

```
In [1]: import atoti as tt
```

Welcome to Atoti 0.8.12!

By using this community edition, you agree with the license available at <https://docs.atoti.io/latest/eula.html>.

Browse the official documentation at <https://docs.atoti.io>.

Join the community at <https://www.atoti.io/register>.

Atoti collects telemetry data, which is used to help understand how to improve the product.

If you don't wish to send usage data, you can request a trial license at <https://www.atoti.io/evaluation-license-request>.

You can hide this message by setting the `ATOTI_HIDE_EULA_MESSAGE` environment variable to True.

```
In [2]: session = tt.Session()
```

```
In [4]: dbname = "sakila"
        uname = "postgres"
        passwd = "postgres"
```

```
connSql = f"postgresql://localhost:55432/{dbname}?user={uname}&password={passwd}"
print(connSql)
```

postgresql://localhost:55432/sakila?user=postgres&password=postgres

```
In [25]: host = "localhost"
        port = "55432"

        # Membuat string koneksi
        connSql = f"postgresql://{uname}:{passwd}@{host}:{port}/{dbname}"
        print(connSql)

        # Membuat engine SQLAlchemy
        engine = create_engine(connSql)
```

postgresql://postgres:postgres@localhost:55432/sakila

```
In [5]: products=session.read_sql(
        "select * from actor;",
        url=connSql,
        table_name= "actor",
        keys={"actor_id"})
```

```
In [6]: products.head()
```

Out[6]:

	first_name	last_name	last_update
actor_id			
20	LUCILLE	TRACY	2006-02-15 04:34:33
7	GRACE	MOSTEL	2006-02-15 04:34:33
30	SANDRA	PECK	2006-02-15 04:34:33
23	SANDRA	KILMER	2006-02-15 04:34:33
57	JUDE	CRUISE	2006-02-15 04:34:33

In [34]: `import pandas as pd`
`from sqlalchemy import create_engine`

1.Kategori Film Terpopuler dari Masing-masing Negara:

In [39]: `# Mendefinisikan query untuk mendapatkan kategori film terpopuler dari setiap negara`
`query = """`
`SELECT country.country, category.name as category, COUNT(category.name) as category`
`FROM customer`
`JOIN address ON customer.address_id = address.address_id`
`JOIN city ON address.city_id = city.city_id`
`JOIN country ON city.country_id = country.country_id`
`JOIN rental ON customer.customer_id = rental.customer_id`
`JOIN inventory ON rental.inventory_id = inventory.inventory_id`
`JOIN film ON inventory.film_id = film.film_id`
`JOIN film_category ON film.film_id = film_category.film_id`
`JOIN category ON film_category.category_id = category.category_id`
`GROUP BY country.country, category.name`
`ORDER BY country.country, category_count DESC;`
`"""`

`# Membaca hasil query ke dalam DataFrame`
`df = pd.read_sql(query, engine)`

`# Mendapatkan kategori terpopuler dari setiap negara`
`popular_genres = df.groupby('country').first().reset_index()`
`popular_genres = popular_genres[['country', 'category']]`

`# Menampilkan hasil`
`popular_genres`

Out[39]:

	country	category
0	Afghanistan	Comedy
1	Algeria	Sci-Fi
2	American Samoa	Sports
3	Angola	Animation
4	Anguilla	Travel
...
103	Vietnam	New
104	Virgin Islands, U.S.	Animation
105	Yemen	Action
106	Yugoslavia	Animation
107	Zambia	Animation

108 rows × 2 columns

penjelasan langkah-langkah yang dilakukan oleh kode di atas:

1. **Pendefinisian Query SQL:** Query SQL yang diberikan bertujuan untuk mengambil data kategori film terpopuler dari setiap negara. Query tersebut menggabungkan beberapa tabel, seperti tabel `customer`, `address`, `city`, `country`, `rental`, `inventory`, `film`, `film_category`, dan `category`. Query tersebut melakukan JOIN antar tabel berdasarkan hubungan antar kunci primer dan kunci asing di antara tabel-tabel tersebut.
2. **Eksekusi Query dan Pembacaan ke DataFrame:** Setelah query SQL didefinisikan, selanjutnya query dieksekusi menggunakan fungsi `pd.read_sql()`. Hasil dari query kemudian dibaca ke dalam DataFrame Pandas yang disimpan dalam variabel `df`.

untuk fungsi SQL dalam kueri tersebut:

1. **SELECT:** Kita memilih kolom-kolom `country.country`, `category.name` (yang di-alias sebagai `category`), dan juga menggunakan fungsi agregat `COUNT(category.name)` untuk menghitung jumlah kategori film yang ada.
2. **FROM:** Data diambil dari tabel `customer`, yang bergabung dengan tabel-tabel lainnya untuk mendapatkan informasi yang dibutuhkan.
3. **JOIN:** Dalam total ada 7 JOIN yang dilakukan dengan tabel-tabel berikut:
 - `customer`, `address`, `city`, dan `country`: Ini dilakukan untuk mendapatkan informasi tentang negara dari alamat pelanggan.

- `rental` dan `inventory` : Digunakan untuk mengaitkan transaksi sewa dengan inventaris film.
 - `film` : Digunakan untuk mendapatkan informasi tentang film.
 - `film_category` dan `category` : Untuk mengaitkan kategori film dengan film itu sendiri.
4. **GROUP BY**: Digunakan untuk mengelompokkan hasil berdasarkan nilai-nilai tertentu dari satu atau beberapa kolom. Di sini, kita mengelompokkan hasil berdasarkan `country.country` (negara) dan `category.name` (kategori film) untuk menghitung jumlah masing-masing kategori film di setiap negara.
 5. **COUNT(category.name)**: Fungsi agregat ini digunakan untuk menghitung jumlah kategori film yang muncul dalam setiap kelompok (negara dan kategori).
 6. **ORDER BY**: Digunakan untuk mengurutkan hasil berdasarkan kolom tertentu. Di sini, kita mengurutkan hasil berdasarkan nama negara (`country.country`) secara ascending (default), dan jumlah kategori film terbanyak (`category_count`) dalam setiap negara secara descending (`DESC`).

Setelah menjalankan kueri SQL menggunakan `pd.read_sql()` , hasilnya dibaca ke dalam DataFrame Pandas yang disimpan dalam variabel `df` . Selanjutnya, kita menggunakan metode `groupby()` untuk mengambil kategori terpopuler dari setiap negara dengan memilih baris pertama dari setiap kelompok negara, dan kemudian menampilkan hasilnya. 3. **Menampilkan Hasil**: Hasil akhir dari kategori film terpopuler dari setiap negara kemudian ditampilkan dengan menggunakan perintah `print(popular_genres)` .

Kode tersebut melakukan pengambilan dan pengolahan data untuk mendapatkan kategori film terpopuler dari setiap negara dalam database yang digunakan.

2. Sebutkan jumlah customer masing-masing negara!

In [102...

```
#kueri SQL untuk menghitung jumlah customer masing-masing negara
query = '''
SELECT c.country, COUNT(*) AS total_customers
FROM customer cu
JOIN address a ON cu.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country c ON ci.country_id = c.country_id
GROUP BY c.country

'''

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
df_customers_by_country = pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
df_customers_by_country
import psycopg2
import pandas as pd

# Establishing a connection to the database
```

```

conn = psycopg2.connect(
    dbname="sakila",
    user="postgres",
    password="postgres",
    host="localhost",
    port="55432"
)

# SQL query
sql_query = """
SELECT country, COUNT(customer_id) AS total_customers
FROM customer
JOIN address ON customer.address_id = address.address_id
JOIN city ON address.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
GROUP BY country
ORDER BY total_customers DESC;
"""

# Reading data into pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Closing the connection
conn.close()

# Displaying the DataFrame as a table
print(df)

```

	country	total_customers
0	India	60
1	China	53
2	United States	36
3	Japan	31
4	Mexico	30
..
103	Anguilla	1
104	Brunei	1
105	Tonga	1
106	Nepal	1
107	New Zealand	1

[108 rows x 2 columns]

C:\Users\ASUS\AppData\Local\Temp\ipykernel_11288\2515987948.py:40: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
df = pd.read_sql_query(sql_query, conn)
```

alur code sebagai berikut:

1. **Pendefinisian Query SQL:** Pertama-tama, sebuah query SQL didefinisikan untuk menghitung jumlah pelanggan (customer) dari setiap negara. Query tersebut melakukan JOIN antara tabel `customer`, `address`, `city`, dan `country` berdasarkan hubungan antara kunci primer dan kunci asing.

2. **Eksekusi Query dan Pembacaan ke DataFrame:** Setelah query SQL didefinisikan, langkah selanjutnya adalah mengeksekusi query tersebut menggunakan fungsi `pd.read_sql_query()`. Hasil dari query kemudian dibaca ke dalam DataFrame Pandas yang disimpan dalam variabel `df_customers_by_country`.

fungsi SQL yang digunakan dalam kueri tersebut:

1. **SELECT:** Fungsi ini digunakan untuk memilih kolom-kolom dari tabel yang ingin ditampilkan dalam hasil query. Pada kasus ini, kita memilih kolom `country` dari tabel `country`, dan juga menggunakan fungsi agregat `COUNT(*)` untuk menghitung jumlah baris yang dihasilkan setelah pengelompokan.
2. **FROM:** Fungsi ini menentukan tabel-tabel mana yang digunakan dalam query. Di sini, kita menggunakan tabel `customer`, `address`, `city`, dan `country` sebagai sumber data.
3. **JOIN:** Fungsi ini digunakan untuk menggabungkan data dari beberapa tabel berdasarkan kunci-kunci yang sesuai. Kita menggunakan klausa JOIN untuk menghubungkan tabel `customer` dengan tabel `address`, `city`, dan `country` sesuai dengan relasi kunci yang telah ditentukan.
4. **ON:** Klausa ini digunakan untuk menentukan kriteria penggabungan antara dua tabel. Pada kasus ini, kita menggunakan klausa ON untuk menentukan relasi antara kunci-kunci dalam tabel yang di-join.
5. **GROUP BY:** Fungsi ini digunakan untuk mengelompokkan hasil query berdasarkan nilai-nilai tertentu dari satu atau beberapa kolom. Di sini, kita mengelompokkan hasil berdasarkan nilai kolom `country`.
6. **COUNT(*):** Fungsi agregat ini digunakan untuk menghitung jumlah baris yang dihasilkan setelah pengelompokan. Dalam konteks ini, kita menghitung jumlah baris yang mewakili jumlah customer untuk setiap negara.
7. **Tampilkan DataFrame:** DataFrame yang berisi jumlah pelanggan masing-masing negara kemudian ditampilkan dengan menggunakan perintah `print(df_customers_by_country)`.

Dengan demikian, kode ini melakukan pengambilan dan pengolahan data untuk menghitung jumlah pelanggan dari setiap negara dalam database yang digunakan. Hasilnya adalah DataFrame yang berisi dua kolom: nama negara dan jumlah pelanggan di negara tersebut.

3. 5 Aktor yang Paling Banyak Bermain Film

In [101...

```
#kueri SQL untuk mencari 5 Aktor yang Paling Banyak Bermain Film
query = '''
SELECT
```

```

CONCAT(a.first_name,' ', a.last_name) AS full_name,
COUNT(*) AS total_films
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
GROUP BY a.actor_id, full_name
ORDER BY total_films DESC
LIMIT 5;

'''

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
jumlah_customer_pernegara = pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
jumlah_customer_pernegara

```

Out[101]...

	full_name	total_films
0	GINA DEGENERES	42
1	WALTER TORN	41
2	MARY KEITEL	40
3	MATTHEW CARREY	39
4	SANDRA KILMER	37

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `actor` dan `film_actor`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kolom `actor_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `CONCAT(a.first_name,' ', a.last_name) AS full_name`: Menggabungkan kolom `first_name` dan `last_name` dari tabel `actor` menjadi satu kolom bernama `full_name`.
 - `COUNT(*) AS total_films`: Menghitung jumlah baris atau film yang dimainkan oleh setiap aktor.
3. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`COUNT`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan `actor_id` (ID aktor) dan `full_name` (nama lengkap aktor).
4. **ORDER BY Clause:** Hasil kueri akan diurutkan berdasarkan `total_films` (jumlah film yang dimainkan oleh aktor) secara menurun (`DESC`).
5. **LIMIT Clause:** Karena kita hanya tertarik pada lima aktor teratas yang paling banyak bermain dalam film, kita menggunakan klausa `LIMIT 5` untuk membatasi hasil hanya

untuk lima baris pertama.

Jadi, hasil kueri ini akan memberikan lima aktor teratas yang paling banyak bermain dalam film, diurutkan berdasarkan jumlah film yang dimainkan oleh mereka.

4. Jumlah Store 1 di Masing-masing Negara:

```
In [43]: #kueri SQL untuk menghitung Jumlah Store 1 di Masing-masing Negara:
query = '''
SELECT c.country, COUNT(*) AS total_store_1
FROM store s
JOIN address a ON s.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country c ON ci.country_id = c.country_id
WHERE s.store_id = 1
GROUP BY c.country
'''

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
jumlah_Store_1 = pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
jumlah_Store_1
```

```
Out[43]:
```

	country	total_store_1
0	Canada	1

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `store`, `address`, `city`, dan `country`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubungan yang sesuai. Misalnya, `JOIN address a ON s.address_id = a.address_id` menggabungkan tabel `store` dengan `address` berdasarkan kolom `address_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `c.country`: Kolom ini menunjukkan nama negara.
 - `COUNT(*) AS total_store_1`: Menghitung jumlah toko dengan ID 1 (`store_id = 1`) di setiap negara.
3. **WHERE Clause:** Klausa WHERE digunakan untuk memfilter baris berdasarkan kondisi tertentu. Di sini, kita membatasi hasil hanya pada toko dengan ID 1 (`store_id = 1`).
4. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`COUNT`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan nama negara (`c.country`).

Jadi, hasil kueri ini akan memberikan jumlah toko dengan ID 1 di masing-masing negara.

5. Jumlah Store 2 di Masing-masing Negara:

```
In [44]: #kueri SQL untuk menghitung Jumlah Store 2 di Masing-masing Negara:
query = '''
SELECT c.country, COUNT(*) AS total_store_2
FROM store s
JOIN address a ON s.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country c ON ci.country_id = c.country_id
WHERE s.store_id = 2
GROUP BY c.country

'''

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
jumlah_Store_2 = pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
jumlah_Store_2
```

```
Out[44]:
```

	country	total_store_2
0	Australia	1

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `store`, `address`, `city`, dan `country`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN address a ON s.address_id = a.address_id` menggabungkan tabel `store` dengan `address` berdasarkan kolom `address_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `c.country`: Kolom ini menunjukkan nama negara.
 - `COUNT(*) AS total_store_2`: Menghitung jumlah toko dengan ID 2 (`store_id = 2`) di setiap negara.
3. **WHERE Clause:** Klausa WHERE digunakan untuk memfilter baris berdasarkan kondisi tertentu. Di sini, kita membatasi hasil hanya pada toko dengan ID 2 (`store_id = 2`).
4. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`COUNT`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan nama negara (`c.country`).

Jadi, hasil kueri ini akan memberikan jumlah toko dengan ID 2 di masing-masing negara.

6. Jumlah Revenue Store 1 di Setiap Negara:

```
In [45]: #kueri SQL untuk menghitung Jumlah Revenue Store 1 di Setiap Negara:
query = '''
```

```

SELECT c.country, SUM(p.amount) AS total_revenue_store_1
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN store s ON i.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country c ON ci.country_id = c.country_id
WHERE s.store_id = 1
GROUP BY c.country
'''

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
jumlah_revenue_store_1= pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
jumlah_revenue_store_1

```

```

Out[45]:
  country  total_revenue_store_1
0  Canada                33689.74

```

Alur kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `payment`, `rental`, `inventory`, `store`, `address`, `city`, dan `country`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubungan yang sesuai. Misalnya, `JOIN rental r ON p.rental_id = r.rental_id` menggabungkan tabel `payment` dengan `rental` berdasarkan kolom `rental_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `c.country`: Kolom ini menunjukkan nama negara.
 - `SUM(p.amount) AS total_revenue_store_1`: Menghitung jumlah pendapatan dari setiap transaksi pembayaran (`amount`) di toko dengan ID 1 (`store_id = 1`) di setiap negara.
3. **WHERE Clause:** Klausa WHERE digunakan untuk memfilter baris berdasarkan kondisi tertentu. Di sini, kita membatasi hasil hanya pada toko dengan ID 1 (`store_id = 1`).
4. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`SUM`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan nama negara (`c.country`).

Jadi, hasil kueri ini akan memberikan total pendapatan dari toko dengan ID 1 di setiap negara.

7. Jumlah Revenue Store 2 di Setiap Negara:

```

In [46]: #kueri SQL untuk menghitung Jumlah Revenue Store 2 di Setiap Negara:
query = '''

```

```

SELECT c.country, SUM(p.amount) AS total_revenue_store_2
FROM payment p
JOIN rental r ON p.rental_id = r.rental_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN store s ON i.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
JOIN city ci ON a.city_id = ci.city_id
JOIN country c ON ci.country_id = c.country_id
WHERE s.store_id = 2
GROUP BY c.country

...

# Jalankan kueri dan baca hasilnya ke dalam DataFrame Pandas
jumlah_revenue_store_2 = pd.read_sql_query(query, connSql)

# Tampilkan DataFrame
jumlah_revenue_store_2

```

Out[46]:

	country	total_revenue_store_2
0	Australia	33726.77

Alur kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `payment`, `rental`, `inventory`, `store`, `address`, `city`, dan `country`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubungan yang sesuai. Misalnya, `JOIN rental r ON p.rental_id = r.rental_id` menggabungkan tabel `payment` dengan `rental` berdasarkan kolom `rental_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `c.country`: Kolom ini menunjukkan nama negara.
 - `SUM(p.amount) AS total_revenue_store_2`: Menghitung jumlah pendapatan dari setiap transaksi pembayaran (`amount`) di toko dengan ID 2 (`store_id = 2`) di setiap negara.
3. **WHERE Clause:** Klausa WHERE digunakan untuk memfilter baris berdasarkan kondisi tertentu. Di sini, kita membatasi hasil hanya pada toko dengan ID 2 (`store_id = 2`).
4. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`SUM`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan nama negara (`c.country`).

Jadi, hasil kueri ini akan memberikan total pendapatan dari toko dengan ID 2 di setiap negara.

In [90]:

```

import pandas as pd
from sqlalchemy import create_engine

# Konfigurasi koneksi ke database PostgreSQL

```

```

dbname = "sakila"
uname = "postgres"
passwd = "postgres"
host = "localhost"
port = "55432"

# Membuat string koneksi
connSql = f"postgresql://{uname}:{passwd}@{host}:{port}/{dbname}"

# Membuat engine SQLAlchemy
engine = create_engine(connSql)

# Mendefinisikan query untuk mendapatkan jumlah revenue Store 2 dari setiap negara
query = """
SELECT
    country.country AS country,
    SUM(payment.amount) AS revenue
FROM
    payment
    JOIN rental ON payment.rental_id = rental.rental_id
    JOIN customer ON rental.customer_id = customer.customer_id
    JOIN address ON customer.address_id = address.address_id
    JOIN city ON address.city_id = city.city_id
    JOIN country ON city.country_id = country.country_id
WHERE
    customer.store_id = 2
GROUP BY
    country.country
ORDER BY
    revenue DESC;
"""

# Membaca hasil query ke dalam DataFrame
df = pd.read_sql(query, engine)

# Menampilkan DataFrame
print(df)

```

	country	revenue
0	China	2770.07
1	India	2410.17
2	Mexico	1741.86
3	United States	1701.01
4	Japan	1512.43
..
76	Israel	84.83
77	Kazakistan	77.77
78	American Samoa	71.80
79	Switzerland	58.86
80	Sudan	57.81

[81 rows x 2 columns]

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `payment`, `rental`, `customer`, `address`, `city`, dan `country`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN rental ON payment.rental_id = rental.rental_id` menggabungkan tabel `payment` dengan `rental` berdasarkan kolom `rental_id`.
2. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `country.country AS country`: Kolom ini menunjukkan nama negara.
 - `SUM(payment.amount) AS revenue`: Menghitung total pendapatan (`amount`) dari setiap transaksi pembayaran (`payment`) yang dilakukan oleh pelanggan dari toko dengan ID 2 (`customer.store_id = 2`) di setiap negara.
3. **WHERE Clause:** Klausa WHERE digunakan untuk memfilter baris berdasarkan kondisi tertentu. Di sini, kita membatasi hasil hanya pada pelanggan yang terkait dengan toko dengan ID 2 (`customer.store_id = 2`).
4. **GROUP BY Clause:** Karena kita menggunakan fungsi agregat (`SUM`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan nama negara (`country.country`).
5. **ORDER BY Clause:** Hasil kueri akan diurutkan berdasarkan total pendapatan (`revenue`) secara menurun (`DESC`).

Jadi, hasil kueri ini akan memberikan total pendapatan dari toko dengan ID 2 di setiap negara, diurutkan dari total pendapatan tertinggi hingga terendah.

8. Jumlah Revenue Masing-masing Store Setiap Minggunya:

```
In [89]: import psycopg2
import pandas as pd
import matplotlib.pyplot as plt

# Establishing a connection to the database
conn = psycopg2.connect(
    dbname="sakila",
    user="postgres",
    password="postgres",
    host="localhost",
    port="55432"
)

# SQL query to get weekly revenue for each store
sql_query = """
SELECT
    store.store_id,
    EXTRACT(WEEK FROM payment.payment_date) AS week,
    EXTRACT(YEAR FROM payment.payment_date) AS year,
    SUM(payment.amount) AS weekly_revenue
FROM
    payment
```

```

JOIN
    rental ON payment.rental_id = rental.rental_id
JOIN
    inventory ON rental.inventory_id = inventory.inventory_id
JOIN
    store ON inventory.store_id = store.store_id
GROUP BY
    store.store_id, year, week
ORDER BY
    store.store_id, year, week;
"""

# Reading data into pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Closing the connection
conn.close()

# Ensure 'year' and 'week' are integers
df['year'] = df['year'].astype(int)
df['week'] = df['week'].astype(int)

# Calculate week_date directly
df['week_date'] = pd.to_datetime(df['year'].astype(str) + df['week'].astype(str) +

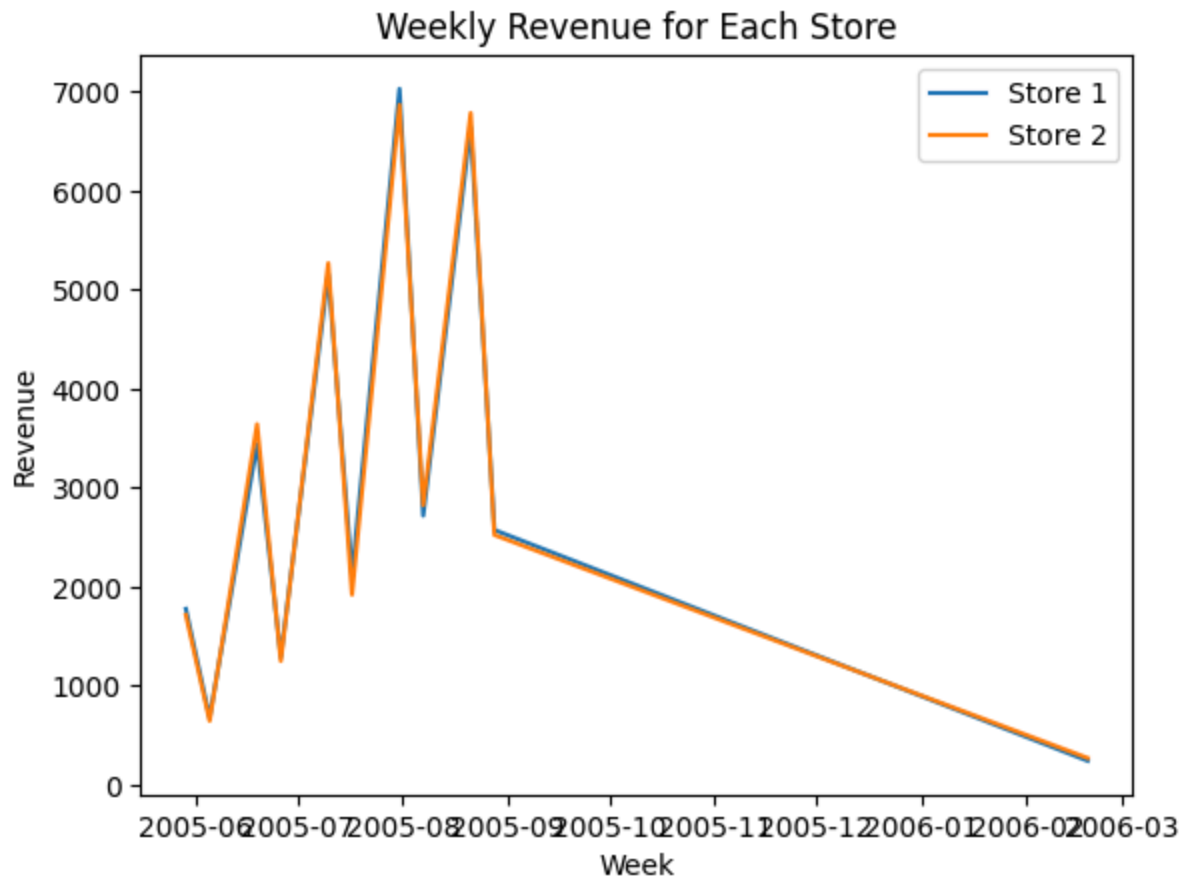
# Plotting the weekly revenue for each store
stores = df['store_id'].unique()
for store_id in stores:
    store_data = df[df['store_id'] == store_id]
    plt.plot(store_data['week_date'], store_data['weekly_revenue'], label=f'Store {

plt.title('Weekly Revenue for Each Store')
plt.xlabel('Week')
plt.ylabel('Revenue')
plt.legend()
plt.show()
df

```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_11288\3250296664.py:36: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
df = pd.read_sql_query(sql_query, conn)
```



Out[89]:

	store_id	week	year	weekly_revenue	week_date
0	1	21	2005	1777.87	2005-05-29
1	1	22	2005	682.37	2005-06-05
2	1	24	2005	3442.77	2005-06-19
3	1	25	2005	1294.01	2005-06-26
4	1	27	2005	5159.84	2005-07-10
5	1	28	2005	2123.09	2005-07-17
6	1	30	2005	7030.71	2005-07-31
7	1	31	2005	2719.64	2005-08-07
8	1	33	2005	6641.48	2005-08-21
9	1	34	2005	2574.86	2005-08-28
10	1	7	2006	243.10	2006-02-19
11	2	21	2005	1717.77	2005-05-29
12	2	22	2005	646.42	2005-06-05
13	2	24	2005	3643.17	2005-06-19
14	2	25	2005	1251.93	2005-06-26
15	2	27	2005	5270.19	2005-07-10
16	2	28	2005	1921.34	2005-07-17
17	2	30	2005	6868.72	2005-07-31
18	2	31	2005	2824.22	2005-08-07
19	2	33	2005	6787.04	2005-08-21
20	2	34	2005	2524.89	2005-08-28
21	2	7	2006	271.08	2006-02-19

Alur kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `payment`, yang berisi informasi tentang setiap transaksi pembayaran.
2. **JOIN Clauses:** Kueri menggunakan beberapa klausa JOIN untuk menggabungkan tabel-tabel lain yang terkait dengan transaksi pembayaran:
 - `JOIN rental ON payment.rental_id = rental.rental_id`: Menggabungkan tabel `rental`, yang berisi informasi tentang setiap transaksi penyewaan film, berdasarkan kolom `rental_id`.
 - `JOIN inventory ON rental.inventory_id = inventory.inventory_id`: Menggabungkan tabel `inventory`, yang berisi informasi tentang setiap barang

inventaris film yang tersedia untuk disewakan, berdasarkan kolom `inventory_id`.

- `JOIN store ON inventory.store_id = store.store_id` : Menggabungkan tabel `store`, yang berisi informasi tentang setiap toko yang menyediakan layanan sewa film, berdasarkan kolom `store_id`.

3. **SELECT Clause:** Kueri ini memilih beberapa kolom untuk ditampilkan dalam hasil kueri:

- `store.store_id` : Kolom ini menunjukkan ID dari setiap toko.
- `EXTRACT(WEEK FROM payment.payment_date) AS week` : Kolom ini mengekstrak informasi minggu (week) dari tanggal pembayaran.
- `EXTRACT(YEAR FROM payment.payment_date) AS year` : Kolom ini mengekstrak informasi tahun (year) dari tanggal pembayaran.
- `SUM(payment.amount) AS weekly_revenue` : Kolom ini menghitung total pendapatan mingguan (weekly revenue) untuk setiap toko.

4. **GROUP BY Clause:** Kueri menggunakan klausa GROUP BY untuk mengelompokkan hasil berdasarkan ID toko, tahun, dan minggu.

5. **ORDER BY Clause:** Hasil kueri akan diurutkan berdasarkan ID toko, tahun, dan minggu untuk menjaga urutan data yang benar.

6. **Manipulasi Data:** Setelah mendapatkan data, baris yang terkait dengan minggu dan tahun dikonversi menjadi tanggal dengan bantuan kolom 'week' dan 'year'. Ini membantu dalam visualisasi data.

Jadi, hasil kueri ini memberikan informasi tentang pendapatan mingguan untuk setiap toko dalam basis data Sakila, yang dapat digunakan untuk menganalisis kinerja penjualan setiap toko dari waktu ke waktu.

9. Lamanya Peminjaman Masing-masing Customer:

```
In [87]: import psycopg2
import pandas as pd

# Establishing a connection to the database
conn = psycopg2.connect(
    dbname="sakila",
    user="postgres",
    password="postgres",
    host="localhost",
    port="55432"
)

# SQL query to calculate the rental duration for each customer and group them
sql_query = """
SELECT
    customer_id,
    CASE
        WHEN rental_duration BETWEEN 2 AND 4 THEN '2 - 4 days'
        WHEN rental_duration BETWEEN 5 AND 7 THEN '5 - 7 days'
        WHEN rental_duration BETWEEN 8 AND 9 THEN '8 - 9 days'
        WHEN rental_duration BETWEEN 10 AND 14 THEN '10 - 14 days'
        ELSE 'More than 14 days'
    END AS rental_duration_group,
```

```

COUNT(*) AS customer_count
FROM
(
    SELECT
        rental.customer_id,
        EXTRACT(DAY FROM rental.return_date - rental.rental_date) AS rental
    FROM
        rental
) AS subquery
GROUP BY
    customer_id,
    rental_duration_group
ORDER BY
    customer_id,
    rental_duration_group;
"""

# Reading data into pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Closing the connection
conn.close()

# Displaying the DataFrame
df

```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_11288\3017912100.py:42: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
df = pd.read_sql_query(sql_query, conn)
```

Out[87]:

	customer_id	rental_duration_group	customer_count
0	1	2 - 4 days	11
1	1	5 - 7 days	7
2	1	8 - 9 days	6
3	1	More than 14 days	8
4	2	2 - 4 days	9
...
2381	598	8 - 9 days	5
2382	599	2 - 4 days	4
2383	599	5 - 7 days	8
2384	599	8 - 9 days	4
2385	599	More than 14 days	3

2386 rows × 3 columns

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `rental`, yang berisi informasi tentang setiap transaksi penyewaan film.
2. **Subquery:** Kueri ini menggunakan subquery untuk menghitung durasi penyewaan untuk setiap pelanggan. Subquery ini mengambil data dari tabel `rental` dan menghitung durasi penyewaan dengan mengurangi tanggal pengembalian (`return_date`) dari tanggal penyewaan (`rental_date`). Hasilnya adalah jumlah hari yang dibutuhkan pelanggan untuk mengembalikan film setelah menyewanya.
3. **SELECT Clause:** Kueri ini memilih beberapa kolom untuk ditampilkan dalam hasil kueri:
 - `customer_id`: ID pelanggan yang melakukan transaksi penyewaan.
 - `CASE ... END AS rental_duration_group`: Klausa CASE digunakan untuk mengelompokkan durasi penyewaan ke dalam kategori yang lebih bermakna. Misalnya, durasi penyewaan antara 2 dan 4 hari akan masuk ke dalam kategori '2 - 4 days'.
 - `COUNT(*) AS customer_count`: Menghitung jumlah pelanggan untuk setiap kategori durasi penyewaan.
4. **GROUP BY Clause:** Kueri ini menggunakan klausa GROUP BY untuk mengelompokkan hasil berdasarkan ID pelanggan dan kategori durasi penyewaan.
5. **ORDER BY Clause:** Hasil kueri akan diurutkan berdasarkan ID pelanggan dan kategori durasi penyewaan untuk menjaga urutan data yang benar.

Jadi, hasil kueri ini memberikan informasi tentang jumlah pelanggan yang melakukan transaksi penyewaan film dalam berbagai kategori durasi penyewaan. Kategori durasi penyewaan ini membantu untuk menganalisis kebiasaan penyewaan pelanggan dan pola peminjaman film.

10. Revenue Masing-masing Kategori Film:

In [103...

```
import psycopg2
import pandas as pd
import matplotlib.pyplot as plt

# Establishing a connection to the database
conn = psycopg2.connect(
    dbname="sakila",
    user="postgres",
    password="postgres",
    host="localhost",
    port="55432"
)

# SQL query to calculate revenue for each film category
sql_query = """
```

```

SELECT
    c.name AS category,
    SUM(p.amount) AS total_revenue
FROM
    film_category fc
JOIN
    film f ON fc.film_id = f.film_id
JOIN
    category c ON fc.category_id = c.category_id
JOIN
    inventory i ON f.film_id = i.film_id
JOIN
    rental r ON i.inventory_id = r.inventory_id
JOIN
    payment p ON r.rental_id = p.rental_id
GROUP BY
    c.name
ORDER BY
    total_revenue DESC;
"""

# Reading data into pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Closing the connection
conn.close()

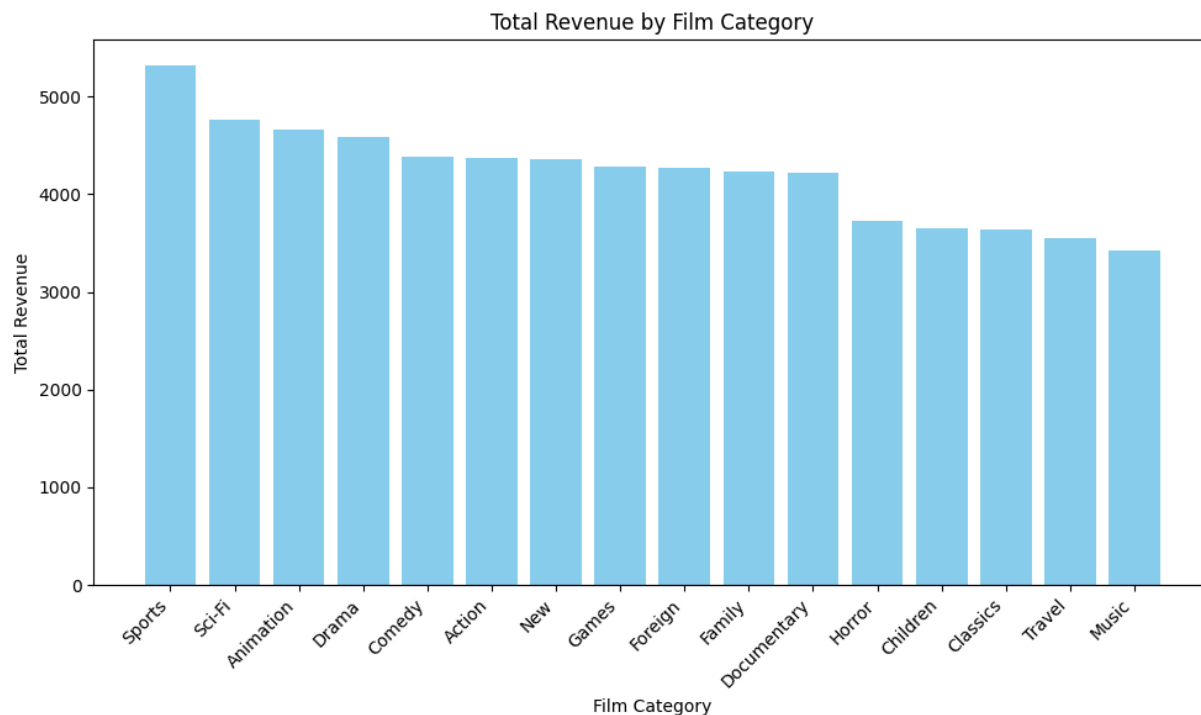
# Plotting the revenue for each film category
plt.figure(figsize=(10, 6))
plt.bar(df['category'], df['total_revenue'], color='skyblue')
plt.xlabel('Film Category')
plt.ylabel('Total Revenue')
plt.title('Total Revenue by Film Category')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

df

```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_11288\2306563887.py:38: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
df = pd.read_sql_query(sql_query, conn)
```



Out[103...

	category	total_revenue
0	Sports	5314.21
1	Sci-Fi	4756.98
2	Animation	4656.30
3	Drama	4587.39
4	Comedy	4383.58
5	Action	4375.85
6	New	4352.61
7	Games	4281.33
8	Foreign	4270.67
9	Family	4235.03
10	Documentary	4217.52
11	Horror	3722.54
12	Children	3655.55
13	Classics	3639.59
14	Travel	3549.64
15	Music	3417.72

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause:** Kueri dimulai dengan mengambil data dari tabel `film_category`, yang berisi informasi tentang kategori film. Tabel ini digabungkan dengan tabel `film` untuk mendapatkan informasi lebih lanjut tentang film-film dalam setiap kategori. Selanjutnya, tabel `category` digabungkan untuk mendapatkan nama kategori berdasarkan ID kategori.
2. **JOIN Clauses:** Kueri menggunakan beberapa klausa JOIN untuk menggabungkan tabel-tabel terkait:
 - `JOIN film ON film_category.film_id = film.film_id` : Menggabungkan tabel `film` dengan `film_category` berdasarkan ID film.
 - `JOIN category ON film_category.category_id = category.category_id` : Menggabungkan tabel `category` dengan `film_category` berdasarkan ID kategori.
 - `JOIN inventory ON film.film_id = inventory.film_id` : Menggabungkan tabel `inventory` dengan `film` untuk mendapatkan informasi tentang inventaris film.
 - `JOIN rental ON inventory.inventory_id = rental.inventory_id` : Menggabungkan tabel `rental` dengan `inventory` untuk mendapatkan informasi tentang transaksi penyewaan film.
 - `JOIN payment ON rental.rental_id = payment.rental_id` : Menggabungkan tabel `payment` dengan `rental` untuk mendapatkan informasi tentang transaksi pembayaran film.
3. **SELECT Clause:** Kueri ini memilih dua kolom untuk ditampilkan dalam hasil kueri:
 - `c.name AS category` : Kolom ini menunjukkan nama kategori film.
 - `SUM(p.amount) AS total_revenue` : Menghitung total pendapatan (`amount`) dari setiap transaksi pembayaran (`payment`) yang terkait dengan film-film dalam setiap kategori.
4. **GROUP BY Clause:** Kueri ini menggunakan klausa GROUP BY untuk mengelompokkan hasil berdasarkan nama kategori film (`c.name`). Ini memungkinkan kita untuk menghitung total pendapatan untuk setiap kategori film.
5. **ORDER BY Clause:** Hasil kueri akan diurutkan berdasarkan total pendapatan (`total_revenue`) secara menurun (`DESC`). Ini akan membantu dalam menampilkan kategori film dengan pendapatan tertinggi di bagian atas.

Jadi, hasil kueri ini memberikan informasi tentang total pendapatan yang dihasilkan oleh setiap kategori film, yang dapat digunakan untuk menganalisis performa relatif dari kategori film dalam basis data Sakila.