# ST.XAVIER’S COLLEGE

# MAITIGHAR, KATHMANDU

****

**ASSIGNMENT #8**

**Database Management System**

**Submitted By:**

Yub Raj Basnet

013BSCCSIT048

**Submitted To:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Er. Sanjay Kumar Yadav

Department of Computer Science

Lecturer

**4.2 Data Definition Language**

A DDL is a language used to define data structures and modify data. For example, DDL commands can be used to add, remove, or modify tables within in a database. DDLs used in database applications are considered a subset of SQL, the Structured Query Language. A Data Definition Language has a pre-defined syntax for describing data.[1]

For example, to build a new table using SQL syntax, the CREATE command is used, followed by parameters for the table name and column definitions. The DDL can also define the name of each column and the associated data type. Once a table is created, it can be modified using the ALTER command. If the table is no longer needed, the DROP command can be used to delete the table.[1]

Since DDL is a subset of SQL, it does not include all the possible SQL commands. For example, commands such as SELECT and INSERT are considered part of the Data Manipulation Language (DML), while access commands such as CONNECT and EXECUTE are part of the Data Control Language (DCL). The DDL, DML, and DCL languages include most of the commands supported by SQL.[1]

**4.2.1 Domain Type in SQl**

1. The SQL standard supports a variety of built-in domain types[3]:
   * **char**(n) (or **character**(n)): fixed-length character string, with user-specified length.
   * **varchar**(n) (or **character varying**): variable-length character string, with user-specified maximum length.
   * **int** or **integer**: an integer (length is machine-dependent).
   * **smallint**: a small integer (length is machine-dependent).
   * **numeric**(*p, d*): a fixed-point number with user-specified precision, consists of *p* digits (plus a sign) and *d* of *p* digits are to the right of the decimal point. E.g., **numeric**(*3, 1*) allows 44.5 to be stored exactly but not 444.5.
   * **real** or **double precision**: floating-point or double-precision floating-point numbers, with machine-dependent precision.
   * **float**(n): floating-point, with user-specified precision of at least *n* digits.
   * **date**: a calendar date, containing four digit year, month, and day of the month.
   * **time**: the time of the day in hours, minutes, and seconds.
2. allows arithmetic and comparison operations on various numeric domains, including, **interval** and *cast* (*type coercion*) such as transforming between *smallint* and *int*. It considers strings with different length are compatible types as well.
3. allows **create domain** statement, e.g.,

**create domain** *person-name* **char**(20)

4.2.2 Schema Definition in SQL

In computer programming, a schema (pronounced SKEE-mah) is the organization or structure for a database. The activity of data modeling leads to a schema. (The plural form is schemata. The term is from a Greek word for "form" or "figure." Another word from the same source is "schematic.") The term is used in discussing both relational databases and object-oriented databases. The term sometimes seems to refer to a visualization of a structure and sometimes to a formal text-oriented description.[3]

Duplicate Tuples

Microsoft SQL Server tables should never contain duplicate rows, nor non-unique primary keys. For brevity, we will sometimes refer to primary keys as "key" or "PK" in this article, but this will always denote "primary key." Duplicate PKs are a violation of entity integrity, and should be disallowed in a relational system. SQL Server has various mechanisms for enforcing entity integrity, including indexes, UNIQUE constraints, PRIMARY KEY constraints, and triggers.  
  
Despite this, under unusual circumstances duplicate primary keys may occur, and if so they must be eliminated. One way they can occur is if duplicate PKs exist in non-relational data outside SQL Server, and the data is imported while PK uniqueness is not being enforced. Another way they can occur is through a database design error, such as not enforcing entity integrity on each table.

There are various times when we need to find duplicate records in SQL Server. It is possible to find duplicates using **DISTINCT, ROW NUMBER as well as the GROUP BY** approach.

Duplicate records can create problems sometimes when displaying reports or performing a Multiple Insert update. Finding duplicate records in a database needs further investigation. In some cases, duplicate records are positive, but it all depends on the data and the database design as well.

For example, if a customer has ordered the same product twice on the same date with the the same shipping and billing address, then this may result in a duplicate record.

Let us create a table **Customer** with First Name, Last Name, and Mobile Number fields.

CREATE TABLE CUSTOMER

(

FirstName VARCHAR(50),

LastName VARCHAR(50),

MobileNo VARCHAR(15)

);

INSERT INTO CUSTOMER VALUES ('Niraj','Yadav',989898);

INSERT INTO CUSTOMER VALUES ('Chetan','Gadodia',959595);

INSERT INTO CUSTOMER VALUES ('Chetan','Gadodia',959595);

INSERT INTO CUSTOMER VALUES ('Atul','Kokam',42424242);

INSERT INTO CUSTOMER VALUES ('Atul','Kokam',42424242);

INSERT INTO CUSTOMER VALUES ('Vishal','Parte',9394453);

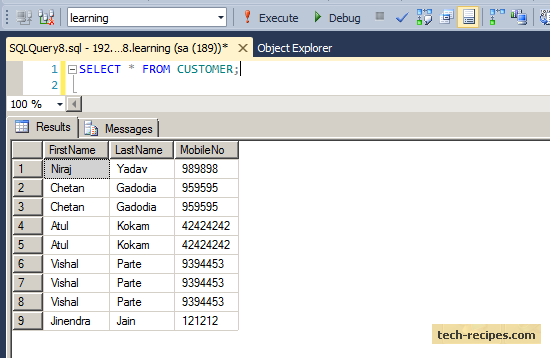
INSERT INTO CUSTOMER VALUES ('Vishal','Parte',9394453);

INSERT INTO CUSTOMER VALUES ('Vishal','Parte',9394453);

INSERT INTO CUSTOMER VALUES ('Jinendra','Jain',121212);

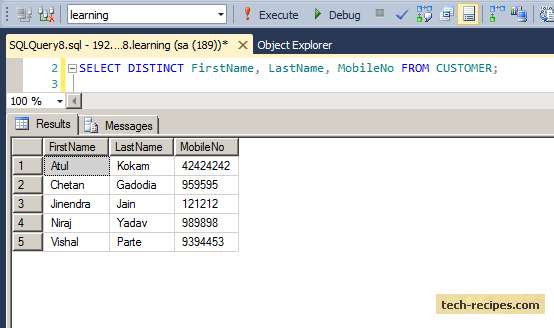
INSERT INTO CUSTOMER VALUES ('Jinendra','Jain',121212);

SELECT \* FROM CUSTOMER;



Using the **DISTINCT**approach, we can quickly get unique rows in a table.

SELECT DISTINCT FirstName, LastName, MobileNo FROM CUSTOMER;



However, this does not show how many times a row has been duplicated. Using the GROUP BY approach, we can find this.

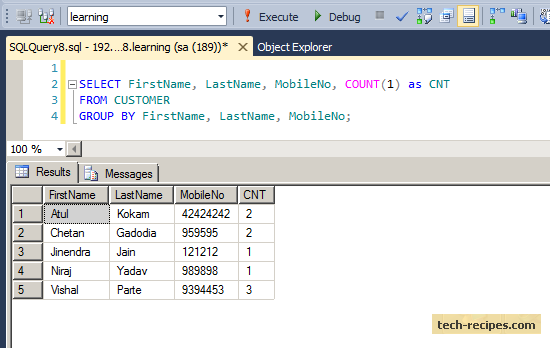
**Finding Duplicates Using GROUP BY**

Adding grouping and a counting field to our display of FirstName, LastName and MobileNo combination shows how many times each customer’s name appears.

SELECT FirstName, LastName, MobileNo, COUNT(1) as CNT

FROM CUSTOMER

GROUP BY FirstName, LastName, MobileNo;



**GROUP BY will show just one record for each combination of FirstName, LastName and MobileNo.**

The count CNT shows how many times the row has been duplicated.  
CNT = 1 indicates that row appears only once.

Let us filter out using the **Having clause** to exclude rows that appear only once.

SELECT FirstName, LastName, MobileNo, COUNT(1) as CNT

FROM CUSTOMER

GROUP BY FirstName, LastName, MobileNo

HAVING COUNT(1) > 1;

