EJERCICIOS SQL

Máster Data Analytics para la Empresa - EDEM

Carlos Donoso Cabero

1. Proporciona una SQL que muestre los siguientes datos:

- - Nombre Actor
- - Apellido Actor

dvdrental/postgres@server1

Query Editor Query History

```
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

select first_name, last_name from actor;

2. Proporciona una SQL que muestre los siguientes datos:

- - Nombre Actor
- - Título de la Película

dvdrental/postgres@server1

Query Editor Query History

```
select
    concat(actor.first_name, ' ', actor.last_name) as actor_name,
    film.title
from actor
    inner join film_actor on actor.actor_id = film_actor.actor_id
    inner join film on film_actor.film_id = film.film_id;
```

Data Output Explain Messages Notifications

	actor_name text	title character varying (255)
1	Penelope Guiness	Academy Dinosaur
2	Penelope Guiness	Anaconda Confessions
3	Penelope Guiness	Angels Life
4	Penelope Guiness	Bulworth Commandments
5	Penelope Guiness	Cheaper Clyde

```
select
        concat(actor.first_name, ' ', actor.last_name) as actor_name,
        film.title
from actor
        inner join film_actor on actor.actor_id = film_actor.actor_id
        inner join film on film_actor.film_id = film.film_id;
```

3. Proporciona una SQL que muestre los siguientes datos:

- - Nombre Actor
- - Número de películas
- - Ordenar de mayor a menor

dvdrental/postgres@server1

Query Editor Query History

```
select
    concat(actor.first_name, ' ', actor.last_name) as actor_name,
    count(distinct film.film_id) as num_films
from actor
    inner join film_actor on actor.actor_id = film_actor.actor_id
    inner join film on film_actor.film_id = film.film_id
group by actor.actor_id
order by num_films desc;
```

Data Output Explain Messages Notifications

	actor_name text	num_films bigint
1	Gina Degeneres	42
2	Walter Torn	41
3	Mary Keitel	40
4	Matthew Carrey	39
5	Sandra Kilmer	37

```
select
        concat(actor.first_name, ' ', actor.last_name) as actor_name,
        count(distinct film.film_id) as num_films
from actor
        inner join film_actor on actor.actor_id = film_actor.actor_id
        inner join film on film_actor.film_id = film.film_id
group by actor.actor_id
order by num_films desc;
```

4. Proporciona una SQL que muestre los siguientes datos:

- - Película
- - Número de veces alquilada

dvdrental/postgres@server1

Query Editor Query History

```sql
select
    film.title,
    count(distinct rental.rental_id) as num_rentals
from film
    inner join inventory on film.film_id = inventory.film_id
    inner join rental on inventory.inventory_id = rental.inventory_id
group by film.film_id
order by num_rentals desc;
```

Data Output Explain Messages Notifications

	title character varying (255)	num_rentals bigint
1	Bucket Brotherhood	34
2	Rocketeer Mother	33
3	Grit Clockwork	32
4	Scalawag Duck	32
5	Juggler Hardly	32

```sql
select
        film.title,
        count(distinct rental.rental_id) as num_rentals
from film
        inner join inventory on film.film_id = inventory.film_id
        inner join rental on inventory.inventory_id = rental.inventory_id
group by film.film_id
order by num_rentals desc;
```

5. Proporciona una SQL que muestre los siguientes datos:

- - Película
- - Dinero recaudado por película

dvdrental/postgres@server1

Query Editor Query History

```
select
    film.title,
    sum(payment.amount) as amount
from film
    inner join inventory on film.film_id = inventory.film_id
    inner join rental on inventory.inventory_id = rental.inventory_id
    inner join payment on rental.rental_id = payment.rental_id
group by film.film_id
order by amount desc;
```

Data Output Explain Messages Notifications

	title character varying (255)	amount numeric
1	Telegraph Voyage	215.75
2	Zorro Ark	199.72
3	Wife Turn	198.73
4	Innocent Usual	191.74
5	Hustler Party	190.78

```
select
        film.title,
        sum(payment.amount) as amount
from film
        inner join inventory on film.film_id = inventory.film_id
        inner join rental on inventory.inventory_id = rental.inventory_id
        inner join payment on rental.rental_id = payment.rental_id
group by film.film_id
order by amount desc;
```

6. Proporciona una SQL que muestre los siguientes datos:

- Nombre del mejor cliente (mayor gasto)

dvdrental/postgres@server1

Query Editor Query History

```
select
    concat(customer.first_name, ' ', customer.last_name) as customer_name,
    sum(payment.amount) as total_expenditure
from customer
    inner join payment on customer.customer_id = payment.customer_id
group by customer.customer_id
order by total_expenditure desc
limit 1;
```

Data Output Explain Messages Notifications

	customer_name text	total_expenditure numeric
1	Eleanor Hunt	211.55

```
select
        concat(customer.first_name, ' ', customer.last_name) as customer_name,
        sum(payment.amount) as total_expenditure
from customer
        inner join payment on customer.customer_id = payment.customer_id
group by customer.customer_id
order by total_expenditure desc
limit 1;
```

7. Proporciona una SQL que muestre los siguientes datos:

- Nombre del mejor cliente (mayor número alquileres)

dvdrental/postgres@server1

Query Editor Query History

```
select
    concat(customer.first_name, ' ', customer.last_name) as customer_name,
    count(rental.rental_id) as total_rentals
from customer
    inner join rental on customer.customer_id = rental.customer_id
group by customer.customer_id
order by total_rentals desc
limit 1;
```

Data Output Explain Messages Notifications

	customer_name text	total_rentals bigint
1	Eleanor Hunt	46

```
select
        concat(customer.first_name, ' ', customer.last_name) as customer_name,
        count(rental.rental_id) as total_rentals
from customer
        inner join rental on customer.customer_id = rental.customer_id
group by customer.customer_id
order by total_rentals desc
limit 1;
```