







SAKILA DATABASE SQL PROJECT











- Chiranjeev Mishra

Easy Questions:

List films with their respective categories and order by Title:



No limit










Query




Query History



```
1 select f."title",ca."name" from public."film" as "f"
2 join public."film_category" as "fc"
3 on f."film_id" = fc."film_id"
4 join public."category" as "ca"
5 on ca."category_id" = fc."category_id"
6 order by f."title";|
```



Count the number of films in each category and order by filmcount in descending order:






No limit



E







Query

Query History

1

```
select ca."name",count(*) as category_count from public."film_category" as "fc"
```

2

```
join public."category" as "ca"
```

3

```
on ca."category_id" = fc."category_id"
```

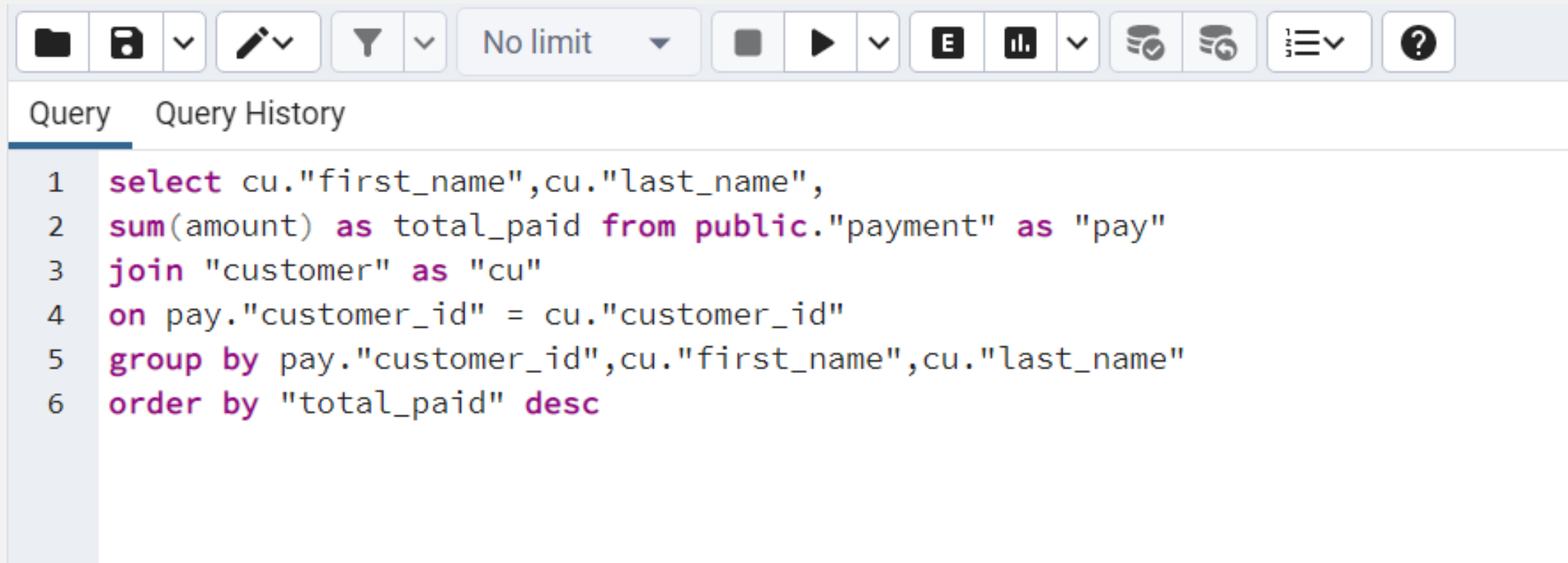
4

```
group by ca."name"
```

5

```
order by "category_count" desc
```

Find the total rental amount paid by each customer and order by total amount in descending order:

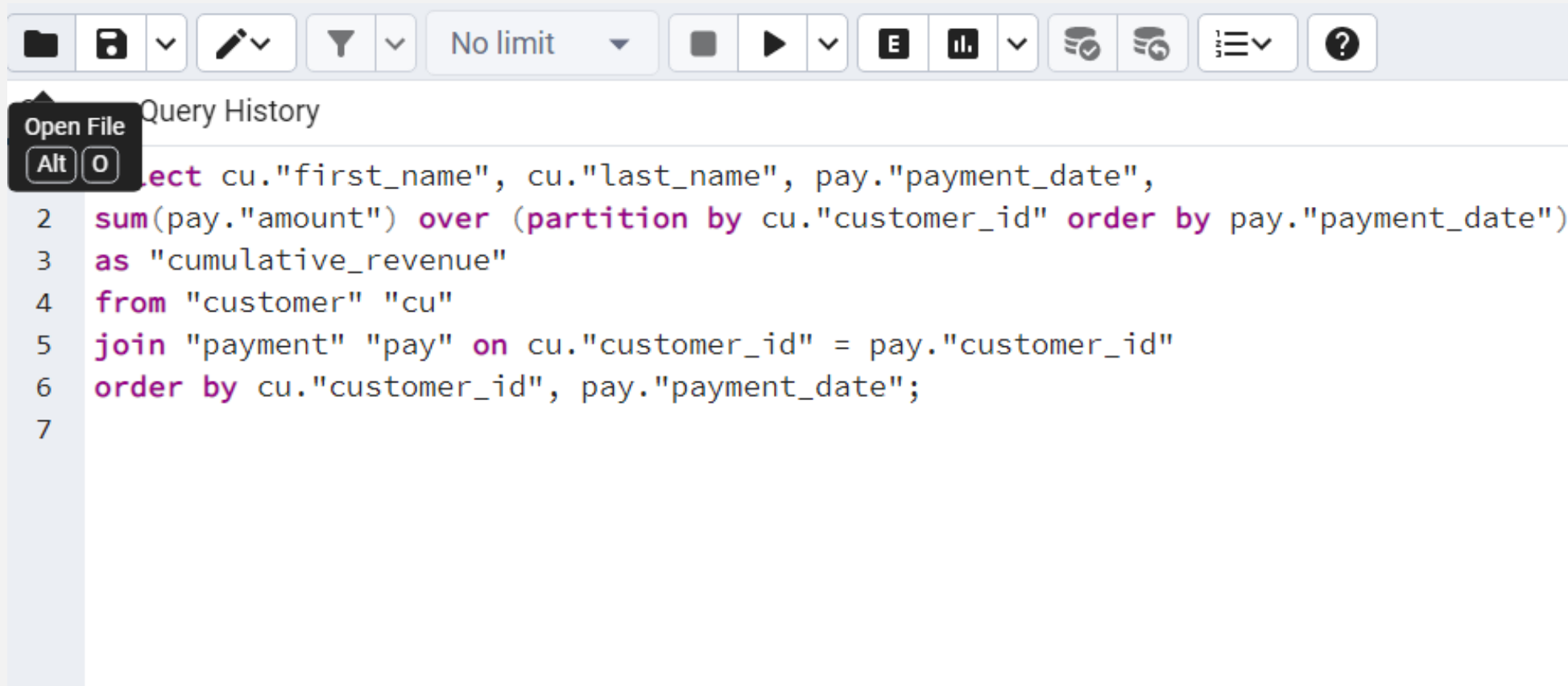


The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, editing, filtering, execution, and visualization. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query with line numbers 1 through 6. The query is a SELECT statement that joins the 'customer' and 'payment' tables, groups the results by customer ID, first name, and last name, and orders them by the total amount paid in descending order.

```
1 select cu."first_name",cu."last_name",  
2 sum(amount) as total_paid from public."payment" as "pay"  
3 join "customer" as "cu"  
4 on pay."customer_id" = cu."customer_id"  
5 group by pay."customer_id",cu."first_name",cu."last_name"  
6 order by "total_paid" desc
```

Moderate Questions:


Calculate the cumulative revenue generated by each customer and order by first customerid then paymentdate:



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, filters, execution, and help. Below the toolbar is a 'Query History' panel. The main editor area contains a SQL query that calculates the cumulative revenue for each customer, ordered by customer ID and then payment date.

```
1 select cu."first_name", cu."last_name", pay."payment_date",  
2 sum(pay."amount") over (partition by cu."customer_id" order by pay."payment_date")  
3 as "cumulative_revenue"  
4 from "customer" "cu"  
5 join "payment" "pay" on cu."customer_id" = pay."customer_id"  
6 order by cu."customer_id", pay."payment_date";  
7
```

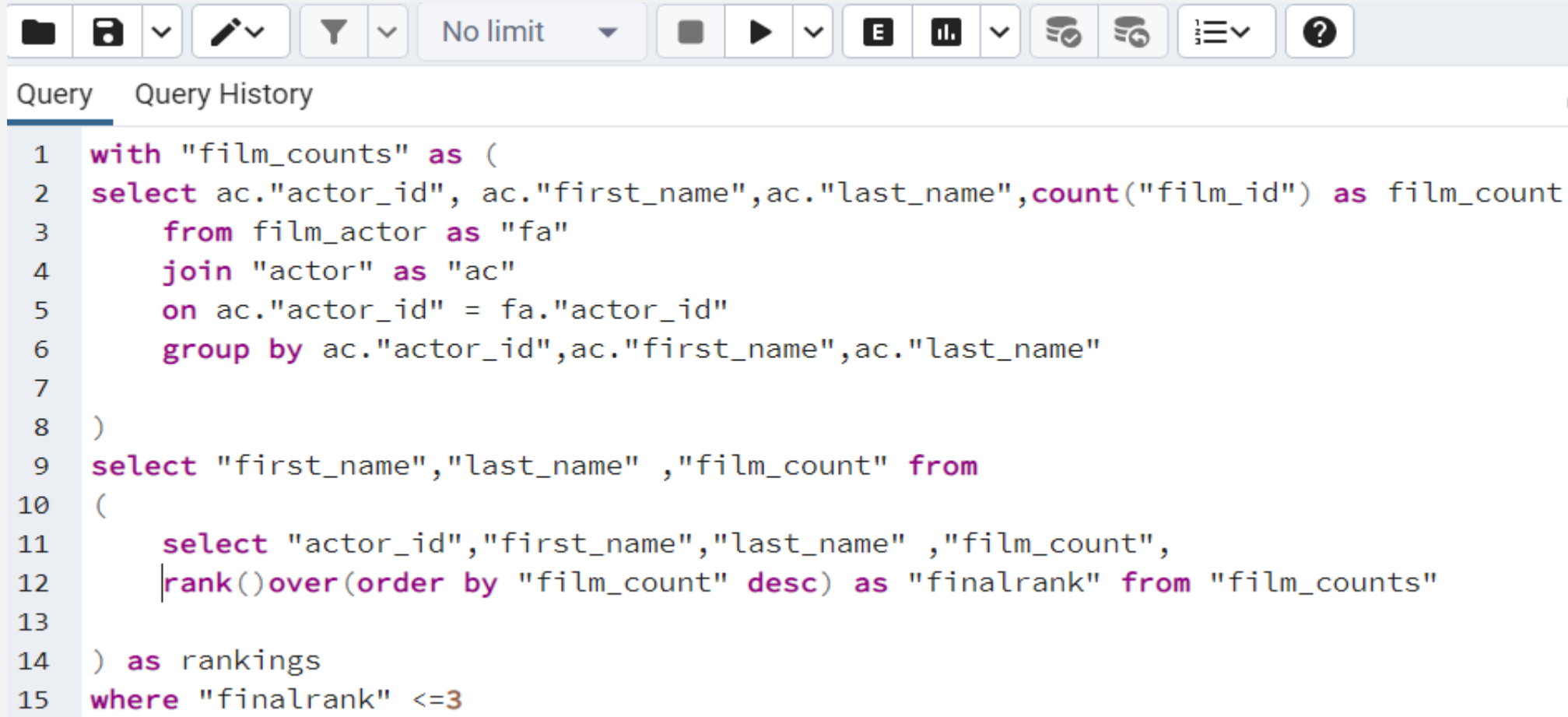
Find the average film rental rate for each store:

A toolbar for a database query editor. It contains icons for file operations (folder, save, dropdown), editing (pencil, dropdown), filtering (funnel, dropdown), a 'No limit' button with a dropdown, execution (square, play, dropdown), a table icon with 'E', a bar chart icon with a dropdown, connection status (two server icons with checkmarks), a list icon with a dropdown, and a help icon (question mark).

```
1 with "store_rentals" as (  
2     select i."store_id", f."rental_rate"  
3     from "inventory" "i"  
4     join "film" "f" on i."film_id" = f."film_id"  
5 )  
6 select s."store_id", avg(sr."rental_rate") as "avg_rental_rate"  
7 from "store_rentals" "sr"  
8 join "store" "s" on sr."store_id" = s."store_id"  
9 group by s."store_id";  
10 |
```

Difficult Questions:







Find the top 3 actors with the most appearances:






The image shows a SQL query editor interface. At the top, there is a toolbar with various icons for file operations, filters, and execution. Below the toolbar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query. The query is as follows:

```
1 with "film_counts" as (  
2   select ac."actor_id", ac."first_name",ac."last_name",count("film_id") as film_count  
3     from film_actor as "fa"  
4     join "actor" as "ac"  
5     on ac."actor_id" = fa."actor_id"  
6     group by ac."actor_id",ac."first_name",ac."last_name"  
7  
8   )  
9   select "first_name","last_name" ,"film_count" from  
10  (  
11    select "actor_id","first_name","last_name" ,"film_count",  
12    |rank()over(order by "film_count" desc) as "finalrank" from "film_counts"  
13  
14  ) as rankings  
15  where "finalrank" <=3
```

Find the average rental duration per film category:









No limit



E

⏏



Query

Query History

```
1 with "rental_duration" as (  
2     select f."film_id", age(r."return_date", r."rental_date") as "rental_duration"  
3     from public."rental" "r"  
4     join "inventory" "i" on r."inventory_id" = i."inventory_id"  
5     join "film" "f" on i."film_id" = f."film_id"  
6 )  
7 select ca."name" as "category",  
8 Extract(day from avg(rd."rental_duration")) as "avg_duration"  
9 from "rental_duration" "rd"  
10 join "film_category" "fc" on rd."film_id" = fc."film_id"  
11 join "category" "ca" on fc."category_id" = ca."category_id"  
12 group by ca."name";  
13
```

Data Output

Messages

Notifications