Contents

1. Order by Clause

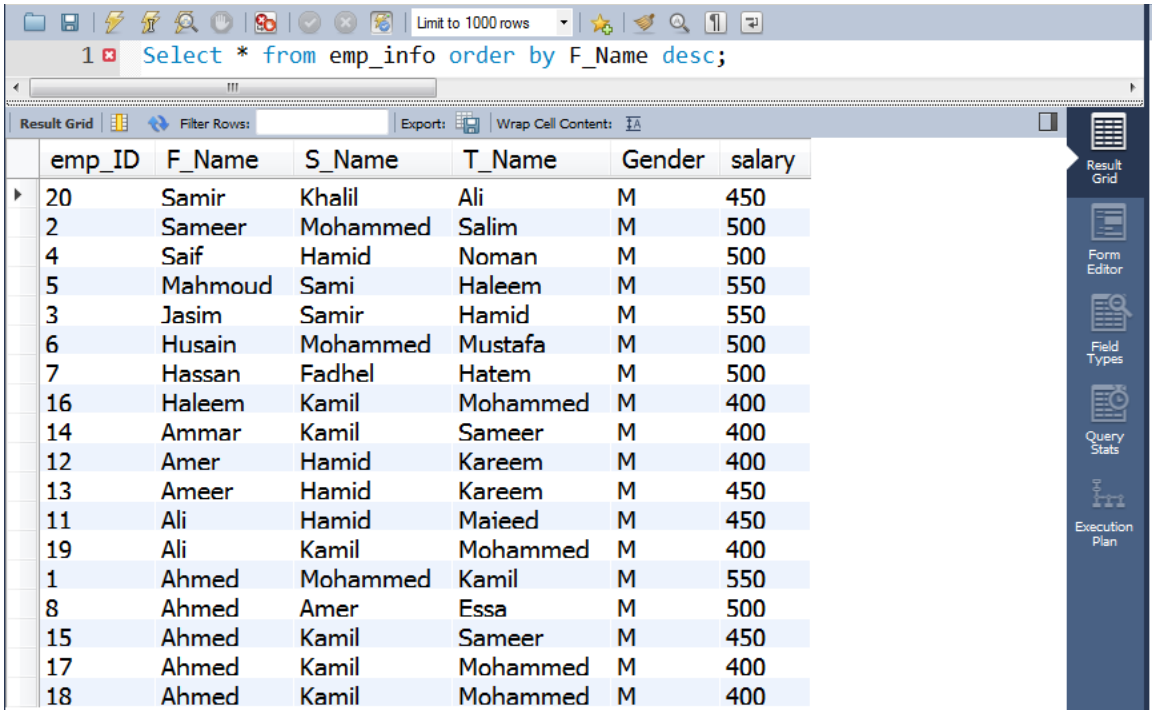
The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.

```
SELECT column-list
FROM table name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

Example: suppose you have the following relation with name emp_info:

emp_ID	F_Name	S_Name	T_Name	Gender	salary
1	Ahmed	Mohammed	Kamil	M	550
2	Sameer	Mohammed	Salim	M	500
3	Jasim	Samir	Hamid	M	550
4	Saif	Hamid	Noman	M	500
5	Mahmoud	Sami	Haleem	M	550
6	Husain	Mohammed	Mustafa	M	500
7	Hassan	Fadhel	Hatem	M	500
8	Ahmed	Amer	Essa	M	500
11	Ali	Hamid	Maieed	M	450
12	Amer	Hamid	Kareem	M	400
13	Ameer	Hamid	Kareem	M	450
14	Ammar	Kamil	Sameer	M	400
15	Ahmed	Kamil	Sameer	M	450
16	Haleem	Kamil	Moha…	M	400
17	Ahmed	Kamil	Moha…	M	400
18	Ahmed	Kamil	Moha…	M	400
19	Ali	Kamil	Moha…	M	400
20	Samir	Khalil	Ali	M	450

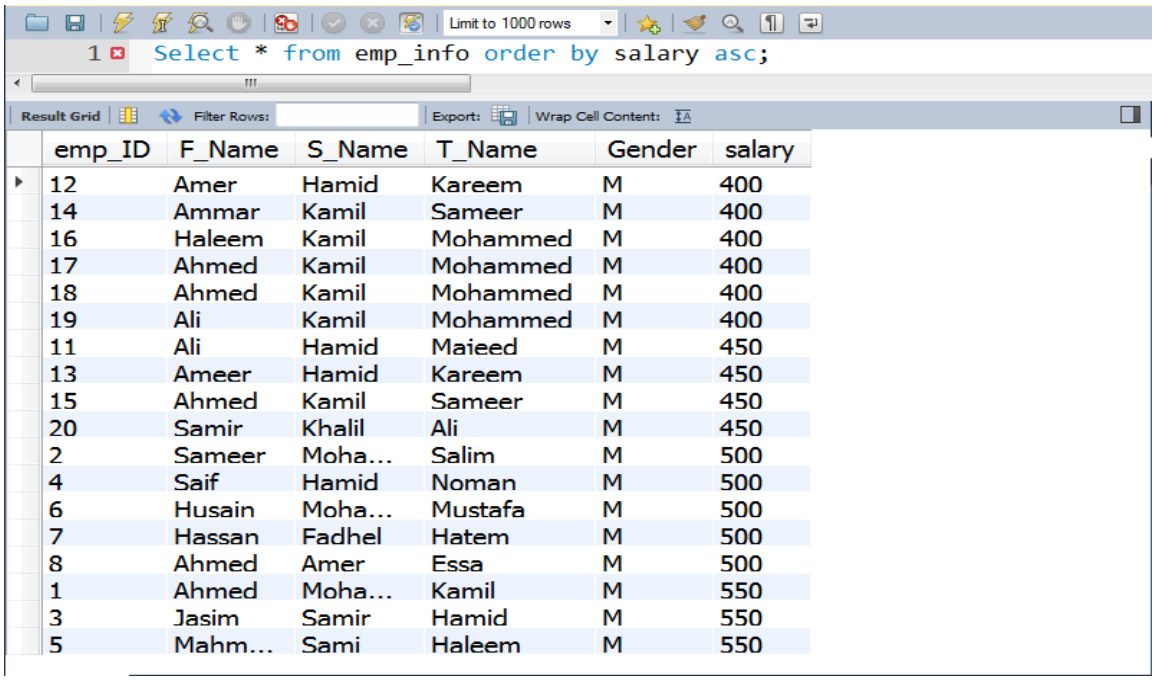
Following is an example, which would sort the result in descending order by F_Name:



```
Select * from emp_info order by F_Name desc;
```

emp_ID	F_Name	S_Name	T_Name	Gender	salary
20	Samir	Khalil	Ali	M	450
2	Sameer	Mohammed	Salim	M	500
4	Saif	Hamid	Noman	M	500
5	Mahmoud	Sami	Haleem	M	550
3	Jasim	Samir	Hamid	M	550
6	Husain	Mohammed	Mustafa	M	500
7	Hassan	Fadhel	Hatem	M	500
16	Haleem	Kamil	Mohammed	M	400
14	Ammar	Kamil	Sameer	M	400
12	Amer	Hamid	Kareem	M	400
13	Ameer	Hamid	Kareem	M	450
11	Ali	Hamid	Maieed	M	450
19	Ali	Kamil	Mohammed	M	400
1	Ahmed	Mohammed	Kamil	M	550
8	Ahmed	Amer	Essa	M	500
15	Ahmed	Kamil	Sameer	M	450
17	Ahmed	Kamil	Mohammed	M	400
18	Ahmed	Kamil	Mohammed	M	400

To sort the table ascending by salary:



```
Select * from emp_info order by salary asc;
```

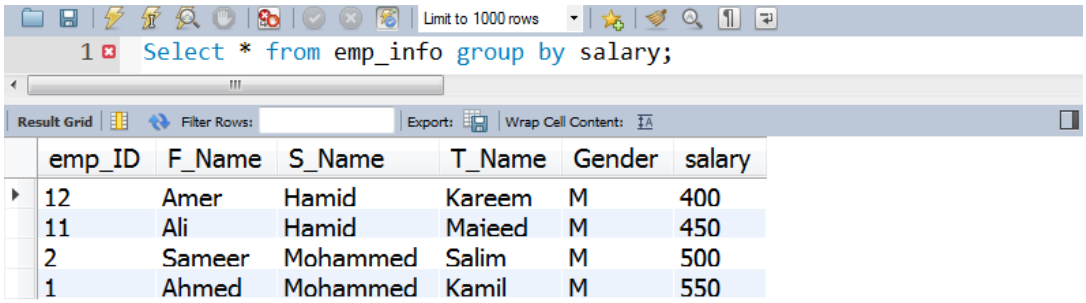
emp_ID	F_Name	S_Name	T_Name	Gender	salary
12	Amer	Hamid	Kareem	M	400
14	Ammar	Kamil	Sameer	M	400
16	Haleem	Kamil	Mohammed	M	400
17	Ahmed	Kamil	Mohammed	M	400
18	Ahmed	Kamil	Mohammed	M	400
19	Ali	Kamil	Mohammed	M	400
11	Ali	Hamid	Maieed	M	450
13	Ameer	Hamid	Kareem	M	450
15	Ahmed	Kamil	Sameer	M	450
20	Samir	Khalil	Ali	M	450
2	Sameer	Moha...	Salim	M	500
4	Saif	Hamid	Noman	M	500
6	Husain	Moha...	Mustafa	M	500
7	Hassan	Fadhel	Hatem	M	500
8	Ahmed	Amer	Essa	M	500
1	Ahmed	Moha...	Kamil	M	550
3	Jasim	Samir	Hamid	M	550
5	Mahm...	Sami	Haleem	M	550

2. Group by Clause

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

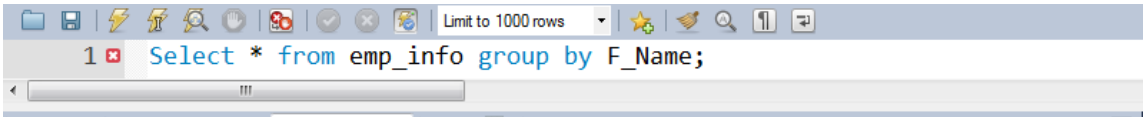
```
SELECT column1, column2
FROM table name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

Example: group emp_info by salary:



emp_ID	F_Name	S_Name	T_Name	Gender	salary
12	Amer	Hamid	Kareem	M	400
11	Ali	Hamid	Maieed	M	450
2	Sameer	Mohammed	Salim	M	500
1	Ahmed	Mohammed	Kamil	M	550

By F_Name:



emp_ID	F_Name	S_Name	T_Name	Gender	salary
1	Ahmed	Mohammed	Kamil	M	550
11	Ali	Hamid	Maieed	M	450
13	Ameer	Hamid	Kareem	M	450
12	Amer	Hamid	Kareem	M	400
14	Ammar	Kamil	Sameer	M	400
16	Haleem	Kamil	Mohammed	M	400
7	Hassan	Fadhel	Hatem	M	500
6	Husain	Mohammed	Mustafa	M	500
3	Jasim	Samir	Hamid	M	550
5	Mahmoud	Sami	Haleem	M	550
4	Saif	Hamid	Noman	M	500
2	Sameer	Mohammed	Salim	M	500
20	Samir	Khalil	Ali	M	450

3. Constraints

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be column level or table level. Column level constraints are applied only to one column where table level constraints are applied to the whole table.

- NOT NULL
- DEFAULT
- UNIQUE Constraint
- PRIMARY Key
- FOREIGN Key
- CHECK Constraint
- INDEX

3.1 NOT NULL Constraint

By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column.

A NULL is not the same as no data, rather, it represents unknown data.

Example: For example, the following SQL creates a new table called customer and adds five columns (ID, f_name, Age, email, salary), three of which, ID and f_Name and Age, specify not to accept NULLs:

```
create table customer
(ID int NOT NULL,salary int, email varchar(20),
f_name varchar(20) NOT NULL, Age int NOT NULL);
```

To modify column constraint and make it NOTNULL:

```
alter table customer
modify salary int not null;
```

3.2 DEFAULT Constraint:

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

Example: For example, the following SQL creates a new table called CUSTOMERS and adds five columns. Here, SALARY column is set to 5000.00 by default, so in case INSERT INTO statement does not provide a value for this column, then by default this column would be set to 5000.00.

```
CREATE TABLE customers
( ID INT NOT NULL,
 name VARCHAR (20) NOT NULL,
 age INT NOT NULL,
 address CHAR (25) ,
 salary DECIMAL (18, 2) DEFAULT 5000.00 );
```

If CUSTOMERS table has already been created, then to add a DFAULT constraint to SALARY column, you would write a statement similar to the following:

```
Alter table customers
modify column salary decimal(10,2) default 5000.00;
```

To drop DEFAULT constraint:

```
Alter table customer
alter column salary drop default;
```

3.3 Unique Constraint

The UNIQUE Constraint prevents two records from having identical values in a particular column. In the CUSTOMERS table, for example, you might want to prevent two or more people from having identical age.

Example: For example, the following SQL creates a new table called customers and adds five columns. Here, AGE column is set to UNIQUE, so that you can not have two records with same age:

```
CREATE TABLE customers(
ID INT NOT NULL,
name VARCHAR (20) NOT NULL,
address CHAR (25) ,
salary DECIMAL (18, 2),
 Age INT NOT NULL UNIQUE, PRIMARY KEY (ID) );
```

If customers table has already been created, then to add a UNIQUE constraint to AGE column, you would write a statement similar to the following:

```
ALTER TABLE customers MODIFY Age INT NOT NULL UNIQUE;
```

Or

```
ALTER TABLE CUSTOMERS
ADD CONSTRAINT myUniqueConstraint UNIQUE(Age, salary);
```

3.4 PRIMARY Key

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a **composite key**.

If a table has a primary key defined on any field(s), then you can not have two records having the same value of that field(s).

Note: You would use these concepts while creating database tables.

Create Primary Key: Here is the syntax to define ID attribute as a primary key in a CUSTOMERS table.

```
CREATE TABLE customers(
ID INT NOT NULL,
name VARCHAR (20)NOT NULL,
Age INT NOT NULL,
address CHAR (25) ,
salary DECIMAL (18, 2),PRIMARY KEY (ID) );
```

To create a PRIMARY KEY constraint on the "ID" column when CUSTOMERS table already exists, use the following SQL syntax:

```
ALTER TABLE customers ADD primary key (ID);
```

NOTE: If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

For defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE customers(
ID INT NOT NULL,
name VARCHAR (20)NOT NULL,
Age INT NOT NULL,
address CHAR (25) ,
salary DECIMAL (18, 2),PRIMARY KEY (ID,name) );
```

Delete Primary Key: You can clear the primary key constraints from the table, Use Syntax:

```
ALTER TABLE customers DROP primary key;
```

3.5 FOREIGN Key:

A foreign key is a key used to link two tables together. This is sometimes called a referencing key. Primary key field from one table and insert it into the other table where it becomes a foreign key i.e., Foreign Key is a column or a combination of columns, whose values match a Primary Key in a different table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

If a table has a primary key defined on any field(s), then you can not have two records having the same value of that field(s).

Example: Consider the structure of the two tables as follows:

customers table:

```
CREATE TABLE customers(
ID INT NOT NULL,
name VARCHAR (20)NOT NULL,
Age INT NOT NULL,
address CHAR (25) ,
salary DECIMAL (18, 2),PRIMARY KEY (ID) );
```

orders table:

```
CREATE TABLE orders
( id INT NOT NULL,
 date DATETIME,
 amount DOUBLE,
 customer_id INT,
 CONSTRAINT ord_cust
 FOREIGN KEY (customer_id)
 REFERENCES customers(id)
 ON DELETE NO ACTION
 ON UPDATE CASCADE);
```

Referential Actions

For storage engines supporting foreign keys, MySQL rejects any INSERT or UPDATE operation that attempts to create a foreign key value in a child table if there is no a matching candidate key value in the parent table.

- CASCADE: Delete or update the row from the parent table, and automatically delete or update the matching rows in the child table. Both ON DELETE CASCADE and ON UPDATE CASCADE are supported. Between two tables, do not define several ON UPDATE CASCADE clauses that act on the same column in the parent table or in the child table.

- SET NULL: Delete or update the row from the parent table, and set the foreign key column or columns in the child table to NULL. Both ON DELETE SET NULL and ON UPDATE SET NULL clauses are supported.
    - If you specify a SET NULL action, *make sure that you have not declared the columns in the child table as* NOT NULL.
- RESTRICT: Rejects the delete or update operation for the parent table. Specifying RESTRICT (or NO ACTION) is the same as omitting the ON DELETE or ON UPDATE clause.
- NO ACTION: A keyword from standard SQL. In MySQL, equivalent to RESTRICT. The MySQL Server rejects the delete or update operation for the parent table if there is a related foreign key value in the referenced table. Some database systems have deferred checks, and NO ACTION is a deferred check. In MySQL, foreign key constraints are checked immediately, so NO ACTION is the same as RESTRICT.

If ORDERS table has already been created, and the foreign key has not yet been, use the syntax for specifying a foreign key by altering a table.

```
ALTER TABLE tablename
DROP FOREIGN KEY fkconstraint;

ALTER TABLE tablename ADD CONSTRAINT newfkconstraint
 FOREIGN KEY (column1)
 REFERENCES parenttablname(pkcolumn)
 ON DELETE SET NULL
 ON UPDATE SET NULL;
```

DROP a FOREIGN KEY Constraint:

To drop a FOREIGN KEY constraint, use the following SQL:

```
ALTER TABLE tablename DROP foreign key foreignKeyName;
```

NOTE: if you did not determine FOREIGN KEY name, an automatic name will assigned, so if you want to delete the foreign key you should determine the name of foreign key.