Database Systems
Lab 2
Data Definition Language (DDL) and Data Modification Language (DML)
Version 2.1

To create a table, in SQL the command CREATE TABLE is used. In this part of lab, we will create a database that consists of 3 tables. The database is made for a car dealer. CARS table has the *model number, name, style and year* of manufacture. SPECS table will consist of *model number* (that is actually referring to a car in CARS table), *additional equipments and engine specification*. STOCK table has *the model number* (that is actually referring to a car in CARS table), *quantity and price*.

CARS	SPECS	STOCK
MD_NUM	MD_NUM	MD_NUM
MD_NAME	MPG	QTY
STYLE	RADIO	PRICE
YEAR	ENGINE	

Figure: Tables for Car dealer database

To create CARS table, write down and execute the query

```
CREATE TABLE cars(
              md_num integer,
              md_name      varchar(10),
              style  varchar(10),
              year   integer
       );
```

To create SPECS table, write down and execute the query

```
CREATE TABLE specs(
              md_num integer,
              mpg    integer,
              radio  varchar(10),
              engine varchar(10)
       );
```

To create STOCK table, write down and execute the query

```
CREATE TABLE stock(
              md_num integer,
              qty    integer,
              price  integer
       );
```

To drop a table completely from database, DROP TABLE command is used. It is very simple but be careful when you are dropping a table. Because once you drop a table, you will never be able to get that table.

The syntax is DROP TABLE TABLE_NAME;

So far, we have created three tables. Now, let us drop them. Before dropping them, let us see their existence in the database.

Type the following syntax on SQL plus.

```
SELECT * FROM tab;
```

This syntax is used to see the tables in the Oracle database. With many other tables you should find your tables- cars, specs and stock.

Now let us do the followings step by step.

```
DROP TABLE cars;
SELECT * FROM tab;
DROP TABLE specs;
SELECT * FROM tab;
DROP TABLE stock;
SELECT * FROM tab;
```

All of your tables now have been dropped from the database.

The ALTER TABLE command allows you to add, modify, or drop a column from an existing table.

Go to SQL Plus. Run 1create.sql by **Start drive:/1create.sql**. Then take a look at the table format by using DESCRIBE command.

DESCRIBE specs;
DESCRIBE stock;
DESCRIBE cars;

```
ALTER TABLE TABLE_NAME
       ADD COLUMN_NAME COLUMN_DEFINITION;
```

Now, add a column on specs table

```
ALTER TABLE specs
       ADD tyre      VARCHAR(10);
```

Use DESCRIBE command to see the effect.

```
ALTER TABLE TABLE_NAME
       ADD (  COLUMN_NAME1  COLUMN_DEFINITION1,
              COLUMN_NAME2  COLUMN_DEFINITION 2,
              ……..,
              COLUMN_NAMEn  COLUMN_DEFINITIONn);
```

Now add multiple columns on cars table

```
ALTER TABLE cars
       ADD (  company       VARCHAR(10),
              Supplier      VARCHAR(10));
```

Use DESCRIBE command to see the effect.

Various aspects of ALTER statement

```
ALTER TABLE TABLE_NAME
       MODIFY COLUMN_NAME COLUMN_DEFINITION;
```

Now modify the column in specs table you just entered

```
ALTER TABLE specs
       MODIFY tyre   integer;
```

Use DESCRIBE command to see the effect.

```
ALTER TABLE TABLE_NAME
       MODIFY (      COLUMN_NAME1  COLUMN_DEFINITION1,
                     COLUMN_NAME2  COLUMN_DEFINITION 2,
                     …….,
                     COLUMN_NAMEn  COLUMN_DEFINITIONn);
```

Now modify the columns you just entered on cars table

```
ALTER TABLE cars
       MODIFY (      company       VARCHAR(20),
                     Supplier      VARCHAR(20));
```

Use DESCRIBE command to see the effect.

To drop a column from a table

```
ALTER TABLE TABLE_NAME
       DROP COLUMN COLUMN_NAME;
```

Now drop the column you just created from specs table

```
ALTER TABLE specs
       DROP COLUMN tyre;
```

Use DESCRIBE command to see the effect.

To rename a column in a table

```
ALTER TABLE TABLE_NAME
       RENAME COLUMN old_name to new_name;
```

Now, rename the column company in cars to manufacturer

```
ALTER TABLE cars
       RENAME COLUMN company to manufacturer;
```

Use DESCRIBE command to see the effect.

In this part of the practical, we will insert data into the tables we have created so far. Our three tables will have the data as follows.

```
MD_NUM MD_NAME  STYLE   YEAR
------ -------  -----   ----
  1    HONDA    COUPE   1983
  2    TOYOTA   SALOON  1990
  3    BUICK    ESTATE  1991
  4    NISSAN   VAN     1992
  5    FORD     SALOON  1993

The Cars Table


MD_NUM  MPG  RADIO  ENGINE
------  ---  -----  ------
  1     43   YES    2L-4CYL
  2     25   NO     4L-V8
  3     18   YES    5L-V8
  4     50   NO     2L-4CYL
  5     45   YES    3L-V6

The Specs Table


MD_NUM  QTY  PRICE
------  ---  -----
  1     10    4980
  2      3   13865
  3      5   14900
  4      1   11000
  5      2   24600

The Stock Table
```

Figure: 3 Tables after Insertion of Data

In case of inserting data into tables, oracle uses the following INSERT syntax.

```
INSERT INTO table_name(column_1, column_2, ….., column_n)
       VALUES (value_for_column_1,value_for_column_2,……,
       value_for_column_n);
```

REMEMBER- when you are inserting values that are varchar, char or other string types (if any) then you will have to put ‘’ around that value (e.g. ‘value_for_column_X’).

Run the script named ‘1.sql’ by providing the command **START drive:/1.sql;** Before that, take a look at the script- how the insert statements are put into that script.

Use the following commands for viewing the tables with data. REMEMBER- we will take a deep look into the SELECT syntax in the later practical. It is the mostly used SQL syntax and you can do many things with this one syntax. At the moment just know that- SELECT is used to view tables. The ‘*’ sign means ‘ALL’. The basic structure of SELECT statement is as follows (for this lab purpose only).
SELECT * FROM table_name;

Now, execute the following commands one by one.

SELECT * FROM cars;
SELECT * FROM specs;
SELECT * FROM stock;

You will find the exact look of Figure 1 at this point.

UPDATE syntax is used in case of updating a value which is already in the table. Say, you have inserted a value 'Rudshi' in the 'Name' column of a table. But it should be 'Rushdi'. So, you need to use UPDATE statement to change such thing.
UPDATE syntax used by oracle is as follows.

```
UPDATE table_name
       SET column_name=corrected_value
       < WHERE criteria_to_find_desired_place_in_that_column >;
```

The WHERE clause in the above syntax is optional (means when you require it, you use it, when you don't you may not use it). It is used to find out the exact point in the table where you need to update that particular value. Now, consider the following examples, you will understand the principle.

Update the cars table so that the year for model name Ford will be set to 2007 instead of 1993 (See Figure 1).

```
UPDATE cars SET year=2007 WHERE md_name='Ford';
```

Now, take a look again by using SELECT statement to see- what effect the UPDATE command had on cars table.

```
SELECT * FROM cars;
```

The year is set to 2007 instead of 1993 for Ford. Isn't it?

In some cases, you will like to delete a whole row of a table. Say, accidentally you entered an entry which does not belong to that table. So at that stage, you will require DELETE command to delete that entry from table. The basic structure of DELETE command is as follows.

```
DELETE FROM table_name
       WHERE criteria_to_delete_item;
```

NOTE- WHERE clause is not optional this case. If you do not put WHERE clause when using with DELETE statement, then all the data in that table will be deleted. SO, BE CAREFUL!

Now, consider that, we will not have the car Ford in the showroom at all. What do we do? We simply delete that. So, it requires the following statement to be executed.

```
DELETE FROM cars WHERE md_name='Ford';
```

Executing the command, again, take a look at your table named cars if really it has been deleted by the command or not by executing the SELECT command again.

```
SELECT * FROM cars;
```

You will see that there are only 4 rows now in the cars table (previously it had 5 rows, see Figure).