESTINATION VARCHAR2(50)
);

-- Alter table to add a new column
ALTER TABLE Airways ADD DEPARTURE_TIME DATE;

– Display table
SELECT * FROM Airways ;
```

DMS - Practical - QB - Solutions - By - Th3_

12. Create a table for Railways, use `INSERT` and `UPDATE` commands

```
CREATE TABLE Railways (
    TRAIN_NO NUMBER(5) PRIMARY KEY,
    TRAIN_NAME VARCHAR2(50),
    SOURCE VARCHAR2(50),
    DESTINATION VARCHAR2(50)
);

-- Insert values
INSERT INTO Railways VALUES (4101, 'Local', 'Dahanu Road', 'Churchgate');
INSERT INTO Railways VALUES (4102, 'AC', 'Virar', 'Borivali');

-- Update command
UPDATE Railways SET DESTINATION = 'Dadar' WHERE TRAIN_NO = 4102;

– Display table
SELECT * FROM Railways;
```