**WHERE ……………… > ……….. AND ………… < ……….. 🡪**

SELECT \*

FROM actor

WHERE first\_name = 'Penelope' AND last name = 'Monroe' ;

WHERE first\_name = 'Penelope' OR first\_name = 'Bob' ;

WHERE NOT 🡪 CLAUSE

WHERE NOT (rental\_rate = 4.99 OR rental\_rate = 2.99)

**Homework-1**

1. **Sort the data in the title and description columns in the first film table.**

SELECT title, description FROM film;

1. **Sort the data in all columns in the movie table with the film length greater than 60 AND less than 75.**

SELECT \* FROM film

WHERE length >60 and length < 75;

1. **Sort the data in all columns in the film table with rental\_rate 0.99 AND replacement\_cost 12.99 OR 28.99.**

SELECT \* from film

WHERE rental\_rate = 0.99

AND replacement\_cost = 28.99;

1. **What is the value in the last\_name column of the customer whose value is 'Mary' in the first\_name column of the customer table?**

SELECT first\_name, last\_name FROM customer

WHERE first\_name = 'Mary';

1. **Sort the data in the movie table whose length is NOT greater than 50, but whose rental\_rate is NOT 2.99 or 4.99.**

SELECT \* FROM film

WHERE NOT (length<50)

AND NOT (rental\_rate = 2.99 OR rental\_rate = 4.99);

**BETWEEN AND SYNTAX**

SELECT <column\_name>, <column\_name>, ...

FROM <table\_name>

WHERE <condition>;

--------------------------------------------------------------------------------------

SELECT \*

FROM film

WHERE length BETWEEN 100 AND 140;

------------------------------------------------------------------------------------------

-- WHERE length > 100 AND length < 140

**IN SYNTAX**

SELECT \*

FROM film

WHERE length IN (30,60,90,120);

We can also use the NOT IN construct for values out of the list.

**Homework-2**

1. **Sort all column data in the film table provided that the replacement cost value is greater than 12.99, equal and less than 16.99 (Use BETWEEN - AND structure.)**

SELECT \* FROM film

WHERE replacement\_cost BETWEEN 12.98 AND 16.98;

--12.99 and 16.99 included

1. **Sort the data in the first\_name and last\_name columns in the actor table provided that first\_name is the values 'Penelope' or 'Nick' or 'Ed'. (Use the IN operator.)**

SELECT first\_name, last\_name FROM actor

WHERE first\_name IN ('Penelope', 'Nick', 'Ed');

1. **Sort the data in all columns in the film table with rental\_rate 0.99, 2.99, 4.99 AND replacement\_cost 12.99, 15.99, 28.99. (Use the IN operator.)**

SELECT \* FROM film

WHERE rental\_rate IN (0.99, 2.99, 4.99)

AND replacement\_cost IN (12.99, 15.99, 28.99);

**LIKE / NOT LIKE**

**For multi character use ‘%’ but for single character use ‘\_’ symbol**

SELECT \*

FROM actor

WHERE first\_name LIKE 'P%';

SELECT \*

FROM actor

WHERE first\_name -- 'P%';

Both uses are same 😊

--\* 🡺 ILIKE

-- 🡺 LIKE

!-- 🡺 NOT LIKE

!--\* 🡺 NOT ILIKE

NOTE : The ILIKE operator is the case - insensitive version of the LIKE operator!

**Homework-3**

1. **List the country names in the country column of the country table, starting with the 'A' character and ending with the 'a' character.**

SELECT \* FROM country

WHERE country ILIKE 'A%a';

1. **List the country names in the country column of the country table, consisting of at least 6 characters and ending with the 'n' character.**

SELECT country FROM country

WHERE country ILIKE '\_\_\_\_%n';

1. **In the title column of the film table, list the movie names containing at least 4 'T' characters, regardless of upper- or lower-case letters.**

SELECT title FROM film

WHERE title ILIKE '%T%T%T%T%';

1. **From the data in all the columns in the film table, sort the data that starts with the title 'C' character, has a length greater than 90 and a rental\_rate of 2.99.**

SELECT title, length, rental\_rate FROM film

WHERE title LIKE 'C%' AND

length > 90 AND rental\_rate = 2.99;

**SELECT DISTINCT SYNTAX**

SELECT DISTINCT <columnName>, <columnName>, ...

FROM <tableName>;

**SELECT COUNT SYNTAX**

SELECT COUNT(\*)

FROM actor

WHERE first\_name = 'Penelope';

**MORE**

SELECT COUNT(DISTINCT <columnName>)

FROM actor

**Homework-4**

1. **Sort the different values in the replacement cost column in the film table.**

SELECT DISTINCT replacement\_cost FROM film;

1. **How many different data are there in the replacement cost column in the film table?**

SELECT COUNT(DISTINCT replacement\_cost) FROM film;

1. **How many of the film titles in the film table start with the character T and at the same time the rating is equal to 'G'?**

SELECT COUNT(title) FROM film

WHERE title LIKE 'T%' AND

rating = 'G';

1. **How many of the country names (country) in the country table consist of 5** **characters?**

SELECT COUNT(country) FROM country

WHERE country LIKE '\_\_\_\_\_';

1. **How many of the city names in the city table end with the character 'R' or r?**

SELECT COUNT(city) FROM city

WHERE city ILIKE '%r';

**ORDER BY SYNTAX**

SELECT <columnName>, <columnName>, ...

FROM <tableName>

ORDER BY <columnName>, <columnName>, ... ASC|DESC;

ASC 🡺 INCREASING

DESC 🡺 DECREASING

SELECT \*

FROM film

WHERE title LIKE 'A%'

ORDER BY title ASC length DESC;

## **LIMIT**

SELECT \*

FROM film

WHERE title LIKE 'B%'

ORDER BY length DESC

LIMIT 10;

* Gives the 10 longest films.

## **OFFSET**

SELECT \*

FROM film

WHERE title LIKE 'B%'

ORDER BY length DESC

OFFSET 6

LIMIT 4;

* Skips the 6 longest film and gives other 4 film.

**Homework-5**

1. **List the 5 longest (length) films in the film table and the film title (title) ends with the 'n' character.**

SELECT \* FROM film

WHERE title LIKE '%n'

ORDER BY length DESC

LIMIT 5;

1. **List the shortest (length) second (6,7,8,9,10) 5 films (6,7,8,9,10) in the film table and the film title ends with the 'n' character.**

SELECT \* FROM film

WHERE title LIKE '%n'

ORDER BY length DESC

OFFSET 1

LIMIT 5;

1. **Sort the first 4 data, provided that store\_id is 1 in the descending order according to the last\_name column in the customer table.**

SELECT \* from customer

WHERE store\_id = 1

ORDER BY last\_name DESC

LIMIT 4;

# **Aggregate Functions - MIN, MAX, SUM, AVG**

SELECT AVG(length)

FROM film;

**Homework-6**

1. **What is the average of the values in the rental\_rate column in the film table?**

SELECT AVG(rental\_rate) FROM film;

1. **How many of the movies in the film table start with the character 'C'?**

SELECT COUNT(title) FROM film

WHERE title LIKE 'C%';

1. **Among the movies in the film table, how many minutes is the longest (length) film with a rental\_rate equal to 0.99?**

SELECT MAX(length) FROM film

WHERE rental\_rate = 0.99;

1. **How many different replacement\_cost values are there for the films longer than 150 minutes in the film table?**

SELECT COUNT(replacement\_cost) FROM film

WHERE length > 150 ;

## **GROUP BY**

SELECT rental\_rate, MAX(length)

FROM film

GROUP BY rental\_rate;

## **HAVING**

SELECT rental\_rate, COUNT(\*)

FROM film

GROUP BY rental\_rate

HAVING COUNT(\*) > 325;

**Homework-7**

1. Group the films in the film table according to their rating values.

SELECT rating FROM film

GROUP BY rating;

1. When we group the films in the film table according to the replacement\_cost column, list the replacement\_cost value with more than 50 films and the corresponding number of films.

SELECT replacement\_cost, COUNT(\*) FROM film

GROUP BY replacement\_cost

HAVING COUNT(\*) > 50;

1. What are the customer numbers corresponding to the store\_id values in the customer table?

SELECT store\_id, COUNT(\*) FROM customer

GROUP BY store\_id;

1. After grouping the city data in the city table according to the country\_id column, share the country\_id information with the highest number of cities and the number of cities.

SELECT country\_id, COUNT(\*) FROM city

GROUP BY country\_id

ORDER BY COUNT(\*) DESC

LIMIT 1; --maximum city

Text

Description automatically generated **CREATING TABLE**

**INSERT USES:**

Text

Description automatically generated

**OUTPUT :**

Graphical user interface, application

Description automatically generated

**COPYING TABLE SCHEMA :**

**A picture containing application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

* ***Copied schema but author2 table has not any data!***

**Graphical user interface, text, application, email

Description automatically generated**

**INSERTING author TO author2 :**

**Graphical user interface, application, Teams

Description automatically generated**

**COPYING TABLE WITH DATAS:**

**Graphical user interface, text, application

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence**

**DROP TABLE:**

**Graphical user interface, text, application

Description automatically generated**

**UPDATE SYNTAX**

UPDATE <tableName>

SET <columnName> = value,

<columnName> = value,

----

WHERE <condition>;

**DELETE SYNTAX**

DELETE FROM <tablo\_adı>

WHERE <koşul\_adı>;

**Homework-8**

Background pattern

Description automatically generated with low confidence

Text

Description automatically generated with medium confidence Table

Description automatically generated with medium confidence

Background pattern

Description automatically generated with low confidence

**PRIMARY KEY – FOREIGN KEY**

**A picture containing graphical user interface

Description automatically generated**

### ALTER

### The ALTER keyword is used to modify an existing table.

ALTER TABLE <table\_name>

ALTER COLUMN <column\_name>

SET --NOT NULL—(constraint);

**UNIQUE**

CREATE TABLE Employees (

---

email VARCHAR(100) UNIQUE,

----

);

**ALTER and UNIQUE**

ALTER TABLE <table\_name>

ADD UNIQUE <column\_name>

**CHECK**

CREATE TABLE Employees (

---

age INTEGER CHECK (age>=18)

----

);

**ALTER and CHECK**

ALTER TABLE <table\_name>

ADD CHECK (age>=18)

## **INNER JOIN = JOIN**

SELECT book.title, author.first\_name, author.last\_name

FROM book

JOIN author ON author.id = book.author\_id;

**Homework-9**

1. **Write the INNER JOIN query where we can see the city and country names in the city table and the country table together.**

Graphical user interface, text, application

Description automatically generated

1. **Write the INNER JOIN query where we can see the customer table and the payment\_id in the payment table and the first\_name and last\_name names in the customer table together.**

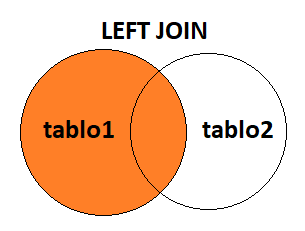
Table

Description automatically generated

1. **Write the INNER JOIN query where we can see the customer table and the rental\_id in the rental table and the first\_name and last\_name names in the customer table together.**

Graphical user interface, table

Description automatically generated



### LEFT JOIN

SELECT book.title, author.first\_name, author.last\_name

FROM book

LEFT JOIN author

ON author.id = book.author\_id;

### Right Join

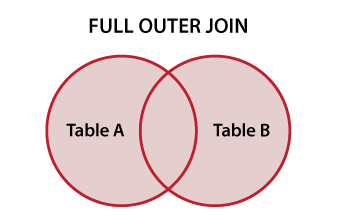
### RIGHT JOIN

SELECT book.title, author.first\_name, author.last\_name

FROM book

RIGHT JOIN author

ON author.id = book.author\_id;



# **FULL JOIN = FULL OUTER JOIN**

SELECT book.title, author.first\_name, author.last\_name

FROM book

FULL JOIN author

ON author.id = book.author\_id;

Graphical user interface, text, application

Description automatically generatedWe can use, FULL JOIN WITH WHERE CONDITION as INNER JOIN ! 😊

**Homework-10**

1. **Write the LEFT JOIN query where we can see the city and country names in the city table and the country table together.**

SELECT city.city, country.country FROM city

LEFT JOIN country ON city.country\_id = country.country\_id;

1. **Write the RIGHT JOIN query where we can see the customer table and the payment\_id in the payment table and the first\_name and last\_name names in the customer table together.**

Table

Description automatically generated

1. **Write the FULL JOIN query where we can see the customer table and the rental\_id in the rental table and the first\_name and last\_name names in the customer table together.**

Table

Description automatically generated

### UNION SYNTAX

SELECT <sütun\_adı>, <sütun\_adı>...

FROM <table1>

UNION

SELECT <sütun\_adı>, <sütun\_adı>...

FROM <table2>

**NOTE THAT :**

***UNION, gets unduplicated lines. But if we want to duplicate the same lines, we can use UNION ALL!***

**INTERSECT(ION)**

(

SELECT \*

FROM book

ORDER BY title

LIMIT 5

)

INTERSECT

(

SELECT \*

FROM book

ORDER BY page\_number DESC

LIMIT 5

);

🡺 INTERSECT keyword, gives to us intersection of 2 queries.

**EXCEPT**

(

SELECT \*

FROM book

ORDER BY title

LIMIT 5

)

EXCEPT

(

SELECT \*

FROM book

ORDER BY page\_number DESC

LIMIT 5

);

🡺 EXCEPT keyword, returns rows that are not in the intersection set.

**Homework-11**

1. **Let's sort all the data for the first\_name columns in the actor and customer tables.**

SELECT first\_name FROM actor

UNION

SELECT first\_name FROM customer;

1. **Let's sort the intersecting data for the first\_name columns in the actor and customer tables.**

SELECT first\_name FROM actor

INTERSECT

SELECT first\_name FROM customer;

1. **For the first\_name columns in the actor and customer tables, let's sort the data in the first table but not in the second table.**

SELECT first\_name FROM actor

EXCEPT

SELECT first\_name FROM customer;

1. **Let's also do the first 3 queries for repeating data.**
2. SELECT first\_name FROM actor

UNION ALL

SELECT first\_name FROM customer;

1. INTERSECT = INTERSECT ALL, so we can use same query.

Table

Description automatically generated with medium confidence