
Strive
SCHOOL

SELECT

What it does?

/ SELECT statements allow you you query data from tables.

/ SELECT statement can be combined with various SQL Keyword for re-shaping data.

/ You can do operations like,filter,order,join,projection,exclusion etc…

Syntax

```
SELECT
    select_list
FROM
    table_name;
```

Select all columns

```
SELECT * FROM customer;
```

Example

dvdrental/postgres@strive

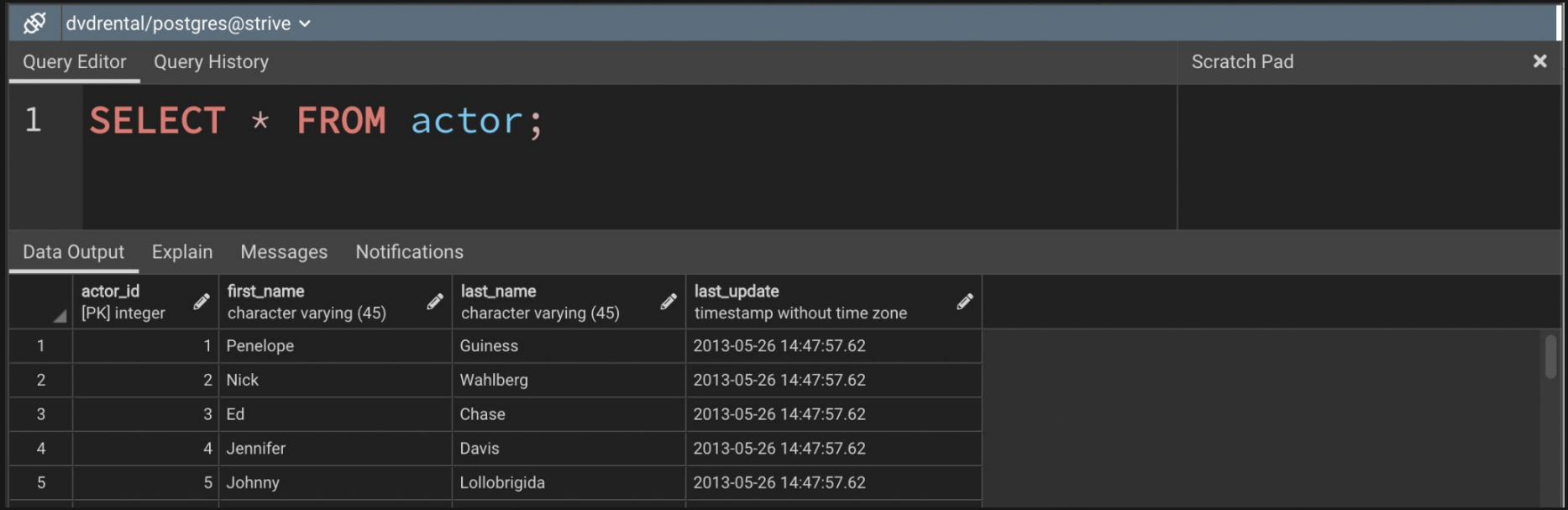
Query Editor | Query History | Scratch Pad

```
1  SELECT * FROM customer;
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp w
1	524	1	Jared	Ely	jared.ely@sakilacustomer.org	530	true	2006-02-14	2013-05-2
2	1	1	Mary	Smith	mary.smith@sakilacustomer...	5	true	2006-02-14	2013-05-2
3	2	1	Patricia	Johnson	patricia.johnson@sakilacust...	6	true	2006-02-14	2013-05-2
4	3	1	Linda	Williams	linda.williams@sakilacusto...	7	true	2006-02-14	2013-05-2

Example



	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	Penelope	Guiness	2013-05-26 14:47:57.62
2	2	Nick	Wahlberg	2013-05-26 14:47:57.62
3	3	Ed	Chase	2013-05-26 14:47:57.62
4	4	Jennifer	Davis	2013-05-26 14:47:57.62
5	5	Johnny	Lollobrigida	2013-05-26 14:47:57.62

Select specific columns

```
SELECT
    first_name,
    last_name,
    email
FROM
    customer;
```

Example

```
1  SELECT first_name,last_name FROM actor;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Penelope	Guiness
2	Nick	Wahlberg
3	Ed	Chase
4	Jennifer	Davis
5	Johnny	Lollobrigida

Example

```
1  SELECT first_name,last_name FROM customer;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Jared	Ely
2	Mary	Smith
3	Patricia	Johnson
4	Linda	Williams
5	Barbara	Jones

COLUMN ALIASES

What it does?

/ A column alias allows you to change column name temporarily on the fly.

Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      first_name,
3      last_name AS surname
4  FROM customer;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	surname character varying (45)
1	Jared	Ely
2	Mary	Smith
3	Patricia	Johnson
4	Linda	Williams

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      title AS name,
3      release_year AS year
4  FROM film;
```

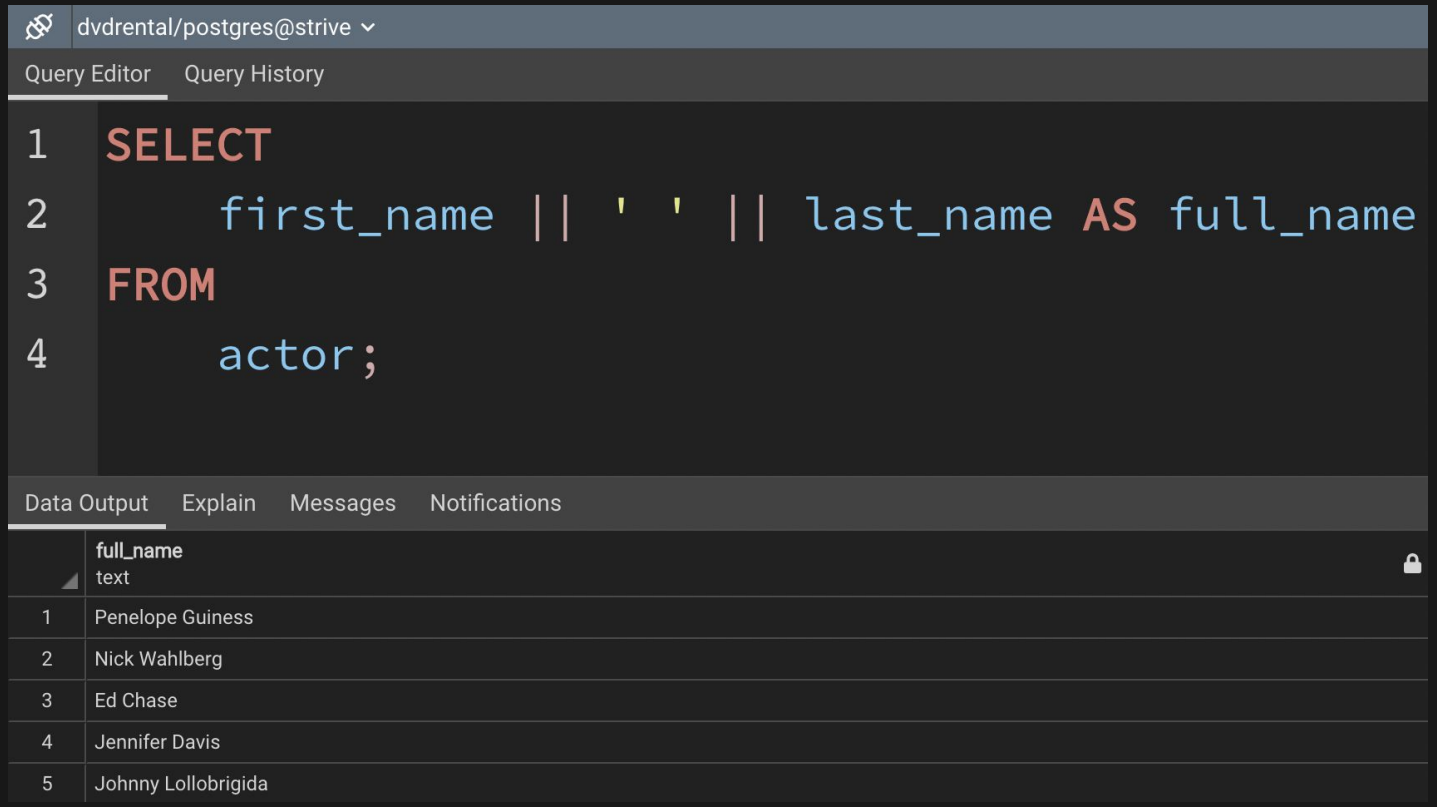
Data Output | Explain | Messages | Notifications

	name character varying (255)	year integer
1	Chamber Italian	2006
2	Grosse Wonderful	2006
3	Airport Pollock	2006
4	Bright Encounters	2006
5	Academy Dinosaur	2006

Syntax

```
SELECT
    first_name || ' ' || last_name AS full_name
FROM
    customer;
```

Alias for expression



Exercise 1

- SELECT all columns from film table.
- SELECT district,phone,postal_code from “address” table.
- SELECT address,district,postal_code and concat them and get as “full_address”.

Exercise 1

```
-- SELECT * FROM film; -- exercise 1

-- SELECT district,phone,postal_code FROM address; -- exercise 2

-- SELECT district || ' '  || phone || ' ' || postal_code  AS full_address FROM
address; -- exercise 3
```

WHERE

What it does?

/ Using only SELECT statement returns you all rows from a table. To select rows that satisfies specific condition you need to use WHERE clause.

/ WHERE clause must be used after the FROM clause of SELECT statement.

/ Condition must be evaluated to true or false values. It can be single boolean value or combination of boolean value.

Syntax

```
SELECT select_list
FROM table_name
WHERE condition
```

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<> or !=	Not equal

Operator	Description
AND	Logical operator AND
OR	Logical operator OR
IN	Return true if a value matches any value in a list
BETWEEN	Return true if a value is between a range of values
LIKE	Return true if a value matches a pattern
IS NULL	Return true if a value is NULL

Example

dvdrental/postgres@strive

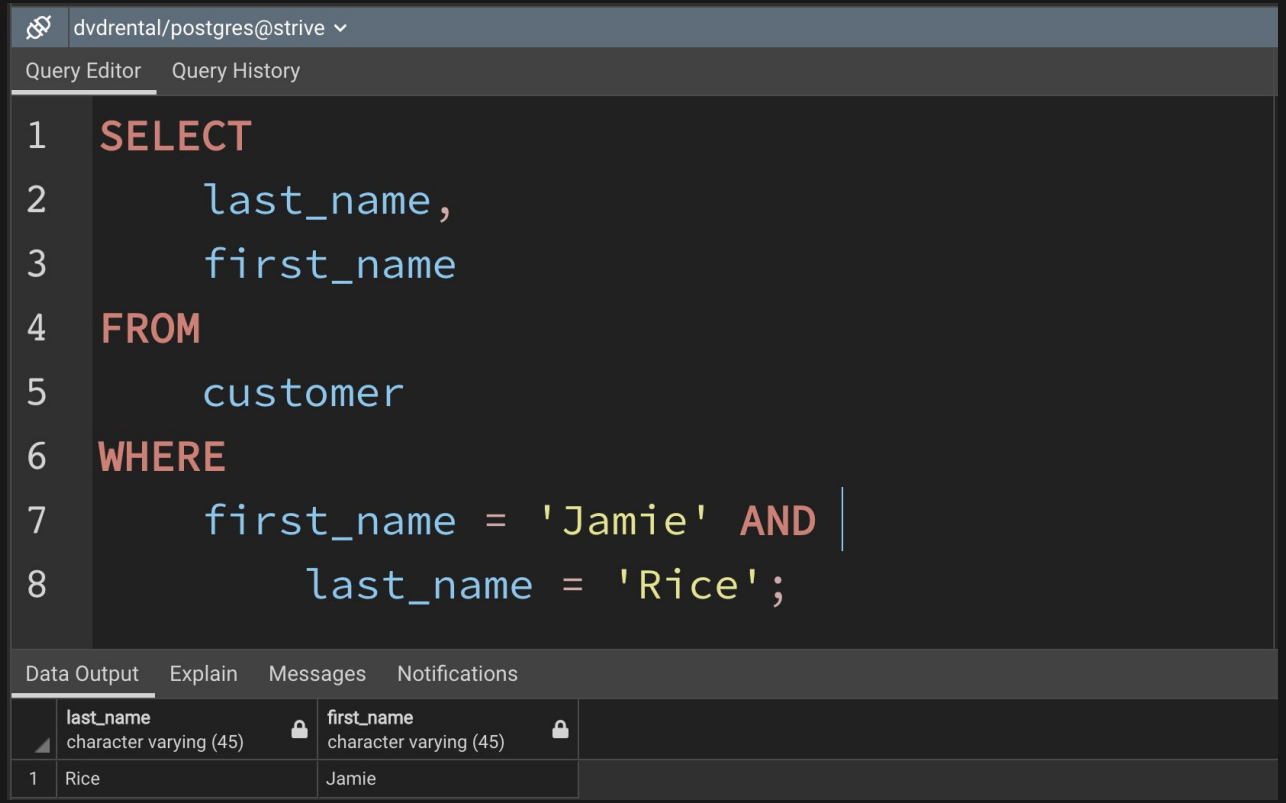
Query Editor | Query History

```
1  SELECT
2      last_name,
3      first_name
4  FROM
5      customer
6  WHERE
7      first_name = 'Jamie';
```

Data Output | Explain | Messages | Notifications

	last_name character varying (45)	first_name character varying (45)
1	Rice	Jamie
2	Waugh	Jamie

Example


dvdrental/postgres@strive
Query Editor
Query History
1 SELECT
2 last_name,
3 first_name
4 FROM
5 customer
6 WHERE
7 first_name = 'Jamie' AND
8 last_name = 'Rice';
Data Output
Explain
Messages
Notifications
last_name
character varying (45)
first_name
character varying (45)
1
Rice
Jamie

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT * FROM payment
2      WHERE amount > 5
```

Data Output | Explain | Messages | Notifications

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp without time zone
1	17503	341	2	1520	7.99	2007-02-15 22:25:46.996577
2	17505	341	1	1849	7.99	2007-02-16 22:41:45.996577
3	17507	341	2	3130	7.99	2007-02-20 17:31:48.996577
4	17508	341	1	3382	5.99	2007-02-21 12:33:49.996577
5	17509	342	2	2190	5.99	2007-02-17 23:58:17.996577

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT * FROM payment
2      WHERE amount < 1
```

Data Output | Explain | Messages | Notifications

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp without time zone
1	17514	343	2	1879	0.99	2007-02-17 01:26:00.996577
2	17515	343	2	1922	0.99	2007-02-17 04:32:51.996577
3	17518	343	1	3407	0.99	2007-02-21 14:42:28.996577
4	17521	344	1	1731	0.99	2007-02-16 14:00:38.996577
5	17522	345	2	1210	0.99	2007-02-15 01:26:17.996577

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT * FROM payment
2      WHERE amount <= 9
```

Data Output | Explain | Messages | Notifications

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp without time zone
1	17503	341	2	1520	7.99	2007-02-15 22:25:46.996577
2	17504	341	1	1778	1.99	2007-02-16 17:23:14.996577
3	17505	341	1	1849	7.99	2007-02-16 22:41:45.996577
4	17506	341	2	2829	2.99	2007-02-19 19:39:56.996577
5	17507	341	2	3130	7.99	2007-02-20 17:31:48.996577

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch Pad

```
1  SELECT * FROM address
2      WHERE district != 'Alberta';
```

Data Output | Explain | Messages | Notifications

	address_id [PK] integer	address character varying (50)	address2 character varying (50)	district character varying (20)	city_id smallint	postal_code character varying (10)	phone character varying (20)
1	2	28 MySQL Boulevard	[null]	QLD	576		
2	4	1411 Lillydale Drive	[null]	QLD	576		6172235589
3	5	1913 Hanoi Way		Nagasaki	463	35200	28303384290
4	6	1121 Loja Avenue		California	449	17886	838635286649
5	7	692 Joliet Street		Attika	38	83579	448477190408

IN

What it does?

/ IN operator in the WHERE clause to check if a value matches value in a list of values.

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      first_name,
3      last_name
4  FROM
5      customer
6  WHERE
7      first_name IN ('Ann','Anne','Annie');
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Ann	Evans
2	Anne	Powell
3	Annie	Russell

Usage with sub-query

Query Editor | Query History

```sql
SELECT
    customer_id,
    first_name,
    last_name
FROM
    customer
WHERE
    customer_id IN (SELECT customer_id FROM rental);
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	first_name character varying (45)	last_name character varying (45)
1	524	Jared	Ely
2	1	Mary	Smith
3	2	Patricia	Johnson
4	3	Linda	Williams
5	4	Barbara	Jones

LIKE & ILIKE

What it does?

/ Returns you columns that match with specific pattern

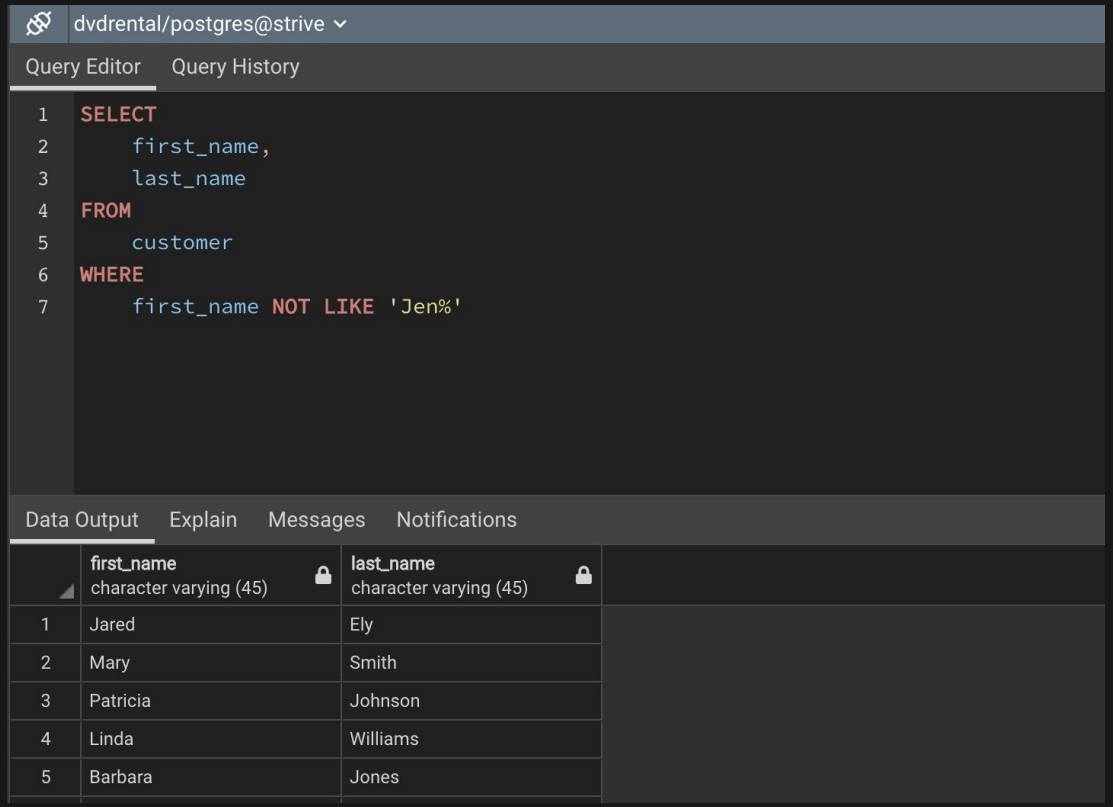
Syntax

```
SELECT
        first_name,
        last_name
FROM
        customer
WHERE
        first_name LIKE '%er%'
```

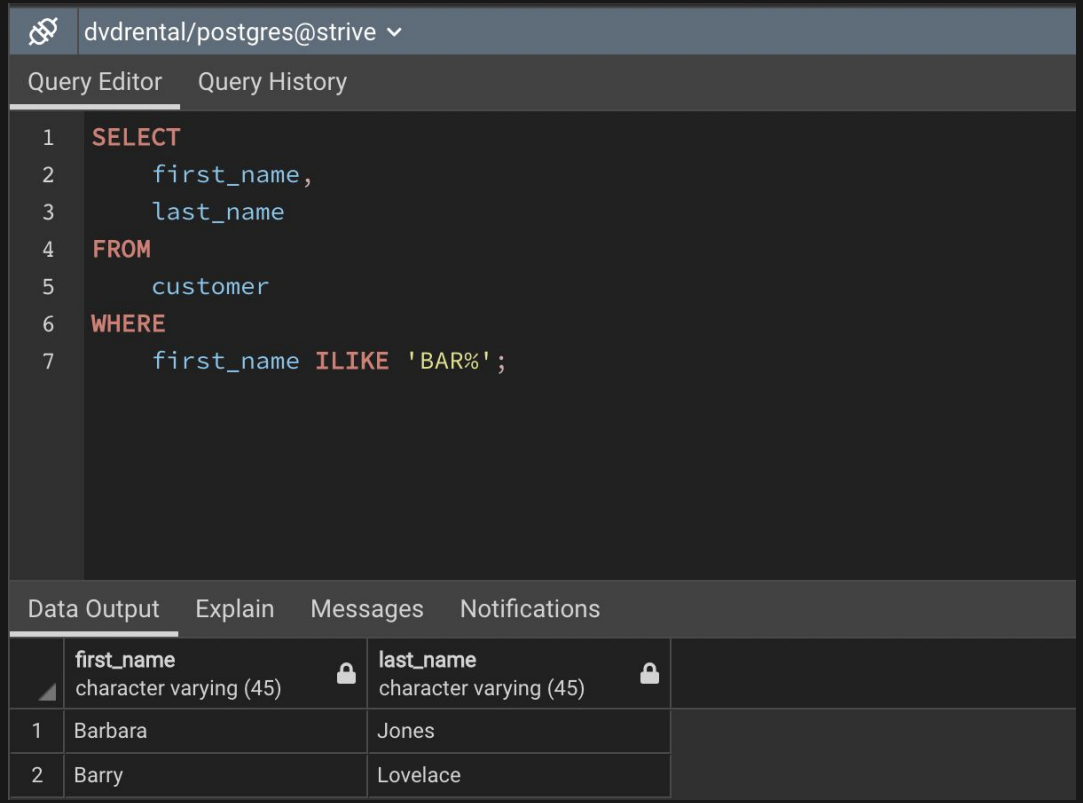
Like Patterns

Expression	Meaning
LIKE `'A%'`	Begins with A
LIKE `'%a'`	Ends with a
LIKE `'%a%'`	Contains a
LIKE `'Ab_'`	Begins with Ab and is followed by at most one character e.g., Abc, Abd...
LIKE `'_ab'`	Ends with ab and is preceded by at most one character e.g., fab
LIKE `'%are_'`	Contains `are`, begins with any number of characters and ends with at most one character
LIKE `'_are%'`	Contains `are`, begins with at most one character and ends with any number of characters

Example


dvdrental/postgres@strive
Query Editor
Query History
1 SELECT
2 first_name,
3 last_name
4 FROM
5 customer
6 WHERE
7 first_name NOT LIKE 'Jen%'
Data Output
Explain
Messages
Notifications
first_name
character varying (45)
last_name
character varying (45)
1 Jared Ely
2 Mary Smith
3 Patricia Johnson
4 Linda Williams
5 Barbara Jones

Example


dvdrental/postgres@strive
Query Editor
Query History
1 SELECT
2 first_name,
3 last_name
4 FROM
5 customer
6 WHERE
7 first_name ILIKE 'BAR%';
Data Output
Explain
Messages
Notifications
first_name
character varying (45)
last_name
character varying (45)
1 Barbara Jones
2 Barry Lovelace

Bonus

Operator	Equivalent
~~	LIKE
~~*	ILIKE
!~~	NOT LIKE
!~~*	NOT ILIKE

BETWEEN

What it does?

- BETWEEN operator used to find a value in BETWEEN low and high values.

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```
1  SELECT
2      customer_id,
3      payment_id,
4      amount
5  FROM
6      payment
7  WHERE
8      amount BETWEEN 8 AND 9;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	payment_id [PK] integer	amount numeric (5,2)
1	343	17517	8.99
2	347	17529	8.99
3	347	17532	8.99
4	348	17535	8.99
5	349	17540	8.99

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
SELECT
    customer_id,
    payment_id,
    amount
FROM
    payment
WHERE
    amount NOT BETWEEN 8 AND 9;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	payment_id [PK] integer	amount numeric (5,2)
1	341	17503	7.99
2	341	17504	1.99
3	341	17505	7.99
4	341	17506	2.99
5	341	17507	7.99

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
SELECT
    customer_id,
    payment_id,
    amount,
    payment_date
FROM
    payment
WHERE
    payment_date BETWEEN '2007-02-07' AND '2007-02-15';
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	payment_id [PK] integer	amount numeric (5,2)	payment_date timestamp without time zone
1	368	17610	0.99	2007-02-14 23:25:11.996577
2	370	17617	6.99	2007-02-14 23:33:58.996577
3	402	17743	4.99	2007-02-14 23:53:34.996577
4	416	17793	2.99	2007-02-14 21:21:59.996577
5	432	17854	5.99	2007-02-14 23:07:27.996577

ORDER BY

What it does?

/ When you query your data from a table with SELECT it returns rows with insertion order. To sort the rows of result set you need to use ORDER BY clause.

/ Default sorting direction is ASC (ascending) so you can omit ASC after ORDER BY if you want ascending order

Syntax

```
SELECT
        select_list
FROM
        table_name
ORDER BY
        sort_expression1 [ASC | DESC],
        ...
        sort_expressionN [ASC | DESC];
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      first_name,
3      last_name
4  FROM
5      customer
6  ORDER BY
7      first_name ASC;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Aaron	Selby
2	Adam	Gooch
3	Adrian	Clary
4	Agnes	Bishop
5	Alan	Kahn

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      first_name,
3      last_name
4  FROM
5      customer
6  ORDER BY
7      first_name;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Aaron	Selby
2	Adam	Gooch
3	Adrian	Clary
4	Agnes	Bishop
5	Alan	Kahn

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
SELECT
    first_name,
    last_name
FROM
    customer
ORDER BY
    first_name DESC;
```

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)
1	Zachary	Hite
2	Yvonne	Watkins
3	Yolanda	Weaver
4	Wilma	Richards
5	Willie	Markham

LIMIT & OFFSET

What it does?

/ LIMIT is an optional clause of the SELECT statement that constraints the number of returned by the query.

/ OFFSET will skip number of specified after OFFSET clause.

Syntax

```
SELECT select_list
FROM table_name
LIMIT row_count OFFSET row_to_skip;
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
SELECT
    film_id,
    title,
    release_year
FROM
    film
ORDER BY
    film_id
LIMIT 5;
```

Data Output | Explain | Messages | Notifications

	film_id [PK] integer	title character varying (255)	release_year integer
1	1	Academy Dinosaur	2006
2	2	Ace Goldfinger	2006
3	3	Adaptation Holes	2006
4	4	Affair Prejudice	2006
5	5	African Egg	2006

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  SELECT
2      film_id,
3      title,
4      release_year
5  FROM
6      film
7  ORDER BY
8      film_id
9  LIMIT 4 OFFSET 3;
```

Data Output | Explain | Messages | Notifications

	film_id [PK] integer	title character varying (255)	release_year integer
1	4	Affair Prejudice	2006
2	5	African Egg	2006
3	6	Agent Truman	2006
4	7	Airplane Sierra	2006

Exercise 2

- SELECT customers that their name starts with “J”
- SELECT payments that amount value is between 3 and 5
- SELECT payments that happened between 2007-02-15 and 2007-02-20
- SELECT movies that in inventory. (hint : use sub-query)
- SELECT payments that amounts between 4-6 order desc by payment_date
- SELECT first 5 customers ORDER by name DESC
- SELECT first 5 customers ORDER by name ASC but skip first 10

INSERT

What it does?

/ INSERT statement allows you to add new row to table.

Syntax

```
INSERT INTO table_name(column1, column2, …)
VALUES (value1, value2, …);
```

Syntax

```
INSERT INTO table_name (column_list)
VALUES
    (value_list_1),
    (value_list_2),
    ...
    (value_list_n);
```

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```
1  INSERT
2      INTO
3      customer(
4          store_id,
5          first_name,
6          last_name,
7          email,
8          address_id
9      )
10     VALUES(1,
11            'ubeyt',
12            'demir',
13            'ubeytdemir4se@gmail.com',
14            1) RETURNING *;
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date	last_update timestamp w
1	601	1	ubeyt	demir	ubeytdemir4se@gmail.com	1	true	2021-12-13	2021-12-13 1

UPDATE

What it does?

/ UPDATE statement allows you to update existing record.

Syntax

```
UPDATE table_name
SET column1 = value1,
    column2 = value2,
    ...
WHERE condition;
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  UPDATE customer
2      SET
3          email = 'ubeytdemir.dev@gmail.com',
4          address_id=2
5      WHERE customer_id=601
6          RETURNING *;
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)	email character varying (50)	address_id smallint	activebool boolean	create_date date
1	601	1	ubeyt	demir	ubeytdemir.dev@gmail.com	2	true	2021-12-13

DELETE

What it does?

/ DELETE statement allows you to delete existing record.

Syntax

```
DELETE FROM table_name
WHERE condition;
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```
1  DELETE FROM
2      customer
3      WHERE customer_id=601;
```

Exercise 3

- Insert 5 customer to database.
- Edit second customer’s name
- Delete last customer

GROUP BY

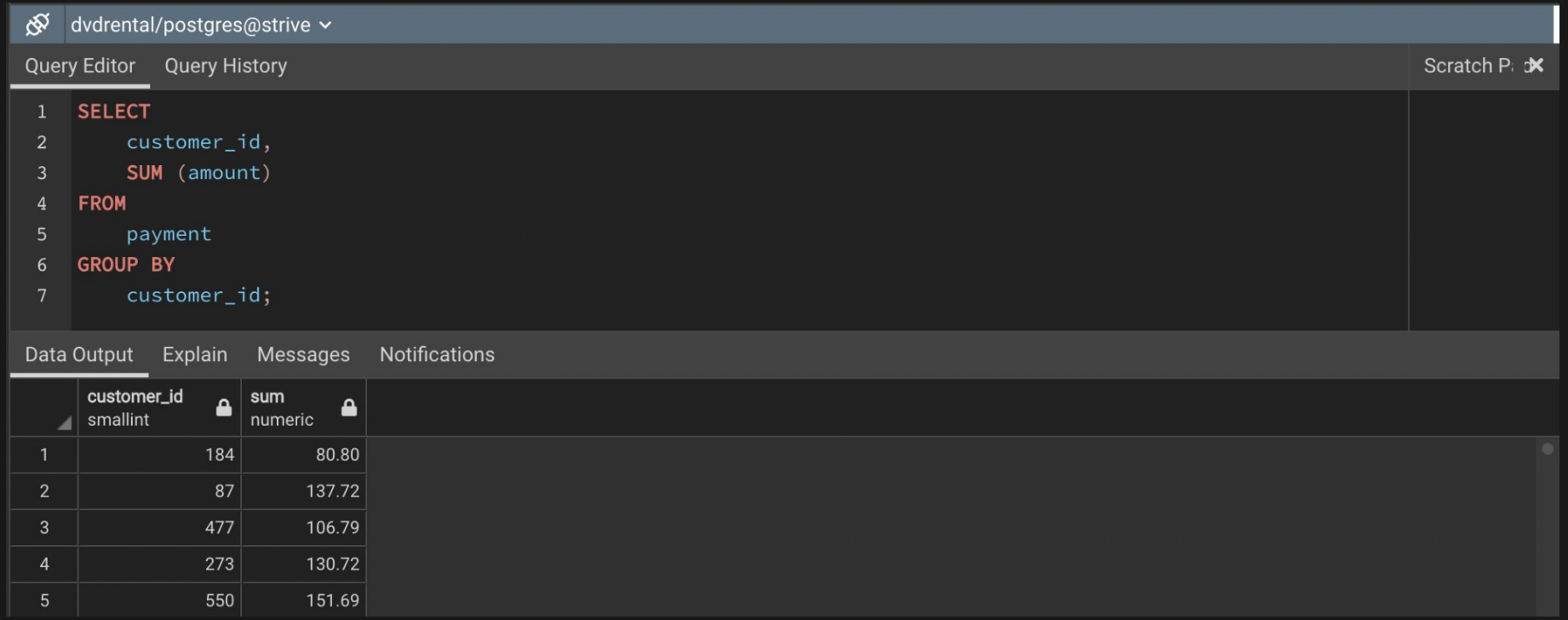
What it does?

/ GROUP BY clause divides the rows returned from the SELECT statement into groups. For each group you can apply an aggregate function such as SUM(), COUNT().

Syntax

```
SELECT
   column_1,
   column_2,
   ...,
   aggregate_function(column_3)
FROM
   table_name
GROUP BY
   column_1,
   column_2,
   ...;
```

Example


dvdrental/postgres@strive
Query Editor
Query History
Scratch P
1 SELECT
2 customer_id,
3 SUM (amount)
4 FROM
5 payment
6 GROUP BY
7 customer_id;
Data Output
Explain
Messages
Notifications
customer_id
smallint
sum
numeric
1 184 80.80
2 87 137.72
3 477 106.79
4 273 130.72
5 550 151.69

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```
1  SELECT
2      customer_id,
3      SUM (amount) as total_paid
4  FROM
5      payment
6  GROUP BY
7      customer_id
8  ORDER BY
9      total_paid DESC;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	total_paid numeric
1	148	211.55
2	526	208.58
3	178	194.61
4	137	191.62
5	144	189.60

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```
1  SELECT
2      customer_id,
3      SUM (amount) as total_paid
4  FROM
5      payment
6  GROUP BY
7      customer_id
8  ORDER BY
9      total_paid DESC;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	total_paid numeric
1	148	211.55
2	526	208.58
3	178	194.61
4	137	191.62
5	144	189.60

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
-- Find total sale for each staff.
SELECT
    staff_id,
    COUNT (payment_id) AS total_sale
FROM
    payment
GROUP BY
    staff_id;
```

Data Output | Explain | Messages | Notifications

	staff_id smallint	total_sale bigint
1	1	7292
2	2	7304

Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```sql
-- find total sales made by staff to for each customer
SELECT
    customer_id,
    staff_id,
    SUM(amount)  AS total_amount_of_sale
FROM
    payment
GROUP BY
    staff_id,
    customer_id
ORDER BY
    customer_id;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	staff_id smallint	total_amount_of_sale numeric
1	1	2	53.85
2	1	1	60.85
3	2	2	67.88
4	2	1	55.86
5	3	1	59.88

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
-- find total sales groupped by day
SELECT
    SUM(amount) AS total_amount,
    DATE(payment_date) paid_date
FROM
    payment
GROUP BY
    DATE(payment_date);
```

HAVING

What it does?

/ WHERE clause was filtering rows based on specific condition. HAVING is filtering groups of rows according to specific condition.

Syntax

```
SELECT
        column1,
        aggregate_function (column2)
FROM
        table_name
GROUP BY
        column1
HAVING
        condition;
```

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
 -- find total sales groupped by customer_id and filter if amount is > 200
SELECT
    customer_id,
    SUM (amount)
FROM
    payment
GROUP BY
    customer_id
HAVING
    SUM (amount) > 200;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint	sum numeric
1	526	208.58
2	148	211.55

Example

dvdrental/postgres@strive

Query Editor | Query History

```sql
-- find total number of customers groupped by store_id and show if total customer is more than 300
SELECT
    store_id,
    COUNT (customer_id)
FROM
    customer
GROUP BY
    store_id
HAVING
    COUNT (customer_id) > 300;
```

Data Output | Explain | Messages | Notifications

	store_id smallint	count bigint
1	1	332

SUB QUERIES

What it does?

/ With sub queries you can build complex queries with a clearer syntax.

/ Let’s say we want to get average rental_rate value from film table

/ And we want to filter films that they are above rental_rate

Sub Query

dvdrental/postgres@strive

Query Editor | Query History

```
1  -- find averate rental rate of all movies;
2  SELECT
3      AVG (rental_rate)
4  FROM
5      film;
```

Data Output | Explain | Messages | Notifications

	avg numeric
1	2.9800000000000000

Sub Query is placed

dvdrental/postgres@strive

Query Editor | Query History | Scratch P.

```sql
-- find films that above the average rental rate
SELECT
    film_id,
    title,
    rental_rate
FROM
    film
WHERE
    rental_rate > (
        SELECT
            AVG (rental_rate)
        FROM
            film
    );
```

Data Output | Explain | Messages | Notifications

	film_id [PK] integer	title character varying (255)	rental_rate numeric (4,2)
1	133	Chamber Italian	4.99
2	384	Grosse Wonderful	4.99
3	8	Airport Pollock	4.99
4	98	Bright Encounters	4.99
5	2	Ace Goldfinger	4.99

Exercise 4

/ List the total spend that are above the total average spend by grouping them with the customer_id.

/ Round average with 2 precision using ROUND() function

	customer_id smallint	total_spent_by_customer numeric	average numeric
1	184	80.80	4.20
2	87	137.72	4.20
3	477	106.79	4.20
4	273	130.72	4.20
5	550	151.69	4.20
6	51	123.70	4.20
7	394	77.80	4.20
8	272	65.87	4.20
9	70	75.83	4.20
10	190	102.75	4.20

Exercise 4 Solution

dvdrental/postgres@strive

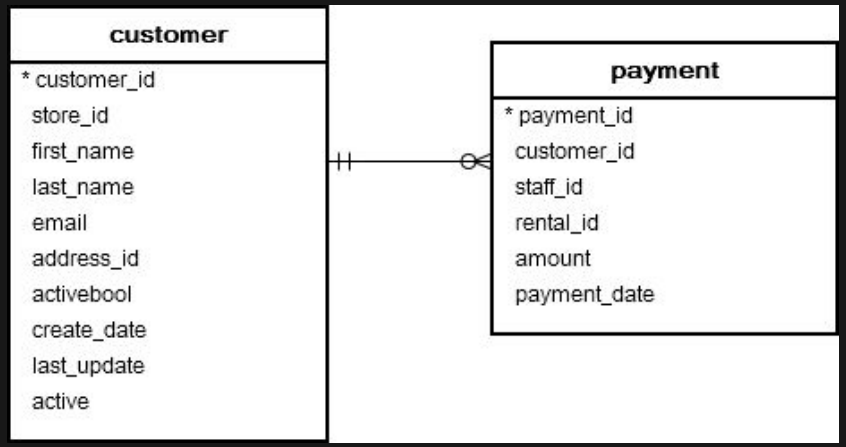
Query Editor | Query History

```sql
 -- find sales that is above average
SELECT
    customer_id,
    SUM(amount) AS total_spent_by_customer,
    ROUND((SELECT AVG(amount) FROM payment),2) AS average
    FROM
    payment
    GROUP BY customer_id
    HAVING SUM(amount) > (
        SELECT AVG(amount) FROM payment
    )
```

JOIN

What it does?

/ In a relation database you have related tables. Join allows you merge them ON their relation



Example

dvdrental/postgres@strive

Query Editor | Query History | Scratch P:

```sql
-- join payment and customer tables on relation customer_id and order by payment_date
SELECT
    customer.customer_id,
    first_name,
    last_name,
    amount,
    payment_date
FROM
    customer
INNER JOIN payment
    ON payment.customer_id = customer.customer_id
ORDER BY payment_date;
```

Data Output | Explain | Messages | Notifications

	customer_id integer	first_name character varying (45)	last_name character varying (45)	amount numeric (5,2)	payment_date timestamp without time zone
1	416	Jeffery	Pinson	2.99	2007-02-14 21:21:59.996577
2	516	Elmer	Noe	4.99	2007-02-14 21:23:39.996577
3	239	Minnie	Romero	4.99	2007-02-14 21:29:00.996577
4	592	Terrance	Roush	6.99	2007-02-14 21:41:12.996577
5	49	Joyce	Edwards	0.99	2007-02-14 21:44:52.996577

Alternative

dvdrental/postgres@strive

Query Editor | Query History | Scratch P

```
1  -- Since both tables have the same customer_id column, you can use the USING syntax:
2  SELECT
3      customer_id,
4      first_name,
5      last_name,
6      amount,
7      payment_date
8  FROM
9      customer
10 INNER JOIN payment USING(customer_id)
11 ORDER BY payment_date;
```

Data Output | Explain | Messages | Notifications

	customer_id integer	first_name character varying (45)	last_name character varying (45)	amount numeric (5,2)	payment_date timestamp without time zone
1	416	Jeffery	Pinson	2.99	2007-02-14 21:21:59.996577
2	516	Elmer	Noe	4.99	2007-02-14 21:23:39.996577
3	239	Minnie	Romero	4.99	2007-02-14 21:29:00.996577
4	592	Terrance	Roush	6.99	2007-02-14 21:41:12.996577
5	49	Joyce	Edwards	0.99	2007-02-14 21:44:52.996577

Join more tables

dvdrental/postgres@strive

Query Editor | Query History | Scratch P:

```sql
-- Let's join staff who made that sale too
SELECT
    c.customer_id,
    c.first_name customer_first_name,
    c.last_name customer_last_name,
    s.first_name staff_first_name,
    s.last_name staff_last_name,
    amount,
    payment_date
FROM
    customer c
INNER JOIN payment p
    ON p.customer_id = c.customer_id
INNER JOIN staff s
    ON p.staff_id = s.staff_id
ORDER BY payment_date;
```

Data Output | Explain | Messages | Notifications

	customer_id integer	customer_first_name character varying (45)	customer_last_name character varying (45)	staff_first_name character varying (45)	staff_last_name character varying (45)	amount numeric (5,2)	payment_date timestamp with
1	416	Jeffery	Pinson	Jon	Stephens	2.99	2007-02-14 2
2	516	Elmer	Noe	Jon	Stephens	4.99	2007-02-14 2
3	239	Minnie	Romero	Mike	Hillyer	4.99	2007-02-14 2
4	592	Terrance	Roush	Jon	Stephens	6.99	2007-02-14 2
5	49	Joyce	Edwards	Mike	Hillyer	0.99	2007-02-14 2

Q&A

Don’t be SHY