Basic SQL

DDL

- Create a database

Create a schema → Create a new database

Example:

```
create database My_DB;

use My_DB;
```

DDL

- Create

- The SQL syntax for **CREATE TABLE** is

•CREATE TABLE "table_name"
("column 1" "data_type_for_column_1",
"column 2" "data_type_for_column_2",
...)

- So, if we are to create the customer table specified as above, we would type in

```
CREATE TABLE customer
(First_Name varchar(50),
Last_Name varchar(50),
Address varchar(50),
City varchar(50),
Country varchar(25),
Birth_Date date)
```

Constraint

NOT NULL

By default, a column can hold NULL. If you do not want to allow NULL value in a column, you will want to place a constraint on this column specifying that NULL is now not an allowable value.

For example, in the following statement,

```
CREATE TABLE Customer
(CID integer NOT NULL,
Last_Name varchar (30) NOT NULL,
First_Name varchar(30));
```

Columns “CID" and "Last_Name" cannot include NULL, while "First_Name" can include NULL.

UNIQUE

The UNIQUE constraint ensures that all values in a column are distinct.

For example, in the following statement,

```
CREATE TABLE Customer
(CID integer Unique,
Last_Name varchar (30),
First_Name varchar(30));
```

Column “CID" cannot include duplicate values, while such constraint does not hold for columns "Last_Name" and "First_Name".

Please note that a column that is specified as a primary key must also be unique. At the same time, a column that's unique may or may not be a primary key.

CHECK

The CHECK constraint ensures that all values in a column satisfy certain conditions.

For example, in the following statement,

```
CREATE TABLE Customer
(CID number  CHECK (CID > 0),
Last_Name varchar (30),
First_Name varchar(30));
```

Column “CID" must only include number greater than 0.

Primary Key

A primary key is used to uniquely identify each row in a table. It can either be part of the actual record itself , or it can be an artificial field (one that has nothing to do with the actual record). A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they are called a composite key.

```
CREATE TABLE Customer
(CID integer PRIMARY KEY,
Last_Name varchar(30),
First_Name varchar(30));
```

Foreign Key

A foreign key is a field (or fields) that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.

```
CREATE TABLE ORDERS
(Order_ID number primary key,
Order_Date date,
Customer_CID  number references CUSTOMER(CID),
Amount number(7,2));
```

•Drop Table Statement

•Sometimes we may decide that we need to get rid of a table in the database for some reason. In fact, it would be problematic if we cannot do so because this could create a maintenance nightmare for the DBA's. Fortunately, SQL allows us to do it, as we can use the **DROP TABLE** command. The syntax for **DROP TABLE** is

•DROP TABLE "table_name"

•So, if we wanted to drop the table called customer that we created in the CREATE TABLE section, we simply type

•DROP TABLE customer;

•Drop table tablename cascade constraint;

Alter Table Statement

Once a table is created in the database, there are many occasions where one may wish to change the structure of the table. Typical cases include the following:

1.Add, modify, or drop a column

2.Add or drop a constraint

3.Enable or disable a constraint

•Please note that the above is not an exhaustive list. There are other instances where ALTER TABLE is used to change the table structure, such as changing the primary key specification or adding a unique constraint to a column.

•The SQL syntax for **ALTER TABLE** is

•ALTER TABLE "table_name"
[alter specification]

[alter specification] is dependent on the type of alteration we wish to perform. For the uses cited above, the [alter specification] statements are:

- Add a column: ADD "column 1" "data type for column 1"
- Drop a column: DROP "column 1"
- Change the data type for a column: MODIFY "column 1" "new data type"

- **Adding column(s) to a table**
- **Syntax #1**
- To add a column to an existing table, the ALTER TABLE syntax is:
- ALTER TABLE table_name
  ADD column_name column-definition;
- **For example:**
- ALTER TABLE supplier
  ADD (supplier_name varchar(50));
- This will add a column called *supplier_name* to the *supplier* table.

•**Syntax #2**

•To add multiple columns to an existing table, the ALTER TABLE syntax is:

•**ALTER TABLE table_name**

ADD (column_1 column-definition, column_2 column-definition,

column_n column_definition);

•**For example:**

•**ALTER TABLE supplier**

ADD (supplier_name varchar(50), city varchar(45));

•This will add two columns (*supplier_name* and *city*) to the *supplier* table.

- **Modifying column(s) in a table**

- **Syntax #1**

- To modify a column in an existing table, the ALTER TABLE syntax is:

- ALTER TABLE table_name
  MODIFY column_name column_type;

- **For example:**

- ALTER TABLE supplier
  MODIFY (supplier_name varchar(100) not null);

- This will modify the column called *supplier_name* to be a data type of varchar(100) and force the column to not allow null values

•**Syntax #2**

•To modify multiple columns in an existing table, the ALTER TABLE syntax is:

•**ALTER TABLE table_name**

MODIFY (column_1 column_type, column_2 column_type, column_n column_type);

•**For example:**

•**ALTER TABLE supplier**

MODIFY (supplier_name varchar(100) not null,

City varchar(75));

•This will modify both the *supplier_name* and *city* columns

- **Drop column(s) in a table**

- **Syntax #1**

- To drop a column in an existing table, the ALTER TABLE syntax is:

- ALTER TABLE table_name
  DROP COLUMN column_name;

- **For example:**

- ALTER TABLE supplier
  DROP COLUMN supplier_name;

- This will drop the column called *supplier_name* from the table called *supplier*.

- **Rename column(s) in a table**
- **Syntax #1**
- To rename a column in an existing table, the ALTER TABLE syntax is:
- ALTER TABLE table_name
  RENAME COLUMN old_name to new_name;
- **For example:**
- ALTER TABLE supplier
  RENAME COLUMN supplier_name to sname;
- This will rename the column called *supplier_name* to *sname*.

•Alter table to add new constraint

- **alter table customer**

  add constraint ckgender check (gender in ('M', 'F'));

•ALTER TABLE to Enable a CONSTRAINT

set ckgender = 1;

•ALTER TABLE to DISABLE a CONSTRAINT

set ckgender = 0;

•Alter table to add primary key

- create table student (SID integer (30), Sname varchar(50), Sage varchar (40));
- Alter table student add Primary key (SID);

- **Insert values in a table**

```
insert into customer (CID, Cname, C_adress, gender)
values('9','dd','xxx','h');
```

•Truncate Table Statement

•Sometimes we wish to get rid of all the data in a table. One way of doing this is with **DROP TABLE**, which we saw in the last section. But what if we wish to simply get rid of the data but not the table itself? For this, we can use the **TRUNCATE TABLE** command. The syntax for **TRUNCATE TABLE** is

•**TRUNCATE TABLE "table_name"**

•So, if we wanted to truncate the table called customer that we created in SQL CREATE TABLE, we simply type,

• **truncate table customer;**