

# **Data Analysis with DVD Rental Dataset**

by Rahma Amalia

using dvdrental database you've uploaded to your local postgresql db, please answer these questions

Exercise :

1. Identify the top 10 customers and their email so we can reward them
2. Identify the bottom 10 customers and their emails
3. What are the most profitable movie genres (ratings)?
4. How many rented movies were returned late, early, and on time?
5. What is the customer base in the countries where we have a presence?
6. Which country is the most profitable for the business?
7. What is the average rental rate per movie genre (rating)?

Output : .pdf file contains :

- written sql query
- screenshot of query result
- u can also visualize the result using ppt chart / python EDA library (optional)

# 1. Identify the top 10 customers and their email so we can reward them

SQL Query:

```
SELECT customer.first_name, customer.last_name,  
customer.email, SUM(payment.amount) AS  
total_spent  
FROM customer  
JOIN payment ON customer.customer_id =  
payment.customer_id  
GROUP BY customer.customer_id  
ORDER BY total_spent DESC  
LIMIT 10;
```

# 1. Identify the top 10 customers and their email so we can reward them

Penjelasan:

1. ``SELECT customer.first_name, customer.last_name, customer.email, SUM(payment.amount) AS total_spent``

- Ini adalah klausa SELECT yang digunakan untuk memilih kolom-kolom yang akan ditampilkan dalam hasil query.
- `customer.first_name`, `customer.last_name`, dan `customer.email` adalah kolom-kolom dari tabel `customer` yang akan ditampilkan.
- `SUM(payment.amount) AS total_spent` adalah kolom baru yang akan ditampilkan. `SUM(payment.amount)` digunakan untuk menjumlahkan jumlah pembayaran (`amount`) dari tabel `payment` untuk setiap pelanggan.
- Alias `total_spent` diberikan pada kolom yang dihasilkan agar lebih mudah diidentifikasi dalam hasil query.

2. ``FROM customer``

- Ini adalah klausa FROM yang menunjukkan bahwa kita akan mengambil data dari tabel `customer`.

# **1. Identify the top 10 customers and their email so we can reward them**

Penjelasan:

3. ``JOIN payment ON customer.customer_id = payment.customer_id``

- Ini adalah klausa JOIN yang menghubungkan tabel customer dengan tabel payment menggunakan kolom customer\_id.
- customer.customer\_id adalah kolom customer\_id dari tabel customer.
- payment.customer\_id adalah kolom customer\_id dari tabel payment.
- Dengan melakukan join ini, kita menggabungkan data dari kedua tabel berdasarkan kolom customer\_id yang sesuai.

4. ``GROUP BY customer.customer_id``

- Ini adalah klausa GROUP BY yang digunakan untuk mengelompokkan data berdasarkan customer\_id.
- Dengan mengelompokkan data, kita dapat melakukan operasi agregasi seperti SUM() pada kolom-kolom lainnya yang tidak termasuk dalam klausa GROUP BY.

# **1. Identify the top 10 customers and their email so we can reward them**

Penjelasan:

## **5. `ORDER BY total\_spent DESC`**

- Ini adalah klausa ORDER BY yang digunakan untuk mengurutkan hasil query berdasarkan kolom total\_spent secara menurun (descending).
- Hasil query akan diurutkan berdasarkan jumlah total yang dihabiskan oleh pelanggan.

## **6. `LIMIT 10`**

- Ini adalah klausa LIMIT yang digunakan untuk membatasi jumlah baris yang akan ditampilkan dalam hasil query.
- Dalam hal ini, hanya 10 pelanggan teratas yang akan ditampilkan.

# 1. Identify the top 10 customers and their email so we can reward them

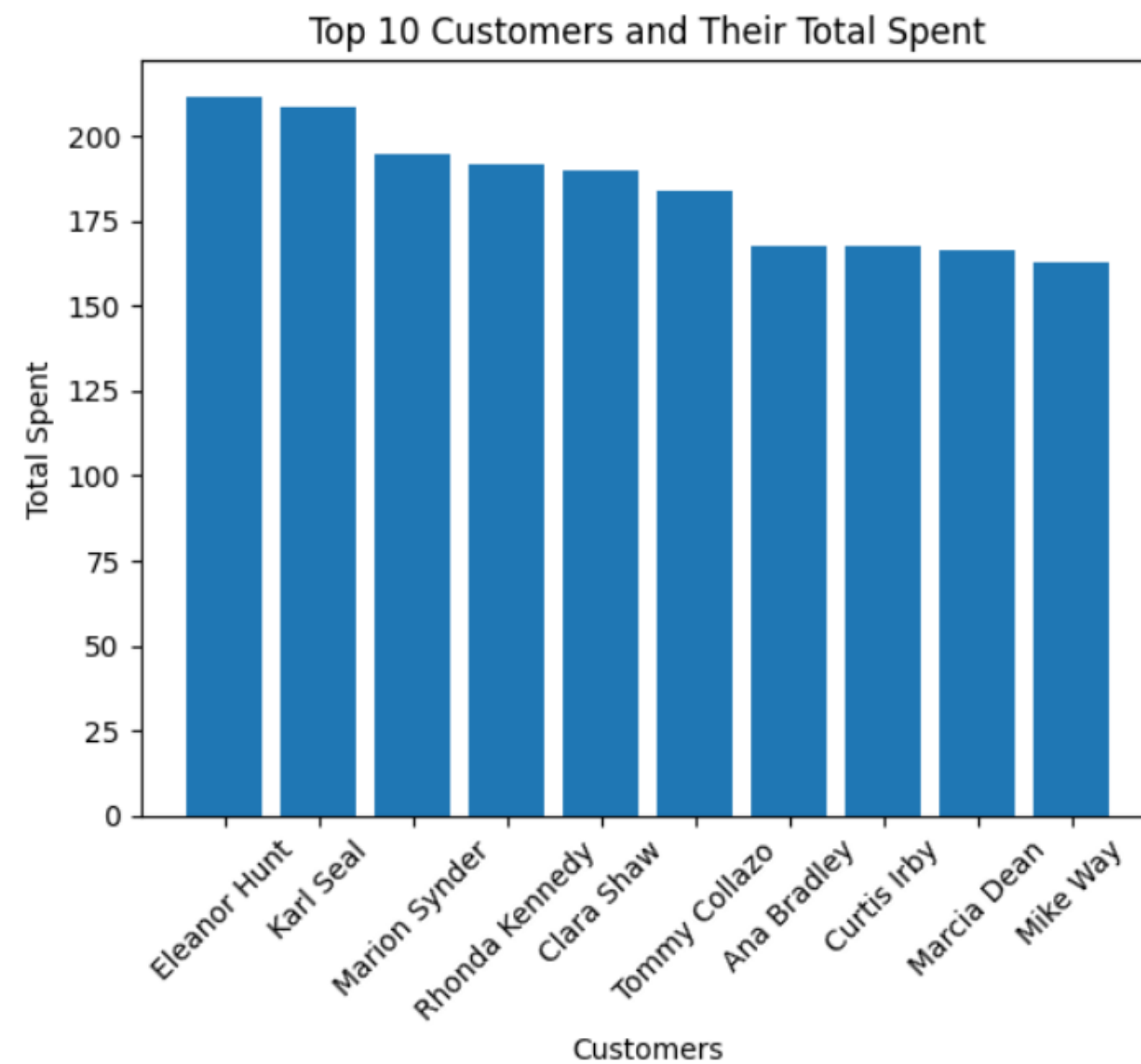
Output:

	first_name character varying (45) 🔒	last_name character varying (45) 🔒	email character varying (50) 🔒	total_spent numeric 🔒
1	Eleanor	Hunt	eleanor.hunt@sakilacustomer.org	211.55
2	Karl	Seal	karl.seal@sakilacustomer.org	208.58
3	Marion	Snyder	marion.snyder@sakilacustomer.org	194.61
4	Rhonda	Kennedy	rhonda.kennedy@sakilacustomer.org	191.62
5	Clara	Shaw	clara.shaw@sakilacustomer.org	189.60
6	Tommy	Collazo	tommy.collazo@sakilacustomer.org	183.63
7	Ana	Bradley	ana.bradley@sakilacustomer.org	167.67
8	Curtis	Irby	curtis.irby@sakilacustomer.org	167.62
9	Marcia	Dean	marcia.dean@sakilacustomer.org	166.61
10	Mike	Way	mike.way@sakilacustomer.org	162.67

Dapat dilihat pada output diatas bahwa telah ditampilkan top 10 customer berdasarkan total spent mereka beserta emailnya.

# 1. Identify the top 10 customers and their email so we can reward them

Visualisasi menggunakan Colab Google:





## 2. Identify the bottom 10 customers and their emails

SQL Query:

```
SELECT customer.first_name, customer.last_name,  
customer.email, SUM(payment.amount) AS  
total_spent  
FROM customer  
JOIN payment ON customer.customer_id =  
payment.customer_id  
GROUP BY customer.customer_id  
ORDER BY total_spent ASC  
LIMIT 10;
```

## 2. Identify the bottom 10 customers and their emails

Penjelasan:

1. `SELECT customer.first_name, customer.last_name, customer.email, SUM(payment.amount) AS total_spent`: Menentukan kolom yang ingin ditampilkan dalam hasil query. Di sini, kita memilih kolom `first_name`, `last_name`, dan `email` dari tabel `customer`. Selain itu, kita juga menghitung jumlah total yang dihabiskan oleh pelanggan dengan menggunakan fungsi agregat `SUM(payment.amount)`, dan memberikan alias `total_spent` pada hasilnya.
2. `FROM customer`: Menentukan tabel yang akan digunakan dalam query ini, yaitu tabel `customer`.
3. `JOIN payment ON customer.customer_id = payment.customer_id`: Melakukan operasi JOIN antara tabel `customer` dan `payment` berdasarkan kolom `customer_id` yang sama pada kedua tabel. Hal ini menghubungkan data pelanggan dengan data pembayaran untuk pelanggan yang sama.

## 2. Identify the bottom 10 customers and their emails

Penjelasan:

4. ``GROUP BY customer.customer_id``: Mengelompokkan hasil berdasarkan kolom ``customer_id``. Dalam konteks ini, kita ingin menghitung jumlah total yang dihabiskan oleh setiap pelanggan, sehingga kita perlu mengelompokkan hasil berdasarkan pelanggan.

5. ``ORDER BY total_spent ASC``: Mengurutkan hasil berdasarkan kolom ``total_spent`` secara ascending (menaik). Dengan kata lain, hasil query akan diurutkan mulai dari jumlah total yang paling kecil.

6. ``LIMIT 10``: Membatasi jumlah baris hasil yang ditampilkan menjadi 10. Dalam kasus ini, kita hanya tertarik dengan 10 pelanggan dengan jumlah total pengeluaran terendah.

## 2. Identify the bottom 10 customers and their emails

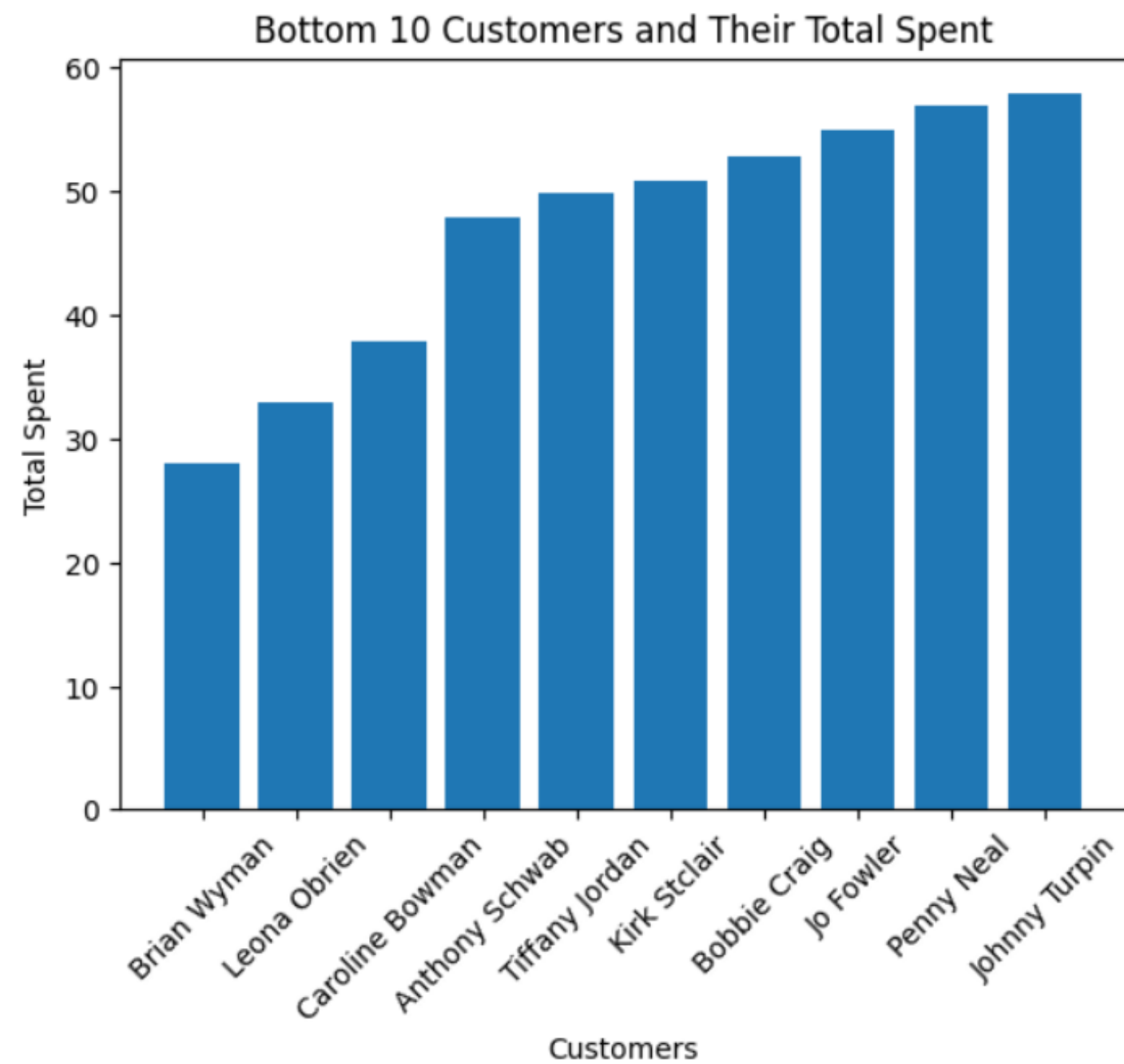
Output:

	first_name character varying (45) 🔒	last_name character varying (45) 🔒	email character varying (50) 🔒	total_spent numeric 🔒
1	Brian	Wyman	brian.wyman@sakilacustomer.org	27.93
2	Leona	Obrien	leona.obrien@sakilacustomer.org	32.90
3	Caroline	Bowman	caroline.bowman@sakilacustomer.org	37.87
4	Anthony	Schwab	anthony.schwab@sakilacustomer.org	47.85
5	Tiffany	Jordan	tiffany.jordan@sakilacustomer.org	49.88
6	Kirk	Stclair	kirk.stclair@sakilacustomer.org	50.83
7	Bobbie	Craig	bobbie.craig@sakilacustomer.org	52.81
8	Jo	Fowler	jo.fowler@sakilacustomer.org	54.85
9	Penny	Neal	penny.neal@sakilacustomer.org	56.84
10	Johnny	Turpin	johnny.turpin@sakilacustomer.org	57.81

Dapat dilihat pada output diatas bahwa telah ditampilkan 10 customer terbawah berdasarkan total spent mereka beserta emailnya.

## 2. Identify the bottom 10 customers and their emails

Visualisasi menggunakan Colab Google:



### 3. What are the most profitable movie genres (ratings)?

SQL Query:

```
SELECT film.rating, SUM(payment.amount) AS  
total_revenue  
FROM rental  
JOIN inventory ON rental.inventory_id =  
inventory.inventory_id  
JOIN film ON inventory.film_id = film.film_id  
JOIN payment ON rental.rental_id = payment.rental_id  
GROUP BY film.rating  
ORDER BY total_revenue DESC;
```

### **3. What are the most profitable movie genres (ratings)?**

Penjelasan:

1. **SELECT film.rating, SUM(payment.amount) AS total\_revenue:** Ini adalah bagian SELECT statement yang menentukan kolom-kolom yang akan ditampilkan dalam hasil query. Di sini, kita memilih kolom film.rating (rating film) dan menghitung jumlah total pendapatan menggunakan fungsi SUM(payment.amount), dan memberikan alias total\_revenue untuk hasil perhitungan tersebut.
2. **FROM rental:** Ini adalah klausa FROM yang menentukan tabel utama dari mana data akan diambil, yaitu tabel rental.
3. **JOIN inventory ON rental.inventory\_id = inventory.inventory\_id:** Ini adalah klausa JOIN yang menghubungkan tabel rental dengan tabel inventory berdasarkan kolom inventory\_id. Hal ini dilakukan untuk mendapatkan informasi tentang film yang disewakan dalam setiap transaksi penyewaan.

### **3. What are the most profitable movie genres (ratings)?**

Penjelasan:

4. JOIN film ON inventory.film\_id = film.film\_id: Ini adalah klausa JOIN tambahan yang menghubungkan tabel inventory dengan tabel film berdasarkan kolom film\_id. Ini diperlukan untuk mendapatkan informasi tentang rating film yang terkait dengan setiap transaksi penyewaan.

5. JOIN payment ON rental.rental\_id = payment.rental\_id: Ini adalah klausa JOIN tambahan yang menghubungkan tabel rental dengan tabel payment berdasarkan kolom rental\_id. Ini diperlukan untuk menghubungkan informasi pembayaran dengan transaksi penyewaan yang sesuai.



6. GROUP BY film.rating: Ini adalah klausa GROUP BY yang mengelompokkan data berdasarkan kolom film.rating. Ini berarti hasil query akan mengelompokkan total pendapatan berdasarkan rating film.

7. ORDER BY total\_revenue DESC: Ini adalah klausa ORDER BY yang mengurutkan hasil query berdasarkan kolom total\_revenue (total pendapatan) secara descending, sehingga rating film dengan total pendapatan tertinggi akan muncul terlebih dahulu.



### 3. What are the most profitable movie genres (ratings)?

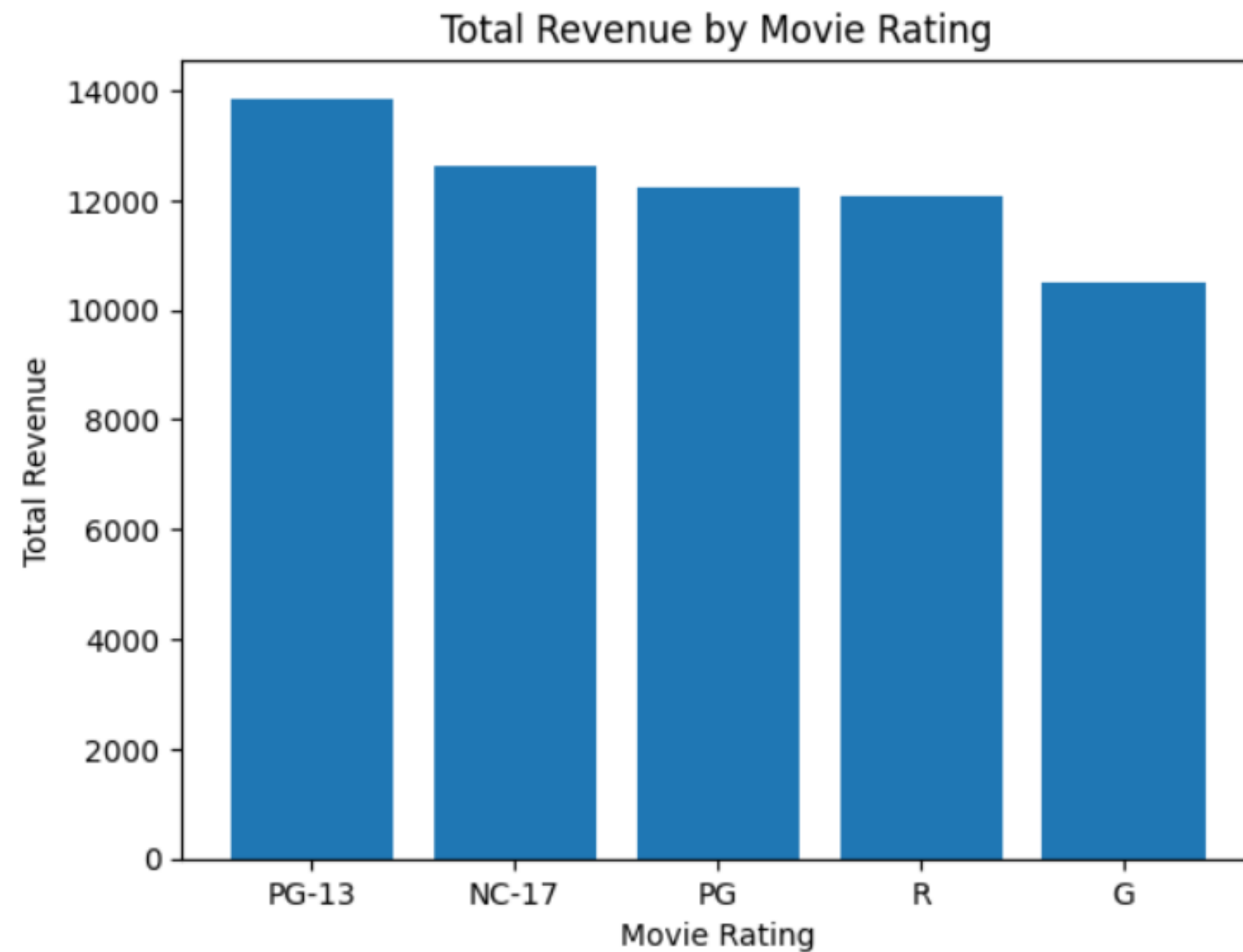
Output:

	rating mpaa_rating 	total_revenue numeric 
1	PG-13	13855.56
2	NC-17	12634.92
3	PG	12236.65
4	R	12073.03
5	G	10511.88

Dapat dilihat pada output diatas bahwa telah ditampilkan daftar 5 genres (ratings) yang telah diurutkan sesuai total revenue untuk mencari tahu genre apa yang paling menghasilkan profit. Genre PG-13 merupakan genre yang paling menghasilkan profit karena memiliki total revenue sebesar 13855.56.

### 3. What are the most profitable movie genres (ratings)?

Visualisasi menggunakan Colab Google:



## 4. How many rented movies were returned late, early, and on time?

SQL Query:

```
SELECT  
  CASE  
    WHEN return_date > rental_date THEN 'late'  
    WHEN return_date < rental_date THEN 'early'  
    ELSE 'on time'  
  END AS status,  
  COUNT(*) AS count  
FROM rental  
GROUP BY status;
```

## 4. How many rented movies were returned late, early, and on time?

Penjelasan:

1. **SELECT**: Kata kunci untuk memulai pemilihan kolom yang ingin ditampilkan dalam hasil query.
2. **CASE**: Ini adalah ekspresi kondisional yang digunakan untuk mengevaluasi kondisi-kondisi tertentu dan menghasilkan nilai berdasarkan kondisi yang cocok.
3. **WHEN return\_date > rental\_date THEN 'late'**: Ini adalah bagian pertama dari ekspresi CASE. Ini mengevaluasi apakah tanggal pengembalian (return\_date) lebih besar dari tanggal sewa (rental\_date). Jika benar, maka nilai yang dihasilkan adalah 'late' (terlambat).
4. **WHEN return\_date < rental\_date THEN 'early'**: Ini adalah bagian kedua dari ekspresi CASE. Ini mengevaluasi apakah tanggal pengembalian (return\_date) lebih kecil dari tanggal sewa (rental\_date). Jika benar, maka nilai yang dihasilkan adalah 'early' (lebih awal).
5. **ELSE 'on time'**: Ini adalah bagian terakhir dari ekspresi CASE. Jika kedua kondisi sebelumnya tidak terpenuhi, maka nilai yang dihasilkan adalah 'on time' (tepat waktu).

## 4. How many rented movies were returned late, early, and on time?

Penjelasan:

6. AS status: Ini adalah bagian untuk memberikan alias atau nama kolom baru pada hasil ekspresi CASE. Dalam hal ini, kolom hasil ekspresi CASE akan diberi nama 'status'.

7. COUNT(\*) AS count: Ini adalah bagian untuk menghitung jumlah baris atau nilai dalam setiap grup. COUNT(\*) menghitung jumlah baris yang ada dalam setiap grup, dan AS count memberikan alias 'count' pada kolom hasil.

8. FROM rental: Ini adalah bagian yang menunjukkan tabel sumber data dari mana data diambil. Dalam hal ini, data diambil dari tabel 'rental'.

9. GROUP BY status: Ini adalah bagian yang digunakan untuk mengelompokkan hasil berdasarkan kolom 'status'. Hal ini memungkinkan kita untuk menghitung jumlah film dalam setiap kategori (late, early, on time) secara terpisah.

## 4. How many rented movies were returned late, early, and on time?

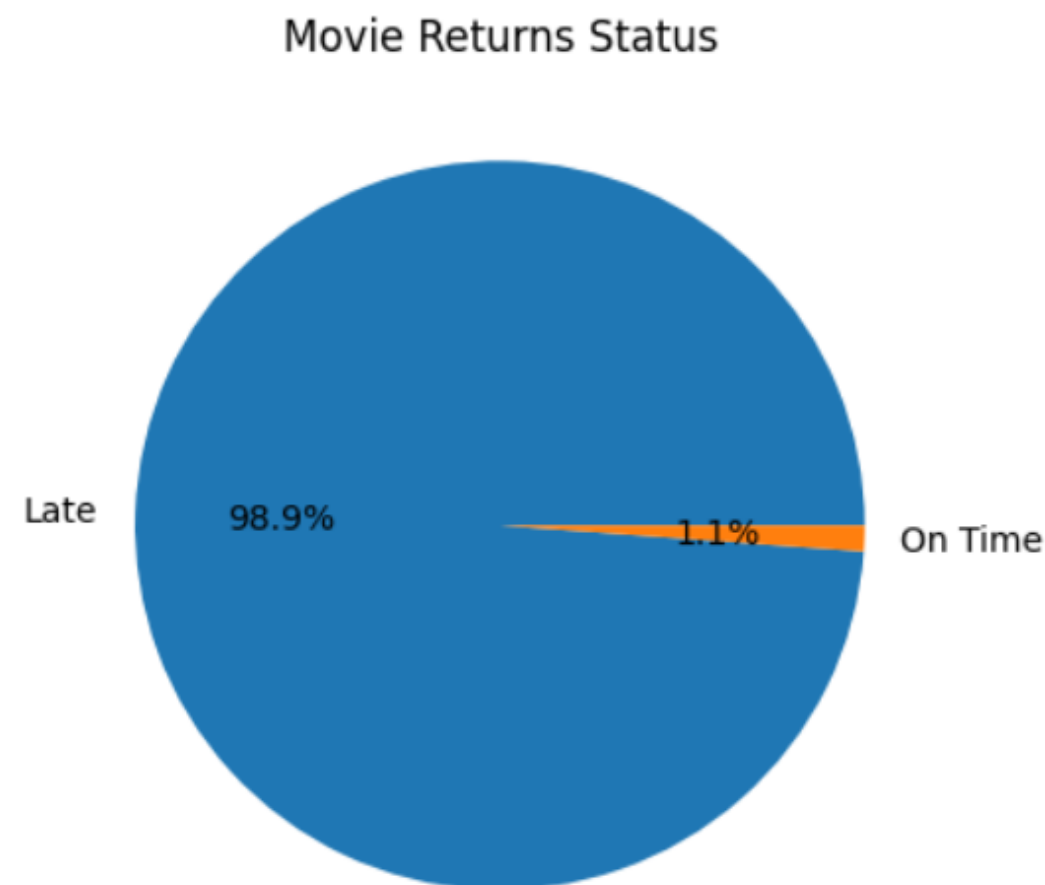
Output:

	status text	count bigint
1	late	15861
2	on time	183

Dapat dilihat pada output diatas bahwa telah ditampilkan hasil status dari peminjaman. Hanya terdapat status "late" dan "on time" yang masing-masing berjumlah 15861 dan 183. Tidak terdapat status "early" karena ternyata setelah dicek menggunakan SQL Query tidak terdapat peminjam yang mengembalikan dvd film lebih awal.

## 4. How many rented movies were returned late, early, and on time?

Visualisasi menggunakan Colab Google:



Visualisasi menggunakan pie chart diatas menunjukkan bahwa 98.9% peminjam terlambat dalam mengembalikan dvd film, sedangkan yang on time hanya sebesar 1.1%.

## 5. What is the customer base in the countries where we have a presence?

SQL Query:

```
SELECT country.country, COUNT(*) AS count
FROM address
JOIN city ON address.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
JOIN customer ON address.address_id =
customer.address_id
GROUP BY country.country
ORDER BY count DESC;
```



## **5. What is the customer base in the countries where we have a presence?**

Penjelasan:

1. **SELECT country.country, COUNT(\*) AS count:** Mengawali query dengan SELECT statement yang akan memilih kolom country.country (nama negara) dan menghitung jumlah baris menggunakan fungsi COUNT(\*). Alias count diberikan pada kolom hasil perhitungan tersebut.
2. **FROM address:** Menggunakan tabel address sebagai basis untuk query.
3. **JOIN city ON address.city\_id = city.city\_id:** Menggabungkan tabel address dengan tabel city berdasarkan kolom city\_id agar dapat mengakses data kota yang terkait dengan alamat pelanggan.
4. **JOIN country ON city.country\_id = country.country\_id:** Menggabungkan tabel city dengan tabel country berdasarkan kolom country\_id agar dapat mengakses data negara yang terkait dengan kota.

## **5. What is the customer base in the countries where we have a presence?**

Penjelasan:

5. JOIN customer ON address.address\_id = customer.address\_id: Menggabungkan tabel address dengan tabel customer berdasarkan kolom address\_id agar dapat mengakses data pelanggan yang terkait dengan alamat.

6. GROUP BY country.country: Mengelompokkan hasil berdasarkan nilai pada kolom country.country, sehingga setiap negara akan memiliki satu baris dalam hasil dengan jumlah pelanggan yang sesuai.

7. ORDER BY count DESC: Mengurutkan hasil secara menurun (descending) berdasarkan kolom count, yaitu jumlah pelanggan di setiap negara. Negara dengan jumlah pelanggan terbanyak akan muncul di atas.

# 5. What is the customer base in the countries where we have a presence?

Output:

	country character varying (50)	count bigint
1	India	60
2	China	53
3	United States	36
4	Japan	31
5	Mexico	30
6	Brazil	28
7	Russian Federation	28
8	Philippines	20
9	Turkey	15
10	Indonesia	14
11	Argentina	13
12	Nigeria	13
13	South Africa	11
14	Taiwan	10
15	United Kingdom	9
16	Poland	8
17	Iran	8
18	Germany	7
19	Italy	7
20	Venezuela	7
21	Egypt	6
22	Colombia	6
23	Ukraine	6

	country character varying (50)	count bigint
24	Vietnam	6
25	Spain	5
26	Canada	5
27	South Korea	5
28	Pakistan	5
29	Netherlands	5
30	Saudi Arabia	5
31	Israel	4
32	France	4
33	Yemen	4
34	Peru	4
35	Dominican Republic	3
36	Bangladesh	3
37	Thailand	3
38	Algeria	3
39	Chile	3
40	Malaysia	3
41	Mozambique	3
42	Switzerland	3
43	Ecuador	3
44	Paraguay	3
45	Austria	3
46	United Arab Emirates	3

	country character varying (50)	count bigint
47	Morocco	3
48	Tanzania	3
49	Myanmar	2
50	Bulgaria	2
51	Cameroon	2
52	Cambodia	2
53	Congo, The Democratic Republic of the	2
54	Sudan	2
55	Romania	2
56	Latvia	2
57	Yugoslavia	2
58	Puerto Rico	2
59	Kazakstan	2
60	Greece	2
61	Bolivia	2
62	Kenya	2
63	French Polynesia	2
64	Angola	2
65	Belarus	2
66	Oman	2
67	Azerbaijan	2
68	Hungary	1
69	American Samoa	1

	country character varying (50)	count bigint
70	Armenia	1
71	Sri Lanka	1
72	French Guiana	1
73	Sweden	1
74	Faroe Islands	1
75	Ethiopia	1
76	Bahrain	1
77	Czech Republic	1
78	Lithuania	1
79	Turkmenistan	1
80	Tunisia	1
81	Virgin Islands, U.S.	1
82	Malawi	1
83	Chad	1
84	Afghanistan	1
85	Greenland	1
86	Moldova	1
87	Gambia	1
88	Holy See (Vatican City State)	1
89	Estonia	1
90	Slovakia	1
91	North Korea	1
92	Nauru	1

# 5. What is the customer base in the countries where we have a presence?

Output:

	country character varying (50)	count bigint
93	Liechtenstein	1
94	Senegal	1
95	Zambia	1
96	Hong Kong	1
97	Kuwait	1
98	Madagascar	1
99	Runion	1
100	Saint Vincent and the Grenadines	1
101	Tuvalu	1
102	Finland	1
103	Iraq	1
104	Anguilla	1
105	Brunei	1
106	Tonga	1
107	Nepal	1
108	New Zealand	1

Dapat dilihat pada output diatas bahwa telah ditampilkan yaitu kolom "country" yang menampilkan nama-nama negara dan kolom "count" yang menampilkan jumlah pelanggan di setiap negara. Terdapat 108 negara yang menjadi basis pelanggan dan negara dengan pelanggan terbanyak yaitu India.

## 6. Which country is the most profitable for the business?

SQL Query:

```
SELECT country.country, SUM(payment.amount) AS total_revenue
FROM address
JOIN city ON address.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
JOIN customer ON address.address_id = customer.address_id
JOIN payment ON customer.customer_id = payment.customer_id
GROUP BY country.country
ORDER BY total_revenue DESC
LIMIT 1;
```

## 6. Which country is the most profitable for the business?

Penjelasan:

1. `SELECT country.country, SUM(payment.amount) AS total_revenue`: Bagian ini menentukan kolom-kolom yang akan ditampilkan dalam hasil query. Kita memilih kolom `country.country` untuk mendapatkan nama negara, dan `SUM(payment.amount)` untuk menghitung total pendapatan dari pembayaran.
2. `FROM address`: Klausa `FROM` menentukan tabel-tabel yang digunakan dalam query. Di sini, kita menggunakan tabel `address` sebagai titik awal query.
3. `JOIN city ON address.city_id = city.city_id`: Bagian ini adalah klausa `JOIN` yang menggabungkan tabel `address` dengan tabel `city` berdasarkan kolom `city_id` yang sesuai di kedua tabel.
4. `JOIN country ON city.country_id = country.country_id`: Klausa `JOIN` ini menggabungkan tabel `city` dengan tabel `country` berdasarkan kolom `country_id` yang sesuai di kedua tabel.

## 6. Which country is the most profitable for the business?

Penjelasan:

5. JOIN customer ON address.address\_id = customer.address\_id: Klausula JOIN ini menggabungkan tabel address dengan tabel customer berdasarkan kolom address\_id yang sesuai di kedua tabel.
6. JOIN payment ON customer.customer\_id = payment.customer\_id: Bagian ini menggabungkan tabel customer dengan tabel payment berdasarkan kolom customer\_id yang sesuai di kedua tabel.
7. GROUP BY country.country: Bagian ini menyatakan bahwa hasil query akan dikelompokkan berdasarkan nilai unik pada kolom country.country. Ini memungkinkan perhitungan yang dilakukan pada kolom lain, seperti SUM(payment.amount), dilakukan dalam kelompok negara yang berbeda.
8. ORDER BY total\_revenue DESC: Bagian ini menyatakan bahwa hasil query akan diurutkan berdasarkan kolom total\_revenue (pendapatan total) secara menurun (descending). Dengan demikian, negara dengan pendapatan tertinggi akan muncul pertama dalam hasil query.
9. LIMIT 1: Klausula LIMIT mengatur jumlah baris hasil query yang ingin ditampilkan. Dalam kasus ini, kita hanya ingin menampilkan satu baris hasil, yaitu negara dengan pendapatan tertinggi.

## 6. Which country is the most profitable for the business?

Output:

	country character varying (50) 🔒	total_revenue numeric 🔒
1	India	6034.78

Dapat dilihat pada output diatas bahwa negara India merupakan negara yang paling profit untuk bisnis. Hal tersebut dikarenakan total revenue negara India merupakan total revenue terbanyak. Didukung juga India merupakan negara dengan pelanggan terbanyak dalam database dvdrental.



## 7. What is the average rental rate per movie genre (rating)?

SQL Query:

```
SELECT film.rating, AVG(rental_rate) AS  
avg_rental_rate  
FROM film  
GROUP BY film.rating  
ORDER BY avg_rental_rate DESC;
```



## 7. What is the average rental rate per movie genre (rating)?

Penjelasan:

1. `SELECT film.rating, AVG(rental_rate) AS avg_rental_rate`: Query ini memilih kolom `film.rating` dan menghitung rata-rata (`AVG()`) dari kolom `rental_rate` dari tabel `film`. Alias `avg_rental_rate` diberikan untuk menggantikan nama kolom hasil rata-rata.
2. `FROM film`: Ini menentukan tabel yang digunakan dalam query ini, yaitu tabel `film`.
3. `GROUP BY film.rating`: Dengan menggunakan klausa `GROUP BY`, data akan dikelompokkan berdasarkan nilai unik pada kolom `film.rating`. Hal ini berarti query akan menghasilkan rata-rata harga sewa untuk setiap rating film yang berbeda.
4. `ORDER BY avg_rental_rate DESC`: Klausa `ORDER BY` digunakan untuk mengurutkan hasil query berdasarkan kolom `avg_rental_rate` secara menurun (`DESC`). Dalam konteks ini, rating film dengan rata-rata harga sewa yang lebih tinggi akan muncul terlebih dahulu dalam hasil query.

# 7. What is the average rental rate per movie genre (rating)?

Output:

	rating mpaa_rating 	avg_rental_rate numeric 
1	PG	3.0518556701030928
2	PG-13	3.0348430493273543
3	NC-17	2.9709523809523810
4	R	2.9387179487179487
5	G	2.8888764044943820

Dapat dilihat pada output diatas bahwa genre (rating) PG merupakan genre dengan tarif sewa rata-rata per film tertinggi yaitu 3.0518. Sedangkan tarif sewa rata-rata terendah yaitu film dengan genre (rating) G yaitu 2.8888.

## 7. What is the average rental rate per movie genre (rating)?

Visualisasi menggunakan Colab Google:

