

OLTP DATABASE PROGRESS

Rabu, 10 Februari 2021

1. Coba untuk familiarisasi dengan dbeaver, coba untuk membuat database baru di dalam PostgreSQL. Coba familiarisasi dengan koneksi di dbeaver.
2. Masih gagal untuk load data di DBeaver, menggunakan SQL Server untuk sementara untuk melihat data

Kamis, 11 Februari 2021

1. Mulai bisa load data ke dalam dbeaver.
2. Mencoba untuk membangun erd oltp.
3. Belajar untuk membackup dan merestore data ke dalam dbeaver.

Jumat, 12 Februari 2021

1. Mulai membaca data
 - a. Feedback Dataset
Dari feedback_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu feedback_id.

Query	Hasil
select count(feedback_id) from feedback_dataset	100000
select count(distinct feedback_id) from feedback_dataset	99173
select count(distinct order_id) from feedback_dataset	99441
select count(*) from feedback_dataset	100000
select feedback_id , order_id , count(feedback_id) from feedback_dataset group by feedback_id, order_id having count(feedback_id) > 1	No value

order by feedback_id	
-------------------------	--

feedback_id dan order_id composite dan merupakan primary key

b. Order Dataset

Dari order dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu order_id.

Query	Hasil
select count(order_id) from order_dataset	99441
select count(distinct order_id) from order_dataset	99441
select count(*) from order_dataset	99441

order_id merupakan primary key

c. Order Item Dataset

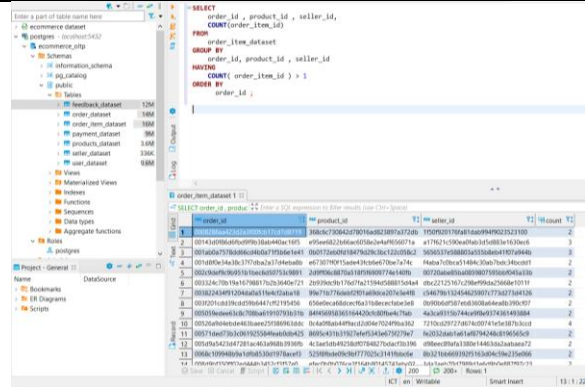
Dari order_item_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu order_item_id.

Query	Hasil
select count(order_item_id) from order_item_dataset	112650
select count(distinct order_item_id) from order_item_dataset	21
select count(*) from order_item_dataset	112650

```

select
    order_id , product_id ,
seller_id,
    count(order_item_id)
from
    order_item_dataset
group by
    order_id, product_id ,
seller_id
having
    count( order_item_id ) >
1
order by
    order_id ;

```



order_id	product_id	seller_id	T2
1	1	1	2
2	1	1	2
3	1	1	2
4	1	1	2
5	1	1	2
6	1	1	2
7	1	1	2
8	1	1	2
9	1	1	2
10	1	1	2
11	1	1	2
12	1	1	2
13	1	1	2

```

select
    order_id , product_id ,
seller_id, order_item_id,
    count(order_item_id)
from
    order_item_dataset
group by
    order_id, product_id ,
seller_id, order_item_id
having
    count( order_item_id ) >
1
order by
    order_id ;

```

no value

```

select
    order_id , product_id ,
seller_id, order_item_id,
    count(order_item_id)
from
    order_item_dataset
group by
    order_id, product_id ,
seller_id, order_item_id
having
    count( order_item_id ) >
1
order by
    order_id ;

```

no value

```

select
    order_id ,
order_item_id,
    count(order_item_id)
from
    order_item_dataset
group by

```

no value

```

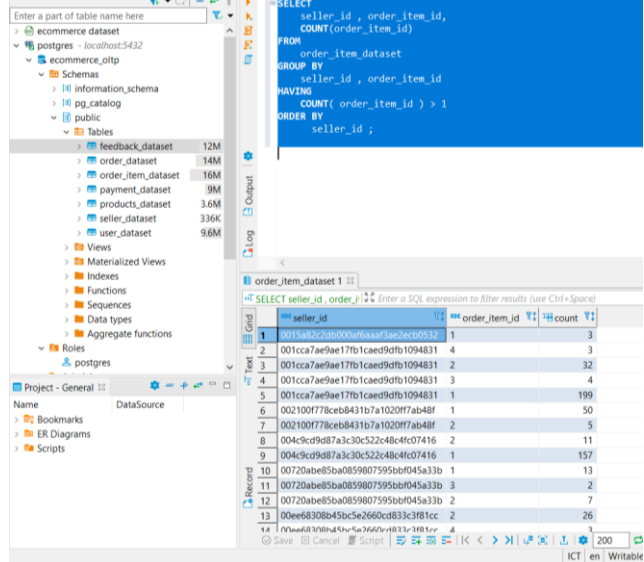
order_id, order_item_id
having
count( order_item_id ) >
1
order by
order_id ;

```

```

select
seller_id ,
order_item_id,
count(order_item_id)
from
order_item_dataset
group by
seller_id , order_item_id
having
count( order_item_id ) >
1
order by
seller_id ;

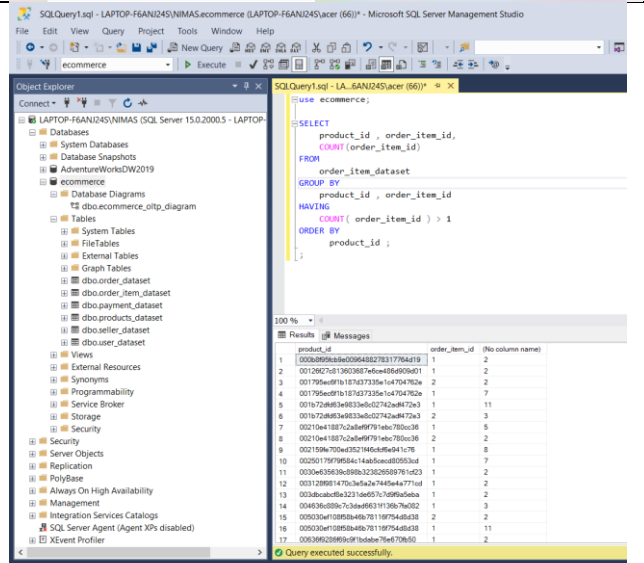
```



```

select
product_id ,
order_item_id,
count(order_item_id)
from
order_item_dataset
group by
product_id ,
order_item_id
having
count( order_item_id ) >
1
order by
product_id ;

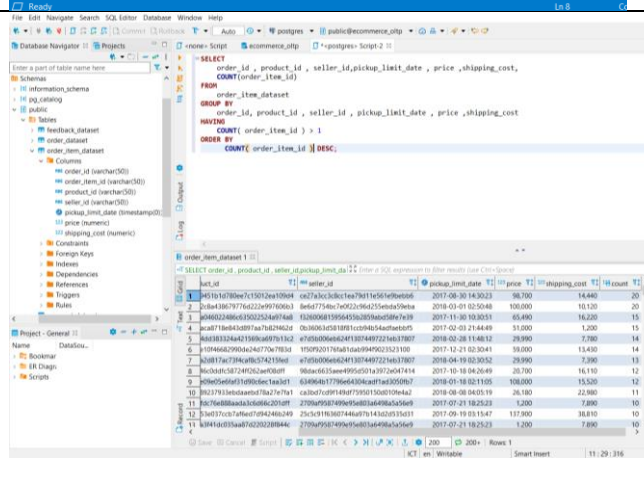
```



```

select
order_id , product_id ,
seller_id,pickup_limit_date
, price ,shipping_cost,
count(order_item_id)
from
order_item_dataset
group by
order_id, product_id ,
seller_id ,
pickup_limit_date , price
,shipping_cost
having

```



count(order_item_id) > 1 order by count(order_item_id) desc;	
--	--

order_item_id dan order_id composite dan merupakan primary key

d. Payment Dataset

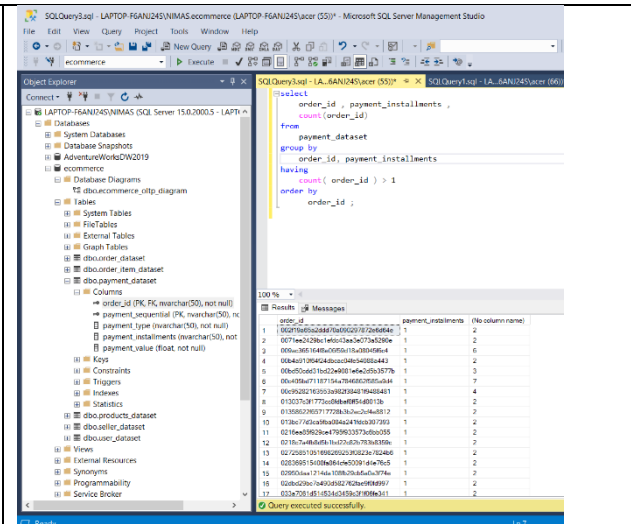
Dari payment_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu order_id.

Query	Hasil
select count(*) from payment_dataset	103886
select count(order_id) from payment_dataset	103886
select count(distinct order_id) from payment_dataset	99440
select order_id , payment_sequential , count(order_id) from payment_dataset group by order_id, payment_sequential having count(order_id) > 1 order by order_id ;	no value

```

select
    order_id ,
    payment_installments ,
    count(order_id)
from
    payment_dataset
group by
    order_id,
    payment_installments
having
    count( order_id ) > 1
order by
    order_id ;

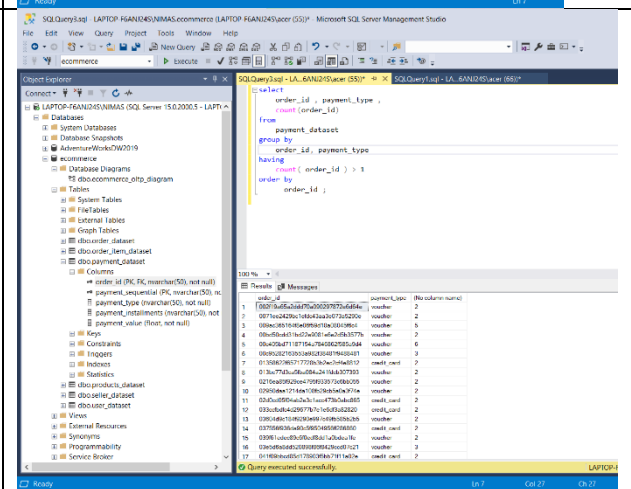
```



```

select
    order_id , payment_type ,
    count(order_id)
from
    payment_dataset
group by
    order_id, payment_type
having
    count( order_id ) > 1
order by
    order_id ;

```



Payment_sequential dan order_id composite dan merupakan primary key

e. Product Dataset

Dari products_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu product_id.

Query	Hasil
<pre> select count(product_id) from products_dataset </pre>	32951
<pre> select count(distinct product_id) from products_dataset </pre>	32951
<pre> select count(*) from products_dataset </pre>	32951

product_id merupakan primary key

f. Seller Dataset

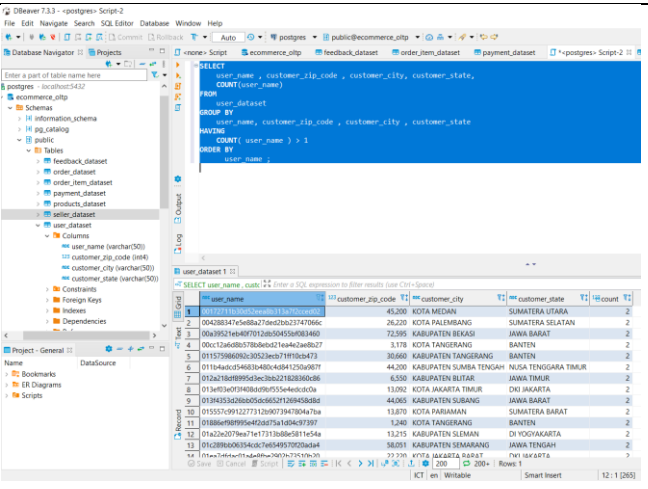
Dari seller_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu seller_id.

Query	Hasil
select count(seller_id) from seller_dataset	3095
select count(distinct seller_id) from seller_dataset	3095
select count(*) from seller_dataset	3095

Seller_id merupakan primary key

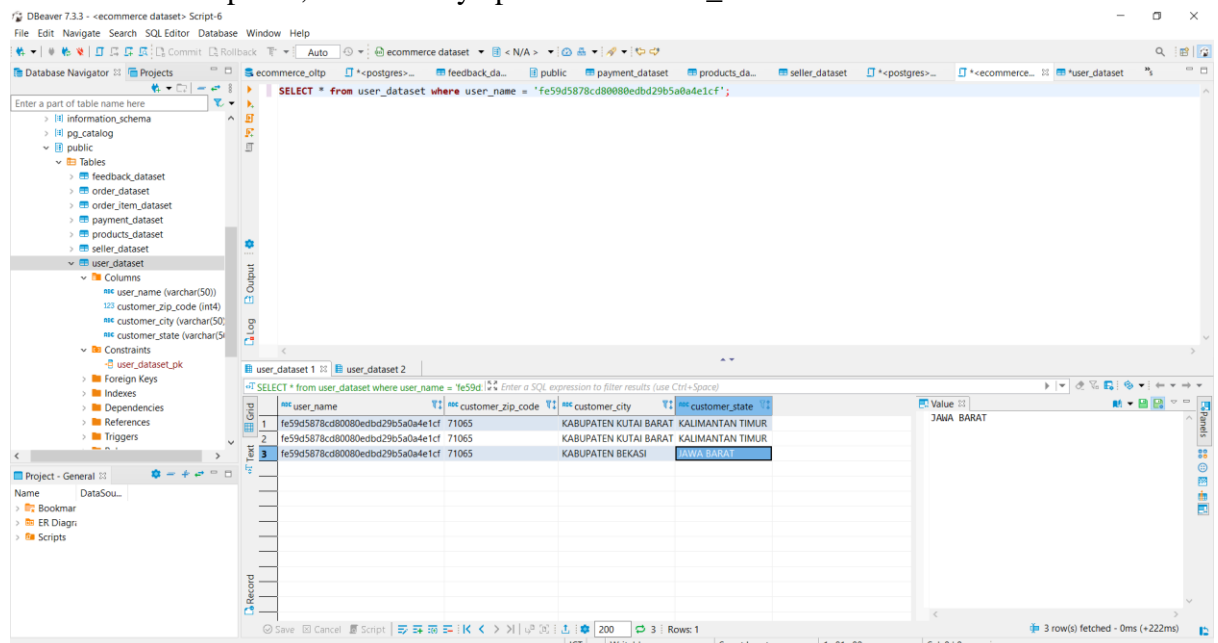
g. User Dataset

Dari user_dataset, saya ingin melihat kolom yang saya asumsikan adalah unique row, yaitu user_id.

Query	Hasil
select count(user_name) from user_dataset	99441
select count(distinct user_name) from user_dataset	96096
select user_name , customer_zip_code , customer_city, customer_state, count(user_name) from user_dataset group by user_name, customer_zip_code , customer_city , customer_state having count(user_name) > 1 order by user_name	

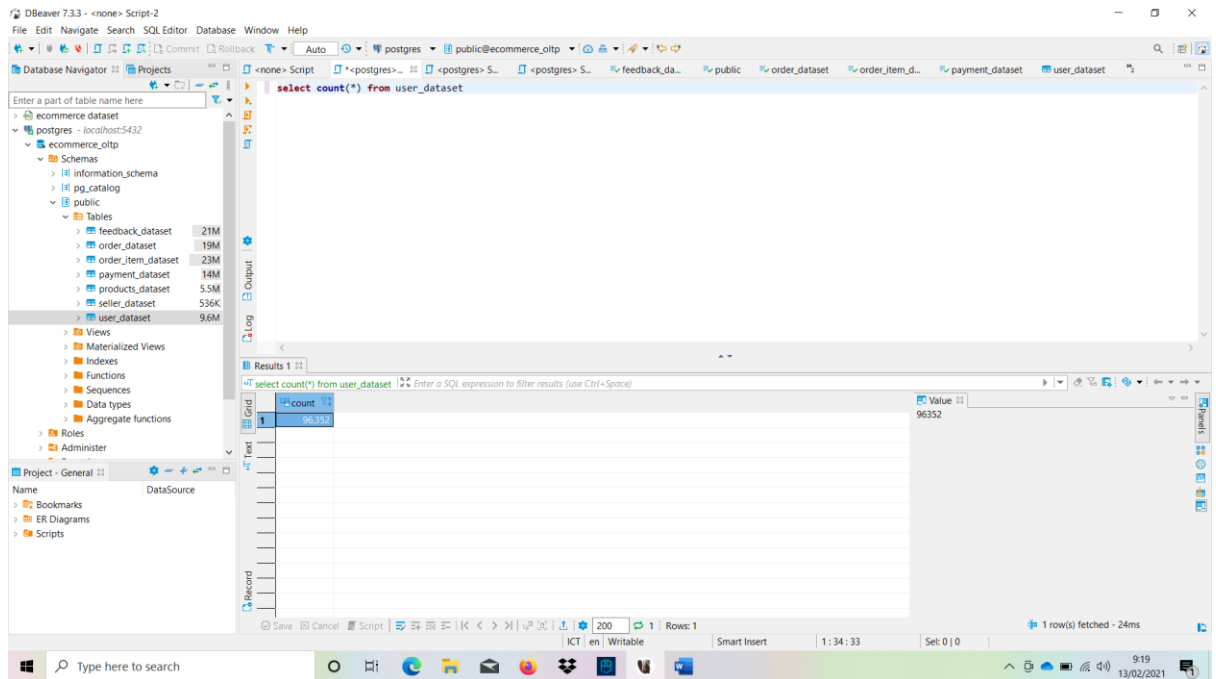
2. Cleaning Data

Dataset yang perlu dilakukan cleaning adalah user_dataset. Terdapat user_name yang memiliki data duplikat, salah satunya pada kolom user_name berikut.

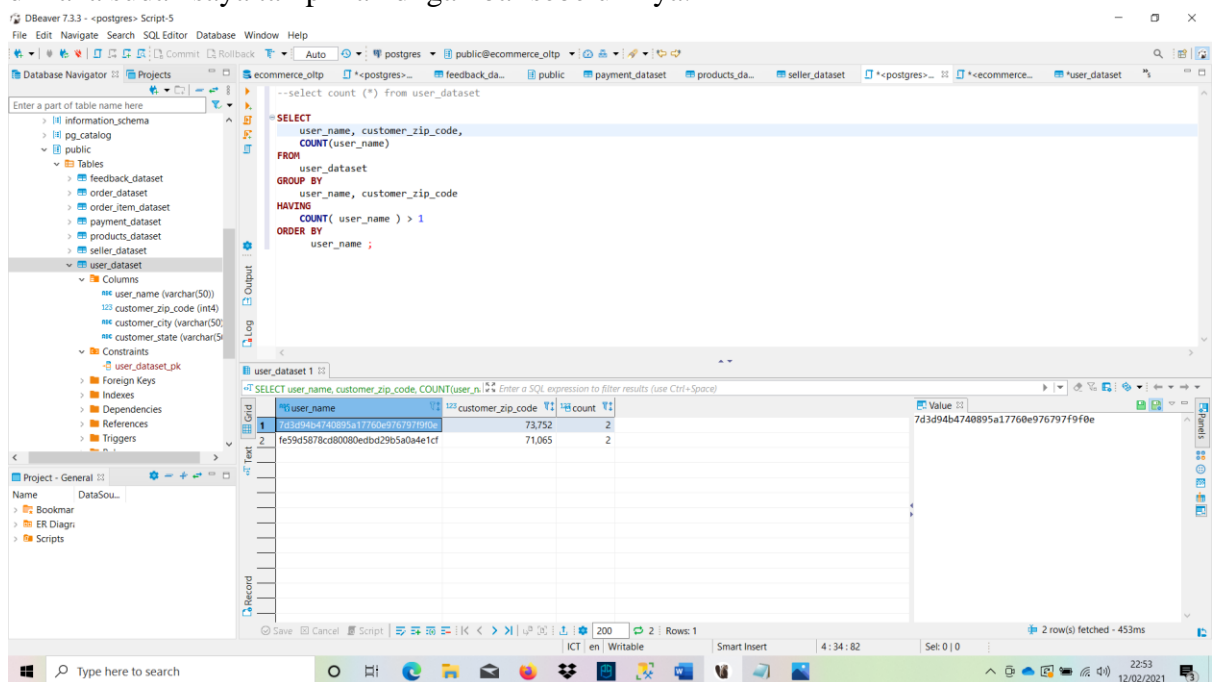


Dalam gambar tersebut, terdapat user_name yang memiliki customer_city dan customer_state yang berbeda, meskipun memiliki customer_zip_code yang berbeda. Karena hal tersebut, user_name tidak dapat menjadi primary key karena tidak unique. Ekspektasi saya di awal adalah terdapat satu user_name yang memiliki 2 alamat. Karena tidak ada alamat, maka acuannya bisa menggunakan kode pos atau customer_zip_code. Namun dilihat pada gambar diatas, tidak mungkin suatu lokasi yang berbeda jauh bisa memiliki customer_zip_code yang sama. Jika saya menganggap bahwa primary key adalah user_name dan customer_zip_code, tentu tidak akan relevan dengan data tersebut. Saya kemudian berpikir bahwa bisa jadi satu orang, memiliki 2 alamat, dengan kode pos yang sama. Namun tentu dengan syarat berada di city dan state yang sama, dan alamat yang berbeda. Karena di data ini tidak terdapat alamat, maka hal yang unique adalah user_name atau customer_zip_code atau gabungan keduanya. Sehingga awalnya saya melakukan cleaning dimana menghapus semua data yang redundan saja hingga mendapat 96352 record dengan menggunakan query berikut:

```
delete from user_dataset t1
using user_dataset t2
where t1.ctid < t2.ctid
and t1.user_name = t2.user_name
and t1.customer_zip_code = t2.customer_zip_code
and t1.customer_city = t2.customer_city
and t1.customer_state = t2.customer_state ;
```

Namun ternyata, masih tetap ada user_name yang memiliki data yang cukup aneh, dimana sudah saya tampilkan di gambar sebelumnya.



Akhirnya karena pertimbangan dari table order_dataset hanya memiliki user_name, maka user_name sendiri harus unique. Sehingga saya membuat user_name ini unique dengan hanya mengambil satu record yang paling awal. Tidak ada alasan khusus karena di table user_dataset sendiri tidak ada timestamp terkait alamat yang baru. Sehingga diperoleh hanya satu user_name. Total record menjadi 96096 dan user_name menjadi unique. Berikut query untuk membersihkannya (di query menggunakan SQL Server):

USE ecommerce;

WITH cte AS (

```

SELECT
    *,
    ROW_NUMBER() OVER (
        PARTITION BY
            user_name
        ORDER BY
            user_name
    ) row_num
FROM
    user_dataset
)
DELETE FROM cte
WHERE row_num > 1;

```

3. Assign Primary

Untuk assign primary key, saya langsung assign di GUI Dbeaver.

Tabel	Primary Key
feedback_dataset	feedback_id, order_id
order_dataset	order_id
order_item_dataset	order_id, order_item_id
payment_dataset	order_id, payment_sequential
products_dataset	product_id
seller_dataset	seller_id
user_dataset	user_name

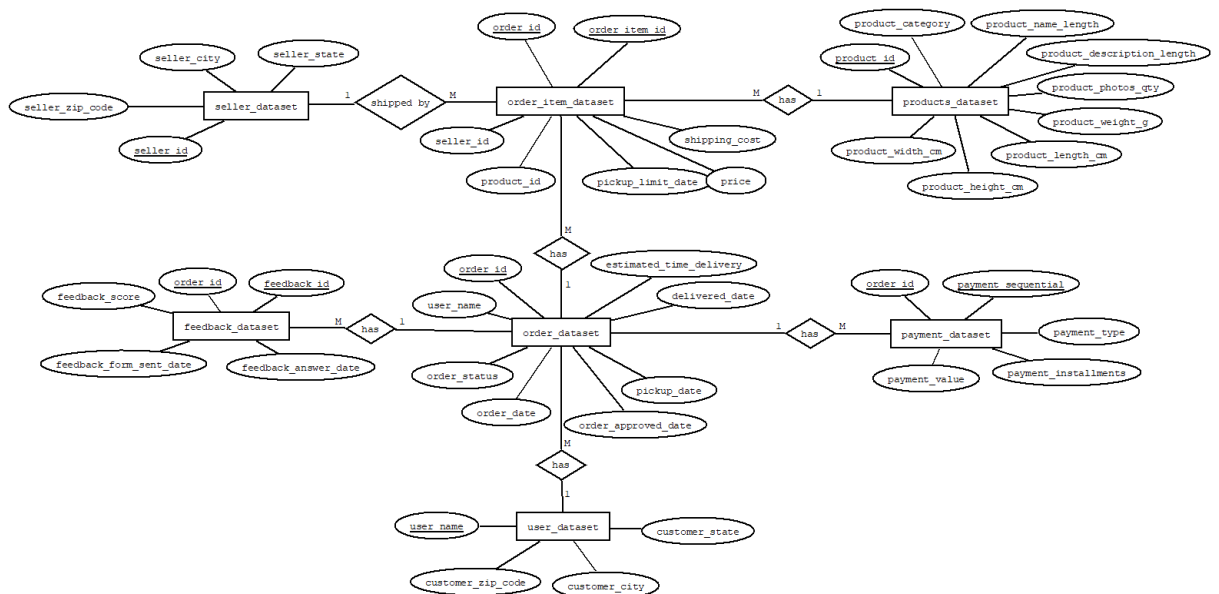
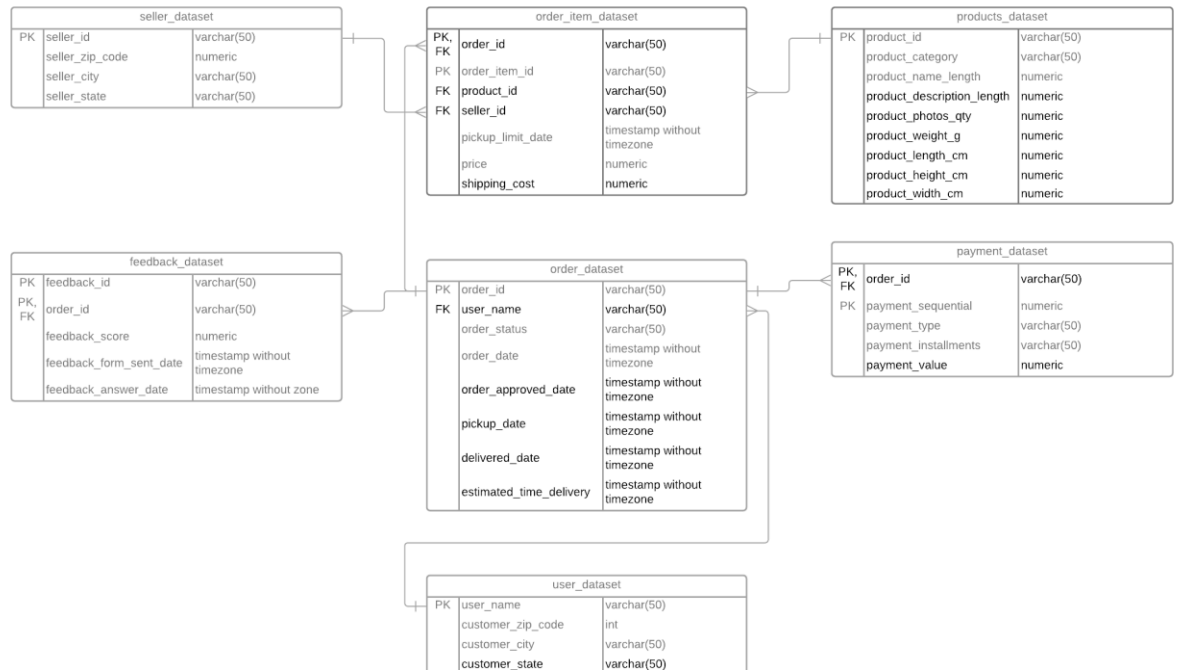
4. Assign Foreign Key

5. Untuk assign foreign key, saya langsung assign di GUI Dbeaver.

Tabel	Foreign Key	Refer
feedback_dataset	order_id	order_dataset(order_id)
order_dataset	user_name	user_dataset(user_name)
order_item_dataset	order_id, product_id, seller_id	order_id(order_dataset), product_id(products_dataset), seller_id(seller_dataset)

payment_dataset	order_id	order_dataset(order_id)
-----------------	----------	-------------------------

6. Design ERD OLTP



Tabel	Relationship	Tabel
feedback_dataset	Many to one	order_dataset
order_item_dataset	Many to one	order_dataset
payment_dataset	Many to one	order_dataset
products_dataset	One to many	order_item_dataset
seller_dataset	One to many	order_item_dataset
user_dataset	One to many	order_dataset

Sabtu, 13 Februari 2021

1. Check feedback dataset

Pada feedback_dataset saya masih ingin mengecek hubungan dari feedback_dataset dan order_dataset. Disitu terdapat satu order_id yang memiliki banyak feedback_id, dengan score yang berbeda, feedback_form_sent_date yang hamper sama dan feedback_answer_date yang berbeda.

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the 'Object Explorer' with the 'ecommerce' database selected. The right pane shows a SQL query window with the following query:

```
USE ecommerce;
SELECT
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.order_id LIKE '%c88b1d1b157a9999ce368f218a407141%';
```

The 'Results' pane displays the following data:

	delivered_date	estimated_time_delivery	feedback_id	order_id	feedback_score	feedback_form_sent_date	feedback_answer_date	
1	5.2741.00000000	2017-07-21 17:06:30.0000000	2017-08-01 00:00:00.0000000	8b0e872a25632ac83eb188543c	c88b1d1b157a9999ce368f218a407141	3	2017-07-22 00:00:00.0000000	2017-07-26 13:41:07.0000000
2	5.2741.00000000	2017-07-21 17:06:30.0000000	2017-08-01 00:00:00.0000000	8b0e872a25632ac83eb188543c	c88b1d1b157a9999ce368f218a407141	5	2017-07-21 00:00:00.0000000	2017-07-26 13:48:15.0000000
3	5.2741.00000000	2017-07-21 17:06:30.0000000	2017-08-01 00:00:00.0000000	202b5944d09cd3dc0d8bd1261b044c	c88b1d1b157a9999ce368f218a407141	5	2017-07-22 00:00:00.0000000	2017-07-26 13:40:22.0000000

The status bar at the bottom indicates 'Query executed successfully. LAPTOP-F6ANU245\NIMAS (15.0... LAPTOP-F6ANU245\acer (64) - ecommerce 00:00:42 3 rows'.

SQLQuery2.sql - LAPTOP-F6AN245\NIMAS.ecommerce (LAPTOP-F6AN245\acer (58)) - Microsoft SQL Server Management Studio

Object Explorer

- LAPTOP-F6AN245\NIMAS (SQL Server 15.0.2000.5 - LAPTOP-F6AN245\acer (58))
- Databases
 - System Databases
 - AdventureWorksDW2019
 - ecommerce
 - Database Diagrams
 - dbo.ecommerce_oltp_diagram
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.feedback_dataset
 - dbo.order_dataset
 - dbo.order_item_dataset
 - dbo.payment_dataset
 - dbo.products_dataset
 - dbo.seller_dataset
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

SQLQuery2.sql - LA...6AN245\acer (58))

```
SELECT *
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.feedback_id LIKE '%38821b5c496b678c-f91acc34892805ad%';
```

Results

	user_name	order_status	order_date	order_approved_date	pickup_date	delivered_date	estimated_time_delivery	feedback_id
1	ab6a742a782b10f5c824b4b49e5cd1	cancelled	2017-08-18 14:30:19.0000000	NULL	NULL	NULL	2017-09-21 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad
2	ab6a742a782b10f5c824b4b49e5cd1	delivered	2017-08-18 14:30:22.0000000	2017-08-18 16:10:14.0000000	2017-08-22 14:37:12.0000000	2017-08-31 21:50:18.0000000	2017-09-18 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad
3	ab6a742a782b10f5c824b4b49e5cd1	delivered	2017-08-18 14:30:22.0000000	2017-08-18 16:10:13.0000000	2017-08-21 19:42:16.0000000	2017-09-02 16:28:00.0000000	2017-09-12 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad

Query executed successfully. LAPTOP-F6AN245\NIMAS (15.0.2000.5 - LAPTOP-F6AN245\acer (58)) ecommerce 00:00:00 3 rows

SQLQuery2.sql - LAPTOP-F6AN245\NIMAS.ecommerce (LAPTOP-F6AN245\acer (58)) - Microsoft SQL Server Management Studio

Object Explorer

- LAPTOP-F6AN245\NIMAS (SQL Server 15.0.2000.5 - LAPTOP-F6AN245\acer (58))
- Databases
 - System Databases
 - AdventureWorksDW2019
 - ecommerce
 - Database Diagrams
 - dbo.ecommerce_oltp_diagram
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.feedback_dataset
 - Columns
 - Keys
 - PK_feedback_dataset
 - FK_feedback_order_4f7CD00D
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.order_dataset
 - dbo.order_item_dataset
 - dbo.payment_dataset
 - dbo.products_dataset
 - dbo.seller_dataset
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

SQLQuery2.sql - LA...6AN245\acer (58))

```
SELECT *
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.feedback_id LIKE '%38821b5c496b678c-f91acc34892805ad%';
```

Results

	delivered_date	estimated_time_delivery	feedback_id	order_id	feedback_score	feedback_form_sent_date	feedback_answer_date
1	NULL	2017-09-21 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad	02a723e8e8b4a123414f56c9c4a605e	5	2017-09-03 00:00:00.0000000	2017-09-05 12:12:51.0000000
2	2017-09-18 00:00:00.0000000	2017-09-18 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad	261783a32bc5f1096a518b9a0c95ae	5	2017-09-03 00:00:00.0000000	2017-09-05 12:12:51.0000000
3	2017-09-02 16:28:00.0000000	2017-09-12 00:00:00.0000000	38821b5c496b678c-f91acc34892805ad	2648bc14df7b5aef56c2e9f6a3b7c0d8	5	2017-09-03 00:00:00.0000000	2017-09-05 12:12:51.0000000

Query executed successfully. LAPTOP-F6AN245\NIMAS (15.0.2000.5 - LAPTOP-F6AN245\acer (58)) ecommerce 00:00:00 3 rows

SQLQuery2.sql - LAPTOP-F6AN24S\NIMAS.ecommerce (LAPTOP-F6AN24S\acer (58)) - Microsoft SQL Server Management Studio

Object Explorer: LAPTOP-F6AN24S\NIMAS (SQL Server 15.0.2000.5 - LAPTOP-F6AN24S\acer (58))

Query: SQLQuery2.sql - LA...6AN24S\acer (58))

```

SELECT
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.feedback_id LIKE '%f4b69d6d4f6bdcc2298f6e7b17b8e1e%';

```

Results:

	estimated_time_delivery	feedback_id	order_id	feedback_score	feedback_form_sent_date	feedback_revised_date
1	244.40.000000	2018-03-19 21:45:52.000000	2018-04-02 00:00:00.000000	4	2018-03-29 00:00:00.000000	2018-03-30 00:29:09.000000
2	244.43.000000	2018-03-05 15:35:13.000000	2018-04-02 00:00:00.000000	4	2018-03-29 00:00:00.000000	2018-03-30 00:29:09.000000
3	042.31.000000	2018-03-26 16:46:49.000000	2018-04-02 00:00:00.000000	4	2018-03-29 00:00:00.000000	2018-03-30 00:29:09.000000

Query executed successfully. LAPTOP-F6AN24S\NIMAS (15.0... LAPTOP-F6AN24S\acer (58)) ecommerce 00:00:00 3 rows

SQLQuery1.sql - LAPTOP-F6AN24S\NIMAS.ecommerce (LAPTOP-F6AN24S\acer (53)) - Microsoft SQL Server Management Studio

Object Explorer: LAPTOP-F6AN24S\NIMAS (SQL Server 15.0.2000.5 - LAPTOP-F6AN24S\acer (53))

Query: SQLQuery1.sql - LA...6AN24S\acer (53))

```

USE ecommerce;
SELECT
feedback_dataset.order_id, order_dataset.user_name,
COUNT (feedback_dataset.order_id),
COUNT (order_dataset.user_name)
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
GROUP BY
feedback_dataset.order_id, order_dataset.user_name
HAVING
COUNT (order_dataset.user_name) > 1
ORDER BY
COUNT (order_dataset.user_name) DESC;

```

Results:

order_id	user_name	(No column name)	(No column name)
1	c8bb141b157a9999ce368021ba407141	b54ebaf3d11b7209e368364cc359a51	3
2	03a9394f763b389455a96579919b	295a8f310963652a20394a797111	3
3	8e170729c720c90e1111e589a0c85	6680b377553753677b43474e52095	3
4	d56136b031ec028a200b18e6d8b2e	2e43e031f10d628e55735af6899396	3
5	03ebaf69af8563e8114ab5a7ccafcd	32abf7b9770eb67a5143006895485	2
6	03515a336baf558a3f89dew52a3b6	199308b0e762097eb9a4e29e3a294d3	2
7	013056d4978a0d96b4a03396c4e3	51af1e5ab725531b3a2b64586809a	2
8	04f182708b4972a622445203a071500	b4699ac11422a050c8b98a77da35a5	2
9	003524640520710769010752e7507	d56de4569d49821b283710a56abde	2
10	0176e684bcb3b0a3a3116a9a768997	13613cae6e3b6a8327ad16245c9ec9	2
11	ca203a695a8a1200905acdb4b63c7f9	c0394d496256a4e365499919a3a6b	2
12	e9e7a6e6f033a371a2494a428e495	6e7808445ab9d71b654848a0c9f90	2
13	0749426f1c48b5943c8d1316ace0aa	8495960cbe76a3631a2253c6445d17	2
14	095e9e011478409362d726a75d73a6c	623674d39d0350492b191c469ba39	2
15	02355029a040a0a056a99000413	24acdf7ba5d1a88240b376c338da501	2
16	071564c733a372a181ab4a786a5509	83a5b393b76a6d189b3a3b15478d6	2
17	02885348b985a1e5a5a02702c662c5	78a665a6d76b0a8754a20a0484764e7	2

Query executed successfully. LAPTOP-F6AN24S\NIMAS (15.0... LAPTOP-F6AN24S\acer (53)) ecommerce 00:00:00 555 rows

The top screenshot shows a SQL query in Microsoft SQL Server Management Studio. The query is as follows:

```
USE ecommerce;
SELECT
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.order_id LIKE 'X1b9ecfe83cdc259250e1a8aca174f0ad';
```

The results pane shows one row of data:

order_id	user_name	order_status	order_date	order_approved_date	pickup_date	delivered_date	estimated_time_delivery	feedback_id
1b9ede83cdc259250e1a8aca174f0ad	8ea097b1824db4617a77b5d4e04301	cancelled	2018-08-04 14:29:27.0000000	2018-08-07 04:10:28.0000000	NULL	NULL	2018-08-14 00:00:00.0000000	14d540d4e5632307a16088e978

The bottom screenshot shows a similar SQL query, but with an additional join to order_dataset:

```
USE ecommerce;
SELECT
FROM
order_dataset INNER JOIN feedback_dataset ON order_dataset.order_id = feedback_dataset.order_id
INNER JOIN order_dataset ON order_dataset.order_id = feedback_dataset.order_id
WHERE feedback_dataset.order_id LIKE 'X1b9ecfe83cdc259250e1a8aca174f0ad';
```

The results pane shows one row of data:

id_date	pickup_date	delivered_date	estimated_time_delivery	feedback_id	order_id	feedback_score	feedback_form_sent_date	feedback_answer_date
4:10:28.0000000	NULL	NULL	2018-08-14 00:00:00.0000000	14d540d4e5632307a389e9416088e978	1b9ede83cdc259250e1a8aca174f0ad	1	2018-08-17 00:00:00.0000000	2018-08-19 13:51:01.0000000

Kemudian terdapat satu feedback_id yang memiliki banyak order_id. Dan yang unik adalah, terdapat order yang dicancel, tapi ternyata memiliki feedback_score sebesar 5. Jika dipikirkan rasanya aneh. Saya awalnya ingin menghapus saja order yang cancel, karena saya juga tidak ingin memvisualisasikan order yang cancel. Kemudian saya menemukan suatu data dimana order yang batal mempengaruhi feedback_score dimana dinilai 1.

Jika saya mempertahankan feedback_id menjadi primary key, maka saya harus hanya mengambil 1 order_id. Hanya saja, satu feedback ini mewakili banyak order yang berbeda. Namun dari sisi bisnis saya berpikir, setiap order itu berhak memiliki feedback. Akhirnya saya mempertahankan feedback_dataset dan order_dataset menjadi many to many.