In [68]:
```
import atoti as tt
```

In [69]:
```
sessions = tt.Session()
```

```
Closing existing "Unnamed" session to create the new one.
```

In [70]:
```
dbname = "northwind"
uname = "postgres"
passwd = "postgres"
connSql = f"postgresql://localhost:55432/{dbname}?user={uname}&password={passwd}"
print(connSql)
```

```
postgresql://localhost:55432/northwind?user=postgres&password=postgres
```

Mencoba membaca beberapa kolom

In [71]:
```
data = sessions.read_sql(
        "select * from public.Categories;",
        url=connSql,
        table_name="Categories",
        keys={"category_id"},)
```

In [72]:
```
data.head()
```

Out[72]:

category_id	category_name	description	picture
5	Grains/Cereals	Breads, crackers, pasta, and cereal	
7	Produce	Dried fruit and bean curd	
3	Confections	Desserts, candies, and sweet breads	
1	Beverages	Soft drinks, coffees, teas, beers, and ales	
2	Condiments	Sweet and savory sauces, relishes, spreads, an...	

Import dan Membaca kolom database

In [73]:
```
import pandas as pd
from sqlalchemy import create_engine
import sqlite3
```

In [85]:
```
from sqlalchemy import create_engine, MetaData

# Membuat koneksi dengan database PostgreSQL
engine = create_engine(connSql)

# Membuat objek MetaData
metadata = MetaData()

# Memuat metadata dari database
metadata.reflect(bind=engine)
```

```
# Mendapatkan daftar semua tabel
tables = metadata.tables.keys()
tables
```

Out[85]:
```
dict_keys(['territories', 'region', 'order_details', 'orders', 'customers', 'employees', 'shippers', 'products', 'categories', 'suppliers', 'employee_territories', 'us_states', 'customer_demographics', 'customer_customer_demo'])
```

1. Sebutkan total jumlah penjualan per customer, per tahun, dan per kategori produk!

In [108…
```
import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan total penjualan per customer, per tahun, dan per
query = """
SELECT c.customer_id,
EXTRACT(YEAR FROM o.order_date) AS order_year,
p.category_id,
SUM(od.quantity * od.unit_price) AS total_sales
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN order_details od ON o.order_id = od.order_id
JOIN products p ON od.product_id = p.product_id
GROUP BY c.customer_id, order_year, p.category_id;
"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
rekap_customer = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
rekap_customer
```

Out[108…

	customer_id	order_year	category_id	total_sales
0	BERGS	1996.0	1	229.199993
1	KOENE	1997.0	4	4642.000000
2	ERNSH	1997.0	3	10736.199951
3	WELLI	1998.0	1	783.000000
4	ROMEY	1996.0	3	7.300000
...	...	...	...	...
1071	WANDK	1997.0	3	510.399998
1072	WHITC	1997.0	6	2136.999981
1073	SAVEA	1998.0	3	4636.650017
1074	FRANR	1998.0	1	360.000000
1075	FRANS	1998.0	8	500.000000

1076 rows × 4 columns

Alur kueri yang dilakukan melalui agregasi data penjualan dari tabel `orders`, `customers`, `order_details`, dan `products` berdasarkan customer, tahun pesanan (`order_year`), dan kategori produk (`category_id`). Kueri tersebut kemudian menghitung total penjualan dengan mengalikan jumlah produk yang dibeli (`od.quantity`) dengan harga satuan produk (`od.unit_price`), dan menjumlahkannya untuk setiap kelompok customer, tahun, dan kategori produk.

Berikut adalah alur kueri secara lebih detail:

1. Kueri menghubungkan tabel `orders`, `customers`, `order_details`, dan `products` menggunakan klausa JOIN untuk memungkinkan akses ke informasi yang diperlukan.
2. Kueri menggunakan klausa GROUP BY untuk mengelompokkan data berdasarkan `customer_id`, `order_year` (tahun dari tanggal pesanan), dan `category_id` (kategori produk).
3. Dalam klausa SELECT, kueri menggunakan fungsi `EXTRACT` untuk mengekstraksi tahun dari `order_date` dan menghitung total penjualan dengan mengalikan `quantity` dengan `unit_price` dari setiap produk (`SUM(od.quantity * od.unit_price)`).
4. Hasil kueri disimpan dalam DataFrame `rekap_customer` menggunakan `pd.read_sql_query`.

Jadi, kueri tersebut memberikan rangkuman (rekapitulasi) total penjualan per customer, per tahun, dan per kategori produk.

2. Sebutkan tiga employee dengan penjualan terbanyak!

In [113…

```
import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan tiga employee dengan penjualan terbanyak
query = """
SELECT CONCAT(e.first_name, ' ', e.last_name) AS full_name
FROM employees e
JOIN orders o ON e.employee_id = o.employee_id
JOIN order_details od ON o.order_id = od.order_id
GROUP BY e.employee_id, full_name
ORDER BY SUM(od.quantity * od.unit_price) DESC
LIMIT 3;
"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
tiga_terbanyak = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
tiga_terbanyak
```

Out[113…

	full_name
0	Margaret Peacock
1	Janet Leverling
2	Nancy Davolio

penjelasan alur dari kueri SQL:

1. **FROM clause**: Kueri dimulai dengan mengambil data dari tabel `employees`, `orders`, dan `order_details`. Ini dilakukan menggunakan klausa JOIN untuk menggabungkan data dari tabel-tabel ini berdasarkan relasi antara mereka. Misalnya, `JOIN orders o ON e.employee_id = o.employee_id` menggabungkan data dari tabel `employees` dan `orders` berdasarkan kolom `employee_id`.

2. **SELECT clause**: Kita memilih kolom `first_name` dan `last_name` dari tabel `employees`. Namun, karena kita ingin menampilkan nama lengkap, kita menggunakan fungsi `CONCAT` untuk menggabungkan nama depan dan belakang menjadi satu kolom bernama `full_name`. Ini dilakukan dengan menambahkan spasi di antara nama depan dan belakang menggunakan tanda kutip spasi: `CONCAT(e.first_name, ' ', e.last_name) AS full_name`.

3. **GROUP BY clause**: Karena kita menggunakan fungsi agregasi (`SUM`) dalam klausa SELECT, kita perlu mengelompokkan data berdasarkan kolom yang tidak diagregasi. Dalam hal ini, kita ingin mengelompokkan data berdasarkan `employee_id` (untuk memastikan setiap karyawan hanya muncul satu kali dalam hasil) dan `full_name` (nama lengkap).

4. **ORDER BY clause**: Kita ingin mengurutkan hasil berdasarkan total penjualan dari yang tertinggi ke yang terendah. Oleh karena itu, kita menggunakan klausa `ORDER BY SUM(od.quantity * od.unit_price) DESC` untuk mengurutkan hasil berdasarkan jumlah penjualan secara menurun (`DESC`).

5. **LIMIT clause**: Akhirnya, kita hanya ingin menampilkan tiga baris pertama dari hasil (yaitu, tiga karyawan dengan penjualan terbanyak). Kita mencapai ini dengan menggunakan klausa `LIMIT 3`.

Dengan demikian, kueri tersebut menghasilkan daftar tiga karyawan dengan penjualan terbanyak, diurutkan berdasarkan total penjualan mereka, dan hanya menampilkan nama lengkap mereka.

3. Sebutkan employee dengan penjualan terbanyak per produk dan per tahun!

In [115…

```
 import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
```

```
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan employee dengan penjualan terbanyak per produk d
query = """
SELECT
    CONCAT(e.first_name, ' ', e.last_name) AS full_name,
    EXTRACT(YEAR FROM o.order_date) AS order_year,
    p.product_id,
    p.product_name,
    SUM(od.quantity * od.unit_price) AS total_sales
FROM
    employees e
JOIN orders o ON e.employee_id = o.employee_id
JOIN order_details od ON o.order_id = od.order_id
JOIN products p ON od.product_id = p.product_id
GROUP BY
    full_name,
    order_year,
    p.product_id,
    p.product_name
ORDER BY
    order_year,
    p.product_id,
    total_sales DESC;

"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
terbanyak_employe = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
terbanyak_employe
```

Out[115...

	full_name	order_year	product_id	product_name	total_sales
0	Nancy Davolio	1996.0	1	Chai	647.999983
1	Michael Suyama	1996.0	1	Chai	503.999987
2	Margaret Peacock	1996.0	1	Chai	475.199987
3	Laura Callahan	1996.0	1	Chai	172.799995
4	Nancy Davolio	1996.0	2	Chang	1139.999986
...	...	...	...	...	...
1148	Nancy Davolio	1998.0	77	Original Frankfurter grüne Soße	1092.000000
1149	Michael Suyama	1998.0	77	Original Frankfurter grüne Soße	793.000000
1150	Andrew Fuller	1998.0	77	Original Frankfurter grüne Soße	754.000000
1151	Steven Buchanan	1998.0	77	Original Frankfurter grüne Soße	650.000000
1152	Laura Callahan	1998.0	77	Original Frankfurter grüne Soße	559.000000

1153 rows × 5 columns

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause**: Kueri dimulai dengan mengambil data dari tabel `employees`, `orders`, `order_details`, dan `products`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN orders o ON e.employee_id = o.employee_id` menggabungkan tabel `employees` dengan `orders` berdasarkan kolom `employee_id`.

2. **SELECT Clause**: Kita memilih beberapa kolom untuk ditampilkan dalam hasil kueri:

   - `CONCAT(e.first_name, ' ', e.last_name) AS full_name`: Ini menggabungkan kolom `first_name` dan `last_name` dari tabel `employees` menjadi satu kolom bernama `full_name`.
   - `EXTRACT(YEAR FROM o.order_date) AS order_year`: Ini mengekstrak tahun dari kolom `order_date` dari tabel `orders` dan memberikan alias `order_year`.
   - `p.product_id` dan `p.product_name`: Kolom ini mengidentifikasi produk yang dijual.
   - `SUM(od.quantity * od.unit_price) AS total_sales`: Ini menghitung total penjualan untuk setiap kombinasi employee, tahun, dan produk dengan

mengalikan jumlah barang yang dibeli (`od.quantity`) dengan harga satuan barang (`od.unit_price`) untuk setiap pesanan dan menjumlahkannya.

3. **GROUP BY Clause**: Karena kita menggunakan fungsi agregat (`SUM`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom-kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan `full_name` , `order_year` , `p.product_id` , dan `p.product_name` .

4. **ORDER BY Clause**: Hasil kueri akan diurutkan berdasarkan `order_year` , `p.product_id` , dan total penjualan (`total_sales`) secara menurun (`DESC`), sehingga memberikan kita employee dengan penjualan terbanyak per produk dan per tahun.

Jadi, hasil kueri ini akan memberikan daftar employee dengan penjualan terbanyak untuk setiap produk dan tahun yang ada dalam basis data.

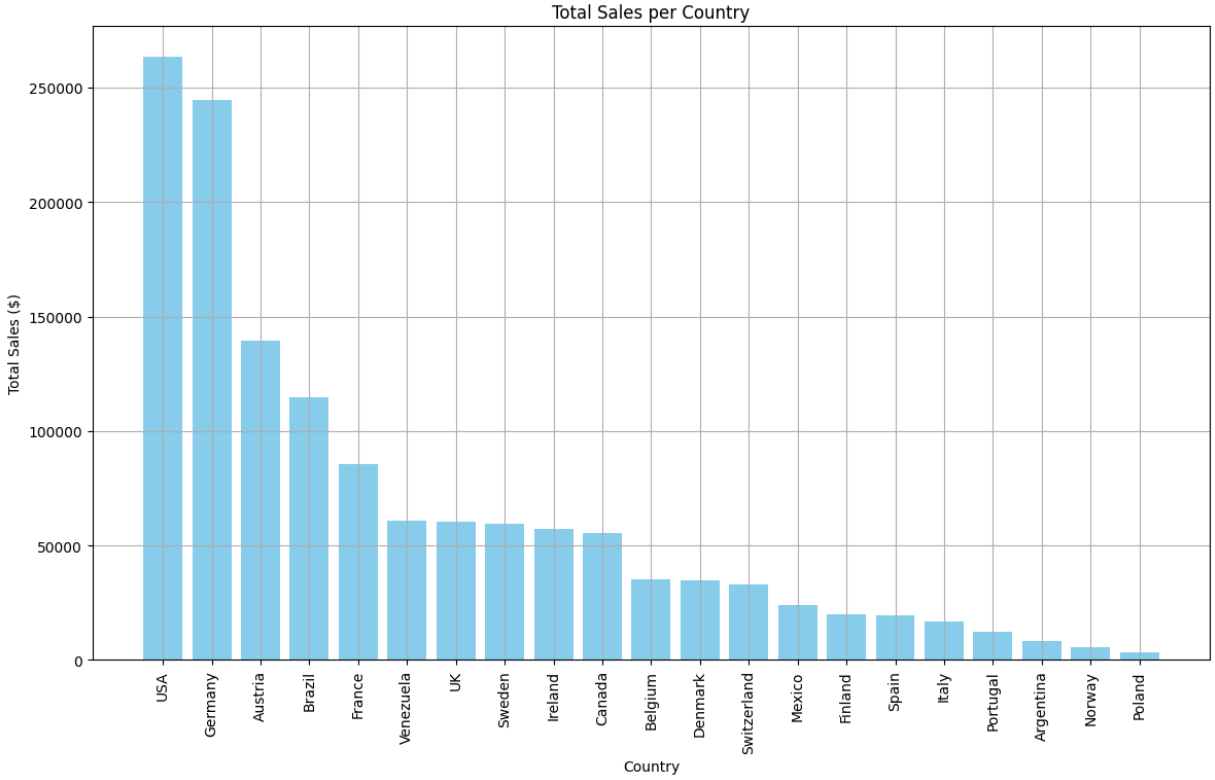
4. Sebutkan negara dengan penjualan terbanyak! Visualisasikan!

In [97]:
```
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
# Konfigurasi koneksi ke database PostgreSQL
dbname = "northwind"
uname = "postgres"
passwd = "postgres"
host = "localhost"
port = "55432"
# Membuat string koneksi
connSql = f"postgresql://{uname}:{passwd}@{host}:{port}/{dbname}"
print(connSql)
# Membuat engine SQLAlchemy
engine = create_engine(connSql)
# Mendefinisikan query untuk mendapatkan total penjualan per negara
query = """
SELECT
c.country,
SUM(od.unit_price * od.quantity) AS total_sales
FROM
orders o
JOIN
customers c ON o.customer_id = c.customer_id
JOIN
order_details od ON o.order_id = od.order_id
GROUP BY
c.country
ORDER BY
total_sales DESC;
"""
# Membaca hasil query ke dalam DataFrame
# Membaca hasil query ke dalam DataFrame
data = pd.read_sql(query, engine)

# Visualisasi data
```

```
plt.figure(figsize=(14, 8))
plt.bar(df['country'], df['total_sales'], color='skyblue')
plt.title('Total Sales per Country')
plt.xlabel('Country')
plt.ylabel('Total Sales ($)')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
# Menampilkan DataFrame
data
```

```
postgresql://postgres:postgres@localhost:55432/northwind
```



Out[97]:

	country	total_sales
0	USA	263566.980017
1	Germany	244640.629969
2	Austria	139496.629867
3	Brazil	114968.480040
4	France	85498.760092
5	Venezuela	60814.889832
6	UK	60616.509948
7	Sweden	59523.699950
8	Ireland	57317.390162
9	Canada	55334.100187
10	Belgium	35134.980321
11	Denmark	34782.250007
12	Switzerland	32919.499998
13	Mexico	24073.449984
14	Finland	19778.450103
15	Spain	19431.889992
16	Italy	16705.149927
17	Portugal	12468.650044
18	Argentina	8119.099991
19	Norway	5735.150015
20	Poland	3531.949996

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause**: Kueri dimulai dengan mengambil data dari tabel `orders`, `customers`, dan `order_details`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN customers c ON o.customer_id = c.customer_id` menggabungkan tabel `orders` dengan `customers` berdasarkan kolom `customer_id`.

2. **SELECT Clause**: Kita memilih dua kolom untuk ditampilkan dalam hasil kueri:

   - `c.country`: Kolom ini menunjukkan negara tempat pelanggan berada.
   - `SUM(od.unit_price * od.quantity) AS total_sales`: Ini menghitung total penjualan untuk setiap negara dengan mengalikan harga satuan barang

(`od.unit_price`) dengan jumlah barang yang dibeli (`od.quantity`) untuk setiap pesanan dan menjumlahkannya.

3. **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 `c.country` , yaitu negara.

4. **ORDER BY Clause**: Hasil kueri akan diurutkan berdasarkan total penjualan (`total_sales`) secara menurun (`DESC`), sehingga memberikan kita total penjualan per negara yang terbesar.

Jadi, hasil kueri ini akan memberikan total penjualan per negara, yang kemudian diurutkan dari yang tertinggi ke terendah. Dalam kode yang diberikan, hasil kueri ini kemudian disajikan dalam bentuk bar chart menggunakan Matplotlib.

5. Sebutkan total penjualan dan rata-rata penjualan bulanan berdasarkan employee dan tahun!

In [121…

```
import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan total penjualan dan rata-rata penjualan bulanan
query = """
SELECT
CONCAT(e.first_name, ' ', e.last_name) AS full_name,
EXTRACT(YEAR FROM o.order_date) AS order_year,
EXTRACT(MONTH FROM o.order_date) AS order_month,
SUM(od.quantity * od.unit_price) AS total_sales,
AVG(od.quantity * od.unit_price) AS average_monthly_sales
FROM employees e
JOIN orders o ON e.employee_id = o.employee_id
JOIN order_details od ON o.order_id = od.order_id
GROUP BY e.employee_id, e.first_name, e.last_name, order_year, order_month
ORDER BY e.employee_id, order_year, order_month;
"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
penjualan_employe = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
penjualan_employe
```

Out[121...

	full_name	order_year	order_month	total_sales	average_monthly_sales
0	Nancy Davolio	1996.0	7.0	2018.599993	672.866664
1	Nancy Davolio	1996.0	8.0	6007.100028	500.591669
2	Nancy Davolio	1996.0	9.0	6883.700181	529.515399
3	Nancy Davolio	1996.0	10.0	4061.399942	507.674993
4	Nancy Davolio	1996.0	11.0	10261.200030	932.836366
...	...	...	...	...	...
187	Anne Dodsworth	1997.0	12.0	1941.500000	388.300000
188	Anne Dodsworth	1998.0	1.0	5627.139982	432.856922
189	Anne Dodsworth	1998.0	2.0	19325.510010	1756.864546
190	Anne Dodsworth	1998.0	3.0	7566.599998	630.550000
191	Anne Dodsworth	1998.0	4.0	9501.499993	950.149999

192 rows × 5 columns

Alur dari kueri SQL sebagai berikut:

1. **FROM Clause**: Kueri dimulai dengan mengambil data dari tabel `employees`, `orders`, dan `order_details`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN orders o ON e.employee_id = o.employee_id` menggabungkan tabel `employees` dengan `orders` berdasarkan kolom `employee_id`.

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

   - `CONCAT(e.first_name, ' ', e.last_name) AS full_name`: Menggabungkan kolom `first_name` dan `last_name` dari tabel `employees` menjadi satu kolom bernama `full_name`.
   - `EXTRACT(YEAR FROM o.order_date) AS order_year`: Mengekstrak tahun dari kolom `order_date` dari tabel `orders` dan memberikan alias `order_year`.
   - `EXTRACT(MONTH FROM o.order_date) AS order_month`: Mengekstrak bulan dari kolom `order_date` dari tabel `orders` dan memberikan alias `order_month`.
   - `SUM(od.quantity * od.unit_price) AS total_sales`: Menghitung total penjualan untuk setiap employee, tahun, dan bulan dengan mengalikan harga satuan barang (`od.unit_price`) dengan jumlah barang yang dibeli (`od.quantity`) untuk setiap pesanan dan menjumlahkannya.
   - `AVG(od.quantity * od.unit_price) AS average_monthly_sales`: Menghitung rata-rata penjualan bulanan untuk setiap employee, tahun, dan bulan

dengan menghitung rata-rata dari hasil perkalian harga satuan barang dengan jumlah barang yang dibeli.

3. **GROUP BY Clause**: Karena kita menggunakan fungsi agregat (`SUM` dan `AVG`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan `e.employee_id` , `e.first_name` , `e.last_name` , `order_year` , dan `order_month` .

4. **ORDER BY Clause**: Hasil kueri akan diurutkan berdasarkan `e.employee_id` , `order_year` , dan `order_month` .

Jadi, hasil kueri ini akan memberikan total penjualan dan rata-rata penjualan bulanan untuk setiap employee, tahun, dan bulan yang ada dalam basis data.

6. Sebutkan total penjualan dan diskon per produk dan per bulan!

In [127…

```
 import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan total penjualan dan diskon per produk dan per bu
query = """
SELECT
    p.product_name,
    EXTRACT(MONTH FROM o.order_date) AS order_month,
    SUM(od.quantity * od.unit_price) AS total_sales,
    SUM(od.quantity * od.discount) AS total_discount
FROM
    orders o
JOIN
    order_details od ON o.order_id = od.order_id
JOIN
    products p ON od.product_id = p.product_id
GROUP BY
    p.product_name,
    order_month
ORDER BY
    order_month,
    p.product_name;
"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
produk = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
produk
```

Out[127...

	product_name	order_month	total_sales	total_discount
0	Alice Mutton	1.0	4898.400074	21.50
1	Aniseed Syrup	1.0	1190.000000	0.00
2	Boston Crab Meat	1.0	2880.399946	21.70
3	Camembert Pierrot	1.0	6936.000046	12.00
4	Carnarvon Tigers	1.0	3437.500000	6.25
...	...	...	...	...
771	Uncle Bob's Organic Dried Pears	12.0	1380.000000	2.45
772	Valkoinen suklaa	12.0	195.000000	0.00
773	Vegie-spread	12.0	1878.600021	1.60
774	Wimmers gute Semmelknödel	12.0	1961.750011	7.00
775	Zaanse koeken	12.0	285.000000	0.00

776 rows × 4 columns

Alur kueri SQL sebagai berikut:

1. **FROM Clause**: Kueri dimulai dengan mengambil data dari tabel `orders`, `order_details`, dan `products`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN order_details od ON o.order_id = od.order_id` menggabungkan tabel `orders` dengan `order_details` berdasarkan kolom `order_id`.

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

   - `p.product_name`: Kolom ini menunjukkan nama produk.
   - `EXTRACT(MONTH FROM o.order_date) AS order_month`: Mengambil bulan dari tanggal pesanan menggunakan fungsi `EXTRACT` dan memberikan alias `order_month`.
   - `SUM(od.quantity * od.unit_price) AS total_sales`: Menghitung total penjualan untuk setiap produk dan bulan dengan mengalikan harga satuan barang (`od.unit_price`) dengan jumlah barang yang dibeli (`od.quantity`) untuk setiap pesanan dan menjumlahkannya.
   - `SUM(od.quantity * od.discount) AS total_discount`: Menghitung total diskon yang diberikan untuk setiap produk dan bulan dengan mengalikan jumlah barang yang dibeli dengan diskon yang diberikan (`od.discount`) untuk setiap pesanan dan menjumlahkannya.

3. **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 `p.product_name` (nama produk) dan `order_month` (bulan dari tanggal pesanan).

4. **ORDER BY Clause**: Hasil kueri akan diurutkan berdasarkan bulan (`order_month`) dan nama produk (`p.product_name`).

Jadi, hasil kueri ini akan memberikan total penjualan dan diskon per produk per bulan, diurutkan berdasarkan bulan dan nama produk.

7. Sebutkan total penjualan, banyaknya produk, dan jumlah produk terjual untuk masing-masing pemesanan (order)?

In [128…

```
import pandas as pd
from sqlalchemy import create_engine, text
# Buat koneksi dengan database PostgreSQL
engine = create_engine(connSql)
# Buat kueri SQL untuk mendapatkan total penjualan, banyaknya produk, dan jumlah pr
query = """
SELECT
o.order_id,
COUNT(od.product_id) AS total_products_ordered,
SUM(od.quantity) AS total_products_sold,
SUM(od.quantity * od.unit_price) AS total_sales
FROM
orders o
JOIN
order_details od ON o.order_id = od.order_id
GROUP BY
o.order_id
ORDER BY
o.order_id;
"""
# Jalankan kueri dan simpan hasilnya ke dalam DataFrame
rekap_penjualan = pd.read_sql_query(query, engine)
# Tampilkan DataFrame
rekap_penjualan
```

Out[128...

	order_id	total_products_ordered	total_products_sold	total_sales
0	10248	3	27	439.999998
1	10249	2	49	1863.400064
2	10250	3	60	1813.000040
3	10251	3	41	670.799986
4	10252	3	105	3730.000153
...	...	...	...	...
825	11073	2	30	300.000000
826	11074	1	14	244.300011
827	11075	3	42	586.000000
828	11076	3	50	1056.999998
829	11077	25	72	1374.600000

830 rows × 4 columns

Alur kueri SQL sebagai berikut:

1. **FROM Clause**: Kueri dimulai dengan mengambil data dari tabel `orders` dan `order_details`. Ini dilakukan dengan menggabungkan tabel-tabel ini menggunakan klausa JOIN berdasarkan kunci hubung yang sesuai. Misalnya, `JOIN order_details od ON o.order_id = od.order_id` menggabungkan tabel `orders` dengan `order_details` berdasarkan kolom `order_id`.

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

   - `o.order_id`: Kolom ini menunjukkan ID pemesanan.
   - `COUNT(od.product_id) AS total_products_ordered`: Menghitung jumlah produk yang dipesan dalam setiap pemesanan dengan menghitung jumlah entri dalam kolom `product_id` dari tabel `order_details`.
   - `SUM(od.quantity) AS total_products_sold`: Menghitung jumlah produk yang terjual dalam setiap pemesanan dengan menjumlahkan nilai dalam kolom `quantity` dari tabel `order_details`.
   - `SUM(od.quantity * od.unit_price) AS total_sales`: Menghitung total penjualan dalam setiap pemesanan dengan mengalikan jumlah produk yang terjual (`od.quantity`) dengan harga satuan produk (`od.unit_price`) dari tabel `order_details` dan menjumlahkannya.

3. **GROUP BY Clause**: Karena kita menggunakan fungsi agregat (`COUNT` dan `SUM`) dalam klausa SELECT, kita perlu mengelompokkan data dengan kolom yang tidak diagregasi. Kita mengelompokkan data berdasarkan `o.order_id` (ID pemesanan).

4. **ORDER BY Clause**: Hasil kueri akan diurutkan berdasarkan `o.order_id` (ID pemesanan).

Jadi, hasil kueri ini akan memberikan total penjualan, banyaknya produk, dan jumlah produk terjual untuk setiap pemesanan, diurutkan berdasarkan ID pemesanan.