Business Data Management
Final Project Report

Group 4

Under the guidance of Professor Meng, Qu

Deviprasad Saka
Kalyan Venkat Mangipudi
Sathya Siva Sairam Kamma

12.20.2023
Information Technology and Analytics

INTRODUCTION

Our project focuses on the critical task of working with raw datasets and importing them into a SQL database. The primary objective is to empower decision-makers with valuable insights through the creation of an Entity-Relationship (ER) diagram and the development of SQL code for effective data extraction. We have also created visualization using the tool Tableau for gaining better insights about the data and presenting the results to the stakeholders.

PROJECT SCOPE AND AIM

The aim of this project is to work with raw dataset and import them into the SQL Database for Informed managerial decisions. We have modeled a DVD rental store database, featuring things like films, actors, film-actor relationships, and a central inventory table that connects films, stores, and rentals.The scope of our project is defined by the need to transform raw data into actionable information. Our aim is to enable strategic decision-making by creating a robust ER diagram and implementing SQL code to extract meaningful insights from the datasets. For the sake of this project, we have leveraged the database MySQL to perform all kinds of database operations.

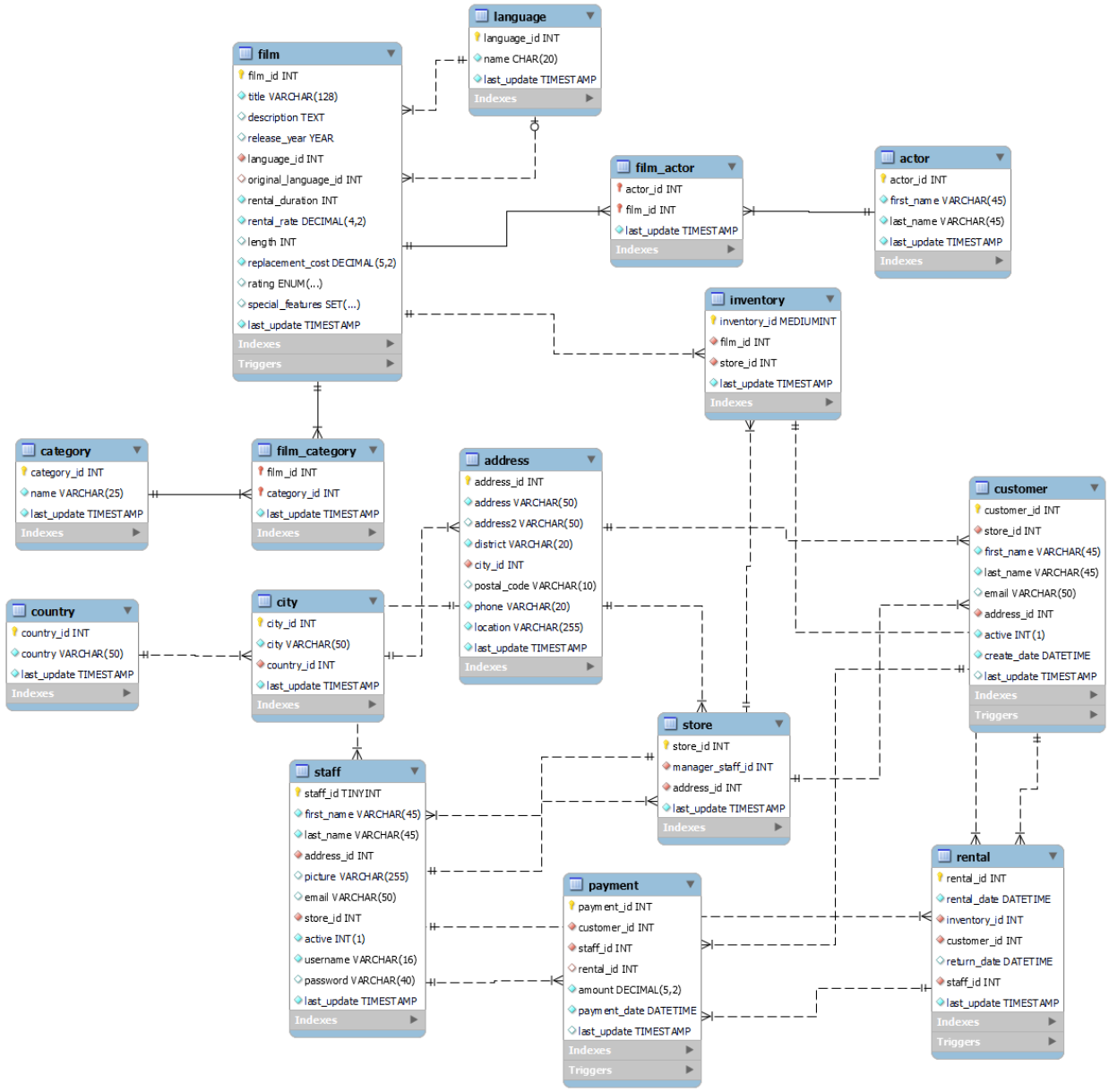
RAW DATASETS

We worked with a DVD rental store dataset which is a centralized database, a collection of data with featuring things like films, actors, film-actor relationships. This data forms the foundation of our project, allowing us to derive valuable insights for managerial decision-making.

ER DIAGRAM CREATION

Our project began with the creation of a comprehensive Entity-Relationship (ER) diagram. This diagram visually represents the relationships between entities, their attributes, and the overall structure of the data. The key elements of the ER diagram include the tables, column names and nature of the relationship between entities. Also defined are the constraints on each table which help in maintaining database consistency and integrity.

ER Diagram of the Data Model

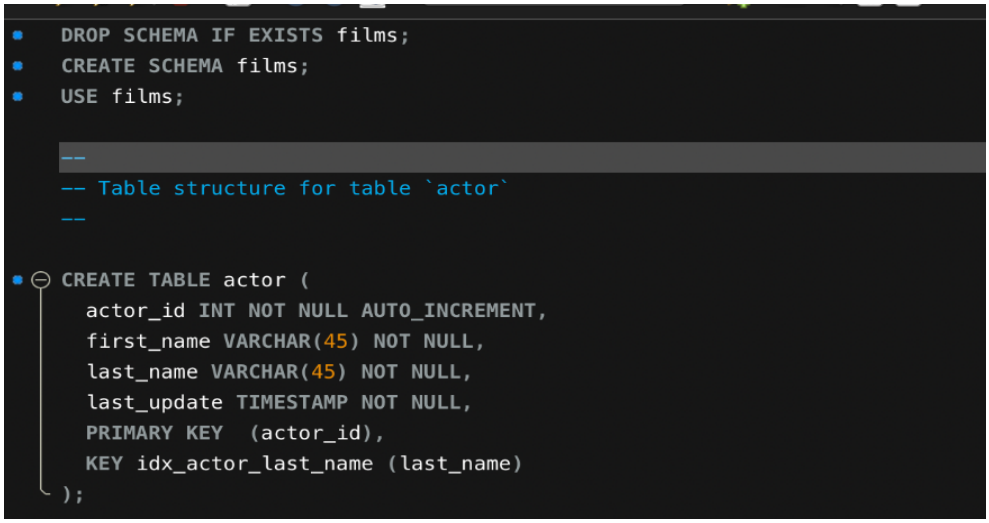


SQL Code Implementation

To operationalize our insights, we wrote SQL code to import the raw datasets into an SQL database. The code not only ensured efficient data storage but also laid the groundwork

for subsequent extraction queries. At the same time, in order to reduce any inconsistency in the data, .csv files have been directly imported into the database for the creation of tables.

Example below shows the creation of schema in MySql database and creating first table named actors in the database.



```
DROP SCHEMA IF EXISTS films;
CREATE SCHEMA films;
USE films;

--
-- Table structure for table `actor`
--

CREATE TABLE actor (
  actor_id INT NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  last_update TIMESTAMP NOT NULL,
  PRIMARY KEY  (actor_id),
  KEY idx_actor_last_name (last_name)
);
```

Dataset Cleaning

Data cleaning is a crucial step in preparing raw datasets for our analysis, hence we have used some data cleaning techniques in SQL to check for any inconsistencies and correct them. Below mentioned are some of the data cleaning techniques that have operated on the data to clean the dataset.

1. **Handling Missing Data**

-- 'length' column in 'film' table has missing values. Update missing values with the average length.

UPDATE film

SET length = COALESCE(length, (SELECT AVG(length) FROM film WHERE length IS NOT NULL));

2. 'last_update' column in 'actor' table has missing values. Update missing values with the current timestamp.

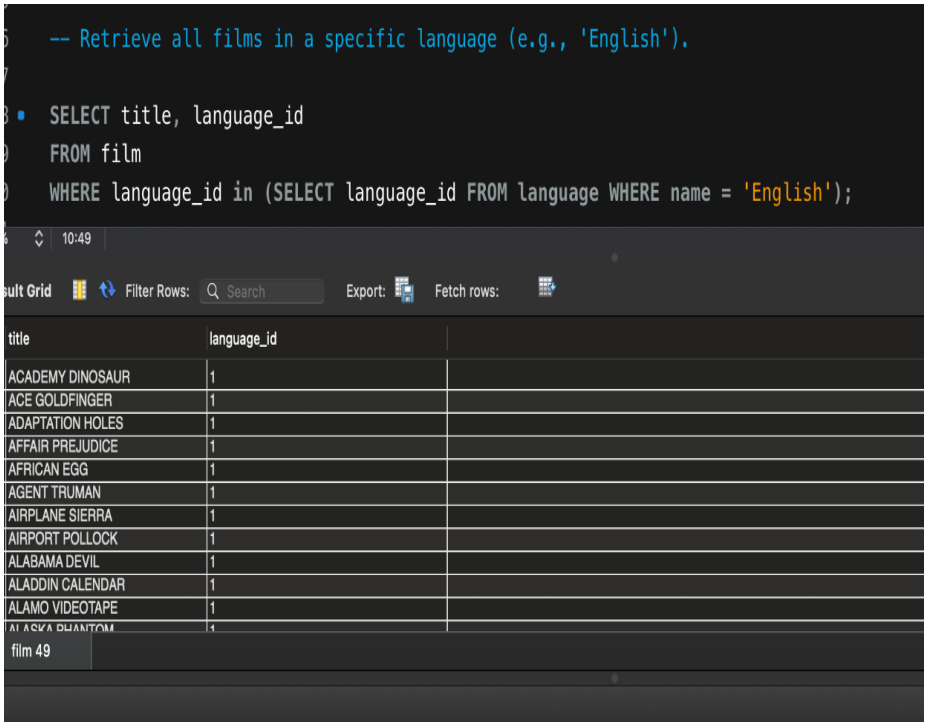
```
UPDATE actor
SET last_update = COALESCE(last_update, CURRENT_TIMESTAMP);
```

3. **Deduplication**

```
-- Remove duplicate actors based on 'first_name' and 'last_name'
DELETE FROM actor
WHERE actor_id NOT IN (
    SELECT MAX(actor_id)
    FROM actor
    GROUP BY first_name, last_name
);
```

Useful Information Extraction

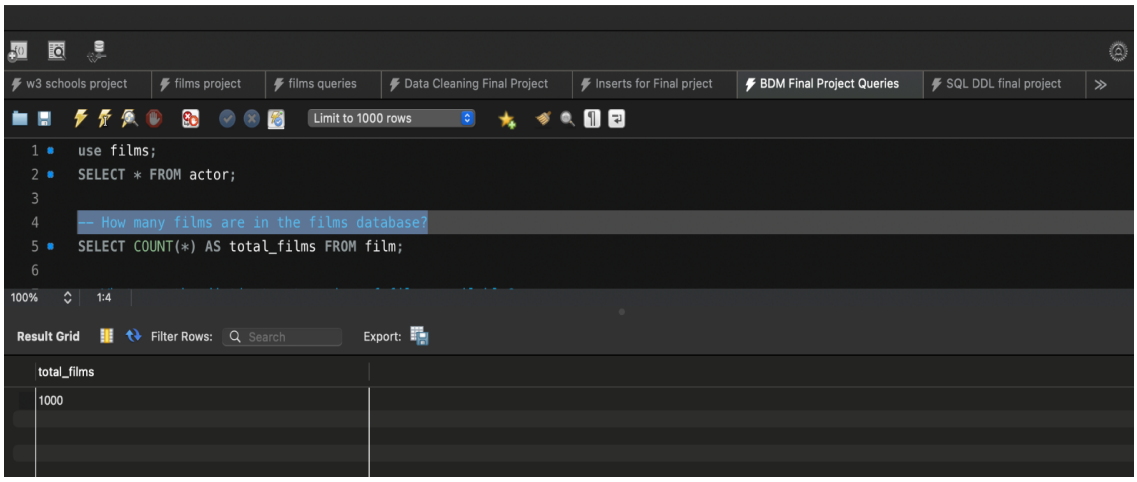
Through strategic SQL queries, we extracted useful information from the database. Our focus was on generating insights that directly contribute to informed managerial decision-making. For example, we wanted to retrieve all films in a specific language (i.e., English). We can write a query for the same using the required tables and obtain results for making decisions.



SQL Queries

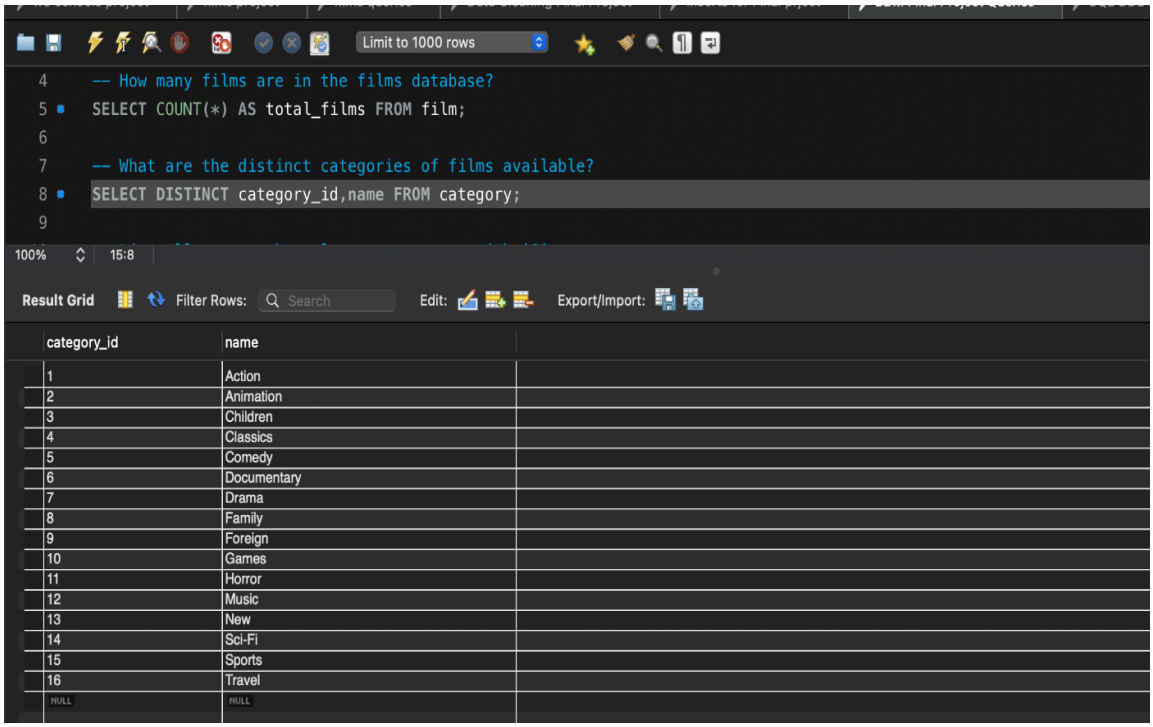
1. **How many films are in the films database?**

   SELECT COUNT(*) AS total_films FROM film;



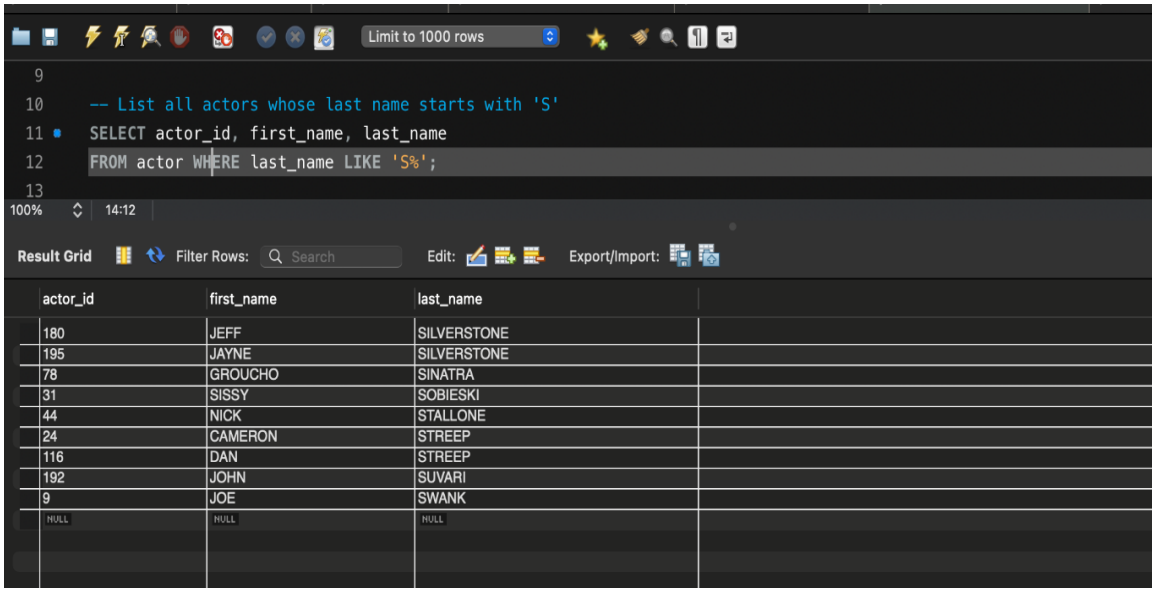
2. **What are the distinct categories of films available?**

   SELECT DISTINCT category_id,name FROM category;



3. **List all actors whose last name starts with 'S'.**

SELECT actor_id, first_name, last_name
FROM actor WHERE last_name LIKE 'S%';



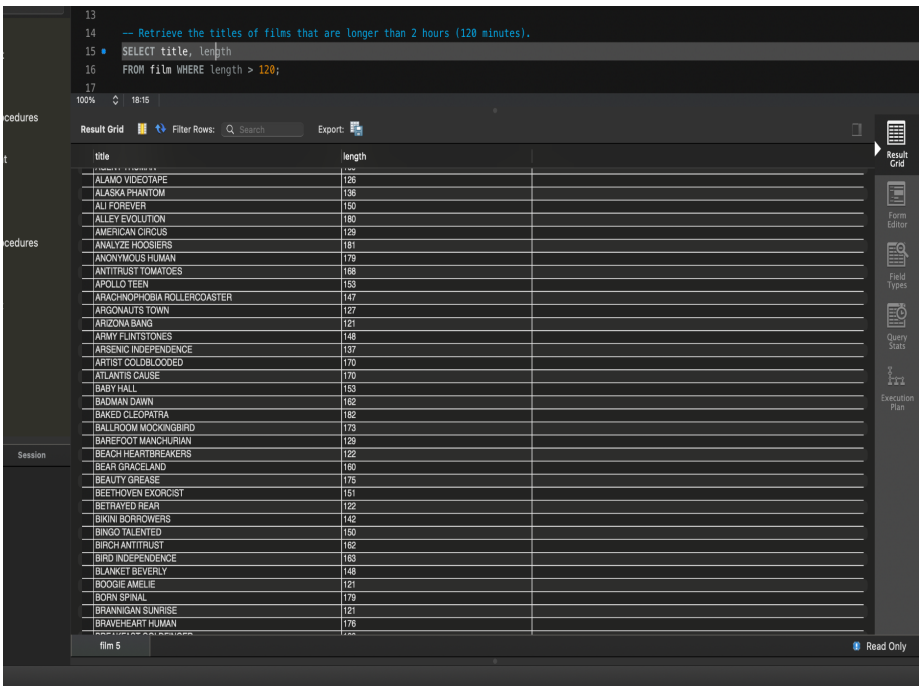
```
-- List all actors whose last name starts with 'S'
SELECT actor_id, first_name, last_name
FROM actor WHERE last_name LIKE 'S%';
```

actor_id	first_name	last_name
180	JEFF	SILVERSTONE
195	JAYNE	SILVERSTONE
78	GROUCHO	SINATRA
31	SISSY	SOBIESKI
44	NICK	STALLONE
24	CAMERON	STREEP
116	DAN	STREEP
192	JOHN	SUVARI
9	JOE	SWANK
NULL	NULL	NULL

4. **Retrieve the titles of films that are longer than 2 hours (120 minutes).**

SELECT title, length

FROM film WHERE length > 120;

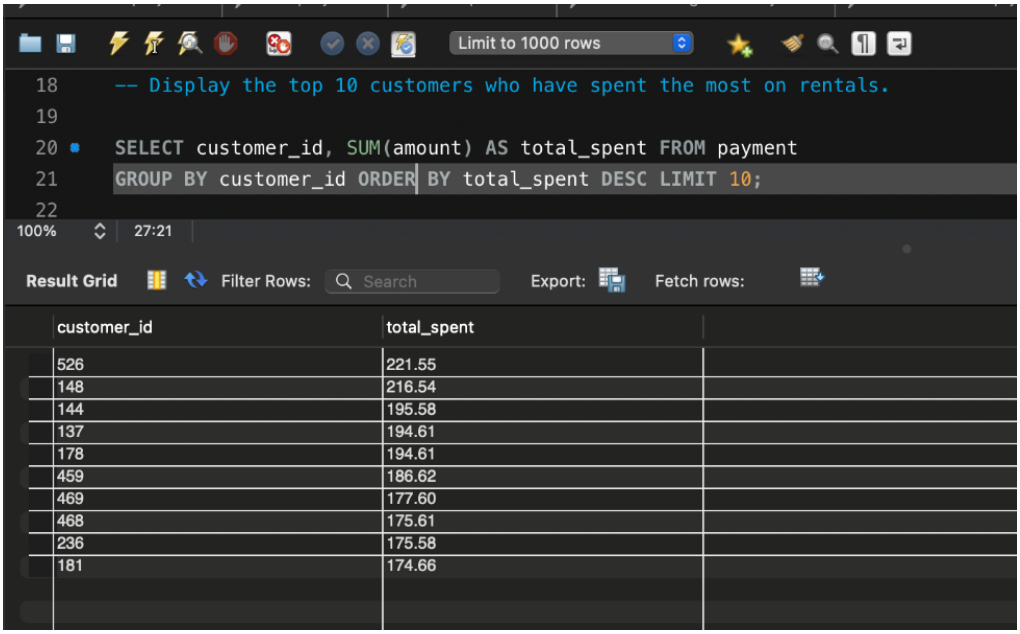


```
-- Retrieve the titles of films that are longer than 2 hours (120 minutes).
SELECT title, length
FROM film WHERE length > 120;
```

title	length
ALAMO VIDEOTAPE	126
ALASKA PHANTOM	136
ALI FOREVER	150
ALLEY EVOLUTION	180
AMERICAN CIRCUS	129
ANALYZE HOOSIERS	181
ANONYMOUS HUMAN	179
ANTITRUST TOMATOES	168
APOLLO TEEN	153
ARACHNOPHOBIA ROLLERCOASTER	147
ARGONAUTS TOWN	127
ARIZONA BANG	121
ARMY FLINTSTONES	148
ARSENIC INDEPENDENCE	137
ARTIST COLDBLOODED	170
ATLANTIS CAUSE	170
BABY HALL	153
BADMAN DAWN	162
BAKED CLEOPATRA	182
BALLROOM MOCKINGBIRD	173
BAREFOOT MANCHURIAN	129
BEACH HEARTBREAKERS	122
BEAR GRACELAND	160
BEAUTY GREASE	175
BEETHOVEN EXORCIST	151
BETRAYED REAR	122
BIKINI BORROWERS	142
BINGO TALENTED	150
BIRCH ANTITRUST	162
BIRD INDEPENDENCE	163
BLANKET BEVERLY	148
BOOGIE AMELIE	121
BORN SPINAL	179
BRANNIGAN SUNRISE	121
BRAVEHEART HUMAN	176

5. **Display the top 10 customers who have spent the most on rentals.**

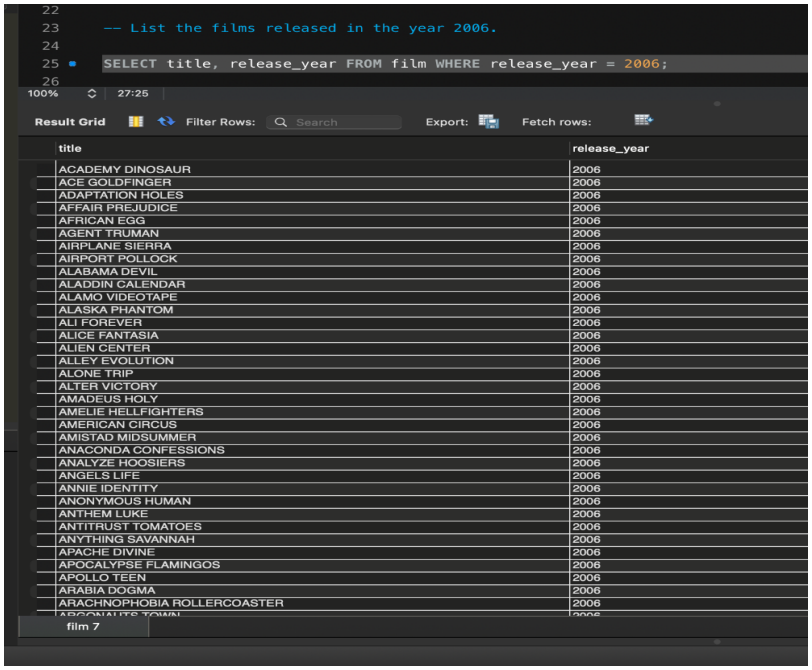
   SELECT customer_id, SUM(amount) AS total_spent FROM payment GROUP BY customer_id ORDER BY total_spent DESC LIMIT 10;



customer_id	total_spent
526	221.55
148	216.54
144	195.58
137	194.61
178	194.61
459	186.62
469	177.60
468	175.61
236	175.58
181	174.66

6. **List the films released in the year 2006.**

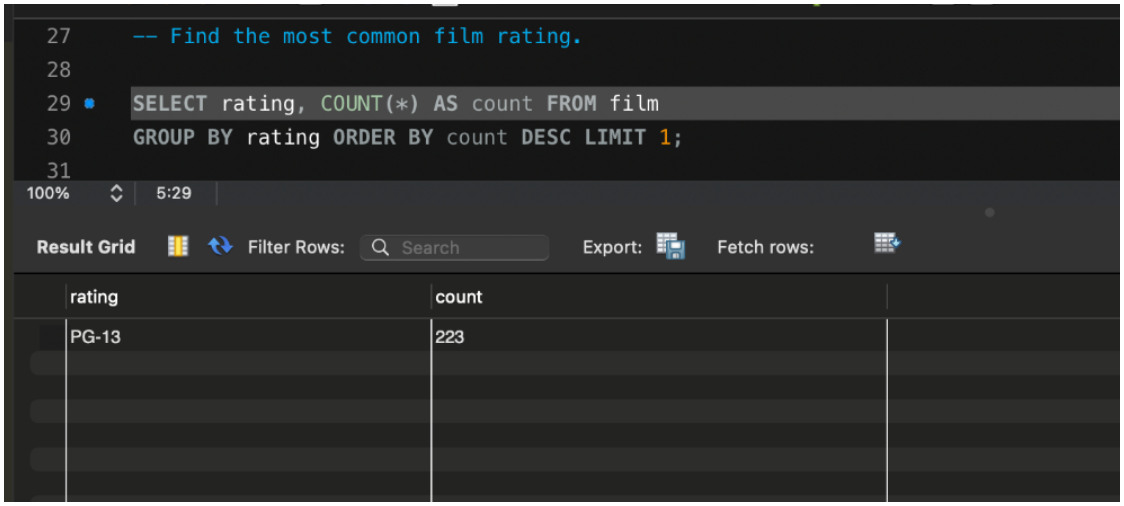
   SELECT title, release_year FROM film WHERE release_year = 2006;



title	release_year
ACADEMY DINOSAUR	2006
ACE GOLDFINGER	2006
ADAPTATION HOLES	2006
AFFAIR PREJUDICE	2006
AFRICAN EGG	2006
AGENT TRUMAN	2006
AIRPLANE SIERRA	2006
AIRPORT POLLOCK	2006
ALABAMA DEVIL	2006
ALADDIN CALENDAR	2006
ALAMO VIDEOTAPE	2006
ALASKA PHANTOM	2006
ALI FOREVER	2006
ALICE FANTASIA	2006
ALIEN CENTER	2006
ALLEY EVOLUTION	2006
ALONE TRIP	2006
ALTER VICTORY	2006
AMADEUS HOLY	2006
AMELIE HELLFIGHTERS	2006
AMERICAN CIRCUS	2006
AMISTAD MIDSUMMER	2006
ANACONDA CONFESSIONS	2006
ANALYZE HOOSIERS	2006
ANGELS LIFE	2006
ANNIE IDENTITY	2006
ANONYMOUS HUMAN	2006
ANTHEM LUKE	2006
ANTITRUST TOMATOES	2006
ANYTHING SAVANNAH	2006
APACHE DIVINE	2006
APOCALYPSE FLAMINGOS	2006
APOLLO TEEN	2006
ARABIA DOGMA	2006
ARACHNOPHOBIA ROLLERCOASTER	2006

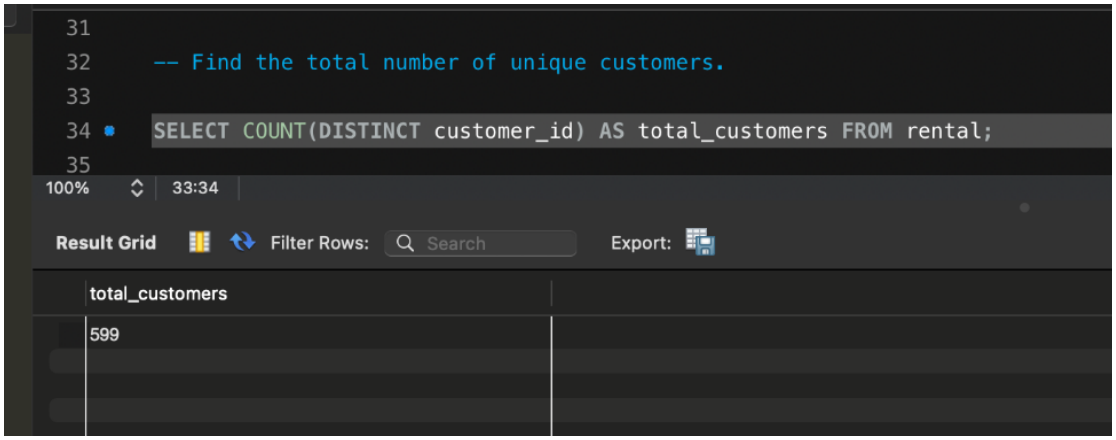
7. **Find the most common film rating.**

SELECT rating, COUNT(*) AS count FROM film
GROUP BY rating ORDER BY count DESC LIMIT 1;



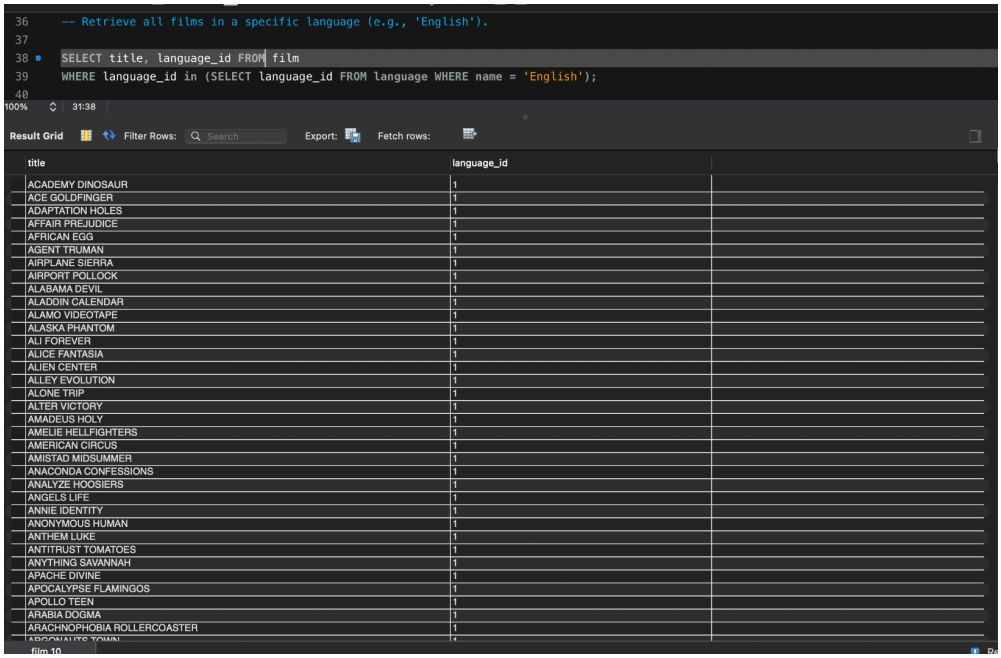
8. **Find the total number of unique customers.**

SELECT COUNT(DISTINCT customer_id) AS total_customers FROM rental;



9. **Retrieve all films in a specific language (e.g., 'English').**

SELECT title, language_id FROM film
WHERE language_id in (SELECT language_id FROM language WHERE name = 'English');



```
36    -- Retrieve all films in a specific language (e.g., 'English').
37
38 •  SELECT title, language_id FROM film
39    WHERE language_id in (SELECT language_id FROM language WHERE name = 'English');
40
```

100% 31:38

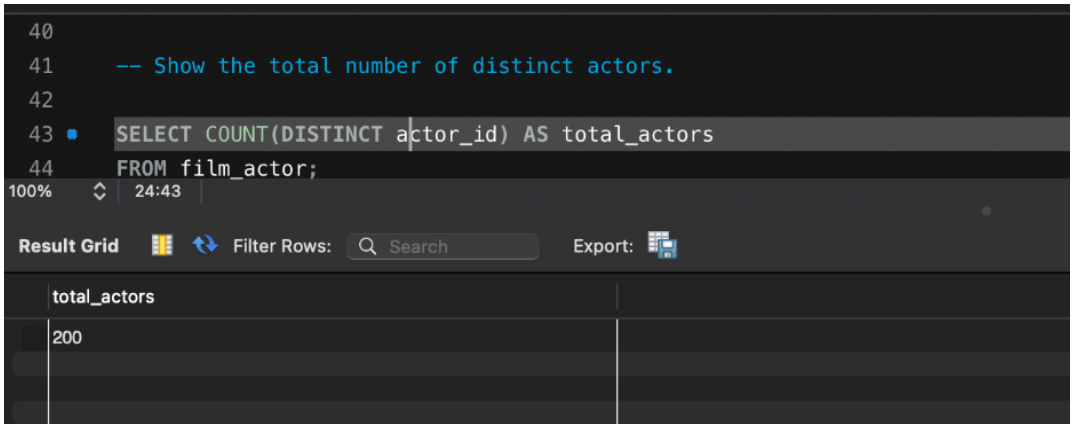
Result Grid Filter Rows: Search Export: Fetch rows:

title	language_id
ACADEMY DINOSAUR	1
ACE GOLDFINGER	1
ADAPTATION HOLES	1
AFFAIR PREJUDICE	1
AFRICAN EGG	1
AGENT TRUMAN	1
AIRPLANE SIERRA	1
AIRPORT POLLOCK	1
ALABAMA DEVIL	1
ALADDIN CALENDAR	1
ALAMO VIDEOTAPE	1
ALASKA PHANTOM	1
ALI FOREVER	1
ALICE FANTASIA	1
ALIEN CENTER	1
ALLEY EVOLUTION	1
ALONE TRIP	1
ALTER VICTORY	1
AMADEUS HOLY	1
AMELIE HELLFIGHTERS	1
AMERICAN CIRCUS	1
AMISTAD MIDSUMMER	1
ANACONDA CONFESSIONS	1
ANALYZE HOOSIERS	1
ANGELS LIFE	1
ANNIE IDENTITY	1
ANONYMOUS HUMAN	1
ANTHEM LUKE	1
ANTITRUST TOMATOES	1
ANYTHING SAVANNAH	1
APACHE DIVINE	1
APOCALYPSE FLAMINGOS	1
APOLLO TEEN	1
ARABIA DOGMA	1
ARACHNOPHOBIA ROLLERCOASTER	1

film 10

10. Show the total number of distinct actors.

```
SELECT COUNT(DISTINCT actor_id) AS total_actors
FROM film_actor;
```



```
40
41    -- Show the total number of distinct actors.
42
43 •  SELECT COUNT(DISTINCT actor_id) AS total_actors
44    FROM film_actor;
```

100% 24:43

Result Grid Filter Rows: Search Export:

total_actors
200

11. Retrieve films with the word 'Love' in the title.

```
SELECT title FROM film
WHERE title LIKE '%Love%';
```

```
45
46    -- Retrieve films with the word 'Love' in the title.
47
48    SELECT title FROM film
49    WHERE title LIKE '%Love%';
```

100% 7:48

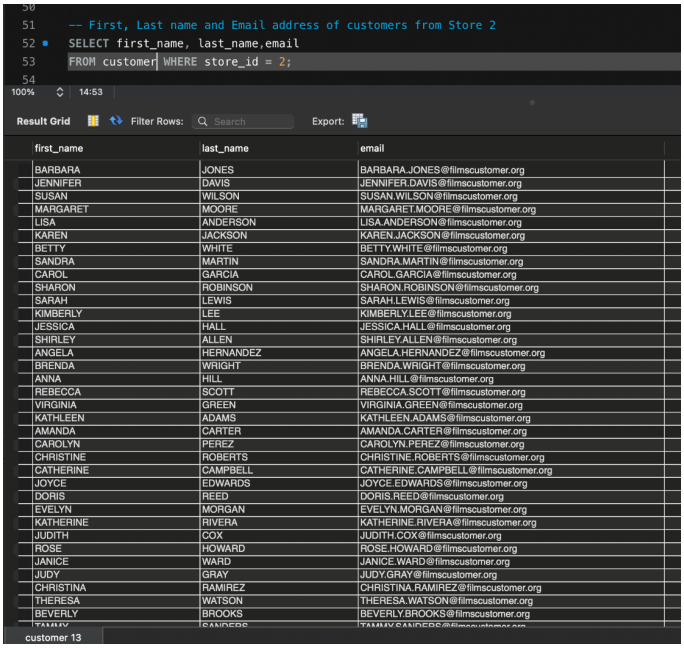
Result Grid | Filter Rows: Search | Export:

title
GRAFFITI LOVE
IDAHO LOVE
IDENTITY LOVER
INDIAN LOVE
LAWRENCE LOVE
LOVE SUICIDES
LOVELY JINGLE
LOVER TRUMAN
LOVERBOY ATTACKS
STRANGELOVE DESIRE

12. First, Last name and Email address of customers from Store 2

SELECT first_name, last_name,email

FROM customer WHERE store_id = 2;



13. We want to see rental rate and how many movies are in each rental rate categories

SELECT rental_rate, COUNT(*) AS total_number_of_movies

FROM film GROUP BY rental_rate;

```
54
55    -- we want to see rental rate and how many movies are in each rental rate categories*/
56    SELECT rental_rate, COUNT(*) AS total_number_of_movies
57    FROM film GROUP BY rental_rate;
58
```

100% 20:57

Result Grid Filter Rows: Search Export:

rental_rate	total_number_of_movies
0.99	341
4.99	336
2.99	323

-- Complex SQL Questions

14. List the top 5 categories with the most films.

SELECT category_id, COUNT(*) AS film_count

FROM film_category GROUP BY category_id ORDER BY film_count DESC LIMIT 5;

```
61    -- List the top 5 categories with the most films.
62
63    SELECT category_id, COUNT(*) AS film_count
64    FROM film_category GROUP BY category_id ORDER BY film_count DESC LIMIT 5;
65
```

100% 15:63

Result Grid Filter Rows: Search Export: Fetch rows:

category_id	film_count
15	74
9	73
8	69
6	68
2	66

15. Calculate the average length of films in each language

SELECT language_id, AVG(length) AS avg_length

FROM film GROUP BY language_id;

```
66    -- Calculate the average length of films in each language.
67
68    SELECT language_id, AVG(length) AS avg_length
69    FROM film GROUP BY language_id;
70
```

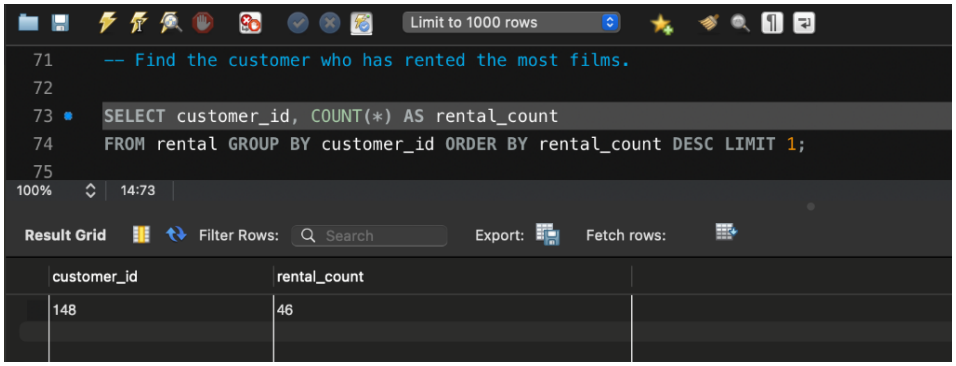
100% 21:69

Result Grid Filter Rows: Search Export:

language_id	avg_length
1	115.2720

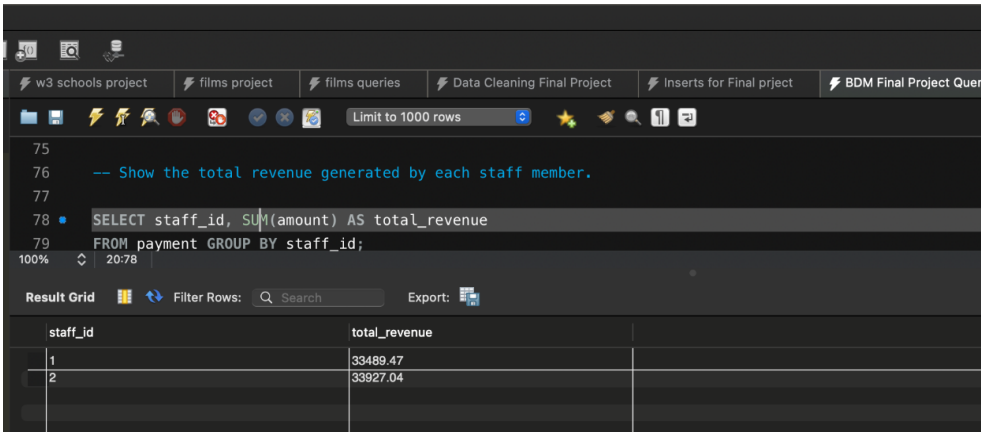
16. Find the customer who has rented the most films.

```
SELECT customer_id, COUNT(*) AS rental_count
FROM rental GROUP BY customer_id ORDER BY rental_count DESC LIMIT 1;
```



17. Show the total revenue generated by each staff member.

```
SELECT staff_id, SUM(amount) AS total_revenue
FROM payment GROUP BY staff_id;
```



18. Retrieve the customer who has spent the least on rentals.

```
SELECT customer_id, SUM(amount) AS total_spent
FROM payment GROUP BY customer_id ORDER BY total_spent
LIMIT 1;
```

```
81    -- Retrieve the customer who has spent the least on rentals.
82
83    SELECT customer_id, SUM(amount) AS total_spent
84    FROM payment GROUP BY customer_id ORDER BY total_spent
85    LIMIT 1;
```

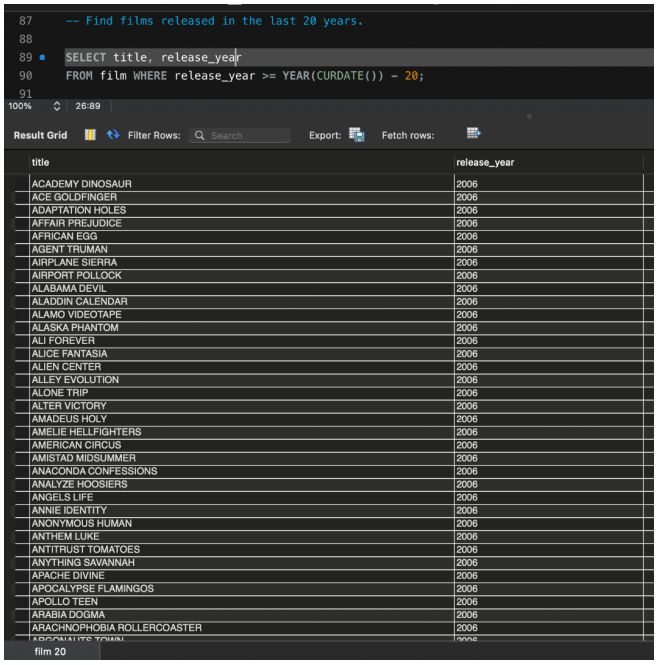
100% 20:83

Result Grid | Filter Rows: Search | Export: | Fetch rows:

customer_id	total_spent
248	50.85

19. Find films released in the last 20 years

```
SELECT title, release_year
FROM film WHERE release_year >= YEAR(CURDATE()) - 20;
```



20. Show the average rental duration for each film category

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

```
93
94 •  SELECT category.name, AVG(film.length) AS avg_duration
95     FROM film JOIN film_category ON film.film_id = film_category.film_id
96     JOIN category ON film_category.category_id = category.category_id
97     GROUP BY category.name;
```

100% 17:95

Result Grid | Filter Rows: Search | Export:

name	avg_duration
Action	111.6094
Animation	111.0152
Children	109.8000
Classics	111.6667
Comedy	115.8276
Documentary	108.7500
Drama	120.8387
Family	114.7826
Foreign	121.6986
Games	127.8361
Horror	112.4821
Music	113.6471
New	111.1270
Sci-Fi	108.1967
Sports	128.2027
Travel	113.3158

21. Calculate the total number of films rented each month

SELECT MONTH(rental_date) AS rental_month, COUNT(*) AS rental_count
FROM rental GROUP BY rental_month;

```
99   -- Calculate the total number of films rented each month.
100
101 •  SELECT MONTH(rental_date) AS rental_month, COUNT(*) AS rental_count
102    FROM rental GROUP BY rental_month;
103
```

100% 2:101

Result Grid | Filter Rows: Search | Export:

rental_month	rental_count
5	1156
6	2311
7	6709
8	5686
2	182

22. List films that were rented on a Sunday.

SELECT title, rental_date
FROM film JOIN inventory ON film.film_id = inventory.film_id
JOIN rental ON inventory.inventory_id = rental.inventory_id
WHERE DAYOFWEEK(rental_date) = 1;

```
105
106 •  SELECT title, rental_date
107    FROM film JOIN inventory ON film.film_id = inventory.film_id
108    JOIN rental ON inventory.inventory_id = rental.inventory_id
109    WHERE DAYOFWEEK(rental_date) = 1;
```

100% 22:106

Result Grid | Filter Rows: Search | Export: | Fetch rows:

title	rental_date
ACADEMY DINOSAUR	2005-08-21 21:27:43
ACADEMY DINOSAUR	2005-07-31 21:36:07
ACADEMY DINOSAUR	2005-08-21 18:32:42
ACADEMY DINOSAUR	2005-08-21 00:30:32
ACADEMY DINOSAUR	2005-07-10 13:07:31
ACADEMY DINOSAUR	2005-07-31 22:08:29
AFFAIR PREJUDICE	2005-07-10 08:51:39
AFFAIR PREJUDICE	2005-07-31 13:05:29
AFRICAN EGG	2005-07-31 12:50:24
AGENT TRUMAN	2005-07-10 15:11:54
AGENT TRUMAN	2005-06-19 04:19:04
AGENT TRUMAN	2005-08-21 16:03:01
AIRPLANE SIERRA	2005-08-21 04:34:11
AIRPLANE SIERRA	2005-07-31 09:20:50
AIRPORT POLLOCK	2005-08-21 13:43:59
AIRPORT POLLOCK	2005-05-29 15:08:41
AIRPORT POLLOCK	2005-07-31 18:31:51
AIRPORT POLLOCK	2005-06-19 17:54:48
ALABAMA DEVIL	2005-07-10 12:10:11
ALABAMA DEVIL	2005-07-31 18:47:20
ALADDIN CALENDAR	2005-08-21 20:49:21
ALADDIN CALENDAR	2005-08-21 11:06:33
ALADDIN CALENDAR	2005-06-19 03:15:05
ALAMO VIDEOTAPE	2005-06-19 00:11:26
ALAMO VIDEOTAPE	2005-07-31 21:20:59
ALAMO VIDEOTAPE	2005-07-31 19:04:35
ALAMO VIDEOTAPE	2005-07-10 09:32:22
ALASKA PHANTOM	2005-07-10 04:33:36
ALASKA PHANTOM	2005-05-29 21:58:43
ALASKA PHANTOM	2005-06-19 15:01:23
ALASKA PHANTOM	2005-07-10 00:27:21
ALASKA PHANTOM	2005-07-31 06:42:09
ALASKA PHANTOM	2005-06-19 19:34:45
ALASKA PHANTOM	2005-06-19 10:47:42
ALI FOREVER	2005-07-31 18:11:17
ALIEN CENTER	2005-06-19 13:20:28

23. Retrieve customers who have rented films from multiple categories.

SELECT customer_id, COUNT(DISTINCT category_id) AS distinct_categories
FROM rental JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film_category ON inventory.film_id = film_category.film_id
GROUP BY customer_id HAVING distinct_categories > 1;

```
112
113 •  SELECT customer_id, COUNT(DISTINCT category_id) AS distinct_categories
114    FROM rental JOIN inventory ON rental.inventory_id = inventory.inventory_id
115    JOIN film_category ON inventory.film_id = film_category.film_id
116    GROUP BY customer_id HAVING distinct_categories > 1;
```

100% 37:115

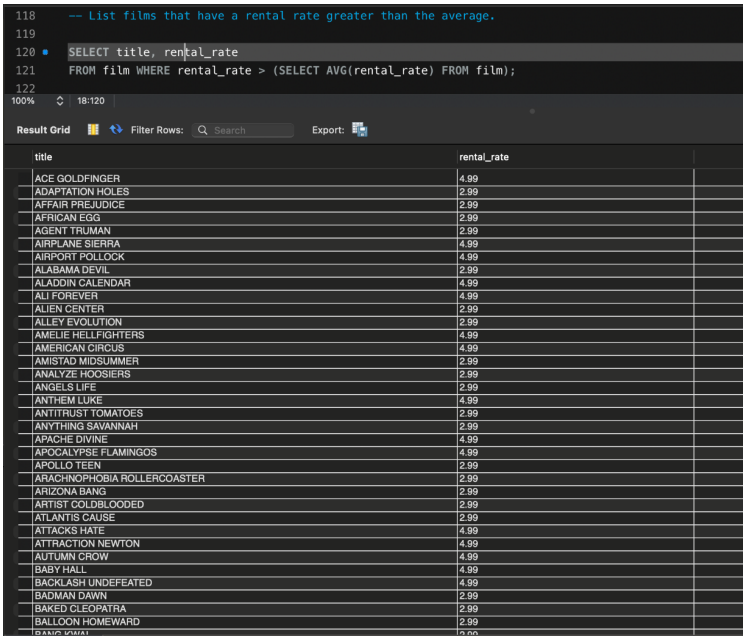
Result Grid | Filter Rows: Search | Export:

customer_id	distinct_categories
1	14
2	13
3	13
4	14
5	15
6	13
7	14
8	12
9	11
10	13
11	12
12	12
13	11
14	13
15	13
16	13
17	9
18	14
19	12
20	14
21	15
22	12
23	12
24	14
25	14
26	14
27	13
28	16
29	15
30	15
31	12
32	13
33	14
34	12
35	13

24. List films that have a rental rate greater than the average.

SELECT title, rental_rate

FROM film WHERE rental_rate > (SELECT AVG(rental_rate) FROM film);



```
118    -- List films that have a rental rate greater than the average.
119
120    SELECT title, rental_rate
121    FROM film WHERE rental_rate > (SELECT AVG(rental_rate) FROM film);
122
```

100% 18:120

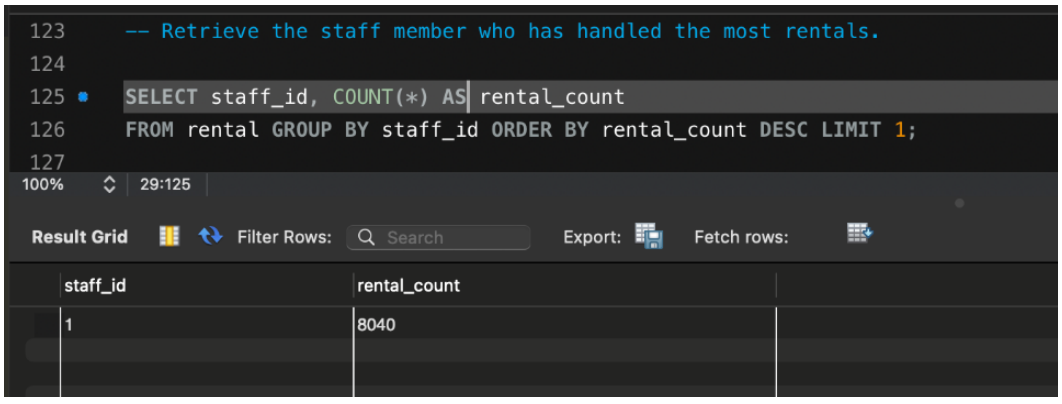
Result Grid Filter Rows: Search Export:

title	rental_rate
ACE GOLDFINGER	4.99
ADAPTATION HOLES	2.99
AFFAIR PREJUDICE	2.99
AFRICAN EGG	2.99
AGENT TRUMAN	2.99
AIRPLANE SIERRA	4.99
AIRPORT POLLOCK	4.99
ALABAMA DEVIL	2.99
ALADDIN CALENDAR	4.99
ALI FOREVER	4.99
ALIEN CENTER	2.99
ALLEY EVOLUTION	2.99
AMELIE HELLFIGHTERS	4.99
AMERICAN CIRCUS	4.99
AMISTAD MIDSUMMER	2.99
ANALYZE HOOSIERS	2.99
ANGELS LIFE	2.99
ANTHEM LUKE	4.99
ANTITRUST TOMATOES	2.99
ANYTHING SAVANNAH	2.99
APACHE DIVINE	4.99
APOCALYPSE FLAMINGOS	4.99
APOLLO TEEN	2.99
ARACHNOPHOBIA ROLLERCOASTER	2.99
ARIZONA BANG	2.99
ARTIST COLDBLOODED	2.99
ATLANTIS CAUSE	2.99
ATTACKS HATE	4.99
ATTRACTION NEWTON	4.99
AUTUMN CROW	4.99
BABY HALL	4.99
BACKLASH UNDEFEATED	4.99
BADMAN DAWN	2.99
BAKED CLEOPATRA	2.99
BALLOON HOMEWARD	2.99

25. Retrieve the staff member who has handled the most rentals

SELECT staff_id, COUNT(*) AS rental_count

FROM rental GROUP BY staff_id ORDER BY rental_count DESC LIMIT 1;



```
123    -- Retrieve the staff member who has handled the most rentals.
124
125    SELECT staff_id, COUNT(*) AS rental_count
126    FROM rental GROUP BY staff_id ORDER BY rental_count DESC LIMIT 1;
127
```

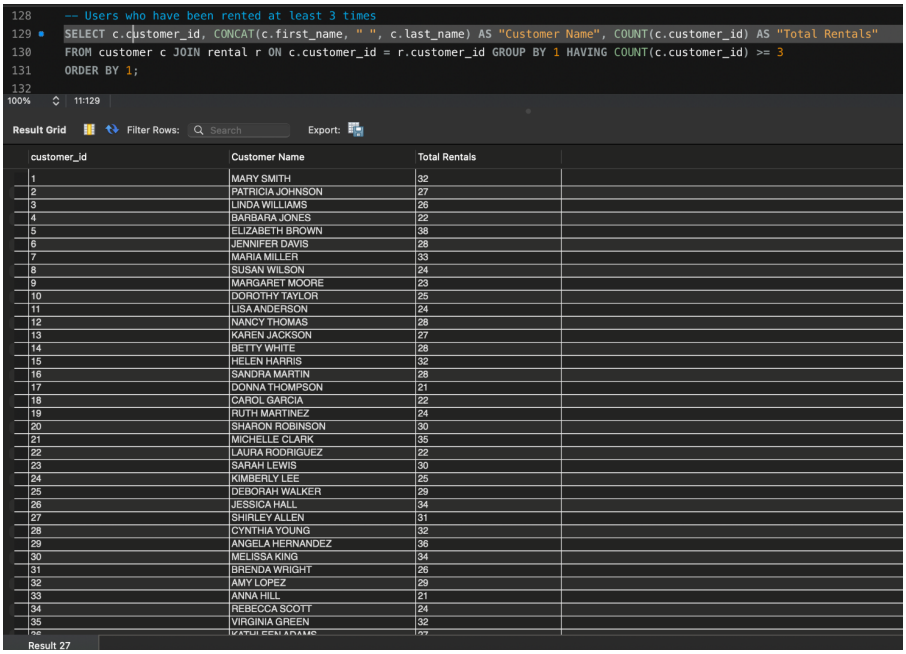
100% 29:125

Result Grid Filter Rows: Search Export: Fetch rows:

staff_id	rental_count
1	8040

26. Users who have been rented at least 3 times

```
SELECT c.customer_id, CONCAT(c.first_name, " ", c.last_name) AS "Customer Name",
COUNT(c.customer_id) AS "Total Rentals"
FROM customer c JOIN rental r ON c.customer_id = r.customer_id GROUP BY 1
HAVING COUNT(c.customer_id) >= 3
ORDER BY 1;
```



```
128  -- Users who have been rented at least 3 times
129  SELECT c.customer_id, CONCAT(c.first_name, " ", c.last_name) AS "Customer Name", COUNT(c.customer_id) AS "Total Rentals"
130  FROM customer c JOIN rental r ON c.customer_id = r.customer_id GROUP BY 1 HAVING COUNT(c.customer_id) >= 3
131  ORDER BY 1;
132
```

100% 11:129

Result Grid Filter Rows: Search Export:

customer_id	Customer Name	Total Rentals
1	MARY SMITH	32
2	PATRICIA JOHNSON	27
3	LINDA WILLIAMS	26
4	BARBARA JONES	22
5	ELIZABETH BROWN	38
6	JENNIFER DAVIS	28
7	MARIA MILLER	33
8	SUSAN WILSON	24
9	MARGARET MOORE	23
10	DOROTHY TAYLOR	25
11	LISA ANDERSON	24
12	NANCY THOMAS	28
13	KAREN JACKSON	27
14	BETTY WHITE	26
15	HELEN HARRIS	32
16	SANDRA MARTIN	28
17	DONNA THOMPSON	21
18	CAROL GARCIA	22
19	RUTH MARTINEZ	24
20	SHARON ROBINSON	30
21	MICHELLE CLARK	35
22	LAURA RODRIGUEZ	22
23	SARAH LEWIS	30
24	KIMBERLY LEE	25
25	DEBORAH WALKER	29
26	JESSICA HALL	34
27	SHIRLEY ALLEN	31
28	CYNTHIA YOUNG	32
29	ANGELA HERNANDEZ	36
30	MELISSA KING	34
31	BRENDA WRIGHT	26
32	AMY LOPEZ	29
33	ANNA HILL	21
34	REBECCA SCOTT	24
35	VIRGINIA GREEN	32

Result 27

27. Number of Rentals in Comedy , Sports and Family

```
SELECT c.name, COUNT(c.name) AS "Number of Rentals"
FROM film f JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON r.inventory_id = i.inventory_id
WHERE c.name IN ("Comedy", "Sports", "Family")
GROUP BY 1;
```

```
133    -- Number of Rentals in Comedy , Sports and Family
134    SELECT c.name, COUNT(c.name) AS "Number of Rentals"
135    FROM film f JOIN film_category fc ON fc.film_id = f.film_id
136    JOIN category c ON c.category_id = fc.category_id
137    JOIN inventory i ON f.film_id = i.film_id
138    JOIN rental r ON r.inventory_id = i.inventory_id
139    WHERE c.name IN ("Comedy", "Sports", "Family")
140    GROUP BY 1;
141
```

100% 37:136

Result Grid Filter Rows: Search Export:

name	Number of Rentals
Comedy	941
Family	1096
Sports	1179

28. How many distinct Renters per month

```
SELECT LEFT(rental_date,7) AS "Month",
        COUNT(DISTINCT(rental_id)) AS "Total Rentals",
        COUNT(DISTINCT(customer_id)) AS "Number Of Unique Renter",
   COUNT(DISTINCT(rental_id))/COUNT(DISTINCT(customer_id)) AS "Average Rent Per Renter"
FROM rental GROUP BY 1;
```

```
141
142    -- How many distint Renters per month
143
144    SELECT LEFT(rental_date,7) AS "Month",
145        COUNT(DISTINCT(rental_id)) AS "Total Rentals",
146        COUNT(DISTINCT(customer_id)) AS "Number Of Unique Renter",
147        COUNT(DISTINCT(rental_id))/COUNT(DISTINCT(customer_id)) AS "Average Rent Per Renter"
148    FROM rental GROUP BY 1;
149
```

100% 3:145

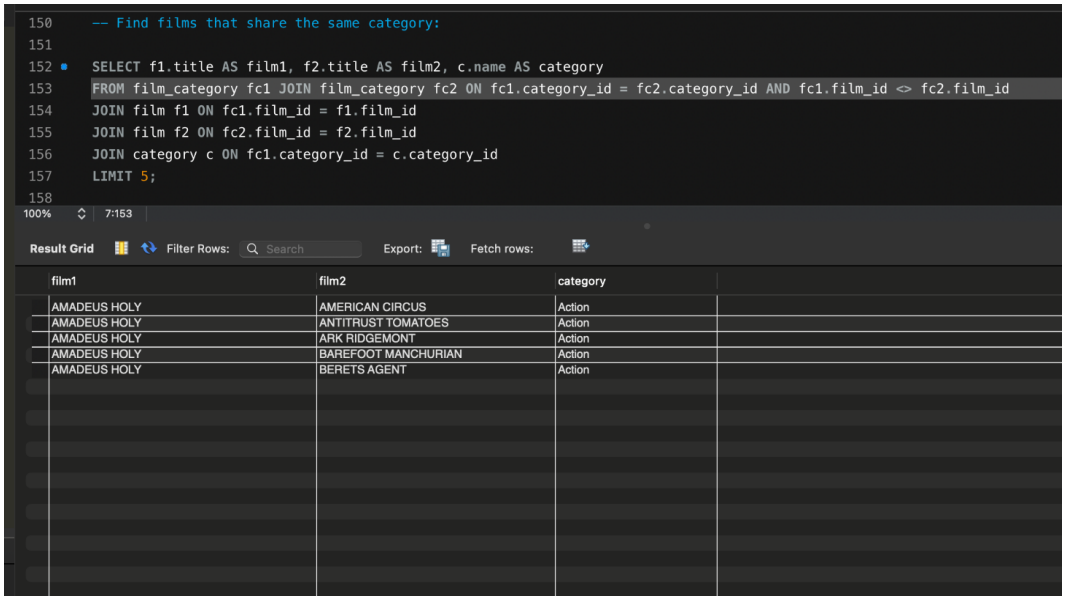
Result Grid Filter Rows: Search Export:

Export recordset to an external file

Month	Total Rentals	Numbe	Renter
2005-05	1156	520	2.2231
2005-06	2311	590	3.9169
2005-07	6709	599	11.2003
2005-08	5686	599	9.4925
2006-02	182	158	1.1519

29. Find films that share the same category.

```
SELECT f1.title AS film1, f2.title AS film2, c.name AS category
FROM film_category fc1 JOIN film_category fc2 ON fc1.category_id = fc2.category_id
AND fc1.film_id <> fc2.film_id
JOIN film f1 ON fc1.film_id = f1.film_id
JOIN film f2 ON fc2.film_id = f2.film_id
JOIN category c ON fc1.category_id = c.category_id
LIMIT 5;
```



—-- Advanced Topics

30. Create a trigger to update the last_update timestamp when a row in the film table is updated.

```
CREATE TRIGGER film_last_update_trigger
BEFORE UPDATE ON film
FOR EACH ROW
SET NEW.last_update = CURRENT_TIMESTAMP;
```

31. Create a trigger to automatically insert a new record into a log table when a film is rented

```
CREATE TRIGGER rental_log_trigger
AFTER INSERT ON rental
FOR EACH ROW
INSERT INTO rental_log (rental_id, rental_date, customer_id)
VALUES (NEW.rental_id, NEW.rental_date, NEW.customer_id);
```

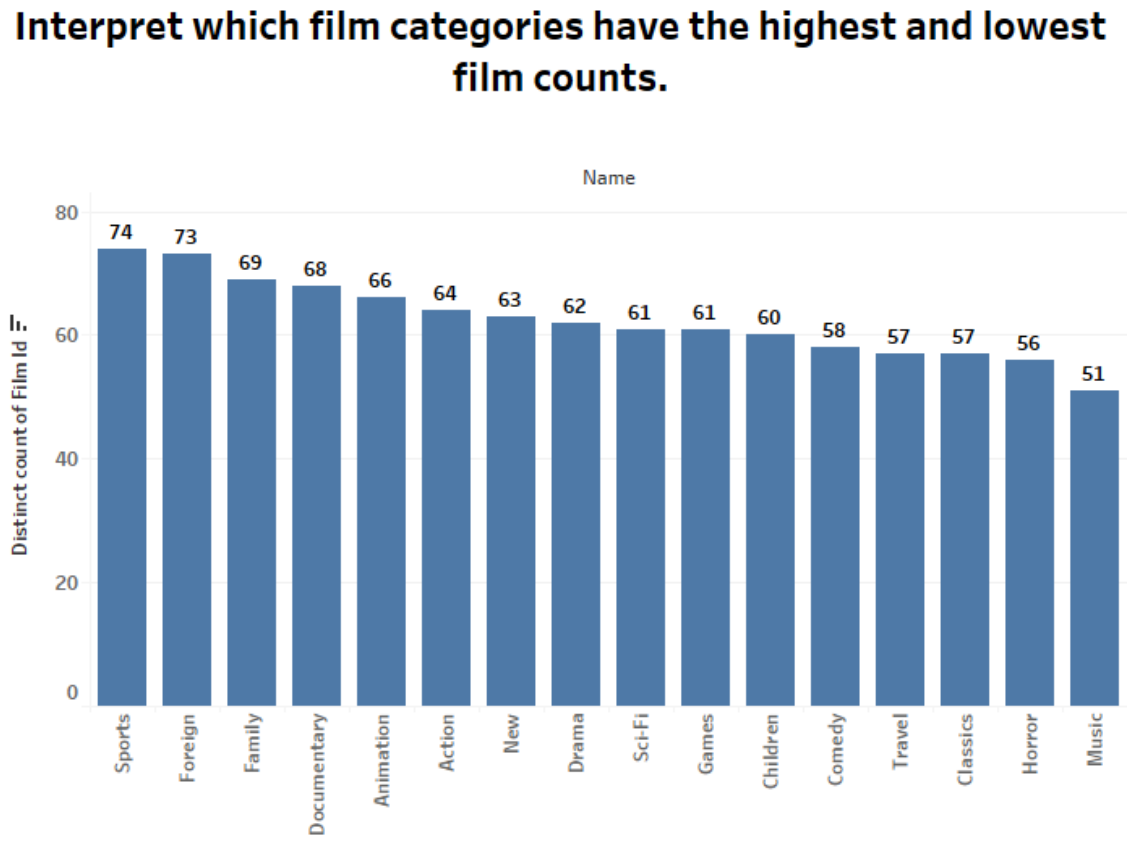
32. Adding a unique constraint on the email column in the customer table.

```
ALTER TABLE customer
ADD CONSTRAINT unique_email UNIQUE (email);
```

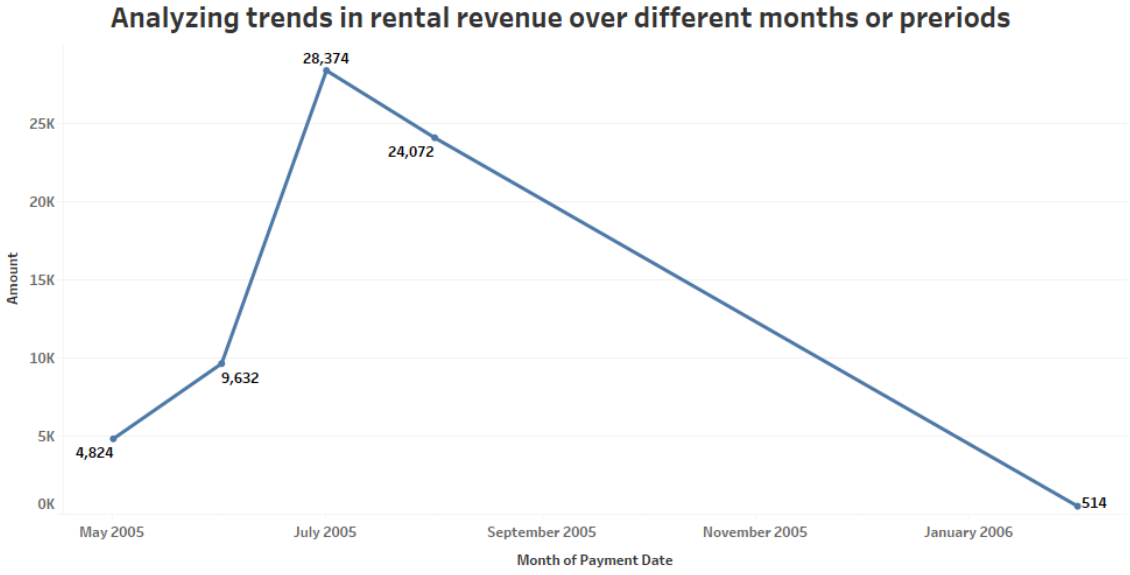
33. Adding a NOT NULL constraint on the last_name column in the customer table:

```
ALTER TABLE customer
MODIFY COLUMN last_name VARCHAR(45) NOT NULL;
```

Tableau Visualizations

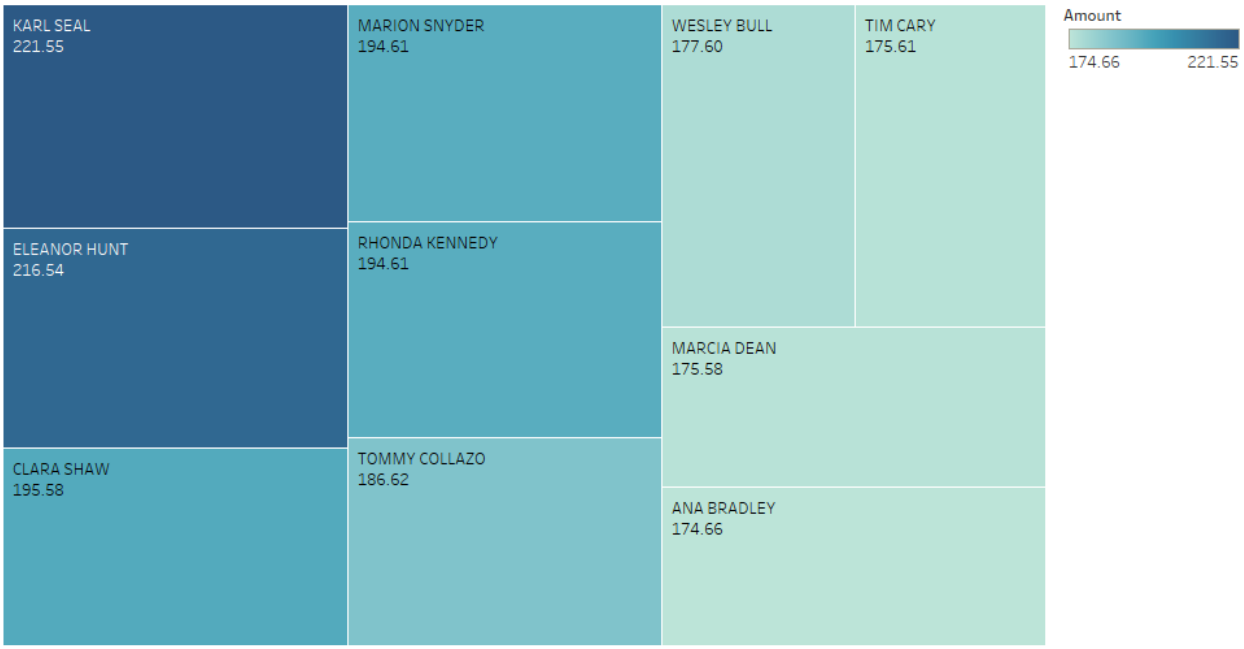


Distinct count of Film Id for each Name. The marks are labeled by distinct count of Film Id.



The trend of sum of Amount for Payment Date Month. The marks are labeled by sum of Amount.

The Tree map visualizes top 10 customers by total spending.
Created new field "Full name" by concatenating the columns 'last_name' and 'first_name'



Full name and sum of Amount. Color shows sum of Amount. Size shows sum of Amount. The marks are labeled by Full name and sum of Amount. The data is filtered on Customer Id, which has multiple members selected.

Accountability Contract

Work Breakdown	Group Member	Group Member	Group Member
Search and create datasets	Sairam	Deviprasad	Kalyan Venkat
ER Diagram	Sairam		
Create SQL code and import datasets to database	Kalyan Venkat		
Dataset cleaning with SQL Code	Deviprasad		
Design your own SQL question and write SQL queries	Sairam	Deviprasad	Kalyan Venkat

Presentation	Deviprasad		
Visualizations	Sairam		
PDF Report	Kalyan Venkat		

Conclusion

In conclusion, our project marks a significant achievement in harnessing the potential of raw datasets. The ER diagram and SQL code we developed lay the foundation for ongoing data-driven decision-making. The insights derived from the data hold the promise of positively impacting managerial strategies.