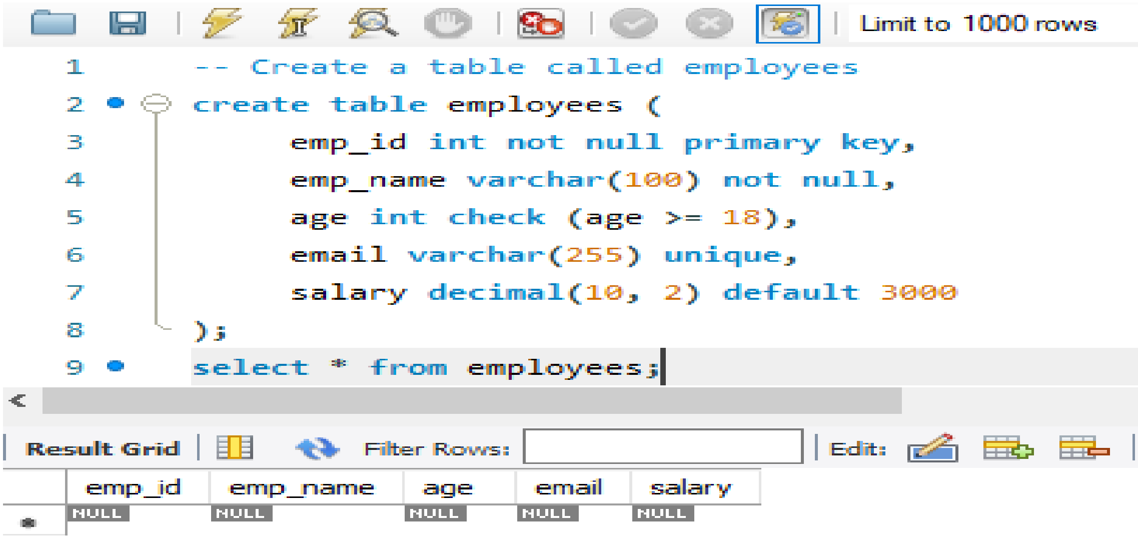
SQL Assignment Answers

1. Create a table called employees with the following structure; emp_id (integer, should not be NULL and should be a primary key) emp_name (text, should not be NULL) age (integer, should have a check constraint to ensure the age is at least 18) email (text, should be unique for each employee) salary (decimal, with a default value of 30,000). Write the SQL query to create the above table with all constraints.



```sql
-- Create a table called employees
create table employees (
        emp_id int not null primary key,
        emp_name varchar(100) not null,
        age int check (age >= 18),
        email varchar(255) unique,
        salary decimal(10, 2) default 3000
);
select * from employees;
```

emp_id	emp_name	age	email	salary
NULL	NULL	NULL	NULL	NULL

2. Explain the purpose of constraints and how they help maintain data integrity in a database. Provide examples of common types of constraints.

Constraints in a database are rules that enforce data integrity by ensuring data accuracy, consistency, and validity. They prevent invalid data from being stored and help enforce business rules.

Common Types of Constraints:

1. **Primary Key**: Ensures each record is unique and not null.
2. **Foreign Key**: Links tables and ensures valid references.
3. **Unique**: Ensures all values in a column are distinct.
4. **Not Null**: Prevents null values in a column.
5. **Check**: Ensures data meets specific conditions.
6. **Default**: Assigns a default value if none is provided.

These constraints maintain data reliability and consistency in the database.

3. Why would you apply the NOT NULL constraint to a column? Can a primary key contain NULL values? Justify your answer.

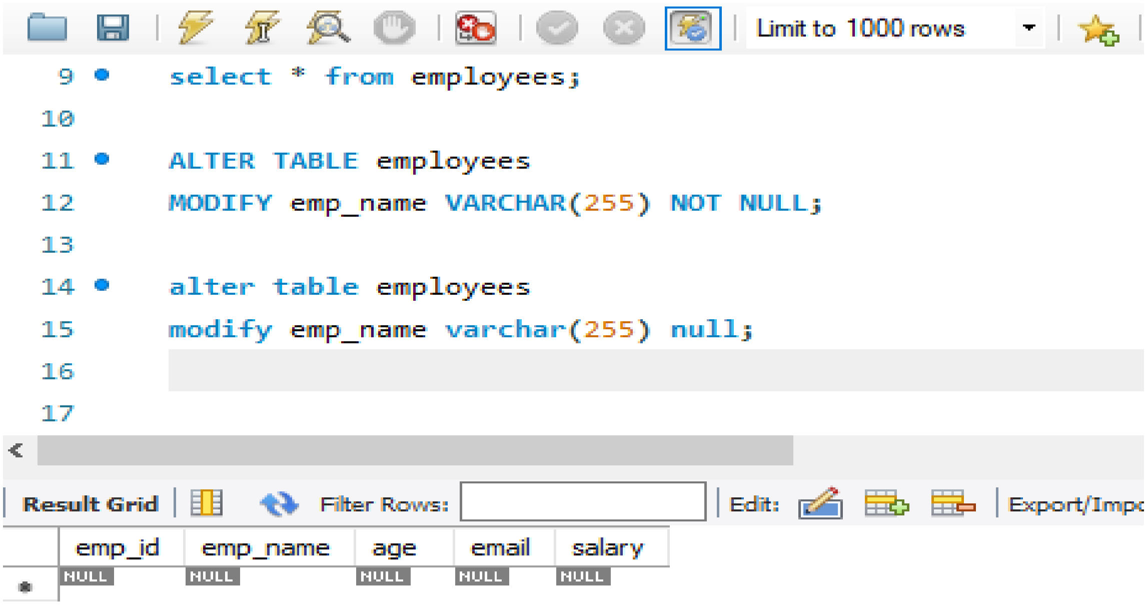
A **NOT NULL** constraint ensures a column always has a valid value, preventing empty entries. This guarantees that the column cannot be left empty or contain NULL values, which helps maintain data completeness.

A **primary key** cannot contain NULL values because it must uniquely identify each record, and NULL represents missing data, which would violate this uniqueness.

4. Explain the steps and SQL commands used to add or remove constraints on an existing table. Provide an example for both adding and removing a constraint.

We use ALTER TABLE to add a rule (constraint) to ensure data integrity.

We use ALTER TABLE to remove an existing constraint.



```
select * from employees;

ALTER TABLE employees
MODIFY emp_name VARCHAR(255) NOT NULL;

alter table employees
modify emp_name varchar(255) null;
```

emp_id	emp_name	age	email	salary
NULL	NULL	NULL	NULL	NULL

5. Explain the consequences of attempting to insert, update, or delete data in a way that violates constraints. Provide an example of an error message that might occur when violating a constraint.

When you attempt to insert, update, or delete data in a way that violates constraints, the database management system (DBMS) will reject the operation and return an error message. This helps maintain data integrity and prevents invalid data from being stored.

Consequences of Violating Constraints:

1. Data Integrity Issues: Violating constraints can lead to inconsistent or incorrect data in the database, undermining its reliability.
2. Transaction Failure: The operation (insert, update, delete) will fail, and the DBMS will roll back any changes made during that transaction, ensuring the database remains in a consistent state.
3. Error Messages: The DBMS will provide specific error messages indicating which constraint was violated, helping developers troubleshoot the issue.

6. You created a products table without constraints as follows:

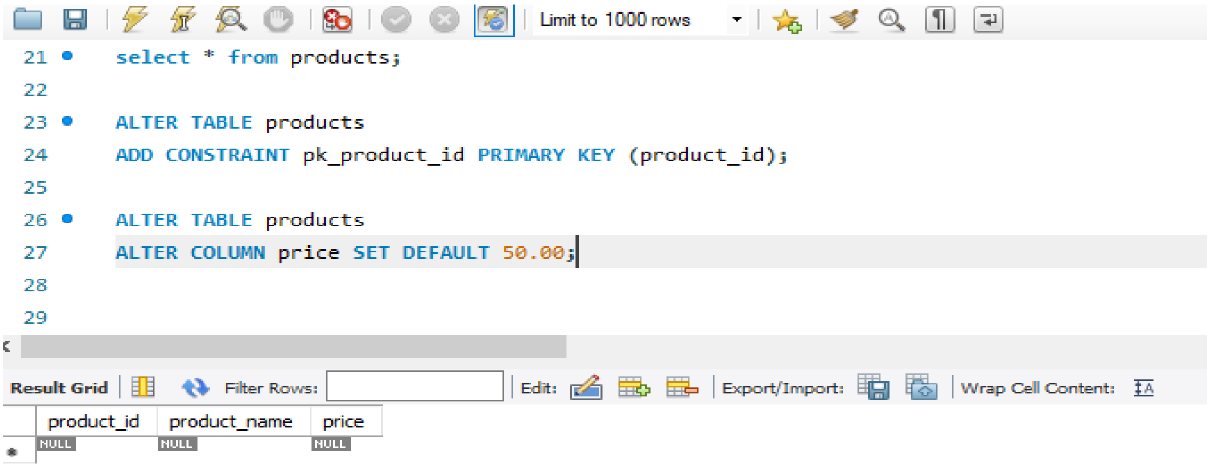
CREATE TABLE products (

product_id INT,

product_name VARCHAR(50),

price DECIMAL(10, 2));

Now, you realise that; The product_id should be a primary key The price should have a default value of 50.00.

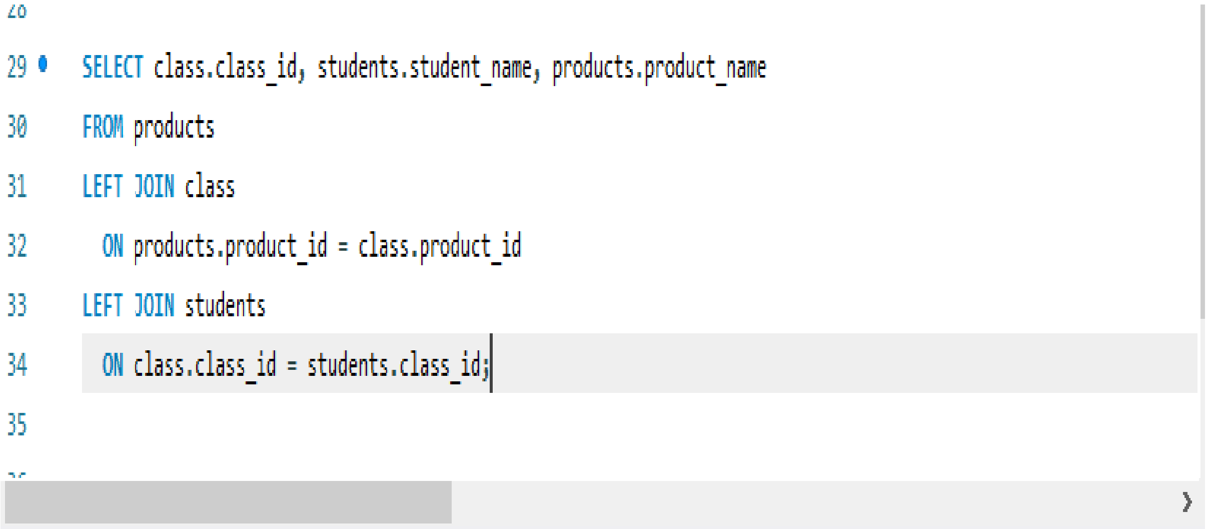


```sql
select * from products;

ALTER TABLE products
ADD CONSTRAINT pk_product_id PRIMARY KEY (product_id);

ALTER TABLE products
ALTER COLUMN price SET DEFAULT 50.00;
```

product_id	product_name	price
NULL	NULL	NULL

7. You have two tables:

Write a query that shows all order_id, customer_name, and product_name, ensuring that all products are listed even if they are not associated with an order Hint: (use INNER JOIN and LEFT JOIN)



```sql
SELECT class.class_id, students.student_name, products.product_name
FROM products
LEFT JOIN class
  ON products.product_id = class.product_id
LEFT JOIN students
  ON class.class_id = students.class_id;
```

9. Given the following tables:

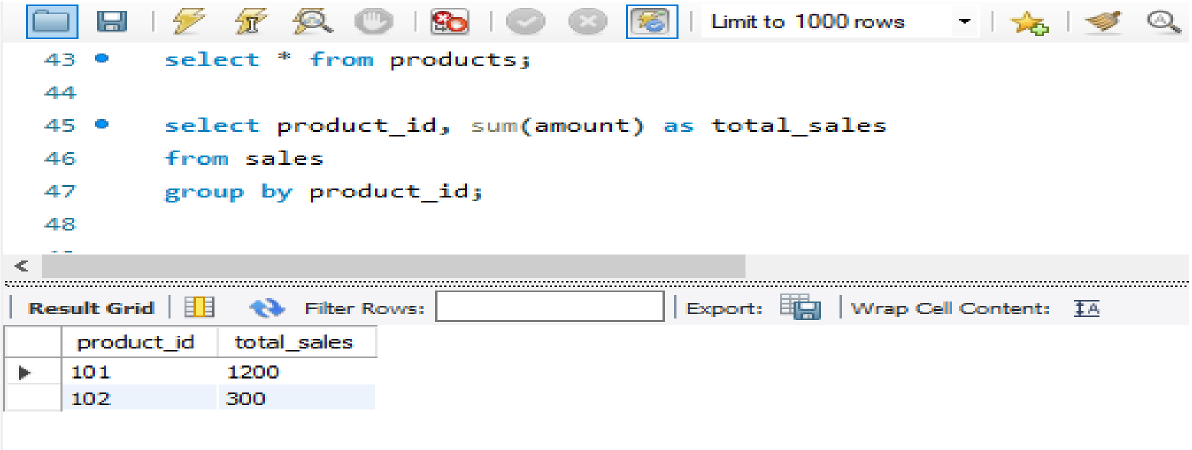
- Sales:

```
+------------+-------------+-------------+
| sale_id    | product_id  | amount      |
+------------+-------------+-------------+
| 1          | 101         | 500         |
| 2          | 102         | 300         |
| 3          | 101         | 700         |
+------------+-------------+-------------+
```

- Products:

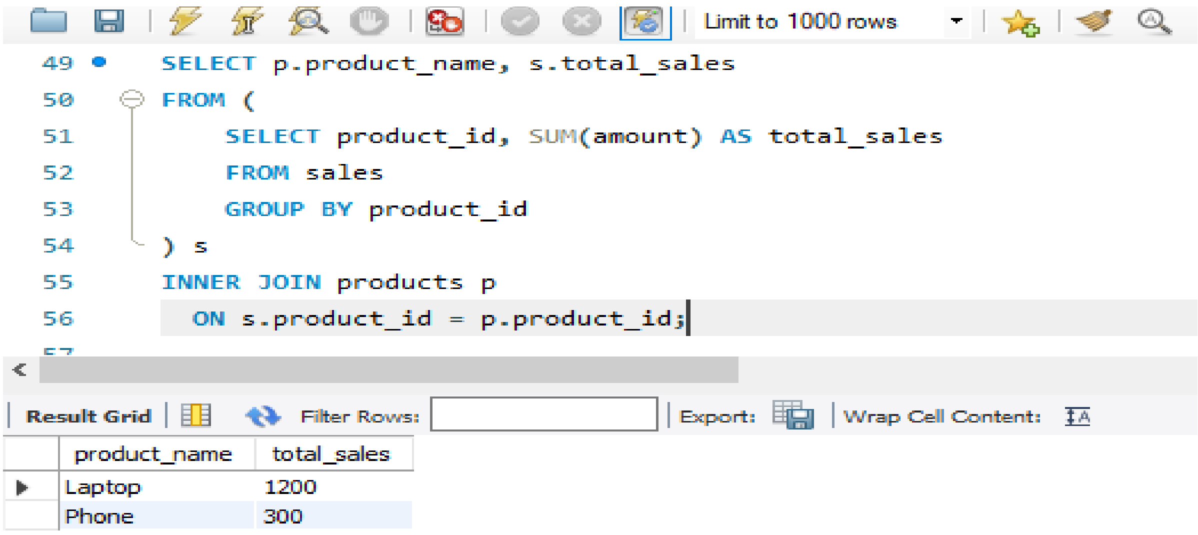
```
+------------+------------+
| product_id | product_name|
+------------+------------+
| 101        | Laptop      |
| 102        | Phone       |
+------------+------------+
```

Write a query to find the total sales amount for each product using an INNER JOIN and the SUM() function.



```
select * from products;

select product_id, sum(amount) as total_sales
from sales
group by product_id;
```

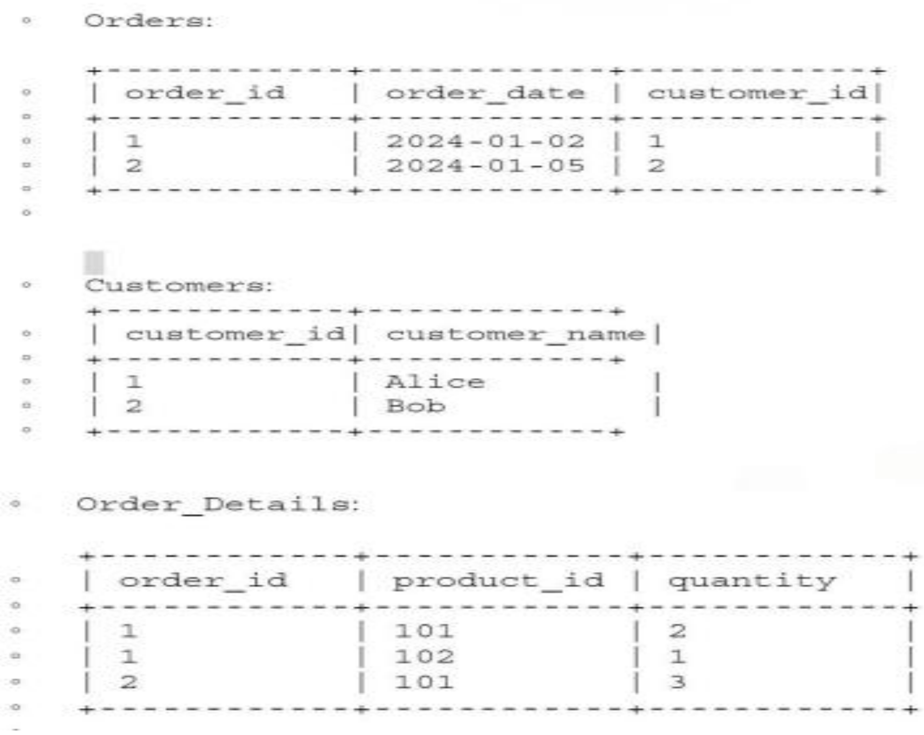
product_id	total_sales
101	1200
102	300



```
SELECT p.product_name, s.total_sales
FROM (
    SELECT product_id, SUM(amount) AS total_sales
    FROM sales
    GROUP BY product_id
) s
INNER JOIN products p
  ON s.product_id = p.product_id;
```

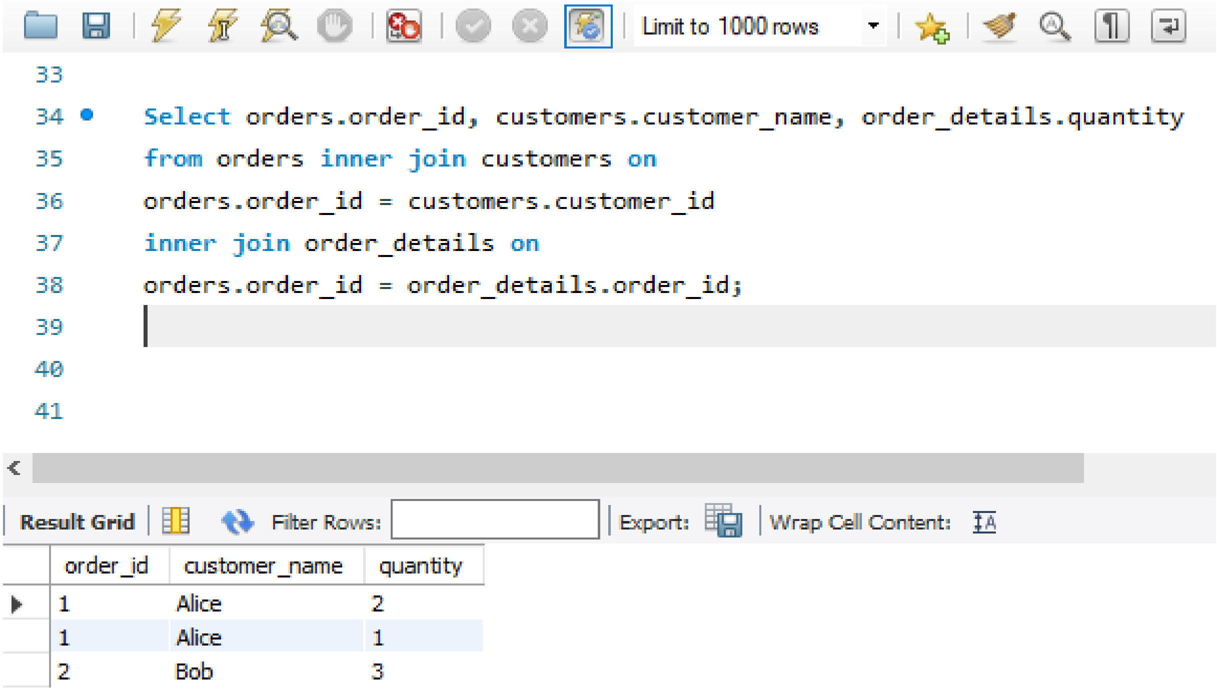
product_name	total_sales
Laptop	1200
Phone	300

10. You are given three tables:



```
◦ Orders:

  +-------------+-------------+-------------+
◦ | order_id    | order_date  | customer_id|
◦ +-------------+-------------+-------------+
◦ | 1           | 2024-01-02  | 1           |
◦ | 2           | 2024-01-05  | 2           |
◦ +-------------+-------------+-------------+
◦

◦ Customers:
  +-------------+-------------+
◦ | customer_id| customer_name|
◦ +-------------+-------------+
◦ | 1           | Alice       |
◦ | 2           | Bob         |
◦ +-------------+-------------+

◦ Order_Details:

  +-------------+-------------+-------------+
◦ | order_id    | product_id  | quantity    |
◦ +-------------+-------------+-------------+
◦ | 1           | 101         | 2           |
◦ | 1           | 102         | 1           |
◦ | 2           | 101         | 3           |
◦ +-------------+-------------+-------------+
```

Write a query to display the order_id, customer_name, and the quantity of products ordered by each customer using an INNER JOIN between all three tables.

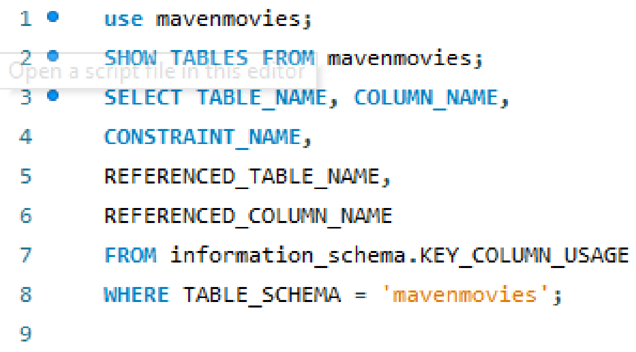


```sql
Select orders.order_id, customers.customer_name, order_details.quantity
from orders inner join customers on
orders.order_id = customers.customer_id
inner join order_details on
orders.order_id = order_details.order_id;
```

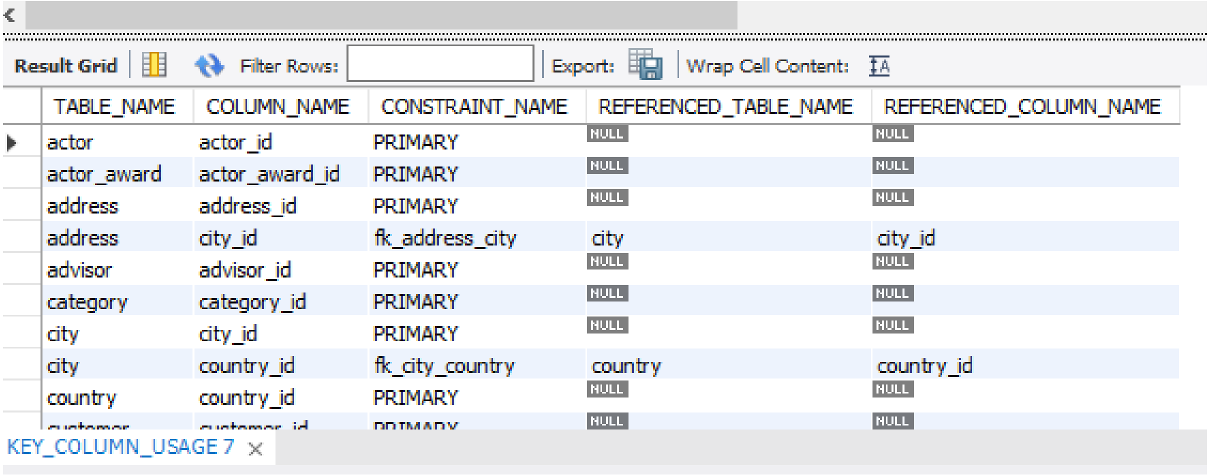
order_id	customer_name	quantity
1	Alice	2
1	Alice	1
2	Bob	3

SQL Commands

1-Identify the primary keys and foreign keys in maven movies db. Discuss the differences



```
use mavenmovies;
SHOW TABLES FROM mavenmovies;
SELECT TABLE_NAME, COLUMN_NAME,
CONSTRAINT_NAME,
REFERENCED_TABLE_NAME,
REFERENCED_COLUMN_NAME
FROM information_schema.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'mavenmovies';
```

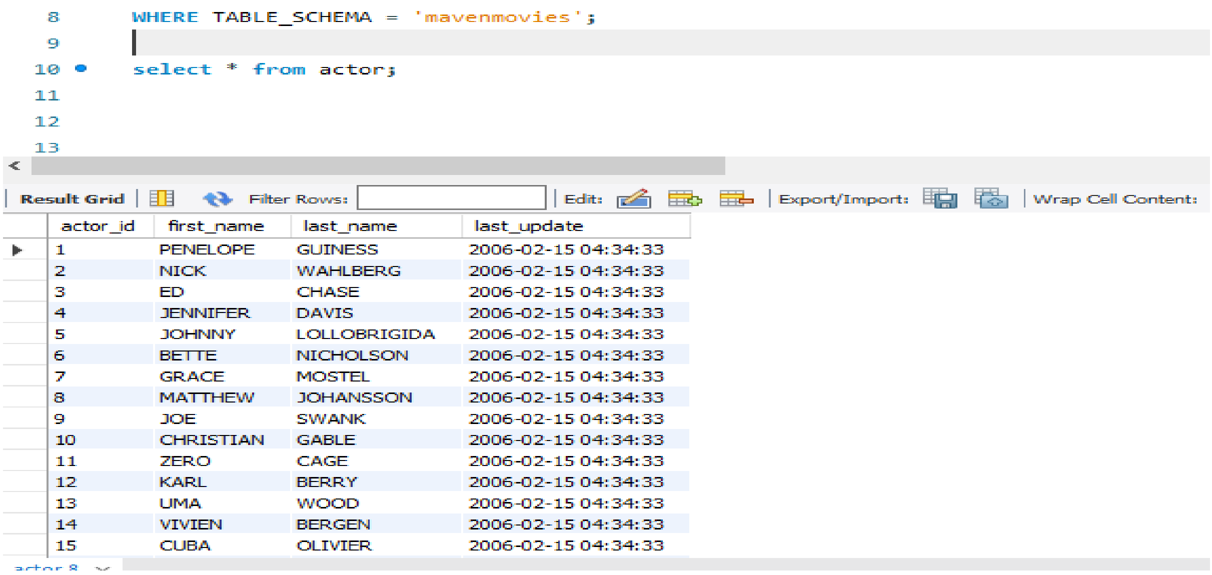


TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
actor	actor_id	PRIMARY	NULL	NULL
actor_award	actor_award_id	PRIMARY	NULL	NULL
address	address_id	PRIMARY	NULL	NULL
address	city_id	fk_address_city	city	city_id
advisor	advisor_id	PRIMARY	NULL	NULL
category	category_id	PRIMARY	NULL	NULL
city	city_id	PRIMARY	NULL	NULL
city	country_id	fk_city_country	country	country_id
country	country_id	PRIMARY	NULL	NULL
customer	customer_id	PRIMARY	NULL	NULL

Difference:

- Primary Key: Uniquely identifies a row in its own table.
- Foreign Key: Establishes a relationship between two tables by linking to the primary key of another table.

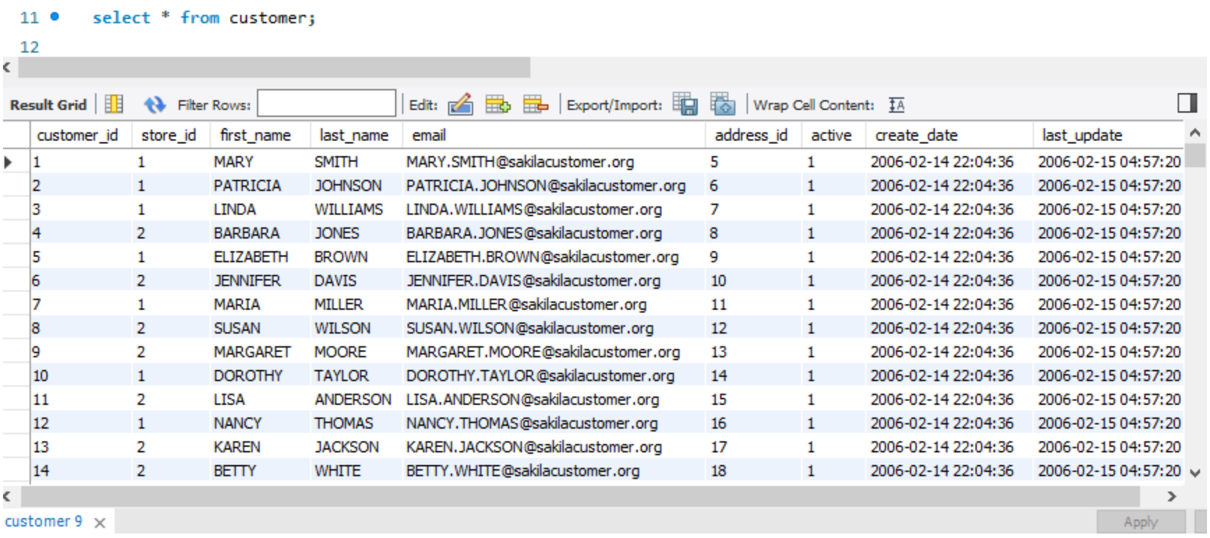
2- List all details of actors



```
WHERE TABLE_SCHEMA = 'mavenmovies';

select * from actor;
```

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 04:34:33
2	NICK	WAHLBERG	2006-02-15 04:34:33
3	ED	CHASE	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
6	BETTE	NICHOLSON	2006-02-15 04:34:33
7	GRACE	MOSTEL	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
9	JOE	SWANK	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
11	ZERO	CAGE	2006-02-15 04:34:33
12	KARL	BERRY	2006-02-15 04:34:33
13	UMA	WOOD	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
15	CUBA	OLIVIER	2006-02-15 04:34:33

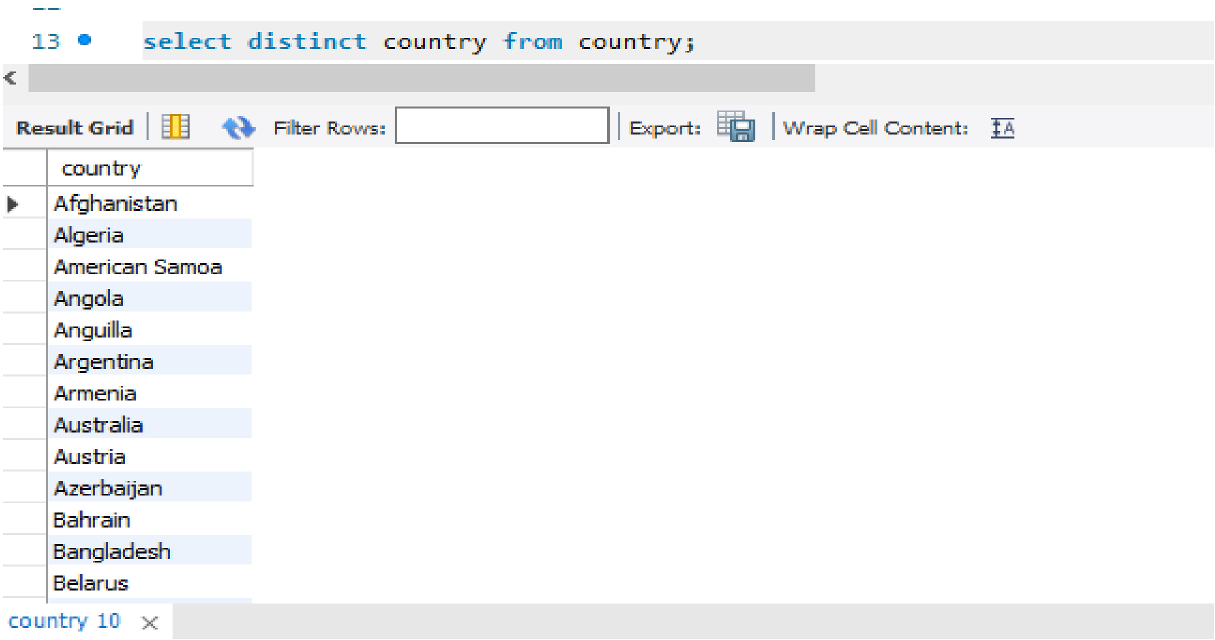
3 -List all customer information from DB.



```
select * from customer;
```

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
▶	1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacustomer.org	16	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustomer.org	17	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:36	2006-02-15 04:57:20

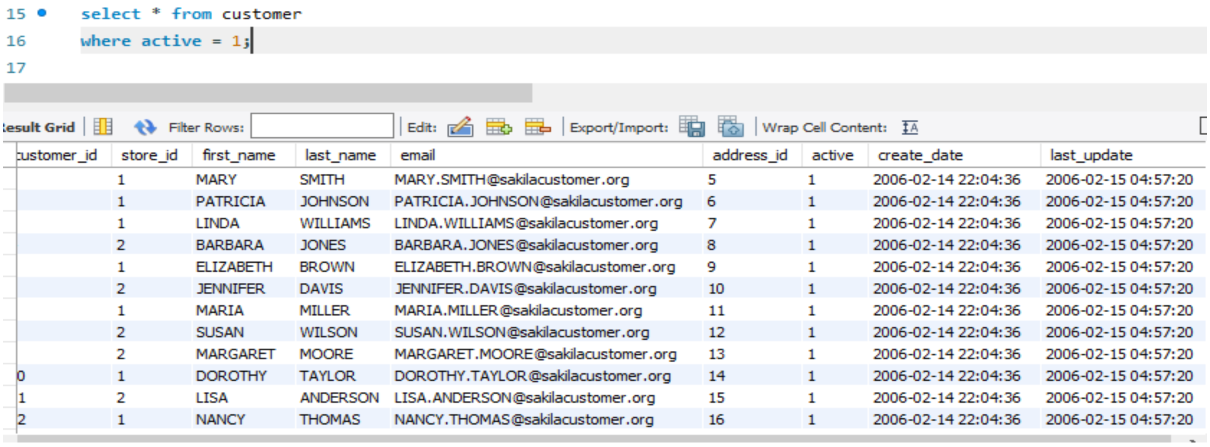
4 -List different countries



```
--
select distinct country from country;
```

	country
▶	Afghanistan
	Algeria
	American Samoa
	Angola
	Anguilla
	Argentina
	Armenia
	Australia
	Austria
	Azerbaijan
	Bahrain
	Bangladesh
	Belarus

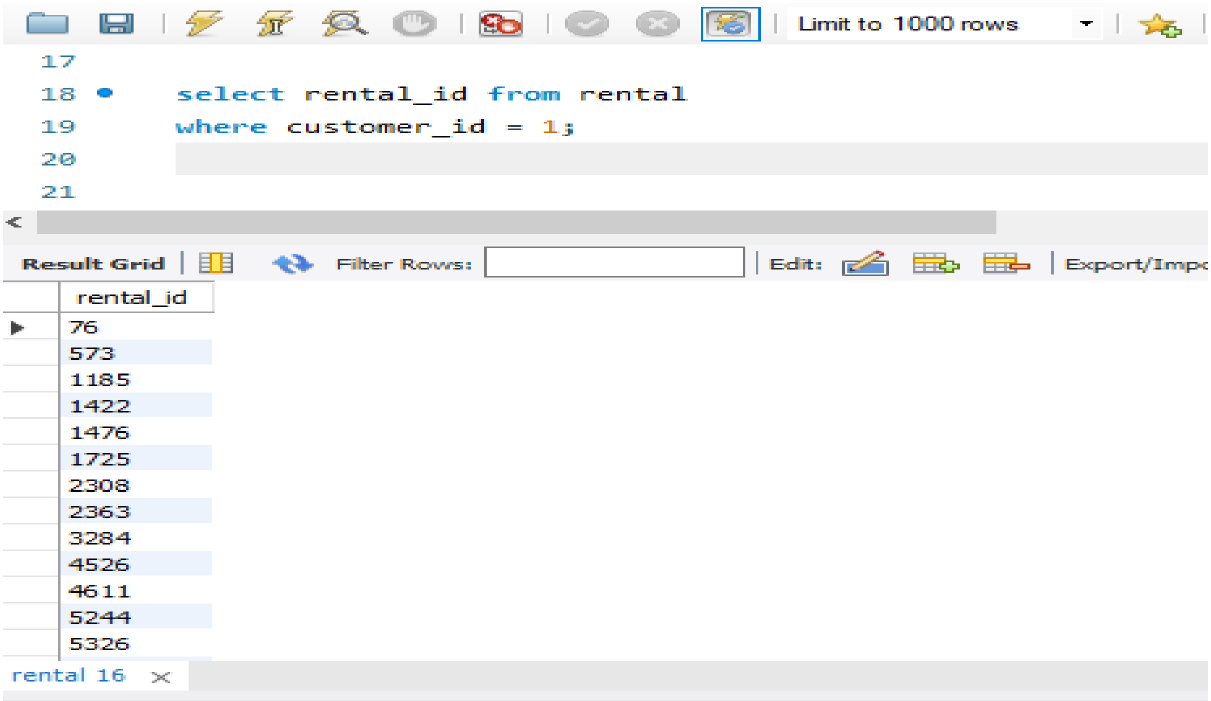
5 -Display all active customers.



```
select * from customer
where active = 1;
```

ustomer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
0	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
1	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	NANCY	THOMAS	NANCY.THOMAS@sakilacustomer.org	16	1	2006-02-14 22:04:36	2006-02-15 04:57:20

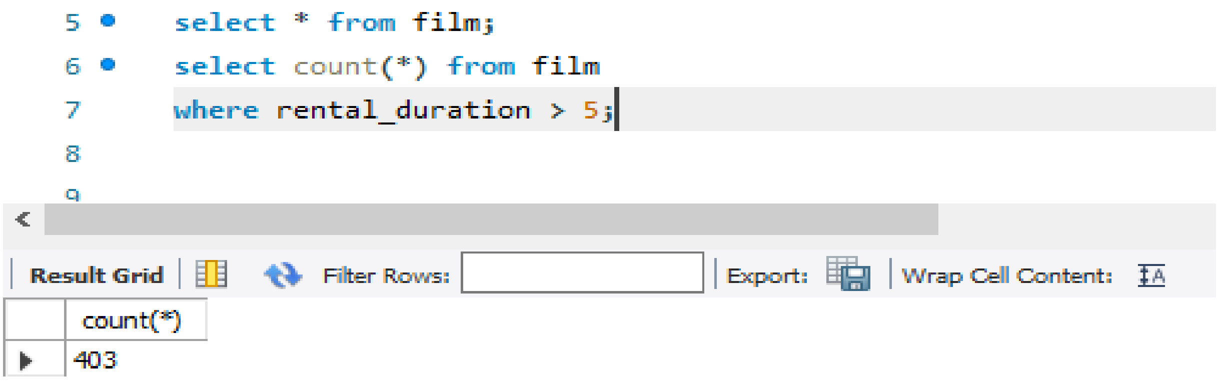
6 -List of all rental IDs for customer with ID 1.



```
select rental_id from rental
where customer_id = 1;
```

```
rental_id
76
573
1185
1422
1476
1725
2308
2363
3284
4526
4611
5244
5326
```

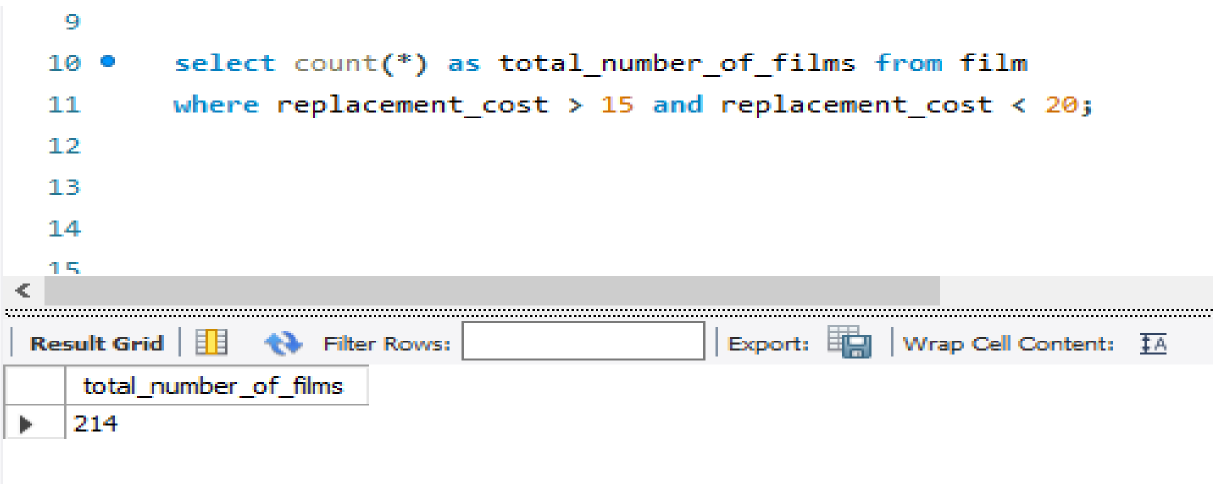
7 - Display all the films whose rental duration is greater than 5.



```
select * from film;
select count(*) from film
where rental_duration > 5;
```

```
count(*)
403
```

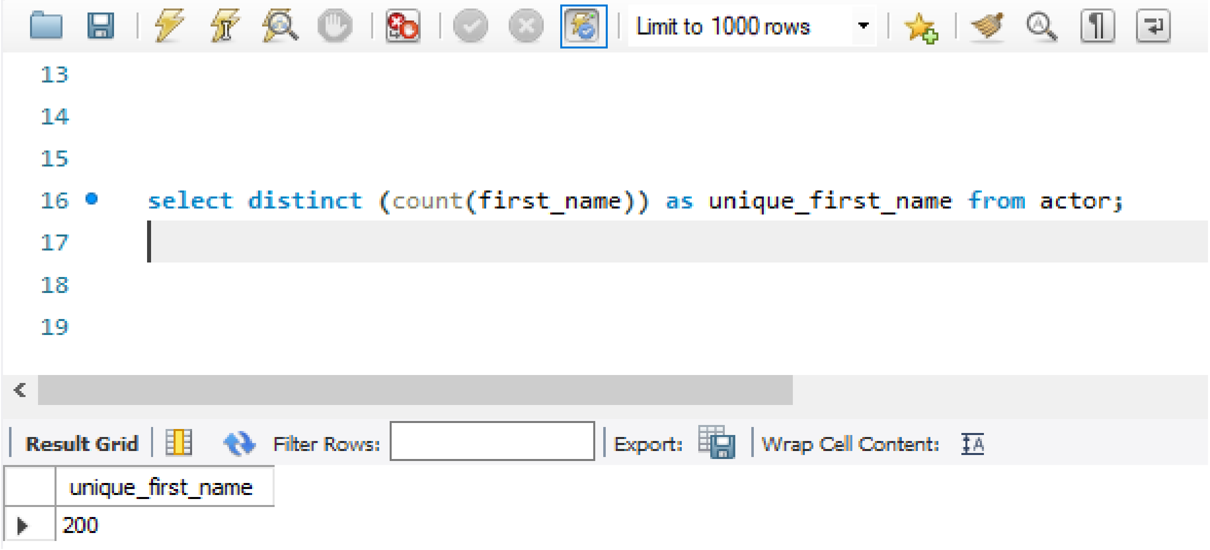
8 - List the total number of films whose replacement cost is greater than $15 and less than $20.



```
select count(*) as total_number_of_films from film
where replacement_cost > 15 and replacement_cost < 20;
```

```
total_number_of_films
214
```

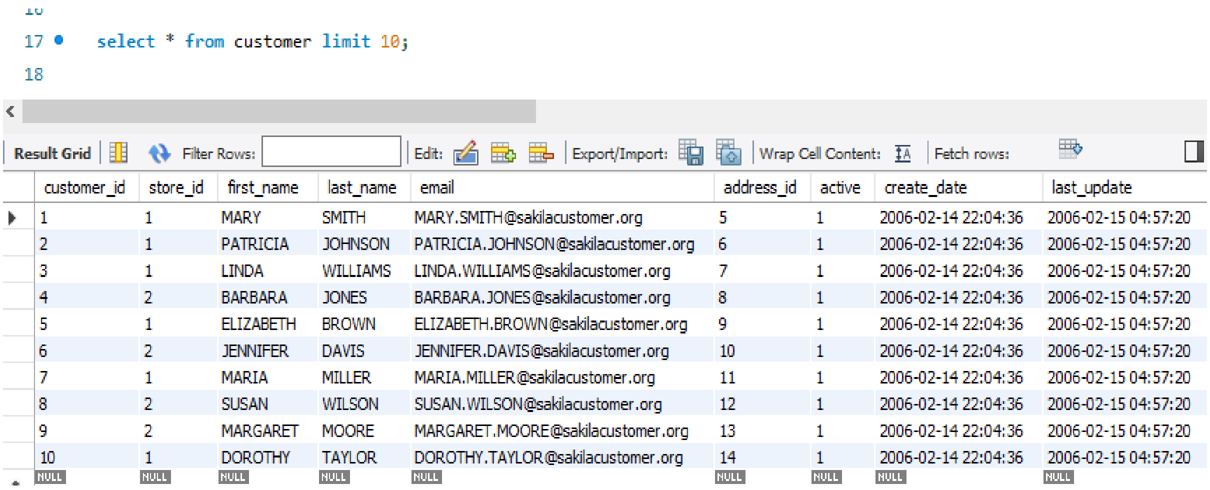
9 - Display the count of unique first names of actors.



```sql
select distinct (count(first_name)) as unique_first_name from actor;
```

unique_first_name
200

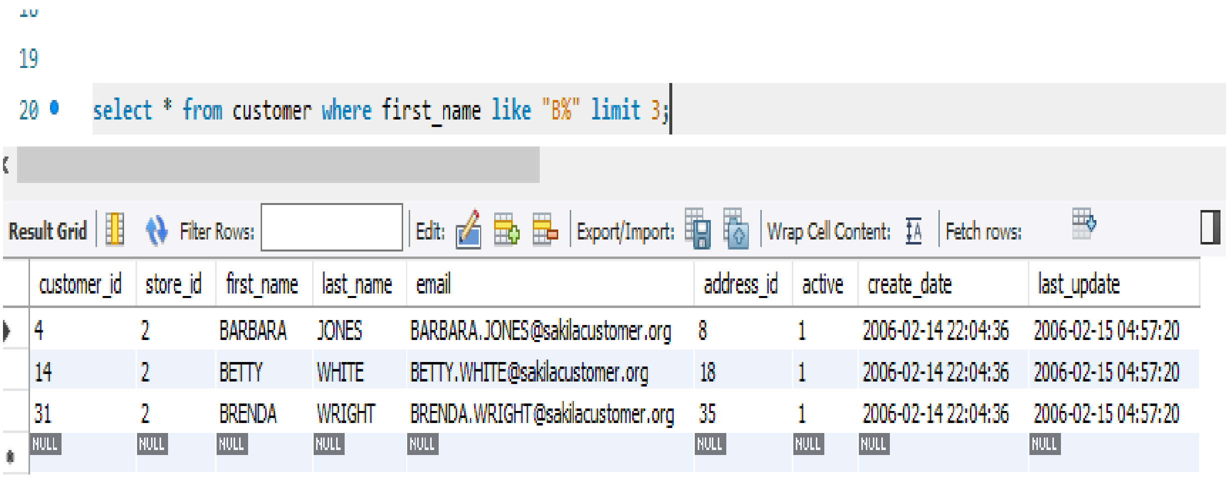
10- Display the first 10 records from the customer table.



```sql
select * from customer limit 10;
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

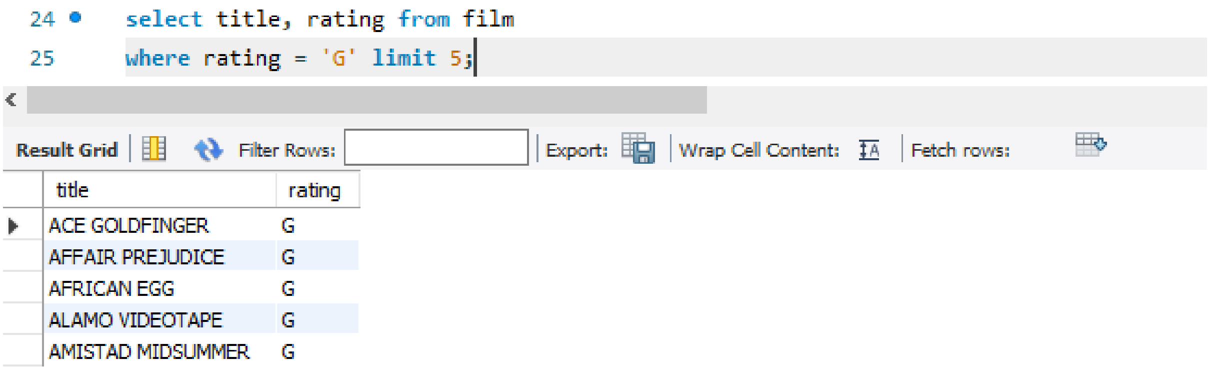
11 - Display the first 3 records from the customer table whose first name starts with ‘b’.



```sql
select * from customer where first_name like "B%" limit 3;
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:36	2006-02-15 04:57:20
31	2	BRENDA	WRIGHT	BRENDA.WRIGHT@sakilacustomer.org	35	1	2006-02-14 22:04:36	2006-02-15 04:57:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

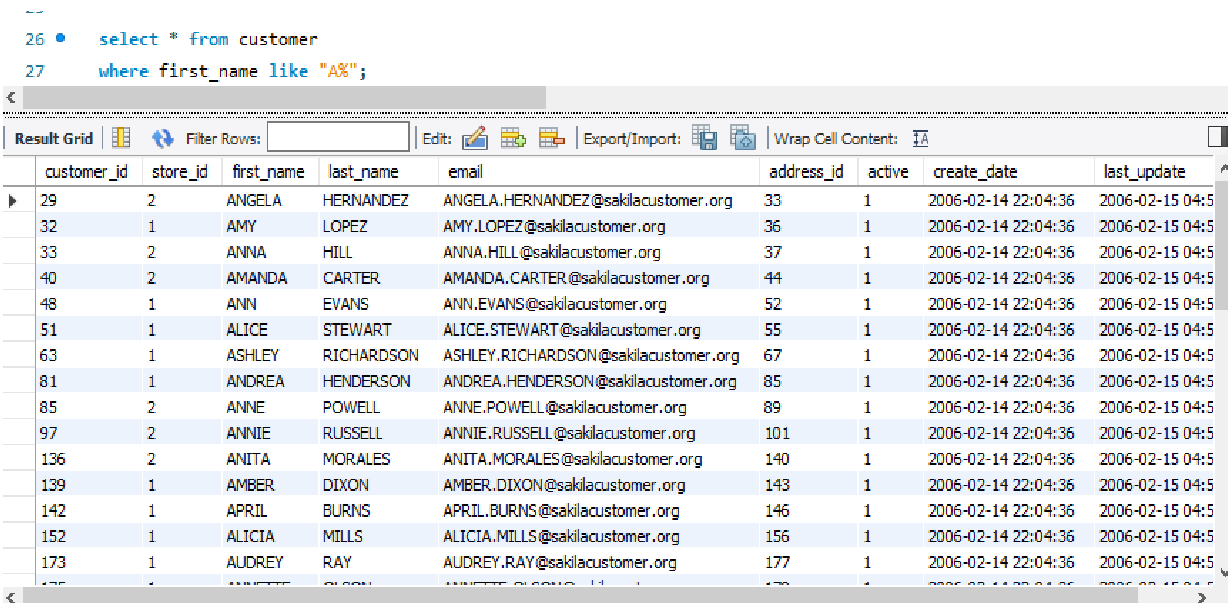
12 -Display the names of the first 5 movies that are rated as ‘G’.



```
select title, rating from film
where rating = 'G' limit 5;
```

title	rating
ACE GOLDFINGER	G
AFFAIR PREJUDICE	G
AFRICAN EGG	G
ALAMO VIDEOTAPE	G
AMISTAD MIDSUMMER	G

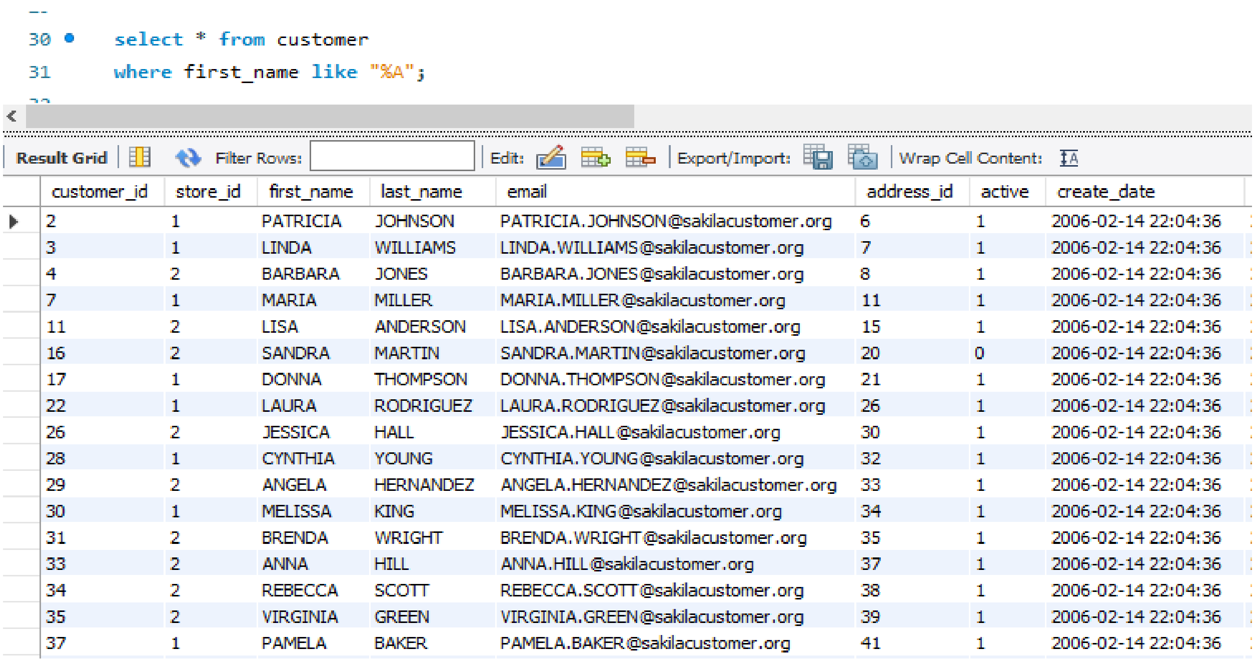
13-Find all customers whose first name starts with "a".



```
select * from customer
where first_name like "A%";
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
29	2	ANGELA	HERNANDEZ	ANGELA.HERNANDEZ@sakilacustomer.org	33	1	2006-02-14 22:04:36	2006-02-15 04:5
32	1	AMY	LOPEZ	AMY.LOPEZ@sakilacustomer.org	36	1	2006-02-14 22:04:36	2006-02-15 04:5
33	2	ANNA	HILL	ANNA.HILL@sakilacustomer.org	37	1	2006-02-14 22:04:36	2006-02-15 04:5
40	2	AMANDA	CARTER	AMANDA.CARTER@sakilacustomer.org	44	1	2006-02-14 22:04:36	2006-02-15 04:5
48	1	ANN	EVANS	ANN.EVANS@sakilacustomer.org	52	1	2006-02-14 22:04:36	2006-02-15 04:5
51	1	ALICE	STEWART	ALICE.STEWART@sakilacustomer.org	55	1	2006-02-14 22:04:36	2006-02-15 04:5
63	1	ASHLEY	RICHARDSON	ASHLEY.RICHARDSON@sakilacustomer.org	67	1	2006-02-14 22:04:36	2006-02-15 04:5
81	1	ANDREA	HENDERSON	ANDREA.HENDERSON@sakilacustomer.org	85	1	2006-02-14 22:04:36	2006-02-15 04:5
85	2	ANNE	POWELL	ANNE.POWELL@sakilacustomer.org	89	1	2006-02-14 22:04:36	2006-02-15 04:5
97	2	ANNIE	RUSSELL	ANNIE.RUSSELL@sakilacustomer.org	101	1	2006-02-14 22:04:36	2006-02-15 04:5
136	2	ANITA	MORALES	ANITA.MORALES@sakilacustomer.org	140	1	2006-02-14 22:04:36	2006-02-15 04:5
139	1	AMBER	DIXON	AMBER.DIXON@sakilacustomer.org	143	1	2006-02-14 22:04:36	2006-02-15 04:5
142	1	APRIL	BURNS	APRIL.BURNS@sakilacustomer.org	146	1	2006-02-14 22:04:36	2006-02-15 04:5
152	1	ALICIA	MILLS	ALICIA.MILLS@sakilacustomer.org	156	1	2006-02-14 22:04:36	2006-02-15 04:5
173	1	AUDREY	RAY	AUDREY.RAY@sakilacustomer.org	177	1	2006-02-14 22:04:36	2006-02-15 04:5

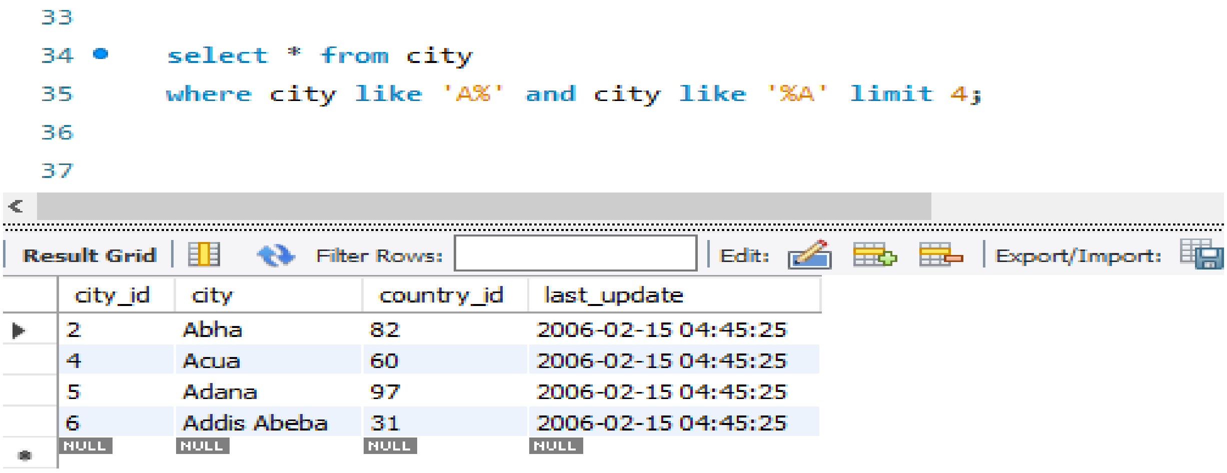
14- Find all customers whose first name ends with "a".



```
select * from customer
where first_name like "%A";
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36
16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustomer.org	20	0	2006-02-14 22:04:36
17	1	DONNA	THOMPSON	DONNA.THOMPSON@sakilacustomer.org	21	1	2006-02-14 22:04:36
22	1	LAURA	RODRIGUEZ	LAURA.RODRIGUEZ@sakilacustomer.org	26	1	2006-02-14 22:04:36
26	2	JESSICA	HALL	JESSICA.HALL@sakilacustomer.org	30	1	2006-02-14 22:04:36
28	1	CYNTHIA	YOUNG	CYNTHIA.YOUNG@sakilacustomer.org	32	1	2006-02-14 22:04:36
29	2	ANGELA	HERNANDEZ	ANGELA.HERNANDEZ@sakilacustomer.org	33	1	2006-02-14 22:04:36
30	1	MELISSA	KING	MELISSA.KING@sakilacustomer.org	34	1	2006-02-14 22:04:36
31	2	BRENDA	WRIGHT	BRENDA.WRIGHT@sakilacustomer.org	35	1	2006-02-14 22:04:36
33	2	ANNA	HILL	ANNA.HILL@sakilacustomer.org	37	1	2006-02-14 22:04:36
34	2	REBECCA	SCOTT	REBECCA.SCOTT@sakilacustomer.org	38	1	2006-02-14 22:04:36
35	2	VIRGINIA	GREEN	VIRGINIA.GREEN@sakilacustomer.org	39	1	2006-02-14 22:04:36
37	1	PAMELA	BAKER	PAMELA.BAKER@sakilacustomer.org	41	1	2006-02-14 22:04:36

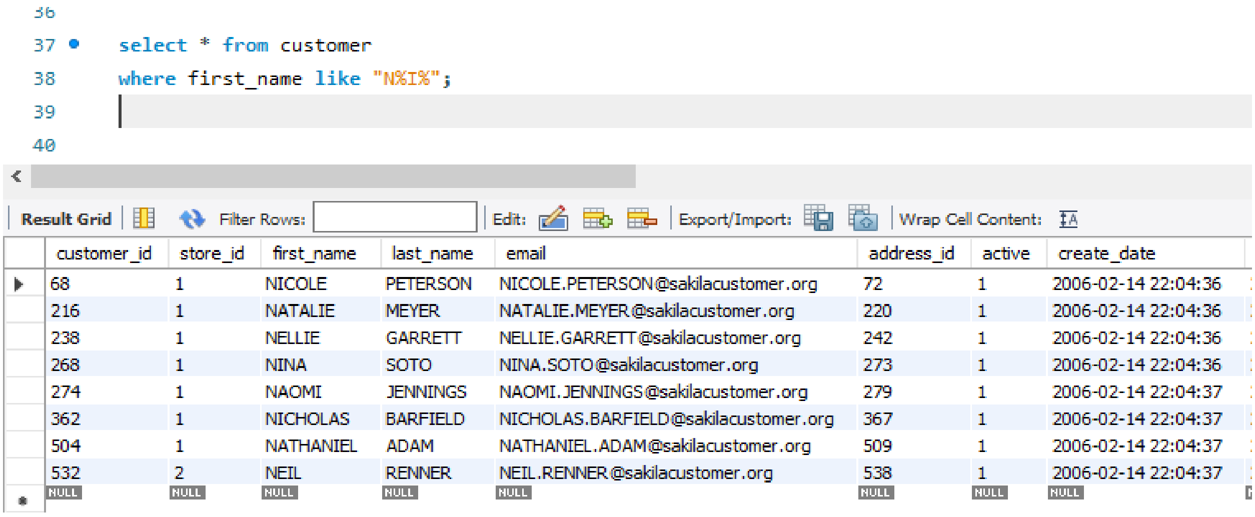
15- Display the list of first 4 cities which start and end with ‘a’.



```
select * from city
where city like 'A%' and city like '%A' limit 4;
```

city_id	city	country_id	last_update
2	Abha	82	2006-02-15 04:45:25
4	Acua	60	2006-02-15 04:45:25
5	Adana	97	2006-02-15 04:45:25
6	Addis Abeba	31	2006-02-15 04:45:25
NULL	NULL	NULL	NULL

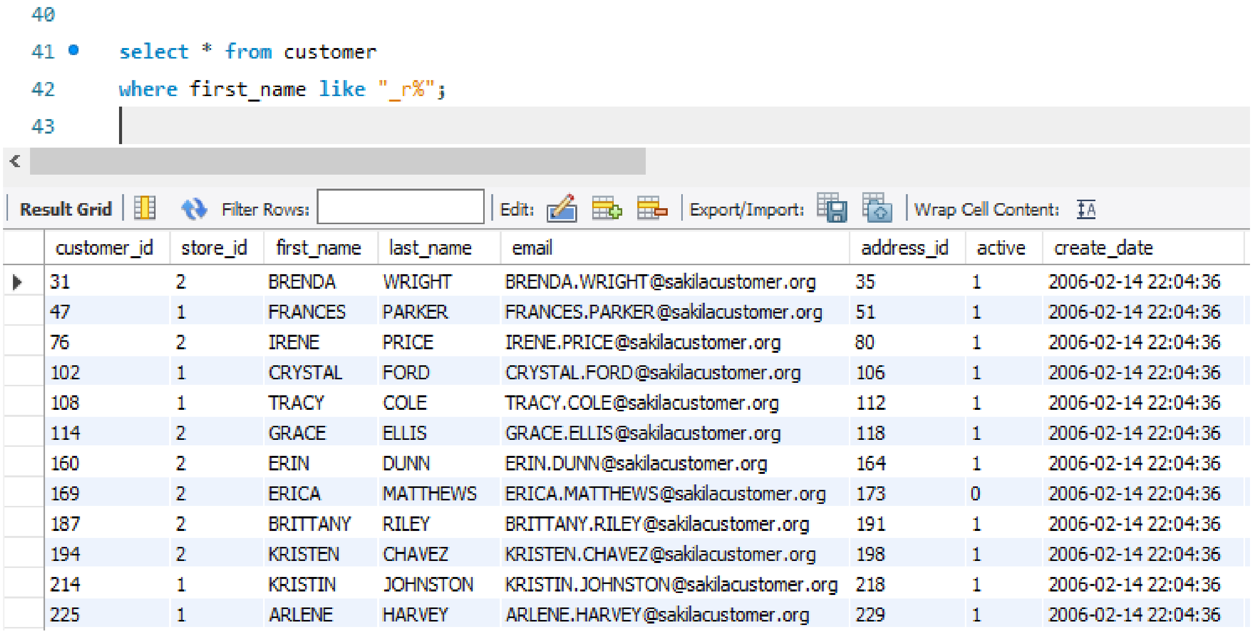
16- Find all customers whose first name have "NI" in any position.



```
select * from customer
where first_name like "N%I%";
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date
68	1	NICOLE	PETERSON	NICOLE.PETERSON@sakilacustomer.org	72	1	2006-02-14 22:04:36
216	1	NATALIE	MEYER	NATALIE.MEYER@sakilacustomer.org	220	1	2006-02-14 22:04:36
238	1	NELLIE	GARRETT	NELLIE.GARRETT@sakilacustomer.org	242	1	2006-02-14 22:04:36
268	1	NINA	SOTO	NINA.SOTO@sakilacustomer.org	273	1	2006-02-14 22:04:36
274	1	NAOMI	JENNINGS	NAOMI.JENNINGS@sakilacustomer.org	279	1	2006-02-14 22:04:37
362	1	NICHOLAS	BARFIELD	NICHOLAS.BARFIELD@sakilacustomer.org	367	1	2006-02-14 22:04:37
504	1	NATHANIEL	ADAM	NATHANIEL.ADAM@sakilacustomer.org	509	1	2006-02-14 22:04:37
532	2	NEIL	RENNER	NEIL.RENNER@sakilacustomer.org	538	1	2006-02-14 22:04:37
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

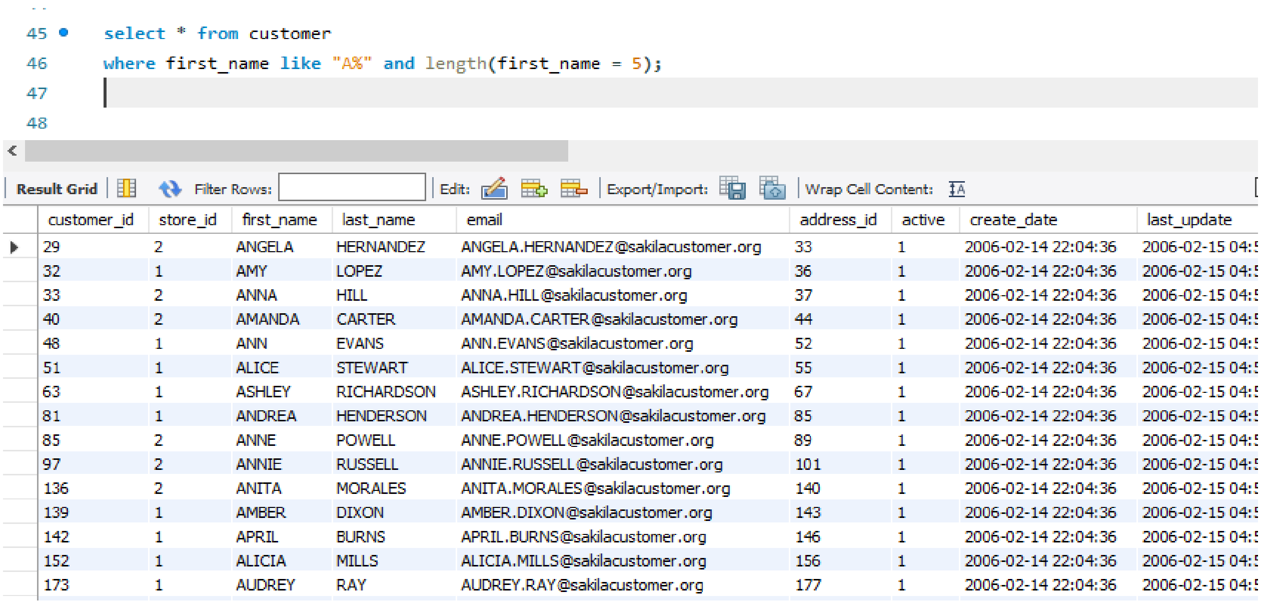
17- Find all customers whose first name have "r" in the second position.



```
select * from customer
where first_name like "_r%";
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date
31	2	BRENDA	WRIGHT	BRENDA.WRIGHT@sakilacustomer.org	35	1	2006-02-14 22:04:36
47	1	FRANCES	PARKER	FRANCES.PARKER@sakilacustomer.org	51	1	2006-02-14 22:04:36
76	2	IRENE	PRICE	IRENE.PRICE@sakilacustomer.org	80	1	2006-02-14 22:04:36
102	1	CRYSTAL	FORD	CRYSTAL.FORD@sakilacustomer.org	106	1	2006-02-14 22:04:36
108	1	TRACY	COLE	TRACY.COLE@sakilacustomer.org	112	1	2006-02-14 22:04:36
114	2	GRACE	ELLIS	GRACE.ELLIS@sakilacustomer.org	118	1	2006-02-14 22:04:36
160	2	ERIN	DUNN	ERIN.DUNN@sakilacustomer.org	164	1	2006-02-14 22:04:36
169	2	ERICA	MATTHEWS	ERICA.MATTHEWS@sakilacustomer.org	173	0	2006-02-14 22:04:36
187	2	BRITTANY	RILEY	BRITTANY.RILEY@sakilacustomer.org	191	1	2006-02-14 22:04:36
194	2	KRISTEN	CHAVEZ	KRISTEN.CHAVEZ@sakilacustomer.org	198	1	2006-02-14 22:04:36
214	1	KRISTIN	JOHNSTON	KRISTIN.JOHNSTON@sakilacustomer.org	218	1	2006-02-14 22:04:36
225	1	ARLENE	HARVEY	ARLENE.HARVEY@sakilacustomer.org	229	1	2006-02-14 22:04:36

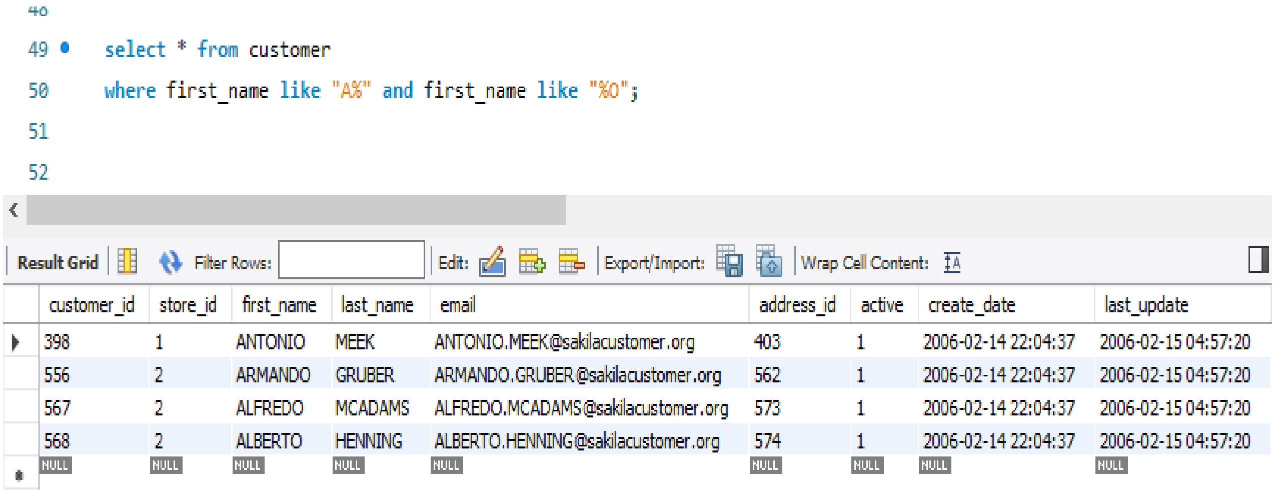
18 - Find all customers whose first name starts with "a" and are at least 5 characters in length.



```
select * from customer
where first_name like "A%" and length(first_name = 5);
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
29	2	ANGELA	HERNANDEZ	ANGELA.HERNANDEZ@sakilacustomer.org	33	1	2006-02-14 22:04:36	2006-02-15 04:5
32	1	AMY	LOPEZ	AMY.LOPEZ@sakilacustomer.org	36	1	2006-02-14 22:04:36	2006-02-15 04:5
33	2	ANNA	HILL	ANNA.HILL@sakilacustomer.org	37	1	2006-02-14 22:04:36	2006-02-15 04:5
40	2	AMANDA	CARTER	AMANDA.CARTER@sakilacustomer.org	44	1	2006-02-14 22:04:36	2006-02-15 04:5
48	1	ANN	EVANS	ANN.EVANS@sakilacustomer.org	52	1	2006-02-14 22:04:36	2006-02-15 04:5
51	1	ALICE	STEWART	ALICE.STEWART@sakilacustomer.org	55	1	2006-02-14 22:04:36	2006-02-15 04:5
63	1	ASHLEY	RICHARDSON	ASHLEY.RICHARDSON@sakilacustomer.org	67	1	2006-02-14 22:04:36	2006-02-15 04:5
81	1	ANDREA	HENDERSON	ANDREA.HENDERSON@sakilacustomer.org	85	1	2006-02-14 22:04:36	2006-02-15 04:5
85	2	ANNE	POWELL	ANNE.POWELL@sakilacustomer.org	89	1	2006-02-14 22:04:36	2006-02-15 04:5
97	2	ANNIE	RUSSELL	ANNIE.RUSSELL@sakilacustomer.org	101	1	2006-02-14 22:04:36	2006-02-15 04:5
136	2	ANITA	MORALES	ANITA.MORALES@sakilacustomer.org	140	1	2006-02-14 22:04:36	2006-02-15 04:5
139	1	AMBER	DIXON	AMBER.DIXON@sakilacustomer.org	143	1	2006-02-14 22:04:36	2006-02-15 04:5
142	1	APRIL	BURNS	APRIL.BURNS@sakilacustomer.org	146	1	2006-02-14 22:04:36	2006-02-15 04:5
152	1	ALICIA	MILLS	ALICIA.MILLS@sakilacustomer.org	156	1	2006-02-14 22:04:36	2006-02-15 04:5
173	1	AUDREY	RAY	AUDREY.RAY@sakilacustomer.org	177	1	2006-02-14 22:04:36	2006-02-15 04:5

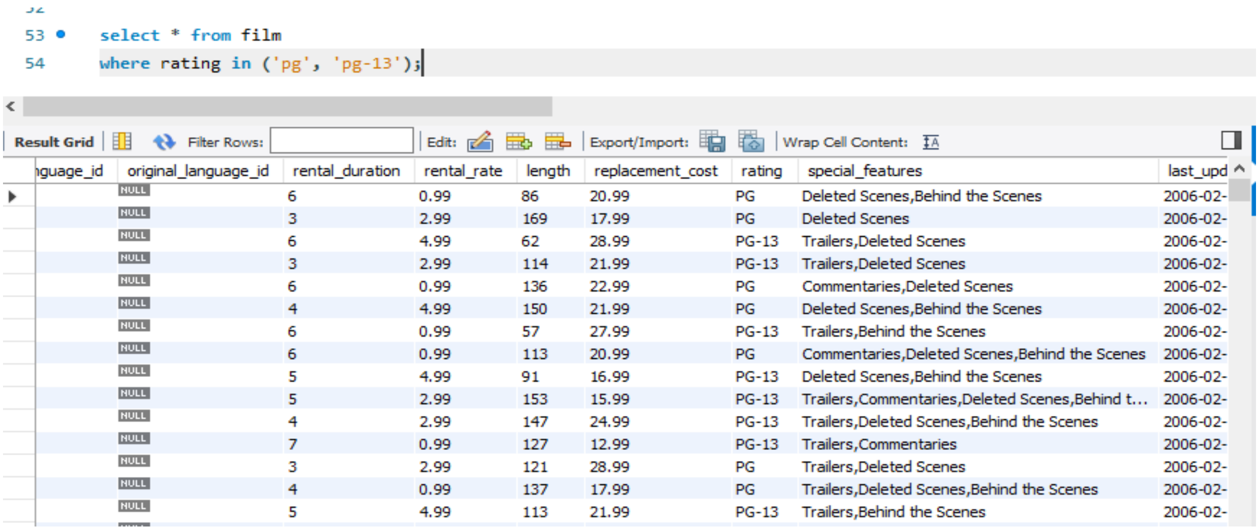
19- Find all customers whose first name starts with "a" and ends with "o".



```
select * from customer
where first_name like "A%" and first_name like "%O";
```

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
398	1	ANTONIO	MEEK	ANTONIO.MEEK@sakilacustomer.org	403	1	2006-02-14 22:04:37	2006-02-15 04:57:20
556	2	ARMANDO	GRUBER	ARMANDO.GRUBER@sakilacustomer.org	562	1	2006-02-14 22:04:37	2006-02-15 04:57:20
567	2	ALFREDO	MCADAMS	ALFREDO.MCADAMS@sakilacustomer.org	573	1	2006-02-14 22:04:37	2006-02-15 04:57:20
568	2	ALBERTO	HENNING	ALBERTO.HENNING@sakilacustomer.org	574	1	2006-02-14 22:04:37	2006-02-15 04:57:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

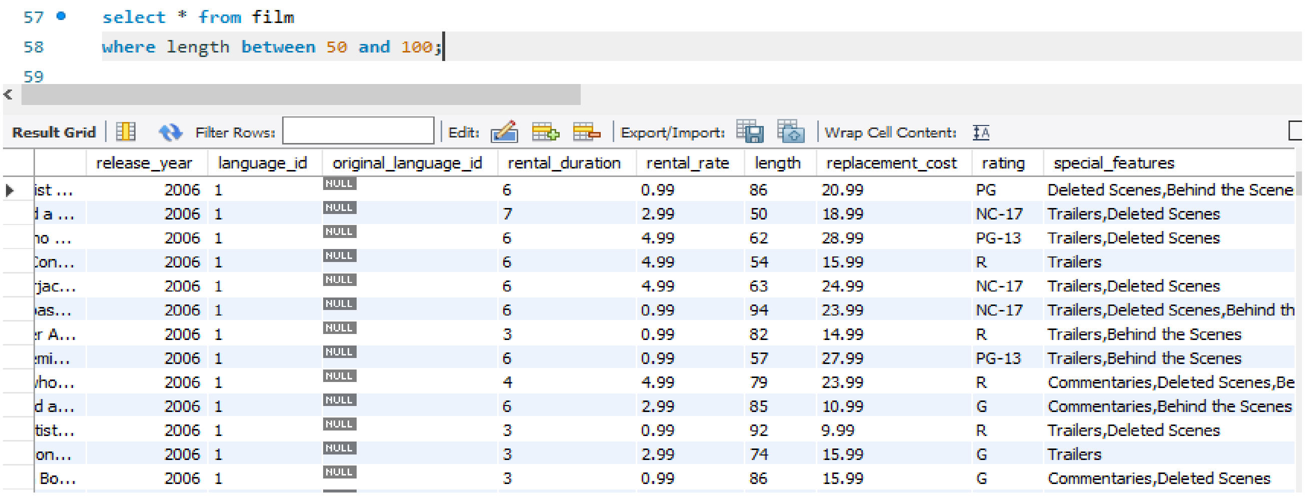
20 - Get the films with pg and pg-13 rating using IN operator.



```
select * from film
where rating in ('pg', 'pg-13');
```

guage_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_upd
	NULL	6	0.99	86	20.99	PG	Deleted Scenes,Behind the Scenes	2006-02-
	NULL	3	2.99	169	17.99	PG	Deleted Scenes	2006-02-
	NULL	6	4.99	62	28.99	PG-13	Trailers,Deleted Scenes	2006-02-
	NULL	3	2.99	114	21.99	PG-13	Trailers,Deleted Scenes	2006-02-
	NULL	6	0.99	136	22.99	PG	Commentaries,Deleted Scenes	2006-02-
	NULL	4	4.99	150	21.99	PG	Deleted Scenes,Behind the Scenes	2006-02-
	NULL	6	0.99	57	27.99	PG-13	Trailers,Behind the Scenes	2006-02-
	NULL	6	0.99	113	20.99	PG	Commentaries,Deleted Scenes,Behind the Scenes	2006-02-
	NULL	5	4.99	91	16.99	PG-13	Deleted Scenes,Behind the Scenes	2006-02-
	NULL	5	2.99	153	15.99	PG-13	Trailers,Commentaries,Deleted Scenes,Behind t...	2006-02-
	NULL	4	2.99	147	24.99	PG-13	Trailers,Deleted Scenes,Behind the Scenes	2006-02-
	NULL	7	0.99	127	12.99	PG-13	Trailers,Commentaries	2006-02-
	NULL	3	2.99	121	28.99	PG	Trailers,Deleted Scenes	2006-02-
	NULL	4	0.99	137	17.99	PG	Trailers,Deleted Scenes,Behind the Scenes	2006-02-
	NULL	5	4.99	113	21.99	PG-13	Trailers,Behind the Scenes	2006-02-

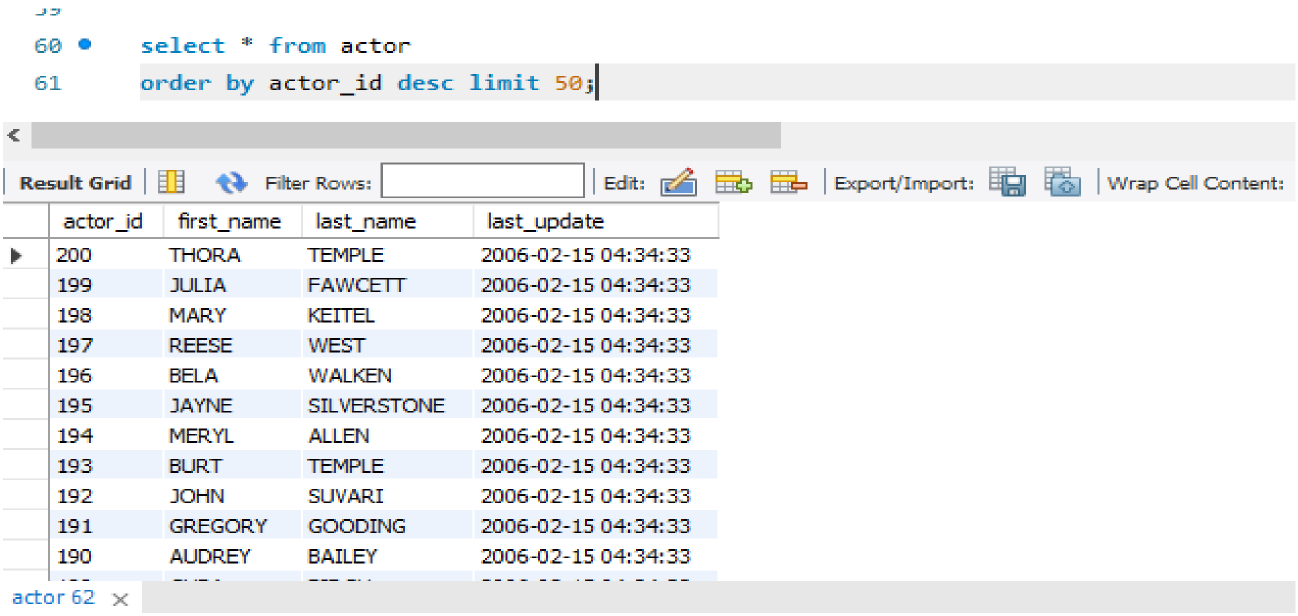
21 - Get the films with length between 50 to 100 using between operator.



```
select * from film
where length between 50 and 100;
```

	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features
ist ...	2006	1	NULL	6	0.99	86	20.99	PG	Deleted Scenes,Behind the Scene
d a ...	2006	1	NULL	7	2.99	50	18.99	NC-17	Trailers,Deleted Scenes
no ...	2006	1	NULL	6	4.99	62	28.99	PG-13	Trailers,Deleted Scenes
Con...	2006	1	NULL	6	4.99	54	15.99	R	Trailers
rjac...	2006	1	NULL	6	4.99	63	24.99	NC-17	Trailers,Deleted Scenes
as...	2006	1	NULL	6	0.99	94	23.99	NC-17	Trailers,Deleted Scenes,Behind th
r A...	2006	1	NULL	3	0.99	82	14.99	R	Trailers,Behind the Scenes
mi...	2006	1	NULL	6	0.99	57	27.99	PG-13	Trailers,Behind the Scenes
vho...	2006	1	NULL	4	4.99	79	23.99	R	Commentaries,Deleted Scenes,Be
d a...	2006	1	NULL	6	2.99	85	10.99	G	Commentaries,Behind the Scenes
tist...	2006	1	NULL	3	0.99	92	9.99	R	Trailers,Deleted Scenes
on...	2006	1	NULL	3	2.99	74	15.99	G	Trailers
Bo...	2006	1	NULL	3	0.99	86	15.99	G	Commentaries,Deleted Scenes

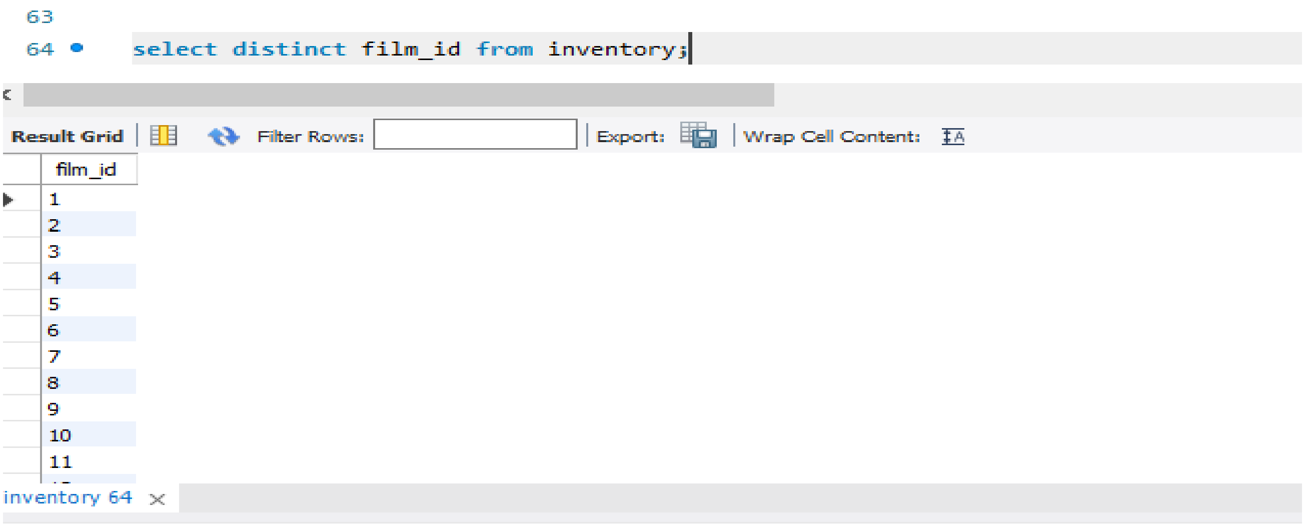
22 - Get the top 50 actors using limit operator.



```
select * from actor
order by actor_id desc limit 50;
```

actor_id	first_name	last_name	last_update
200	THORA	TEMPLE	2006-02-15 04:34:33
199	JULIA	FAWCETT	2006-02-15 04:34:33
198	MARY	KEITEL	2006-02-15 04:34:33
197	REESE	WEST	2006-02-15 04:34:33
196	BELA	WALKEN	2006-02-15 04:34:33
195	JAYNE	SILVERSTONE	2006-02-15 04:34:33
194	MERYL	ALLEN	2006-02-15 04:34:33
193	BURT	TEMPLE	2006-02-15 04:34:33
192	JOHN	SUVARI	2006-02-15 04:34:33
191	GREGORY	GOODING	2006-02-15 04:34:33
190	AUDREY	BAILEY	2006-02-15 04:34:33

23 - Get the distinct film ids from inventory table.

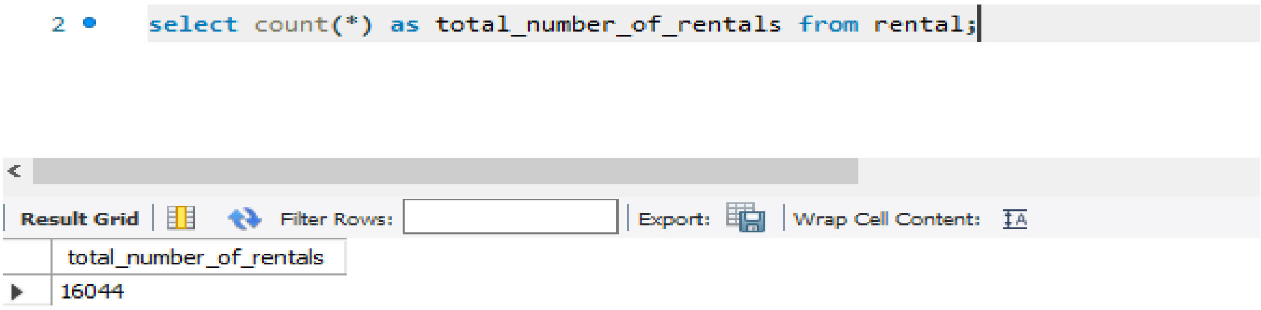


```
select distinct film_id from inventory;
```

film_id
1
2
3
4
5
6
7
8
9
10
11

Basic Aggregate Functions:

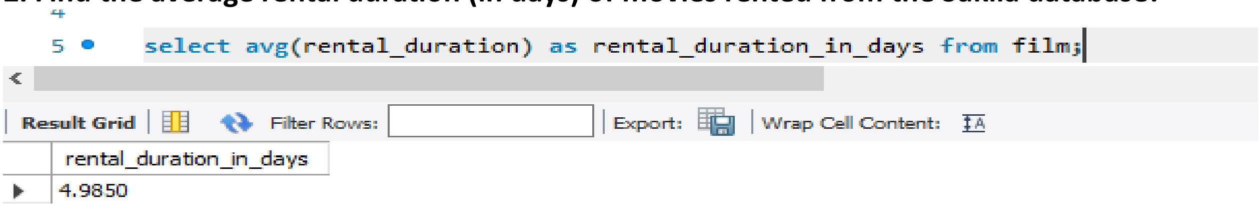
1: Retrieve the total number of rentals made in Sakila database.



```sql
select count(*) as total_number_of_rentals from rental;
```

total_number_of_rentals
16044

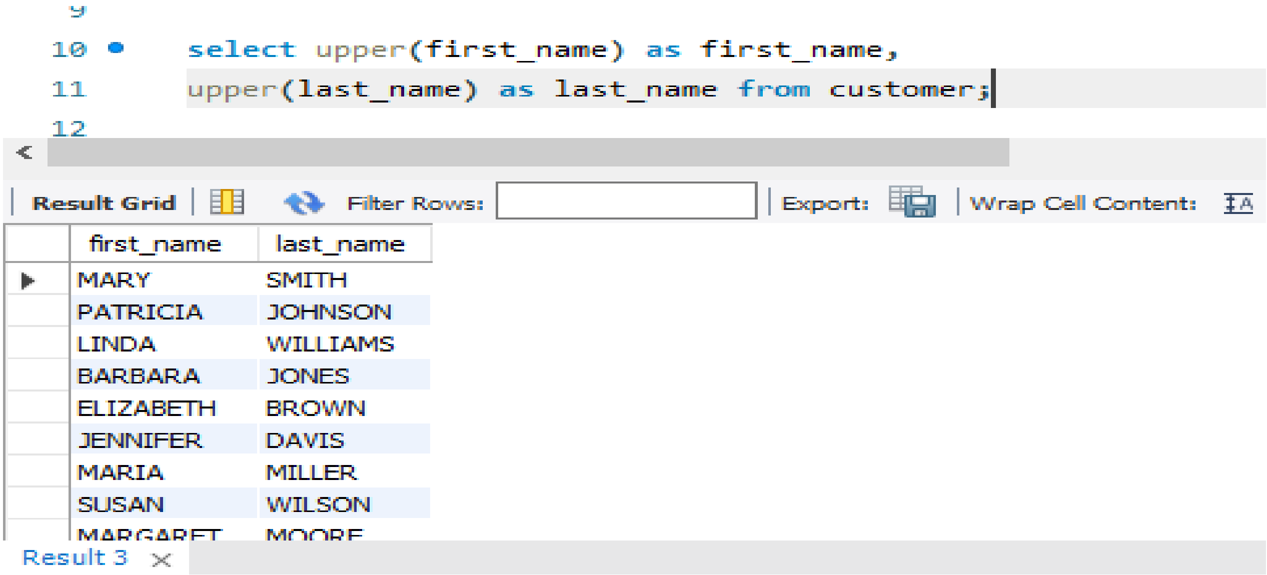
2. Find the average rental duration (in days) of movies rented from the Sakila database.



```sql
select avg(rental_duration) as rental_duration_in_days from film;
```

rental_duration_in_days
4.9850

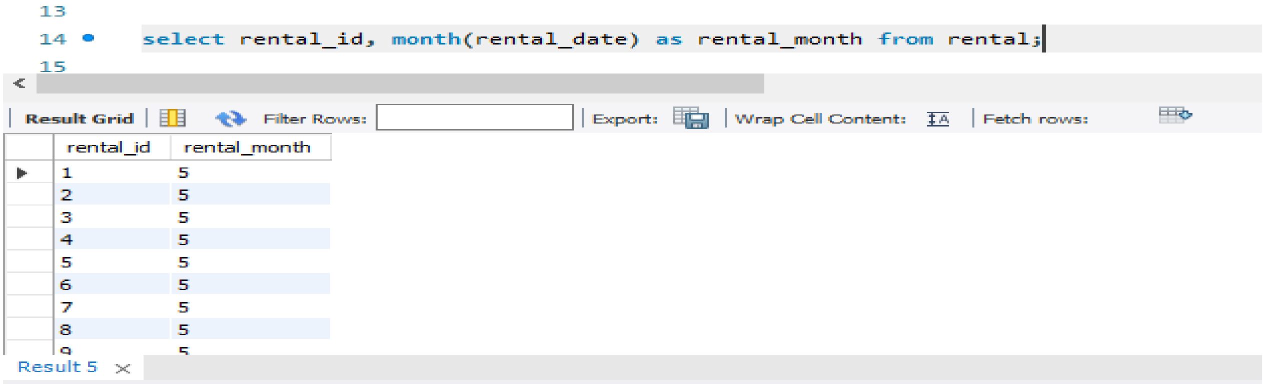
3. Display the first name and last name of customers in uppercase.



```sql
select upper(first_name) as first_name,
upper(last_name) as last_name from customer;
```

first_name	last_name
MARY	SMITH
PATRICIA	JOHNSON
LINDA	WILLIAMS
BARBARA	JONES
ELIZABETH	BROWN
JENNIFER	DAVIS
MARIA	MILLER
SUSAN	WILSON
MARGARET	MOORE

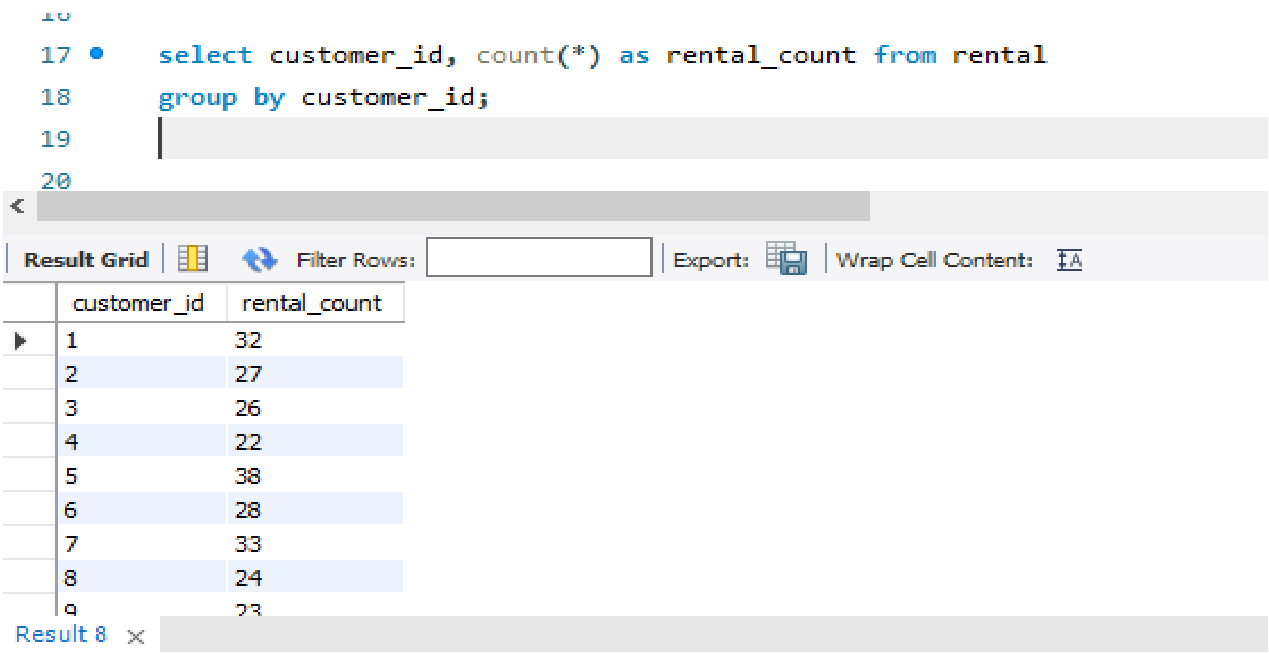
4. Extract the month from the rental date and display it alongside the rental ID.



```sql
select rental_id, month(rental_date) as rental_month from rental;
```

rental_id	rental_month
1	5
2	5
3	5
4	5
5	5
6	5
7	5
8	5
9	5

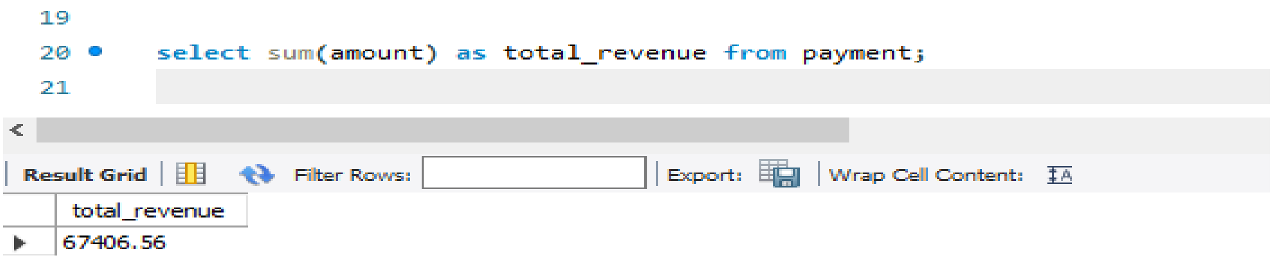
5. Retrieve the count of rentals for each customer (display customer ID and the count of rentals).



```
select customer_id, count(*) as rental_count from rental
group by customer_id;
```

customer_id	rental_count
1	32
2	27
3	26
4	22
5	38
6	28
7	33
8	24
9	23

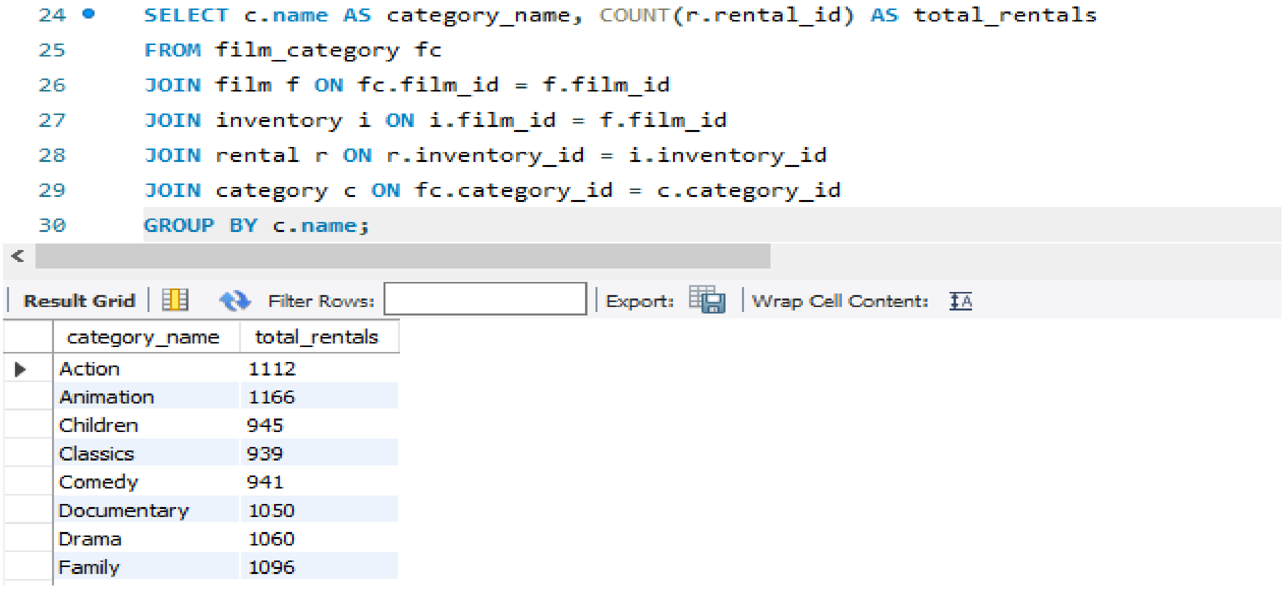
6. Find the total revenue generated by each store.



```
select sum(amount) as total_revenue from payment;
```

total_revenue
67406.56

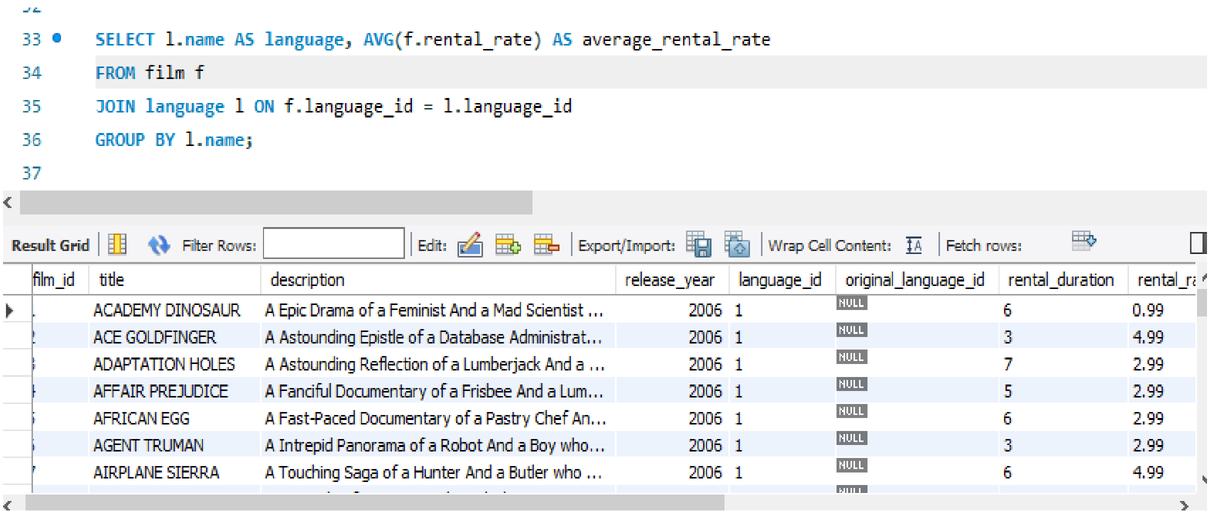
7. Determine the total number of rentals for each category of movies.



```
SELECT c.name AS category_name, COUNT(r.rental_id) AS total_rentals
FROM film_category fc
JOIN film f ON fc.film_id = f.film_id
JOIN inventory i ON i.film_id = f.film_id
JOIN rental r ON r.inventory_id = i.inventory_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name;
```

category_name	total_rentals
Action	1112
Animation	1166
Children	945
Classics	939
Comedy	941
Documentary	1050
Drama	1060
Family	1096

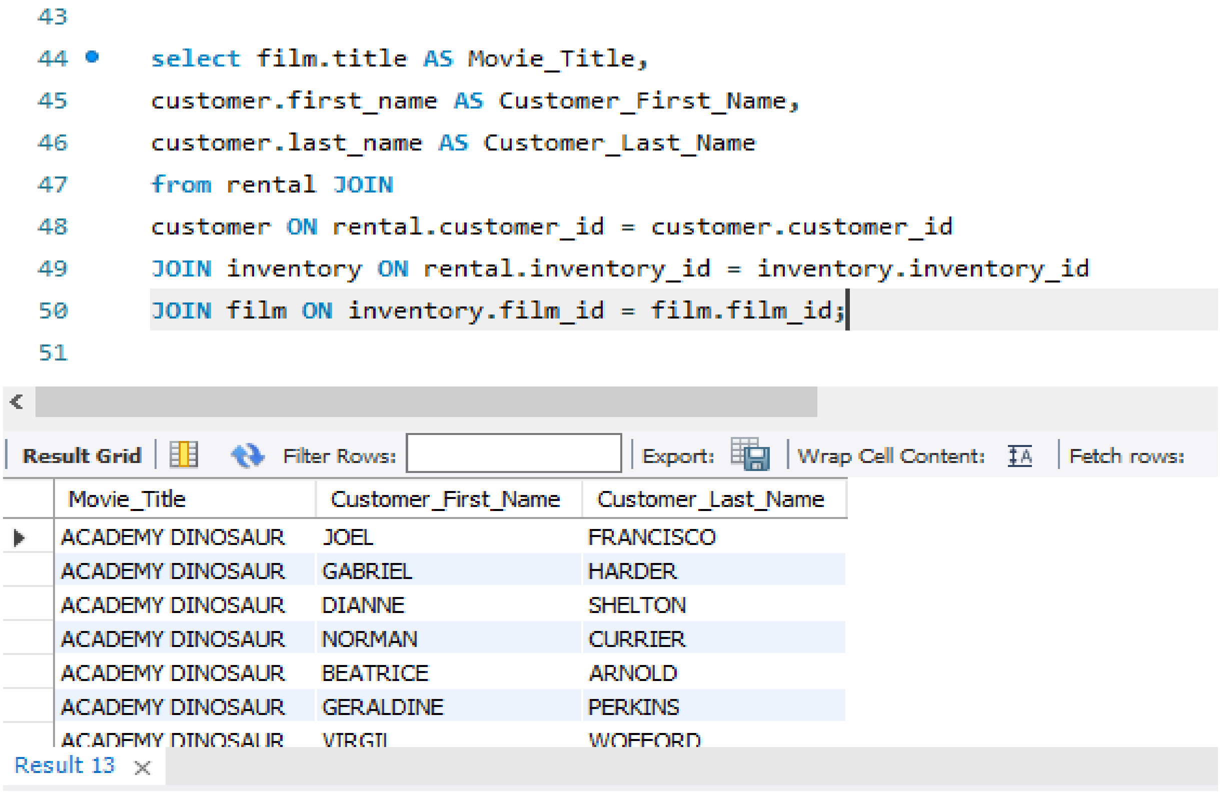
8. Find the average rental rate of movies in each language.



```
SELECT l.name AS language, AVG(f.rental_rate) AS average_rental_rate
FROM film f
JOIN language l ON f.language_id = l.language_id
GROUP BY l.name;
```

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_ra
	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist ...	2006	1	NULL	6	0.99
	ACE GOLDFINGER	A Astounding Epistle of a Database Administrat...	2006	1	NULL	3	4.99
	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a ...	2006	1	NULL	7	2.99
	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lum...	2006	1	NULL	5	2.99
	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...	2006	1	NULL	6	2.99
	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who...	2006	1	NULL	3	2.99
	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who ...	2006	1	NULL	6	4.99

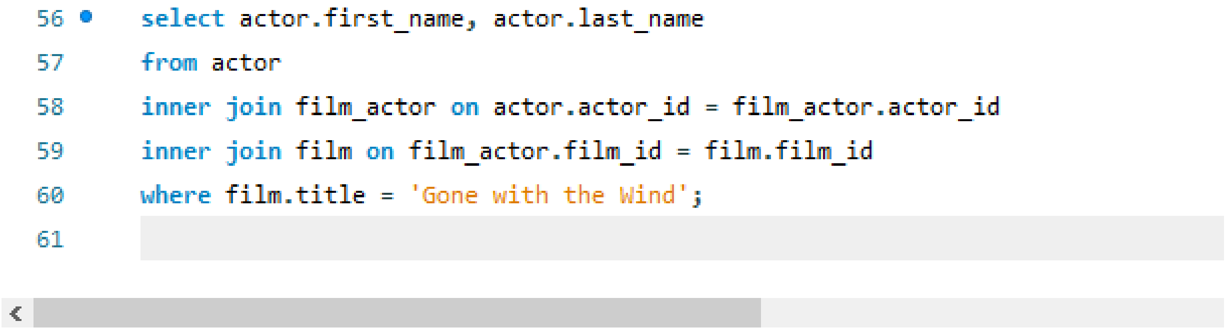
9. Display the title of the movie, customer s first name, and last name who rented it.



```
select film.title AS Movie_Title,
customer.first_name AS Customer_First_Name,
customer.last_name AS Customer_Last_Name
from rental JOIN
customer ON rental.customer_id = customer.customer_id
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id;
```

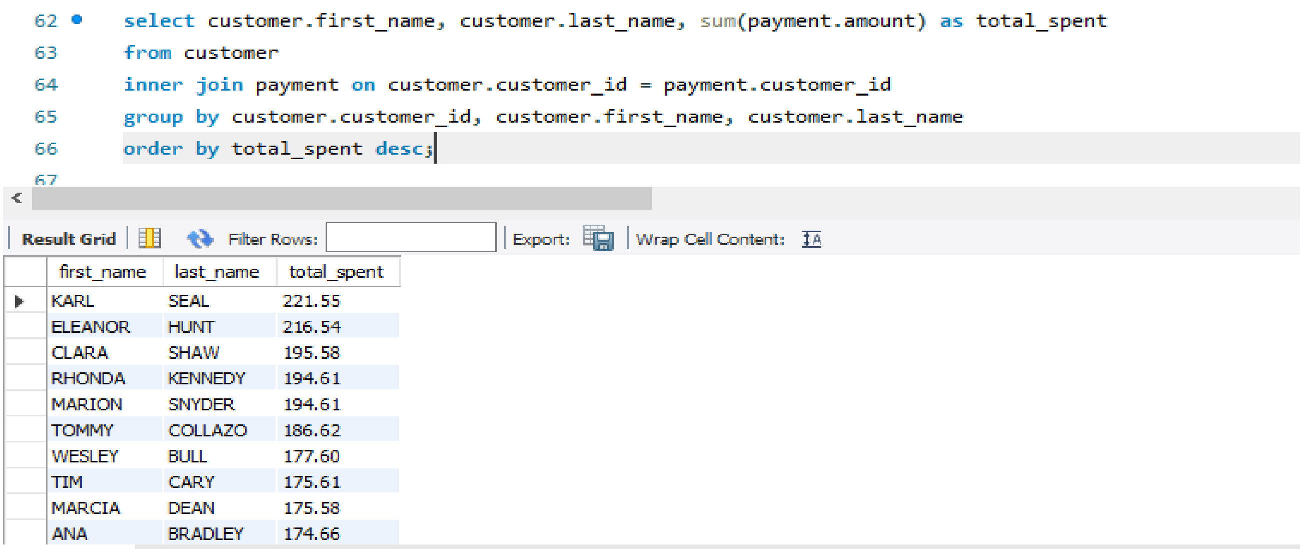
Movie_Title	Customer_First_Name	Customer_Last_Name
ACADEMY DINOSAUR	JOEL	FRANCISCO
ACADEMY DINOSAUR	GABRIEL	HARDER
ACADEMY DINOSAUR	DIANNE	SHELTON
ACADEMY DINOSAUR	NORMAN	CURRIER
ACADEMY DINOSAUR	BEATRICE	ARNOLD
ACADEMY DINOSAUR	GERALDINE	PERKINS
ACADEMY DINOSAUR	VIRGIL	WOFFORD

10. Retrieve the names of all actors who have appeared in the film "Gone with the Wind."



```
select actor.first_name, actor.last_name
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
where film.title = 'Gone with the Wind';
```

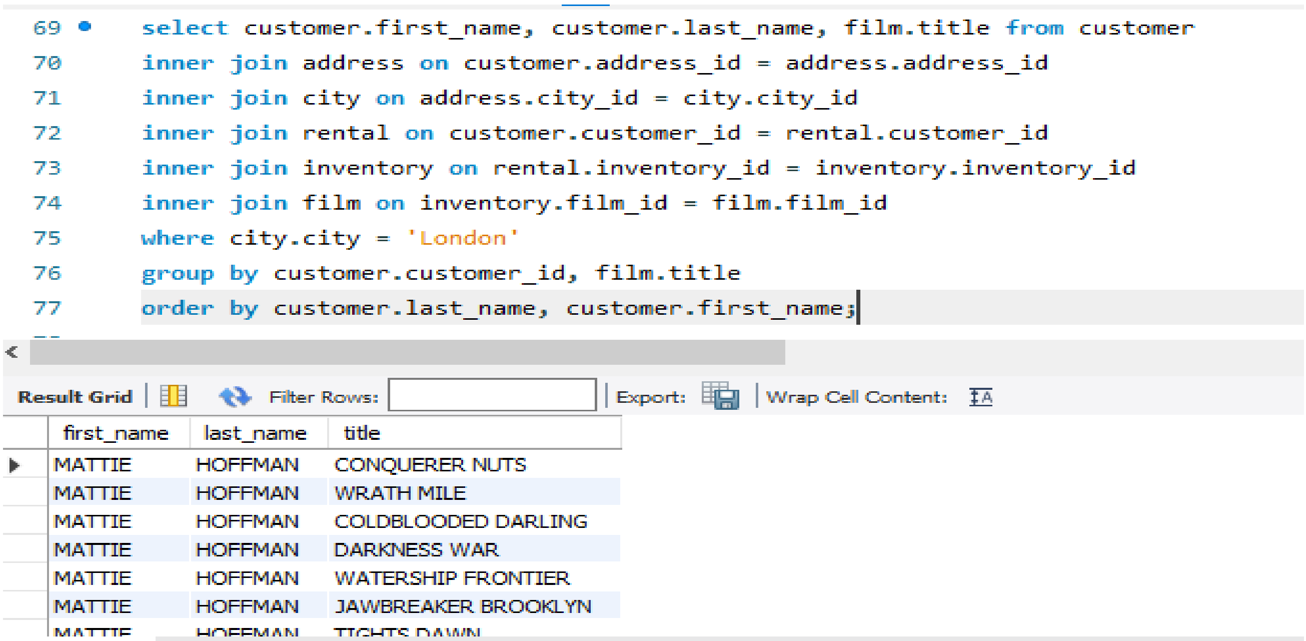
11. Retrieve the customer names along with the total amount they've spent on rentals.



```sql
select customer.first_name, customer.last_name, sum(payment.amount) as total_spent
from customer
inner join payment on customer.customer_id = payment.customer_id
group by customer.customer_id, customer.first_name, customer.last_name
order by total_spent desc;
```

first_name	last_name	total_spent
KARL	SEAL	221.55
ELEANOR	HUNT	216.54
CLARA	SHAW	195.58
RHONDA	KENNEDY	194.61
MARION	SNYDER	194.61
TOMMY	COLLAZO	186.62
WESLEY	BULL	177.60
TIM	CARY	175.61
MARCIA	DEAN	175.58
ANA	BRADLEY	174.66

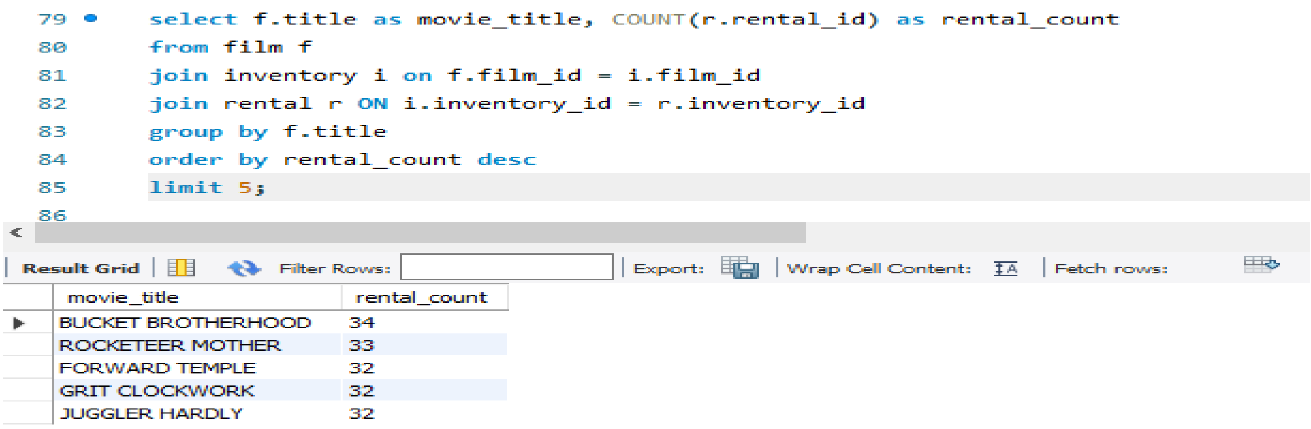
12. List the titles of movies rented by each customer in a particular city (e.g., 'London').



```sql
select customer.first_name, customer.last_name, film.title from customer
inner join address on customer.address_id = address.address_id
inner join city on address.city_id = city.city_id
inner join rental on customer.customer_id = rental.customer_id
inner join inventory on rental.inventory_id = inventory.inventory_id
inner join film on inventory.film_id = film.film_id
where city.city = 'London'
group by customer.customer_id, film.title
order by customer.last_name, customer.first_name;
```

first_name	last_name	title
MATTIE	HOFFMAN	CONQUERER NUTS
MATTIE	HOFFMAN	WRATH MILE
MATTIE	HOFFMAN	COLDBLOODED DARLING
MATTIE	HOFFMAN	DARKNESS WAR
MATTIE	HOFFMAN	WATERSHIP FRONTIER
MATTIE	HOFFMAN	JAWBREAKER BROOKLYN
MATTIE	HOFFMAN	TIGHTS DAWN

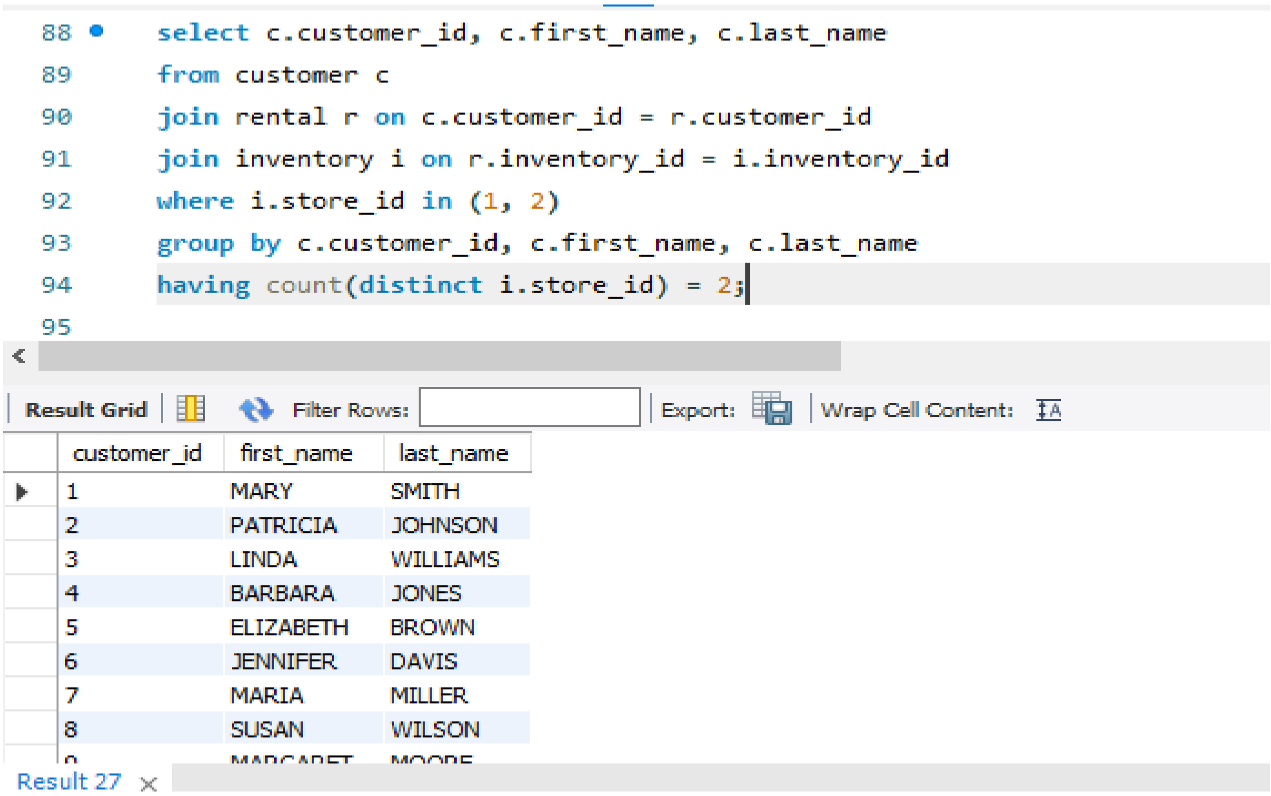
13. Display the top 5 rented movies along with the number of times they've been rented.



```sql
select f.title as movie_title, COUNT(r.rental_id) as rental_count
from film f
join inventory i on f.film_id = i.film_id
join rental r ON i.inventory_id = r.inventory_id
group by f.title
order by rental_count desc
limit 5;
```

movie_title	rental_count
BUCKET BROTHERHOOD	34
ROCKETEER MOTHER	33
FORWARD TEMPLE	32
GRIT CLOCKWORK	32
JUGGLER HARDLY	32

14. Determine the customers who have rented movies from both stores (store ID 1 and store ID 2).

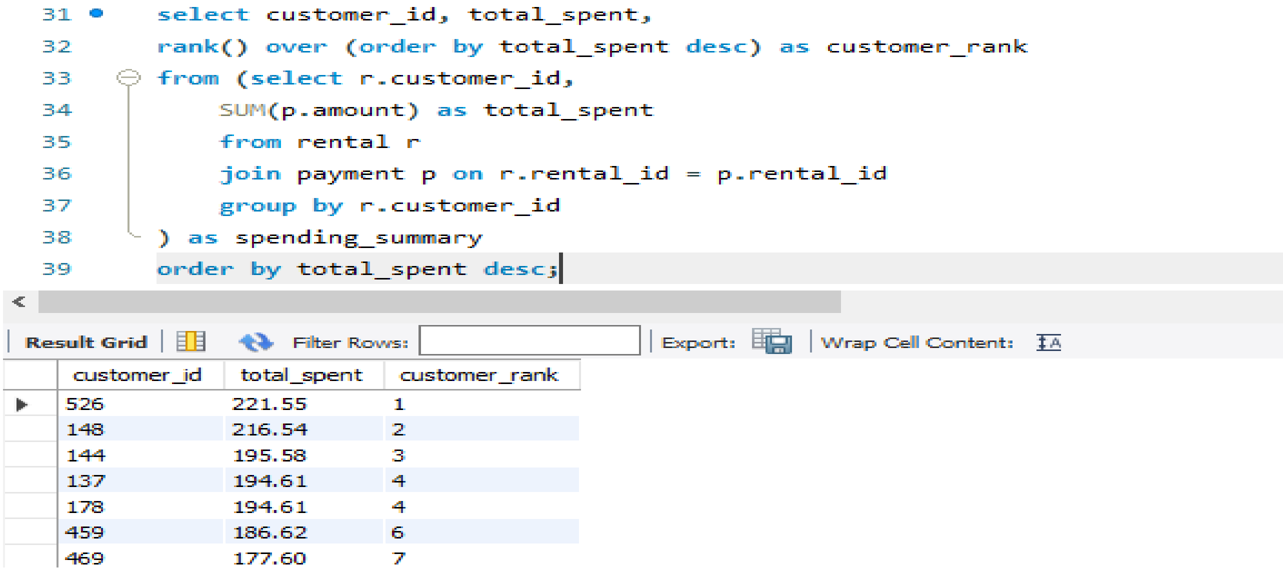


```sql
select c.customer_id, c.first_name, c.last_name
from customer c
join rental r on c.customer_id = r.customer_id
join inventory i on r.inventory_id = i.inventory_id
where i.store_id in (1, 2)
group by c.customer_id, c.first_name, c.last_name
having count(distinct i.store_id) = 2;
```

customer_id	first_name	last_name
1	MARY	SMITH
2	PATRICIA	JOHNSON
3	LINDA	WILLIAMS
4	BARBARA	JONES
5	ELIZABETH	BROWN
6	JENNIFER	DAVIS
7	MARIA	MILLER
8	SUSAN	WILSON
9	MARGARET	MOORE

Windows Function

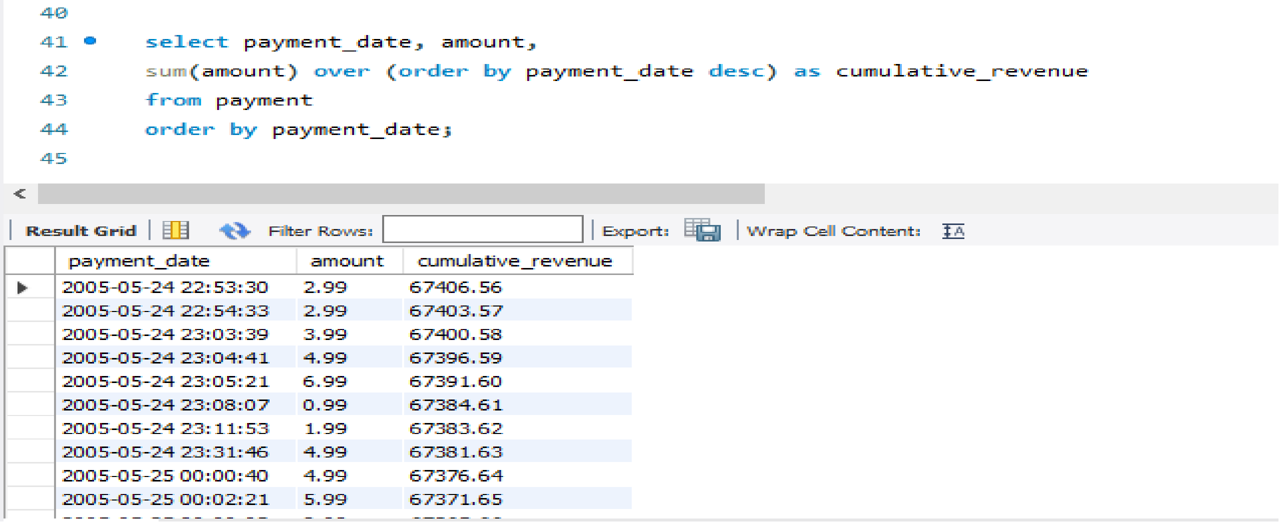
1. Rank the customers based on the total amount they've spent on rentals.



```sql
select customer_id, total_spent,
rank() over (order by total_spent desc) as customer_rank
from (select r.customer_id,
     SUM(p.amount) as total_spent
     from rental r
     join payment p on r.rental_id = p.rental_id
     group by r.customer_id
) as spending_summary
order by total_spent desc;
```

customer_id	total_spent	customer_rank
526	221.55	1
148	216.54	2
144	195.58	3
137	194.61	4
178	194.61	4
459	186.62	6
469	177.60	7

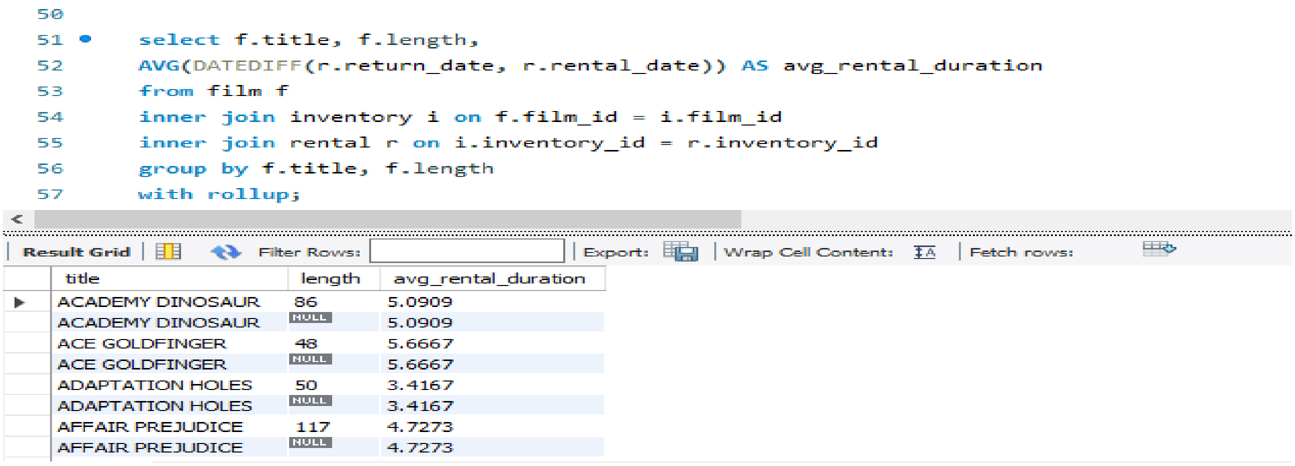
2. Calculate the cumulative revenue generated by each film over time.



```sql
select payment_date, amount,
sum(amount) over (order by payment_date desc) as cumulative_revenue
from payment
order by payment_date;
```

payment_date	amount	cumulative_revenue
2005-05-24 22:53:30	2.99	67406.56
2005-05-24 22:54:33	2.99	67403.57
2005-05-24 23:03:39	3.99	67400.58
2005-05-24 23:04:41	4.99	67396.59
2005-05-24 23:05:21	6.99	67391.60
2005-05-24 23:08:07	0.99	67384.61
2005-05-24 23:11:53	1.99	67383.62
2005-05-24 23:31:46	4.99	67381.63
2005-05-25 00:00:40	4.99	67376.64
2005-05-25 00:02:21	5.99	67371.65

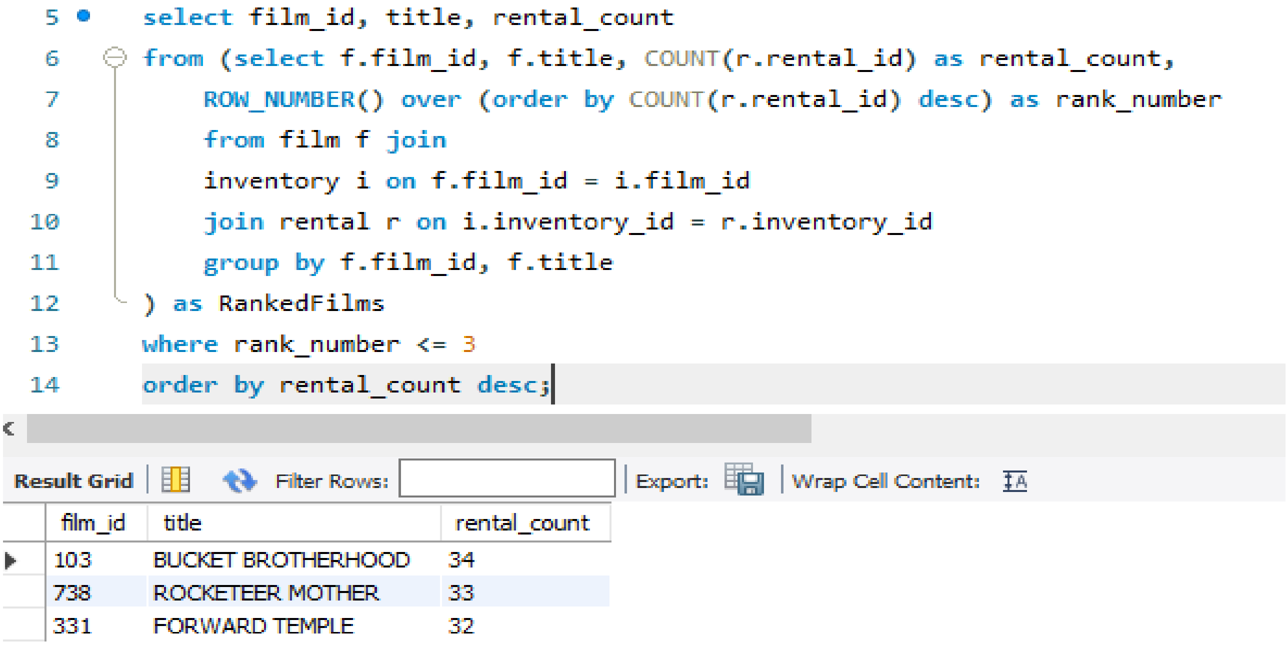
3. Determine the average rental duration for each film, considering films with similar lengths.



```sql
select f.title, f.length,
AVG(DATEDIFF(r.return_date, r.rental_date)) AS avg_rental_duration
from film f
inner join inventory i on f.film_id = i.film_id
inner join rental r on i.inventory_id = r.inventory_id
group by f.title, f.length
with rollup;
```

title	length	avg_rental_duration
ACADEMY DINOSAUR	86	5.0909
ACADEMY DINOSAUR	NULL	5.0909
ACE GOLDFINGER	48	5.6667
ACE GOLDFINGER	NULL	5.6667
ADAPTATION HOLES	50	3.4167
ADAPTATION HOLES	NULL	3.4167
AFFAIR PREJUDICE	117	4.7273
AFFAIR PREJUDICE	NULL	4.7273

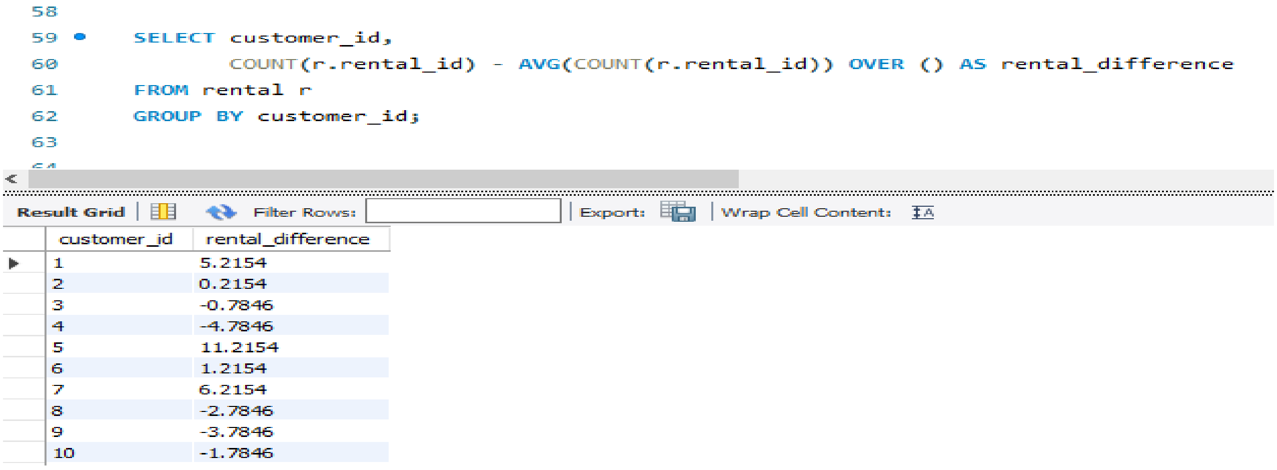
4. Identify the top 3 films in each category based on their rental counts.



```
select film_id, title, rental_count
from (select f.film_id, f.title, COUNT(r.rental_id) as rental_count,
    ROW_NUMBER() over (order by COUNT(r.rental_id) desc) as rank_number
    from film f join
    inventory i on f.film_id = i.film_id
    join rental r on i.inventory_id = r.inventory_id
    group by f.film_id, f.title
) as RankedFilms
where rank_number <= 3
order by rental_count desc;
```

film_id	title	rental_count
103	BUCKET BROTHERHOOD	34
738	ROCKETEER MOTHER	33
331	FORWARD TEMPLE	32

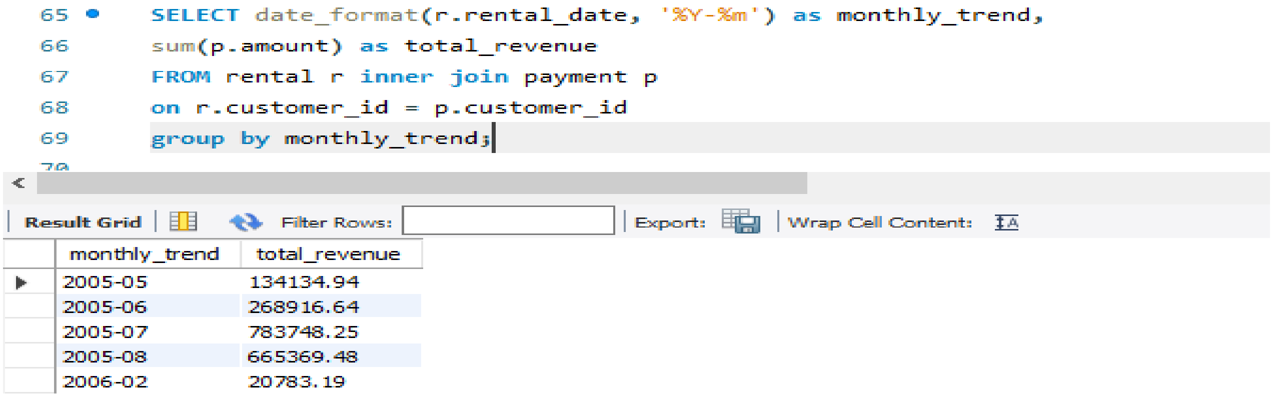
5. Calculate the difference in rental counts between each customer's total rentals and the average rentals across all customers.



```
SELECT customer_id,
       COUNT(r.rental_id) - AVG(COUNT(r.rental_id)) OVER () AS rental_difference
FROM rental r
GROUP BY customer_id;
```

customer_id	rental_difference
1	5.2154
2	0.2154
3	-0.7846
4	-4.7846
5	11.2154
6	1.2154
7	6.2154
8	-2.7846
9	-3.7846
10	-1.7846

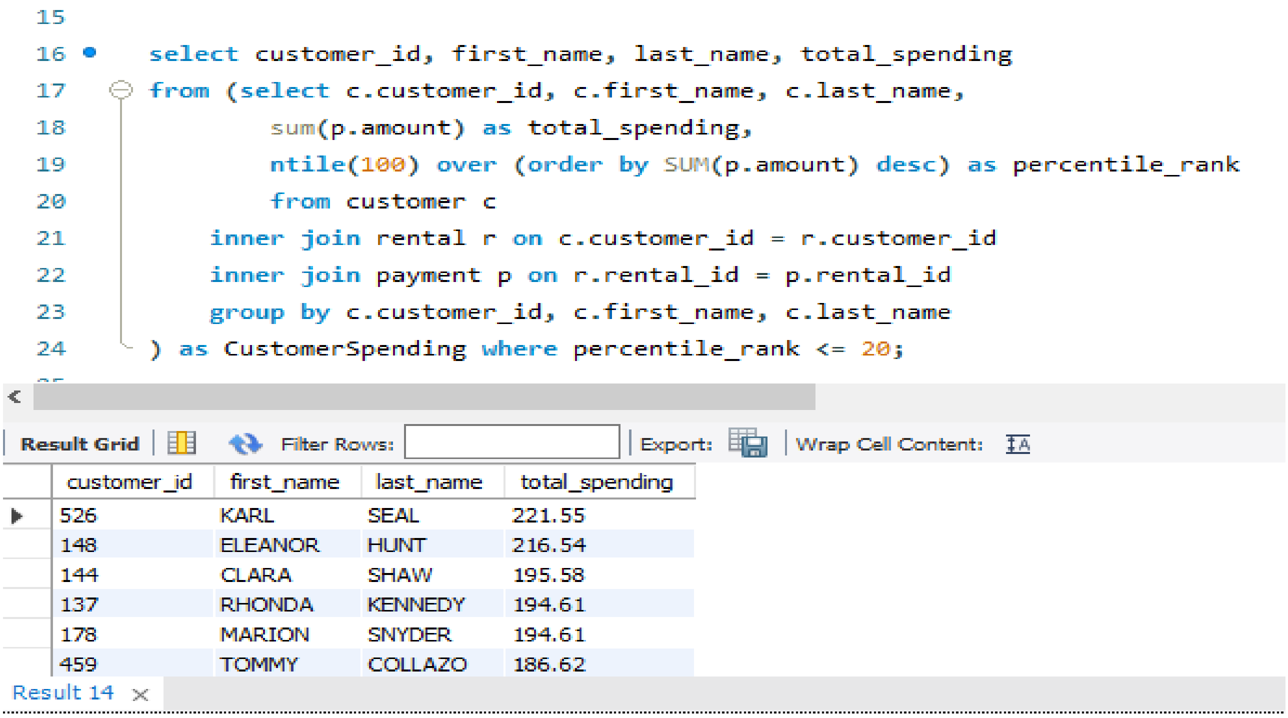
6. Find the monthly revenue trend for the entire rental store over time.



```
SELECT date_format(r.rental_date, '%Y-%m') as monthly_trend,
sum(p.amount) as total_revenue
FROM rental r inner join payment p
on r.customer_id = p.customer_id
group by monthly_trend;
```

monthly_trend	total_revenue
2005-05	134134.94
2005-06	268916.64
2005-07	783748.25
2005-08	665369.48
2006-02	20783.19

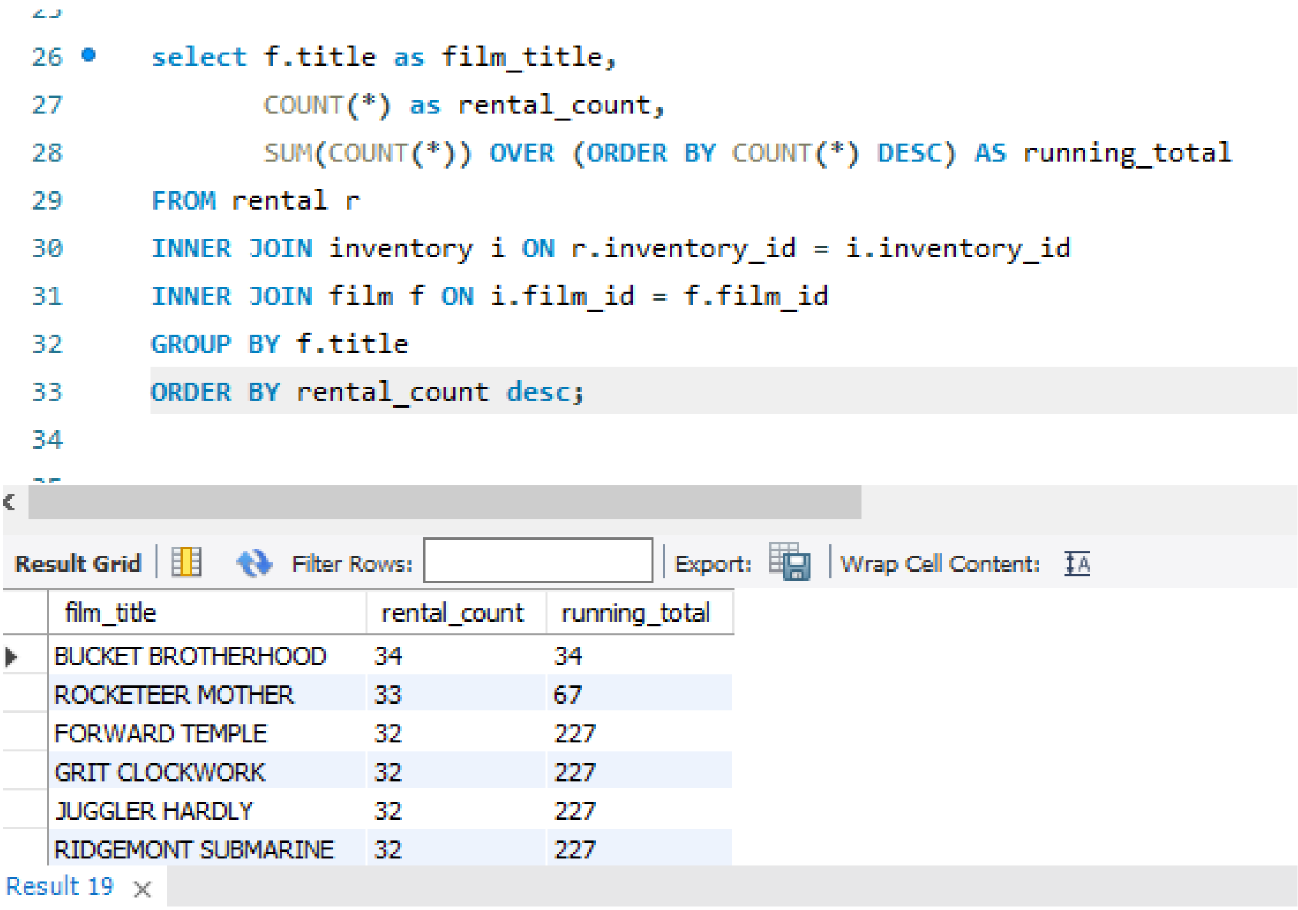
7. Identify the customers whose total spending on rentals falls within the top 20% of all customers.



```sql
select customer_id, first_name, last_name, total_spending
from (select c.customer_id, c.first_name, c.last_name,
        sum(p.amount) as total_spending,
        ntile(100) over (order by SUM(p.amount) desc) as percentile_rank
        from customer c
    inner join rental r on c.customer_id = r.customer_id
    inner join payment p on r.rental_id = p.rental_id
    group by c.customer_id, c.first_name, c.last_name
) as CustomerSpending where percentile_rank <= 20;
```

customer_id	first_name	last_name	total_spending
526	KARL	SEAL	221.55
148	ELEANOR	HUNT	216.54
144	CLARA	SHAW	195.58
137	RHONDA	KENNEDY	194.61
178	MARION	SNYDER	194.61
459	TOMMY	COLLAZO	186.62

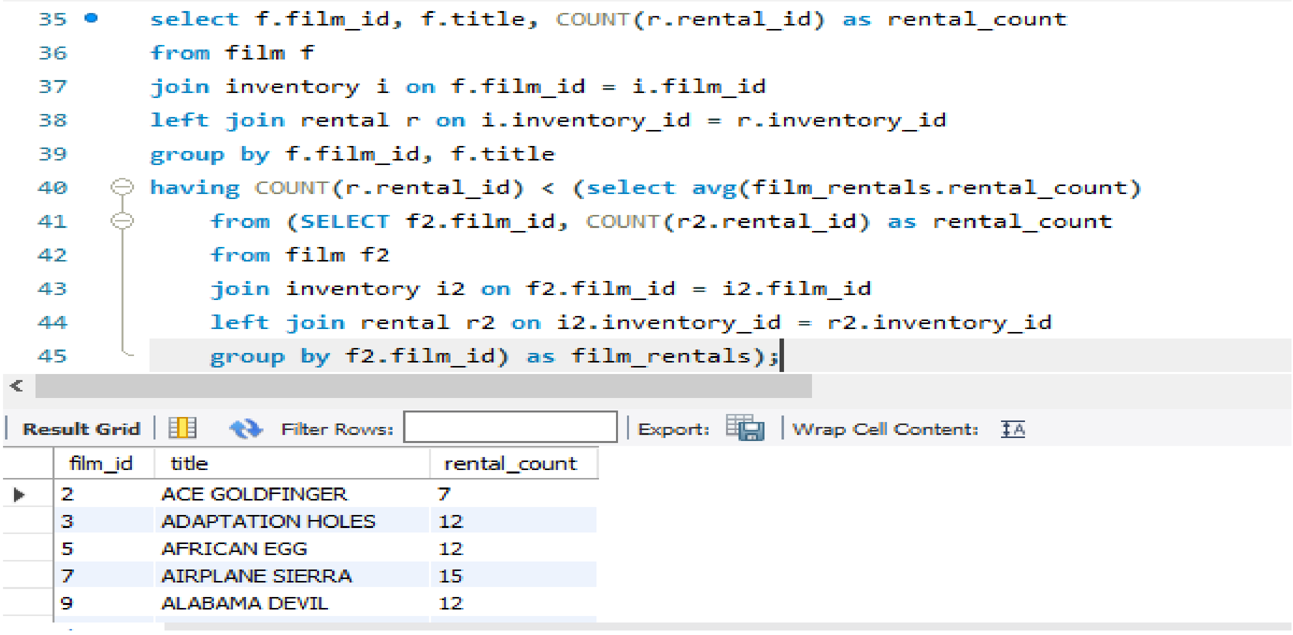
8. Calculate the running total of rentals per category, ordered by rental count.



```sql
select f.title as film_title,
        COUNT(*) as rental_count,
        SUM(COUNT(*)) OVER (ORDER BY COUNT(*) DESC) AS running_total
FROM rental r
INNER JOIN inventory i ON r.inventory_id = i.inventory_id
INNER JOIN film f ON i.film_id = f.film_id
GROUP BY f.title
ORDER BY rental_count desc;
```

film_title	rental_count	running_total
BUCKET BROTHERHOOD	34	34
ROCKETEER MOTHER	33	67
FORWARD TEMPLE	32	227
GRIT CLOCKWORK	32	227
JUGGLER HARDLY	32	227
RIDGEMONT SUBMARINE	32	227

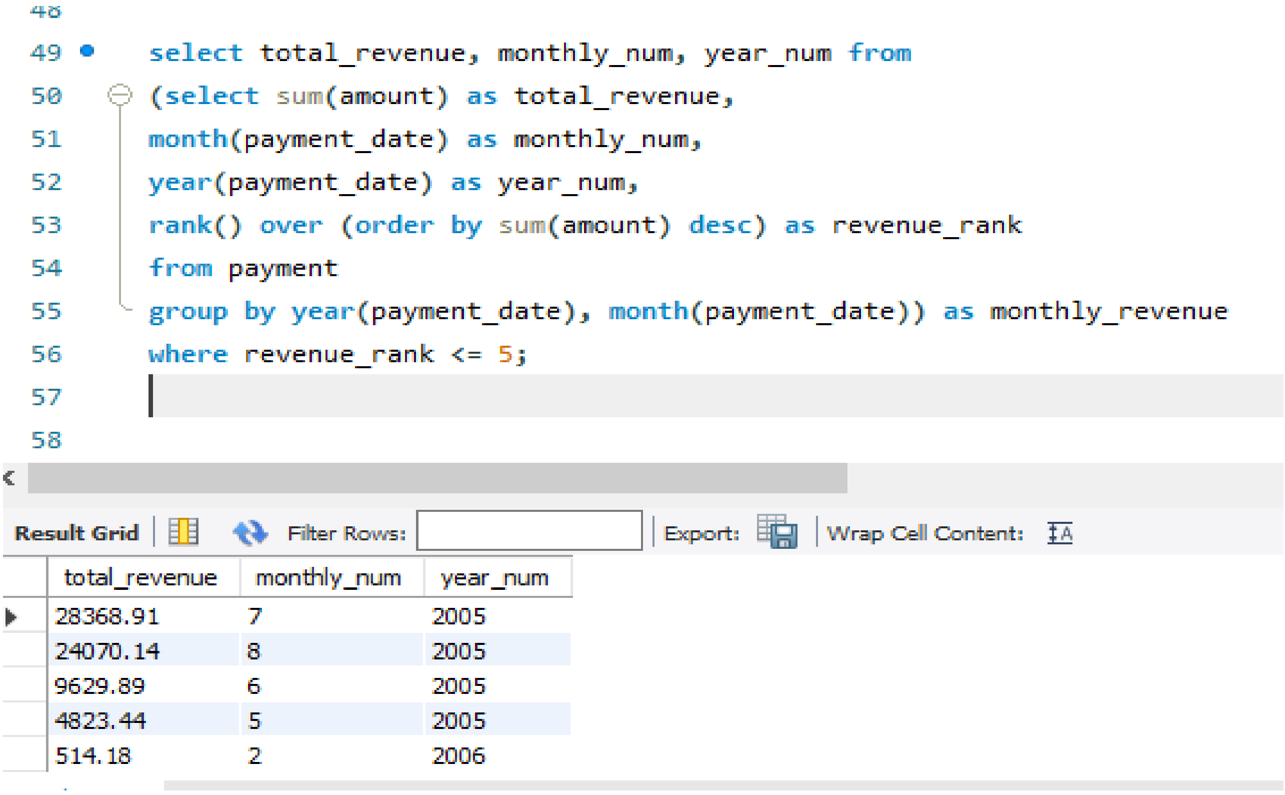
9. Find the films that have been rented less than the average rental count for their respective categories.



```sql
select f.film_id, f.title, COUNT(r.rental_id) as rental_count
from film f
join inventory i on f.film_id = i.film_id
left join rental r on i.inventory_id = r.inventory_id
group by f.film_id, f.title
having COUNT(r.rental_id) < (select avg(film_rentals.rental_count)
    from (SELECT f2.film_id, COUNT(r2.rental_id) as rental_count
    from film f2
    join inventory i2 on f2.film_id = i2.film_id
    left join rental r2 on i2.inventory_id = r2.inventory_id
    group by f2.film_id) as film_rentals);
```

film_id	title	rental_count
2	ACE GOLDFINGER	7
3	ADAPTATION HOLES	12
5	AFRICAN EGG	12
7	AIRPLANE SIERRA	15
9	ALABAMA DEVIL	12

10. Identify the top 5 months with the highest revenue and display the revenue generated in each month.



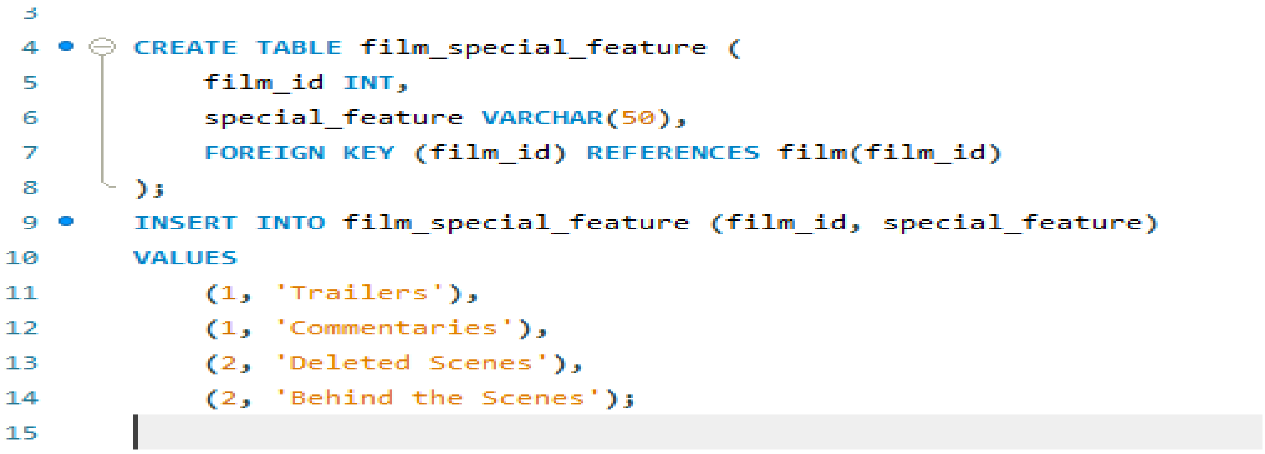
```sql
select total_revenue, monthly_num, year_num from
(select sum(amount) as total_revenue,
month(payment_date) as monthly_num,
year(payment_date) as year_num,
rank() over (order by sum(amount) desc) as revenue_rank
from payment
group by year(payment_date), month(payment_date)) as monthly_revenue
where revenue_rank <= 5;
```

total_revenue	monthly_num	year_num
28368.91	7	2005
24070.14	8	2005
9629.89	6	2005
4823.44	5	2005
514.18	2	2006

Normalisation & CTE

1. First Normal Form (1NF): a. Identify a table in the Sakila database that violates 1NF. Please explain how you would normalize it to achieve 1NF.

In the Sakila database, the film table violates First Normal Form (1NF) because the special_features column contains multiple values (e.g., 'Trailers, Commentaries'), which are not atomic.



```
CREATE TABLE film_special_feature (
    film_id INT,
    special_feature VARCHAR(50),
    FOREIGN KEY (film_id) REFERENCES film(film_id)
);
INSERT INTO film_special_feature (film_id, special_feature)
VALUES
    (1, 'Trailers'),
    (1, 'Commentaries'),
    (2, 'Deleted Scenes'),
    (2, 'Behind the Scenes');
```

2. Second Normal Form (2NF): a. Choose a table in Sakila and describe how you would determine whether it is in 2NF. If it violates 2NF, explain the steps to normalize it.

To determine if a table in Sakila is in Second Normal Form (2NF), we follow these steps:

1. Ensure the table is in 1NF (no repeating groups, atomic values).
2. Check for partial dependencies: Ensure that all non-key columns depend on the entire composite primary key, not just part of it.

Example: film_actor Table

- Primary Key: (actor_id, film_id)
- Non-key column: last_update

Since last_update depends on the full primary key (actor_id, film_id), there are no partial dependencies. Therefore, the film_actor table is in 2NF.

If Violates 2NF:

If a table had partial dependencies, we would:

1. Identify non-key columns that depend only on part of the composite primary key.
2. Move these columns to a new table, creating a foreign key relationship to the original table.

3. Third Normal Form (3NF): a. Identify a table in Sakila that violates 3NF. Describe the transitive dependencies present and outline the steps to normalize the table to 3NF.

Table Violating 3NF in Sakila: actor

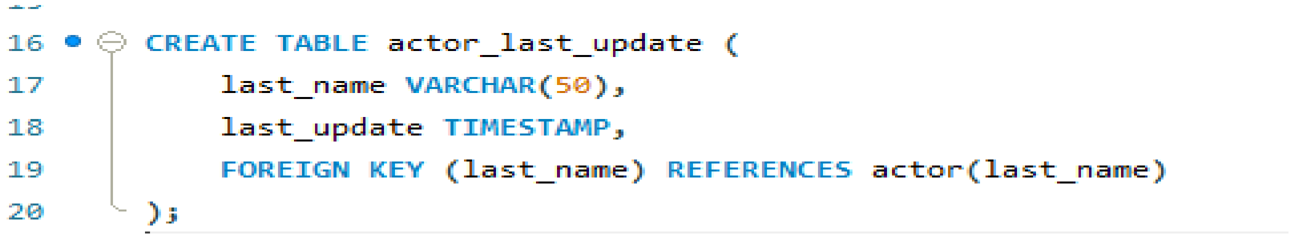
- Primary Key: actor_id
- Non-key Columns: first_name, last_name, last_update

Transitive Dependency:

- last_update depends on last_name, which is a non-key column, rather than directly on the primary key actor_id. This violates 3NF.

Steps to Normalize to 3NF:

1. Remove last_update from the actor table.
2. Create a new table to store last_update with last_name as a foreign key:



```sql
CREATE TABLE actor_last_update (
    last_name VARCHAR(50),
    last_update TIMESTAMP,
    FOREIGN KEY (last_name) REFERENCES actor(last_name)
);
```

4. Normalization Process: a. Take a specific table in Sakila and guide through the process of normalizing it from the initial unnormalized form up to at least 2NF.

Step 1: Unnormalized Form (UNF)

Assume the film table with the columns having film_id, title and special_features.

Step 2: First Normal Form (1NF)

To bring it to 1NF, we remove multi-valued attributes by splitting the special_features into separate row.

Step 3: Second Normal Form (2NF)

To bring it to 2NF, we:

- Ensure the table is in 1NF.
- Remove partial dependencies. The title depends only on film_id, not the full composite key (film_id, special_feature).

Normalization to 2NF:

1. Move title to a separate table.
2. Create a foreign key relationship between film and film_special_feature.

Updated Tables:

- film table:

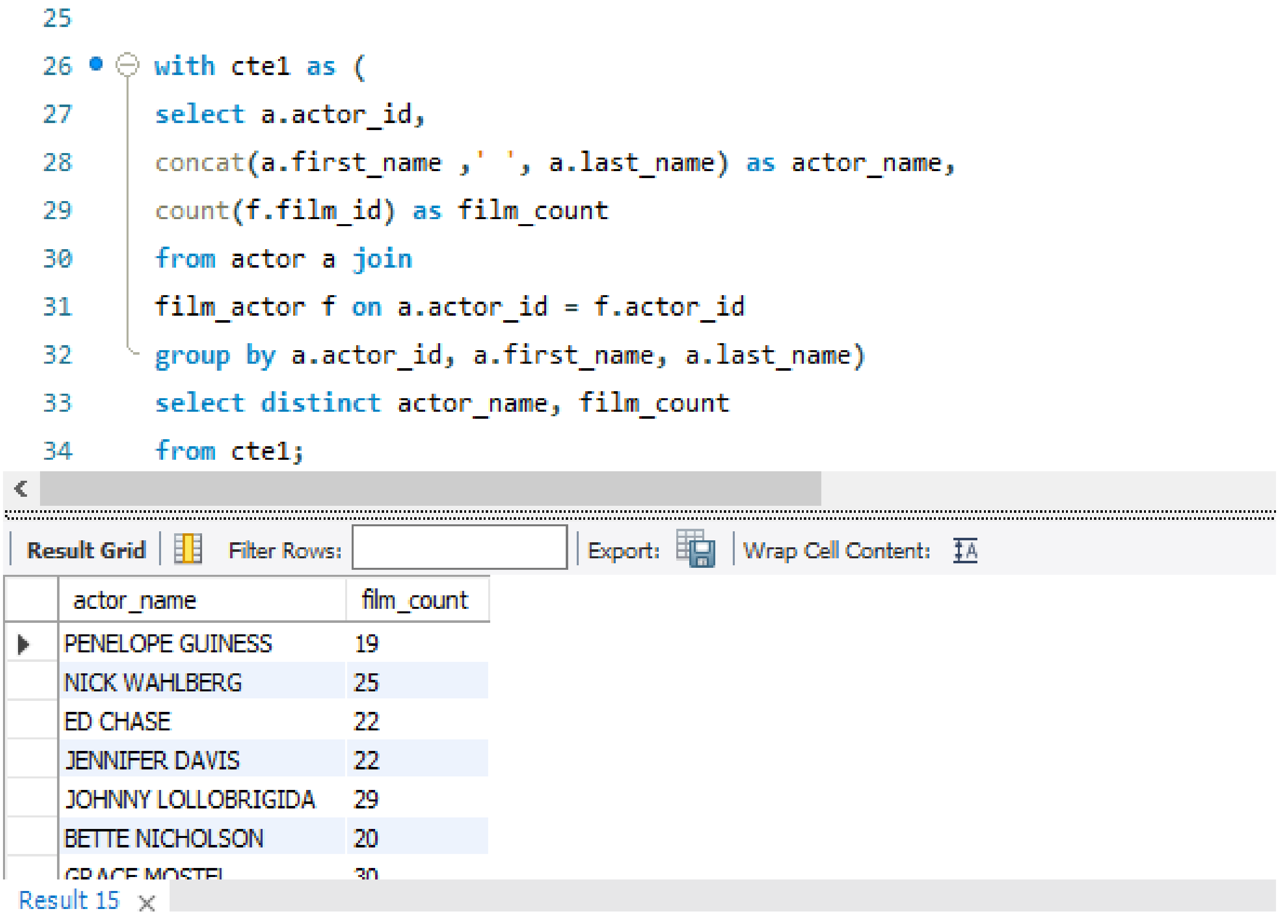
film_id	title
1	Movie A
2	Movie B

- film_special_feature table:

film_id	special_feature
1	Trailers
1	Commentaries
2	Deleted Scenes
2	Behind the Scenes

Now the tables are in 2NF.

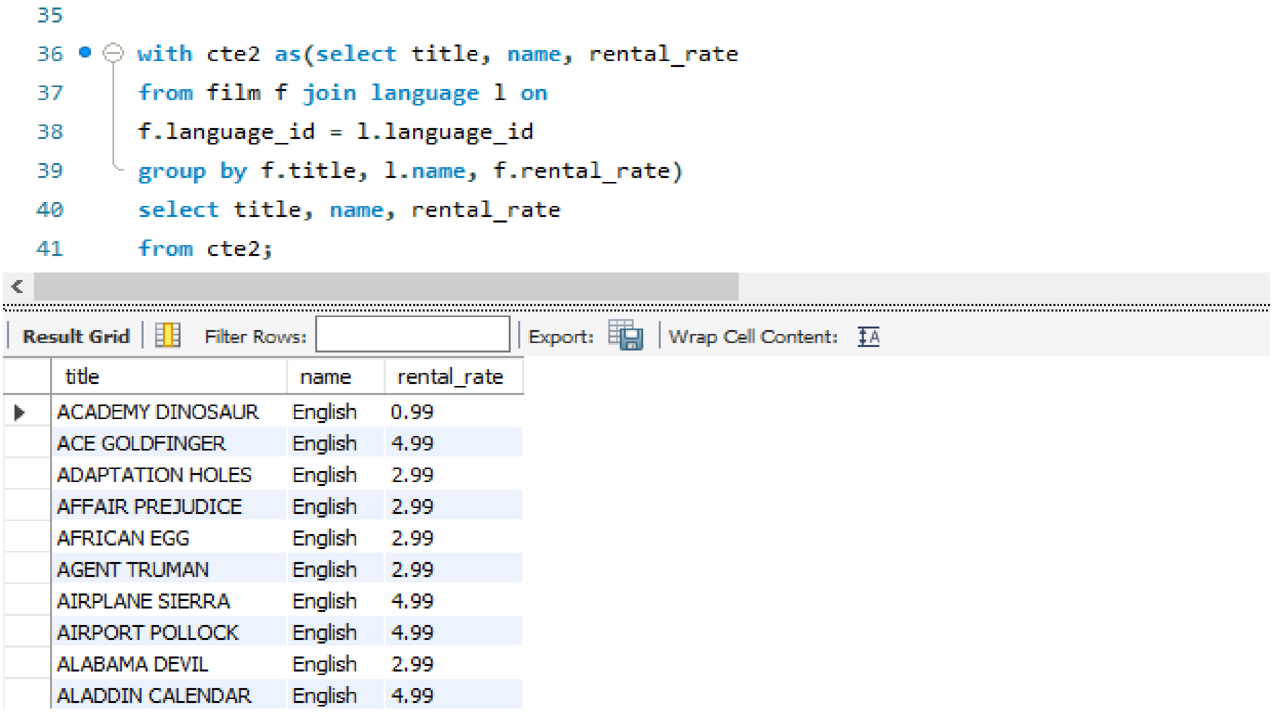
5. CTE Basics: a. Write a query using a CTE to retrieve the distinct list of actor names and the number of films they have acted in from the actor and film_actor tables.



```
with cte1 as (
select a.actor_id,
concat(a.first_name ,' ', a.last_name) as actor_name,
count(f.film_id) as film_count
from actor a join
film_actor f on a.actor_id = f.actor_id
group by a.actor_id, a.first_name, a.last_name)
select distinct actor_name, film_count
from cte1;
```

actor_name	film_count
PENELOPE GUINESS	19
NICK WAHLBERG	25
ED CHASE	22
JENNIFER DAVIS	22
JOHNNY LOLLOBRIGIDA	29
BETTE NICHOLSON	20
GRACE MOSTEL	30

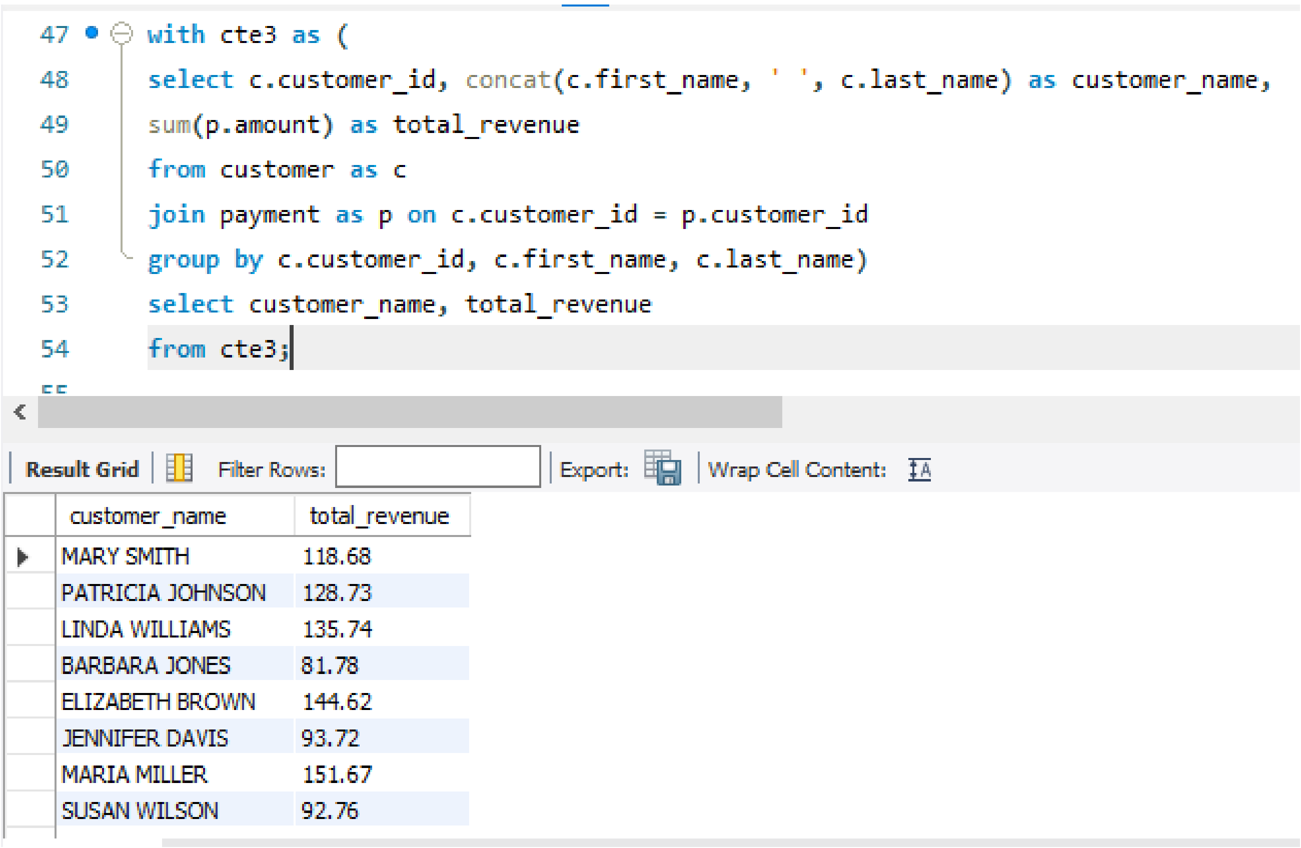
6. CTE with Joins: a. Create a CTE that combines information from the film and language tables to display the film title, language name, and rental rate.



```
with cte2 as(select title, name, rental_rate
from film f join language l on
f.language_id = l.language_id
group by f.title, l.name, f.rental_rate)
select title, name, rental_rate
from cte2;
```

title	name	rental_rate
ACADEMY DINOSAUR	English	0.99
ACE GOLDFINGER	English	4.99
ADAPTATION HOLES	English	2.99
AFFAIR PREJUDICE	English	2.99
AFRICAN EGG	English	2.99
AGENT TRUMAN	English	2.99
AIRPLANE SIERRA	English	4.99
AIRPORT POLLOCK	English	4.99
ALABAMA DEVIL	English	2.99
ALADDIN CALENDAR	English	4.99

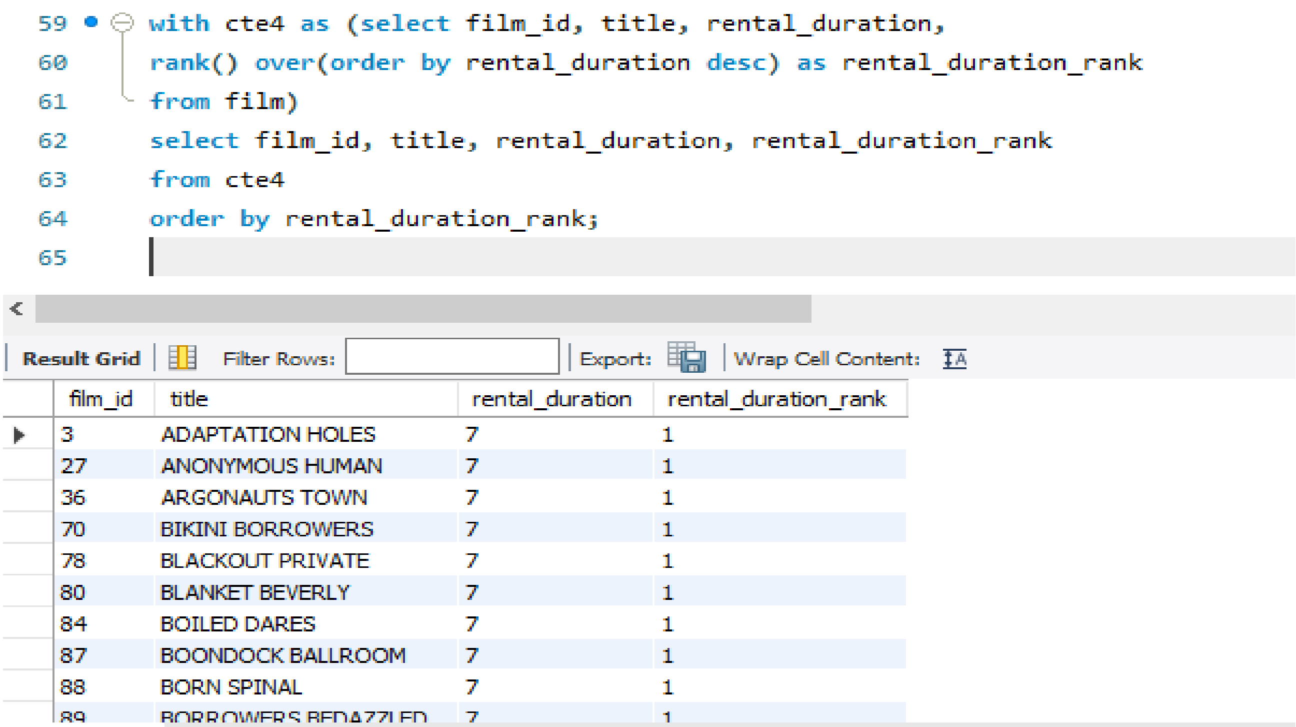
7. CTE for Aggregation: a. Write a query using a CTE to find the total revenue generated by each customer (sum of payments) from the customer and payment tables.



```
with cte3 as (
select c.customer_id, concat(c.first_name, ' ', c.last_name) as customer_name,
sum(p.amount) as total_revenue
from customer as c
join payment as p on c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name)
select customer_name, total_revenue
from cte3;
```

customer_name	total_revenue
MARY SMITH	118.68
PATRICIA JOHNSON	128.73
LINDA WILLIAMS	135.74
BARBARA JONES	81.78
ELIZABETH BROWN	144.62
JENNIFER DAVIS	93.72
MARIA MILLER	151.67
SUSAN WILSON	92.76

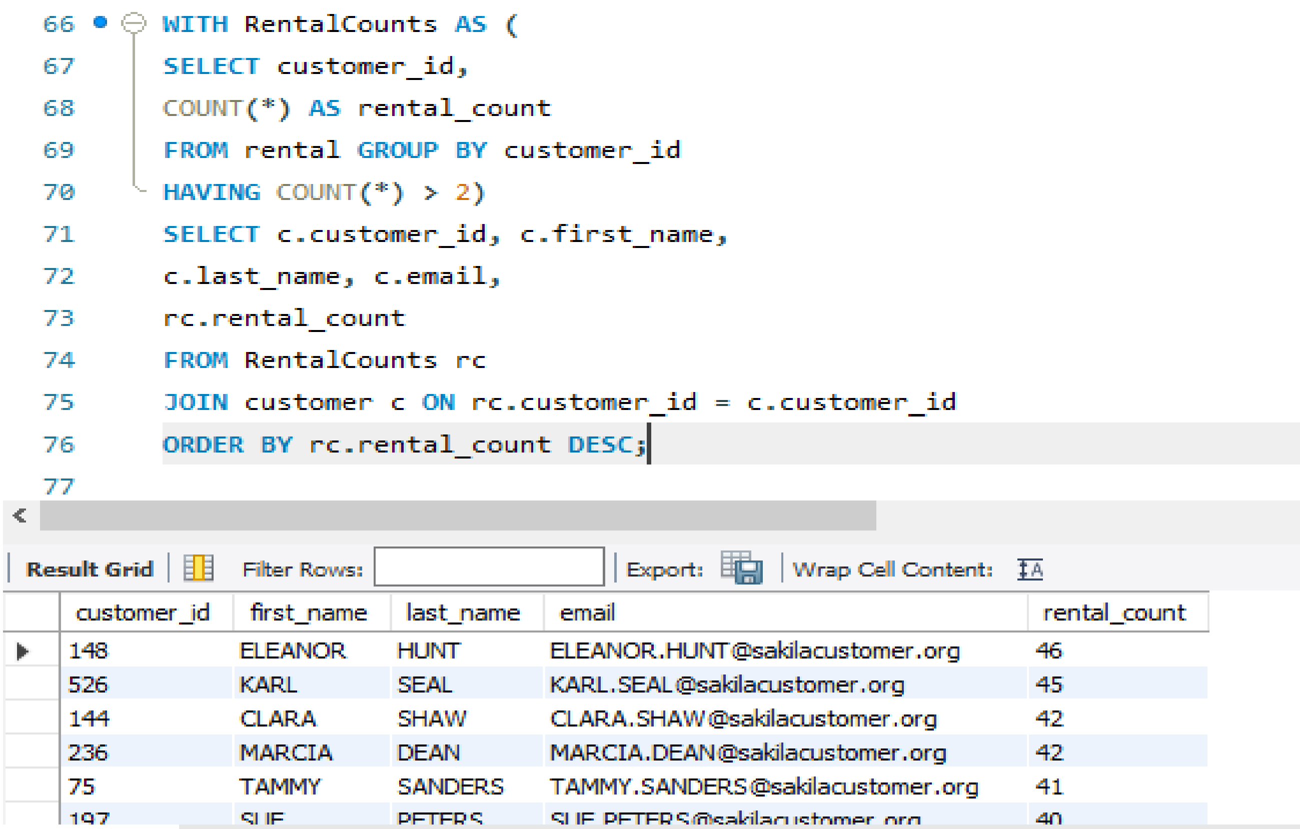
8. CTE with Window Functions: a. Utilize a CTE with a window function to rank films based on their rental duration from the film table.



```sql
with cte4 as (select film_id, title, rental_duration,
rank() over(order by rental_duration desc) as rental_duration_rank
from film)
select film_id, title, rental_duration, rental_duration_rank
from cte4
order by rental_duration_rank;
```

film_id	title	rental_duration	rental_duration_rank
3	ADAPTATION HOLES	7	1
27	ANONYMOUS HUMAN	7	1
36	ARGONAUTS TOWN	7	1
70	BIKINI BORROWERS	7	1
78	BLACKOUT PRIVATE	7	1
80	BLANKET BEVERLY	7	1
84	BOILED DARES	7	1
87	BOONDOCK BALLROOM	7	1
88	BORN SPINAL	7	1
89	BORROWERS BEDAZZLED	7	1

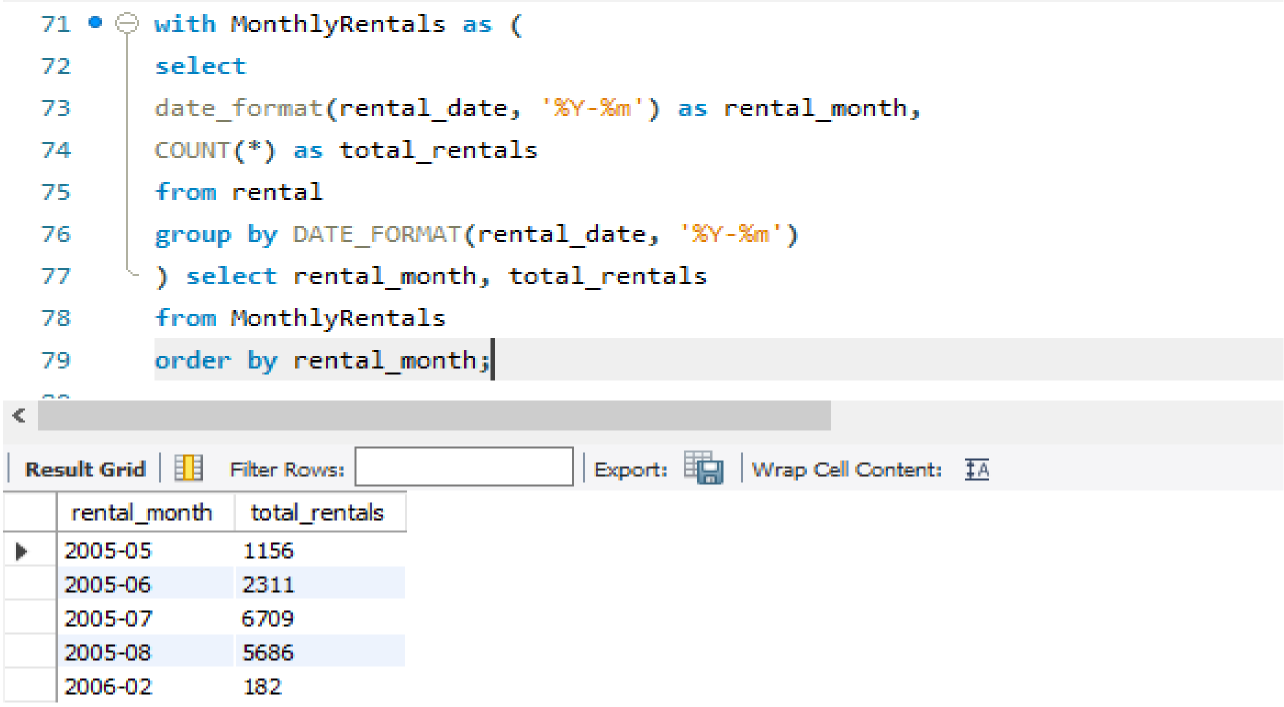
9. CTE and Filtering: a. Create a CTE to list customers who have made more than two rentals, and then join this CTE with the customer table to retrieve additional customer details.



```sql
WITH RentalCounts AS (
SELECT customer_id,
COUNT(*) AS rental_count
FROM rental GROUP BY customer_id
HAVING COUNT(*) > 2)
SELECT c.customer_id, c.first_name,
c.last_name, c.email,
rc.rental_count
FROM RentalCounts rc
JOIN customer c ON rc.customer_id = c.customer_id
ORDER BY rc.rental_count DESC;
```

customer_id	first_name	last_name	email	rental_count
148	ELEANOR	HUNT	ELEANOR.HUNT@sakilacustomer.org	46
526	KARL	SEAL	KARL.SEAL@sakilacustomer.org	45
144	CLARA	SHAW	CLARA.SHAW@sakilacustomer.org	42
236	MARCIA	DEAN	MARCIA.DEAN@sakilacustomer.org	42
75	TAMMY	SANDERS	TAMMY.SANDERS@sakilacustomer.org	41
197	SUE	PETERS	SUE.PETERS@sakilacustomer.org	40

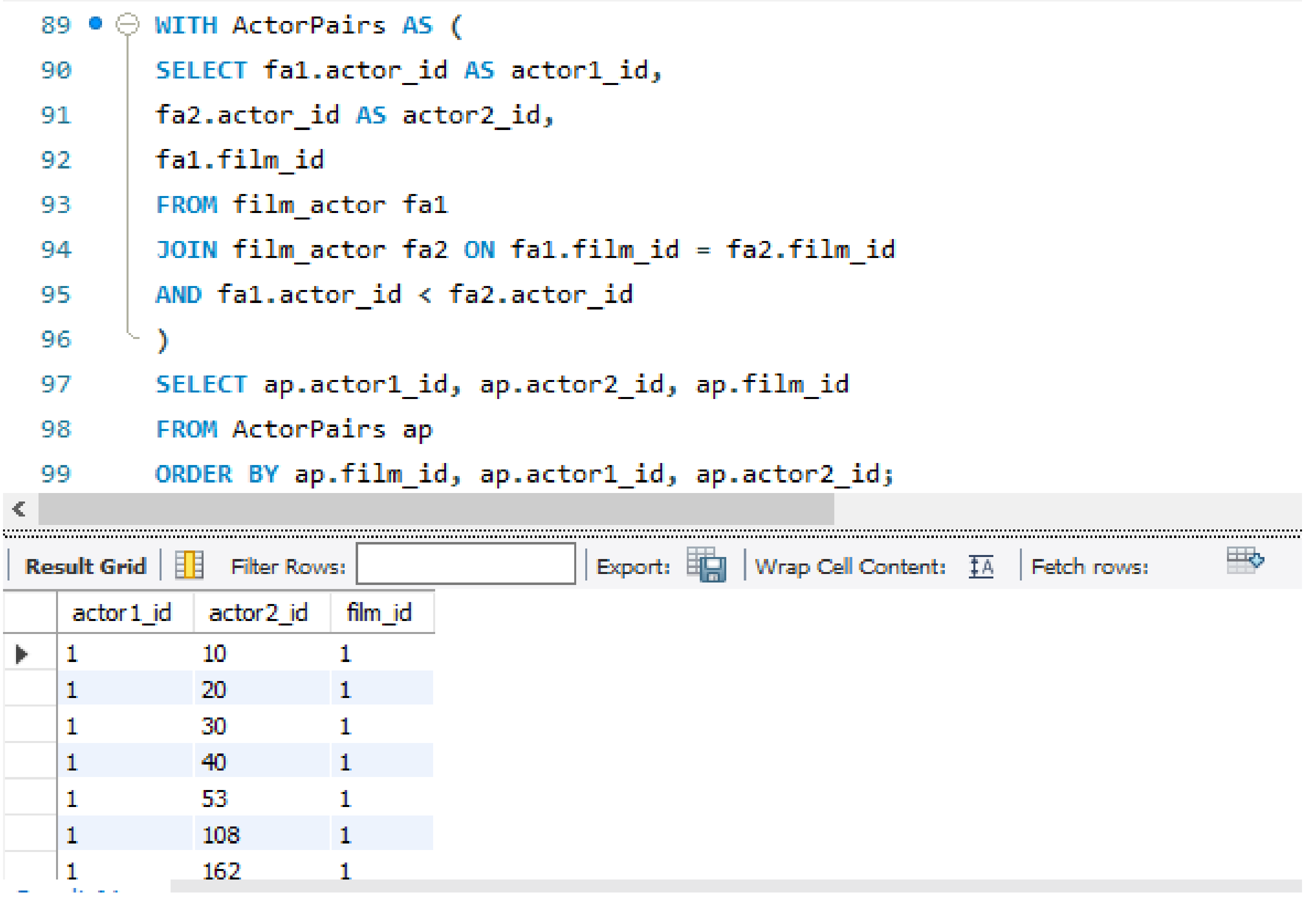
10. CTE for Date Calculations: a. Write a query using a CTE to find the total number of rentals made each month, considering the rental_date from the rental table.



```
with MonthlyRentals as (
select
date_format(rental_date, '%Y-%m') as rental_month,
COUNT(*) as total_rentals
from rental
group by DATE_FORMAT(rental_date, '%Y-%m')
) select rental_month, total_rentals
from MonthlyRentals
order by rental_month;
```

rental_month	total_rentals
2005-05	1156
2005-06	2311
2005-07	6709
2005-08	5686
2006-02	182

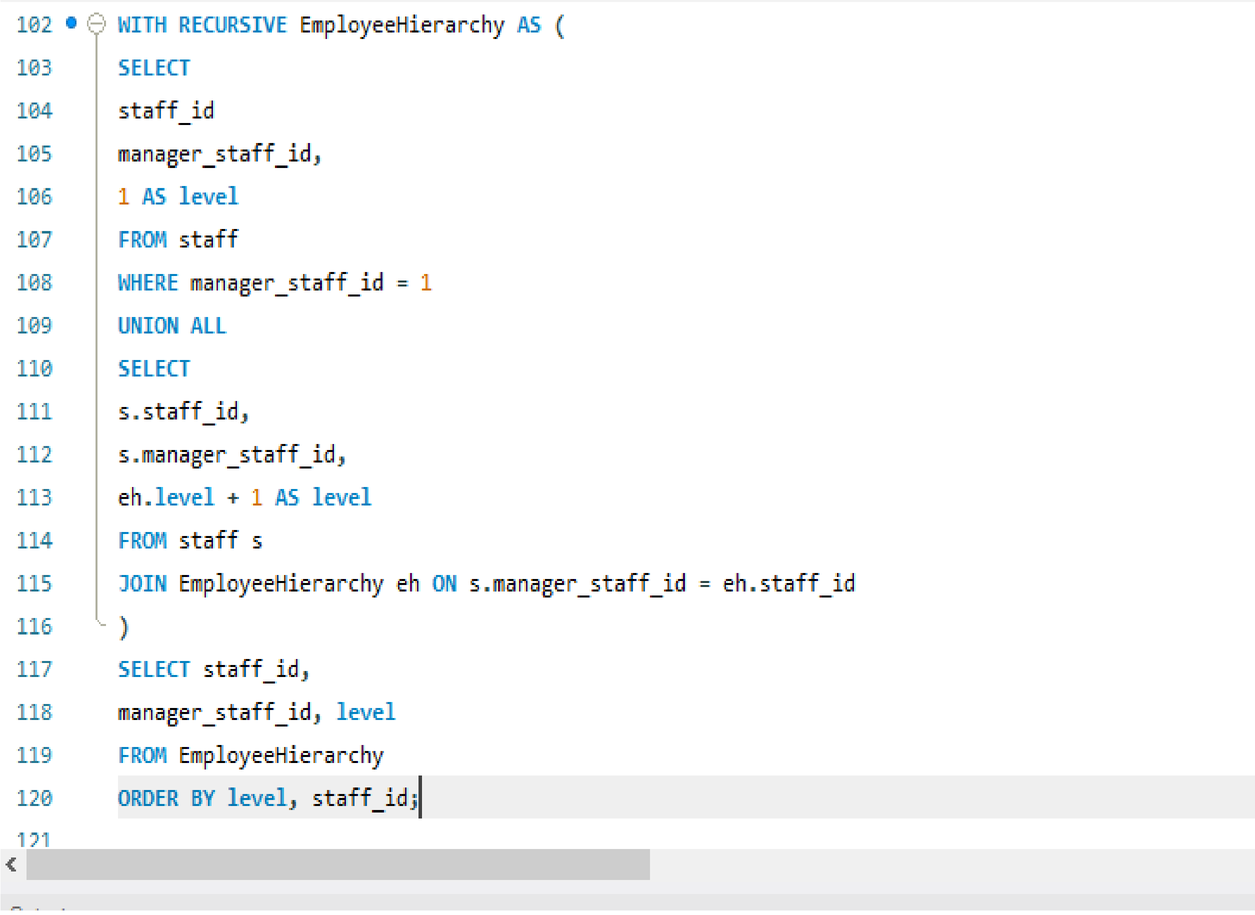
11. CTE and Self-Join: a. Create a CTE to generate a report showing pairs of actors who have appeared in the same film together, using the film_actor table.



```
WITH ActorPairs AS (
SELECT fa1.actor_id AS actor1_id,
fa2.actor_id AS actor2_id,
fa1.film_id
FROM film_actor fa1
JOIN film_actor fa2 ON fa1.film_id = fa2.film_id
AND fa1.actor_id < fa2.actor_id
)
SELECT ap.actor1_id, ap.actor2_id, ap.film_id
FROM ActorPairs ap
ORDER BY ap.film_id, ap.actor1_id, ap.actor2_id;
```

actor1_id	actor2_id	film_id
1	10	1
1	20	1
1	30	1
1	40	1
1	53	1
1	108	1
1	162	1

12. CTE for Recursive Search: a. Implement a recursive CTE to find all employees in the staff table who report to a specific manager, considering the reports_to column.



```
WITH RECURSIVE EmployeeHierarchy AS (
SELECT
staff_id
manager_staff_id,
1 AS level
FROM staff
WHERE manager_staff_id = 1
UNION ALL
SELECT
s.staff_id,
s.manager_staff_id,
eh.level + 1 AS level
FROM staff s
JOIN EmployeeHierarchy eh ON s.manager_staff_id = eh.staff_id
)
SELECT staff_id,
manager_staff_id, level
FROM EmployeeHierarchy
ORDER BY level, staff_id;
```