

Query Query History

```
1 --Q1. List the first name and last name of all customers.  
2 SELECT first_name, last_name FROM customer;
```

Data Output 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
6	Elizabeth	Brown
7	Jennifer	Davis
8	Maria	Miller
9	Susan	Wilson
10	Margaret	Moore
11	Dorothy	Taylor

Total rows: 599 of 599 Query complete 00:00:00.663 Ln 1, Co

Query Query History

```
1 --Q2. Find all the movies that are currently rented out.  
2 SELECT F.film_id, F.title FROM film F JOIN inventory I ON F.film_id = I.film_id  
3 JOIN rental R ON R.inventory_id = I.inventory_id  
4 WHERE R.return_date IS NULL;
```

Data Output Messages Notifications

	film_id [PK] integer	title character varying (255)
1	1	Academy Dinosaur
2	2	Ace Goldfinger
3	4	Affair Prejudice
4	5	African Egg
5	13	Ali Forever
6	17	Alone Trip
7	19	Amadeus Holy
8	21	American Circus
9	22	Amistad Midsummer
10	39	Armageddon Lost
11	50	Baked Cleopatra

Total rows: 183 of 183 Query complete 00:00:00.158

Ln 3

Query Query History

```
1 --Q3. Show the titles of all movies in the 'Action' category.
2 SELECT F.film_id, title FROM film F JOIN film_category FC ON F.film_id = FC.film_id
3 JOIN category C ON FC.category_id = C.category_id
4 WHERE C.name = 'Action';
```

Data Output Messages Notifications



	film_id [PK] integer	title character varying (255)
1	19	Amadeus Holy
2	21	American Circus
3	29	Antitrust Tomatoes
4	38	Ark Ridgemont
5	56	Barefoot Manchurian
6	67	Berets Agent
7	97	Bride Intrigue
8	105	Bull Shawshank
9	111	Caddyshack Jedi
10	115	Campus Remember
11	126	Casualties Encino

Query Query History

```
1 --Q4. Count the number of films in each category.
2 SELECT name AS Category, COUNT(*) AS film_count
3 FROM category JOIN film_category ON category.category_id = film_category.category_id
4 GROUP BY Category ORDER BY film_count;
```

Data Output Messages Notifications

	category character varying (25) 	film_count bigint 
1	Music	51
2	Horror	56
3	Travel	57
4	Classics	57
5	Comedy	58
6	Children	60
7	Sci-Fi	61
8	Games	61
9	Drama	62
10	New	63
11	Action	64
Total rows: 16 of 16 Query complete 00:00:00 211		

Query Query History

```
1 --Q5. What is the total amount spent by each customer?
2 SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name,
3 SUM(P.amount) AS Total_amount
4 FROM customer C JOIN payment P ON C.customer_id = P.customer_id
5 GROUP BY C.customer_id, Customer_name ORDER BY Total_amount;
```

Data Output Messages Notifications

	customer_id [PK] integer	customer_name text	total_amount numeric
1	318	Brian Wyman	27.93
2	281	Leona Obrien	32.90
3	248	Caroline Bowman	37.87
4	320	Anthony Schwab	47.85
5	110	Tiffany Jordan	49.88
6	586	Kirk Stclair	50.83
7	288	Bobbie Craig	52.81
8	250	Jo Fowler	54.85
9	271	Penny Neal	56.84
10	395	Johnny Turpin	57.81

Query Query History

```
1 --Q6. Find the top 5 customers who spent the most.
2 SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name,
3 SUM(P.amount) AS Total_amount
4 FROM customer C JOIN payment P ON C.customer_id = P.customer_id
5 GROUP BY C.customer_id, Customer_name ORDER BY Total_amount DESC LIMIT(5);|
```

Data Output Messages Notifications

	customer_id [PK] integer	customer_name text	total_amount numeric
1	148	Eleanor Hunt	211.55
2	526	Karl Seal	208.58
3	178	Marion Snyder	194.61
4	137	Rhonda Kennedy	191.62
5	144	Clara Shaw	189.60

Query Query History

```
1 --Q7. Display the rental date and return date for each rental.
2 SELECT rental_id, rental_date, return_date FROM rental;
```

Data Output Messages Notifications



	rental_id [PK] integer	rental_date timestamp without time zone	return_date timestamp without time zone
1	2	2005-05-24 22:54:33	2005-05-28 19:40:33
2	3	2005-05-24 23:03:39	2005-06-01 22:12:39
3	4	2005-05-24 23:04:41	2005-06-03 01:43:41
4	5	2005-05-24 23:05:21	2005-06-02 04:33:21
5	6	2005-05-24 23:08:07	2005-05-27 01:32:07
6	7	2005-05-24 23:11:53	2005-05-29 20:34:53
7	8	2005-05-24 23:31:46	2005-05-27 23:33:46
8	9	2005-05-25 00:00:40	2005-05-28 00:22:40
9	10	2005-05-25 00:02:21	2005-05-31 22:44:21
10	11	2005-05-25 00:09:02	2005-06-02 20:56:02

Query Query History

```
1  --Q8. List the names of staff members and the stores they manage.
2  SELECT S.staff_id, CONCAT(S.first_name, ' ', S.last_name) AS Staff_name, ST.store_id
3  FROM staff S JOIN store ST
4  ON S.staff_id = ST.manager_staff_id;
```

Data Output Messages Notifications



	staff_id integer	staff_name text	store_id integer
1	1	Mike Hillyer	1
2	2	Jon Stephens	2

Query Query History

```
1 --Q9. Find all customers living in 'California'.
2 SELECT C.customer_id, CONCAT(C.first_name, ' ', last_name) AS Customer_name, A.district AS Living_in, A.city_id
3 FROM customer C JOIN address A ON C.address_id = A.address_id
4 WHERE A.district = 'California';
5
```

Data Output Messages Notifications



	staff_id integer	staff_name text	store_id integer
1	1	Mike Hillyer	1
2	2	Jon Stephens	2

Query Query History

```
1 --Q10. Count how many customers are from each city.
2 SELECT CT.city_id, CT.city, COUNT(*) AS Total_customer
3 FROM city CT JOIN address A
4 ON CT.city_id = A.city_id
5 JOIN customer C ON A.address_id = C.address_id
6 GROUP BY CT.city_id
7 ORDER BY Total_customer;
8
```

Data Output Messages Notifications

	city_id [PK] integer	city character varying (50)	total_customer bigint
1	148	Duisburg	1
2	87	Botshabelo	1
3	477	Siegen	1
4	273	Klerksdorp	1
5	550	Udaipur	1
6	51	Balurghat	1
7	394	Pak Kret	1
8	272	Kitwe	1
9	70	Bergamo	1

Query Query History

```
1 --Q11. Find the film(s) with the longest duration.
2 SELECT film_id, title, length FROM film
3 WHERE length = (SELECT MAX(length) FROM film);
4
```

Data Output Messages Notifications

	film_id [PK] integer	title character varying (255)	length smallint
1	141	Chicago North	185
2	182	Control Anthem	185
3	212	Darn Forrester	185
4	349	Gangs Pride	185
5	426	Home Pity	185
6	609	Muscle Bright	185
7	690	Pond Seattle	185
8	817	Soldiers Evolution	185
9	872	Sweet Brotherhood	185
10	991	Worst Banger	185

Query Query History

```
1 --Q12. Which actors appear in the film titled 'Alien Center'?
2 SELECT A.actor_id, CONCAT(A.first_name, ' ', A.last_name) AS Actor_name, F.title
3 FROM actor A JOIN film_actor FA ON A.actor_id = FA.actor_id
4 JOIN film F ON FA.film_id = F.film_id
5 WHERE F.title = 'Alien Center';
6
```

Data Output Messages Notifications



	actor_id integer	actor_name text	title character varying (255)
1	36	Burt Dukakis	Alien Center
2	69	Kenneth Paltrow	Alien Center
3	105	Sidney Crowe	Alien Center
4	117	Renee Tracy	Alien Center
5	164	Humphrey Willis	Alien Center
6	170	Mena Hopper	Alien Center

Query Query History

```
1  --Q13. Find the number of rentals made each month.
2  SELECT TO_CHAR(rental_date,'MM') AS month, COUNT(*) AS Total_rentals
3  FROM rental
4  GROUP BY month
5  ORDER BY month;
6
```

Data Output Messages Notifications



	month text	total_rentals bigint
1	02	182
2	05	1156
3	06	2311
4	07	6709
5	08	5686

Query Query History

```
1 --Q14. Show all payments made by customer 'Mary Smith'.
2 SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name, P.payment_id, P.amount AS Payments
3 FROM payment P JOIN customer C
4 ON P.customer_id = C.customer_id
5 WHERE C.first_name = 'Mary' AND C.last_name = 'Smith';
6
```

Data Output Messages Notifications



	customer_id integer	customer_name text	payment_id integer	payments numeric (5,2)
1	1	Mary Smith	18495	5.99
2	1	Mary Smith	18496	0.99
3	1	Mary Smith	18497	9.99
4	1	Mary Smith	18498	4.99
5	1	Mary Smith	18499	4.99
6	1	Mary Smith	18500	0.99
7	1	Mary Smith	18501	3.99
8	1	Mary Smith	22680	4.99
9	1	Mary Smith	22681	3.99
10	1	Mary Smith	22682	0.99

Query

Query History

1

--Q15. List all films that have never been rented.

2

SELECT F.film_id, F.title, F.description, R.rental_id FROM film F JOIN inventory I

3

ON F.film_id = I.film_id

4

LEFT JOIN rental R ON I.inventory_id = R.inventory_id

5

WHERE R.rental_id IS NULL;

6

Data Output

Messages

Notifications

	film_id integer	title character varying (255)	description text	rental_id integer
1	1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rock...	[null]

Query Query History

```
1 --Q16. What is the average rental duration per category?
2 SELECT C.category_id, C.name AS Category, AVG(F.rental_duration) AS Avg_rental_duration
3 FROM category C JOIN film_category FC
4 ON C.category_id = FC.category_id
5 JOIN film F ON FC.film_id = F.film_id
6 GROUP BY Category, C.category_id
7 ORDER BY C.category_id;
8
```












Data Output Messages Notifications

	category_id [PK] integer	category character varying (25)	avg_rental_duration numeric
1	1	Action	4.9531250000000000
2	2	Animation	4.8939393939393939
3	3	Children	5.0333333333333333
4	4	Classics	5.0701754385964912
5	5	Comedy	4.9310344827586207
6	6	Documentary	4.7647058823529412
7	7	Drama	5.0806451612903226
8	8	Family	5.1739130434782609

Query Query History

```
1 --Q17. Which films were rented more than 50 times?
2 SELECT F.title, COUNT(*) AS Total_rented
3 FROM film F JOIN inventory I
4 ON F.film_id = I.film_id
5 JOIN rental R ON I.inventory_id = R.inventory_id
6 GROUP BY F.title
7 HAVING COUNT(*) > 50;
8
```

Data Output Messages Notifications

								
title character varying (255) 						total_rented bigint 		

Query Query History

```
1 --Q18. List all employees hired after the year 2005.
2 SELECT staff_id, CONCAT(first_name, ' ', last_name) AS Employee_name,
3 TO_CHAR(last_update, 'YYYY') AS hired_year
4 FROM staff WHERE last_update > '2005-12-31';
5
6
```

Data Output Messages Notifications

	staff_id [PK] integer	employee_name text	hired_year text
1	1	Mike Hillyer	2006
2	2	Jon Stephens	2006

Query Query History

```
1 --Q19. Show the number of rentals processed by each staff member.
2 SELECT S.staff_id, CONCAT(S.first_name, ' ', S.last_name) AS Staff_name, COUNT(R.rental_id) AS Total_rentals
3 FROM staff S JOIN rental R
4 ON S.staff_id = R.staff_id
5 GROUP BY S.staff_id;
6
7
```

Data Output Messages Notifications















	staff_id [PK] integer	staff_name text	total_rentals bigint
1	1	Mike Hillyer	8040
2	2	Jon Stephens	8004

Query Query History

```
1 --Q21. What is the most popular film (rented the most)?
2 SELECT F.film_id, F.title, COUNT(R.rental_id) AS Rental_count
3 FROM film F JOIN inventory I ON F.film_id = I.film_id
4 JOIN rental R ON I.inventory_id = R.inventory_id
5 GROUP BY F.film_id
6 ORDER BY Rental_count DESC
7 LIMIT(1);
8
```

Data Output Messages Notifications

<div><div></div></div>			
	film_id [PK] integer 	title character varying (255) 	rental_count bigint 
1	103	Bucket Brotherhood	34

Query Query History

```
1 --Q22. Show all films longer than 2 hours.
2 SELECT film_id, title, length FROM film
3 WHERE length > 120
4 ORDER BY length;
5
6
```

Data Output Messages Notifications

	film_id [PK] integer	title character varying (255)	length smallint
1	207	Dangerous Uptown	121
2	86	Boogie Amelie	121
3	403	Harry Idaho	121
4	93	Brannigan Sunrise	121
5	704	Pure Runner	121
6	37	Arizona Bang	121
7	658	Paris Weekend	121
8	490	Jumanji Blade	121

Query

Query History

```

1  --Q23. Find all rentals that were returned late.
2  SELECT R.rental_id, R.rental_date, R.return_date, F.rental_duration*INTERVAL '1 DAY' AS Duration
3  FROM rental R JOIN inventory I
4  ON R.inventory_id = I.inventory_id
5  JOIN film F ON I.film_id = F.film_id
6  WHERE R.return_date > R.rental_date + F.rental_duration*INTERVAL '1 DAY';
7
8

```

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	rental_id [PK] integer	rental_date timestamp without time zone	return_date timestamp without time zone	duration interval
1	3	2005-05-24 23:03:39	2005-06-01 22:12:39	7 days
2	4	2005-05-24 23:04:41	2005-06-03 01:43:41	6 days
3	5	2005-05-24 23:05:21	2005-06-02 04:33:21	5 days
4	7	2005-05-24 23:11:53	2005-05-29 20:34:53	4 days
5	10	2005-05-25 00:02:21	2005-05-31 22:44:21	5 days
6	11	2005-05-25 00:09:02	2005-06-02 20:56:02	4 days
7	13	2005-05-25 00:22:55	2005-05-30 04:28:55	3 days
8	15	2005-05-25 00:39:22	2005-06-03 03:30:22	4 days
9	23	2005-05-25 02:40:21	2005-05-29 06:34:21	4 days

Query Query History

```
1  --Q24. List customers and the number of films they rented.
2  SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name, COUNT(R.rental_id) AS Film_rented
3  FROM customer C JOIN rental R
4  ON C.customer_id = R.customer_id
5  GROUP BY C.customer_id
6  ORDER BY Film_rented;
7
8
```

Data Output Messages Notifications

	customer_id [PK] integer	customer_name text	film_rented bigint
1	318	Brian Wyman	12
2	281	Leona Obrien	14
3	61	Katherine Rivera	14
4	110	Tiffany Jordan	14
5	136	Anita Morales	15
6	248	Caroline Bowman	15
7	492	Lester Kraus	16
8	164	Joann Gardner	16

Query Query History

```
1  --Q25. Write a query to show top 3 rented film categories.
2  SELECT C.category_id, C.name AS Film_category, COUNT(R.rental_id) AS Total_rented
3  FROM rental R JOIN inventory I
4  ON R.inventory_id = I.inventory_id
5  JOIN film_category FC
6  ON I.film_id = FC.film_id
7  JOIN category C
8  ON FC.category_id = C.category_id
9  GROUP BY C.category_id
10 ORDER BY Total_rented DESC
11 LIMIT(3);
12
```

Data Output Messages Notifications



	category_id [PK] integer	film_category character varying (25)	total_rented bigint
1	15	Sports	1179
2	2	Animation	1166
3	1	Action	1112

Query Query History

```
1  --Q26. Create a view that shows all customer names and their payment totals.
2  CREATE VIEW customer_payment_total AS
3  SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name, SUM(P.amount) AS Total_payment
4  FROM customer C JOIN payment P
5  ON C.customer_id = P.customer_id
6  GROUP BY C.customer_id
7  ORDER BY Total_payment;
8
```

Data Output **Messages** Notifications

CREATE VIEW

Query returned successfully in 126 msec.

Query Query History

```
1  --Q27. Update a customer's email address given their ID.
2  UPDATE customer
3  SET email = 'newmail@gmail.com'
4  WHERE customer_id = '1';
5
6
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 113 msec.

Query Query History

```
1 --Q28. Insert a new actor into the actor table.  
2 INSERT INTO actor (first_name, last_name, last_update)  
3 VALUES ('Tony', 'Stark', CURRENT_TIMESTAMP);  
4  
5 |
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 266 msec.

Query Query History

```
1 --Q29. Delete all records from the rentals table where return_date is NULL.  
2 DELETE FROM rental  
3 WHERE rental_date IS NULL  
4  
5
```

Data Output Messages Notifications

DELETE 0

Query returned successfully in 317 msec.

Query Query History

1	--Q30. Add a new column 'age' to the customer table.
2	ALTER TABLE customer
3	ADD COLUMN age INTEGER;
4	
5	

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 211 msec.

Query Query History

```
1 --Q31. Create an index on the 'title' column of the film table.  
2 CREATE INDEX index_title  
3 ON film(title);  
4  
5
```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 128 msec.

Query Query History

```
1  --Q32. Find the total revenue generated by each store.
2  SELECT ST.store_id, SUM(P.amount) AS Total_revenue
3  FROM store ST JOIN staff SF
4  ON ST.store_id = SF.store_id
5  JOIN payment P
6  ON SF.staff_id = P.staff_id
7  GROUP BY ST.store_id;
8
```

Data Output Messages Notifications



	store_id [PK] integer	total_revenue numeric
1	1	30252.12
2	2	31059.92

Query Query History

```
1 --Q34. How many films belong to more than one category?
2 SELECT COUNT(*) AS multi_category_film FROM (
3 SELECT film_id FROM film_category
4 GROUP BY film_id
5 HAVING COUNT(category_id) > 1);
6
7
8
```

Data Output Messages Notifications



multi_category_film
bigint



1

0

Query Query History

```
1 --Q33. What is the city with the highest number of rentals?
2 SELECT CI.city_id, CI.city, COUNT(R.rental_id) AS Highest_rental
3 FROM city CI JOIN address A
4 ON CI.city_id = A.city_id
5 JOIN customer C
6 ON A.address_id = C.address_id
7 JOIN rental R
8 ON C.customer_id = R.customer_id
9 GROUP BY CI.city_id
10 ORDER BY Highest_rental DESC
11 LIMIT(1);
12
```

Data Output Messages Notifications



	city_id [PK] integer	city character varying (50)	highest_rental bigint
1	42	Aurora	50

Query Query History

```
1 --Q35. List the top 10 actors by number of films they appeared in.
2 SELECT A.actor_id, CONCAT(A.first_name, ' ', A.last_name) AS Actor_name, COUNT(FA.film_id) AS Total_films
3 FROM actor A JOIN film_actor FA
4 ON A.actor_id = FA.actor_id
5 GROUP BY A.actor_id
6 ORDER BY Total_films DESC
7 LIMIT(10);
8
```

Data Output Messages Notifications



	actor_id [PK] integer	actor_name text	total_films bigint
1	107	Gina Degeneres	42
2	102	Walter Torn	41
3	198	Mary Keitel	40
4	181	Matthew Carrey	39
5	23	Sandra Kilmer	37
6	81	Scarlett Damon	36
7	144	Angela Witherspoon	35
8	158	Vivien Basinger	35

Query Query History

Execute/Refresh

F5

```
1  --Q36. Retrieve the email addresses of customers who rented 'Matrix Revolutions'.
2  SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name, C.email
3  FROM customer C JOIN rental R
4  ON C.customer_id = R.customer_id
5  JOIN inventory I
6  ON R.inventory_id = I.inventory_id
7  JOIN film F
8  ON I.film_id = F.film_id
9  WHERE F.title = 'Matrix Revolutions';
10
11
```

Data Output Messages Notifications



customer_id
[PK] integer

customer_name
text

email
character varying (50)

Query Query History

```
1 --Q37. Create a stored function to return customer payment total given their ID.
2 CREATE OR REPLACE FUNCTION get_customer_total_payment(Cid INTEGER)
3 RETURNS NUMERIC AS $$
4 BEGIN
5 RETURN (
6 SELECT SUM(amount) FROM payment WHERE customer_id = Cid);
7 END;
8 $$ LANGUAGE plpgsql;
9
10
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 224 msec.

Query Query History

```
1  --Q38. Begin a transaction that updates stock and inserts a rental record.
2  BEGIN;
3  UPDATE inventory
4  SET last_update = CURRENT_TIMESTAMP
5  WHERE inventory_id = 1;
6  INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id, return_date)
7  VALUES (CURRENT_TIMESTAMP, 1, 1, 1, NULL);
8  COMMIT;
9
10
```

Data Output Messages Notifications

COMMIT

Query returned successfully in 118 msec.

Query Query History

```
1  --Q39. Show the customers who rented films in both 'Action' and 'Comedy' categories.
2  SELECT C.customer_id, CONCAT(C.first_name, ' ', C.last_name) AS Customer_name
3  FROM customer C
4  WHERE EXISTS (
5  SELECT 1
6  FROM rental R JOIN inventory I
7  ON R.inventory_id = I.inventory_id
8  JOIN film_category FC
9  ON I.film_id = FC.film_id
10 JOIN category CA
11 ON FC.category_id = CA.category_id
12 WHERE CA.name = 'Action' AND C.customer_id = R.customer_id)
13 AND EXISTS (
14 SELECT 1
15 FROM rental R JOIN inventory I
16 ON R.inventory_id = I.inventory_id
17 JOIN film_category FC
18 ON I.film_id = FC.film_id
19 JOIN category CA
20 ON FC.category_id = CA.category_id
21 WHERE CA.name = 'Comedy' AND C.customer_id = R.customer_id)
22 ORDER BY C.customer_id;
23
```



	customer_id [PK] integer	customer_name text
1	1	Mary Smith
2	3	Linda Williams
3	4	Barbara Jones
4	5	Elizabeth Brown
5	6	Jennifer Davis
6	7	Maria Miller
7	10	Dorothy Taylor
8	12	Nancy Thomas
9	13	Karen Jackson
10	14	Betty White
11	15	Helen Harris
12	16	Sandra Martin
13	18	Carol Garcia
14	20	Sharon Robinson
15	21	Michelle Clark
16	22	Laura Rodriguez
17	23	Sarah Lewis

Query Query History

```
1  --Q40. Find actors who have never acted in a film.
2  SELECT A.actor_id, CONCAT(A.first_name, ' ', A.last_name) AS Actor_name
3  FROM actor A LEFT JOIN film_actor FA
4  ON A.actor_id = FA.actor_id
5  WHERE FA.film_id IS NULL;
6
```

Data Output Messages Notifications



	actor_id [PK] integer	actor_name text
1	201	Tony Stark