**DAY 1**

**WHAT IS SQL:**

sql is the standared language for dealing with

relational database it can be used to create, read,

update and delete database records.

sql stands for structured query language and is used

for communicating with the database.

**NORMALIZATION:**

Normalization is the process of reducing data redundancy

by organizing fields and table of a database.

Insertion anomaly, update anomaly, and deletion

animaly it is a database design technique.

**CONSTRAINTS IN SQL:**

constraints are used to specify rules for data in table.

1.Not null:

Not null constraints ensures that a column cannot

have a null value.

2.Default:

Default constraints sets a default value for a column

when no value is specified.

3.Unique:

Unique constraint ensures that all values in a

column are different.

4.Primary key:

Primary key constraint uniquely identified

each record in a table.

5.Foreign key:

Foreign key is used to link two tables and

it refers primary key.

6.Check:

Ensures that the values in a column

satisfies a specific condition.

**DATA-TYPES:**

The data type of a column defines what

value the column can hold integer, character, money,

date and time, binary and so on.

Numerics:

The exact numeric data types are SMALLINT,

INTEGER, BIGINT, NUMERIC(p,s), DECIMAL(p,s).

Exact sql numeric data type means that the

value is stored as a literal represntation of the

numbers value.

character:

Stores string of letter, numbers, and symbols

data types CHAR, VARCHAR are collectively referred to

as character string types and the values of character

string types are known as character strings.

DateTime:

In sql datetime dete date type is used for values

that contain both date and time.

**DAY 2**

**CREATING DATABASE :**

The CREATE DATABASE statement is used to create a new sql database.

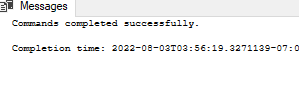
Syntax:

CREATE DATABASE database name;

Example:

create database Training;

use Training;



**CREATING TABLE :**

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

* Name the table
* Define the columns
* Assign data type of columns

Syntax :

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
 );

Example:

create table student(

s\_id int identity(100,1),

s\_name varchar(20),

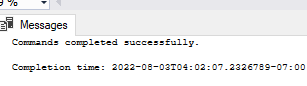
s\_age int check(s\_age>15),

s\_gender varchar(20),

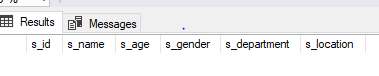
s\_department varchar(20),

s\_location varchar(20)

);



select \* from student;



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S\_id | S\_name | S\_age | S\_gender | S\_department | S\_location |

|  |
| --- |
|  |

**SQL INSERT INTO :**

The INSERT INTO statement is used to insert new records in a table

Syntax :

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

Example :

insert into student values('jacob',21,'male','cse','Tirunelveli');

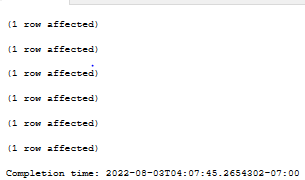
insert into student values('williams',21,'male','cse','chennai');

insert into student values('james',18,'male','civil','mumbai');

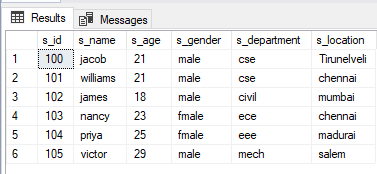
insert into student values('nancy',23,'fmale','ece','chennai');

insert into student values('priya',25,'fmale','eee','madurai');

insert into student values('victor',29,'male','mech','salem');



select \* from student;



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S\_id | S\_name | S\_age | S\_genter | S\_department | S\_locatiom |
| 100 | Jacob | 21 | Male | Cse | Tirnelveli |
| 101 | Williams | 21 | Male | Cse | Chennai |
| 102 | James | 18 | Male | Civil | Mumbai |
| 103 | Nancy | 23 | Female | Ece | Chennai |
| 104 | Priya | 25 | Female | Eee | Madurai |
| 105 | victor | 29 | male | mech | Salem |

**SQL SELECT** :

The SELECT statement is used to select data from a database.

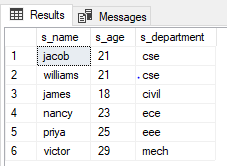
The data returned is stored in a result table, called the result-set.

Syntax :

SELECT column1, column2, ...  
FROM table\_name

Example :

select s\_name,s\_age,s\_department from student;



|  |  |  |
| --- | --- | --- |
| S\_name | S\_age | S\_department |
| Jacob | 21 | Cse |
| Williams | 21 | Cse |
| James | 18 | Civil |
| Nancy | 23 | Ece |
| Priya | 25 | Eee |
| victor | 29 | Mech |

**SQL DELETE** :

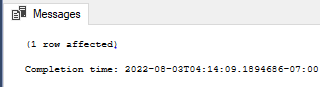
The DELETE statement is used to delete existing records in a table.

Syntax :

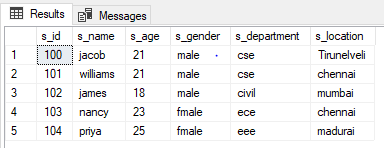
DELETE FROM table\_name WHERE condition;

Example:

delete from student where s\_name='victor’



select \* from student;



**DAY 3**

**SQL WHERE CLAUSE :**

The WHERE clause is used to filter records.

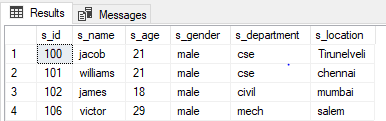
It is used to extract only those records that fulfill a specified condition.

Syntax:

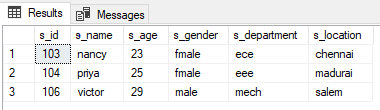
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

Example:

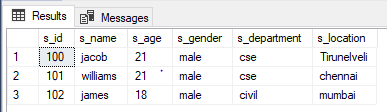
select \* from student where s\_gender='male'



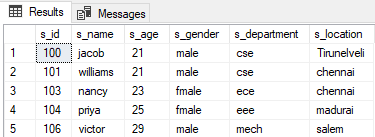
select \* from student where s\_age>21



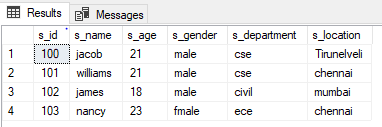
select \* from student where s\_id<103



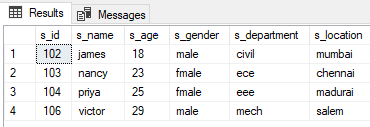
select \* from student where s\_age>=21



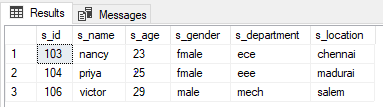
select \* from student where s\_id<=103



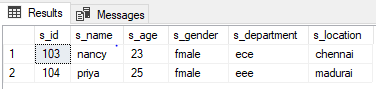
select \* from student where s\_department!='cse'



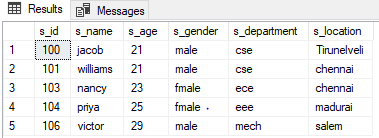
select \* from student where s\_age between 23 and 29



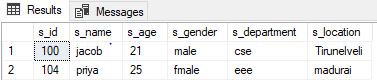
select \* from student where s\_gender like 'f%'



select \* from student where s\_age like '2\_'



select \* from student where s\_location in ('Tirunelveli','madurai')



**SQL UPDATE STATEMENT :**

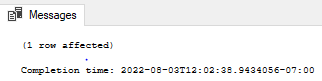
The UPDATE statement is used to modify the existing records in a table.

Syntax :

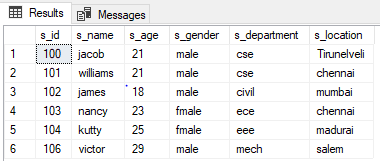
UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

Example :

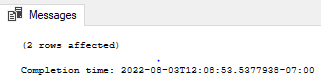
update student set s\_name='kutty' where s\_id=104



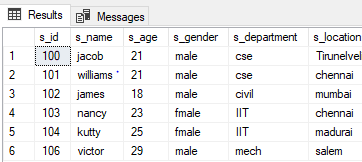
select \* from student;



update student set s\_department='IIT' where s\_gender='fmale'



select \* from student;



**SQL OPERATORS :**

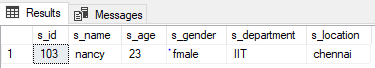
The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition.

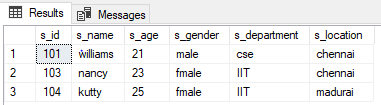
The NOT operator displays a record if the condition(s) is NOT TRUE.

Example :

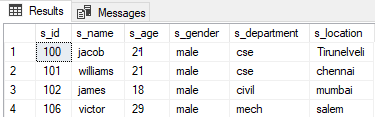
select \* from student where s\_department='IIT' and s\_location='chennai'



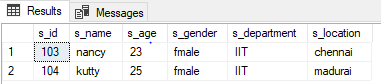
select \* from student where s\_department='IIT' or s\_location='chennai'



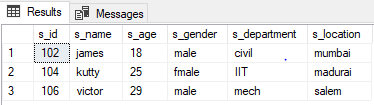
select \* from student where not s\_department='IIT'



select \* from student where s\_department='IIT' and (s\_location='chennai' or s\_location='madurai')



select \* from student where not s\_department='cse' and not s\_location='chennai'



**HAVING CLAUSE :**

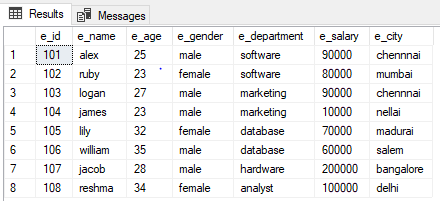
Having clause is used in combination with group by to impose condition on groups.

Syntax:

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

Example:

select \* from employee;



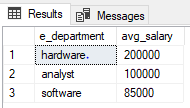
select e\_department,avg(e\_salary) as avg\_salary

from employee

group by e\_department

having avg(e\_salary)>70000

order by avg(e\_salary) desc;



select e\_department,count(e\_id) as total

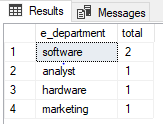
from employee

where e\_salary>75000

group by e\_department

having count(e\_id)>=1

order by count(e\_id) desc;



**DAY 4**

**SQL ORDER BY :**

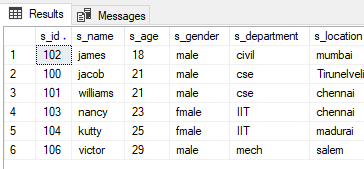
The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax :

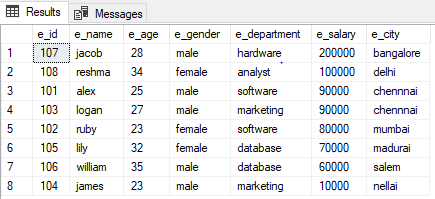
SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

Example :

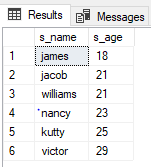
select \* from student order by s\_age asc;



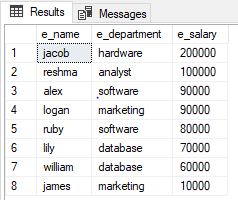
select \* from employee order by e\_salary desc;



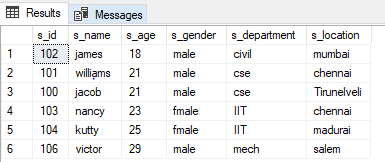
select s\_name,s\_age from student order by s\_age;



select e\_name,e\_department,e\_salary from employee order by e\_salary desc;



select \* from student order by s\_age asc, s\_name desc;



**SQL GROUP BY :**

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax :

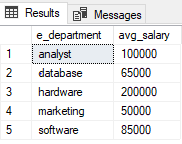
SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

Example :

select e\_department, avg(e\_salary) as avg\_salary

from employee

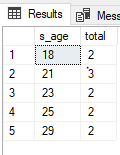
group by e\_department;



select s\_age,count(s\_id) as total

from student

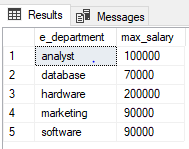
group by s\_age;



select e\_department,max(e\_salary) as max\_salary

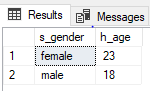
from employee

group by e\_department;



select s\_gender,min(s\_age) as h\_age from student

group by s\_gender;



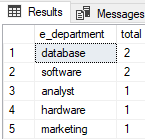
select e\_department,count(e\_id) as total

from employee

where e\_salary>50000

group by e\_department

order by count(e\_id) desc;

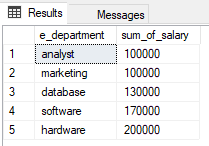


select e\_department, sum(e\_salary) as sum\_of\_salary

from employee

group by e\_department

order by sum(e\_salary) asc;



**AGGREGATE FUNCTION :**

MAX :

The MAX() function returns the largest value of the selected column

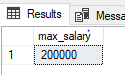
Syantax :

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

Example :

select max(e\_salary) as max\_salary

from employee;

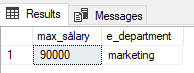


select max(e\_salary) as max\_salary,e\_department

from employee

where e\_department='marketing'

group by e\_department;



MIN :

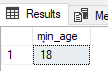
The MIN() function returns the smallest value of the selected column

Syantax :

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

Example :

select min(s\_age) as min\_age from student

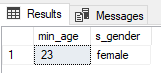


select min(s\_age) as min\_age,s\_gender

from student

where s\_gender='female'

group by s\_gender;



AVG :

The AVG() function returns the average value of a numeric column.

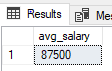
Syantax :

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

Example :

select avg(e\_salary) as avg\_salary

from employee;

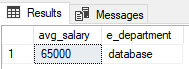


select avg(e\_salary) as avg\_salary,e\_department

from employee

where e\_department='database'

group by e\_department;



COUNT :

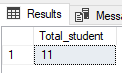
The COUNT() function returns the number of rows that matches a specified criterion.

Syantax :

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

Example :

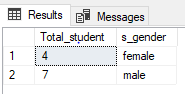
select count(\*) as Total\_student from student;



select count(\*) as Total\_student ,s\_gender

from student

group by s\_gender;



SUM :

The SUM() function returns the total sum of a numeric column.

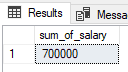
Syantax :

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

Example :

select sum(e\_salary) as sum\_of\_salary

from employee;

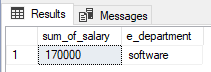


select sum(e\_salary) as sum\_of\_salary,e\_department

from employee

where e\_department='software'

group by e\_department;



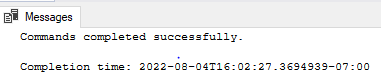
DROP :

The DROP statement is used to drop an existing SQL database and table.

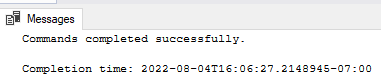
Example :

alter table students

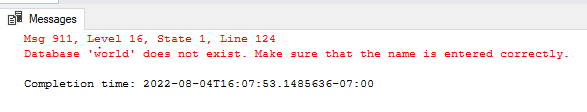
drop column s\_genter;



drop database world;



use world;



**DAY 5**

**SQL JOIN :**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Types :

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

INNER JOIN :

The INNER JOIN keyword selects records that have matching values in both tables.

Syntax :

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;

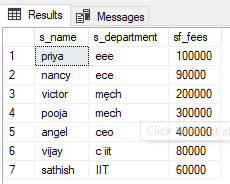
Example :

select s.s\_name,s.s\_department,sf.sf\_fees

from students as s

inner join students\_fee as sf

on s.s\_id=sf.sf\_id;



## LEFT JOIN :

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

### Syntax :

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

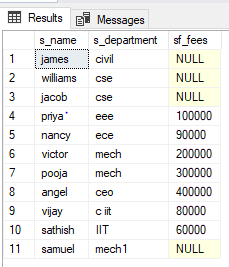
Example :

select s.s\_name,s.s\_department,sf.sf\_fees

from students as s

left join students\_fee as sf

on s.s\_id=sf.sf\_id;



**RIGHT JOIN :**

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

### Syntax :

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

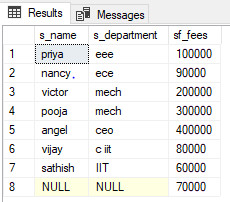
Example :

select s.s\_name,s.s\_department,sf.sf\_fees

from students as s

right join students\_fee as sf

on s.s\_id=sf.sf\_id;



## FULL JOIN :

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

### Syntax :

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;

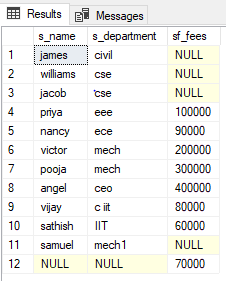
Example :

select s.s\_name,s.s\_department,sf.sf\_fees

from students as s

full join students\_fee as sf

on s.s\_id=sf.sf\_id;



**DAY 6**

**BETWEEN :**

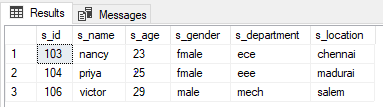
The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

Syntax :

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name*BETWEEN *value1* AND *value2;*

Example :

select \* from student where s\_age between 23 and 29;



## LIKE :

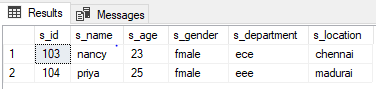
The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Syntax :

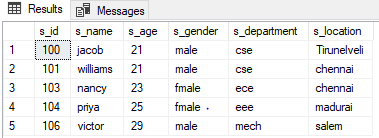
SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

Example :

select \* from student where s\_gender like 'f%'



select \* from student where s\_age like '2\_'



## Aliases :

SQL aliases are used to give a table, or a column in a table, a temporary name.

An alias is created with the AS keyword.

Syntax :

SELECT *column\_name* AS *alias\_name*  
FROM *table\_name;*

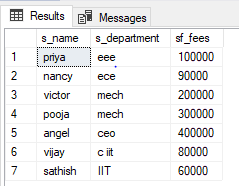
*Example :*

select s.s\_name,s.s\_department,sf.sf\_fees

from students as s

inner join students\_fee as sf

on s.s\_id=sf.sf\_id;



## UNION :

The UNION operator is used to combine the result-set of two or more SELECT statements.

Syntax :

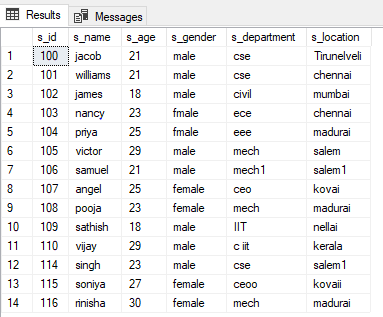
SELECT *column\_name(s)* FROM *table1*  
UNION  
SELECT *column\_name(s)* FROM *table2*;

Example :

select \* from student

union

select \* from students;



select \* from student

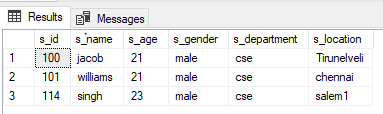
where s\_department='cse'

union

select \* from students

where s\_department='cse'

order by s\_department;



**UNION ALL :**

Union all operator gives all the rows from both the tables including the duplicates.

Syntax :

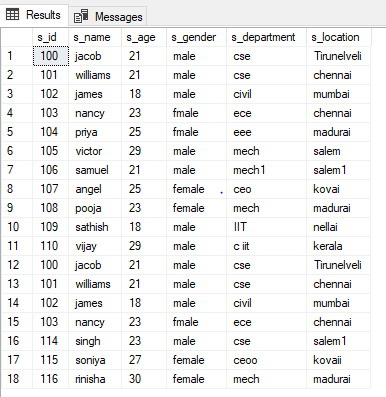
SELECT column\_name(s) FROM table1  
UNION ALL  
SELECT column\_name(s) FROM table2;

Example :

select \* from student

union all

select \* from students;



**Except :**

Except operator combines two select statement and returns unique records from the

Left query which are not part of the right query.

Syntax :

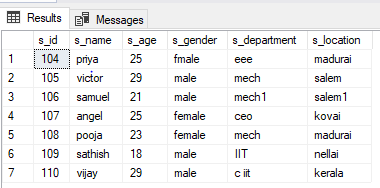
SELECT column\_name(s) FROM table1  
Except  
SELECT column\_name(s) FROM table2;

Example :

select \* from student

except

select \* from students;



**INTERSECT :**

Intersect operator helps to combine two select statement and returns

The records with are common to both the select statement.

Syntax :

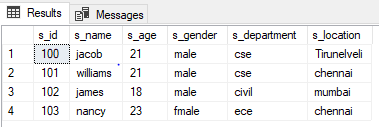
SELECT column\_name(s) FROM table1  
Intersect  
SELECT column\_name(s) FROM table2;

Example :

select \* from student

intersect

select \* from students;



**DAY 7 :**

**CASE :**

Case statement helps in multi way decision making.

Syntax :

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

Example:

select e\_name,e\_salary,e\_level=

case

when e\_salary>=100000 then 'senior employee'

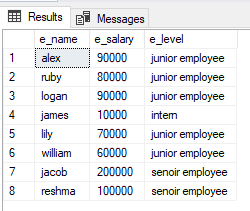
when e\_salary>=50000 then 'junior employee'

else 'intern'

end

from employee

go;



**DATE :**

**SQL Server** comes with the following data types for storing a date or a date/time value in the database:

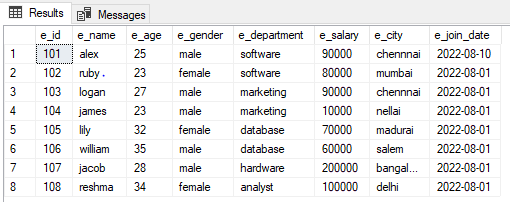
* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* SMALLDATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: a unique number

Example :

alter table employee add e\_join\_date date;

update employee set e\_join\_date='2022-08-10'

where e\_name='alex';



**SUB QUERY :**

A subquery is a query that appears inside another query statement. Subqueries are also referred to as sub- SELECT s or nested SELECT s. The full SELECT syntax is valid in subqueries.

Syntax :

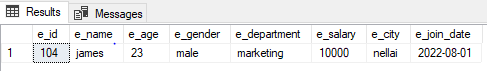
SELECT ALL column\_name(s)  
FROM table\_name  
WHERE condition(select column\_name from table name);

Example :

select \* from employee

where e\_salary=(select min(e\_salary) from

employee);



**RANK :**

If you have three items at rank 3, then the next rank listed would be rank 5. It will skip the next available ranking value.

Example :

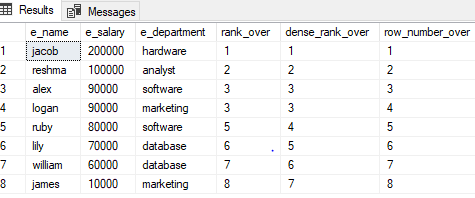
select e\_name,e\_salary,e\_department,

rank() over (order by e\_salary desc) as rank\_over,

dense\_rank() over (order by e\_salary desc) as dense\_rank\_over,

row\_number() over (order by e\_salary desc)as row\_number\_over

from employee;



**DAY 10**

**SUB QUERY :**

A subquery is a query that appears inside another query statement. Subqueries are also referred to as sub- SELECT s or nested SELECT s. The full SELECT syntax is valid in subqueries.

Syntax :

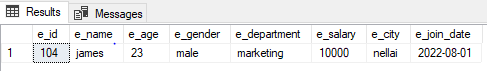
SELECT ALL column\_name(s)  
FROM table\_name  
WHERE condition(select column\_name from table name);

Example :

select \* from employee

where e\_salary=(select min(e\_salary) from

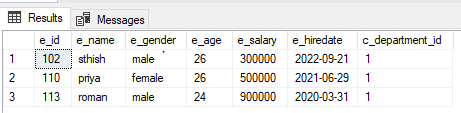
employee);



select \* from employee

where c\_department\_id=(select c\_department\_id from company

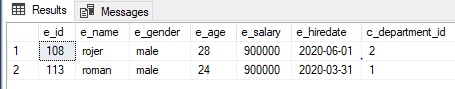
where c\_department='software');



select \* from employee

where e\_salary=(select max(e\_salary)

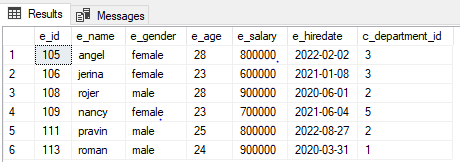
from employee);



select \* from employee

where e\_id in (select e\_id from employee

where e\_salary>500000);

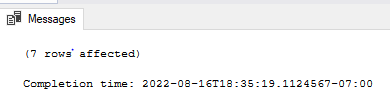


update employee

set e\_salary=e\_salary \* 0.25

where e\_age in (select e\_age from employee

where e\_age>=26);



select \* from employee;

