



The SELECT Statement



DML Sublanguage

- DML (Data Manipulation Language)
 - A collection of commands that enable us to create, read, update and remove data from a database table
- INSERT – Adding a **record** into a table
- SELECT – Retrieval of data from table(s)
- UPDATE – Changing some data in a table
- DELETE – Removing some **records** from a table



SELECT Statement

■ Syntax

SELECT [ALL | DISTINCT] [TOP expr [PERCENT] [WITH TIES]]
 < select_list >

[FROM { <table_source> } [,...n]]

[WHERE <search_condition>]

[GROUP BY]

[HAVING <search_condition>]

[ORDER BY]

- All clauses are optional except the **select_list**



ALL and DISTINCT

■ ALL

- ☐ Specifies that duplicate rows can appear in the result set
- ☐ ALL is the default

■ DISTINCT

- ☐ Specifies that only unique rows can appear in the result set
- ☐ Duplicates are removed based on the column list
 - NOT according to the source table
- ☐ Null values are considered equal for the purposes of the DISTINCT keyword

Examples

1. SELECT *City*
FROM *Employee*
2. SELECT DISTINCT *City*
FROM *Employee*

EmpID	FName	LName	DepID	Salary	City	SupID
1	Taye	Mekonnen	1	27600	Addis Ababa	NULL
2	Sofia	Teferi	1	17600	Addis Ababa	1
3	Genet	Abebe	3	13000	Addis Ababa	4
4	Genet	Abera	4	14600	Adama	1
5	Hanna	Samuel	4	17600	Hawassa	5
6	Kaleb	Kassa	4	6000	Hawassa	5
7	Solomon	Michael	4	6000	Hawassa	5
8	Beza	Seifu	1	9500	Adama	1
9	Michael	Yossef	1	7100	Addis Ababa	4



TOP

TOP expression [PERCENT] [WITH TIES]

- TOP Indicates that only a specified first set or percent of rows will be returned from the query result set
- The expression can be either a **number** or a **percent** of rows
- WITH TIES
 - Indicates that the rows with tie values will be included even the number of rows returned will be more that the specified number
 - Can only be used with the **ORDER BY** clause

Examples

1. SELECT TOP 3 **FName**
FROM **Employee**
2. SELECT TOP 1 WITH TIES FName, Salary
FROM Employee ORDER BY Salary
3. SELECT TOP 1 **FName** FROM **Employee** ORDER BY Salary

EmpID	FName	LName	DepID	Salary	City	SupID
1	Taye	Mekonnen	1	27600	Addis Ababa	NULL
2	Sofia	Teferi	1	17600	Addis Ababa	1
3	Genet	Abebe	3	13000	Addis Ababa	4
4	Genet	Abera	4	14600	Adama	1
5	Hanna	Samuel	4	17600	Hawassa	5
6	Kaleb	Kassa	4	6000	Hawassa	5
7	Solomon	Michael	4	6000	Hawassa	5
8	Beza	Seifu	1	9500	Adama	1
9	Michael	Yossef	1	7100	Addis Ababa	4



The SELECT List

- The select list is a commas-delimited series of expr
 - These are the columns that make up the result set
- The select list can be given as:
 - | { table_name | table_alias }.*
 - | [table_name | table_alias .] column_name
 - | expression
- [*] - Specifies that all columns from all tables in the FROM clause should be returned
 - [tableName.*] Limits the scope of the * to the specified table.
- Expression can be any valid combination of:
 - Constants
 - Functions
 - Column Names
 - Operator(s)



Aliases

- Aliases are used to specify names for the results of expressions
- Column Alias is used to replace the column name in the query result
- Table Alias
 - Is an alias for a table that can be used either for convenience or to distinguish a table in a self-join
 - Frequently a shortened table name is used to qualify specific columns of the tables in a join
 - If a table has an alias, the table name cannot be used in the WHERE clause

```
SELECT  E.FirstName + E.LastName , D.DepName
FROM    Employee AS E JOIN Department AS D
        ON E.DepID = D.DepID
WHERE   E.City = 'Addis Ababa'
```



The FROM Clause

- The FROM Clause lists the TABLE(s) or VIEWS from which the columns are to be retrieved
- The FROM clause is required
 - Except when the select list contains only constants, variables, and arithmetic expressions
- JOINS are typically defined in the FROM clause
- The FROM clause can include:
 - Table names
 - View names
 - Joins
 - Subqueries



The WHERE Clause

- Specifies the search condition that must be satisfied by the rows to be returned by the query
- The Search Condition
 - An expression that defines the condition to be met for the rows to be returned
 - A BOOLEAN expression
- The WHERE clause can also be used to define a JOIN
- Example:

```
SELECT    column-name-list
FROM      table_1 , table_2
WHERE     table_1.column_A = table_2.column_B
```



Operators

- **NOT** : Negates the Boolean expression specified by the predicate
- **AND** : Combines two conditions and evaluates to TRUE when both of the conditions are TRUE
- **OR** : Combines two conditions and evaluates to TRUE when either condition is TRUE
- Other Operators
 - IS NULL
 - BETWEEN
 - IN
 - LIKE



Operators

- IS [NOT] NULL : Specifies a search for null values

```
SELECT * FROM Employee
WHERE GFName IS NOT NULL
```

- [NOT] BETWEEN

- ☐ The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- ☐ The BETWEEN operator is **INCLUSIVE**: begin and end values are included.
 - What about the “NOT BETWEEN”

- Example

```
SELECT artist, album, year
FROM   Collection
WHERE  year BETWEEN 1980 AND 1992
```



IN Operator

■ [NOT] IN :

- Specifies the search for an expression, based on whether the expression is included in or excluded from a **LIST**
- The search expression can be a constant or a column name, and the list can be a set of **CONSTANTS** or a **SUBQUERY**
- The list of values must be enclosed in a **parentheses**

■ Examples

```
SELECT *  
FROM Employee  
WHERE FirstName IN ('Abebe', 'Abeba')
```

```
SELECT album, year  
FROM Collection  
WHERE year NOT IN (1996, 1997)
```



LIKE Operator

- [NOT] LIKE operator is used in a WHERE clause to search for a specified **PATTERN** in a column
- Wildcards are used in with the LIKE operator:
 - [%] - Represents zero, one, or multiple characters
 - [_] - Represents a single character
 - [] - Matches any single character within a specified range
 - [^] - Matches any single character that is not within the range or set specified between the square brackets

- Example

```
SELECT *  
FROM Employee  
WHERE FirstName LIKE ('A%')
```



LIKE Operator – Examples 1

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'm[n-z]%'	start with the letter m second letter must be between n to z followed by
WHERE CustomerName LIKE 'Ap[^a]%'	first names that start with Ap and have a third letter that is not the letter



LIKE Operator – Examples 2

- Select all Employees whose first names contains 'ss'

```
SELECT      *  
FROM        Employee  
WHERE       FirstName LIKE '%ss%'
```

- Select all Employees whose last names do not start with letter from A to E

```
SELECT      FName, LName  
FROM        Customers  
WHERE       LName LIKE '[^A-E]%'
```



LIKE Operator – Examples 2

- [] - Any single character
 - within the specified range
[a-f]
 - within the specified range set
[abcdef]
- [^] - Any single character
 - NOT within the specified range
[^a – f]
 - NOT within the specified set
[^abcdef]



The ORDER BY Clause

- ORDER BY sorts the rows for presentation purpose
- Null values are treated as the lowest possible values for the purpose of the ORDER BY clause
- ORDER BY is the last clause to be processed
- ORDER BY can refer to
 - Columns by Name, Alias, or Ordinal Position
 - Columns that are not included in the select list
- Sort order can be ASC or DESC
- Multiple sort columns can be specified
- A sort column can include an expression



ORDER BY - Examples

- Sort a result set by multiple columns and different orders

```
SELECT    FirstName, LastName
FROM      Customers
ORDER BY  City DESC, FirstName ASC
```

- Sort a result set by an expression

```
SELECT    FirstName, LastName
FROM      Customers
ORDER BY  LEN(FirstName) DESC
```

- Sort by ordinal positions of columns

```
SELECT    FirstName, LastName
FROM      Customers
ORDER BY  2, 1
```



JOINS



JOIN

- A JOIN clause is used to combine rows from one or more tables, based on a related column
- The different types of the JOINS in SQL are:
 - (INNER) JOIN: Returns records that have matching values in both tables
 - LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
 - RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
 - FULL (OUTER) JOIN: Return all records when there is a match in either left or right table
 - CROSS JOIN: Matches each record from the right table with each record from the left table.



INNER JOIN - Syntax

■ Option 1

```
SELECT    column-name-list  
FROM      table_1 , table_2  
WHERE     table_1.column_A = table_2.column_B
```

■ Option 2

```
SELECT    column-name-list  
FROM      table_1 INNER JOIN table_2  
ON table_1.column_A = table_2.column_B
```

JOINS

■ Example

- Find all products under \$200 manufactured in Japan;
- Return their names and prices

Product (PName, Price, Category, Manufacturer)

Company (CompName, StockPrice, Country)

```
SELECT PName, Price
FROM Product, Company
WHERE Manufacturer=CompName AND Country='Japan'
AND Price <= 200
```

Join
between Product
and Company

JOINS

■ Example

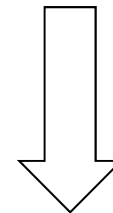
Product

PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

Company

CompName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT PName, Price
FROM Product, Company
WHERE Manufacturer=CName AND Country='Japan'
AND Price <= 200
```



PName	Price
SingleTouch	\$149.99

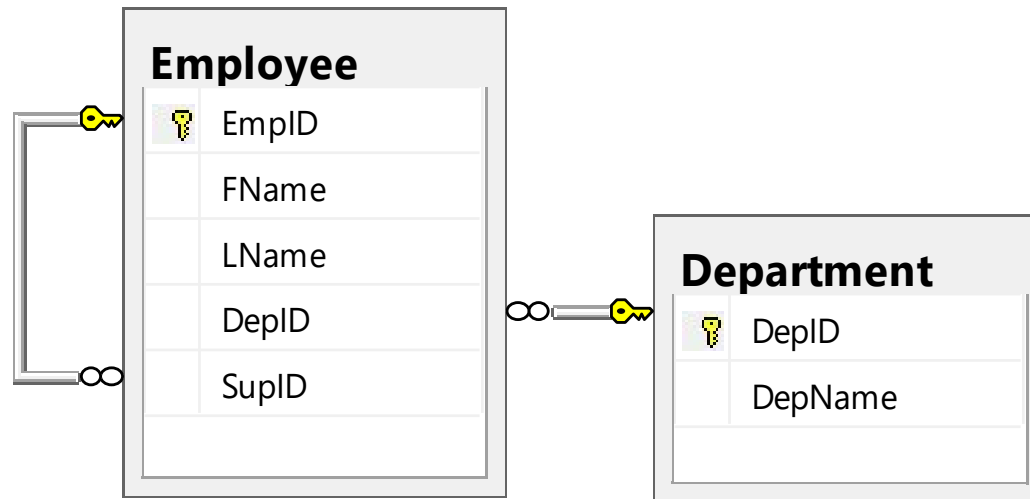
SELF JOIN - Example

```
SELECT      E.FName + ' ' + E.LName AS 'Employee Name'
            , S.FName + ' ' + S.LName AS Supervisor
FROM        Employee E JOIN Employee S
            ON E.SupID = S.EmpID
```

Employee				
EmpID	Fname	Lname	DepID	SupID
1	Taye	Mekonne	1	NULL
2	Sofia	Teferi	1	1
3	Genet	Abebe	2	2
4	Daniel	Abera	2	3
5	Hanna	Samuel	3	1
6	Kaleb	Kassa	3	5
7	Solomon	Michael	3	5
8	Beza	Seifu	2	3
9	Michael	Yossef	3	5
10	Hanna	Natnael	3	5

Employee Name	Supervisor
Sofia Teferi	Taye Mekonnen
Genet Abebe	Sofia Teferi
Daniel Abera	Genet Abebe
Hanna Samuel	Taye Mekonnen
Kaleb Kassa	Hanna Samuel
Solomon Michael	Hanna Samuel
Beza Seifu	Genet Abebe
Michael Yossef	Hanna Samuel
Hanna Natnael	Hanna Samuel

JOIN – Example 2



```
SELECT      E.FName + ' ' + E.LName AS 'Employee Name', D.DepName
FROM        Employee E
JOIN        Department D ON E.DepID = D.DepID
```



JOIN – Example 3

■ Joining 3 tables

```
SELECT    O.OrderNumber, P.ProductName, I.Quantity, I.UnitPrice
FROM      [Order] O
          JOIN OrderItem I ON O.Id = I.OrderId
          JOIN Product P ON P.Id = I.ProductId
ORDER BY  O.OrderNumber
```



JOIN – Example 4

- CROSS Join

```
SELECT *  
FROM   Employee, Department
```

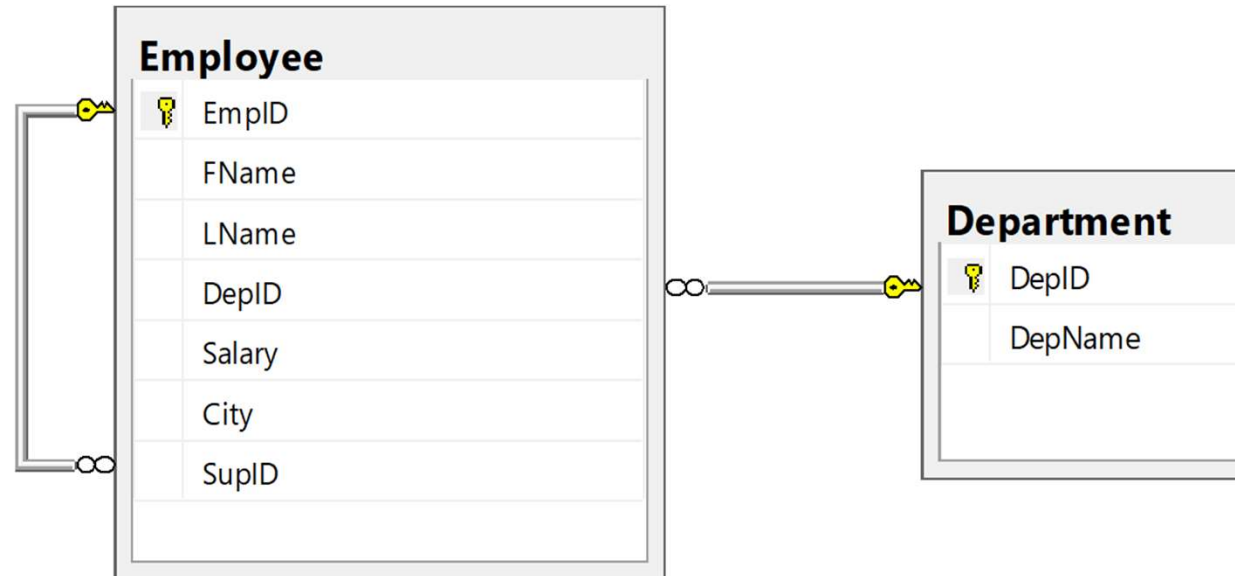
- LEFT OUTER Join

```
SELECT *  
FROM   Department LEFT OUTER JOIN Employee  
       ON Department.DepID = Employee.DepID
```

- FULL OUTER Join

```
SELECT *  
FROM   Department FULL OUTER JOIN Employee  
       ON Department.DepID = Employee.DepID
```

Exercise



1. Get all city names
2. Get employees and their departments
3. Get list of employees with their supervisors
4. Get the list of employees along with their department names and Supervisor names
5. Get employees ordered by city and salary
6. Get the names of the departments where there are no employees
7. Get total salary for each department
8. Get all cities and total salary paid for the employees in the city