

Prepared by
R. Ganesh
Senior Data Engineer



SQL



Agenda

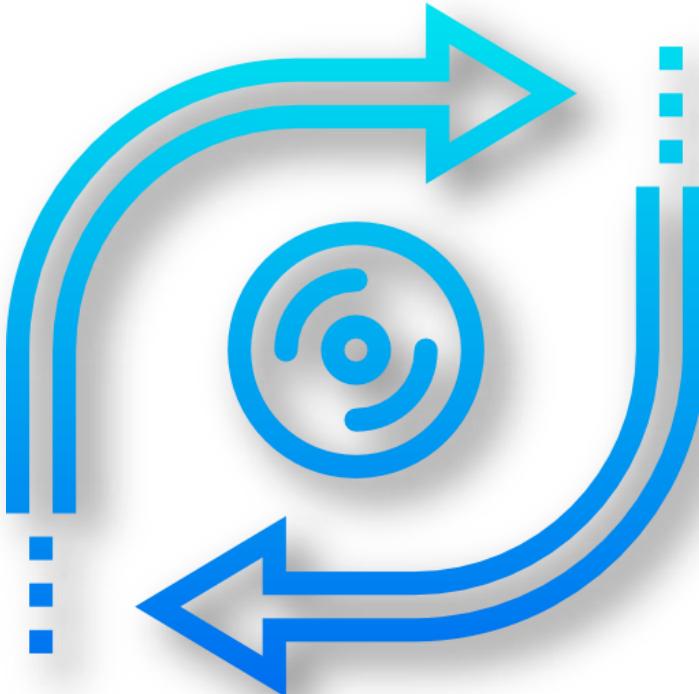
1

Alter Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Alter Table

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



Alter Table: Add Column



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

Alter Table: Drop Column



```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

Agenda

1

AND Operator

2

OR Operator

3

NOT Operator

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

AND Operator

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



Age>60

AND



Occupation="Doctor"

AND Operator: Syntax



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

OR Operator

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



Occupation="Software Engineer"

OR



Occupation="Doctor"

OR Operator: Syntax



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

NOT Operator

NOT operator displays a record if the condition is NOT TRUE.

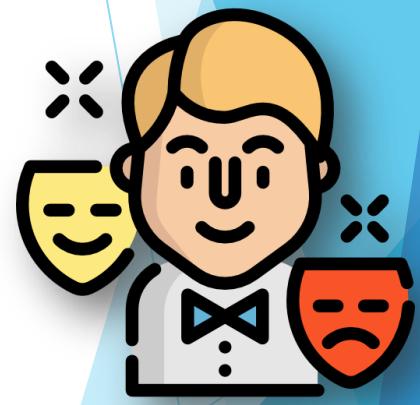
NOT



Occupation="Software Engineer"



Occupation="Doctor"



Occupation="Actor"

NOT Operator: Syntax



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

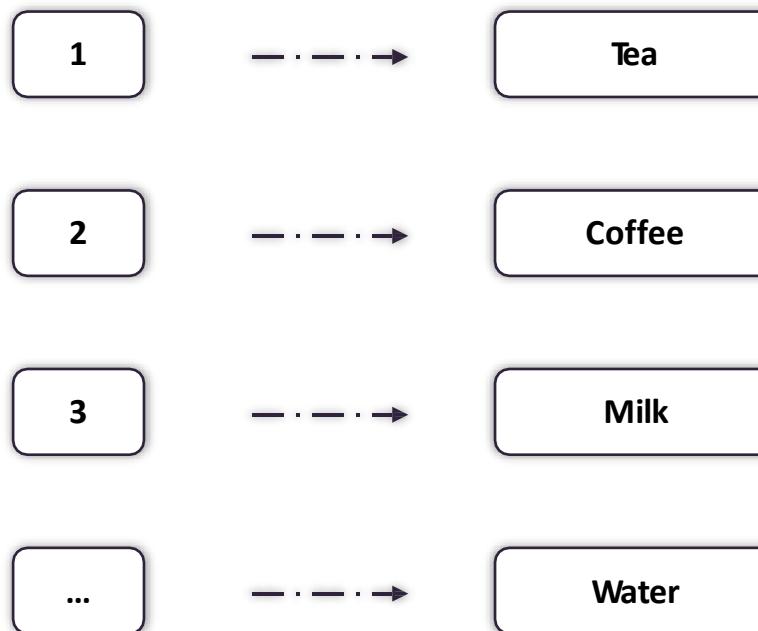
Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Case Statement

Case Statement helps in multi-way decision-making



Case Statement: Syntax



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

Agenda

1

Constraints in SQL

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Constraints in SQL

Constraints are used to specify rules for data in table

Not Null

Default

Unique

Primary Key

Not Null Constraint

Not Null constraint ensures that a column cannot have a Null value

No null values

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Default Constraint

Default constraint sets a default value for a column when no value is specified

	E_id	E_name	E_salary	E_gender	E_dept
1	1	Sam	85000	Male	Analytics
2	2	Anne	85000	Male	Analytics
3	3	Julia	85000	Female	Analytics

Default values

Default values

Unique Constraint

Unique constraint ensures that all values in a column are different

Unique values

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Primary Key Constraint

Primary Key constraint uniquely identifies each record in a table

Not Null + Unique

Primary Key

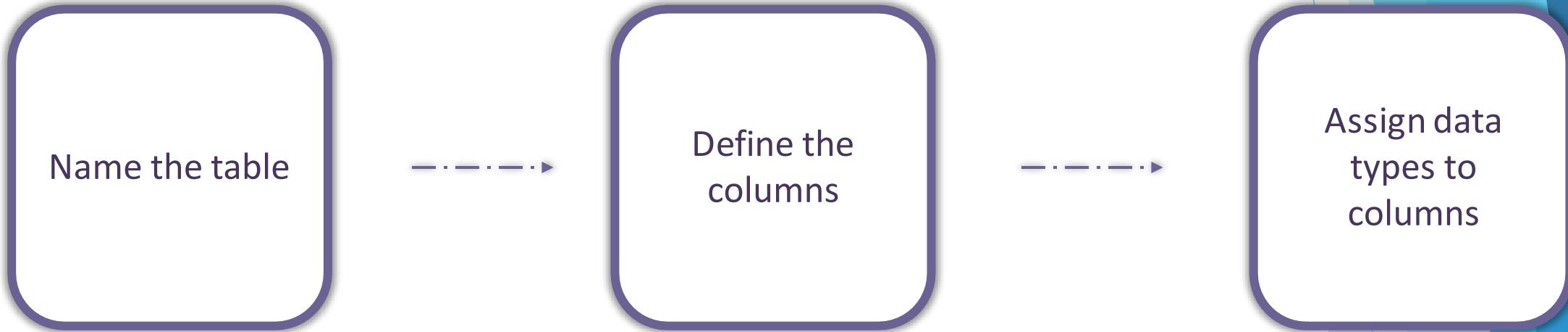
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Agenda

Create Table

e_id	e_name	e_salary	e_age	e_gender	e_dept

Create Table



Create Table Statement: Syntax



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

Agenda

Creating, Using, and Dropping a
Database in SQL



Creating a Database: Syntax



```
CREATE DATABASE databasename;
```

Using a Database: Syntax



```
USE [DatabaseName];
```

Dropping a Database: Syntax



```
DROP DATABASE databasename;
```

Agenda



Data Types in SQL

Data types define what type of data a column can hold.

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations



Integer



Character

Different Data Types in SQL

Numerical Data Types

Data Type

bigint

int

smallint

tinyint

decimal(s,d)

Range

-9223372036854775808 <-> 9223372036854775808

-2147483648 <-> 2147483647

-32768 <-> -32767

0 <-> 255

- $10^{38} + 1$ <-> $10^{38} - 1$

Character Data Types

Data Type

char(s)

varchar(s)

text

Range

255 Characters

255 Characters

65,535 Characters

Date and Time Data Types

Data Type

date

time

Year

Format

YYYY-MM-DD

HH:MM:SS

YYYY

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Delete Statement

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

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

Truncate Statement

TRUNCATE statement deletes all the data inside the table.

```
TRUNCATE TABLE table_name;
```

Agenda



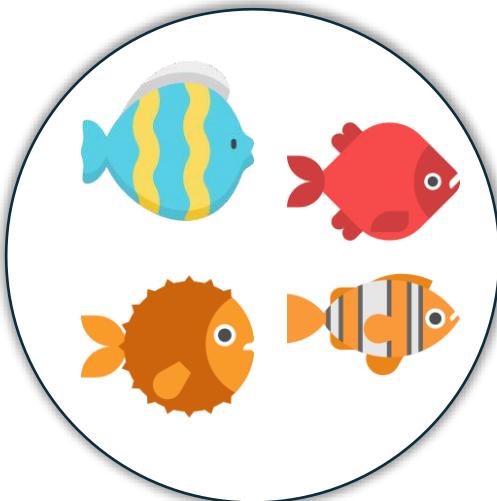
s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92



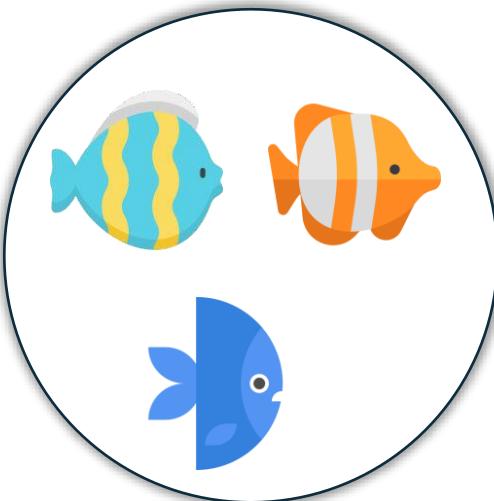
s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Except Operator

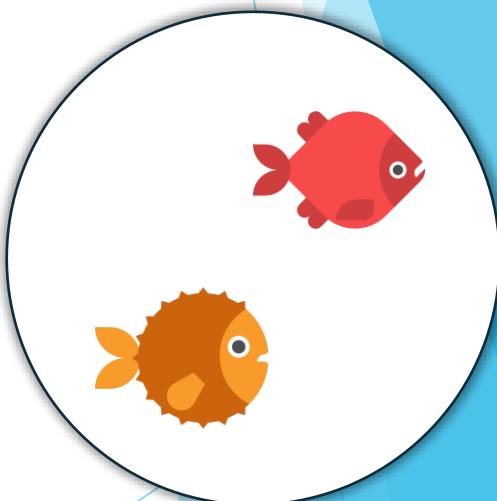
Except Operator combines two select statements and returns unique records from the left query which are not part of the right query.



A



B



A - B

Except Operator: Syntax



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

Except Operator

s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92

Student_Details1

s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Student_Details2

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

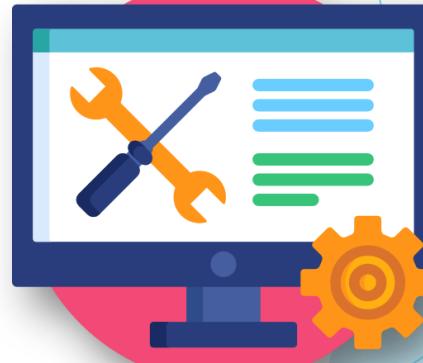
Exception Handling

An error condition during a program execution is called an exception.



Exception

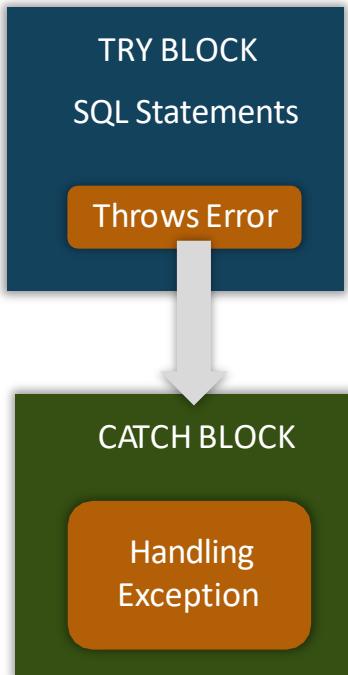
The mechanism for resolving such an exception is exception handling.



Exception Handling

Try/Catch

SQL provides try/catch blocks for exception handling.



Try/Catch: Syntax



BEGIN TRY

SQL Statements

END TRY

BEGIN CATCH

- Print Error OR
- Rollback Transaction

END CATCH

Agenda



Employee

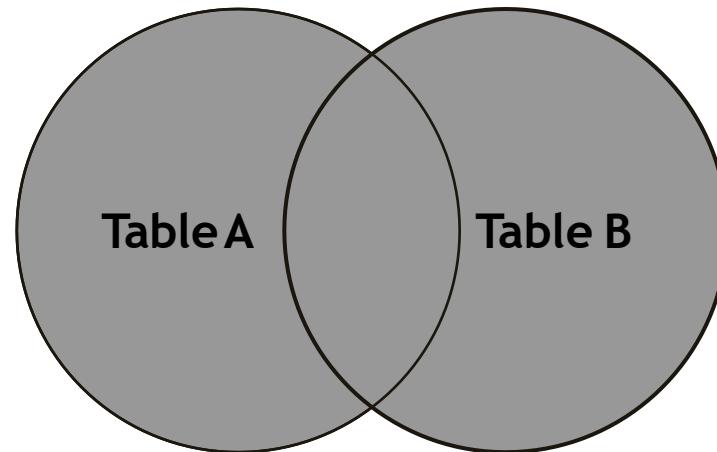
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Department

d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Full Join

It returns all rows from the LEFT table and the RIGHT table with NULL values in place where the join condition is not met.



Full Join: Syntax



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

Agenda

- 1 MIN() Function
- 2 MAX() Function
- 3 COUNT() Function
- 4 SUM() Function
- 5 AVG() Function

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

MIN() Function

MIN() function gives you the smallest value.

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

MAX() Function

MAX() function gives you the largest value.

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

COUNT() Function

COUNT() function returns the number of rows that match with a specific criteria.



FEMALE

33



MALE

67

COUNT() Function

COUNT() function returns the number of rows that match with a specific criteria.

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

SUM() Function

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

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

AVG() Function

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

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

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Group By

Group By is used to get an aggregate result with respect to a group.

Gender	Salary
Male	\$83,000
Female	\$90,000

Group By: Syntax



```
SELECT column_list  
FROM table_name  
WHERE condition  
GROUP BY colname(s)  
ORDER BY colname(s)
```

Agenda



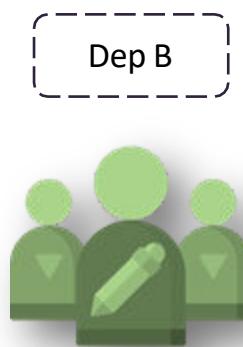
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Having Clause

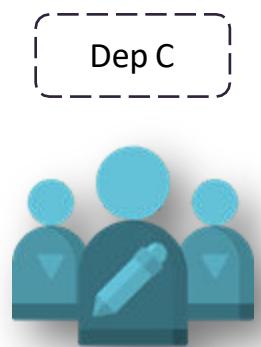
Having clause is used in combination with Group By to impose conditions on groups.



\$123,000



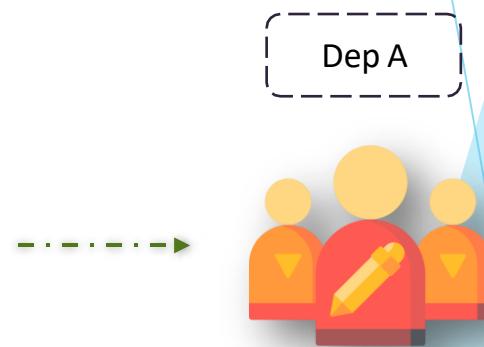
\$73,000



\$115,000



\$92,000



\$123,000



\$115,000

Having Clause: Syntax



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

Agenda



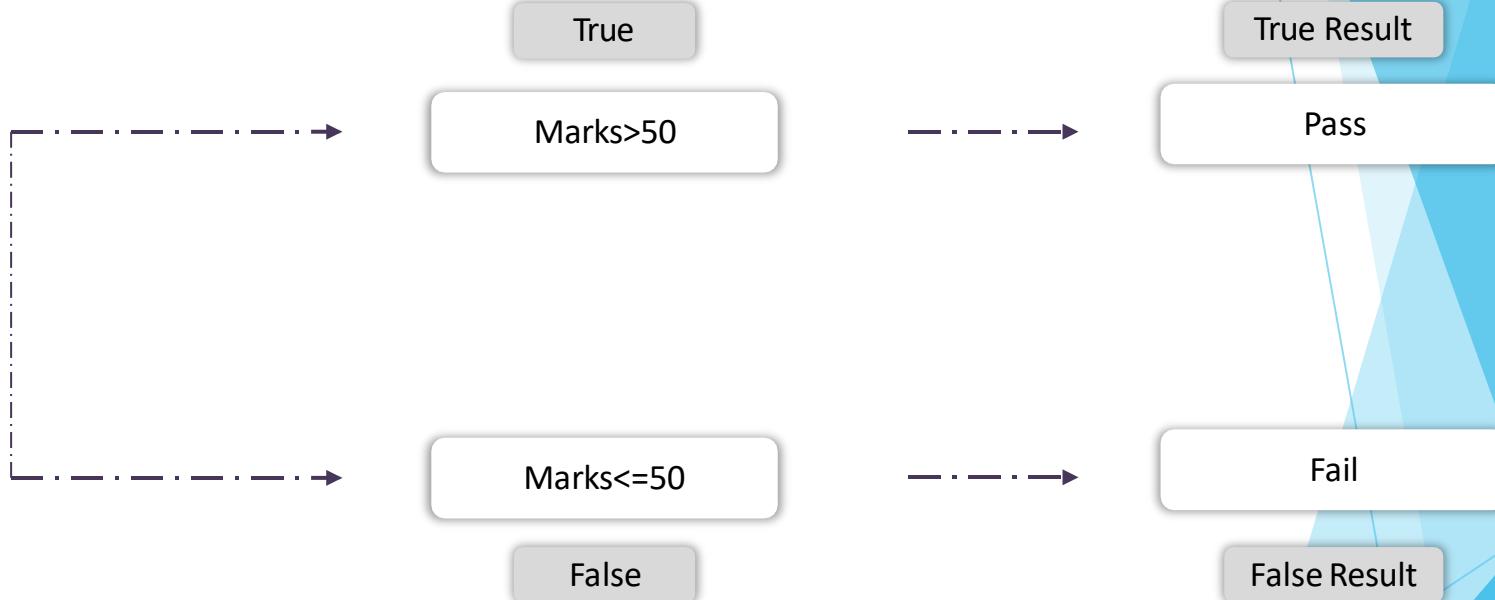
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

IIF() Function

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

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

IIF() Function



Agenda

1

InnerJoin in SQL

Employee

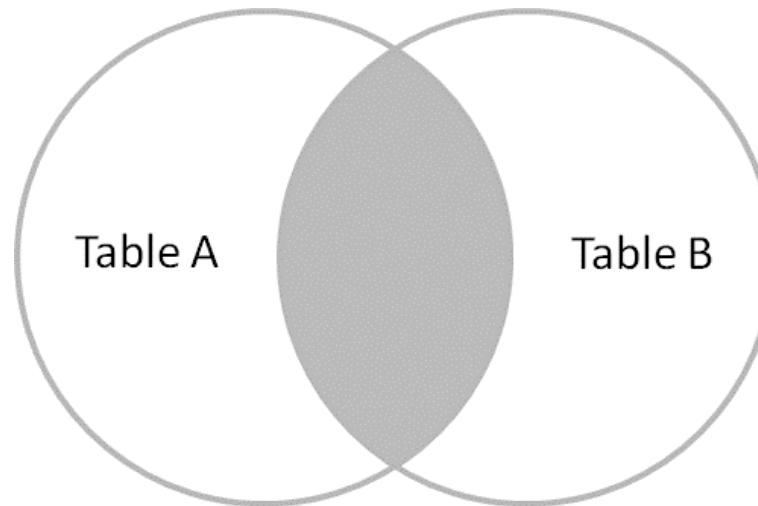
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Department

d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Inner Join

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



Inner Join: Syntax



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

Agenda

'Insert Into' Statement

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Insert Query: Syntax

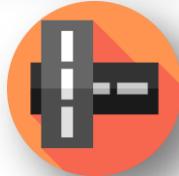


```
INSERT INTO table_name  
VALUES (value1,  
value2,value3,...valueN);
```

Agenda



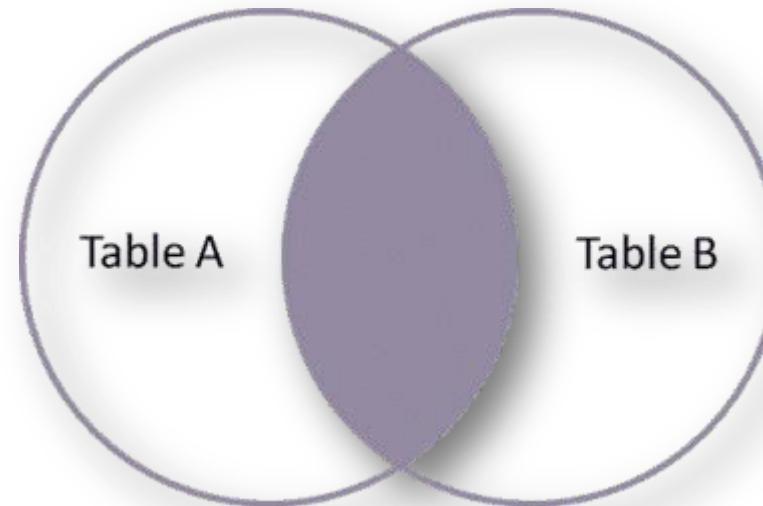
s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92



s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Intersect Operator

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



$$A \cap B$$

Intersect Operator: Syntax



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

Intersect Operator

s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92

Student_Details1

s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Student_Details2

Agenda

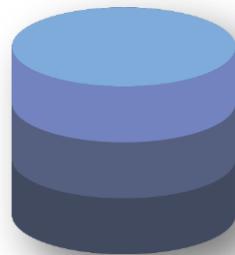
- 1 What Is a Database?
- 2 Types of Database Architecture
- 3 Introduction to SQL

What Is a Database?

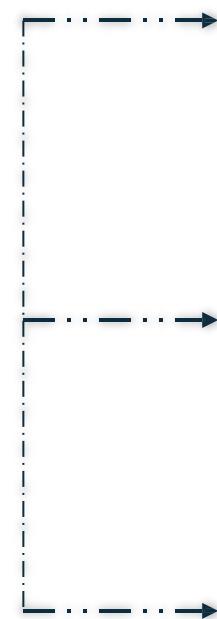
An organized collection of data stored in an electronic format



Data



Database



Access Data



Manipulate Data



Update Data

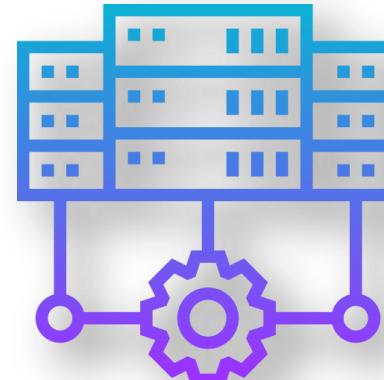


Database Management System

DBMS is a system software for creating and managing databases.

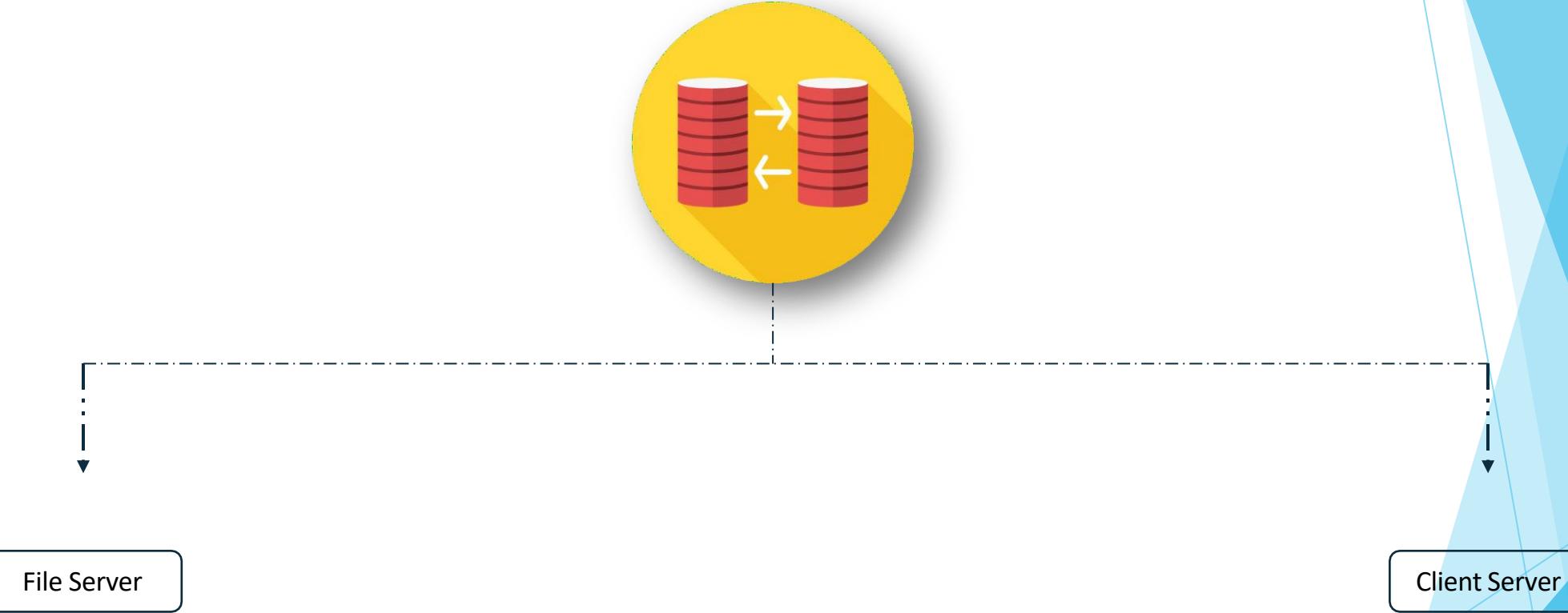


Database

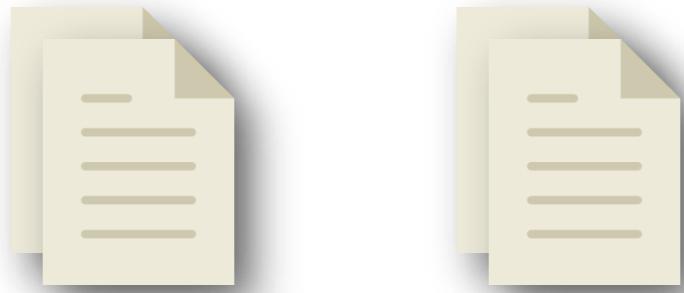


DBMS

Types of Database Architecture



File Server Architecture

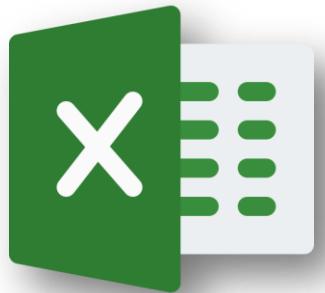


Files



Local System

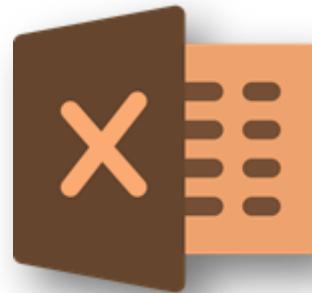
File Server Architecture



Original File



Copy

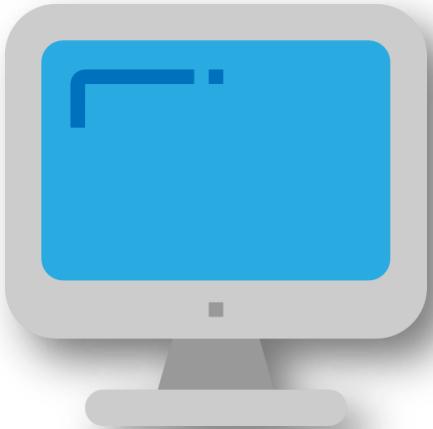


Changes in
the Original
File



Do Not
Reflect in
the Copy

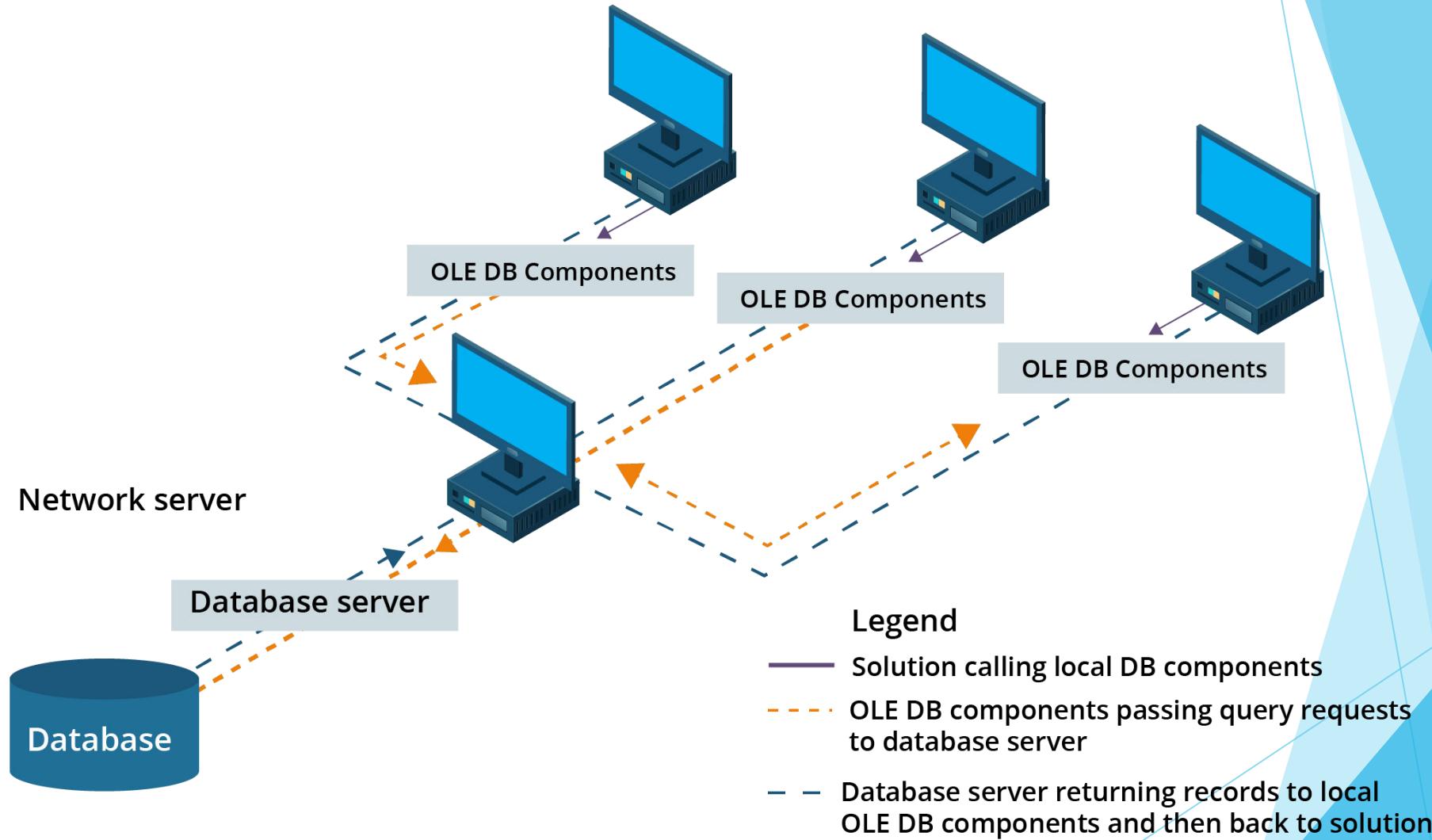
File Server Architecture



Server/Local File System



Client Server Architecture

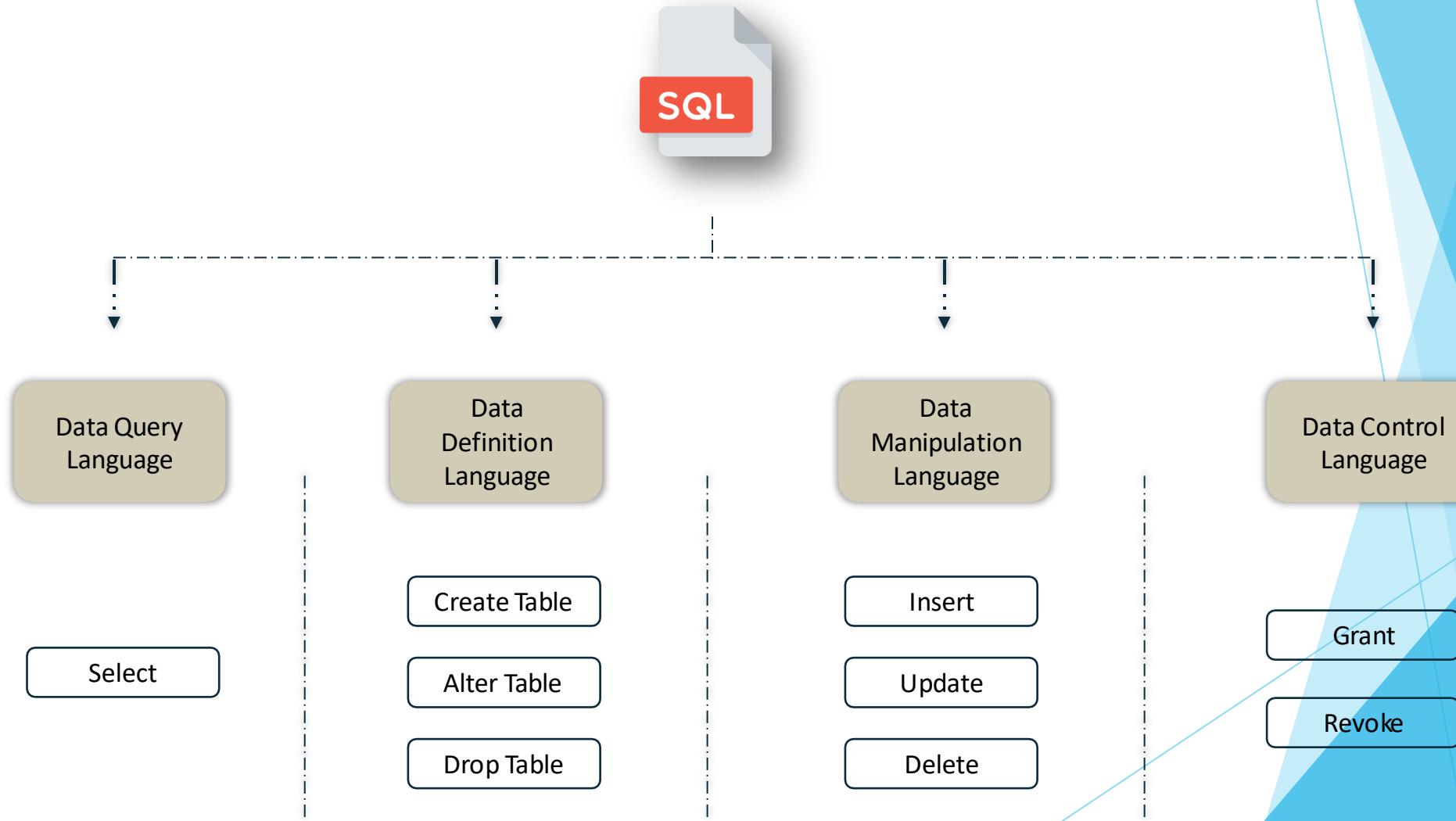


Introduction to SQL

SQL stands for Structured Query Language which is a standard language for accessing and manipulating databases.



Categories of SQL Commands



Agenda



Employee

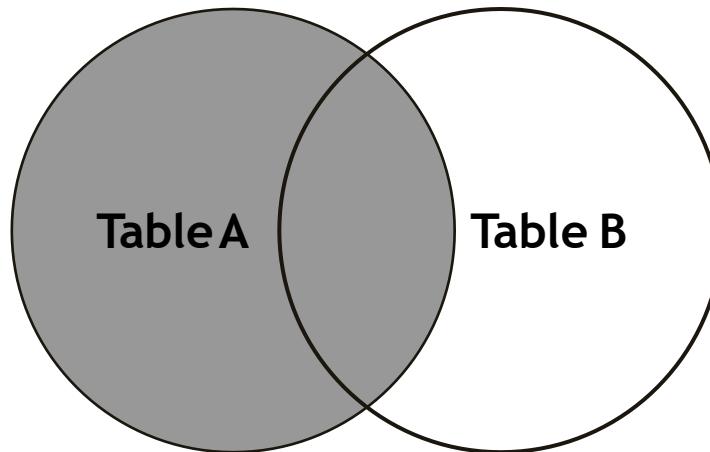
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Department

d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Left Join

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



Left Join: Syntax



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

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

LIKE Operator

LIKE Operator is used to extract records where a particular pattern is present.



John

Johnathon

Johny

Marcus

Wild Card Characters



→ Percentage Symbol →

Represents zero, one, or multiple characters



→ Underscore Symbol →

Represents a single character

LIKE Operator: Syntax



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

BETWEEN Operator

BETWEEN Operator is used to select values within a given range.

AGE



20



25

BETWEEN Operator: Syntax



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

Agenda



Employee

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Department

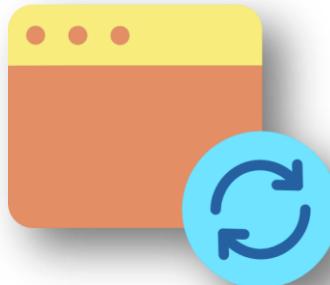
d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Merge

MERGE is the combination of INSERT, DELETE, and UPDATE statements.



Insert



Update



Delete

Merge

Source Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	93000	40	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	130000	25	Female	Analytics
6	Jeff	112000	27	Male	Operations
7	Adam	100000	28	Male	Content
8	Priya	85000	37	Female	Tech

Target Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Merge

Source Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	93000	40	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	130000	25	Female	Analytics
6	Jeff	112000	27	Male	Operations
7	Adam	100000	28	Male	Content
8	Priya	85000	37	Female	Tech

Merge

Target Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Merge



```
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];
```

Merge

Source Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	93000	40	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	130000	25	Female	Analytics
6	Jeff	112000	27	Male	Operations
7	Adam	100000	28	Male	Content
8	Priya	85000	37	Female	Tech

Employee_Source

Target Table

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Employee_Target

Agenda

- 1 Order By
- 2 TOP

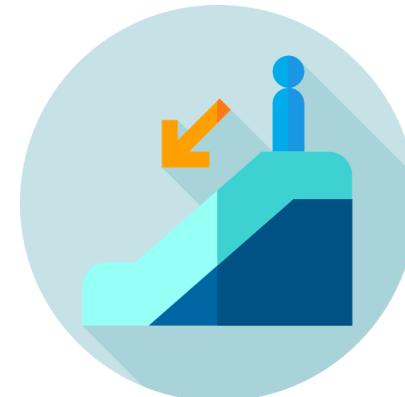
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Order By

ORDER BY is used to sort the data in ascending or descending order.



Ascending



Descending

Order By: Syntax



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

TOP Clause

TOP clause is used to fetch the **top N** records.

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

TOP Clause: Syntax



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

Agenda

1

Right Join in SQL

Employee

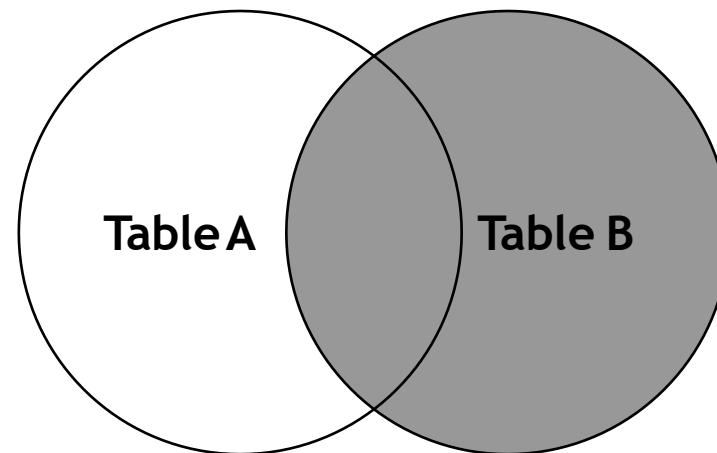
e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Department

d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Right Join

RIGHT JOIN returns all the records from the right table and the matched records from the left table.



Right Join: Syntax



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

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Select Statement: Syntax



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

Select Distinct: Syntax



Select **Distinct** is used to select only distinct values from our columns

```
SELECT DISTINCT column1,  
column2, columnN FROM table_name;
```

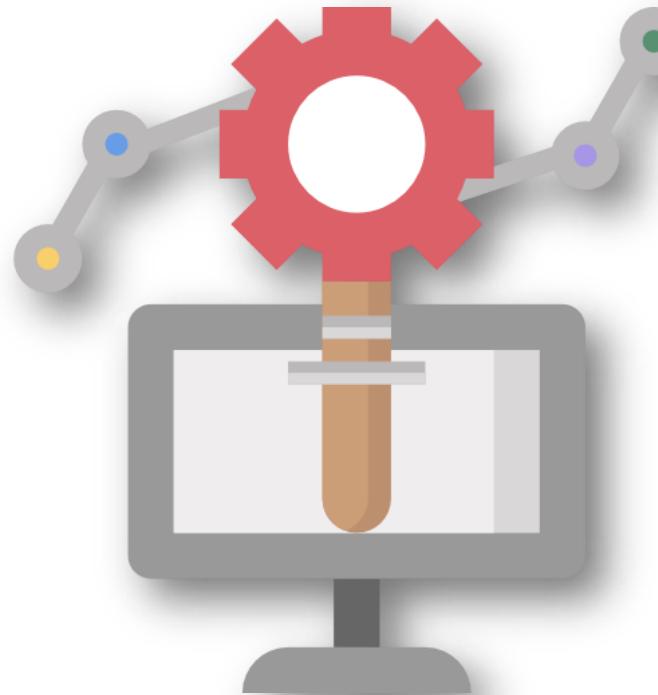
Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Stored Procedure in SQL

STORED PROCEDURE is a prepared SQL code which can be saved and reused.



Stored Procedure without Parameter: Syntax



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



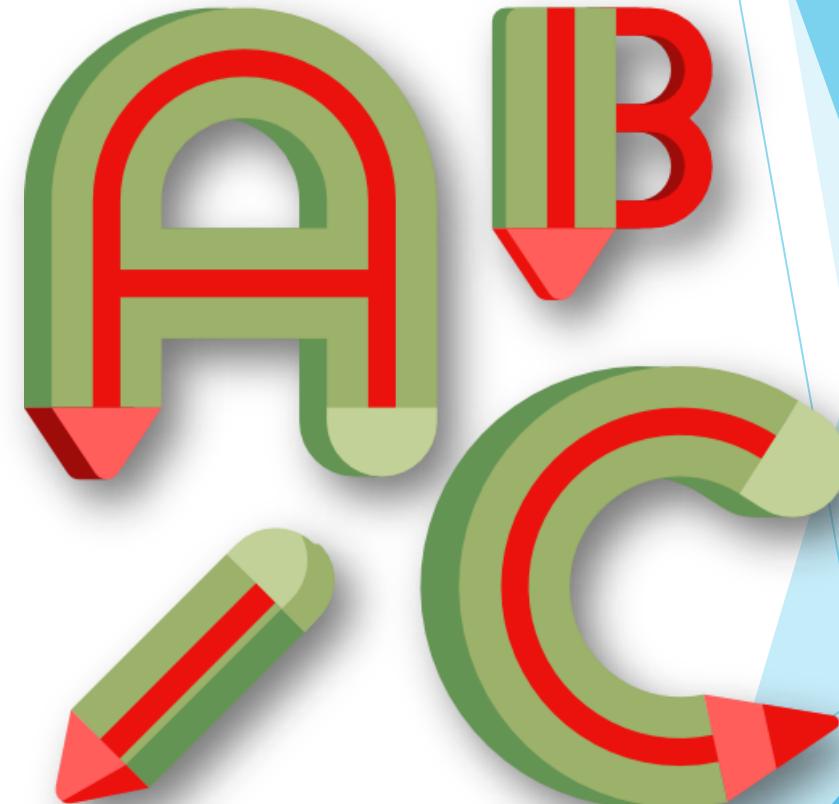
```
EXEC procedure_name
```

Stored Procedure with Parameter: Syntax



```
CREATE PROCEDURE procedure_name  
@param1 data-type, @param2 data-type  
AS  
sql_statement  
GO;
```

Agenda



String Functions

LTRIM()



Removes blanks on the left side of the character expression

LOWER()



Converts all characters to lower case letters

UPPER()



Converts all characters to upper case letters

REVERSE()



Reverses all the characters in the string

SUBSTRING()



Gives a substring from the original string

Agenda



Tables in SQL

A table is a database object which comprises rows and columns.

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Fields and Records

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Record

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Field

Fields

A field provides specific information about the data in a table.

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Records

Each individual entry in a table is called a record.

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Temporary Table

Temporary tables are created in tempDB and deleted as soon as the session is terminated.

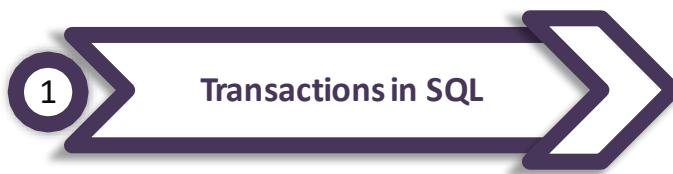


Syntax to Create Temporary Table



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

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Transactions in SQL

Transaction is a group of commands that change the data stored in a database.

```
begin try
    begin transaction
        update employee set e_salary=50
        where e_gender='Male'
        update employee set
        e_salary=195/0 where e_name='Female'
        commit transaction
        Print 'transaction committed'
    end try
    begin catch
        rollback transaction
        print 'transaction rolledback'
    end catch
```



Single Unit

Transactions in SQL

Transaction is a group of commands that change the data stored in a database.

```
begin try
    begin transaction
        update employee set e_salary=50
        where e_gender='Male'
        update employee set
e_salary=195/0 where e_name='Female'
        commit transaction
        Print 'transaction committed'
    end try
    begin catch
        rollback transaction
        print 'transaction rolledback'
    end catch
```



Command Fail



RollBack

Transactions in SQL

Transaction is a group of commands that change the data stored in a database.

```
begin try
    begin transaction
        update employee set
e_salary=50 where e_gender='Male'
        update employee set
e_salary=195 where e_name='Female'
        commit transaction
        Print 'transaction committed'
    end try
    begin catch
        rollback transaction
        print 'transaction rolledback'
    end catch
```

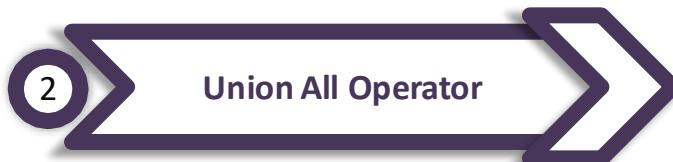


Commands
Success



Committed

Agenda



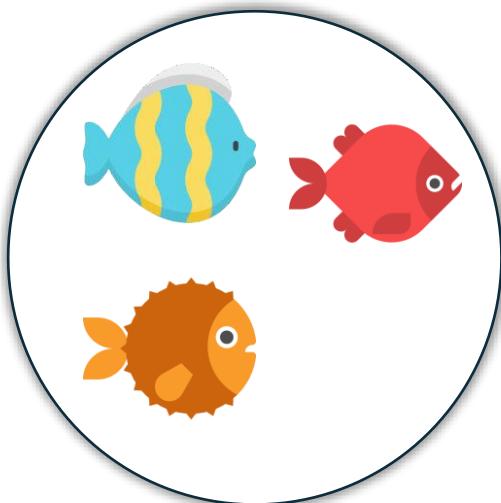
s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92



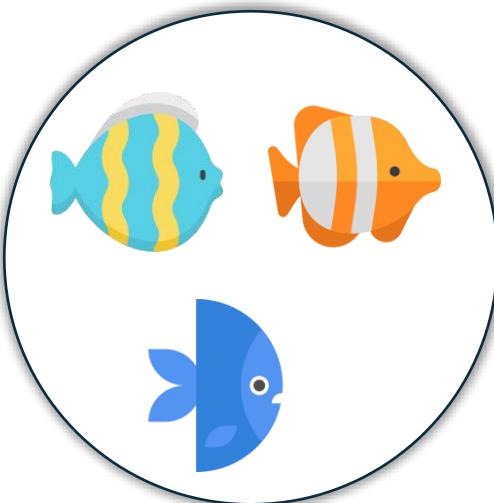
s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Union Operator

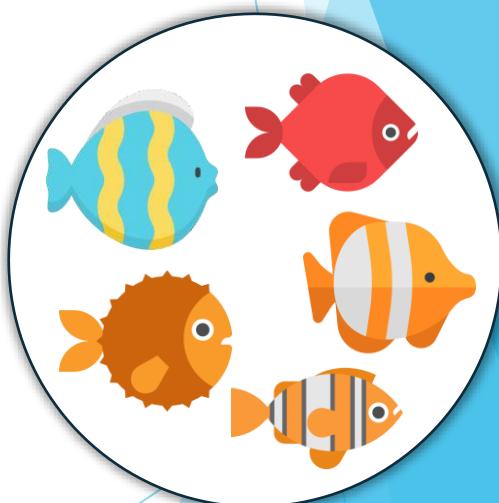
Union operator is used to combine the result set of two or more SELECT statements.



A



B



A ∪ B

Union Operator: Syntax



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

Union Operator

s_id	s_name	s_marks
1	Sam	45
2	Bob	87
3	Anne	73
4	Julia	92

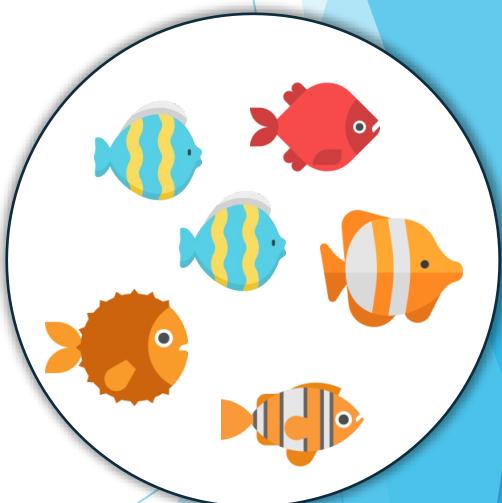
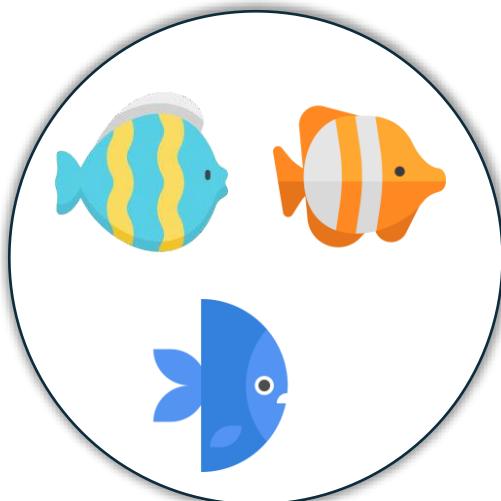
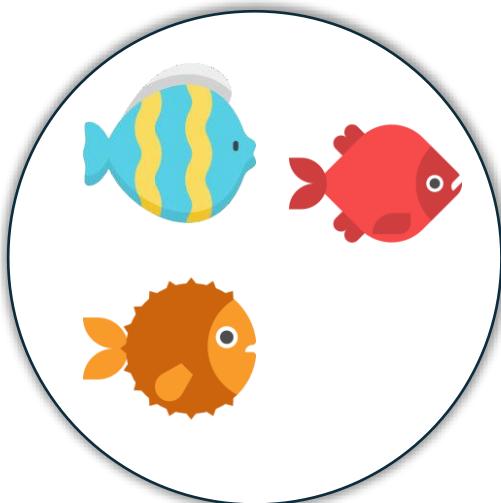
Student_Details1

s_id	s_name	s_marks
3	Anne	73
4	Julia	92
5	Matt	65

Student_Details2

Union All Operator

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



A

B

A union all B

Agenda

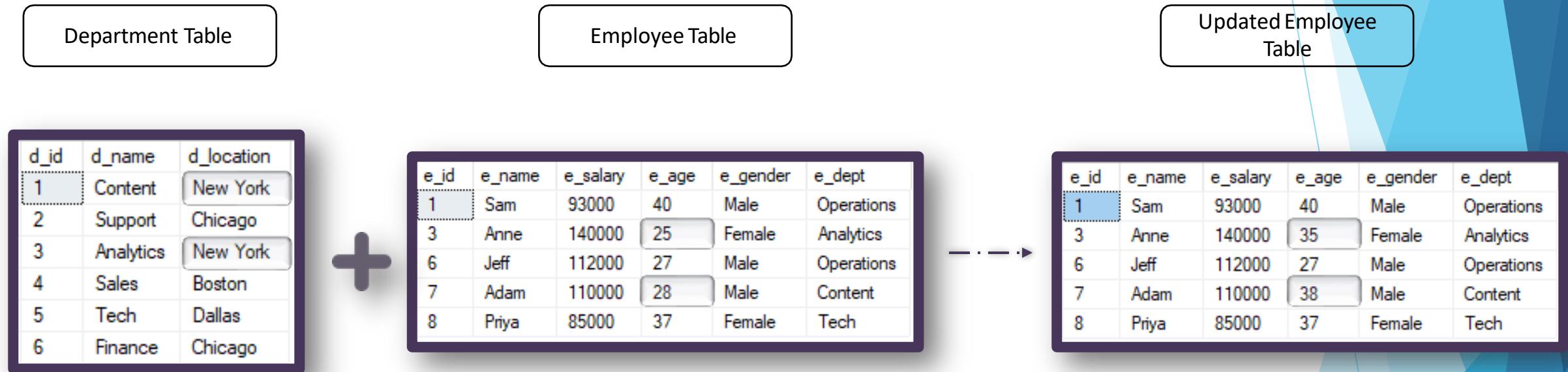


d_id	d_name	d_location
1	Content	New York
2	Support	Chicago
3	Analytics	New York
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

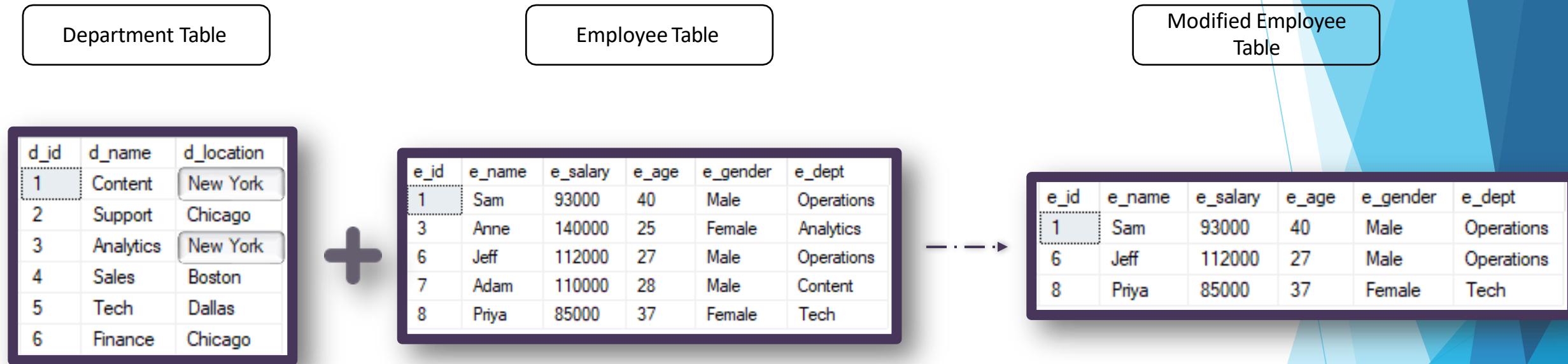


e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	93000	40	Male	Operations
3	Anne	140000	25	Female	Analytics
6	Jeff	112000	27	Male	Operations
7	Adam	110000	28	Male	Content
8	Priya	85000	37	Female	Tech

Update Using Join



Delete Using Join



Agenda

1

Update Statement

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Update Statement

Update is used to modify the existing records in a table.

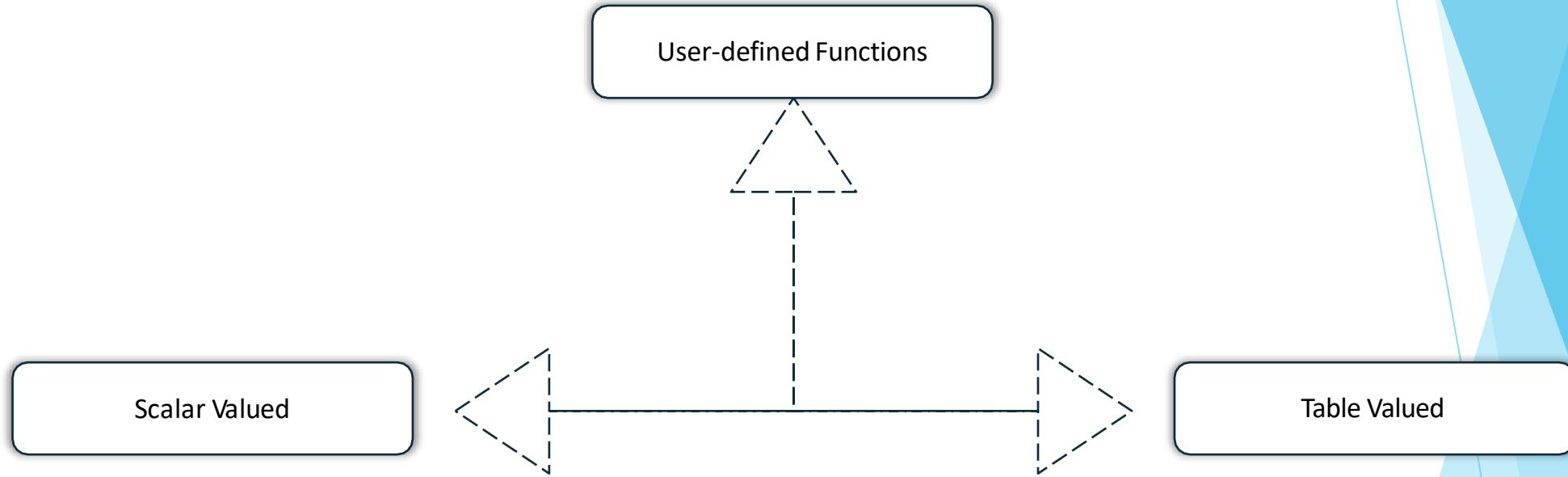
```
UPDATE table_name  
SET col1=val1, col2=val2.....  
[WHERE condition];
```

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Types of User-defined Functions



Scalar Valued Function

Scalar valued function always returns a scalar value.

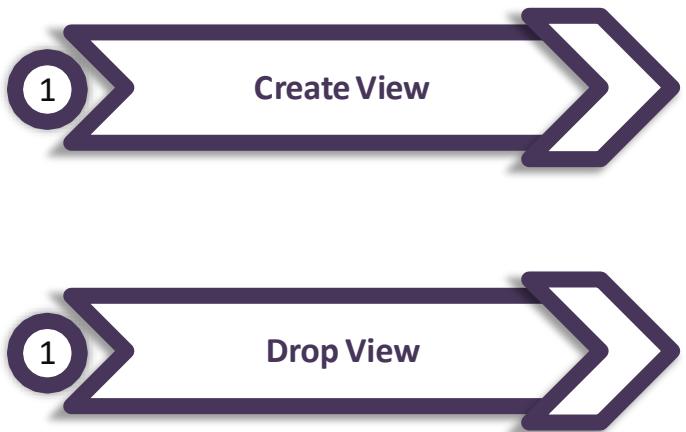
```
CREATE FUNCTION function_name(@param1 data_type, @param2  
data_type...)  
RETURNS return_datatype  
AS  
BEGIN  
    -----Function body  
RETURN value  
END
```

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] )
```

Agenda



e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Views

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

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Employee Table



	e_id	e_name	e_salary	e_age	e_gender	e_dept
1	3	Anne	125000	25	Female	Analytics
2	4	Julia	112000	30	Male	Analytics

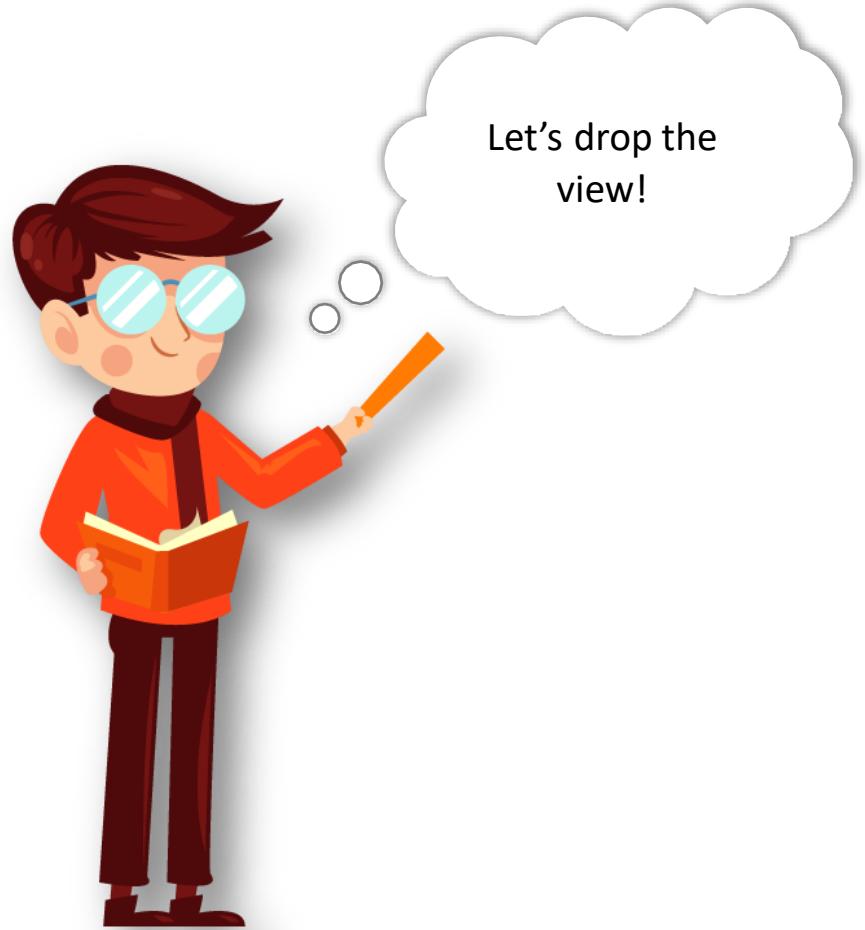
"Female_Employee" View

Create View: Syntax



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

Drop View: Syntax



```
DROP VIEW view_name;
```

Agenda

Where

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

Where Clause

Where clause is used to extract records which satisfy a condition.



Age>60



Occupation="Doctor"

Where Clause: Syntax



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

Thank You