

UNIT-4: Concepts of SQL and Queries (Single Table only)

4.1. Concepts of Structure Query Language (SQL)

4.1.1. SQL datatypes: int, float, double, char, varchar, number, varchar2, Text, date

4.2. DDL , DML and DQL(Data Query Language) Statements

4.2.1. Create, Drop, Truncate, Rename, Alter, insert, Update, Delete, select

4.3. Using where clause and operators with where clause

4.3.1. In, between, like, not in,=,!=,>,<=,wildcard operators, Order by, Group by, Distinct

4.3.2. AND, OR operators, Exists and not Exists, Use of Alias

4.4. Constraints(Table level and Attribute Level)

4.4.1. NOTNULL, CHECK, DEFAULT, UNIQUE, Primary Key, Foreign Key, On Delete Cascade

4.5. SQL Functions

4.5.1. Aggregate Functions: avg(), max(), min(), sum(), count(), first(), last().

4.5.2. Scalar Functions: ucase(), lcase(), round(), mid().

4.6. Creating sequence

4.7. Views

4.7.1. Creating simple view, updating view, dropping view

4.7.2. Difference between View and Table

4.3.1 What is SQL:-

- SQL is a standard language for accessing and manipulating databases.
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

4.3.2 SQL Data Types

- Each column in a database table is required to have a name and a data type.
- An SQL developer must decide what type of data that will be stored inside each column when creating a table.
- The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

SQL data types for OpenOffice Base:-

Numeric Types:-

Name	Data type	No. of Bytes
BOOLEAN	boolean	1
TINYINT	integer	1
SMALLINT	integer	2
INTEGER	integer	4
BIGINT	integer	8
NUMERIC	number	no limit
DECIMAL	decimal	no limit
REAL	real	4
FLOAT	float	4
DOUBLE	double	4

Alphanumeric Types:-

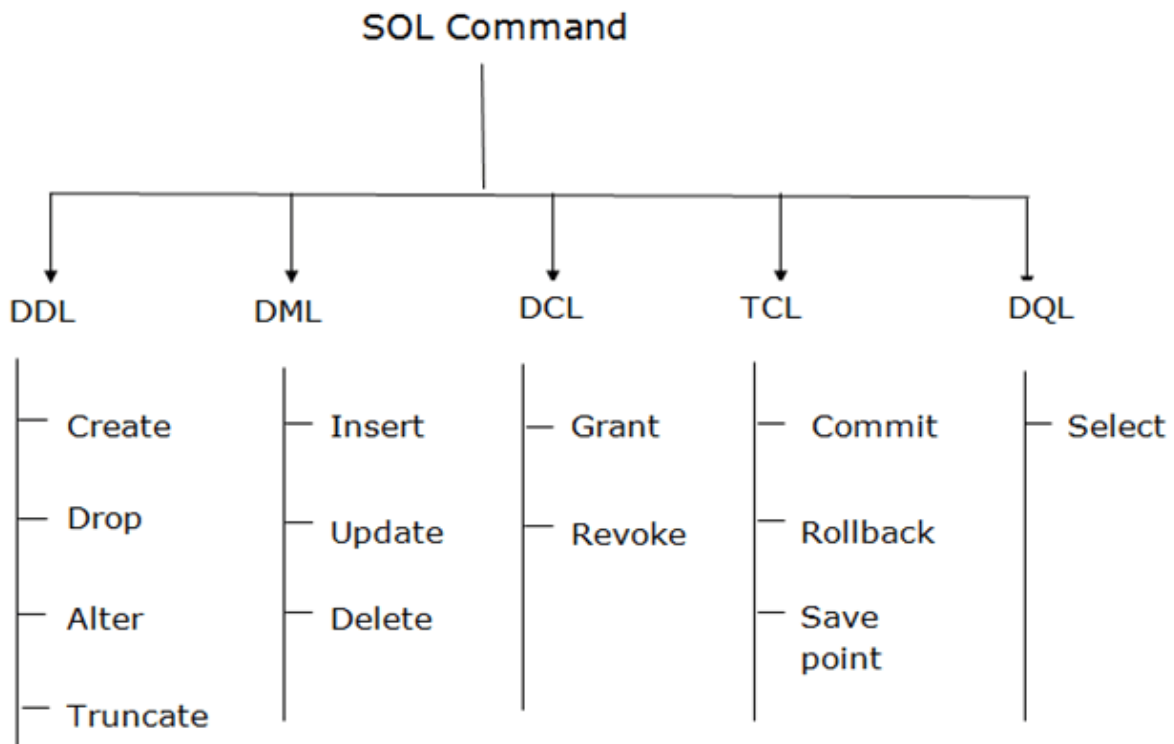
Name	Data type	Max length	Description
Char	text(fix)	2GB for 32 bit OS	Stores exactly the number of characters specified by user.
Varchar	text	2GB for 32 bit OS	Stores up to the specified number of characters.
LONGVARCHA R	text	2GB for 32 bit OS	Stores up to the max number of characters indicated by user.

Date time:-

Name	Description	Format
Date	Stores month, day and year information	1/1/99 to 1/1/9999
Time	Stores hour, minute and second info	Seconds since 1/1/1970
Timestamp	Stores date and time information	Timestamp

4.3.3SQL Statements:-

- There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



4.3.3.1 Data Definition Language (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME

CREATE

- It is used to create a new table in the database.

Syntax:

```
CREATE TABLE TABLE_NAME
(
    COLUMN_NAME1 DATATYPES[SIZE],
    COLUMN_NAME2 DATATYPES[SIZE],
    COLUMN_NAME3 DATATYPES[SIZE],
    .....
);
```

Example:

```
CREATE TABLE STUDENT
(
    ROLLNO NUMBER,
    NAME VARCHAR2 (100),
    EMAIL VARCHAR2 (100),
    DOB DATE
);
```

DROP command

- It is used to delete both the structure and record stored in the table.

Syntax:

```
DROP TABLE TABLE_NAME;
```

Example

```
DROP TABLE STUDENT;
```

ALTER

- 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.

Syntax:

ALTER TABLE - ADD Column

- To add a column in a table, use the following syntax:

```
ALTER TABLE table_name  
ADD column_name datatype;
```

Example

ALTER TABLE STUDENT ADD CITY VARCHAR(20);

ALTER TABLE - DROP COLUMN

- To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

**ALTER TABLE table_name
DROP COLUMN column_name;**

- The following SQL deletes the "CITY" column from the "STUDENT" table:

Example

ALTER TABLE STUDENT

DROP COLUMN CITY;

ALTER TABLE - ALTER COLUMN

- To change the data type of a column in a table, use the following syntax:

**ALTER TABLE TABLE_NAME
MODIFY COLUMN NAME
(COLUMN DEFINITION....);**

Example

ALTER TABLE STUDENT

MODIFY NAME VARCHAR(20);

RENAME TABLE

- RENAME TABLE is used to change the name of a table. Sometimes, we choose non-meaningful name for the table. So it is required to be changed.
- To rename the table, use the following syntax

**ALTER TABLE old_table_name
RENAME To new_table_name;**

Example

**ALTER TABLE STUDENT
RENAME TO STUDENT_MASTER;**

TRUNCATE

- It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE STUDENT;

Note:-TRUNCATE Command is not working in OpenOffice BASE.

4.3.3.2 Data Manipulation Language

DML commands are used to modify the database. It is responsible for all form of changes in the database.

The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

The SQL INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- It is possible to write the INSERT INTO statement in two ways.

1) The first way specifies both the column names and the values to be inserted.

2) If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

Syntax

- 1) INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
- 2) INSERT INTO table_name
VALUES (value1, value2, value3, ...);

Example

```
INSERT INTO STUDENT_MASTER (ROLLNO, NAME,
DOB)
VALUES (1,'Rahul','8-DEC-1991');
```

OR

```
INSERT INTO STUDENT_MASTER
VALUES (1,'Rahul','8-DEC-1991');
```

Insert Data Only in Specified Columns

- It is also possible to only insert data in specific columns.
- The following SQL statement will insert a new record, but only insert data in the ROLLNO, NAME and CITY columns.

```
INSERT INTO STUDENT_MASTER(ROLLNO,NAME,CITY)
VALUES (2,'MEGHA','SURAT');
```

The SQL UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table.

Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
[WHERE condition];
```

Note: The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated.

Example:

```
1)UPDATE STUDENT_MASTER
SET NAME = 'MEGHA';
```

It will update all student detail with name MEGHA.

2) If you want to update only specific student's name for example let's update the name of student to megha whose rollno is 3.

```
UPDATE STUDENT_MASTER  
SET NAME = 'MEGHA'  
WHERE ROLLNO = 3;
```

3) You can update multiple column at the same time. Lets set DOB='2000-10-11' and email='megha123@gmail.com' of student whose roll no 2.

```
UPDATE STUDENT_MASTER  
SET DOB = '13-AUG-2001',  
EMAIL='megha123@gmail.com'  
WHERE ROLLNO = 2;
```

The SQL DELETE Statement

- The DELETE statement is used to delete existing records in a table.

Syntax

```
DELETE FROM table_name  
[WHERE condition];
```

Note: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted.

Example:

1)DELETE FROM STUDENT_MASTER;

It will delete all the records from STUDENT_MASTER table.

2)DELETE FROM STUDENT_MASTER

WHERE NAME='MEGHA';

It will delete the record from STUDENT_MASTER table whose name is 'MEGHA'.

4.3.3.3 Data Query Language

- DQL is used to fetch the data from the database.
- It uses only one command:**SELECT**

The SQL SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.

Syntax

```
SELECT column1, column2, ...  
FROM table_name;
```

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT * FROM table_name;

Example

1) SELECT *

FROM STUDENT_MASTER;

It will retrieve all the rows from STUDENT_MASTER table.

2) SELECT Column

The following SQL statement selects the "NAME" and "CITY" columns from the "STUDENT" table:

Example

SELECT NAME,CITY FROM STUDENT_MASTER;

The SQL SELECT DISTINCT Statement

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

Syntax

SELECT DISTINCT column1, column2, ...
FROM table_name;

Example

SELECT Without DISTINCT

The following SQL statement selects ALL (including the duplicates) values from the "City" column in the "STUDENT_MASTER" table:

```
SELECT CITY FROM STUDENT_MASTER;
```

SELECT With DISTINCT

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

```
SELECT DISTINCT CITY FROM STUDENT_MASTER;
```

Queries

5.1 The SQL WHERE Clause

- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

Demo Database

ROLL NO	NAME	EMAIL	DOB	CITY
1	MIRA	mira123@gmail.com	10/11/00	SURAT
2	RAJ	raj123@gmail.com	06/02/99	NAVSARI
3	MEGHA	megha123@gmail.com	02/04/00	SURAT
4	TEJAS	tejas123@gmail.com	11/08/99	VAPI
5	RATA	ratan123@gmail.com	12/11/00	NAVSARI

	N	com	99	ARI
--	---	-----	----	-----

Example

1)The following SQL statement selects all the student from the city "NAVASARI", in the "STUDENT_MASTER" table:

```
SELECT * FROM STUDENT_MASTER  
WHERE CITY='NAVASARI';
```

Text Fields vs. Numeric Fields

- SQL requires single quotes around text values (most database systems will also allow double quotes).
- However, numeric fields should not be enclosed in quotes:

```
SELECT * FROM STUDENT_MASTER  
WHERE ROLLNO=1;
```

5.1.1 Operators in The WHERE Clause

- The following operators can be used in the WHERE clause:
- Lets two table as a example.

1.STUDENT_MASTER(ROLLNO,NAME,EMAIL,DOB,CITY);

2.STUDENT_MARKS_DETAIL(ROLLNO,101_MARKS,102_MARKS,103_MARKS,104_MARKS,105_MARKS,106_MARKS);

1)=:-It is used as equal operator.

```
SELECT * FROM STUDENT_MASTER
WHERE EMAIL = 'megha123@gmail.com';
```

It displays student detail whose email is
'megha123@gmail.com'

2)> :-It is used as greater than operator.

```
SELECT * FROM
STUDENT_MARKS_DETAIL
WHERE 105_MARKS >19;
```

It displays details of all the student who were passed in 105 subject(ie Those student who got greater than 19 marks in subject 105)

3) < :-It is used as less than operator.

```
SELECT * FROM  
STUDENT_MARKS_DETAIL  
WHERE 105_MARKS < 20;
```

It displays details of all the student who were failed in 105 subject(ie Those student who got less than 20 marks in subject 105)

4) >= :-It is used as greater than or equal to operator.

```
SELECT * FROM  
STUDENT_MARKS_DETAIL  
WHERE 105_MARKS >=45;
```

It displays details of all the student who got greater than or equal to 45 marks in 105 subject

5) <= :-It is used as less than or equal to operator.

```
SELECT * FROM  
STUDENT_MARKS_DETAIL  
WHERE 105_MARKS <= 25;
```

It displays details of all the student who got less than or equal to 25 marks in 105 subject

6) <> (or !=) :-It is used not equal to operator.

```
SELECT * FROM  
STUDENT_MARKS_DETAIL  
WHERE 105_MARKS <>30;
```

It displays details of all the student who didn't 30 marks in 105 subject

7) BETWEEN

- The SQL BETWEEN condition allows you to easily test if an expression is within a range of values (inclusive).
- The values can be text, date, or numbers. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

- The SQL BETWEEN Condition will return the records where expression is within the range of value1 and value2.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1
AND value2;
```

Example**1)Using BETWEEN with Numeric Values:**

List marks details of students whose rollno between 1 and 30.

```
SELECT *
FROM STUDENT_MARKS_DETAIL
WHERE ROLLNO
BETWEEN 1 AND 30;
```

2)Using BETWEEN with Date Values:

Find all the Student having Date of Birth
Between 01-01-2000 and 31-12-2001.

```
SELECT *  
  
FROM STUDENT_MASTER  
  
WHERE DOB  
  
BETWEEN '2000-jan-01' AND '2001-dec-31';
```

3)Using NOT operator with BETWEEN

Find all the Students whose Roll No is not in the
range of 20 to 30.

```
SELECT *  
  
FROM STUDENT_MASTER  
  
WHERE ROLLNO  
  
NOT BETWEEN 20 AND 30;
```

8) IN

- IN operator allows you to easily test if the expression matches any value in the list of values.
- It is used to remove the need of multiple OR condition in SELECT, INSERT, UPDATE or DELETE.
- You can also use NOT IN to exclude the rows in your list.

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (list_of_values);
```

Examples

1) Find the Students details who live in surat or navsari.

```
SELECT *
FROM STUDENT_MASTER
```

WHERE CITY IN ('SURAT','NAVSARI');

2) Find the Students details who does not live in surat or navsari.

SELECT *

FROM STUDENT_MASTER

WHERE CITY NOT IN
('SURAT','NAVSARI');

9) LIKE :-

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards often used in conjunction with the LIKE operator:
- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character
- The percent sign and the underscore can also be used in combinations!

Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE columnName LIKE pattern;
```

Here are some examples showing different LIKE operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE Student NAME LIKE 'a%'	Finds any values that start with "a"
WHERE Student NAME LIKE '%a'	Finds any values that end with "a"
WHERE Student NAME LIKE '%or%'	Finds any values that have "or" in any position
WHERE Student NAME LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Student NAME LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE Student NAME LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in

	length
WHERE Student NAME LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

Examples

1)The following SQL statement selects all Student with a NAME starting with "a":

```
SELECT * FROM STUDENT_MASTER  
WHERE NAME LIKE 'a%';
```

2)The following SQL statement selects all Student with a NAME ending with "a":

```
SELECT * FROM STUDENT_MASTER  
WHERE NAME LIKE '%a';
```

3)Select all records where the value of the City column starts with the letter "s".

```
SELECT * FROM STUDENT_MASTER  
WHERE CITY LIKE 's%';
```

5.1.2ORDER BY:-

- The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

Syntax

SELECT column-list

FROM table_name

[WHERE condition]

[ORDER BY column1, column2, ..columnN]

[ASC | DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort that column should be in the column-list.

Example

Consider the STUDENT_MASTER table having the following records –

ROLL NO	NAM E	EMAIL	DOB	CITY
1	MIRA	mira123@gmail.com	10/11/00	SURAT
2	RAJ	raj123@gmail.com	06/02/99	NAVSARI
3	MEGHA	megha123@gmail.com	02/04/00	SURAT
4	TEJAS	tejas123@gmail.com	11/08/99	VAPI
5	RATAN	ratan123@gmail.com	12/11/99	NAVSARI

1)The following code block has an example, which would sort the result in an ascending order by the NAME and the –

```
SELECT * FROM STUDENT_MASTER
ORDER BY NAME;
```

This would produce the following result –

ROLL NO	NAM E	EMAIL	DOB	CITY
3	MEG	megha123@gmail.com	02/04/00	SURAT

	HA	l.com	00	
1	MIRA	mira123@gmail.com	10/11/00	SURAT
2	RAJ	raj123@gmail.com	06/02/99	NAVSARI
5	RATAN	ratan123@gmail.com	12/11/99	NAVSARI
4	TEJAS	tejas123@gmail.com	11/08/99	VAPI

2)The following code block has an example, which would sort the result in the descending order by NAME.

```
SELECT * FROM STUDENT_MASTER
ORDER BY NAME DESC;
```

This would produce the following result –

ROLL NO	NAME	EMAIL	DOB	CITY
4	TEJAS	tejas123@gmail.com	11/08/99	VAPI
5	RATAN	ratan123@gmail.com	12/11/99	NAVSARI

2	RAJ	raj123@gmail.com	06/02/99	NAVSARI
1	MIRA	mira123@gmail.com	10/11/00	SURAT
3	MEGHA	megha123@gmail.com	02/04/00	SURAT

5.1.3 GROUP BY :-

- The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.
- This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax

- The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

SELECT column1, column2

FROM table_name

WHERE [conditions]

GROUP BY column1, column2

ORDER BY column1, column2;

Example

Consider the CUSTOMERS table is having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	RAMESH	32	AHMEDABAD	2000.00
2	KHILAN	25	DELHI	1500.00
3	KAUSHIK	23	KOTA	2000.00
4	CHAITALI	25	MUMBAI	6500.00
5	HARDIK	27	BHOPAL	8500.00
6	KOMAL	22	MP	4500.00
7	MUFFY	24	INDORE	10000.00

1)If you want to know the total amount of the salary on each customer, then the GROUP BY query would be as follows.

SELECT NAME, SUM(SALARY) FROM
CUSTOMERS GROUP BY NAME;

This would produce the following result –

NAME	SUM(SALARY)
CHAITALI	6500.00
HARDIK	8500.00
KAUSHIK	2000.00
KHILAN	1500.00
KOMAL	4500.00
MUFFY	10000.00
RAMESH	2000.00

Now, let us look at a table where the CUSTOMERS table has the following records with duplicate names

ID	NAME	AGE	ADDRESS	SALARY
1	RAMESH	32	AHMEDABAD	2000.00
2	RAMESH	25	DELHI	1500.00
3	KAUSHIK	23	KOTA	2000.00
4	KAUSHIK	25	MUMBAI	6500.00
5	HARDIK	27	BHOPAL	8500.00
6	KOMAL	22	MP	4500.00
7	MUFFY	24	INDORE	10000.00

2)if you want to know the total amount of salary on each customer, then the GROUP BY query would be as follows –

```
SELECT NAME, SUM(SALARY) FROM  
CUSTOMERS
```

```
GROUP BY NAME;
```

This would produce the following result –

NAME	SUM(SALARY)
HARDIK	8500.00
HARDIK	8500.00
KAUSHIK	8500.00
KOMAL	4500.00
MUFFY	10000.00
RAMESH	3500.00

5.1.4SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.

- An alias only exists for the duration of the query.

Alias Column Syntax

```
SELECT column_name AS alias_name  
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

Example

```
SELECT ROLLNO AS RNO, NAME FROM  
STUDENT ST;
```

5.2 Constraints

- SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.
- Constraints can be divided into the following two types,
- Column level constraints: Limits only column data.
- Table level constraints: Limits whole table data.
- Constraints are used to make sure that the integrity of data is maintained in the database.

Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraint

- NOT NULL constraint restricts a column from having a NULL value.
- Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.
- It enforces a column to contain a proper value.
- One important point to note about this constraint is that it cannot be defined at table level.

Syntax

```
CREATE TABLE table_name
(
    column_name1 datatype NOT NULL,
    column_name2datatype ,
    column_name3 datatype
    .....
);
```

Example

```
CREATE TABLE STUDENT  
(  
    S_ID    INTEGER NOT NULL,  
    NAME VARCHAR(60),  
    AGE    INTEGER  
);
```

The above query will declare that the S_ID field of STUDENT table will not take NULL value.

Note:-In openoffice base except NOT NULL and PRIMARY KEY constraint all other constraints are defined only at table level

UNIQUE Constraint

- UNIQUE constraint ensures that a field or column will only have unique values.
- A UNIQUE constraint field will not have duplicate data.
- It may contain null value.
- This constraint can be applied at column level or table level.

Using UNIQUE constraint when creating a Table (Table Level)

- Here we have a simple CREATE query to create a table, which will have a column S_ID with unique values.
- **Note:- UNIQUE constraint cannot be defined at table level in openofficebase. But it will work fine with MySql, MS-ACCESS, ORECLE.**

Syntax

```
CREATE TABLE table_name
(
    column_name1 datatype ,
    column_name2datatype ,
    column_name3 datatype
    .....
    UNIQUE(column_name1,
    column_name2..),
);
```

Example

```
CREATE TABLE STUDENT6  
(  
    S_ID INTEGER ,  
    NAME VARCHAR(60) NOT NULL,  
    AGE INTEGER,  
    UNIQUE(S_ID)  
);
```

The above query will declare that the S_ID field of STUDENT table will only have unique values.

Using UNIQUE constraint after Table is created

Syntax

```
ALTER TABLE table_name ADD UNIQUE  
(column_name);
```

Example

```
ALTER TABLE STUDENT ADD  
UNIQUE(S_ID);
```

The above query specifies that S_ID field of STUDENT table will only have unique value.

Primary Key Constraint

- Primary key constraint uniquely identifies each record in a database.
- A Primary Key must contain unique value and it must not contain null value.
- Usually Primary Key is used to index the data inside the table.
- This constraint can be applied at column level or table level.

Using PRIMARY KEY constraint at Column Level

Syntax:

```
CREATE TABLE table_name  
(  
    Column_name1 datatype PRIMARY KEY,  
    Column_name2 datatype,
```

```
.....  
);
```

Example

```
CREATE TABLE STUDENT  
(  
    S_ID INTEGER PRIMARY KEY,  
    NAME VARCHAR(60) NOT NULL,  
    AGE INTEGER  
);
```

The above command will create a PRIMARY KEY on the S_ID.

Using PRIMARY KEY constraint at Table Level

Syntax:

```
CREATE TABLE table_name  
(  
    Column_name1 datatype,  
    Column_name2 datatype,  
    .....  
    PRIMARY KEY(Column_name1)  
);
```

Example

```
CREATE TABLE STUDENT
```

```
(  
    S_ID INTEGER,  
    NAME VARCHAR(60) NOT NULL,  
    AGE INTEGER,  
    PRIMARY KEY(S_ID)  
);
```

The above command will creates a PRIMARY KEY on the S_ID.

Using PRIMARY KEY constraint after Table is created

Syntax

```
ALTER TABLE table_name ADD PRIMARY  
KEY (column_name);
```

Example

```
ALTER TABLE STUDENT ADD PRIMARY  
KEY (S_ID);
```

The above command will creates a PRIMARY KEY on the S_ID.

Foreign Key Constraint

- Foreign Key field points to primary key of another table.
- This usually creates a kind of link between the tables.

Consider the two tables as shown below:

TABLE:-CUSTOMER_DETAIL

C_ID	CUSTOMER_NAME	ADDRESS
101	ADAM	NOIDA
102	ALEX	DELHI
103	STUART	ROHTAK

TABLE:-ORDER_DETAIL

ORDER_ID	ORDER_NAME	C_ID
10	ORDER1	101
11	ORDER2	103
12	ORDER3	102

- In CUSTOMER_DETAIL table, C_ID is the primary key which is set as foreign key IN ORDER_DETAIL table.
- The value that is entered in C_ID which is set as foreign key in ORDER_DETAIL table must be present in CUSTOMER_DETAIL table where it is set as primary key.
- This prevents invalid data to be inserted into C_ID column of ORDER_DETAIL table.
- If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.

Using FOREIGN KEY constraint at Table Level

Syntax

```
CREATE TABLE child_table_name  
(  
    column_name1datatype,  
    column_name2datatype,....  
    FOREIGN KEY([child_column_name1,  
    child_column_name2...])  
    REFERENCESparent_table_name(parent_colu  
mn_name1, parent_column_name2...)
```

);

Example

```
CREATE TABLE ORDER_DETAIL
(
    ORDER_ID INTEGER PRIMARY KEY,
    ORDER_NAME VARCHAR(60) NOT
NULL,
    C_ID INTEGER,
    FOREIGN KEY(C_ID) REFERENCES
CUSTOMER_DETAIL(C_ID)
);
```

In this query, C_ID in table ORDER_DETAIL is made as foreign key, which is a reference of C_ID column in CUSTOMER_DETAIL table.

Using FOREIGN KEY constraint after Table is created

Syntax

```
ALTER TABLE child_table_name ADD
FOREIGN KEY (child_column_name)
```

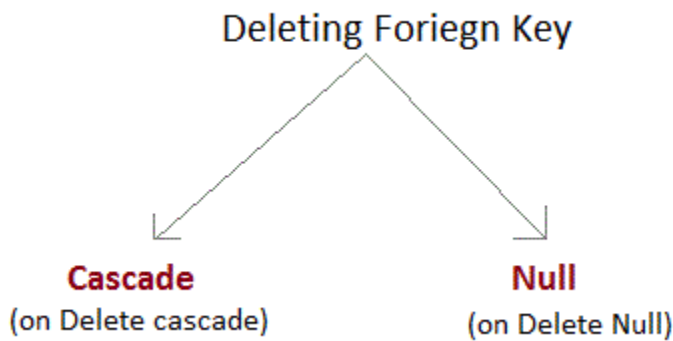
```
REFERENCES parent_table_name  
(parent_column_name);
```

Example

```
ALTER TABLE ORDER_DETAIL ADD  
FOREIGN KEY (C_ID) REFERENCES  
CUSTOMER_DETAIL(C_ID);
```

Behavior of Foreign Key Column on Delete

- There are two ways to maintain the integrity of data in Child table, when a particular record is deleted in the main table.
- When two tables are connected with Foreign key, and certain data in the main table is deleted, for which a record exists in the child table, then we must have some mechanism to save the integrity of data in the child table.



- **On Delete Cascade:** This will remove the record from child table, if that value of foreign key is deleted from the main table.
- **On Delete Null :** This will set all the values in that record of child table as NULL, for which the value of foreign key is deleted from the main table.
- If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

ERROR : Record in child table exist

CHECK Constraint

- CHECK constraint is used to restrict the value of a column between range.

- It performs check on the values, before storing them into the database.
- It's like condition checking before saving data into a column.

Using CHECK constraint at Table Level

Syntax

```
CREATE TABLE table_name
(
    Column_name1 datatype,
    Column_name2 datatype,
    ....
    CHECK (condition)
);
```

Example

```
CREATE TABLE STUDENT
(
    S_ID INTEGER ,
    NAME VARCHAR(60) NOT NULL,
    AGE INTEGER,
    CHECK(S_ID>0)
);
```

The above query will restrict the s_id value to be greater than zero.

Using CHECK constraint after Table is created

Syntax

```
ALTER table_name ADD CHECK(condition);
```

Example

```
ALTER TABLE STUDENT ADD  
CHECK(S_ID > 0);
```

5.3 SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc.

SQL functions are divided into two categories,

- Aggregate Functions
- Scalar Functions

For understanding these functions consider the following EMP table

EID	NAME	AGE	SALARY
401	ANU	22	9000
402	SHANE	29	8000
403	ROHAN	34	6000
404	SCOTT	44	10000
405	TIGER	35	8000

5.3.1 Aggregate Functions

These functions return a single value after performing calculations on a group of values.

Following are some of the frequently used Aggregate functions.

AVG() Function

Average returns average value after calculating it from values in a numeric column.

Syntax

```
SELECT AVG(column_name) FROM  
table_name;
```

Example

SQL query to find average salary will be,

```
SELECT AVG(SALARY) FROM EMP;
```

Output

AVG(SALARY)
8200

COUNT() function

Syntax

```
SELECT COUNT (column_name) FROM  
table_name;
```

Example

SQL query to count employees, satisfying specified condition is,

```
SELECT COUNT(NAME) FROM EMP  
WHERE SALARY = 8000;
```

Output

COUNT(NAME)
2

COUNT(DISTINCT)

```
SELECT COUNT(DISTINCT SALARY)  
  
FROM EMP;
```

Output

COUNT(DISTINCT SALARY)

4

FIRST() Function

First function returns first value of a selected column

Syntax

```
SELECT FIRST(column_name) FROM table-  
name;
```

Example

```
SELECT FIRST(SALARY) FROM EMP;
```

Output

FIRST(SALARY)
9000

LAST() Function

LAST function returns the return last value of the selected column.

Syntax

SELECT LAST(column_name) FROM table-name;

Example

SELECT LAST(SALARY) FROM EMP;

Output

LAST(SALARY)
8000

MAX() Function

MAX function returns maximum value from selected column of the table.

Syntax

SELECT MAX(column_name) FROM table-name;

Example

SQL query to find the Maximum salary will be,

SELECT MAX(SALARY) FROM EMP;

Output

MAX(SALARY)
10000

MIN() Function

MIN function returns minimum value from a selected column of the table.

Syntax

```
SELECT MIN(column_name) FROM table-  
name;
```

Example

SQL query to find minimum salary is,

```
SELECT MIN(SALARY) FROM EMP;
```

Output

MIN(salary)
6000

SUM() Function

SUM function returns total sum of a selected columns numeric values.

Syntax

SELECT SUM(column_name) FROM table-name;

Example

SQL query to find sum of salaries will be,

SELECT SUM(SALARY) FROM EMP;

Output

SUM(SALARY)
41000

5.3.2 Scalar Functions

- Scalar functions return a single value from an input value. Following are some frequently used Scalar Functions in SQL.

UCASE() / UPPER() Function

UCASE() Function in open office and oracle use word UPPER()

- UCASE function is used to convert value of string column to Uppercase characters.

Syntax

SELECT UCASE(column_name) FROM table-name;

Example

SELECT UCASE(name) FROM emp;

SELECT UPPER(name) FROM emp;

Output

UCASE(name)
ANU
SHANE
ROHAN
SCOTT
TIGER

LCASE() / LOWER()Function

LCASE() Function in open office and **oracle use word lower()**

- LCASE function is used to convert value of string columns to Lowecase characters.

Syntax

SELECT LCASE(column_name) FROM table-name;

Example

SELECT LCASE(NAME) FROM EMP;

SELECT lower(NAME) FROM EMP;

Output

LCASE(NAME)
anu
shane
rohan
scott
tiger

MID() / SUBSTR() Function

MID() Function in open office and oracle use word substr()

- MID function is used to extract substrings from column values of string type in a table.

Syntax

```
SELECT MID(column_name, start, length)
FROM table-name;
```

Example

```
SELECT MID(NAME,2,2) FROM EMP;
```

Select substr(name,2,2) from emp;

Output

MID(NAME,2,2)
NU
HA
OH
CO
IG

ROUND() Function

- ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values.

Syntax

```
SELECT ROUND(column_name, decimals)
FROM table-name;
```

Example

```
SELECT ROUND(SALARY) FROM EMP;
```

Output

ROUND(SALARY)
9001
8001
6000
10000
8000

5.4SQL Sequence

- Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like MySQL supports AUTO_INCREMENT in place of Sequence.

- AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is inserted into the table.
- Sequence is also somewhat similar to AUTO_INCREMENT but it has some additional features too.

Creating a Sequence

Syntax

```
CREATE SEQUENCE sequence-name  
  START WITH initial-value  
  INCREMENT BY increment-value  
  MAXVALUE maximum-value  
  CYCLE | NOCYCLE;
```

- The **initial-value** specifies the starting value for the Sequence.
- The **increment-value** is the value by which sequence will be incremented.

- The **maximum-value** specifies the upper limit or the maximum value upto which sequence will increment itself.
- The keyword **CYCLE** specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the beginning.
- And, **NO CYCLE** specifies that if sequence exceeds MAXVALUE value, an error will be thrown.

Example

Let's start by creating a sequence, which will start from 1, increment by 1 with a maximum value of 999.

```
CREATE SEQUENCE SEQ_1  
START WITH 1  
INCREMENT BY 1  
MAXVALUE 999  
CYCLE;
```

Now let's use the sequence that we just created above.

Below we have a class table,

ID	NAME
1	ABHI
2	ADAM
4	ALEX

```
INSERT INTO CLASS  
VALUE(SEQ_1.NEXTVAL, 'ANU');
```

Resultset table will look like,

ID	NAME
1	ABHI
2	ADAM
4	ALEX
1	ANU

Once you use nextval the sequence will increment even if you don't Insert any record into the table.

5.5SQL VIEW

- A VIEW in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

Syntax

```
CREATE or REPLACE VIEW view_name
AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

- A view is created using data fetched from some other table(s). It's more like a temporary table created with data.

Creating a VIEW

Consider following Sale table,

OI D	ORDER_NA ME	PREVIOUS_BAL ANCE	CUSTOM ER
11	ORD1	2000	ALEX
12	ORD2	1000	ADAM

13	ORD3	2000	ABHI
14	ORD4	1000	ADAM
15	ORD5	2000	ALEX

Example

```
CREATE OR REPLACE VIEW SALE_VIEW  
AS
```

```
SELECT * FROM SALE WHERE  
CUSTOMER = 'ALEX';
```

- The data fetched from SELECT statement will be stored in another object called SALE_VIEW.
- We can use CREATE and REPLACE separately too, but using both together works better, as if any view with the specified name exists, this query will replace it with fresh data.

Displaying a VIEW

- The syntax for displaying the data in a view is similar to fetching data from a table using a SELECT statement.

```
SELECT * FROM SALE_VIEW;
```

Update a VIEW

- UPDATE command for view is same as for tables.

Syntax

```
UPDATE view-name  
SET COLUMNNAME= VALUE  
WHERE condition;
```

Example

```
UPDATE SALE_VIEW  
SET PREVIOUS_BALANCE=5000  
WHERE CUSTOMER = 'ALEX';
```

SQL Dropping a View

- A view is deleted with the DROP VIEW command.

Syntax

DROP VIEW view_name;

Example

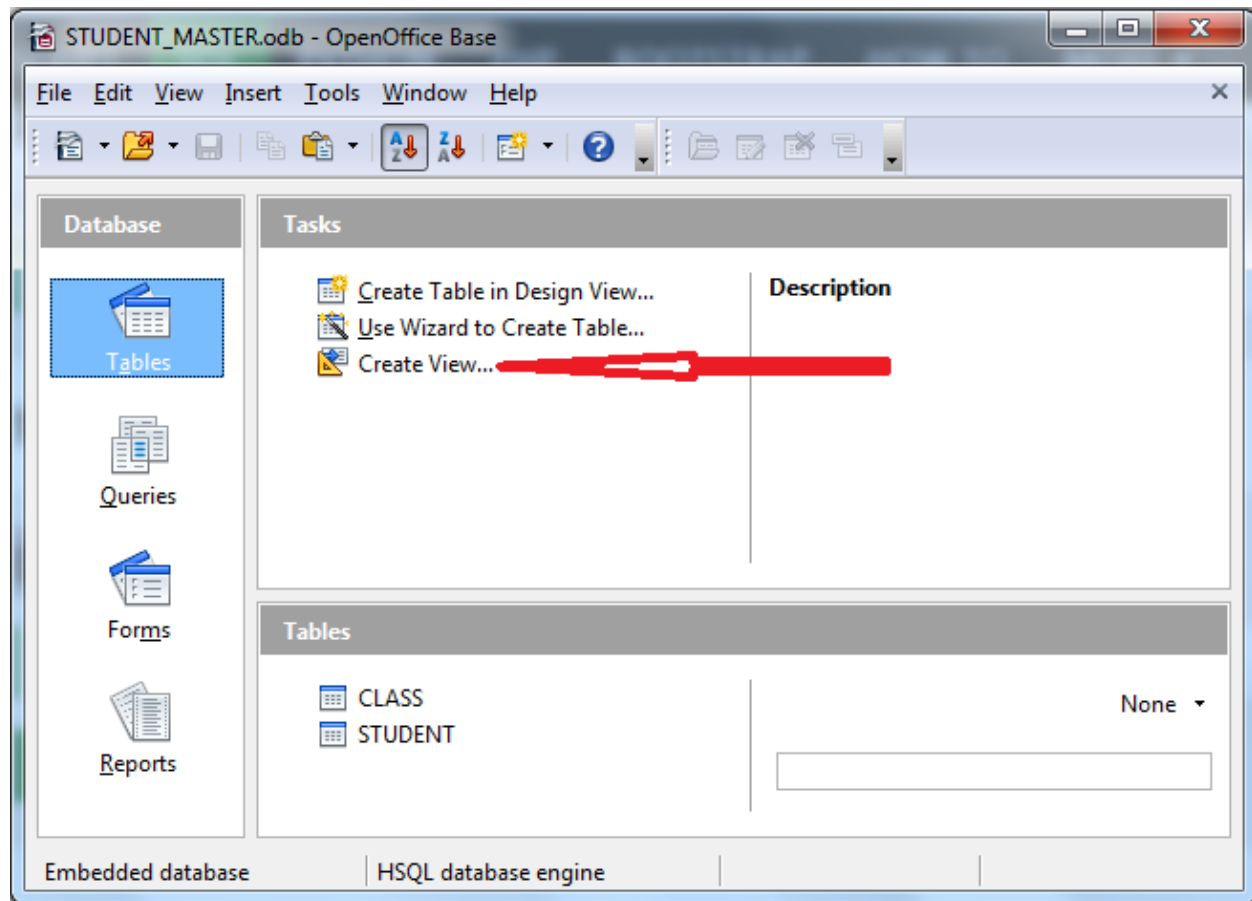
The following SQL drops the " SALE_VIEW " view:

DROP VIEW sale_view ;

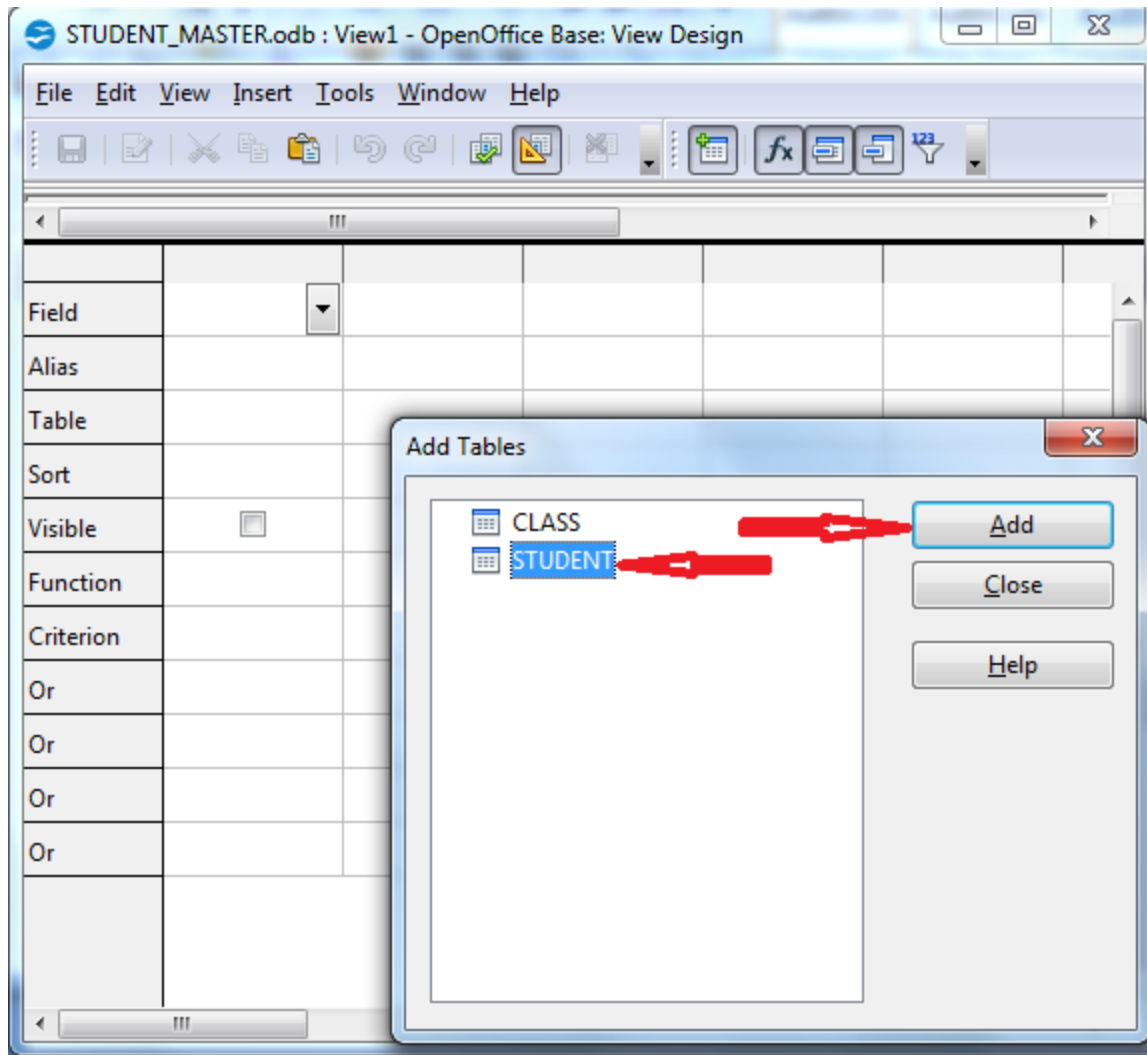
Note:In open office base view is created only in design view.

Steps to create view in openoffice base

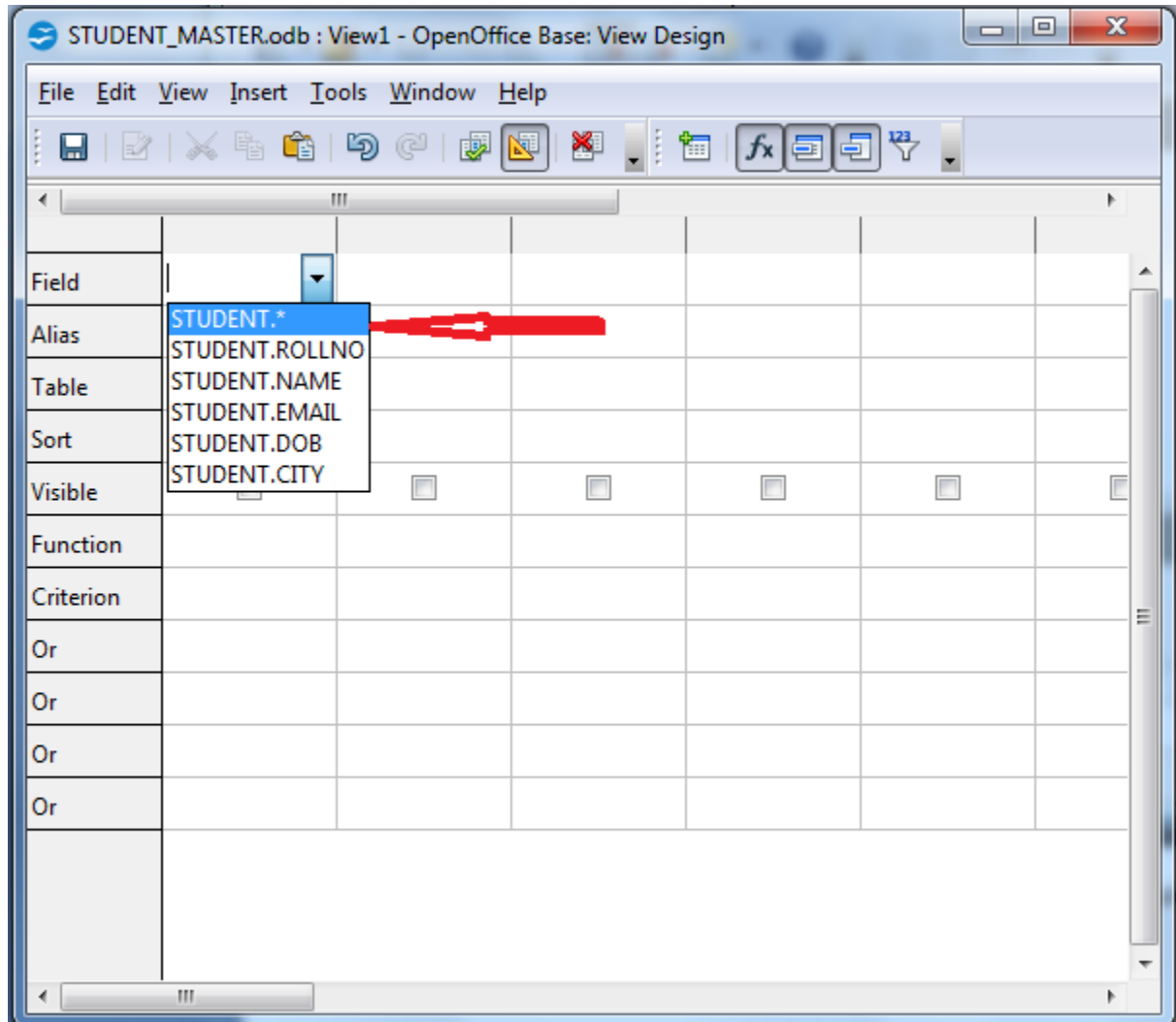
- Click on create view

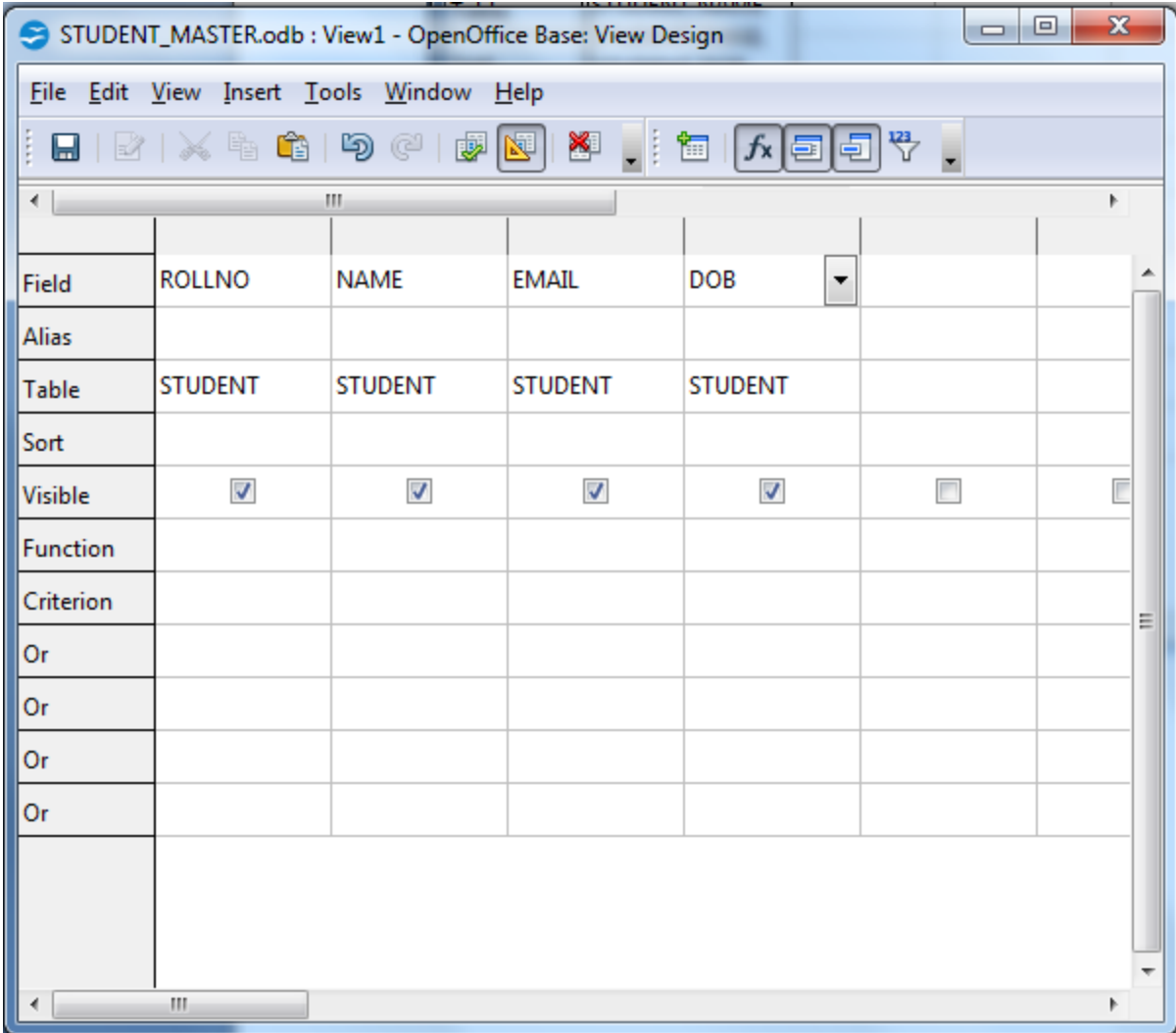


- The following dialogbox is open.

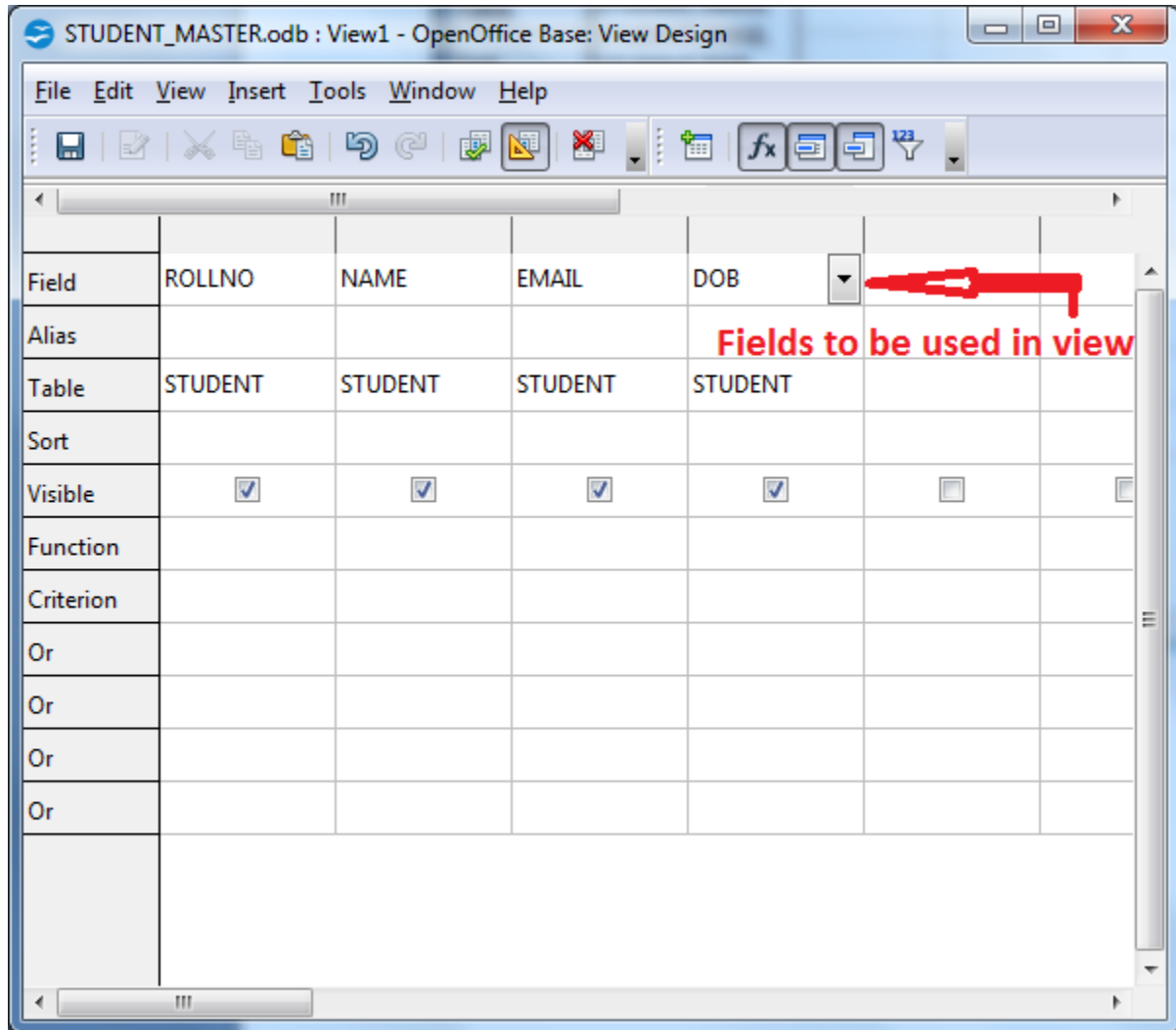


- Select the table for creating the view.lets select the STUDENT table for creating view.

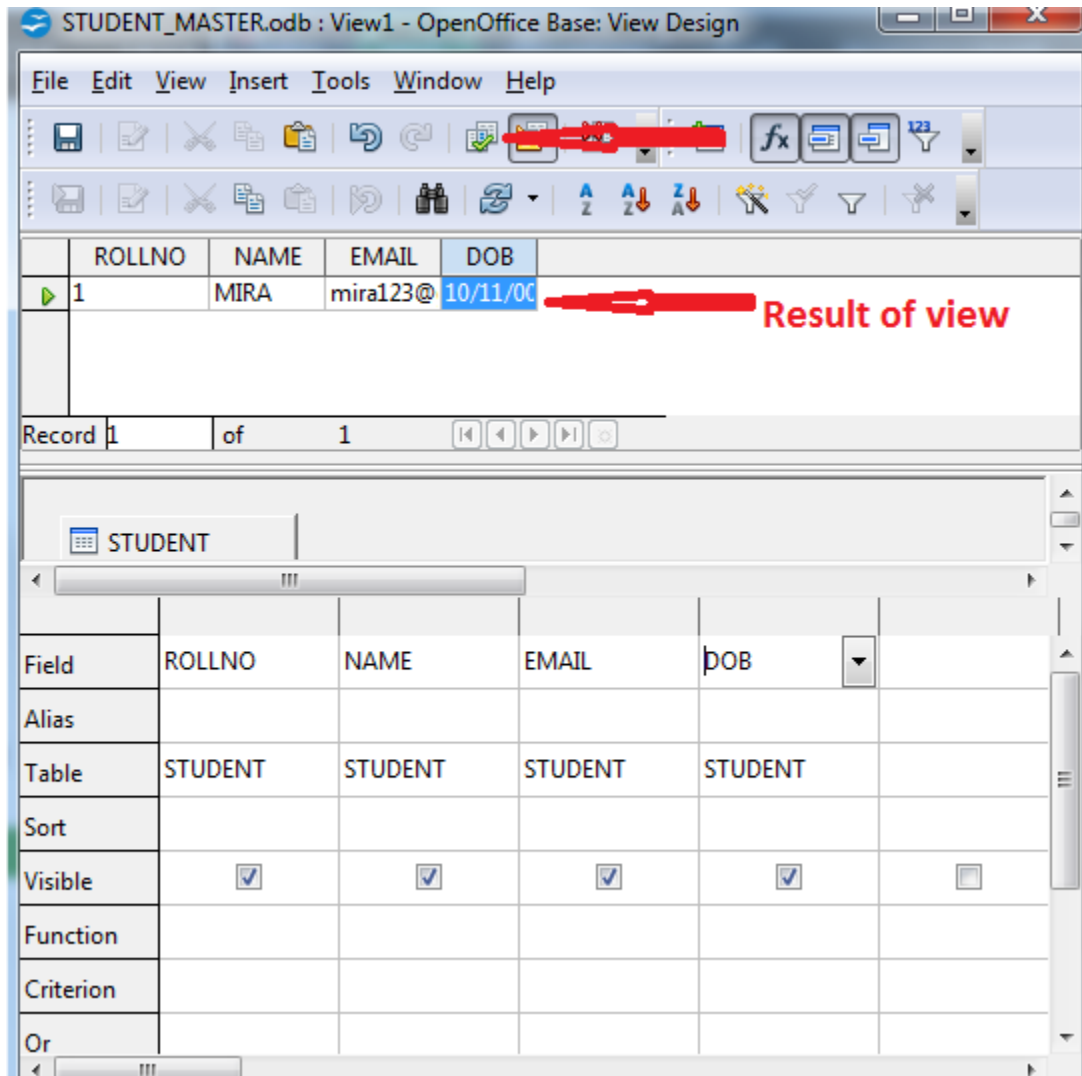




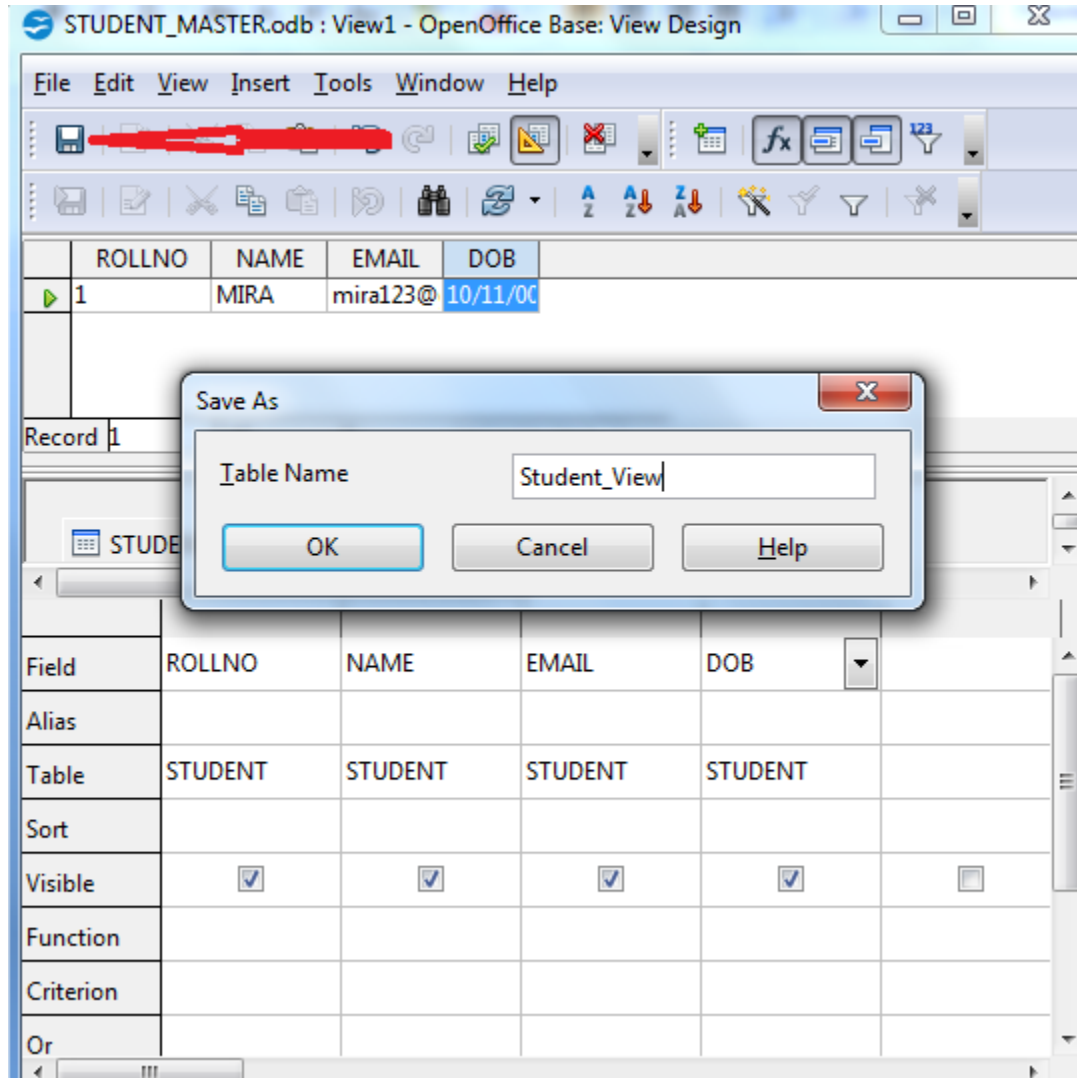
- Select the field which you want to use in the view.



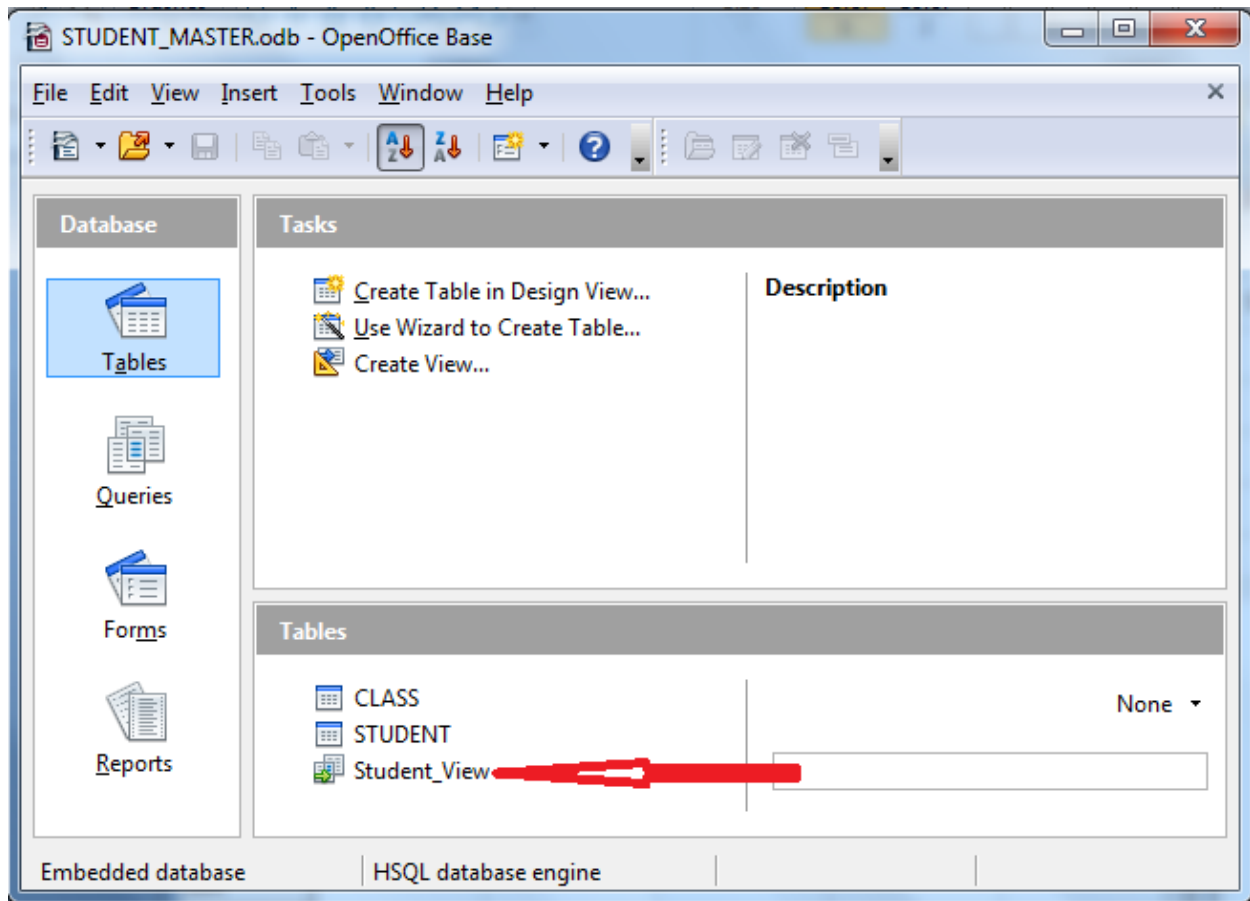
- Click on Run Query.
- The following Dialogbox is open.



- Click on save icon. Give the Proper name to the view Student_view.



- Click ok.
- Created view display in a table list.



- You can use this view same as table. you can do select, update and drop operation on it same as table.

Difference between Table and View.

Sr.No.	Table	View
1.	An RDBMS object which can store data.	An RDBMS object which is a product of multiple tables, overtime you run a “SELECT *” on a view a query run’s in background to fetch you the desired results.
2	You can add/update/delete data from a table.	You cannot add/update/delete any data from a view, to make any changes to the view; you will have to update the data in the source tables that are used to create the view.
3	You can only create or drop the table, basically you	You can easily use replace option to recreate the view, as its

	cannot replace the table object directly as it can physical entry in the RDBMS memory.	just a pseudo name to the query which is running behind.
4	DML operations can be performed.	DML operation is not allowed for the above mentioned reasons.