

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;
```

- 2- 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;
```

- 3- 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;
```

- 4- 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';
```

- 5- 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
```

- 2- 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');
```

- 3- 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';
```

- 2- 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';
```

- 3- 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%';
```

- 4- 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;
```

- 2- How many different data are there in the replacement cost column in the film table?

```
SELECT COUNT(DISTINCT replacement_cost) FROM film;
```

- 3- 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';
```

- 4- How many of the country names (country) in the country table consist of 5 characters?

```
SELECT COUNT(country) FROM country  
WHERE country LIKE '_____';
```

- 5- 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;
```

- 2- 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;
```

- 3- 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;
```

- 2- How many of the movies in the film table start with the character 'C'?

```
SELECT COUNT(title) FROM film
WHERE title LIKE 'C%';
```

- 3- 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;
```

- 4- 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;
```

- 2- 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;
```

- 3- 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;
```

- 4- 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
```

CREATING TABLE

```
--CREATE TABLE <table_name> (  
-- <column_name> <data_type> <constraint>,  
-- ...  
-- <column_name> <data_type> <constraint>  
--);  
  
CREATE TABLE author(  
  id SERIAL PRIMARY KEY, --numeric (auto increases)  
  first_name VARCHAR(50) NOT NULL,  
  last_name VARCHAR(50) NOT NULL,  
  email VARCHAR(100),  
  birthday DATE
```

INSERT USES:

```
1 INSERT INTO author (first_name, last_name, email, birthday)
2 VALUES
3     ('Gökem', 'Töre', 'gorkemtore1@gmail.com', '2002-09-24'),
4     ('Mustafa', 'Çetindağ', 'mchetindag@hotmail.com', '1988-12-24'),
5     ('Beyza', 'Töre', 'beyza@yandex.com', '2004-8-10'),
6     ('Selin', 'Güler', 'selinglr@gmail.com', '1999-02-20'),
7     ('Hasret', 'Yavuz', 'hasret02@hotmail.com', '1995-04-21');
```

OUTPUT :

Query Query History

```
1 SELECT * FROM author;
```

Data Output Messages Notifications

	id [PK] integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	birthday date
1	1	Gökem	Töre	gorkemtore1@gmail.com	2002-09-24
2	2	Mustafa	Çetindağ	mchetindag@hotmail.com	1988-12-24
3	3	Beyza	Töre	beyza@yandex.com	2004-08-10
4	4	Selin	Güler	selinglr@gmail.com	1999-02-20
5	5	Hasret	Yavuz	hasret02@hotmail.com	1995-04-21

COPYING TABLE SCHEMA :

```
1 --copy table
2 CREATE TABLE author2 (LIKE author);
```

Query Query History

```
1 SELECT * FROM author2;
```

Data Output Messages Notifications

	id integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	birthday date

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

INSERTING author TO author2 :

Query Query History

```
1 INSERT INTO author2
2 SELECT * FROM author
3 WHERE first_name = 'Görkem';
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 37 msec.

Query Query History

```
1 SELECT * FROM author2;
```

Data Output Messages Notifications

	id	first_name	last_name	email	birthday
	integer	character varying (50)	character varying (50)	character varying (100)	date
1	1	Görkem	Töre	gorkemtore1@gmail.com	2002-09-24

COPYING TABLE WITH DATAS:

Query Query History

```
1 CREATE TABLE author3 AS
2 SELECT * FROM author;
```

Data Output Messages Notifications

SELECT 5

Query returned successfully in 34 msec.

Query Query History

```
1 SELECT * FROM public.author3
2
```

Data Output Messages Notifications

	id	first_name	last_name	email	birthday
	integer	character varying (50)	character varying (50)	character varying (100)	date
1	1	Görkem	Töre	gorkemtore1@gmail.com	2002-09-24
2	2	Mustafa	Çetindağ	mcetindag@hotmail.com	1988-12-24
3	3	Beyza	Töre	beyza@yandex.com	2004-08-10
4	4	Selin	Güler	selinglr@gmail.com	1999-02-20
5	5	Hasret	Yavuz	hasret02@hotmail.com	1995-04-21

DROP TABLE:

Query Query History

```
1 --DROP TABLE author2;
2 --DROP TABLE IF EXISTS author2;
3 --both are usable
```

UPDATE SYNTAX

```
UPDATE <tableName>
SET <columnName> = value,
    <columnName> = value,
    ----
WHERE <condition>;
```

DELETE SYNTAX

```
DELETE FROM <tablo_adi>
WHERE <koşul_adi>;
```

Homework-8

Query Query History

```
1 CREATE TABLE employee(  
2     id INT,  
3     name VARCHAR(50),  
4     birthday DATE,  
5     email VARCHAR(100)  
6 );  
7  
8
```

```
9 insert into employee (id, name, birthday, email) values (1, 'Rubia', '2010-03-01', 'rraoux0@homestead.com');  
10 insert into employee (id, name, birthday, email) values (2, 'Janessa', '1930-06-25', 'jbaulk1@domainmarket.com');  
11 insert into employee (id, name, birthday, email) values (3, 'Cecilia', '1958-04-08', 'cdeandisie2@cloudflare.com');  
12 insert into employee (id, name, birthday, email) values (4, 'Mathilda', '1906-08-05', 'mmcnally3@fc2.com');  
13 insert into employee (id, name, birthday, email) values (5, 'Editha', '1960-05-18', 'edomney4@phpbb.com');  
14 insert into employee (id, name, birthday, email) values (6, 'Tallie', '1952-06-04', 'tswale5@wp.com');  
15 insert into employee (id, name, birthday, email) values (7, 'Jay', '1922-07-10', 'jrehn6@cafepress.com');  
16 insert into employee (id, name, birthday, email) values (8, 'Lucine', '1952-03-12', 'lfinnimore7@com.com');  
17 insert into employee (id, name, birthday, email) values (9, 'Lavena', '1934-05-12', 'lsmye8@reuters.com');  
18 insert into employee (id, name, birthday, email) values (10, 'Caterina', '1945-07-26', 'cscarrott9@360.cn');  
19 insert into employee (id, name, birthday, email) values (11, 'Thaddeus', '1959-04-16', 'tseagea@usda.gov');  
20 insert into employee (id, name, birthday, email) values (12, 'Wallache', '2017-03-28', 'wtommeib@aboutads.info');  
21 insert into employee (id, name, birthday, email) values (13, 'Yetta', '2016-08-12', 'ymawdittc@wunderground.com');  
22 insert into employee (id, name, birthday, email) values (14, 'Bren', '1903-11-02', 'bmuzzollod@ovh.net');  
23 insert into employee (id, name, birthday, email) values (15, 'Pierrette', '1972-09-10', 'pmariellee@edublogs.org');  
24 insert into employee (id, name, birthday, email) values (16, 'Shelly', '1925-10-06', 'scampkinf@mac.com');  
25 insert into employee (id, name, birthday, email) values (17, 'Corbet', '1936-03-08', 'creynaldsg@unicef.org');  
26 insert into employee (id, name, birthday, email) values (18, 'Virginia', '2004-11-20', 'vmathieuh@phoca.cz');  
27 insert into employee (id, name, birthday, email) values (19, 'Andromache', '1922-08-27', 'adenyukhini@baidu.com');  
28 insert into employee (id, name, birthday, email) values (20, 'Brucie', '1965-06-11', 'bnorthcottj@whitehouse.gov');  
29 insert into employee (id, name, birthday, email) values (21, 'Barrie', '1919-12-24', 'bminerdok@fema.gov');  
30 insert into employee (id, name, birthday, email) values (22, 'Ludovico', '1960-02-20', 'lmussettil@bing.com');  
31 insert into employee (id, name, birthday, email) values (23, 'Gilbertina', '1934-03-01', 'gdeschellem@cargocollective.com');  
32 insert into employee (id, name, birthday, email) values (24, 'Sylvester', '2019-11-07', 'smayburyne@fastcompany.com');  
33 insert into employee (id, name, birthday, email) values (25, 'Joshua', '2019-11-09', 'jmcartano@multiply.com');  
34 insert into employee (id, name, birthday, email) values (26, 'Amery', '2003-08-17', 'azanetellop@phpbb.com');  
35 insert into employee (id, name, birthday, email) values (27, 'Ches', '1945-06-05', 'cperrycostq@cafepress.com');  
36 insert into employee (id, name, birthday, email) values (28, 'Daniella', '2011-08-09', 'dwillr@indiegogo.com');  
37 insert into employee (id, name, birthday, email) values (29, 'Genovera', '2001-06-01', 'gbrittins@sina.com.cn');  
38 insert into employee (id, name, birthday, email) values (30, 'Angelle', '1935-01-31', 'asillist@hud.gov');  
39 insert into employee (id, name, birthday, email) values (31, 'Elliot', '1997-08-09', 'egaliau@blogspot.com');  
40 insert into employee (id, name, birthday, email) values (32, 'Cory', '1974-11-21', 'ckoubekv@mapquest.com');  
41 insert into employee (id, name, birthday, email) values (33, 'Fawne', '1988-09-01', 'fmccritchiew@msu.edu');  
42 insert into employee (id, name, birthday, email) values (34, 'Lazare', '2000-04-19', 'lstealyx@blogs.com');  
43 insert into employee (id, name, birthday, email) values (35, 'Guendolen', '1949-09-07', 'gdruhany@vistaprint.com');  
44 insert into employee (id, name, birthday, email) values (36, 'Nikaniki', '1901-05-29', 'nitzhaiekz@angelfire.com');  
45 insert into employee (id, name, birthday, email) values (37, 'Ottilie', '2009-09-02', 'oshepherdson10@wikia.com');  
46 insert into employee (id, name, birthday, email) values (38, 'Colet', '1925-02-25', 'cmartignon11@sogou.com');  
47 insert into employee (id, name, birthday, email) values (39, 'Celina', '1932-09-07', 'cwarcop12@spotify.com');  
48 insert into employee (id, name, birthday, email) values (40, 'Karl', '1902-11-06', 'kpointer13@msu.edu');  
49 insert into employee (id, name, birthday, email) values (41, 'Jinny', '1938-08-25', 'jcaslake14@examiner.com');  
50 insert into employee (id, name, birthday, email) values (42, 'Mariana', '2018-12-26', 'msco15@bizjournals.com');  
51 insert into employee (id, name, birthday, email) values (43, 'Sydney', '1985-09-21', 'sfraill16@woothemes.com');  
52 insert into employee (id, name, birthday, email) values (44, 'Roderick', '1999-06-20', 'rdunn17@miitbeian.gov.cn');  
53 insert into employee (id, name, birthday, email) values (45, 'Almira', '1915-02-16', 'alaughren18@vimeo.com');  
54 insert into employee (id, name, birthday, email) values (46, 'Rora', '1983-02-27', 'rfacer19@amazon.com');  
55 insert into employee (id, name, birthday, email) values (47, 'Brita', '1997-12-07', 'bgainsford1a@nih.gov');  
56 insert into employee (id, name, birthday, email) values (48, 'Blinny', '1926-08-22', 'bfashion1b@usgs.gov');  
57 insert into employee (id, name, birthday, email) values (49, 'Anna-maria', '1992-07-13', 'ahindmore1c@comsenz.com');  
58 insert into employee (id, name, birthday, email) values (50, 'Moe', '1991-08-10', 'mlosemann1d@tuttocitta.it');
```

```
60 SELECT * FROM employee;  
61  
62 UPDATE employee  
63 SET name = 'Mia'  
64 WHERE id = 50;  
65  
66 SELECT * FROM employee WHERE id = 50;  
67  
68  
69 UPDATE employee  
70 SET birthday = '1990-05-30'  
71 WHERE id = 48;  
72  
73 DELETE FROM employee  
74 WHERE id = 1;  
75  
76 SELECT * FROM employee;
```

PRIMARY KEY – FOREIGN KEY



```
Query  Query History
1  CREATE TABLE book (
2      id SERIAL PRIMARY KEY,
3      titlte VARCHAR(100) NOT NULL,
4      page_number INTEGER NOT NULL,
5      author_id INTEGER REFERENCES author(id)
6  );
7
8  SELECT * FROM book;
```

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.

Query		Query History	
1	SELECT	city, country	FROM city
2	JOIN	country	ON city.country_id = country.country_id;

Data Output		Messages	Notifications
			
			
	city	country	
	character varying (50)	character varying (50)	
1	A Corua (La Corua)	Spain	
2	Abha	Saudi Arabia	
3	Abu Dhabi	United Arab Emirates	
4	Acua	Mexico	
5	Adana	Turkey	

- 2- 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.

Query		Query History	
1	SELECT	payment_id, first_name, last_name	FROM customer
2	INNER JOIN	payment	ON customer.customer_id = payment.customer_id;

Data Output		Messages	Notifications
			
			
	payment_id	first_name	last_name
	integer	character varying (45)	character varying (45)
1	17503	Peter	Menard
2	17504	Peter	Menard
3	17505	Peter	Menard
4	17506	Peter	Menard
5	17507	Peter	Menard
6	17508	Peter	Menard
7	17509	Harold	Martino

- 3- 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.

Query

Query History

1

SELECT rental_id, first_name, last_name **FROM** rental

2

INNER JOIN customer **ON** customer.customer_id = rental.customer_id;

Data Output

Messages

Notifications

rental_id

integer

first_name

character varying (45)

last_name

character varying (45)

1

2

Tommy

Collazo

2

3

Manuel

Murrell

3

4

Andrew

Purdy

4

5

Delores

Hansen

5

6

Nelson

Christenson

6

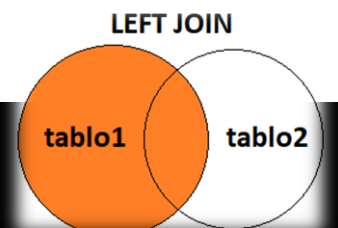
7

Cassandra

Walters

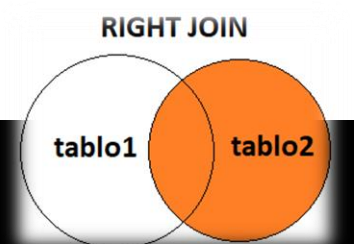
LEFT JOIN

```
SELECT book.title, author.first_name, author.last_name
FROM book
LEFT JOIN author
ON author.id = book.author_id;
```



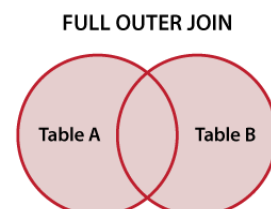
RIGHT JOIN

```
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;
```



We can use, FULL JOIN WITH WHERE CONDITION as INNER JOIN ! 😊

```
bookstore/postgres@PostgreSQL 13
Query Editor  Query History
1  --SELECT * FROM author
2  --FULL OUTER JOIN book ON author.id = book.author_id
3  --WHERE (book.id IS NOT NULL AND author.id IS NOT NULL);
4
5  SELECT * FROM book
6  INNER JOIN author ON author.id = book.author_id;
```

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;
```

- 2- 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.

Query		Query History	
1	SELECT	payment.payment_id , customer.first_name, customer.last_name	FROM payment
2	RIGHT JOIN	customer	ON customer.customer_id = payment.customer_id;
Data Output			
Messages			
Notifications			
	payment_id integer	first_name character varying (45)	last_name character varying (45)
1	17503	Peter	Menard
2	17504	Peter	Menard
3	17505	Peter	Menard
4	17506	Peter	Menard
5	17507	Peter	Menard
6	17508	Peter	Menard
7	17509	Harold	Martino
8	17510	Harold	Martino
9	17511	Harold	Martino
10	17512	Douglas	Graf
11	17513	Douglas	Graf
12	17514	Douglas	Graf
13	17515	Douglas	Graf
14	17516	Douglas	Graf
15	17517	Douglas	Graf
16	17518	Douglas	Graf
17	17519	Henry	Billingsley

- 3- 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.

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT rental.rental_id, customer.first_name, customer.last_name FROM rental
2 FULL JOIN customer ON customer.customer_id = rental.customer_id;
```

The results are displayed in a table with the following columns: rental_id (integer), first_name (character varying (45)), and last_name (character varying (45)).

	rental_id integer	first_name character varying (45)	last_name character varying (45)
17	18	Ruth	Martinez
18	19	Ronnie	Ricketts
19	20	Roberta	Harper
20	21	Craig	Morrell
21	22	Raul	Fortier
22	23	Barry	Lovelace
23	24	Juan	Fraley
24	25	Pamela	Baker
25	26	Billy	Poulin
26	27	Robert	Baughman
27	28	Constance	Reid
28	29	Marie	Turner
29	30	Ray	Houle
30	31	Fred	Wheat
31	32	Joy	George
32	33	Kay	Caldwell
33	34	Freddie	Duggan

Total rows: 1000 of 16044 Query complete 00:00:00.061

UNION SYNTAX

```
SELECT <sütun_ad1>, <sütun_ad1>...
FROM <table1>
UNION
SELECT <sütun_ad1>, <sütun_ad1>...
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;
```

- 2- 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;
```

- 3- 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;
```


4- Let's also do the first 3 queries for repeating data.

1-

```
SELECT first_name FROM actor
UNION ALL
SELECT first_name FROM customer;
```

2- INTERSECT = INTERSECT ALL, so we can use same query.

3-

Query		Query History	
1	SELECT	first_name	FROM actor
2	EXCEPT ALL		
3	SELECT	first_name	FROM customer;
Data Output		Messages	
		Notifications	
	first_name	character varying (45) 	
1	Bela		
2	Christian		
3	Christian		
4	Kenneth		
5	Kenneth		
6	Kenneth		
7	Olympia		
8	Cameron		
9	Cameron		
10	Cameron		
11	Uma		
12	Sylvester		
13	Spencer		
14	Spencer		