DATA RETRIVAL AND SELECT STATEMENT

SELECT STATEMENT

- All retrieval operations can be done using SELECT statement.
- The result returned by a SELECT statement is always a single table.
- This result can be seen by the user but it is not stored in database.
- To store the result we must create a table and put the result of SELECT statement into it using INSERT INTO.

SELECT STATEMENT

- SELECT statement is not only used to retrieve raw data
- It can be used to obtain answers of complicated questions by making inferences on available data
  - Examples:
    - Which students are taking MIS course?
    - What is the number of students assigned to advisor ‘Uğur Yıldıran’?
    - What is GPA of ‘Mehmet Güler’?

OPERATIONS IN RELATIONAL MODEL

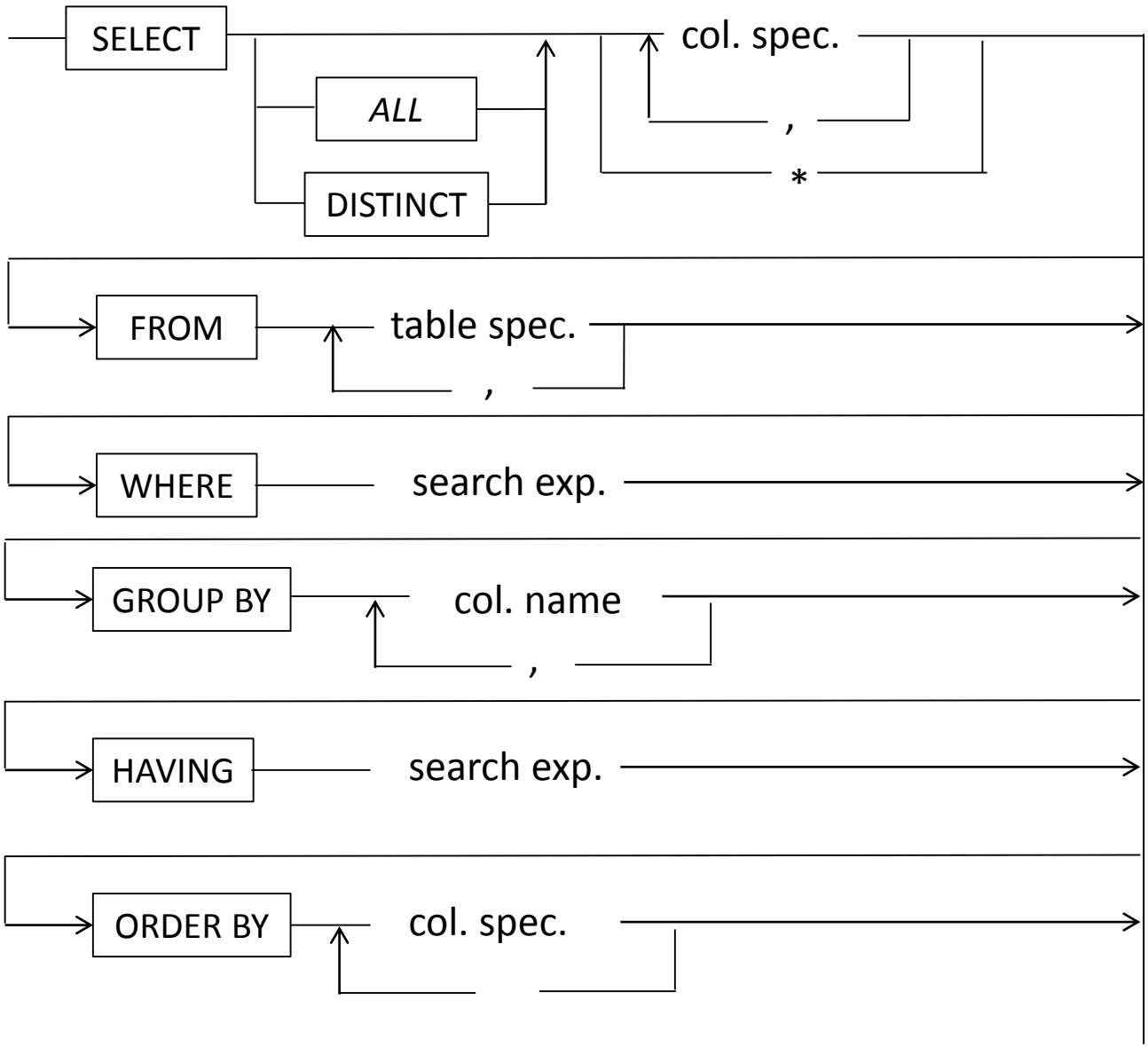
- There are a number of basic operations in relational database theory that can be employed for complicated information retrieval tasks:
  - Column selection (projection)
  - Row selection (restriction)
  - Union
  - Cartesian product
  - Join
  - Difference
  - Intersection
  - Division

OPERATIONS IN RELATIONAL MODEL

- All of these operations are not necessary
- We will study five of them supported by SQL
  - Column selection (projection)
  - Row selection (restriction)
  - Union
  - Cartesian product
  - Join
- Arguments of SELECT statement can be used to perform these operations and some additional tasks

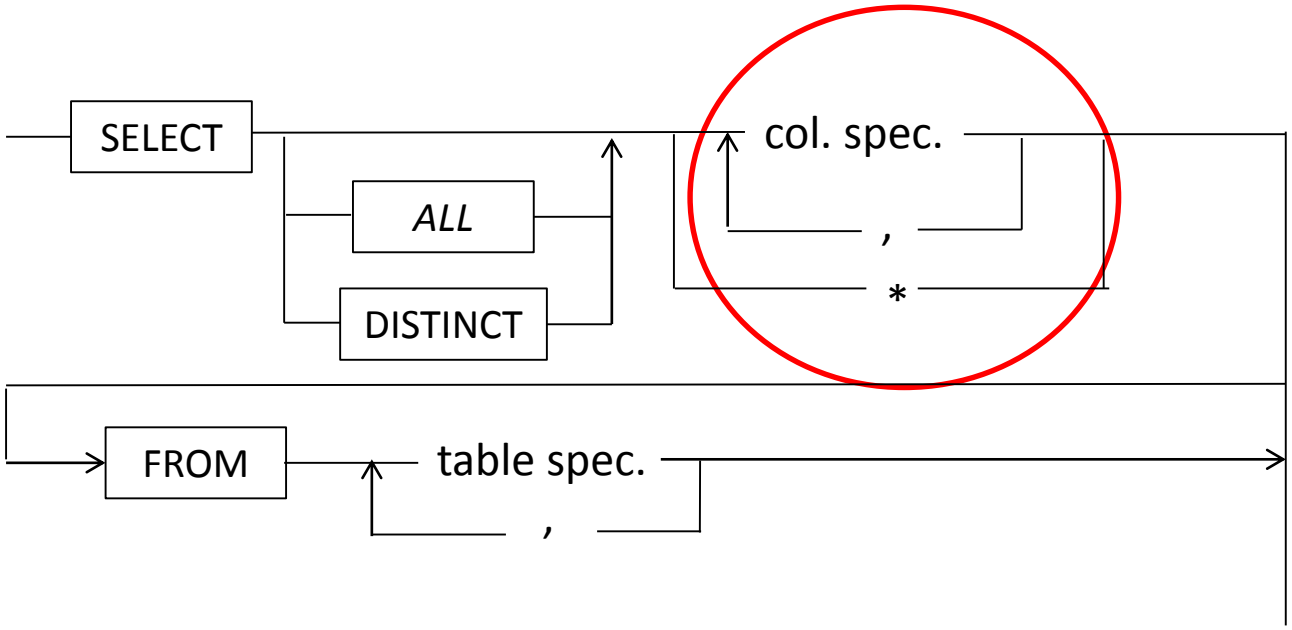
SELECT STATEMENT

Syntax diagram:



COLUMN SELECTION

- This operation is used to obtain all data in specified columns of a table
  - Specify the names of the columns just after the SELECT keyword
  - Specify the name of the table using FROM clause



COLUMN SELECTION

- **Example:** List the names, surnames and salaries of all employees

```
SELECT Name, Surname, Salary
FROM Employees
```

Name	Surname	Salary
Ali	Yeşil	1000
Ozan	Kara	3000
Ali	Ergin	2000
Ayşegül	Eren	2000
Hasan	Gazi	1500
Osman	Güneş	4000
Sezin	Temelli	2500

COLUMN SELECTION

- All columns can be selected using * character
  - SELECT *
    FROM Employees
  - This query returns Employees table

CALCULATED COLUMNS

- As *select-item* we are allowed to use expressions
- Expressions will be evaluated for each row of the table to obtain the values forming the column
- **Example:** List name, surname, age and net salary of all employees
  - SELECT Name, Surname,

    DATEDIFF(year, Birthday, GETDATE()),
    Salary-Salary*Tax/100

    FROM Employees

CALCULATED COLUMNS

Name	Surname		
Ali	Yeşil	34	950
Ozan	Kara	42	2550
Ali	Ergin	32	1800
Ayşegül	Eren	26	1800
Hasan	Gazi	47	1425
Osman	Güneş	52	3400
Sezin	Temelli	40	250

- In the result calculated columns will not have name
- AS can be used to assign names to columns with or without names

CALCULATED COLUMNS

- SELECT Name, Surname,

  DATEDIFF(year, Birthday, GETDATE()) AS Age,
  Salary-Salary*Tax/100 AS Net_Salary

  FROM Employees

Name	Surname	Age	Net_Salary
Ali	Yeşil	34	950
Ozan	Kara	42	2550
Ali	Ergin	32	1800
Ayşegül	Eren	26	1800
Hasan	Gazi	47	1425
Osman	Güneş	52	3400
Sezin	Temelli	40	250

DUPLICATE ROWS

- As we saw, in relational model a table cannot have multiple copies of the same row
- Unfortunately, SQL does not obey this rule
- Table returned by column selection operation can contain duplicate rows

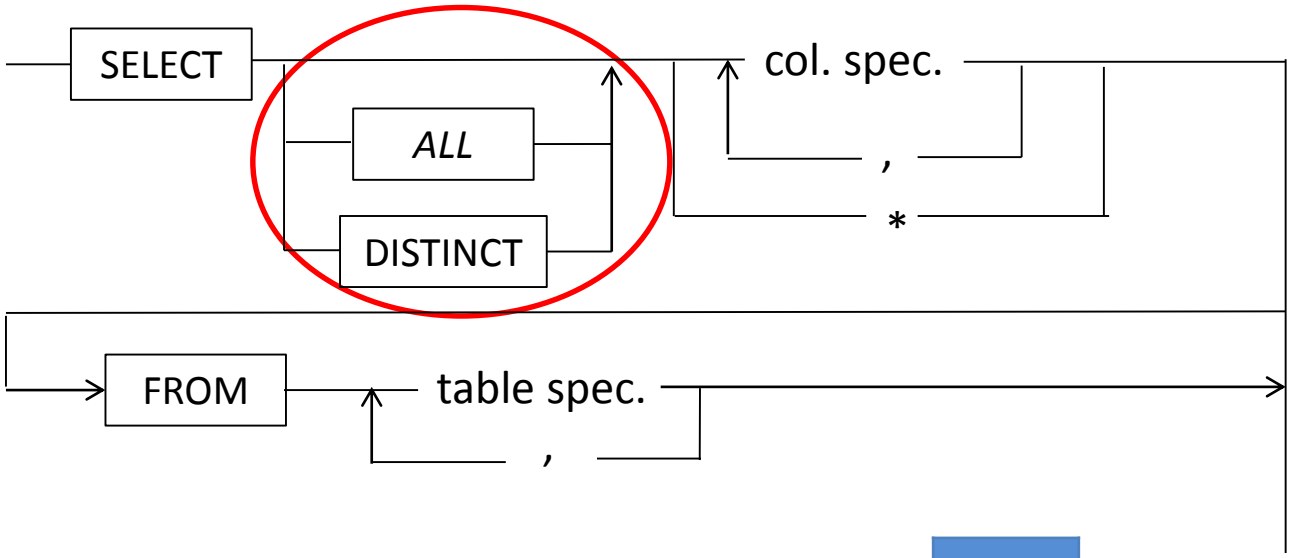
Example: List all departments in the company

```
SELECT Dept
FROM Employees
```

Dept
1
1
2
4
3
2
3

DUPLICATE ROWS

- We can eliminate duplicate rows using **DISTINCT** keyword



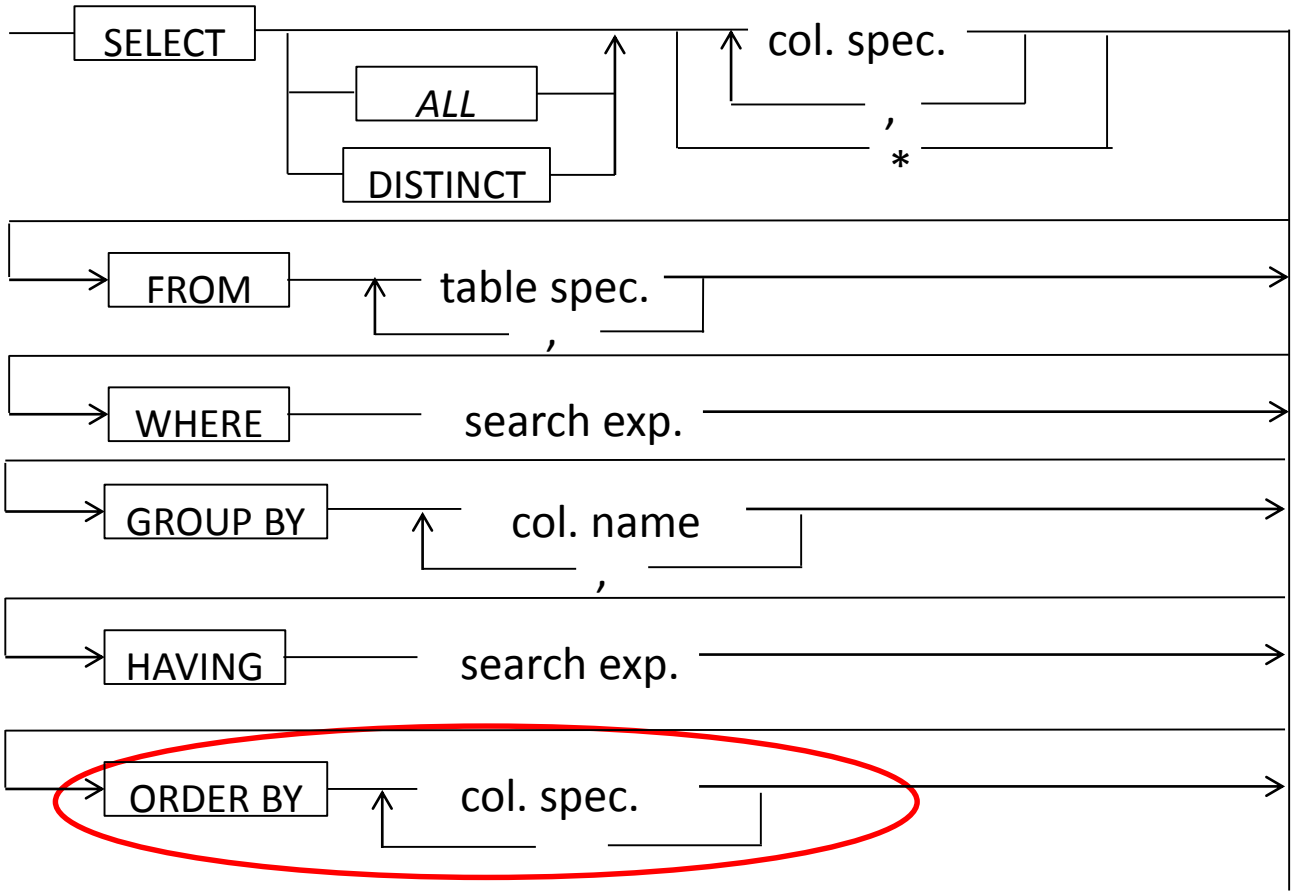
SELECT DISTINCT Dept

FROM Employees

Dept
1
2
4
3

ORDERING

- Results of a query can be ordered with respect to a column or combinations of columns using **ORDER BY** clause



ORDERING

- List the names, surnames and salaries of employees by ordering the result with respect to salaries.

```
SELECT Name, Surname, Salary
FROM Employees
ORDER BY Salary
```

Name	Surname	Salary
Ali	Yeşil	1000
Hasan	Gazi	1500
Ali	Ergin	2000
Ayşegül	Eren	2000
Sezin	Temelli	2500
Ozan	Kara	3000
Osman	Güneş	4000

ORDERING

- Default ordering is ascending (smallest to largest)
- Descending ordering can be employed using DESC keyword just after column name in *short-specification*

```
SELECT Name, Surname, Salary
FROM Employees
ORDER BY Salary DESC
```

Name	Surname	Salary
Osman	Güneş	4000
Ozan	Kara	3000
Sezin	Temelli	2500
Ali	Ergin	2000
Ayşegül	Eren	2000
Hasan	Gazi	1500
Ali	Yeşil	1000

ORDERING

- Ordering can be done with respect to more than one column
- **Example:** List the names, surnames, salary and departments of all employees by first ordering them by department in ascending order and then by salaries in descending order.

```
SELECT Name, Surname, Salary, Dept
FROM Employee
ORDER BY Dept, Salary DESC
```

Name	Surname	Salary	Dept
Ozan	Kara	3000	1
Ali	Yeşil	1000	1
Osman	Güneş	4000	2
Ali	Ergin	2000	2
Sezin	Temelli	2500	3
Hasan	Gazi	1500	3
Ayşegül	Eren	2000	4

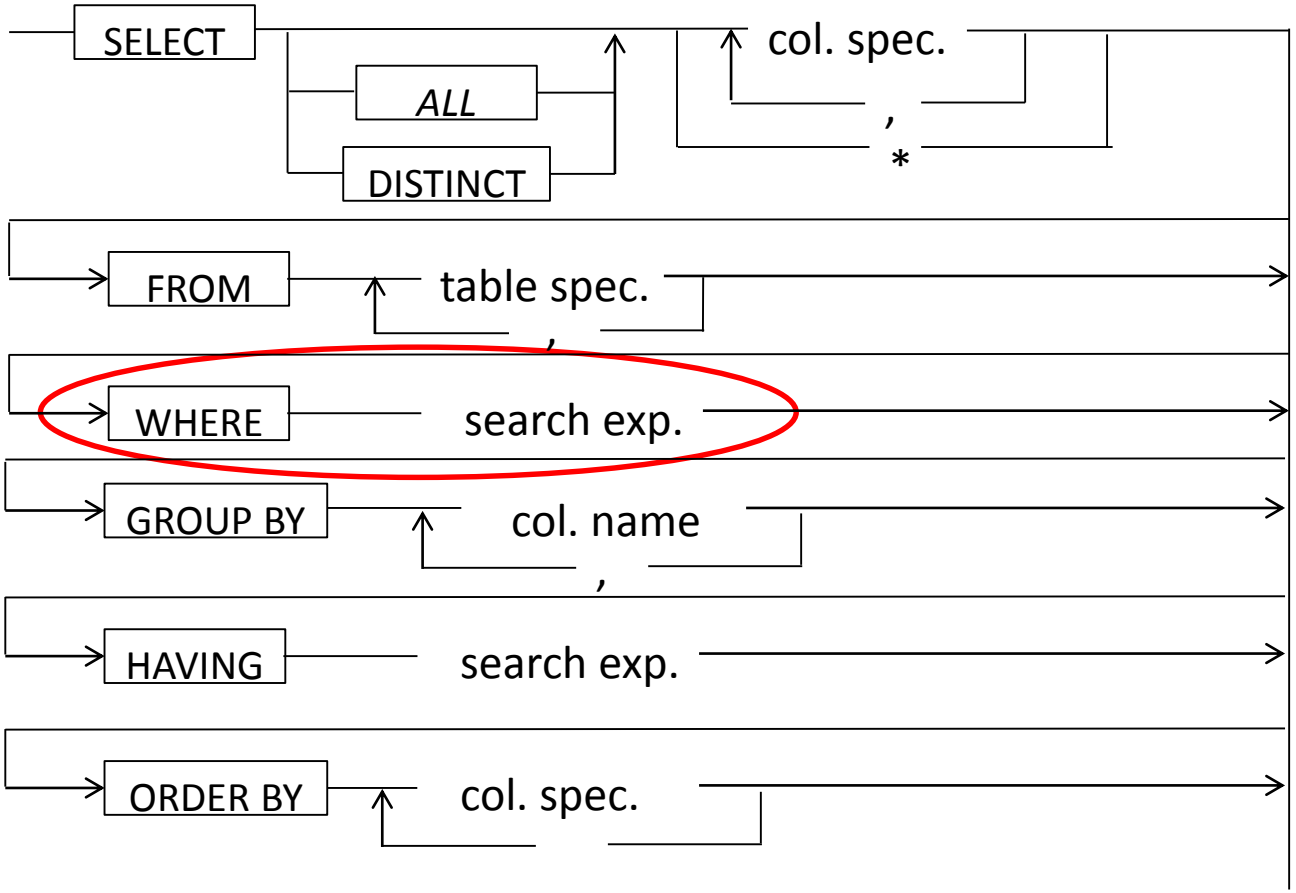
ORDERING

- Ordering can be done with respect to calculated columns
- **Example:** List names, surnames and ages of employees ordering the result with respect to their age

```
SELECT Name, Surname,
        DATEDIFF(year, Birthday, GETDATE())
FROM Employees
ORDER BY DATEDIFF(year, Birthday, GETDATE())
```

ROW SELECTION

- Another important operation is row selection
- This operation can be performed by specifying a *search-condition* in WHERE clause



ROW SELECTION

- *Searh-condition* is a logical expression
- This logical expression is evaluated for each row of the table and the rows that makes it TRUE are selected
- **Example:** List all information of employees working at department 3

```
SELECT *
FROM Employees
WHERE Dept=3
```

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Hasan	Gazi	04.03.1965 00:00:00	1500	%5	3	M	… 20/3 Levent …
Sezin	Temelli	05.04.1972 00:00:00	2500	%10	3	F	…. 3/2 Üsküdar …

ROW SELECTION

- Row selection can be used together with column selection
- List names and surnames of male employees whose salary is more than 2000TL.

ROW SELECTION

- Row selection can be used together with column selection
- List names and surnames of male employees whose salary is more than 2000TL.

```
SELECT Name, Surname
FROM Employees
WHERE Gender='M' AND Salary>2000
```

ROW SELECTION

- List name, surname and age of all female employees together with all male employees older than 35.

ROW SELECTION

- List name, surname and age of all female employees together with all male employees older than 35.

```
SELECT Name, Surname,
        DATEDIFF(year, Birthday, GETDATE())
FROM Employees
WHERE Gender='F' OR
        (Gender='M' AND
        DATEDIFF(year, Birthday, GETDATE())>35)
```

ROW SELECTION

- Comparison can be done on strings and dates
- **Examples:**
  - List the information of employees whose name comes after Hasan

```
SELECT *
FROM Employees
WHERE Name>’Hasan’
```

  - List the information of employees born after 1970

```
SELECT *
FROM Employees
WHERE Birthday>=’01.01.1971 00:00:00’
```

ROW SELECTION

- There are additional functions that can be employed in logical expressions
  - BETWEEN
  - IN
  - LIKE
- The first two do not provide new functionalities
- They just simplifies expressions

ROW SELECTION

- BETWEEN can be used if a value is between two other values (including them)
- **Example:** List the employees whose salary is between (including) 1500 and 2500

```
SELECT *
FROM Employees
WHERE Salary BETWEEN 1500 AND 2500
```

ROW SELECTION

- This query is equivalent to the following:

```
SELECT *
FROM Employees
WHERE Salary>=1500 AND Salary<=2500
```

- BETWEEN can be used with NOT as well

```
SELECT *
FROM Employees
WHERE Salary NOT BETWEEN 1500 AND 2500
```

ROW SELECTION

- IN can be used to check if a value belongs to a set of elements
- **Example:** List the employees working at departments 1, 2 or 3

```
SELECT *
FROM Employees
WHERE Dept IN(1,2,3)
```

ROW SELECTION

- This is equivalent to the following:

```
SELECT *
FROM Employees
WHERE Dept=1 OR Dept=2 OR Dept=3
```

- IN can be used with NOT as well

```
SELECT *
FROM Employees
WHERE Dept NOT IN(1,2,3)
```

ROW SELECTION

- LIKE function simply compares two string and returns true if they are equal
- But using special wildcard characters **‘%’ and ‘_’** we can search a string (a keword) in another string
  - %: is a placeholder for an arbitrariy string having arbitrary length
  - _: is a placeholder for an arbitrariy character

ROW SELECTION

- **Examples:**
  - ‘%Bostancı’ means any string ending with Bostancı
    - ‘İstasyon Caddesi, Bostancı’
    - ‘Bağdat Caddesi, Alt Bostancı’
    - ‘Dr. Hayri Bostancı’
  - ‘Bostancı%’ means any string starting with Bostancı
    - ‘Bostancı Semti’
    - ‘Bostancı/İstanbul, Türkiye’
    - ‘Bostancı canavarı’
  - ‘%Bostancı%’ means any string containing Bostancı
    - ‘İstasyon Caddesi, Bostancı, Kadıköy/İstanbul’

ROW SELECTION

- **Examples:**
  - ‘Bostanc_’
    - ‘Bostancı’
    - ‘Bostanci’
    - ‘Bostanca’
    - etc.
  - ‘Bo_tanc_’
    - ‘Bomtanca’
    - ‘Bortancu’
    - etc.

ROW SELECTION

- List the Names and Surnames of employees living in Kadıköy. Note that some addresses are recorded using Turkish characters while other using English characters

```
SELECT Name, Surname
FROM Employees
WHERE Address LIKE ‘%Kad_k_y%’
```

THREE VALUED LOGIC

- As we saw NULL represents unknown values
- What happens if a NULL value is used in an logical expression?
- Result of comparison of a value with NULL will be NULL
  - 1>NULL, Result: NULL
  - ‘Metin’=NULL, Result: NULL

THREE VALUED LOGIC

- If the result of a logical operation is NULL, the corresponding row will not be selected in row selection operation
- Suppose that the salary of Hasan Gazi is NULL

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Ali	Yeşil	10.10.1978	1000	%5	1	M	…. 114/9 Bostancı
Ozan	Kara	05.01.1970	3000	%15	1	M	… 100/3 Bakırköy
Ali	Ergin	03.12.1980	2000	%10	2	M	…. 4/10 Levent ….
Ayşegül	Eren	07.19.1986	2000	%10	4	F	…. 56/2 Bostancı ….
Hasan	Gazi	04.03.1965	NULL	%5	3	M	… 20/3 Levent …
Osman	Güneş	02.02.1960	4000	%15	2	M	…. 22/19 Kayışdağı
Sezin	Temelli	05.04.1972	2500	%10	3	F	…. 3/2 Üsküdar …

THREE VALUED LOGIC

```
SELECT Name, Surname
FROM Employees
WHERE Salary>2000
```

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Ozan	Kara	05.01.1970	3000	%15	1	M	… 100/3 Bakırköy
Osman	Güneş	02.02.1960	4000	%15	2	M	…. 22/19 Kayışdağı
Sezin	Temelli	05.04.1972	2500	%10	3	F	…. 3/2 Üsküdar …

```
SELECT Name, Surname
FROM Employees
WHERE Salary<=2000
```

Name	Surname	Birthday	Salary	Tax	Dept	Gender	Address
Ali	Yeşil	10.10.1978	1000	%5	1	M	…. 114/9 Bostancı
Ali	Ergin	03.12.1980	2000	%10	2	M	…. 4/10 Levent ….
Ayşegül	Eren	07.19.1986	2000	%10	4	F	…. 56/2 Bostancı ….

THREE VALUED LOGIC

- We cannot see Hasan Gazi in both queries since the result of search condition is NULL in both queries
- Suppose that we want to list employees whose salary is not known
- We cannot use the following query for this purpose

```
SELECT Name, Surname
FROM Employees
WHERE Salary=NULL
```

THREE VALUED LOGIC

- Here result of Salary=NULL will be NULL independent of the value of Salary
- Hence SELECT statement will return an empty table
- We have to use IS NULL function

```
SELECT Name, Surname
FROM Employees
WHERE Salary IS NULL
```

THREE VALUED LOGIC

- Result of a logical operation (AND, OR, NOT) containing a NULL value will depend on operation

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

THREE VALUED LOGIC

- List name and surname of all female employees together with all male employees whose salary is greater than 2000

```
SELECT Name, Surname
FROM Employees
WHERE Gender='F' OR
        (Gender='M' AND Salary>2000)
```

- Does Hasan Gazi appear in the result?

ORDER OF OPERATIONS

- The operations we saw until now are applied in the following order while the query is being executed:
  1. WHERE clause is used to select rows
  2. Expressions in *select-item* list will be used to select/calculate columns
  3. If DISTINCT is used, duplicate rows will be eliminated
  4. ORDER BY is used to order the result

UNION

- Recall result of SELECT query is always a table
- Union can be used to merge results of queries vertically
- To be able to do this:
  - tables must have the same number of columns
  - types of the corresponding columns must be the same
- The second restriction makes use of the ordering of columns, which violates rules of relational model again

UNION

- Suppose there are two employee tables: Employee1 and Employee2
- List names and surnames of all employees

```
SELECT Name, Surname
FROM Employees1
UNION
SELECT Name, Surname
FROM Employee2
```