# SQL Project: Investigating a Relational Database - Sakila DVD Rental

| | |
|---|---|
| Technical Skills | Excel SQL |
| Difficulty | Intermediate |
| Last Updated | @March 13, 2024 |
| Project Description | We investigate a movie rental database to answer key questions about customer preferences and purchases. |
| Statistical Skills | |

## Overview & Database

We will use SQL to explore a database related to movie rentals. As part of the project, we will run SQL queries and build visualizations to showcase the output of our queries.

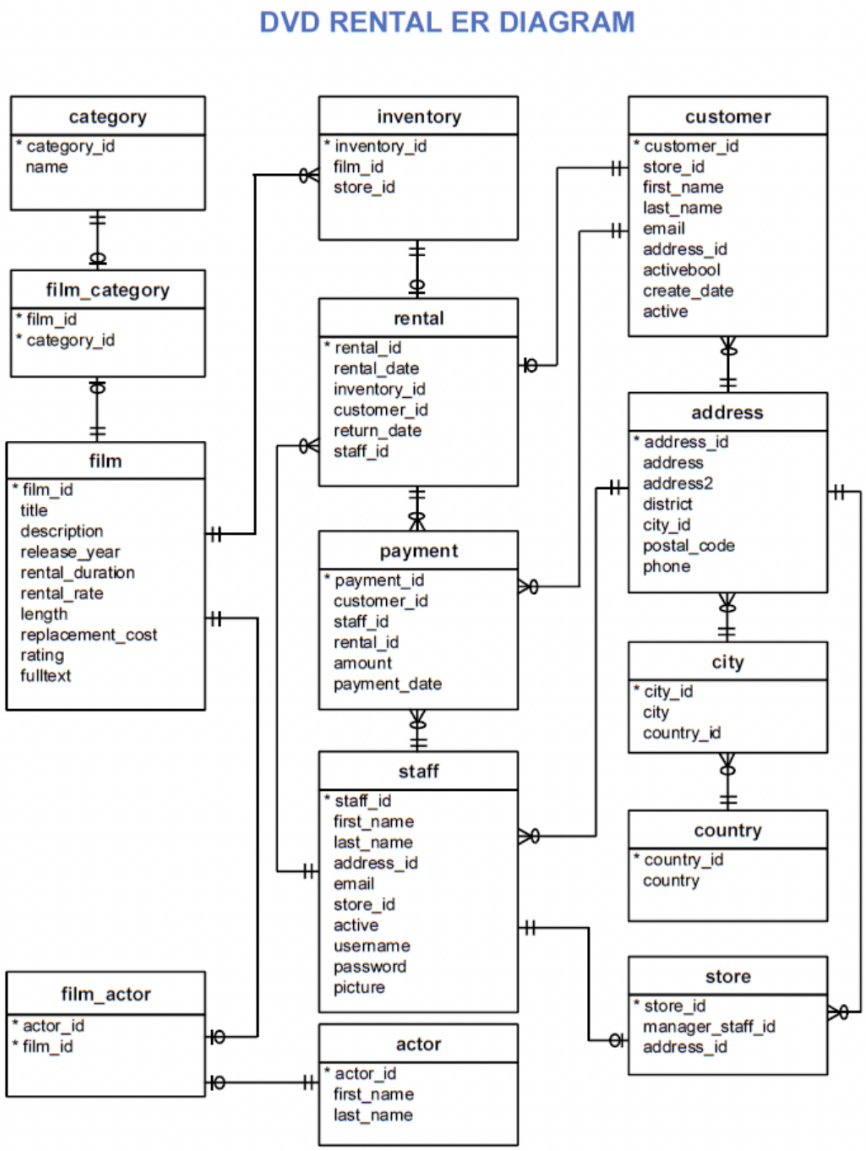
**Database:**

dvdrental.tar

## Introduction

In this project, we will query the **Sakila DVD Rental database**. The Sakila Database holds information about a company that rents movie DVDs. We will be querying the database to gain an understanding of the customer base, such as what the patterns in movie watching are across different customer groups,

how they compare on payment earnings, and how the stores compare in their performance. Here is the database schema:

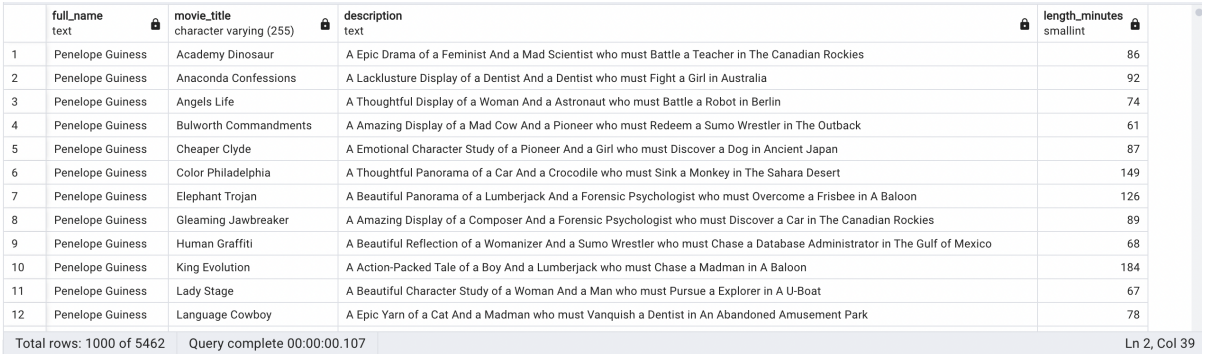



# ▼ Understanding the Database

**1) Let's start with creating a table that provides the following details: actor's first and last name combined as full_name, film title, film description and length of the movie.**

```
SELECT CONCAT(first_name,' ',last_name) AS full_name, f.title
f.description, f.length
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id;
```

**Output:**



| | full_name text | movie_title character varying (255) | description text | length_minutes smallint |
|---|---|---|---|---|
| 1 | Penelope Guiness | Academy Dinosaur | A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies | 86 |
| 2 | Penelope Guiness | Anaconda Confessions | A Lacklusture Display of a Dentist And a Dentist who must Fight a Girl in Australia | 92 |
| 3 | Penelope Guiness | Angels Life | A Thoughtful Display of a Woman And a Astronaut who must Battle a Robot in Berlin | 74 |
| 4 | Penelope Guiness | Bulworth Commandments | A Amazing Display of a Mad Cow And a Pioneer who must Redeem a Sumo Wrestler in The Outback | 61 |
| 5 | Penelope Guiness | Cheaper Clyde | A Emotional Character Study of a Pioneer And a Girl who must Discover a Dog in Ancient Japan | 87 |
| 6 | Penelope Guiness | Color Philadelphia | A Thoughtful Panorama of a Car And a Crocodile who must Sink a Monkey in The Sahara Desert | 149 |
| 7 | Penelope Guiness | Elephant Trojan | A Beautiful Panorama of a Lumberjack And a Forensic Psychologist who must Overcome a Frisbee in A Baloon | 126 |
| 8 | Penelope Guiness | Gleaming Jawbreaker | A Amazing Display of a Composer And a Forensic Psychologist who must Discover a Car in The Canadian Rockies | 89 |
| 9 | Penelope Guiness | Human Graffiti | A Beautiful Reflection of a Womanizer And a Sumo Wrestler who must Chase a Database Administrator in The Gulf of Mexico | 68 |
| 10 | Penelope Guiness | King Evolution | A Action-Packed Tale of a Boy And a Lumberjack who must Chase a Madman in A Baloon | 184 |
| 11 | Penelope Guiness | Lady Stage | A Beautiful Character Study of a Woman And a Man who must Pursue a Explorer in A U-Boat | 67 |
| 12 | Penelope Guiness | Language Cowboy | A Epic Yarn of a Cat And a Madman who must Vanquish a Dentist in An Abandoned Amusement Park | 78 |

Total rows: 1000 of 5462 | Query complete 00:00:00.107 | Ln 2, Col 39

**2) Write a query that creates a list of actors and movies where the movie length was more than 60 minutes.**

```
SELECT CONCAT(first_name,' ',last_name) AS full_name, f.title
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
WHERE f.length > 60
ORDER BY length;
```

**Output:**

| | full_name<br>text | title<br>character varying (255) | length_minutes<br>smallint |
|---|---|---|---|
| 1 | Audrey Bailey | Drifter Commandments | 61 |
| 2 | Vivien Bergen | Polish Brooklyn | 61 |
| 3 | Rip Crawford | Polish Brooklyn | 61 |
| 4 | Vivien Bergen | Drifter Commandments | 61 |
| 5 | Fay Winslet | Polish Brooklyn | 61 |
| 6 | Ian Tandy | Polish Brooklyn | 61 |
| 7 | Jessica Bailey | Chinatown Gladiator | 61 |
| 8 | Gene Mckellen | Reservoir Adaptation | 61 |
| 9 | Kenneth Paltrow | Trouble Date | 61 |
| 10 | Sean Williams | Chinatown Gladiator | 61 |
| 11 | Gary Penn | Birds Perdition | 61 |
| 12 | Sandra Peck | Birds Perdition | 61 |

Total rows: 1000 of 4900 | Query complete 00:00:00.098

**3) Write a query that captures the actor id, full name of the actor, and counts the number of movies each actor has made.**

```
SELECT CONCAT(first_name,' ',last_name) AS full_name, fa.acto
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
GROUP BY 1,2
ORDER BY 3 DESC;
```

**Output:**

| | full_name text | actor_id smallint | count bigint |
|---|---|---|---|
| 1 | Gina Degeneres | 107 | 42 |
| 2 | Walter Torn | 102 | 41 |
| 3 | Mary Keitel | 198 | 40 |
| 4 | Matthew Carrey | 181 | 39 |
| 5 | Sandra Kilmer | 23 | 37 |
| 6 | Scarlett Damon | 81 | 36 |
| 7 | Angela Witherspoon | 144 | 35 |
| 8 | Vivien Basinger | 158 | 35 |
| 9 | Henry Berry | 60 | 35 |
| 10 | Uma Wood | 13 | 35 |
| 11 | Val Bolger | 37 | 35 |
| 12 | Groucho Dunst | 106 | 35 |

Total rows: 200 of 200 Query complete 00:00:00.109

**4) Write a query that displays a table with 4 columns: actor's full name, film title, length of movie, and a column name "filmlen_groups" that classifies movies based on their length. Filmlen_groups should include 4 categories: 1 hour or less, Between 1-2 hours, Between 2-3 hours, More than 3 hours.**

```
SELECT CONCAT(first_name,' ',last_name) AS full_name, f.title
f.length,
CASE WHEN f.length <= 60 THEN '1 hour or less'
WHEN f.length > 60 AND f.length <= 120 THEN 'Between 1-2 hour
WHEN f.length > 120 AND f.length <= 180 THEN 'Between 2-3 hou
ELSE 'More than 3 hours' END AS filmlen_groups
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
ORDER BY 3;
```

**Output:**

| | full_name text | title character varying (255) | length smallint | filmlen_groups text |
|---|---|---|---|---|
| 1 | Scarlett Damon | Labyrinth League | 46 | 1 hour or less |
| 2 | Tom Miranda | Labyrinth League | 46 | 1 hour or less |
| 3 | Angela Witherspoon | Kwai Homeward | 46 | 1 hour or less |
| 4 | Michael Bolger | Ridgemont Submarine | 46 | 1 hour or less |
| 5 | Julianne Dench | Ridgemont Submarine | 46 | 1 hour or less |
| 6 | Anne Cronyn | Iron Moon | 46 | 1 hour or less |
| 7 | Groucho Sinatra | Labyrinth League | 46 | 1 hour or less |
| 8 | Kenneth Paltrow | Alien Center | 46 | 1 hour or less |
| 9 | Frances Tomei | Iron Moon | 46 | 1 hour or less |
| 10 | Chris Depp | Iron Moon | 46 | 1 hour or less |
| 11 | Humphrey Willis | Iron Moon | 46 | 1 hour or less |
| 12 | Kirsten Akroyd | Labyrinth League | 46 | 1 hour or less |

Total rows: 1000 of 5462 | Query complete 00:00:00.130

**5) Revise the query you wrote above to create a count of movies in each of the 4 filmlen_groups: 1 hour or less, Between 1-2 hours, Between 2-3 hours, More than 3 hours. * Match the count of movies in each filmlen_group.**

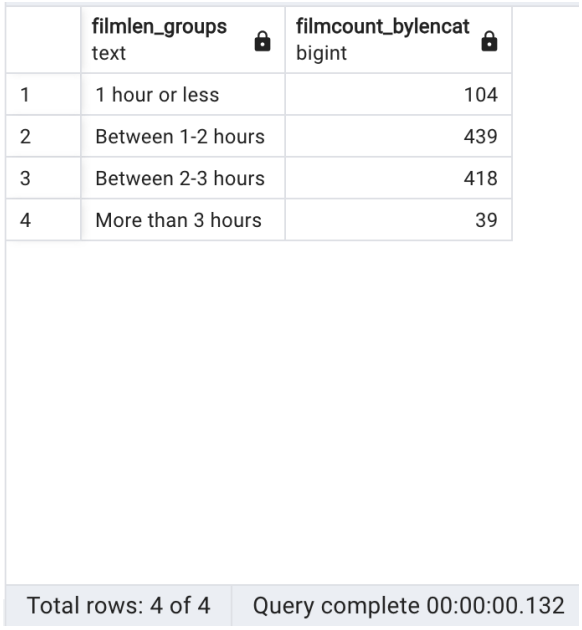
```
SELECT t1.filmlen_groups, COUNT(*) filmcount_bylencat
FROM
(SELECT DISTINCT f.title, f.length,
CASE WHEN f.length <= 60 THEN '1 hour or less'
WHEN f.length > 60 AND f.length <= 120 THEN 'Between 1-2 hour
WHEN f.length > 120 AND f.length <= 180 THEN 'Between 2-3 hou
ELSE 'More than 3 hours' END AS filmlen_groups
FROM film f
)t1
GROUP BY 1
ORDER BY 1;
```

or

```
SELECT    DISTINCT(filmlen_groups),
COUNT(title) OVER (PARTITION BY filmlen_groups) AS filmcount_
FROM
(SELECT title,length,
CASE WHEN length <= 60 THEN '1 hour or less'
```

```
WHEN length > 60 AND length <= 120 THEN 'Between 1-2 hours'
WHEN length > 120 AND length <= 180 THEN 'Between 2-3 hours'
ELSE 'More than 3 hours' END AS filmlen_groups
FROM film ) t1
ORDER BY  filmlen_groups;
```

**Output:**



| | filmlen_groups text | filmcount_bylencat bigint |
|---|---|---|
| 1 | 1 hour or less | 104 |
| 2 | Between 1-2 hours | 439 |
| 3 | Between 2-3 hours | 418 |
| 4 | More than 3 hours | 39 |

Total rows: 4 of 4 | Query complete 00:00:00.132

# ▼ Investigating the Database & Visualizations Part 1

**We want to understand more about the movies that families are watching. The following categories are considered family movies: Animation, Children, Classics, Comedy, Family and Music.**

*** Create a query that lists each movie, the film category it is classified in, and the number of times it has been rented out.***

```
SELECT DISTINCT title, name,
COUNT(rental_id) OVER (PARTITION BY title) AS rental_count
FROM
```

```
    (SELECT f.title title, c.name name, r.rental_id rental_id
    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 i.film_id = f.film_id
    JOIN rental r ON r.inventory_id = i.inventory_id
    WHERE c.name IN ('Animation','Children','Classics','Comed
ORDER BY 2,1;
```

OR

```
SELECT f.title title, c.name name, COUNT(r.rental_id) rental_
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 i.film_id = f.film_id
JOIN rental r ON r.inventory_id = i.inventory_id
GROUP BY 1,2
HAVING name IN ('Animation', 'Children', 'Classics', 'Comedy'
ORDER BY 2,1;
```

**Output:**

| | title<br>character varying (255) | name<br>character varying (25) | rental_count<br>bigint |
|---|---|---|---|
| 1 | Alter Victory | Animation | 22 |
| 2 | Anaconda Confessions | Animation | 21 |
| 3 | Bikini Borrowers | Animation | 17 |
| 4 | Blackout Private | Animation | 27 |
| 5 | Borrowers Bedazzled | Animation | 22 |
| 6 | Canyon Stock | Animation | 19 |
| 7 | Carol Texas | Animation | 18 |
| 8 | Champion Flatliners | Animation | 13 |
| 9 | Clash Freddy | Animation | 25 |
| 10 | Club Graffiti | Animation | 19 |
| 11 | Crossroads Casualties | Animation | 21 |
| 12 | Dares Pluto | Animation | 9 |

Total rows: 350 of 350 | Query complete 00:00:00.111

**How many movies were rented from each category?**

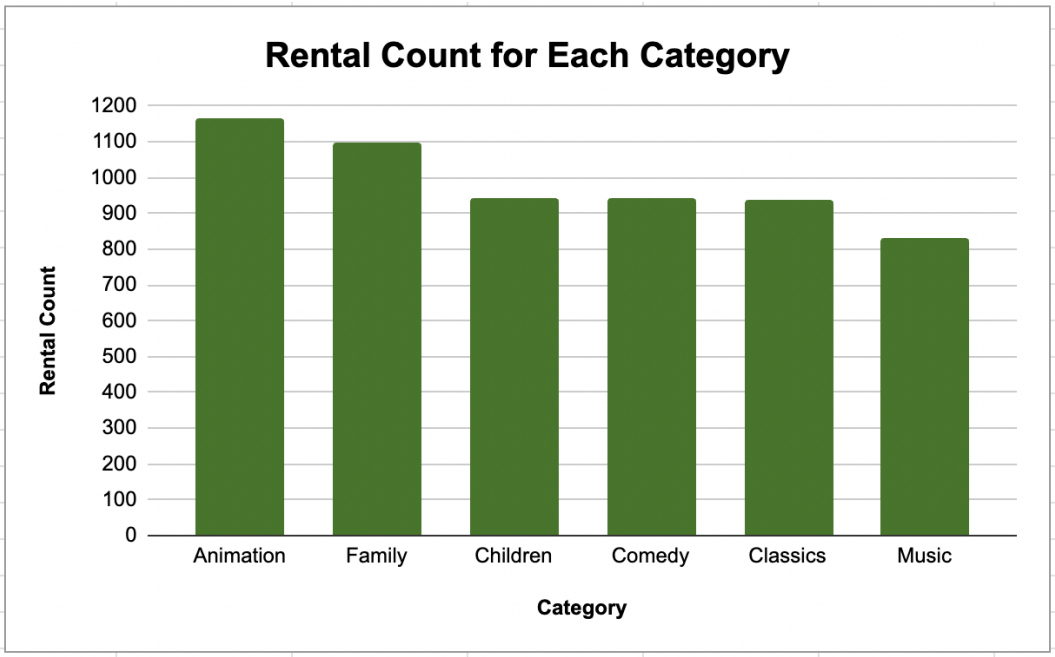
```
SELECT DISTINCT name,
SUM(rental_count) OVER (PARTITION BY name)
FROM
    (SELECT f.title title, c.name name, COUNT(r.rental_id) re
    FROM film f
    JOIN film_category fc ON fc.film_id = f.film_id
    JOIN category c ON c.category_id = fc.category_id
    JOIN inventory i ON i.film_id = f.film_id
    JOIN rental r ON r.inventory_id = i.inventory_id
    GROUP BY 1,2
    HAVING name IN ('Animation', 'Children', 'Classics', 'Com
    ORDER BY 2,1)
ORDER BY 2 DESC;
```

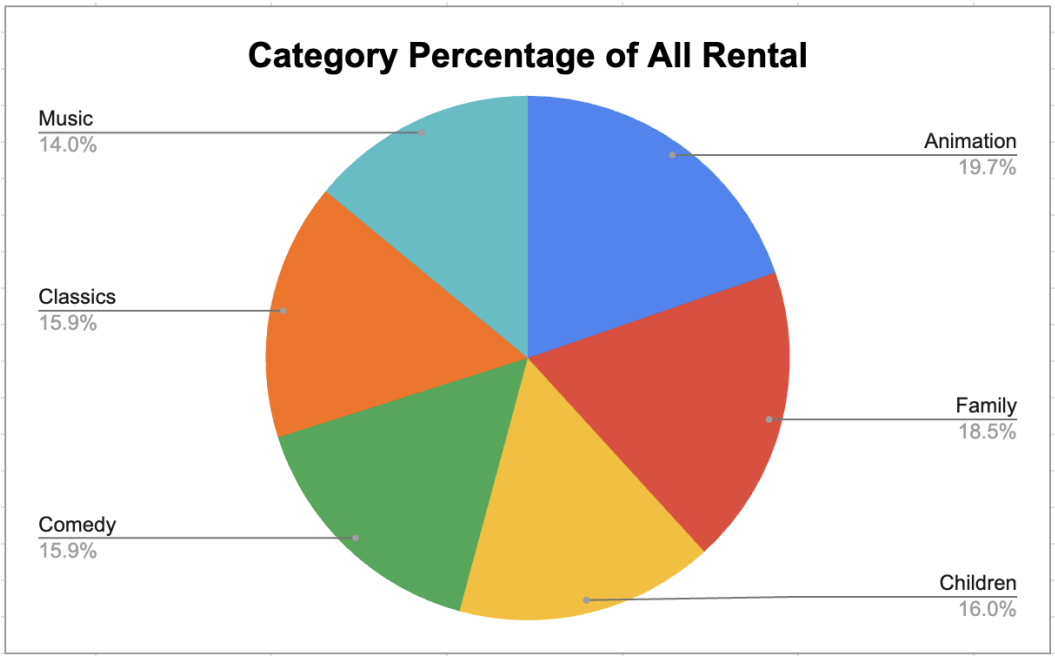
**Output:**

| | name<br>character varying (25) | sum<br>numeric |
|---|---|---|
| 1 | Animation | 1166 |
| 2 | Family | 1096 |
| 3 | Children | 945 |
| 4 | Comedy | 941 |
| 5 | Classics | 939 |
| 6 | Music | 830 |

Total rows: 6 of 6 | Query complete 00:00:00.068

## Visualization:






**Now we need to know how the length of rental duration of these family-friendly movies compares to the duration that all movies are rented for.**

**Provide a table with the movie titles and divide them into 4 levels (first_quarter, second_quarter, third_quarter, and final_quarter) based on the quartiles (25%, 50%, 75%) of the rental duration for movies across all categories?**

```
SELECT title, category,
CASE WHEN quartile = 1 THEN 'first_quartile'
WHEN quartile = 2 THEN 'second_quartile'
WHEN quartile = 3 THEN 'third_quartile' ELSE 'fourth_quartile
END AS rental_quartiles
FROM
    (SELECT title, c.name category, rental_duration,
    NTILE (4) OVER(ORDER BY rental_duration) AS quartile
    FROM film f
    JOIN film_category fc ON fc.film_id = f.film_id
```

```
    JOIN category c ON c.category_id = fc.category_id
    WHERE name IN ('Animation', 'Children', 'Classics', 'Come
```

**Output:**

| | title<br>character varying (255) | category<br>character varying (25) | quartiles<br>text |
|---|---|---|---|
| 1 | Sweethearts Suspects | Children | first_quartile |
| 2 | Go Purple | Music | first_quartile |
| 3 | Bilko Anonymous | Family | first_quartile |
| 4 | Wait Cider | Animation | first_quartile |
| 5 | Daughter Madigan | Children | first_quartile |
| 6 | Turn Star | Animation | first_quartile |
| 7 | Rush Goodfellas | Family | first_quartile |
| 8 | King Evolution | Family | first_quartile |
| 9 | Tracy Cider | Animation | first_quartile |
| 10 | Wisdom Worker | Comedy | first_quartile |
| 11 | Telegraph Voyage | Music | first_quartile |
| 12 | Miracle Virtual | Animation | first_quartile |

Total rows: 361 of 361 | Query complete 00:00:00.118

**Provide a table with the family-friendly film category, each of the quartiles, and the corresponding count of movies within each combination of film category for each corresponding rental duration category.**

```
SELECT t1.name, t1.quartile rental_duration_quartile, COUNT(*
FROM
    (SELECT title, rental_duration, c.name name,
    NTILE (4) OVER(ORDER BY rental_duration) AS quartile
    FROM film f
    JOIN film_category fc ON fc.film_id = f.film_id
    JOIN category c ON c.category_id = fc.category_id
    WHERE name IN ('Animation', 'Children', 'Classics', 'Come
    )t1
```

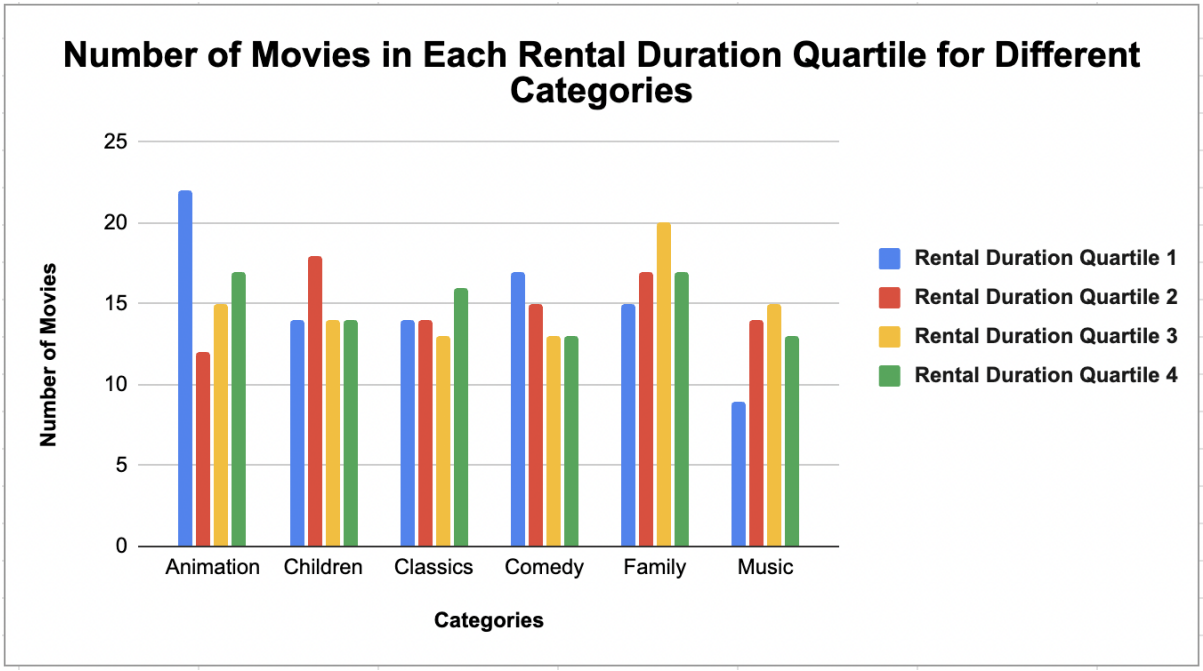
```
GROUP BY 1,2
ORDER BY 1,2;
```

**Output:**

| | name<br>character varying (25) | rental_duration_quartile<br>integer | count<br>bigint |
|---|---|---|---|
| 1 | Animation | 1 | 22 |
| 2 | Animation | 2 | 12 |
| 3 | Animation | 3 | 15 |
| 4 | Animation | 4 | 17 |
| 5 | Children | 1 | 14 |
| 6 | Children | 2 | 18 |
| 7 | Children | 3 | 14 |
| 8 | Children | 4 | 14 |
| 9 | Classics | 1 | 14 |
| 10 | Classics | 2 | 14 |
| 11 | Classics | 3 | 13 |
| 12 | Classics | 4 | 16 |

Total rows: 24 of 24 | Query complete 00:00:00.155

**Visualization:**



### Conclusions for Part 1:

- Analyzing the visualizations, we conclude that while animation movies were the most rented, the rental duration for them were the least. Music movies were the least rented but the rental duration was longer for them.

# ▼ Investigating the Database & Visualizations Part 2

**We want to find out how the two stores compare in their count of rental orders during every month for all the years we have data for.**

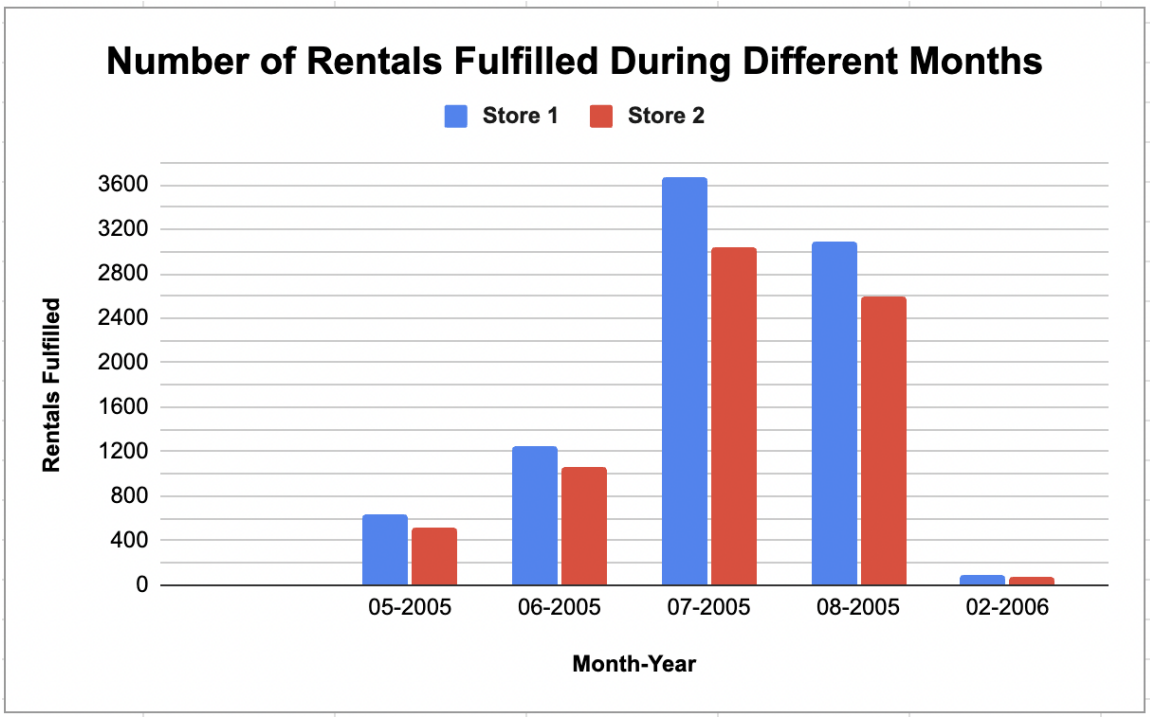
**Write a query that returns the store ID for the store, the year and month and the number of rental orders each store has fulfilled for that month.**

```
SELECT s.store_id, DATE_PART('year',r.rental_date) rental_yea
DATE_PART('month',r.rental_date) rental_month, COUNT(*) numbe
FROM store s
JOIN customer c ON c.store_id = s.store_id
JOIN rental r ON r.customer_id = c.customer_id
GROUP BY 1,2,3
ORDER BY 1,4 DESC;
```

**Output:**

| | store_id [PK] integer | rental_year double precision | rental_month double precision | number_of_rentals bigint |
|---|---|---|---|---|
| 1 | 1 | 2005 | 7 | 3677 |
| 2 | 1 | 2005 | 8 | 3091 |
| 3 | 1 | 2005 | 6 | 1243 |
| 4 | 1 | 2005 | 5 | 638 |
| 5 | 1 | 2006 | 2 | 98 |
| 6 | 2 | 2005 | 7 | 3032 |
| 7 | 2 | 2005 | 8 | 2595 |
| 8 | 2 | 2005 | 6 | 1068 |
| 9 | 2 | 2005 | 5 | 518 |
| 10 | 2 | 2006 | 2 | 84 |

**Visualization:**




**We would like to know who were our top 10 paying customers, how many**

**payments they made on a monthly basis during 2007, and what was the amount of the monthly payments.**

**Write a query to capture the customer name, month and year of payment, and total payment amount for each month by these top 10 paying customers.**

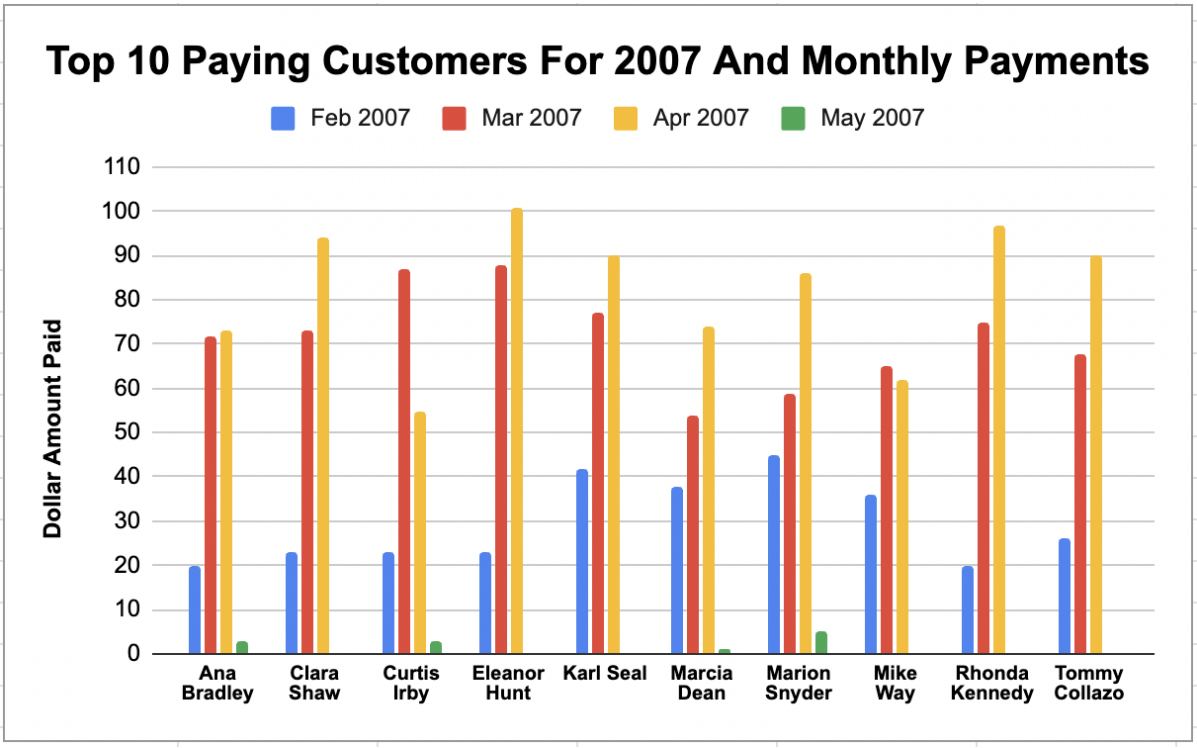
```
SELECT DATE_TRUNC('month',p.payment_date) date,
CONCAT(c.first_name,' ',c.last_name) name,
COUNT(p.amount) payment_count_per_month,
SUM(p.amount) total_payment_per_month
FROM customer c
JOIN payment p ON p.customer_id = c.customer_id
WHERE CONCAT(c.first_name,' ',c.last_name) IN

    (SELECT t1.name FROM
       (SELECT CONCAT(c.first_name,' ',c.last_name) name,
            SUM(p.amount) payment
            FROM customer c
            JOIN payment p ON p.customer_id = c.customer_id
            GROUP BY 1
            ORDER BY 2 DESC
            LIMIT 10)t1)

GROUP BY 1,2
HAVING DATE_TRUNC('month',p.payment_date) BETWEEN '2007-01-01
ORDER BY 2,1;
```

**Output:**

| | date<br>timestamp without time zone | name<br>text | payment_count_per_month<br>bigint | total_payment_per_month<br>numeric |
|---|---|---|---|---|
| 1 | 2007-02-01 00:00:00 | Ana Bradley | 4 | 19.96 |
| 2 | 2007-03-01 00:00:00 | Ana Bradley | 16 | 71.84 |
| 3 | 2007-04-01 00:00:00 | Ana Bradley | 12 | 72.88 |
| 4 | 2007-05-01 00:00:00 | Ana Bradley | 1 | 2.99 |
| 5 | 2007-02-01 00:00:00 | Clara Shaw | 6 | 22.94 |
| 6 | 2007-03-01 00:00:00 | Clara Shaw | 16 | 72.84 |
| 7 | 2007-04-01 00:00:00 | Clara Shaw | 18 | 93.82 |
| 8 | 2007-02-01 00:00:00 | Curtis Irby | 6 | 22.94 |
| 9 | 2007-03-01 00:00:00 | Curtis Irby | 17 | 86.83 |
| 10 | 2007-04-01 00:00:00 | Curtis Irby | 14 | 54.86 |
| 11 | 2007-05-01 00:00:00 | Curtis Irby | 1 | 2.99 |
| 12 | 2007-02-01 00:00:00 | Eleanor Hunt | 5 | 22.95 |

Total rows: 34 of 34 | Query complete 00:00:00.075

## Visualization:



**Finally, for each of these top 10 paying customers, we would like to find out the difference across their monthly payments during 2007.**

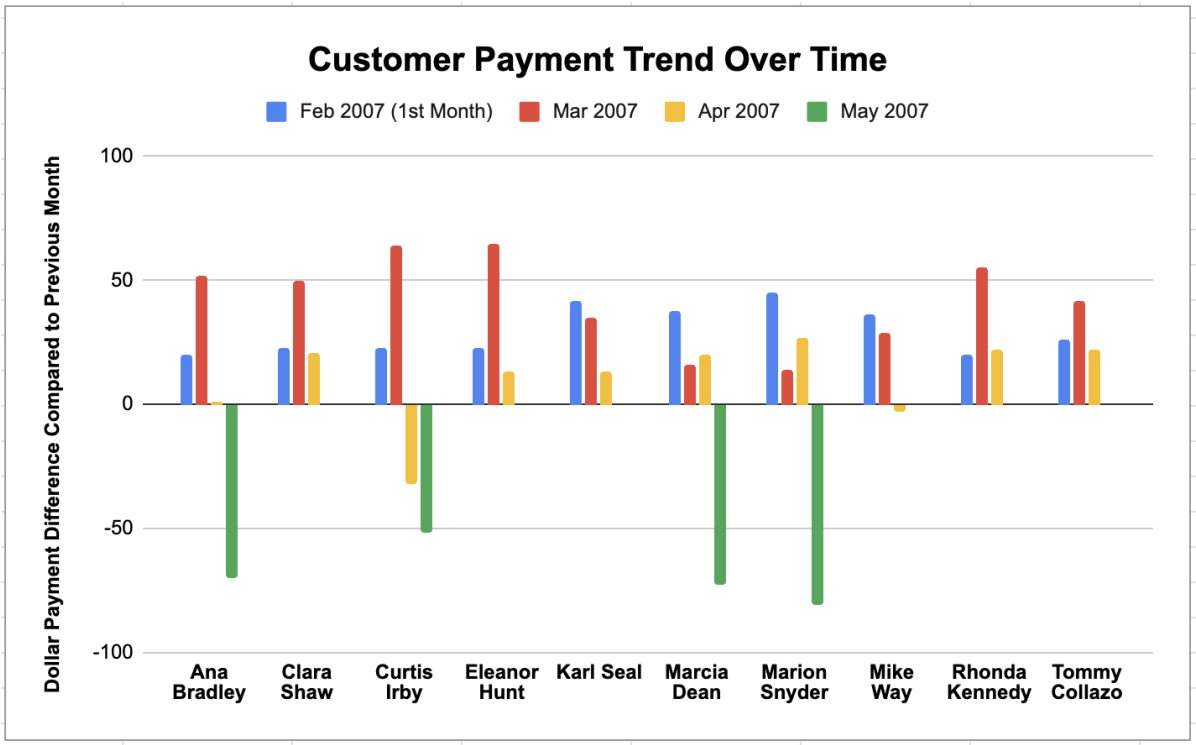
**Write a query to compare the payment amounts in each successive month. Repeat this for each of these 10 paying customers.**

```
WITH t2 AS
    (SELECT DATE_TRUNC('month',p.payment_date) date,
    CONCAT(c.first_name,' ',c.last_name) name,
    SUM(p.amount) total_payment_per_month,
    LAG(SUM(p.amount)) OVER(PARTITION BY CONCAT(c.first_name,
    SUM(p.amount) - LAG(SUM(p.amount)) OVER(PARTITION BY CONC

    FROM customer c
    JOIN payment p ON p.customer_id = c.customer_id
    WHERE CONCAT(c.first_name,' ',c.last_name) IN

        (SELECT t1.name FROM
           (SELECT CONCAT(c.first_name,' ',c.last_name) name,
                SUM(p.amount) payment
                FROM customer c
                JOIN payment p ON p.customer_id = c.customer_
                GROUP BY 1
                ORDER BY 2 DESC
                LIMIT 10)t1)

    GROUP BY 1,2
    HAVING DATE_TRUNC('month',p.payment_date) BETWEEN '2007-0
    ORDER BY 2,1)

SELECT date,name, total_payment_per_month,
COALESCE(previous_month,0) previous_month,
COALESCE(monthly_payment_difference,0) monthly_payment_differ
FROM t2
```

**Output:**

| | date<br>timestamp without time zone | name<br>text | total_payment_per_month<br>numeric | previous_month<br>numeric | monthly_payment_difference<br>numeric |
|---|---|---|---|---|---|
| 1 | 2007-02-01 00:00:00 | Ana Bradley | 19.96 | 0 | 0 |
| 2 | 2007-03-01 00:00:00 | Ana Bradley | 71.84 | 19.96 | 51.88 |
| 3 | 2007-04-01 00:00:00 | Ana Bradley | 72.88 | 71.84 | 1.04 |
| 4 | 2007-05-01 00:00:00 | Ana Bradley | 2.99 | 72.88 | -69.89 |
| 5 | 2007-02-01 00:00:00 | Clara Shaw | 22.94 | 0 | 0 |
| 6 | 2007-03-01 00:00:00 | Clara Shaw | 72.84 | 22.94 | 49.90 |
| 7 | 2007-04-01 00:00:00 | Clara Shaw | 93.82 | 72.84 | 20.98 |
| 8 | 2007-02-01 00:00:00 | Curtis Irby | 22.94 | 0 | 0 |
| 9 | 2007-03-01 00:00:00 | Curtis Irby | 86.83 | 22.94 | 63.89 |
| 10 | 2007-04-01 00:00:00 | Curtis Irby | 54.86 | 86.83 | -31.97 |
| 11 | 2007-05-01 00:00:00 | Curtis Irby | 2.99 | 54.86 | -51.87 |
| 12 | 2007-02-01 00:00:00 | Eleanor Hunt | 22.95 | 0 | 0 |

## Visualization:




**Also, it will be tremendously helpful if you can identify the customer name who paid the most difference in terms of payments.**

```
SELECT date,name, total_payment_per_month,
COALESCE(previous_month,0) previous_month,
COALESCE(monthly_payment_difference,0) monthly_payment_differ
FROM t2
ORDER BY 5 DESC
LIMIT 1;
```