Structured Query Language (SQL)

5.4 Not Conditions

5.5 Like Conditions

5.6 Between Condition

6. Function

6.1 Min()

6.2 Max()

6.3 Count()

6.4 Sum()

6.5 Avg()

6.6 String LTRIM()

6.7 String Lower()

6.8 String Upper()

6.9 Reverse()

6.10 String Substring()

7. Order By

7.1 Ascending Order

7.2 Descending Order

8. Top Clause

9. Group By

10. Having Clause

11. Update Statement

12. Delete Statement

13. Truncate Statement

14. Join Statement

1. Create, Use and Drop the Database

a. Create a Database

Syntax:

create database [databasename];

Example:

```
create database company;
```

Here we create a database with the name of the company

b. Use a Database

Syntax:

Use [databasename];

Example:

```
use company;
```

Here we use a company database.

c. Drop a Database

Syntax:

Drop database [databasename];

Example:

```
drop database company;
```

Here we delete a company database.

2. Constraints in SQL Database

Constraints are used to specify rules for data in table

a. Not Null Constraints

Not Null constraints ensure that a column cannot have a Null value

Example:

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support

For example, if we specify no null constraints on *e_salary* columns then it's meant no null value should be in that column.

b. Default Constraints

Default constraints set a default value for cols when no value is specified

Example:

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	123000	25	Male	Operations
2	Bob	75000	25	Male	Support

For example, if we specify default value 25 in *e_age* columns then it's meant all the values in that column will be 25

c. Unique Constraints

Unique constraints ensure that all values in cols are different.

Example:

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support

For example, if we specify unique constraints in the *e_name* column then it's meant all the values in that column will be unique

d. Primary Key Constraints

- Primary key constraint uniquely identifiers each record in a table.
- It is also a combination of *not null + unique*
- **Note:** in table no more than one primary key

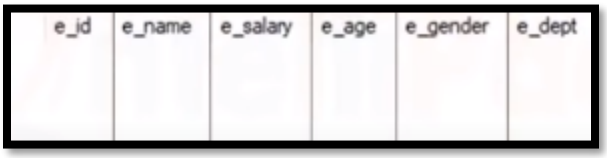
Example:

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support

For example, if we assign a primary key on the e_id column then it means all the values in that column are unique and not null.

3. Create and Store Record in Table

a. Create a Table



e_id	e_name	e_salary	e_age	e_gender	e_dept

Syntax:

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
columnN datatype,
PRIMARY KEY(column_x) );
```

```
create table company(
e_id int not null,
e_name varchar(20),
e_salary int,
e_age int,
e_gender varchar(20),
e_dept varchar(20),

primary key(e_id),
);
```

b. Stored Record in a Table

Syntax:

```
INSERT INTO table_name
VALUES (value1, value2,value3,...valueN);
```

```
insert company values(
1, 'Amir', 7500, 21, 'Male', 'Data Scientist'
);

insert company values(
2, 'Mathew', 7200, 24, 'Male', 'Web Developer'
);

insert company values(
3, 'Szymon', 6900, 25, 'Male', 'Mathematician'
);

insert company values(
4, 'Kurstian', 8000, 22, 'Male', 'Programmer'
);
```

4. Select and Distinct Statement Syntax

Syntax:

```
SELECT column1, column2, columnN

FROM table_name;
```

a. Select a Single Column from Table

Here we select the **name** column from the company table that we create and store the value inside that.

```
select e_name from company;
```

b. Select a Multiple Column from Table

Here we select the **name, age, department** columns from the company table that we create and store the value inside that.

```
select e_name, e_age, e_salary from company;
```

	e_name	e_age	e_salary
1	Amir	21	7500
2	Mathew	24	7200
3	Szymon	24	6900
4	Kurstian	22	8000
5	Kalina	19	5000
6	Ola	20	7100
7	Stanislaw	26	7700

c. Select Entire Columns from Table

Here we select the **all** columns from the company table that we create and store the value inside that.

```
select * from company;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	24	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	5	Kalina	5000	19	Female	Databse Expert
6	6	Ola	7100	20	Female	Data Analysis
7	7	Stanislaw	7700	26	Male	Data Engineer

d. Select Distinct Syntax

Select Distinct (unique value) is used to select only distinct values from our column.
For example, in the gender column, we have multiple male and female records, and we want only unique

```
select distinct e_gender from company;
```

	e_gender
1	Female
2	Male

5. Where Clause

Let's filter records with where clause

Where Clause is used to extract records that satisfy a condition

For example: age >60 , Occupation = Data Scientist

a. Simple Conditions

Syntax:

```
SELECT column1, column2, columnN
FROM table_name WHERE [condition]
```

Example:

1. Age >= 22

```
select * from company where e_age >= 22;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	2	Mathew	7200	24	Male	Web Developer
2	3	Szymon	6900	25	Male	Mathematician
3	4	Kurstian	8000	22	Male	Programmer
4	7	Stanislaw	7700	26	Male	Data Engineer

2. Gender = male

```
select * from company where e_gender  = 'male'
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	7	Stanislaw	7700	26	Male	Data Engineer

3. salary >= 7200

```
select * from company where e_salary >= 7200
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	4	Kurstian	8000	22	Male	Programmer
4	7	Stanislaw	7700	26	Male	Data Engineer

b. And Conditions

AND operator displays records if all the conditions separated by AND are TRUE

For example: age >60 **AND** Occupation = Data Scientist

Let's impose multiple conditions with AND

Syntax:

```
SELECT column1, column2, columnN
FROM table_name WHERE [condition1]
AND [condition2]...AND [conditionN];
```

1. Age >= 22 AND salary >= 7200

```
select * from company where e_age >= 22 and e_salary >= 7200;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	2	Mathew	7200	24	Male	Web Developer
2	4	Kurstian	8000	22	Male	Programmer
3	7	Stanislaw	7700	26	Male	Data Engineer

2. Age <= 20 AND salary >= 5000 AND Gender = female

```
select * from company where e_gender = 'female' AND e_age <= 20 AND e_salary >= 5000;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	5	Kalina	5000	19	Female	Databse Expert
2	6	Ola	7100	20	Female	Data Analysis

c. OR Conditions

OR operator displays records if any of the conditions separated by OR are TRUE

For example: age >60 **OR** Occupation = Data Scientist

Let's impose multiple conditions with OR

Syntax:

```
SELECT column1, column2, columnN
FROM table_name WHERE [condition1]
OR [condition2]...OR [conditionN];
```

1. e_dept = 'Data Scientist' or e_dept = 'Data Analysis'

```
select * from company where e_dept = 'Data Scientist' or e_dept = 'Data Analysis';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	6	Ola	7100	20	Female	Data Analysis

d. Not Conditions

Nor operator displays records if the condition is **NOT** True

For example: Occupation = Data Scientist >> then display except Data Scientist

Let's impose records where the condition is NOT TRUE

Syntax:

```
SELECT column1, column2, columnN
FROM table_name WHERE NOT
[condition];
```

2. Not e_dept = 'Data Scientist' (show all except this)

```
select * from company where not e_dept = 'Data Scientist';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	2	Mathew	7200	24	Male	Web Developer
2	3	Szymon	6900	25	Male	Mathematician
3	4	Kurstian	8000	22	Male	Programmer
4	5	Kalina	5000	19	Female	Databse Expert
5	6	Ola	7100	20	Female	Data Analysis
6	7	Stanislaw	7700	26	Male	Data Engineer

e. Like Conditions

Like operator is used to extract records where a particular pattern is present

For example: pattern John will display Johnathon and Johny etc

Wild Card Character

Syntax:

```
SELECT col_list FROM table_name WHERE
column_N LIKE '_XXXX%';
```

1. Extract name with K letter

```
select * from company where e_name like 'K%';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	4	Kurstian	8000	22	Male	Programmer
2	5	Kalina	5000	19	Female	Databse Expert

2. Extract department only who have data in pattern

```
select * from company where e_dept like 'Data%';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	5	Kalina	5000	19	Female	Databse Expert
3	6	Ola	7100	20	Female	Data Analysis
4	7	Stanislaw	7700	26	Male	Data Engineer

3. Select all employee who age in 20

```
select * from company where e_age LIKE '2_';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	6	Ola	7100	20	Female	Data Analysis
6	7	Stanislaw	7700	26	Male	Data Engineer

f. Between Conditions

Between operator is used to select values within in given age

For example, salary range between 5000 and 6000

Note: here 5000 and 6000 are inclusive

Syntax:

```
SELECT col_list FROM table_name WHERE
column_N BETWEEN val1 AND val2;
```

1. Age between 19 to 22

```
select * from company where e_age between 18 and 23;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	4	Kurstian	8000	22	Male	Programmer
3	5	Kalina	5000	19	Female	Databse Expert
4	6	Ola	7100	20	Female	Data Analysis

6. Functions

Number Function and String Function

1. Min()

Give the smallest value

Syntax:

```
SELECT MIN(col_name) FROM
table_name;
```

Example

```
select MIN(e_salary) from company;
```

	(No column name)
1	5000

2. Max()

Give the largest value

Syntax:

```
SELECT MAX(col_name) FROM
table_name;
```

Example

```
select MAX(e_age) from company;
```

	(No column name)
1	26

3. Count()

Return the number of rows that match specific criteria.

Syntax:

```
SELECT COUNT(*) FROM table_name
WHERE condition;
```

Example

```
select count(*) from company where e_gender = 'male';
```

	(No column name)
1	5

4. sum()

return the sum of selected numeric column

Syntax:

```
SELECT SUM(col_name) FROM
table_name;
```

Example

```
select sum(e_salary) from company;
```

	(No column name)
1	49400

5. Avg()

return the Average of selected numeric column

Syntax:

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

Example

```
select avg(e_age) from company;
```

	(No column name)
1	22

6. String LTRIM()

Removes blank on the left side of the character expression

Example

```
select LTRIM('     Amir');
```

	(No column name)
1	Amir

Before

	(No column name)
1	Amir

After

7. String LOWER()

Convert all characters to lower case letters

Syntax

```
SELECT LOWER(COL NAME) FROM
TABLE_NAME;
```

Example

```
select lower(e_dept) from company;
```

	(No column name)
1	data scientist
2	web developer
3	mathematician
4	programmer
5	databse expert
6	data analysis
7	data engineer

8. lower ()

Convert all characters to lower case letters

Syntax

```
SELECT LOWER(COL NAME) FROM
TABLE_NAME;
```

Example

```
select lower(e_dept) from company;
```

	(No column name)
1	data scientist
2	web developer
3	mathematician
4	programmer
5	databse expert
6	data analysis
7	data engineer

9. Upper ()

Convert all characters to upper case letters

Syntax

```
SELECT UPPER(COL NAME) FROM
TABLE_NAME;
```

Example

```
select upper(e_dept) from company;
```

	(No column name)
1	DATA SCIENTIST
2	WEB DEVELOPER
3	MATHEMATICIAN
4	PROGRAMMER
5	DATABSE EXPERT
6	DATA ANALYSIS
7	DATA ENGINEER

10.Reverse ()

Reverse all case letter into reverse

Syntax

```
SELECT REVERSE(COL NAME) FROM
TABLE_NAME;
```

Example

```
select REVERSE(e_name) from company;
```

	(No column name)
1	rimA
2	wehtaM
3	nomyzS
4	naitsruK
5	anilaK
6	alO
7	walsinatS

12. String SUBSTRING ()

Given a substring from the original strings

Syntax

```
SELECT SUBSTRING(COL NAME, VALUE WHERE THE STRING START, END) FROM
TABLE_NAME;
```

Example:

```
select SUBSTRING(e_dept, 1, 4) from company;
```

	(No column name)
1	Data
2	Web
3	Math
4	Prog
5	Data
6	Data
7	Data

7. Order By

Order by is used to sort the data in ascending or descending order.

```
SELECT column_list FROM table_name
ORDER BY col1, col2,.....ASC|DESC
```

Example of Ascending order (By Default)

```
select * from company order by e_age;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	5	Kalina	5000	19	Female	Databse Expert
2	6	Ola	7100	20	Female	Data Analysis
3	1	Amir	7500	21	Male	Data Scientist
4	4	Kurstian	8000	22	Male	Programmer
5	2	Mathew	7200	24	Male	Web Developer
6	3	Szymon	6900	25	Male	Mathematician
7	7	Stanislaw	7700	26	Male	Data Engineer

Example of Descending order

```
select * from company order by e_salary desc;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	4	Kurstian	8000	22	Male	Programmer
2	7	Stanislaw	7700	26	Male	Data Engineer
3	1	Amir	7500	21	Male	Data Scientist
4	2	Mathew	7200	24	Male	Web Developer
5	6	Ola	7100	20	Female	Data Analysis
6	3	Szymon	6900	25	Male	Mathematician
7	5	Kalina	5000	19	Female	Databse Expert

8. Top Clause

Top is used to fetch the TOP N records

```
SELECT TOP x column_list FROM
table_name;
```

Example

```
select top 4 * from company;
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	21	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer

9. Group By

Group by is used to get aggregate result with respect to a group

```
SELECT column_list
FROM table_name
GROUP BY colname(s);
```

Example

```
select sum(e_salary), avg(e_age), e_gender from company group by e_gender;
```

	(No column name)	(No column name)	e_gender
1	12100	19	Female
2	37300	23	Male

10. Having Clause

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

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Example:

```
select e_gender, avg(e_salary) as e_salary
from company
Group by e_gender
having avg(e_salary) > 6500
```

	e_gender	e_salary
1	Male	7460

11. Update Statement

The update is used to modify the existing records in a table

```
UPDATE table_name
SET col1=val1, col2=val2.....
[WHERE condition];
```

Example: (update the age of amir)

```
update company set e_age = 23 where e_name = 'Amir';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	23	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	5	Kalina	5000	19	Female	Databse Expert
6	6	Ola	7100	20	Female	Data Analysis
7	7	Stanislaw	7700	26	Male	Data Engineer

12. Delete Statement

The Delete statement is used to delete existing records in the table

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

Example: (delete the row of Nader)

```
delete from employee
where e_name='Nader';
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	2393	21	Male	Analytic
2	2	Ansar	1763	24	Male	Operation
3	4	Waqas	3452	29	Male	Expert
4	5	Sana	7533	20	Female	Scientist
5	6	Fahad	4321	27	Male	Student

13. Truncate Statement

Truncate statement deletes all the data inside the table

```
TRUNCATE TABLE table_name ;
```

Example (delete all records of employee)

```
truncate table employee;
```

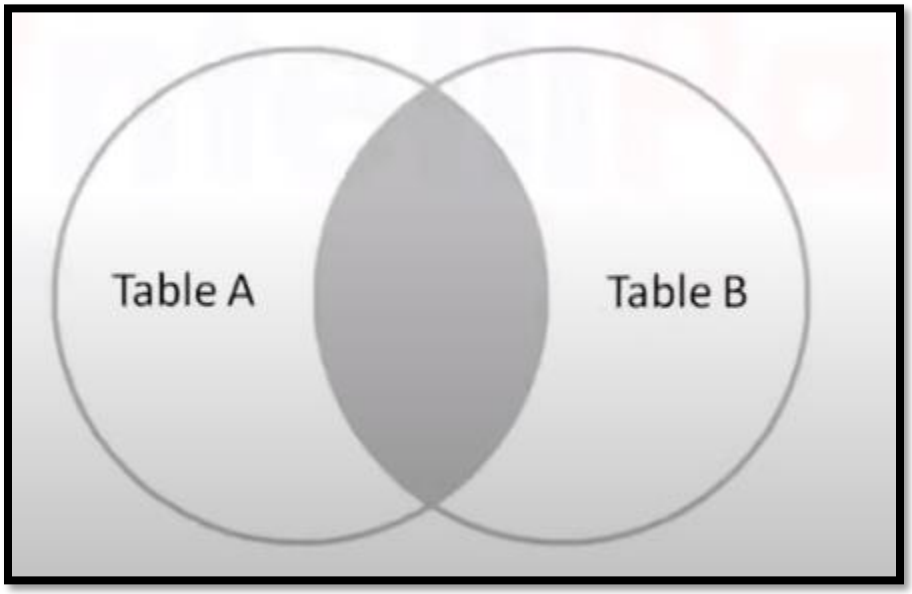
e_id	e_name	e_salary	e_age	e_gender	e_dept

14. Join

By using join we can join multiple tables on the basis of different types which we will discuss now.

a. Inner Join

Inner Join returns records that have matching values in both tables. It is also known as simple join.




```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column_x = table2.column_y;
```

Example

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	23	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	5	Kalina	5000	19	Female	Databse Expert
6	6	Ola	7100	20	Female	Data Analysis
7	7	Stanislaw	7700	26	Male	Data Engineer

	d_id	d_name	d_location
1	1	Data Scientist	Warsaw
2	1	Data Scientist	Warsaw
3	2	12313	katowice
4	3	Mathematician	Gdnask
5	4	1ss1	Lodz
6	5	Programmer	krakow
7	6	Data Analysis	poznan
8	7	Data Engineer	Wloch

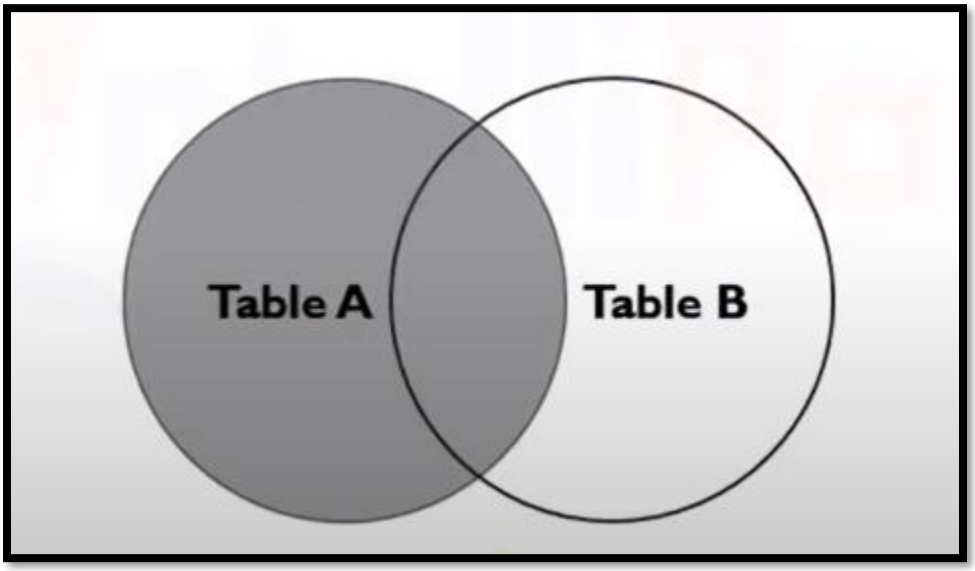
```
select company.e_name, company.e_dept, department.d_name, department.d_location
from company
inner join department
on company.e_dept = department.d_name
```

common only

	e_name	e_dept	d_name	d_location
1	Amir	Data Scientist	Data Scientist	Warsaw
2	Amir	Data Scientist	Data Scientist	Warsaw
3	Szymon	Mathematician	Mathematician	Gdnask
4	Kurstian	Programmer	Programmer	krakow
5	Ola	Data Analysis	Data Analysis	poznan
6	Stanislaw	Data Engineer	Data Engineer	Wloch

b. Left Join

Left Join returns records all the records from the left table, and the matched records from the right table




```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.column_x = table2.column_y;
```

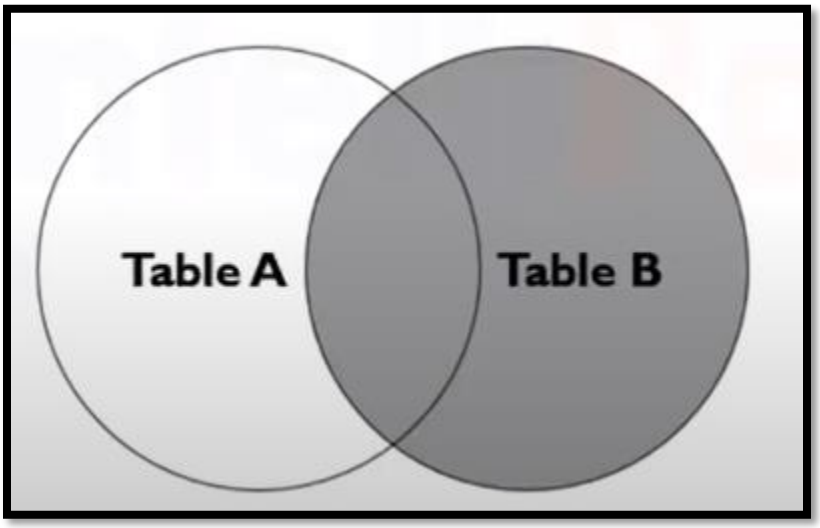
Example

```
select company.e_name, company.e_dept, department.d_name, department.d_location
from company
left join department
on company.e_dept = department.d_name
```

	e_name	e_dept	d_name	d_location
1	Amir	Data Scientist	Data Scientist	Warsaw
2	Amir	Data Scientist	Data Scientist	Warsaw
3	Mathew	Web Developer	NULL	NULL
4	Szymon	Mathematician	Mathematician	Gdnask
5	Kurstian	Programmer	Programmer	krakow
6	Kalina	Databse Expert	NULL	NULL
7	Ola	Data Analysis	Data Analysis	poznan
8	Stanislaw	Data Engineer	Data Engineer	Wloch

c. Right Join

Right join return s all the records from the right table, and the matched records from the left table.




```
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.column_x = table2.column_y;
```

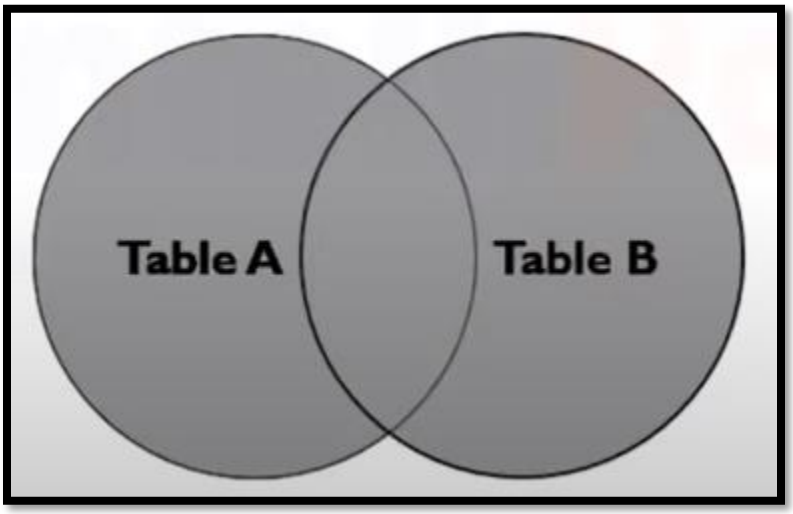
Example

```
select company.e_name, company.e_dept, department.d_name, department.d_location
from company
right join department
on company.e_dept = department.d_name
```

	e_name	e_dept	d_name	d_location
1	Amir	Data Scientist	Data Scientist	Warsaw
2	Amir	Data Scientist	Data Scientist	Warsaw
3	NULL	NULL	12313	katowice
4	Szymon	Mathematician	Mathematician	Gdnask
5	NULL	NULL	1ss1	Lodz
6	Kurstian	Programmer	Programmer	krakow
7	Ola	Data Analysis	Data Analysis	poznan
8	Stanislaw	Data Engineer	Data Engineer	Wloch

d. Full Join

It returns all rows from the left table and the right table with Null values in place where join condition is not met



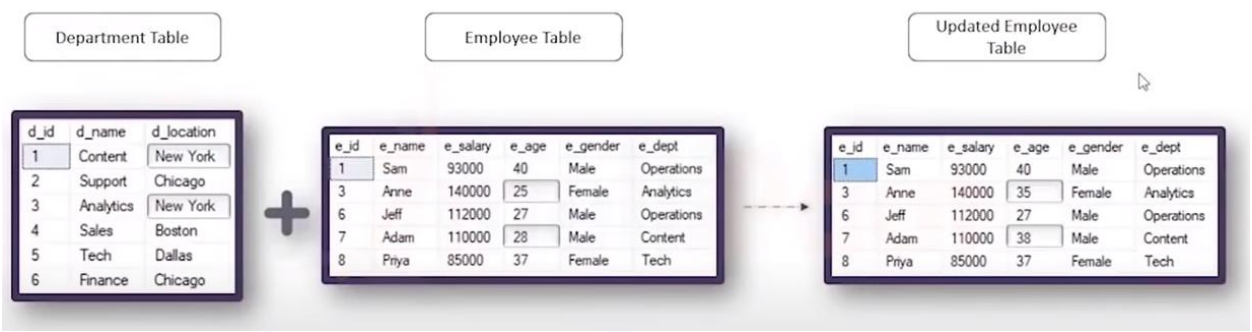

```
SELECT columns
FROM table1
FULL JOIN table2
ON table1.column_x = table2.column_y;
```

Example

```
select company.e_name, company.e_dept, department.d_name, department.d_location
from company
full join  department
on company.e_dept= department.d_name
```

	e_name	e_dept	d_name	d_location
1	Amir	Data Scientist	Data Scientist	Warsaw
2	Amir	Data Scientist	Data Scientist	Warsaw
3	Mathew	Web Developer	NULL	NULL
4	Szymon	Mathematician	Mathematician	Gdnask
5	Kurstian	Programmer	Programmer	krakow
6	Kalina	Databse Expert	NULL	NULL
7	Ola	Data Analysis	Data Analysis	poznan
8	Stanislaw	Data Engineer	Data Engineer	Wloch
9	NULL	NULL	12313	katowice
10	NULL	NULL	1ss1	Lodz

e. Update using join



Example: update the company age where department location is warsaw

```
update company
set e_age = e_Age + 2
from company
join department on company.e_dept = department.d_name
where d_name='Data Scientist'
```

15. Union and Intersection operator

We will use these two tables for union and intersection operation

	s_id	s_name	s_age
1	4	Amir	23
2	2	Raza	24
3	3	Ali	25

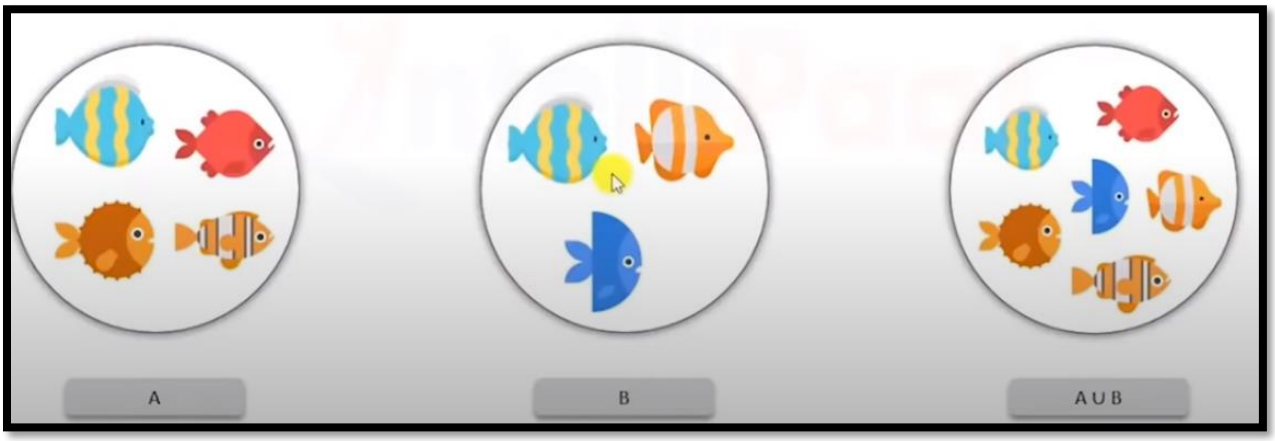
<-1

	s_id	s_name	s_age
1	6	usman	20
2	3	Ali	25
3	4	Amir	23

<-2

a. Union operator

Union operator is used to combine the result-set of two or more SELECT statement




```
SELECT column_list FROM table1
Union
SELECT column_list FROM table2
```

Example:

```
select * from data1
union
select * from data2
```

	s_id	s_name	s_age
1	2	Raza	24
2	3	Ali	25
3	4	Amir	23
4	6	usman	20

b. Union All operator

Union operator gives all the rows from both the tables including the duplicate

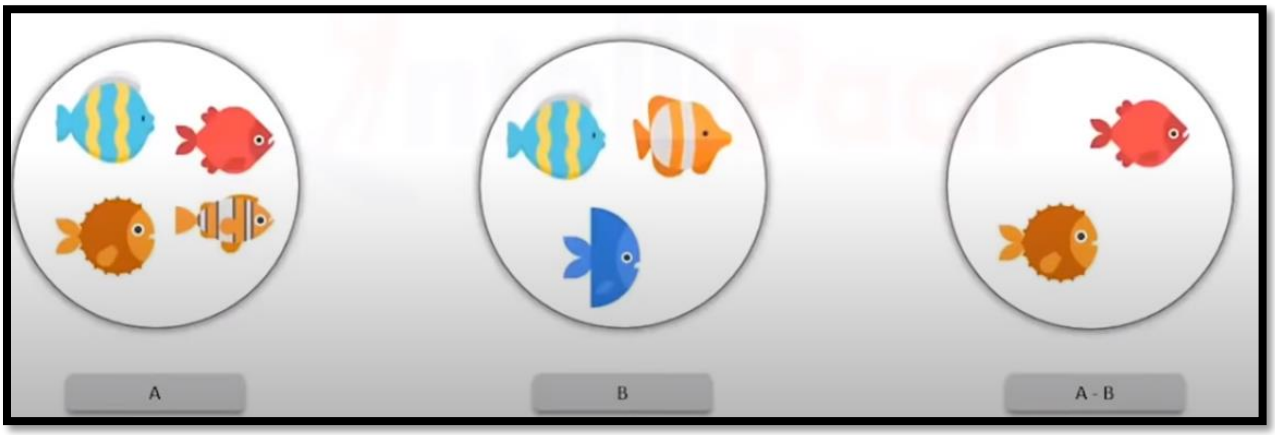
Example:

```
select * from data1
union all
select * from data2
```

	s_id	s_name	s_age
1	4	Amir	23
2	2	Raza	24
3	3	Ali	25
4	6	usman	20
5	3	Ali	25
6	4	Amir	23

c. Except Operator

Except Operator combines two select statements and returns unique records from the left query which is not part of the right query.




```
SELECT column_list FROM table1
EXCEPT
SELECT column_list FROM table2
```

Example:

```
select * from data1
except
select * from data2
```

	s_id	s_name	s_age
1	2	Raza	24

d. Intersect Operator

Intersect Operator helps to combine two select statements and returns the records which are common to both the select statements.

```
SELECT column_list FROM table1
INTERSECT
SELECT column_list FROM table2
```

Example:

```
select * from data1
intersect
select * from data2
```

	s_id	s_name	s_age
1	3	Ali	25
2	4	Amir	23

16. Create View & Drop View

a. Create View

View is a virtual table based on the result of an sql statement.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example:

```
create view male_student as
select * from company
where e_gender = 'Male';
```

```
select * from male_student
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	31	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	7	Stanislaw	7700	26	Male	Data Engineer

b. Drop View

```
DROP VIEW view_name;
```

Example

```
drop view male_student;
```

We drop male student view that we created

15. Alter Table

Alter table statement is used to add, delete or modify columns in a table

a. Add Column

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

Example

```
alter table company
add dob int;

select * from company
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept	dob
1	1	Amir	7500	31	Male	Data Scientist	NULL
2	2	Mathew	7200	24	Male	Web Developer	NULL
3	3	Szymon	6900	25	Male	Mathematician	NULL
4	4	Kurstian	8000	22	Male	Programmer	NULL
5	5	Kalina	5000	19	Female	Databse Expert	NULL
6	6	Ola	7100	20	Female	Data Analysis	NULL
7	7	Stanislaw	7700	26	Male	Data Engineer	NULL

b. Drop Column

```
alter table company
drop column dob;

select * from company
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	31	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	5	Kalina	5000	19	Female	Databse Expert
6	6	Ola	7100	20	Female	Data Analysis
7	7	Stanislaw	7700	26	Male	Data Engineer

18. Merge

Merge is the combination of Insert, delete and update statements.

```
MERGE [Target] AS T
USING [Source] AS S
      ON [Join Condition]
WHEN MATCHED
      THEN [Update Statement]
WHEN NOT MATCHED BY TARGET
      THEN [Insert Statement]
WHEN NOT MATCHED BY SOURCE
      THEN [Delete Statement];
```

Example

```
Merge emplyee_target as T
using emplyee_source as S
    on T.e_id = S.e_id
when Matched
            Then update set T.e_salary = S.e_salary, T.e_age = S.e_age
when Not matched by Target
            then insert (e_id, e_name, e_salary, e_age, e_gender, e_dept)
            values(S.e_id, S.e_name, S.e_salary, S.e_age, S.e_gender, S.e_dept)
When not matched by source
            Then delete;
```

19. Table Valued Function

Table valued function returns a table

```
CREATE FUNCTION function_name(@param1 data_type, @param2 data_type...)
RETURNS table
AS
RETURN (SELECT column_list FROM table_name WHERE [condition] )
```

Example

```
create function select_gender(@gender as varchar(20))
Returns table
As
Return
(
select * from company where e_gender = @gender
)
```

```
select * from dbo.select_gender('Male')
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	1	Amir	7500	31	Male	Data Scientist
2	2	Mathew	7200	24	Male	Web Developer
3	3	Szymon	6900	25	Male	Mathematician
4	4	Kurstian	8000	22	Male	Programmer
5	7	Stanislaw	7700	26	Male	Data Engineer

20. Temporary Table

Temporary Table are created in tempDB and delete as the session is terminated.

```
CREATE TABLE #table_name(
);
```

Example

```
create table #family(
id int,
f_name varchar(20)

)


insert into #family values(
1, 'Amir'
)

insert into #family values(
2, 'Ansar'
)

select * from #family
```

	id	f_name
1	1	Amir
2	2	Ansar

21. Case Statement

Case Statement helps in multi way decision making.

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

Example

```
select * from company

select * ,grade=
case
    when e_salary<6500 then 'C'
    when e_salary<=7500 then 'B'
    else 'A'
End

from company
go
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept	grade
1	1	Amir	7500	31	Male	Data Scientist	B
2	2	Mathew	7200	24	Male	Web Developer	B
3	3	Szymon	6900	25	Male	Mathematician	B
4	4	Kurstian	8000	22	Male	Programmer	A
5	5	Kalina	5000	19	Female	Databse Expert	C
6	6	Ola	7100	20	Female	Data Analysis	B
7	7	Stanislaw	7700	26	Male	Data Engineer	A

22. IIF() Function

IIF() function is an alternative for the case statement.

```
IIF (boolean_expression, true_value, false_value )
```

Example

```
select * from company

select *, iif(e_age > 23, 'Above 23 age', 'below 23 age') as student_age from company
```

	e_id	e_name	e_salary	e_age	e_gender	e_dept	student_age
1	1	Amir	7500	31	Male	Data Scientist	Above 23 age
2	2	Mathew	7200	24	Male	Web Developer	Above 23 age
3	3	Szymon	6900	25	Male	Mathematician	Above 23 age
4	4	Kurstian	8000	22	Male	Programmer	below 23 age
5	5	Kalina	5000	19	Female	Databse Expert	below 23 age
6	6	Ola	7100	20	Female	Data Analysis	below 23 age
7	7	Stanislaw	7700	26	Male	Data Engineer	Above 23 age

23. Stored Procedure

Stored procedure is a prepared sql code which can be saved and reused.

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

```
EXEC procedure_name
```

END