POSTGRESQL ASSIGNMENT

1. List the first name and last name of all customers.

Solution:

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × postgresqlAnswer.sql* ×

dvd_rental/postgres@PostgreSQL 17

No limit

Query Query History

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

Data Output Messages Notifications

Showing rows: 1 to 599

	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
12	Lisa	Anderson
13	Nancy	Thomas
14	Karen	Jackson
15	Betty	White
16	Helen	Harris

Total rows: 599 Query complete 00:00:00.192

2. Find all the movies that are currently rented out.

Solution:

No limit

Query Query History

```sql
SELECT film.title
FROM rental
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL;
```

Data Output Messages Notifications

Show

	title character varying (255)
1	Academy Dinosaur
2	Ace Goldfinger
3	Affair Prejudice
4	African Egg
5	Ali Forever
6	Alone Trip
7	Amadeus Holy
8	American Circus
9	Amistad Midsummer
10	Armageddon Lost
11	Baked Cleopatra
12	Bang Kwai
13	Basic Easy
14	Berets Agent
15	Blade Polish

Total rows: 183 Query complete 00:00:00.163

3. Show the titles of all movies in the 'Action' category.

Solution:

Query Query History

```sql
SELECT film.title FROM film
JOIN film_category ON film.film_id = film_category.film_id
JOIN category ON film_category.category_id = category.category_id
WHERE category.name = 'Action';
```

Data Output Messages Notifications

Showing row

	title character varying (255)
1	Amadeus Holy
2	American Circus
3	Antitrust Tomatoes
4	Ark Ridgemont
5	Barefoot Manchurian
6	Berets Agent
7	Bride Intrigue
8	Bull Shawshank
9	Caddyshack Jedi
10	Campus Remember
11	Casualties Encino
12	Celebrity Horn
13	Clueless Bucket
14	Crow Grease
15	Dances None

Total rows: 64 Query complete 00:00:00.293

4. Count the number of films in each category.

Solution:

Query Query History

```sql
SELECT category.name AS category, COUNT(film.film_id) AS film_count FROM category
JOIN film_category ON category.category_id = film_category.category_id
JOIN film ON film_category.film_id = film.film_id
GROUP BY category.name;
```

Data Output Messages Notifications

Showing rows: 1 to 16

	category character varying (25)	film_count bigint
1	Family	69
2	Games	61
3	Animation	66
4	Classics	57
5	Documentary	68
6	New	63
7	Sports	74
8	Children	60
9	Music	51
10	Travel	57
11	Foreign	73
12	Drama	62
13	Horror	56
14	Action	64
15	Sci-Fi	61

Total rows: 16 Query complete 00:00:00.139

5. What is the total amount spent by each customer?

Solution:

Query Query History

```sql
SELECT customer_id, SUM(amount) AS total_spent
FROM payment
GROUP BY customer_id;
```

Data Output Messages Notifications

	customer_id smallint	total_spent numeric
1	184	80.80
2	87	137.72
3	477	106.79
4	273	130.72
5	550	151.69
6	51	123.70
7	394	77.80
8	272	65.87
9	70	75.83
10	190	102.75
11	350	63.79
12	539	84.80
13	554	95.80
14	278	71.79
15	424	109.71

Total rows: 599 Query complete 00:00:00.149

6. Find the top 5 customers who spent the most.

Solution:

Query Query History

```sql
SELECT customer_id, SUM(amount) AS total_spent FROM payment
GROUP BY customer_id
ORDER BY total_spent DESC
LIMIT 5;
```

Data Output Messages Notifications

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

7. Display the rental date and return date for each rental.

Solution:

Query | Query History

```
SELECT rental_date, return_date FROM rental;
```

Data Output | Messages | Notifications

	rental_date timestamp without time zone	return_date timestamp without time zone
1	2005-05-24 22:54:33	2005-05-28 19:40:33
2	2005-05-24 23:03:39	2005-06-01 22:12:39
3	2005-05-24 23:04:41	2005-06-03 01:43:41
4	2005-05-24 23:05:21	2005-06-02 04:33:21
5	2005-05-24 23:08:07	2005-05-27 01:32:07
6	2005-05-24 23:11:53	2005-05-29 20:34:53
7	2005-05-24 23:31:46	2005-05-27 23:33:46
8	2005-05-25 00:00:40	2005-05-28 00:22:40
9	2005-05-25 00:02:21	2005-05-31 22:44:21
10	2005-05-25 00:09:02	2005-06-02 20:56:02
11	2005-05-25 00:19:27	2005-05-30 05:44:27
12	2005-05-25 00:22:55	2005-05-30 04:28:55
13	2005-05-25 00:31:15	2005-05-26 02:56:15
14	2005-05-25 00:39:22	2005-06-03 03:30:22
15	2005-05-25 00:43:11	2005-05-26 04:42:11
16	2005-05-25 01:06:36	2005-05-27 00:43:36
17	2005-05-25 01:10:47	2005-05-31 06:35:47
18	2005-05-25 01:17:24	2005-05-31 06:00:24

Total rows: 16044 | Query complete 00:00:00.118

8. List the names of staff members and the stores they manage.

Solution:

Query | Query History

```
SELECT staff.first_name, staff.last_name, store.store_id FROM staff
JOIN store ON staff.staff_id = store.manager_staff_id;
```

Data Output | Messages | Notifications

	first_name character varying (45)	last_name character varying (45)	store_id integer
1	Mike	Hillyer	1
2	Jon	Stephens	2

9. Find all customers living in 'California'.

Solution:

Query Query History

```sql
SELECT customer.first_name, customer.last_name FROM customer
JOIN address ON customer.address_id = address.address_id
JOIN city ON address.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
WHERE address.district = 'California';
```

Data Output Messages Notifications

Showing rows: 1 to 2

	first_name character varying (45)	last_name character varying (45)	store_id integer
1	Mike	Hillyer	1
2	Jon	Stephens	2

10. Count how many customers are from each city.

Solution:

Query Query History

```sql
SELECT city.city, COUNT(customer.customer_id) AS customer_count FROM customer
JOIN address ON customer.address_id = address.address_id
JOIN city ON address.city_id = city.city_id
GROUP BY city.city;
```

Data Output Messages Notifications

Showing rows:

	city character varying (50)	customer_count bigint
1	Southport	1
2	Taguig	1
3	Tokat	1
4	Atlixco	1
5	Mukateve	1
6	Pontianak	1
7	Gatineau	1
8	Saint-Denis	1
9	Molodetno	1
10	Yingkou	1
11	Changhwa	1
12	Rampur	1
13	Caracas	1
14	Apeldoorn	1

Total rows: 597 Query complete 00:00:00.125

11. Find the film(s) with the longest duration.

Solution:

Query Query History

```
SELECT title, length FROM film
WHERE length = (SELECT MAX(length) FROM film);
```

Data Output Messages Notifications

Showing rows: 1 to

	title character varying (255)	length smallint
1	Chicago North	185
2	Control Anthem	185
3	Darn Forrester	185
4	Gangs Pride	185
5	Home Pity	185
6	Muscle Bright	185
7	Pond Seattle	185
8	Soldiers Evolution	185
9	Sweet Brotherhood	185
10	Worst Banger	185

Total rows: 10 Query complete 00:00:00.112

12. Which actors appear in the film titled 'Alien Center'?

Solution:

Query Query History

```
SELECT actor.first_name, actor.last_name FROM actor
JOIN film_actor ON actor.actor_id = film_actor.actor_id
JOIN film ON film_actor.film_id = film.film_id
WHERE film.title = 'Alien Center';
```

Data Output Messages Notifications

Showing rows: 1 to 6

	first_name character varying (45)	last_name character varying (45)
1	Burt	Dukakis
2	Kenneth	Paltrow
3	Sidney	Crowe
4	Renee	Tracy
5	Humphrey	Willis
6	Mena	Hopper

13. Find the number of rentals made each month.

Solution:

Query | Query History

```
SELECT DATE_TRUNC('month', rental_date) AS month, COUNT(*) AS rentals FROM rental
GROUP BY month ORDER BY month;
```

Data Output | Messages | Notifications

Showing r

	month timestamp without time zone	rentals bigint
1	2005-05-01 00:00:00	1156
2	2005-06-01 00:00:00	2311
3	2005-07-01 00:00:00	6709
4	2005-08-01 00:00:00	5686
5	2006-02-01 00:00:00	182

14. Show all payments made by customer 'Mary Smith'.

Solution:

Query | Query History

```
SELECT payment.* FROM payment
JOIN customer ON payment.customer_id = customer.customer_id
WHERE customer.first_name = 'Mary' AND customer.last_name = 'Smith';
```

Data Output | Messages | Notifications

Showing rows: 1

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp without time zone
1	18495	1	1	1185	5.99	2007-02-14 23:22:38.996577
2	18496	1	2	1422	0.99	2007-02-15 16:31:19.996577
3	18497	1	2	1476	9.99	2007-02-15 19:37:12.996577
4	18498	1	1	1725	4.99	2007-02-16 13:47:23.996577
5	18499	1	1	2308	4.99	2007-02-18 07:10:14.996577
6	18500	1	2	2363	0.99	2007-02-18 12:02:25.996577
7	18501	1	1	3284	3.99	2007-02-21 04:53:11.996577
8	22680	1	2	10437	4.99	2007-03-01 07:19:30.996577
9	22681	1	2	11299	3.99	2007-03-02 14:05:18.996577
10	22682	1	1	11367	0.99	2007-03-02 16:30:04.996577
11	22683	1	2	11824	4.99	2007-03-17 11:06:20.996577
12	22684	1	1	12250	0.99	2007-03-18 02:25:55.996577
13	22685	1	2	13068	0.99	2007-03-19 08:23:42.996577
14	22686	1	2	13176	2.99	2007-03-19 12:25:20.996577
15	22687	1	1	14762	0.99	2007-03-21 22:02:23.996577
16	22688	1	1	14825	1.99	2007-03-21 23:56:23.996577

Total rows: 30 | Query complete 00:00:00.115

15. List all films that have never been rented.

Solution:

Query Query History

```sql
SELECT film.title FROM film
JOIN inventory ON film.film_id = inventory.film_id
LEFT JOIN rental ON inventory.inventory_id = rental.inventory_id
WHERE rental.rental_id IS NULL;
```

Data Output Messages Notifications

	title character varying (255)
1	Academy Dinosaur

16. What is the average rental duration per category?

Solution:

Query Query History

```sql
SELECT category.name, AVG(film.rental_duration) AS avg_duration FROM category
JOIN film_category ON category.category_id = film_category.category_id
JOIN film ON film_category.film_id = film.film_id
GROUP BY category.name;
```

Data Output Messages Notifications

Showing rows: 1 to 1

	name character varying (25)	avg_duration numeric
1	Family	5.1739130434782609
2	Games	5.0655737704918033
3	Animation	4.8939393939393939
4	Classics	5.0701754385964912
5	Documentary	4.7647058823529412
6	New	4.7460317460317460
7	Sports	4.7162162162162162
8	Children	5.0333333333333333
9	Music	5.2352941176470588
10	Travel	5.3508771929824561
11	Foreign	5.1095890410958904
12	Drama	5.0806451612903226
13	Horror	4.8571428571428571
14	Action	4.9531250000000000
15	Sci-Fi	4.8852459016393443
16	Comedy	4.9310344827586207

Total rows: 16 Query complete 00:00:00.113

17. Which films were rented more than 50 times?

Solution:

Query Query History

```
SELECT film.title, COUNT(*) AS rental_count FROM rental
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id
GROUP BY film.title HAVING COUNT(*) > 50;
```

Data Output Messages Notifications

	title character varying (255)	rental_count bigint

18. List all employees hired after the year 2005.

Solution:

Query Query History

```
SELECT staff_id, first_name, last_name, email, last_update FROM staff
WHERE last_update > '2005-12-31';
```

Data Output Messages Notifications

Showing rows: 1 to 2

	staff_id [PK] integer	first_name character varying (45)	last_name character varying (45)	email character varying (50)	last_update timestamp without time zone
1	1	Mike	Hillyer	Mike.Hillyer@sakilastaff.com	2006-05-16 16:13:11.79328
2	2	Jon	Stephens	Jon.Stephens@sakilastaff.com	2006-05-16 16:13:11.79328

19. Show the number of rentals processed by each staff member.

Solution:

Query Query History

```
SELECT staff_id, COUNT(*) AS rental_count FROM rental GROUP BY staff_id;
```

Data Output Messages Notifications

Showin

	staff_id smallint	rental_count bigint
1	1	8040
2	2	8004

20. Display all customers who have not made any payments.

Solution:

Query Query History

```
SELECT customer.first_name, customer.last_name FROM customer
LEFT JOIN payment ON customer.customer_id = payment.customer_id
WHERE payment.payment_id IS NULL;
```

Data Output Messages Notifications

	first_name character varying (45)	last_name character varying (45)

21. What is the most popular film (rented the most)?

Solution:

Query Query History

```
SELECT film.title, COUNT(*) AS rental_count FROM rental
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id
GROUP BY film.title ORDER BY rental_count DESC LIMIT 1;
```

Data Output Messages Notifications

Showing rows

	title character varying (255)	rental_count bigint
1	Bucket Brotherhood	34

22. Show all films longer than 2 hours.

Solution:

Query Query History

```
SELECT title, length FROM film WHERE length > 120;
```

Data Output Messages Notifications

Showing

	title character varying (255)	length smallint
1	African Egg	130
2	Agent Truman	169
3	Alamo Videotape	126
4	Alaska Phantom	136
5	Ali Forever	150
6	Alley Evolution	180
7	American Circus	129
8	Analyze Hoosiers	181
9	Anonymous Human	179
10	Antitrust Tomatoes	168
11	Apollo Teen	153
12	Arachnophobia Rollercoaster	147
13	Argonauts Town	127
14	Arizona Bang	121
15	Army Flintstones	148
16	Arsenic Independence	137
17	Artist Coldblooded	170

Total rows: 457 Query complete 00:00:00.104

23. Find all rentals that were returned late.

Solution:

Query | Query History | Scratch Pa

```
SELECT * FROM rental WHERE return_date > rental_date + INTERVAL '3 day';
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone
1	2	2005-05-24 22:54:33	1525	459	2005-05-28 19:40:33	1	2006-02-16 02:30:53
2	3	2005-05-24 23:03:39	1711	408	2005-06-01 22:12:39	1	2006-02-16 02:30:53
3	4	2005-05-24 23:04:41	2452	333	2005-06-03 01:43:41	2	2006-02-16 02:30:53
4	5	2005-05-24 23:05:21	2079	222	2005-06-02 04:33:21	1	2006-02-16 02:30:53
5	7	2005-05-24 23:11:53	3995	269	2005-05-29 20:34:53	2	2006-02-16 02:30:53
6	8	2005-05-24 23:31:46	2346	239	2005-05-27 23:33:46	2	2006-02-16 02:30:53
7	9	2005-05-25 00:00:40	2580	126	2005-05-28 00:22:40	1	2006-02-16 02:30:53
8	10	2005-05-25 00:02:21	1824	399	2005-05-31 22:44:21	2	2006-02-16 02:30:53
9	11	2005-05-25 00:09:02	4443	142	2005-06-02 20:56:02	2	2006-02-16 02:30:53
10	12	2005-05-25 00:19:27	1584	261	2005-05-30 05:44:27	2	2006-02-16 02:30:53
11	13	2005-05-25 00:22:55	2294	334	2005-05-30 04:28:55	1	2006-02-16 02:30:53
12	15	2005-05-25 00:39:22	3049	319	2005-06-03 03:30:22	1	2006-02-16 02:30:53
13	18	2005-05-25 01:10:47	3376	19	2005-05-31 06:35:47	2	2006-02-16 02:30:53
14	19	2005-05-25 01:17:24	1941	456	2005-05-31 06:00:24	1	2006-02-16 02:30:53
15	23	2005-05-25 02:40:21	4441	438	2005-05-29 06:34:21	1	2006-02-16 02:30:53
16	26	2005-05-25 03:36:50	4371	371	2005-05-31 00:34:50	1	2006-02-16 02:30:53
17	27	2005-05-25 03:41:50	1225	301	2005-05-30 01:13:50	2	2006-02-16 02:30:53

Total rows: 11472 | Query complete 00:00:00.167

24. List customers and the number of films they rented.

Solution:

Query | Query History

```
SELECT customer_id, COUNT(*) AS film_count FROM rental GROUP BY customer_id;
```

Data Output | Messages | Notifications

Showing

	customer_id smallint	film_count bigint
1	87	30
2	184	23
3	477	22
4	273	35
5	550	32
6	394	22
7	51	33
8	272	20
9	70	18
10	190	27
11	350	23
12	539	22
13	554	22
14	278	26
15	406	32
16	424	30
17	176	37

Total rows: 599 | Query complete 00:00:00.123

25. Write a query to show top 3 rented film categories.

Solution:

Query Query History

```sql
SELECT category.name, COUNT(*) AS rental_count FROM rental
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film_category ON inventory.film_id = film_category.film_id
JOIN category ON film_category.category_id = category.category_id
GROUP BY category.name
ORDER BY rental_count DESC LIMIT 3;
```

Data Output Messages Notifications

	name character varying (25)	rental_count bigint
1	Sports	1179
2	Animation	1166
3	Action	1112

26. Create a view that shows all customer names and their payment totals.

Solution:

Query Query History

```sql
CREATE VIEW customer_payment_totals AS
SELECT customer.first_name, customer.last_name, SUM(payment.amount) AS total_payment FROM customer
JOIN payment ON customer.customer_id = payment.customer_id
GROUP BY customer.first_name, customer.last_name;
```

Data Output Messages Notifications

```
CREATE VIEW

Query returned successfully in 90 msec.
```

27. Update a customer's email address given their ID.

Solution:

Query Query History

```sql
UPDATE customer SET email = 'newemail@example.com' WHERE customer_id = 1;
```

Data Output Messages Notifications

```
UPDATE 1

Query returned successfully in 75 msec.
```

28. Insert a new actor into the actor table.

Solution:

Query Query History

```
INSERT INTO actor (first_name, last_name) VALUES ('John', 'Doe');
```

Data Output Messages Notifications

```
INSERT 0 1

Query returned successfully in 71 msec.
```

29. Delete all records from the rentals table where return_date is NULL.

Solution:

Query Query History Scratch Pa

```
SELECT * FROM rental WHERE return_date IS NULL;
```

Data Output Messages Notifications

Showing rows: 1 to 185 Page No: 1

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone
1	11496	2006-02-14 15:16:03	2047	155	[null]	1	2006-02-16 02:30:53
2	11541	2006-02-14 15:16:03	2026	335	[null]	1	2006-02-16 02:30:53
3	12101	2006-02-14 15:16:03	1556	479	[null]	1	2006-02-16 02:30:53
4	11563	2006-02-14 15:16:03	1545	83	[null]	1	2006-02-16 02:30:53
5	11577	2006-02-14 15:16:03	4106	219	[null]	2	2006-02-16 02:30:53
6	11593	2006-02-14 15:16:03	817	99	[null]	1	2006-02-16 02:30:53
7	11611	2006-02-14 15:16:03	1857	192	[null]	2	2006-02-16 02:30:53
8	11646	2006-02-14 15:16:03	478	11	[null]	2	2006-02-16 02:30:53
9	11652	2006-02-14 15:16:03	1622	597	[null]	2	2006-02-16 02:30:53
10	11657	2006-02-14 15:16:03	3043	53	[null]	2	2006-02-16 02:30:53
11	13719	2006-02-14 15:16:03	3547	208	[null]	1	2006-02-16 02:30:53
12	11672	2006-02-14 15:16:03	3947	521	[null]	2	2006-02-16 02:30:53
13	11676	2006-02-14 15:16:03	4496	216	[null]	2	2006-02-16 02:30:53
14	11709	2006-02-14 15:16:03	1720	330	[null]	1	2006-02-16 02:30:53
15	11739	2006-02-14 15:16:03	4568	373	[null]	2	2006-02-16 02:30:53
16	11754	2006-02-14 15:16:03	3747	163	[null]	2	2006-02-16 02:30:53
17	11757	2006-02-14 15:16:03	1295	550	[null]	2	2006-02-16 02:30:53

Total rows: 185 Query complete 00:00:00.084

30. Add a new column 'age' to the customer table.

Solution:

Query Query History

```
ALTER TABLE customer ADD COLUMN age INT;
```

Data Output Messages Notifications

```
ALTER TABLE

Query returned successfully in 65 msec.
```

31. Create an index on the 'title' column of the film table.

Solution:

Query Query History

```
CREATE INDEX idx_film_title ON film(title);
```

Data Output Messages Notifications

```
CREATE INDEX

Query returned successfully in 64 msec.
```

32. Find the total revenue generated by each store.

Solution:

Query Query History

```
SELECT store.store_id, SUM(payment.amount) AS total_revenue FROM payment
JOIN staff ON payment.staff_id = staff.staff_id
JOIN store ON staff.store_id = store.store_id
GROUP BY store.store_id;
```

Data Output Messages Notifications

Showing rows: 1 t

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

33. What is the city with the highest number of rentals?

Solution:

Query Query History

```sql
SELECT city.city, COUNT(*) AS rental_count FROM rental
JOIN customer ON rental.customer_id = customer.customer_id
JOIN address ON customer.address_id = address.address_id
JOIN city ON address.city_id = city.city_id
GROUP BY city.city
ORDER BY rental_count DESC LIMIT 1;
```

Data Output Messages Notifications

	city character varying (50)	rental_count bigint
1	Aurora	50

34. How many films belong to more than one category?

Solution:

Query Query History

```sql
SELECT COUNT(*)
FROM (
    SELECT film_id FROM film_category
    GROUP BY film_id
    HAVING COUNT(category_id) > 1
) AS multi_category_films;
```

Data Output Messages Notifications

	count bigint
1	0

35. List the top 10 actors by number of films they appeared in.

Solution:

Query Query History

```sql
SELECT actor.first_name, actor.last_name, COUNT(*) AS film_count FROM actor
JOIN film_actor ON actor.actor_id = film_actor.actor_id
GROUP BY actor.actor_id
ORDER BY film_count DESC LIMIT 10;
```

Data Output Messages Notifications

Showing rows: 1 to 10

	first_name character varying (45)	last_name character varying (45)	film_count bigint
1	Gina	Degeneres	42
2	Walter	Torn	41
3	Mary	Keitel	40
4	Matthew	Carrey	39
5	Sandra	Kilmer	37
6	Scarlett	Damon	36
7	Angela	Witherspoon	35
8	Vivien	Basinger	35
9	Val	Bolger	35
10	Henry	Berry	35

36. Retrieve the email addresses of customers who rented 'Matrix Revolutions'.

Solution:

Query Query History

```
SELECT DISTINCT customer.email FROM customer
JOIN rental ON customer.customer_id = rental.customer_id
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id
WHERE film.title = 'Matrix Revolutions';
```

Data Output Messages Notifications

email character varying (50)

37. Create a stored function to return customer payment total given their ID.

Solution:

Query Query History

```
CREATE OR REPLACE FUNCTION get_customer_total_payment(cust_id INT)
RETURNS NUMERIC (10,2) AS $$
DECLARE
    total NUMERIC(10,2);
BEGIN
    SELECT COALESCE(SUM (amount), 0.00)
    INTO total
    FROM payment
    WHERE customer_id = cust_id;
    RETURN total;
END;
$$ LANGUAGE plpgsql;
```

Data Output Messages Notifications

```
CREATE FUNCTION

Query returned successfully in 58 msec.
```

38. Begin a transaction that updates stock and inserts a rental record.

Solution:

Query Query History

```
BEGIN;
UPDATE inventory
SET last_update = NOW()
WHERE inventory_id = 1;
INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id, last_update)
VALUES (NOW(), 1, 1, 1, NOW());
COMMIT;
```

Data Output Messages Notifications

```
COMMIT

Query returned successfully in 94 msec.
```

39. Show the customers who rented films in both 'Action' and 'Comedy' categories.

Solution:

Query Query History

```sql
SELECT customer_id
FROM (
    SELECT customer_id, category.name
    FROM rental
    JOIN inventory ON rental.inventory_id = inventory.inventory_id
    JOIN film_category ON inventory.film_id = film_category.film_id
    JOIN category ON film_category.category_id = category.category_id
    GROUP BY customer_id, category.name
) AS cat_rents GROUP BY customer_id
HAVING COUNT(DISTINCT name) FILTER (WHERE name IN ('Action', 'Comedy')) = 2;
```

Data Output Messages Notifications

Showing r

	customer_id smallint
1	1
2	3
3	4
4	5
5	6
6	7
7	10
8	12
9	13
10	14

Total rows: 419 Query complete 00:00:00.129

40. Find actors who have never acted in a film.

Solution:

Query Query History

```sql
SELECT * FROM actor
WHERE actor_id NOT IN (SELECT actor_id FROM film_actor);
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	201	John	Doe	2025-07-13 17:51:17.327575