**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.

**Renaming a column**

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.

**Database Systems**

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.

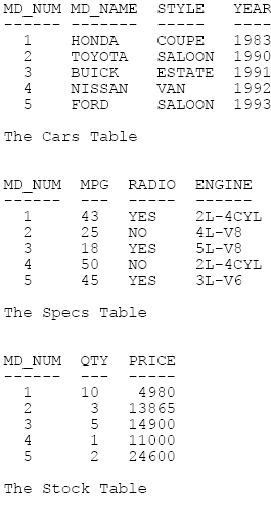


Figure: 3 Tables after Insertion of Data

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**

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).