Creating and Altering tables in SQL

Dr. Seema Gupta Bhol

Creating database

CREATE DATABASE *databasename*;

```
CREATE DATABASE  testDB;
```

- Make sure you have admin privilege before creating any database.
- Once a database is created, one can check it in the list of databases with the following SQL command: SHOW DATABASES;

DROP DATABASE

- The DROP DATABASE statement is used to drop an existing SQL database.
- Deleting a database will result in loss of complete information stored in the database

```
DROP DATABASE testDB;
```

CREATE TABLE

- The CREATE TABLE statement is used to create a new table in a database.

-
  ```
  CREATE TABLE table_name (
      column1 datatype,
      column2 datatype,
      column3 datatype,
     ....
  );
  ```

```
CREATE TABLE Persons (
        ID int,
        LastName varchar(255),
        FirstName varchar(255),
        Address varchar(255),
        City varchar(255)
    );
```

Table creation

- The empty “Persons" table will now look like this:

ID	LastName	FirstName	Address	City

- The empty “Persons" table can now be filled with data with the SQL INSERT INTO statement.

Create Table Using Another Table

- A copy of an existing table can also be created using CREATE TABLE.
- The new table gets the same column definitions. All columns or specific columns can be selected.
- If a new table is created using an existing table, the new table will be filled with the existing values from the old table.

Create Table Using Another Table

-
  ```
  CREATE TABLE new_table_name AS
      SELECT column1, column2,...
      FROM existing_table_name
      WHERE ....;
  ```

-
  ```
  CREATE TABLE TestTable AS
  SELECT id, firstname
  FROM Persons;
  ```

DROP TABLE

- The DROP TABLE statement is used to drop an existing table in a database.
- DROP TABLE *table_name*;

```
DROP TABLE  TestTable;
```

- Deleting a table will result in loss of complete information stored in the table!

ALTER TABLE

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.
- To add a column in a table
- ALTER TABLE *table_name*
  ADD *column_name datatype*;

```
ALTER TABLE Persons
 ADD Email varchar(255);
```

ALTER TABLE - DROP COLUMN

- To delete a column in a table:
- ALTER TABLE *table_name*
  DROP COLUMN *column_name*;

```
ALTER TABLE Persons
 DROP COLUMN Email;
```

ALTER TABLE - RENAME COLUMN

- To rename a column in a table
- ALTER TABLE *table_name*
  RENAME COLUMN *old_name* to *new_name*;

ALTER TABLE *Persons*
RENAME COLUMN *address* to *location*

Create Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Create Constraints

- The following constraints are commonly used in SQL:
- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

NOT NULL Constraint

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
  );
```

NOT NULL on ALTER TABLE

- To create a NOT NULL constraint on the "Age" column when the “Persons" table is already created

```
ALTER TABLE  Persons
MODIFY id NOT NULL;
```

UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, one can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

UNIQUE Constraint on CREATE TABLE

- The following SQL creates a UNIQUE constraint on the "ID" column when the “Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
  );
```

UNIQUE constraint

- To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns:

-
  ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      CONSTRAINT UC_Persons UNIQUE (ID,LastNam
  e)
  );
  ```

UNIQUE Constraint on ALTER TABLE

- To create a UNIQUE constraint on the "ID" column when the table is already created

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

- To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Persons UNIQUE (ID,LastName);
```

DROP a UNIQUE Constraint

- DROP a UNIQUE Constraint

  ```
  ALTER TABLE Persons
  DROP CONSTRAINT UC_Persons;
  ```

PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

PRIMARY KEY on CREATE TABLE

- Create a PRIMARY KEY on the "ID" column when the “Persons" table is created:

```
CREATE TABLE  Persons(
      ID int NOT NULL PRIMARY KEY,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int
   );
```

PRIMARY KEY on CREATE TABLE

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Persons PRIMARY KEY (ID,LastName)
);
```

There is only ONE PRIMARY KEY (PK_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

PRIMARY KEY on ALTER TABLE

- To create a PRIMARY KEY constraint on the "ID" column when the table is already created:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns:

```
ALTER TABLE  Persons
ADD CONSTRAINT PK_ Persons
PRIMARY KEY (ID,LastName);
```

- If ALTER TABLE is used to add a primary key, the primary key column(s) must have been declared to not contain NULL values (when the table was first created).

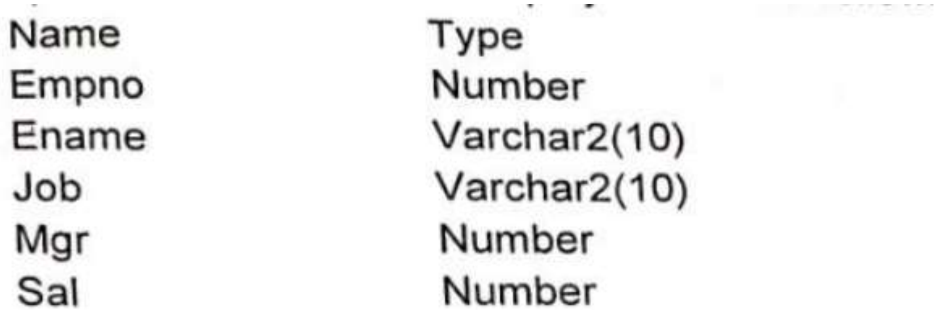
DROP a PRIMARY KEY Constraint

- To drop a PRIMARY KEY constraint:

```
ALTER TABLE Persons
 DROP CONSTRAINT PK_Persons;
```

Assignment

1. Create table worker, with following columns.



Name	Type
Empno	Number
Ename	Varchar2(10)
Job	Varchar2(10)
Mgr	Number
Sal	Number

2. Add a column commission .
3. Rename column job to designation
4. Remove column commission.
5. Create a constraint to make Ename, Empno, salary not null.
6. Create a constraint to make Empno unique.
7. Make Empno primary key
8. Insert five records into table worker.
9. Update salary of all the workers by diving them raise of 10%.
10. Drop not null constraint on salary

FOREIGN KEY Constraint

- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

- "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.
- The "PersonID" column in the "Persons" table is the `PRIMARY KEY` in the "Persons" table.
- The "PersonID" column in the "Orders" table is a `FOREIGN KEY` in the "Orders" table.
- The `FOREIGN KEY` constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

FOREIGN KEY on CREATE TABLE

- The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID
  int FOREIGN KEY REFERENCES Persons(PersonID)
  );
```

FOREIGN KEY on CREATE TABLE

- To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns:

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

FOREIGN KEY on ALTER TABLE

- To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

- To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

DROP a FOREIGN KEY Constraint

- To drop a FOREIGN KEY constraint:

```
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- CHECK constraint defined on a column will allow only certain values for this column.
- CHECK constraint defined a on a table can limit the values in certain columns based on values in other columns in the row.

CHECK on CREATE TABLE

- The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

CHECK on CREATE TABLE

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City=‘Bhubaneshwar')
);
```

CHECK on ALTER TABLE

- To create a CHECK constraint on the "Age" column when the table is already created:

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

- To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns:

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=
18 AND City=‘Bhubaneshwar');
```

DROP a CHECK Constraint

- To drop a CHECK constraint:

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.
- The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:
-
  ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      City varchar(255) DEFAULT ‘Bhubaneswar’)
  ```

DEFAULT on ALTER TABLE

To create a DEFAULT constraint on the "City" column when the table is already created

```
ALTER TABLE Persons
MODIFY City DEFAULT‘Bhubaneswar';
```

DROP a DEFAULT Constraint

To drop a `DEFAULT` constraint, use the following SQL:

```
ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT;
```

CREATE INDEX

- The `CREATE INDEX` statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly than otherwise.
- The users cannot see the indexes, they are just used to speed up searches/queries.
- Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
     ON table_name (column1, column2, ...);
```

```
e.g. CREATE INDEX idx_lastname
     ON Persons (LastName);
```

- To create an index on a combination of columns, list the column names within the parentheses, separated by commas:

```
     CREATE INDEX idx_pname
     ON Persons (LastName, FirstName);
```

DROP INDEX Statement

The `DROP INDEX` statement is used to delete an index in a table.

```
DROP INDEX index_name;
```