LAB REPORT

CSE – 4508

Name : Abdullah

ID : 200041126

Lab Group : 1B

Submitted on : 24th August, 2023

Submitted to : MD. SHIHAB SHAHRIAR SIR

Task 1

In step 1, we need to make 2 separate tablespaces and generate tables in them. Later, we'll shift a table from one tablespace to the other.

Initially, we must initiate a login using the SYS user credentials.

```
conn sys as sysdba;
```

Next, we'll generate 2 tablespaces using the command provided below.

```
CREATE TABLESPACE myspace DATAFILE 'myspace_data.dbf' SIZE 100M;
CREATE TABLESPACE myspace2 DATAFILE 'myspace2_data.dbf' SIZE 100M;
```

Now we will create user in the created tablespace.

```
-- Creating users
CREATE USER myuser IDENTIFIED BY mypass;
GRANT ALL PRIVILEGE TO myuser;


-- Assigning tablespace to user
ALTER USER myuser DEFAULT TABLESPACE myspace;
```

Now we will create 4 tables and give them some random attributes.

```
-- Creating tables
CREATE TABLE T1 (
        val1 INT PRIMARY KEY,
        val2 FLOAT,
        val3 VARCHAR2(15)
);

CREATE TABLE T2 (
        val1 INT PRIMARY KEY,
        val2 FLOAT,
        val3 VARCHAR2(15)
);

CREATE TABLE T3 (
        val1 INT PRIMARY KEY,
        val2 FLOAT,
        val3 VARCHAR2(15)
);

CREATE TABLE T4 (
        val1 INT PRIMARY KEY,
        val2 FLOAT,
        val3 VARCHAR2(15)
);
```

```
-- Moving table T4 to myspace2 tablespace
ALTER TABLE T4 MOVE TABLESPACE myspace2;
```

```
TABLE_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------
T4                             MYSPACE2
```

As we can see, we have successfully moved T4 to another created tablespace.

Task 2

For task 2, our objective is to establish a table using SQL. We'll then input data into this table. Following that, we'll execute SQL commands on the table to verify its functionality.

At first we'll create an Employee table

```
-- Creating related tables
CREATE TABLE Employee (
        ID INT,
        Name VARCHAR2(20),
        Phone VARCHAR2(20),

        CONSTRAINT emp_pk PRIMARY KEY (ID)
);
```

Now, we will insert 12 data in the table

```
-- Inserting data
INSERT INTO Employee VALUES (1, 'John', '01234567890');
INSERT INTO Employee VALUES (2, 'Jane', '02345678990');
INSERT INTO Employee VALUES (3, 'Marchal', '03456789090');
INSERT INTO Employee VALUES (4, 'Chris', '04567890190');
INSERT INTO Employee VALUES (5, 'Aaron', '05678901290');
INSERT INTO Employee VALUES (6, 'Michael', '06789012390');
INSERT INTO Employee VALUES (7, 'Jennifer', '07890123490');
INSERT INTO Employee VALUES (8, 'Sophia', '08901234590');
INSERT INTO Employee VALUES (9, 'Hilfiger', '09012345690');
INSERT INTO Employee VALUES (10, 'Tommy', '01023456790');
INSERT INTO Employee VALUES (11, 'Kenway', '01034567890');
INSERT INTO Employee VALUES (12, 'Darth', '02045678990');
```

This is our table –

```
        ID NAME                 PHONE
---------- -------------------- --------------------
         1 John                 01234567890
         2 Jane                 02345678990
         3 Marchal              03456789090
         4 Chris                04567890190
         5 Aaron                05678901290
         6 Michael              06789012390
         7 Jennifer             07890123490
         8 Sophia               08901234590
         9 Hilfiger             09012345690
        10 Tommy                10023456790
        11 Kenway               11034567890
        12 Darth                12045678990

12 rows selected.
```

Our next step involves crafting a query statement to retrieve employees whose phone numbers conclude with '990'. To accomplish this, we'll utilize the LIKE operator.

```
-- Using query
SELECT * FROM Employee WHERE Phone LIKE '%990';
```

Result of the query is given below-

```
        ID NAME                 PHONE
---------- -------------------- --------------------
         2 Jane                 02345678990
        12 Darth                12045678990
```

As you can see, we've successfully sorted out the numbers ending with '990'

Task 3

In task 3, we have to create 2 tables and test join operations among them.

At first we'll create 2 tables and insert data in the tables.

```
-- Creating tables with primary key and foreign key
CREATE TABLE Customers (
    CustomerID NUMBER,
    CustomerName VARCHAR2(50),
    CONSTRAINT cus_pk PRIMARY KEY (CustomerID)
);

CREATE TABLE Orders (
    OrderID NUMBER,
    CustomerID NUMBER,
    OrderAmount NUMBER,
    CONSTRAINT ord_pk PRIMARY KEY (OrderID),
    CONSTRAINT ord_cus_fk FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Inserting data
INSERT INTO Customers VALUES (1, 'John');
INSERT INTO Customers VALUES (2, 'Jane');
INSERT INTO Customers VALUES (3, 'Alice');
INSERT INTO Customers VALUES (4, 'Robert');

INSERT INTO Orders VALUES (101, 1, 100);
INSERT INTO Orders VALUES (102, 1, 150);
INSERT INTO Orders VALUES (103, 2, 200);
INSERT INTO Orders VALUES (104, 3, null);
INSERT INTO Orders VALUES (105, null, 300);
```

First table:

CUSTOMERID	CUSTOMERNAME
1	John
2	Jane
3	Alice
4	Robert

Second table:

ORDERID	CUSTOMERID	ORDERAMOUNT
101	1	100
102	1	150
103	2	200
104	3	
105		300

Left outer join:

```
-- Left Outer join
SELECT *
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

CUSTOMERID	CUSTOMERNAME	ORDERID	CUSTOMERID	ORDERAMOUNT
1	John	101	1	100
1	John	102	1	150
2	Jane	103	2	200
3	Alice	104	3	
4	Robert			

Right outer join:

```
-- Right Outer join
SELECT *
FROM Customers c
RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

CUSTOMERID	CUSTOMERNAME	ORDERID	CUSTOMERID	ORDERAMOUNT
1	John	102	1	150
1	John	101	1	100
2	Jane	103	2	200
3	Alice	104	3	
		105		300

Natural join:

```
-- Natural join
SELECT *
FROM Customers c
NATURAL JOIN Orders o;
```

CUSTOMERID	CUSTOMERNAME	ORDERID	ORDERAMOUNT
1	John	101	100
1	John	102	150
2	Jane	103	200
3	Alice	104	