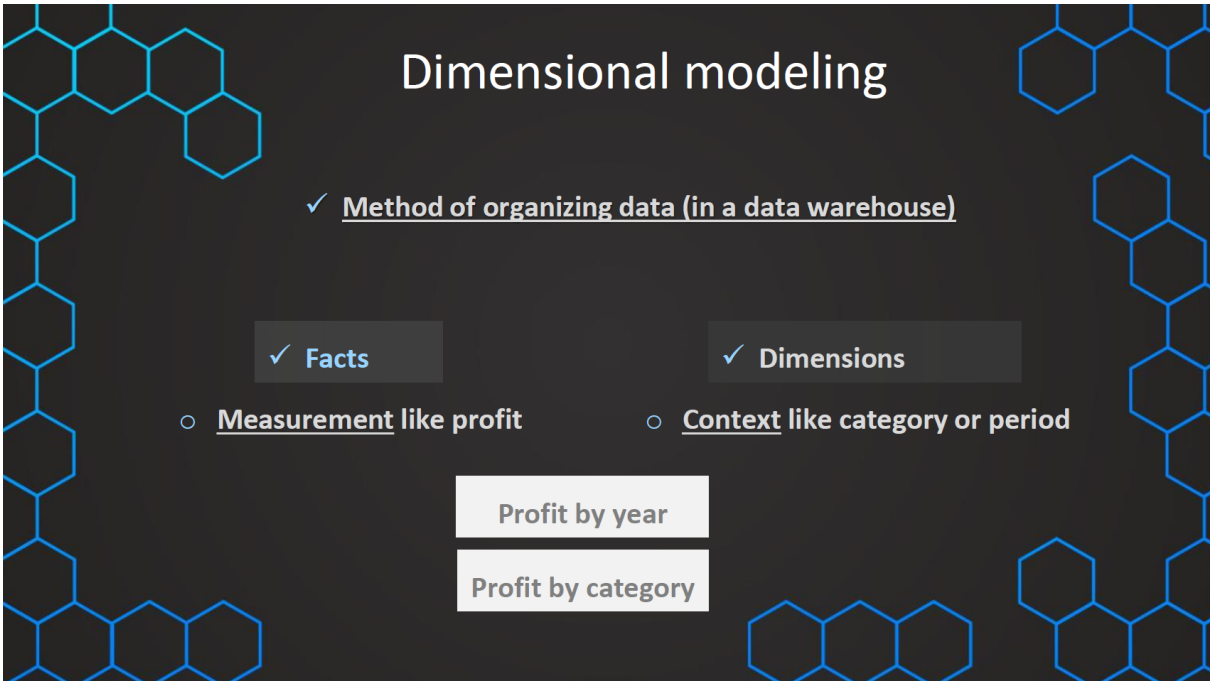
05. Dimensional modeling

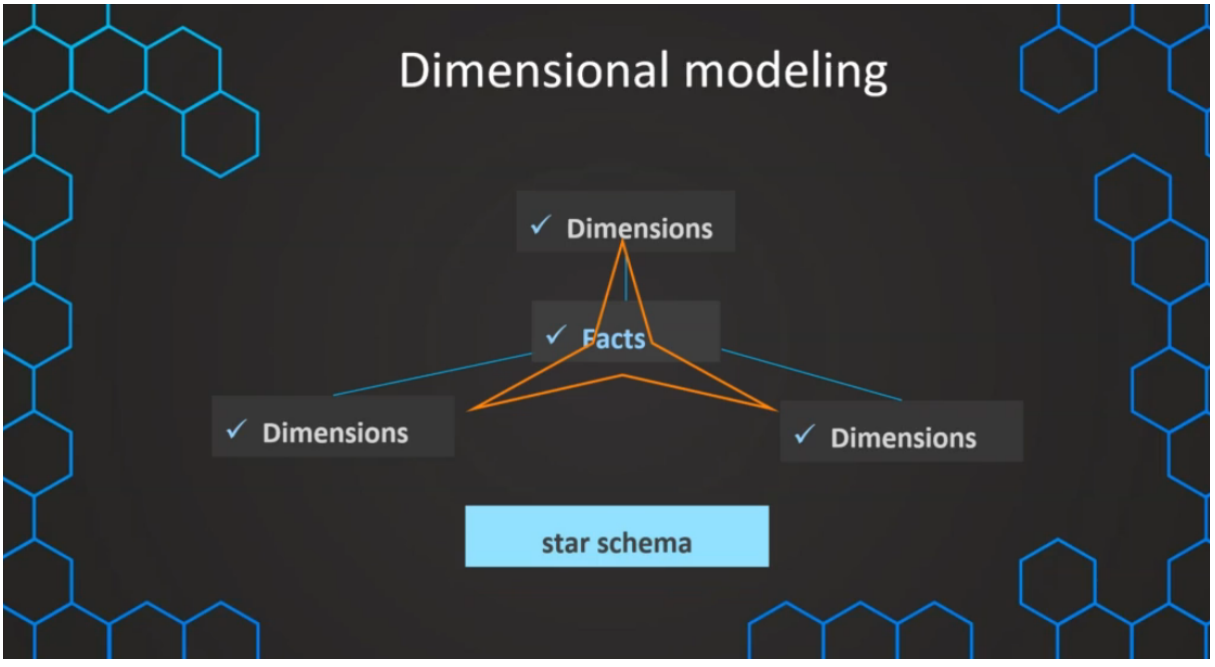
What is dimensional modeling?

- **Dimensional Data Modeling** is one of the data modeling techniques used in data warehouse design.
- The concept of Dimensional Modeling was developed by Ralph Kimball which is comprised of facts and dimension tables.
- Since the main goal of this modeling is to improve the data retrieval so it is optimized for SELECT OPERATION.
- The advantage of using this model is that we can store data in such a way that it is easier to store and retrieve the data once stored in a data warehouse.
- The dimensional model is the data model used by many OLAP systems.



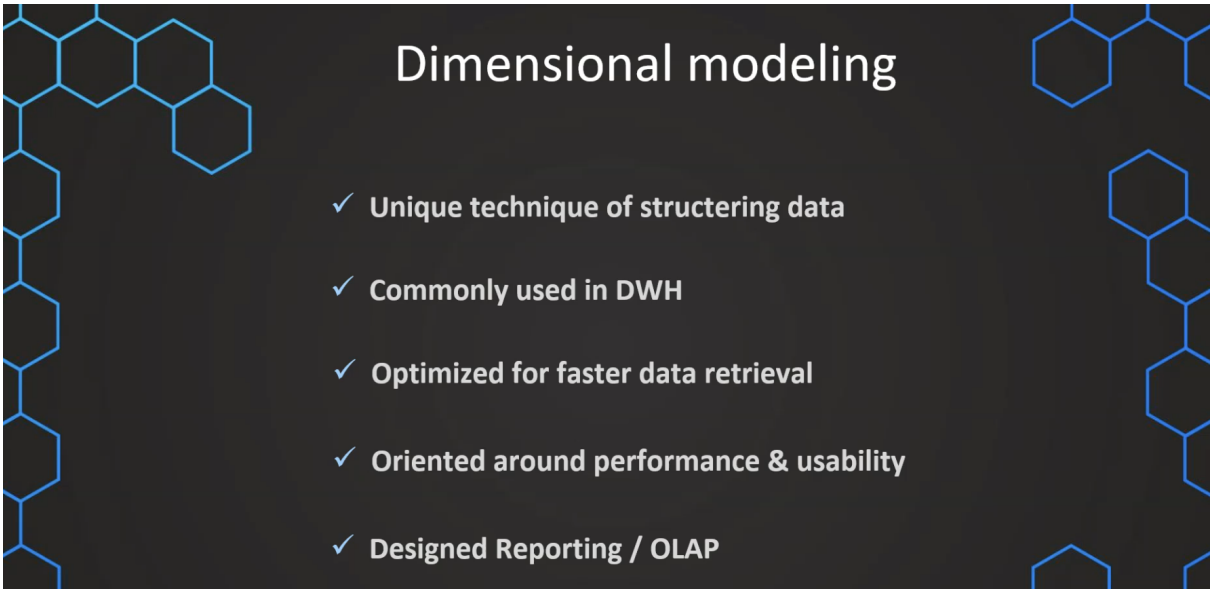
- In a dimensional model, all of our data is organized in either facts or dimensions.
- Fact can be something that is usually measured, so something like a profit that can be aggregated.

- Dimensions on the other side, give additional context to those measurements.
    - For example, it can be something like a month, a time period, or also a product category and then with this dimension, we can give additional context.
- So we can turn the fact, the measurement, with the context into meaningful insight.
    - For example, we can analyze the profit by year or the profit by category and then with this additional context to our measurement, we get the meaningful insight from our data.
- Keywork like ***by*** *year* or ***by*** *category* tell it is a dimension
- We have this fact in the middle, and we have multiple dimensions clustered around this fact.
- So we can use all of these different dimensions to analyze our data, and the measurements in our fact table and because of this visual appearance of a star, we also call this method or this alignment a **Star Schema**.



- So dimensional modeling is referring to a unique technique that is used in a data warehouse to structure our data and it is optimized for the purpose of a data warehouse that is to have fast data retrieval so it is oriented around performance and usability

- This technique is peferred to have high performance for reporting and OLAP use cases



Why dimensional modeling?

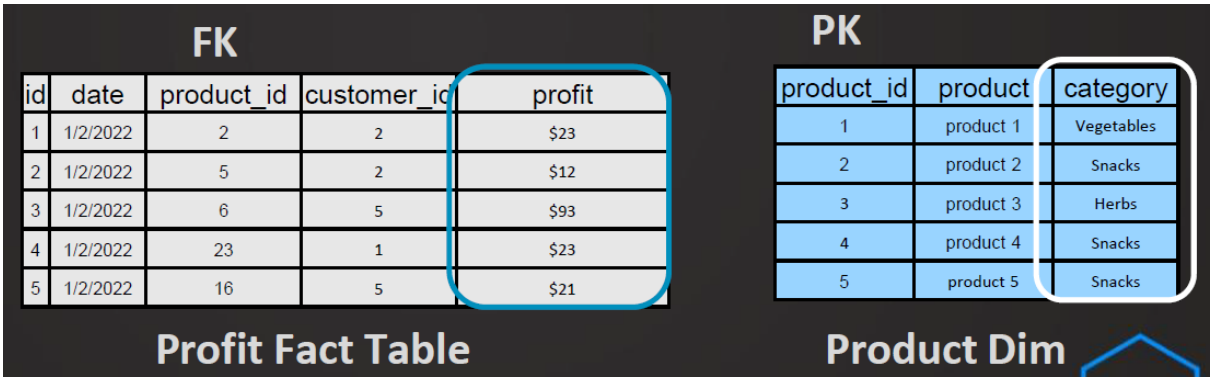
- Goals of dimensional modeling are:
  - fast data retrieval
  - Performance and usability
- Example: We want to calculate total profit. In the database, the table is scanned through each rows. Performance of the wider tables is not as efficient as narrow tables
- Also if we arrange the data by name we see the name of customer repeated over and over again and if product is purchased 1000 times, the product name will be repeated over and over again

id	date	product	category	customer_id	name	profit
1	1/2/2022	Fulltoss Tangy Tomato	Vegetables	2	Sarah	$23
2	1/2/2022	Chilli - Green, Organically Grown	Snacks	2	Sarah	$12
3	1/2/2022	Masala Powder	Herbs	5	Marc	$93
4	1/2/2022	Cheese Cracker (Mcvities)	Snacks	1	Frank	$23
5	1/2/2022	Centre Filled Chocolate Cake	Snacks	5	Marc	$21

- To improve query performance, we need to narrow the table, we can do that by removing the columns which will have repeated data and keep it separately, we keep customer data in separate dimension table and same thing can be done for product and category

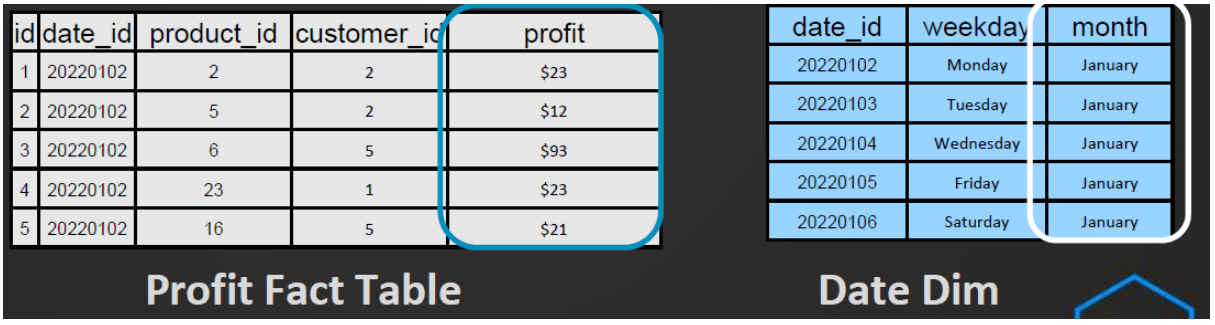
id	date	product	category	customer_id	profit
1	1/2/2022	Fulltoss Tangy Tomato	Vegetables	2	$23
2	1/2/2022	Chilli - Green, Organically Grown	Snacks	2	$12
3	1/2/2022	Masala Powder	Herbs	5	$93
4	1/2/2022	Cheese Cracker (Mcvities)	Snacks	1	$23
5	1/2/2022	Centre Filled Chocolate Cake	Snacks	5	$21

id	date	product	category	customer_id	profit
1	1/2/2022	Fulltoss Tangy Tomato	Vegetables	2	$23
2	1/2/2022	Chilli - Green, Organically Grown	Snacks	2	$12
3	1/2/2022	Masala Powder	Herbs	5	$93
4	1/2/2022	Cheese Cracker (Mcvities)	Snacks	1	$23
5	1/2/2022	Centre Filled Chocolate Cake	Snacks	5	$21

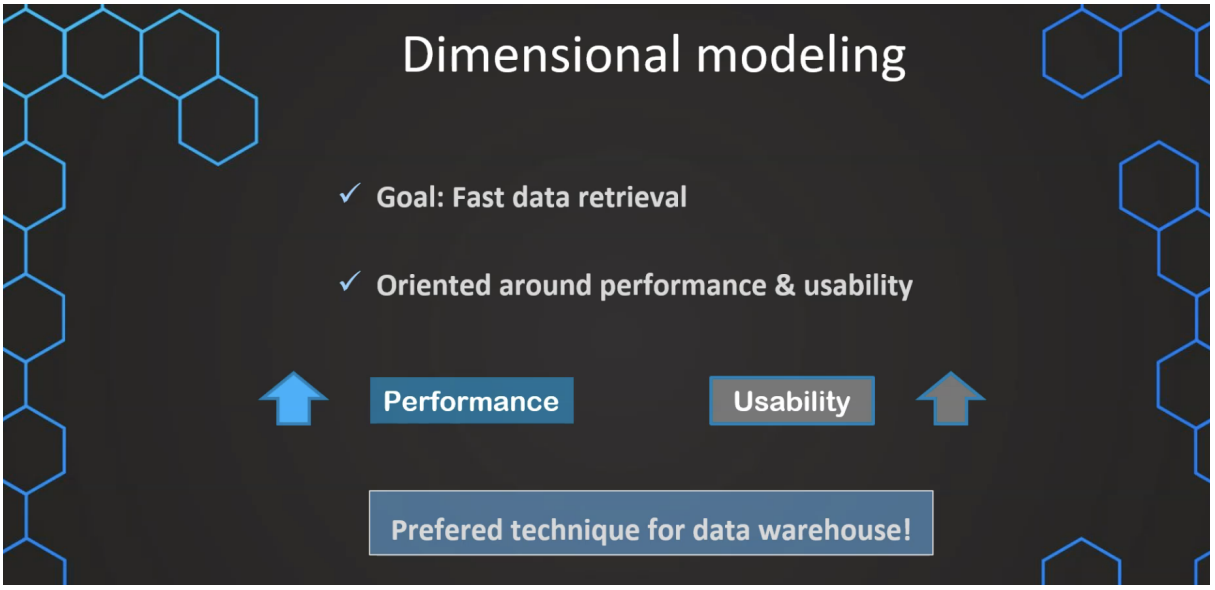


- Now the data is structured into logical units and also have better query performance
- All date information can also be put into different date dimension table and use the foreign key

id	date_id	product_id	customer_id	profit
1	20220102	2	2	$23
2	20220102	5	2	$12
3	20220102	6	5	$93
4	20220102	23	1	$23
5	20220102	16	5	$21

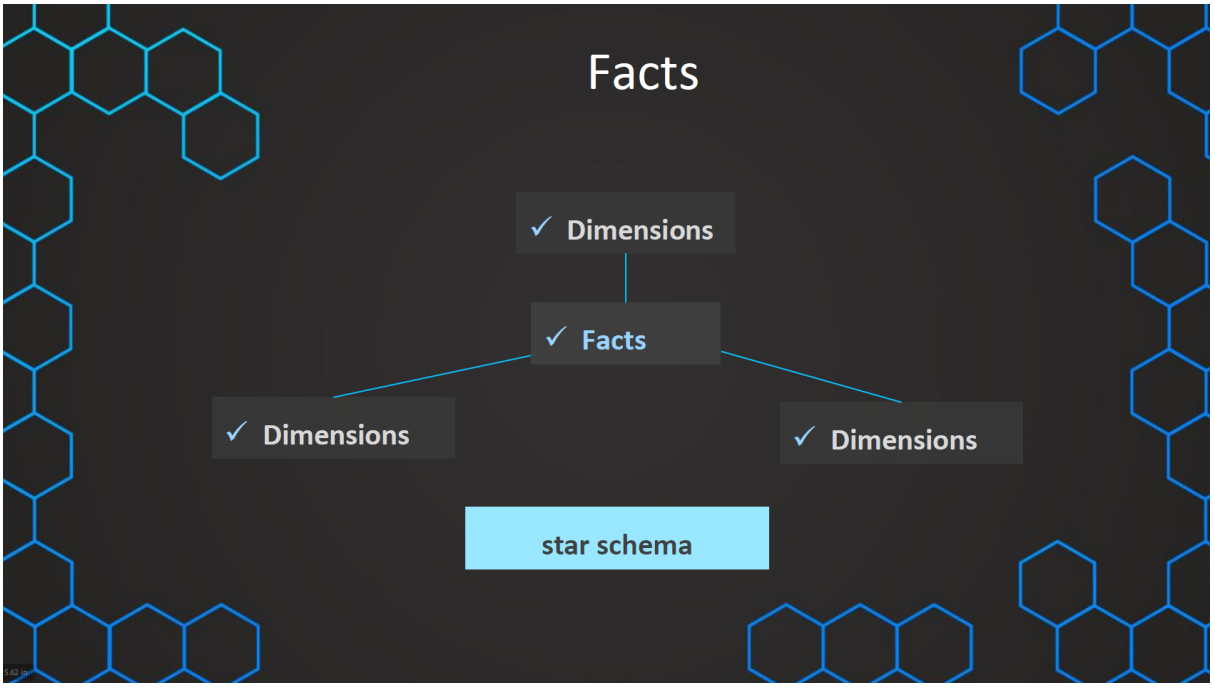


- Now we can find all date related information in date dimension table and we don’t have to search in hundreds of columns in 1 fact table and this makes things easier for users
- Now we can calculate profit just for particular month and particular weekday easily and this is how the benefit of high query performance and usability comes in
- This dimension modling technique of creating facts and dimensions is the preferred way to design a data warehouse to enable fast query performance and usability for OLAP use cases and reporting

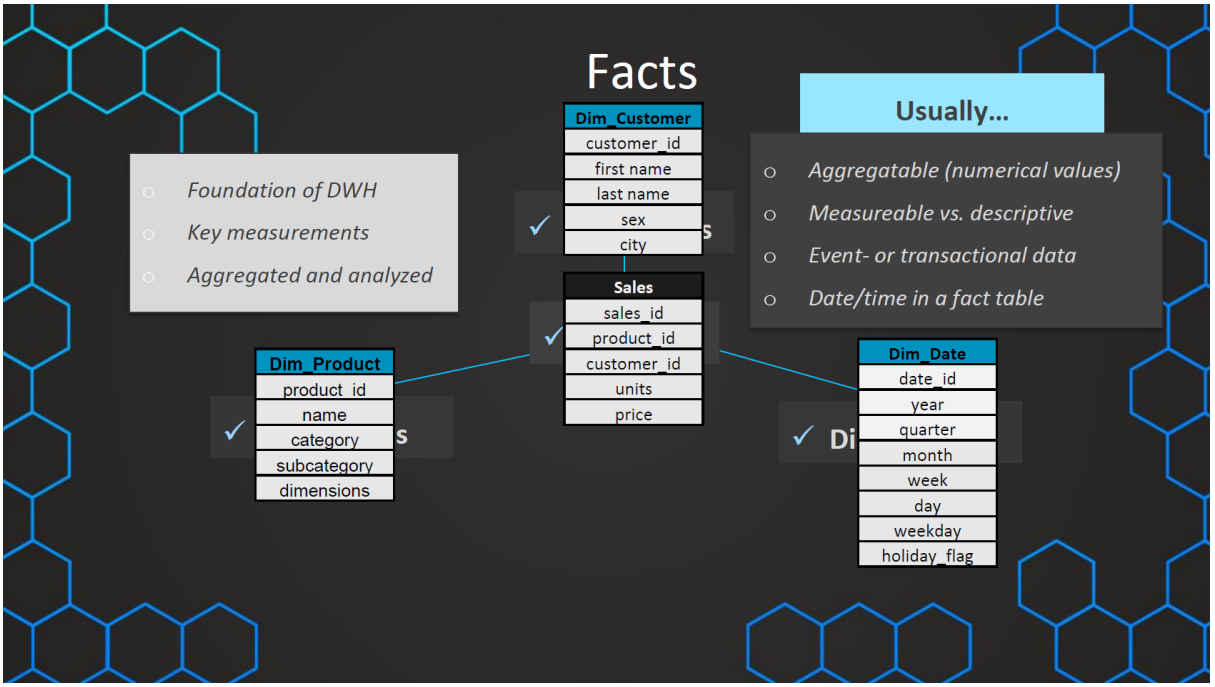


Facts

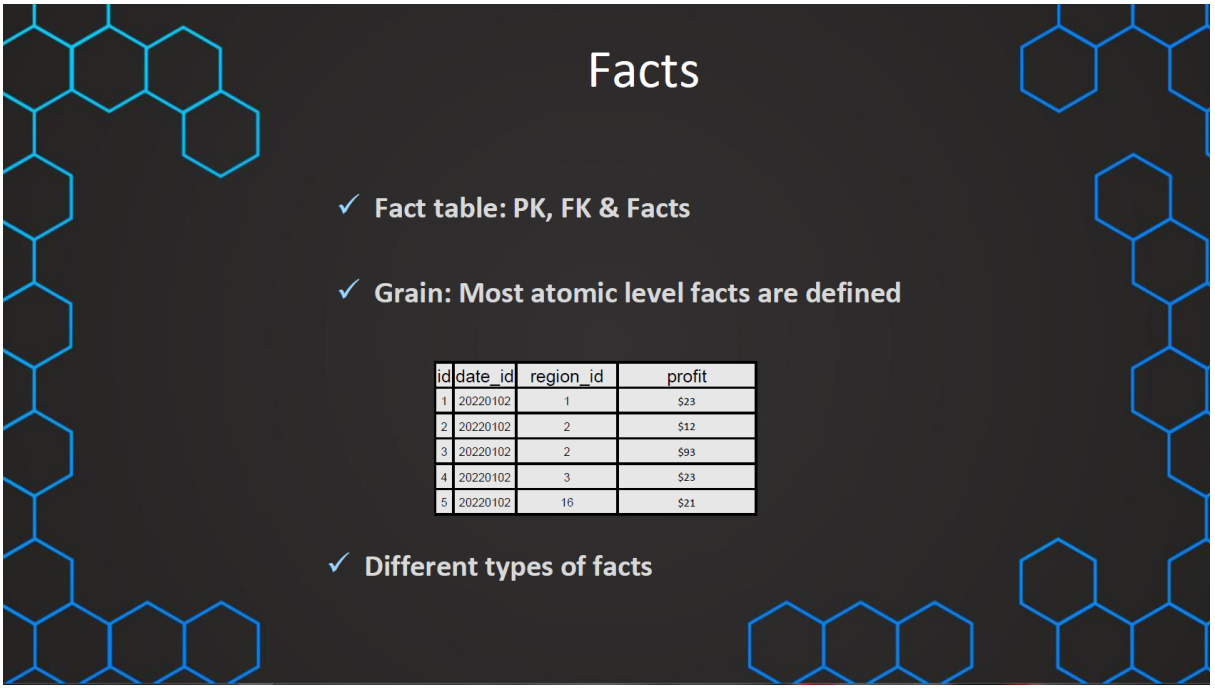
- In a star schema, which is the most common schema in our data warehouse, when we model our data in a dimensional way, we have the fact table in the middle and then we have the dimensions clustered around this fact.



- Example, we can have a fact table of sales , this table will be at the center and all dimension tables clustered around it. This fact table contains all important measurement of the company

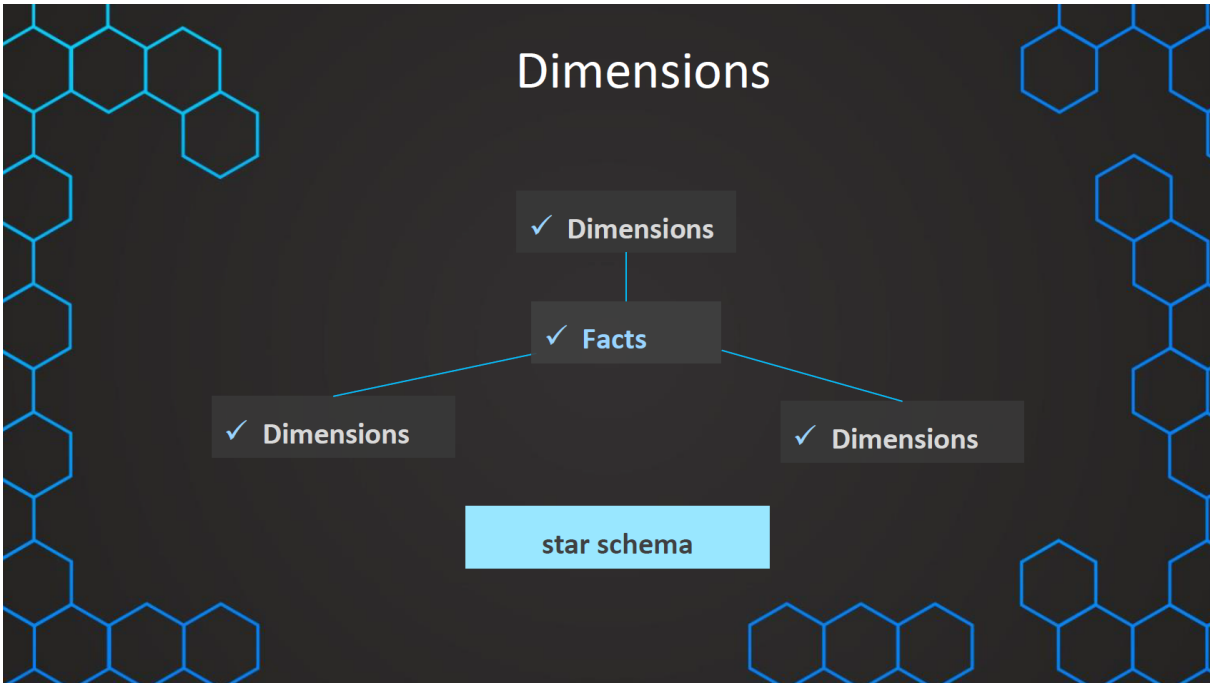


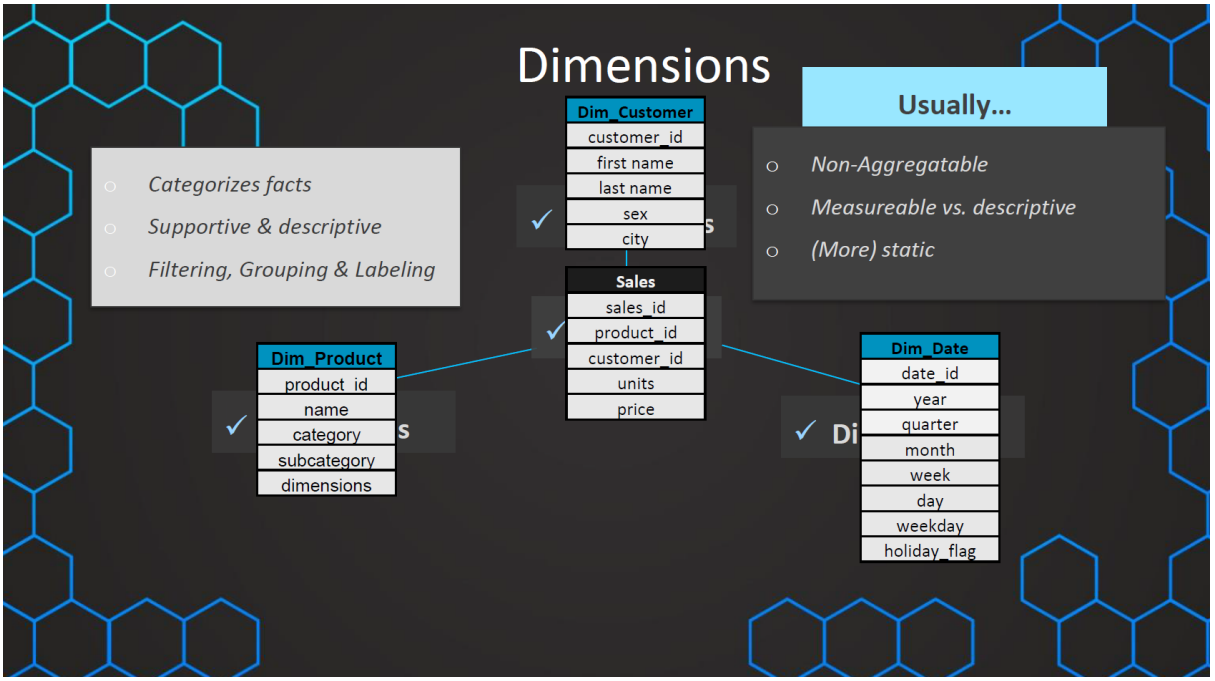
- It can have information like the profit we made and number of units sold
- This fact table is the foundation of our data warehouse because it contains key measurements of our company
- These facts are usually aggregated and analysed BY dimensions
- How to identify fact tables
  - Usually the facts are additive (numerical values), if we add them up , it makes sense like total amounts of units sold
  - They are not descriptive but are measurable
  - Event or transactional based data
  - Date/time is in a fact table. The transaction date itself is not a fact
- Fact table consist os Primary key to uniquely identify each rows of fact table and contains many foreign keys which are reference to our dimensions and we have the facts itself (aggregatable values)
- So these facts can be, for example, sales, profit, budget or any other things that are of key interests and can be measured in our company.
- Fact table is defined by GRAIN (most atomic level facts are defined) We have 1 row for each date and each region a profit defined
- Fact table is containing the facts, which are the key measurements in our company, and they are in the middle of our star schema



Dimension

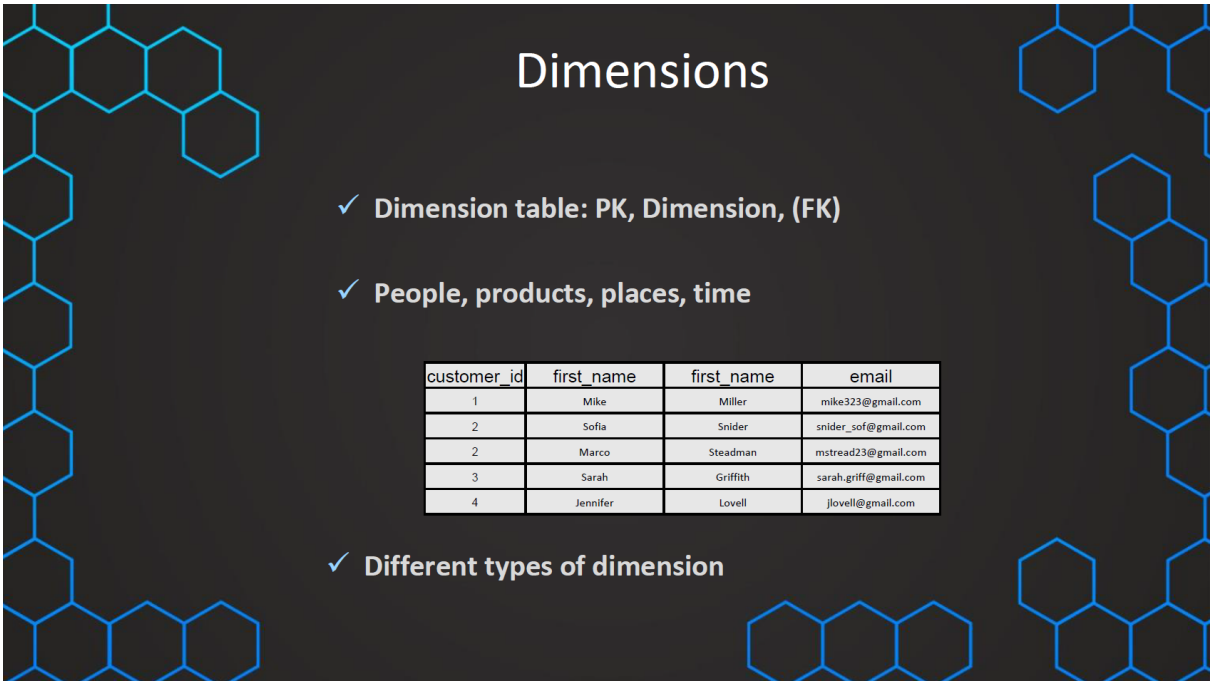
- In a star schema, dimensions are clustered around a fact
- Purpose of dimension is to categorize the facts so we get meaningful context for our measurements
- Character of these dimensions are more supportive and descriptive to the fact
- We can’t measure a dimension but can be used to describe to filter, group and label our data a.k.a slicing and dicing the data
- Later on in a report, for example, we need a filter or we want to group our data in a bar chart, this is where we want to use the dimensions.





- Dimensions can be numerical but are never aggregatable e.g. years in a date table, if we add all years nothing makes sense with the total sum of all years like we can't add 2001 and 2003 to get 4004 to mean anything
- The dimensions are descriptive in nature
- And also while in a fact table we have some things moving or some things happening, we have the dimensions that are a little bit more static.

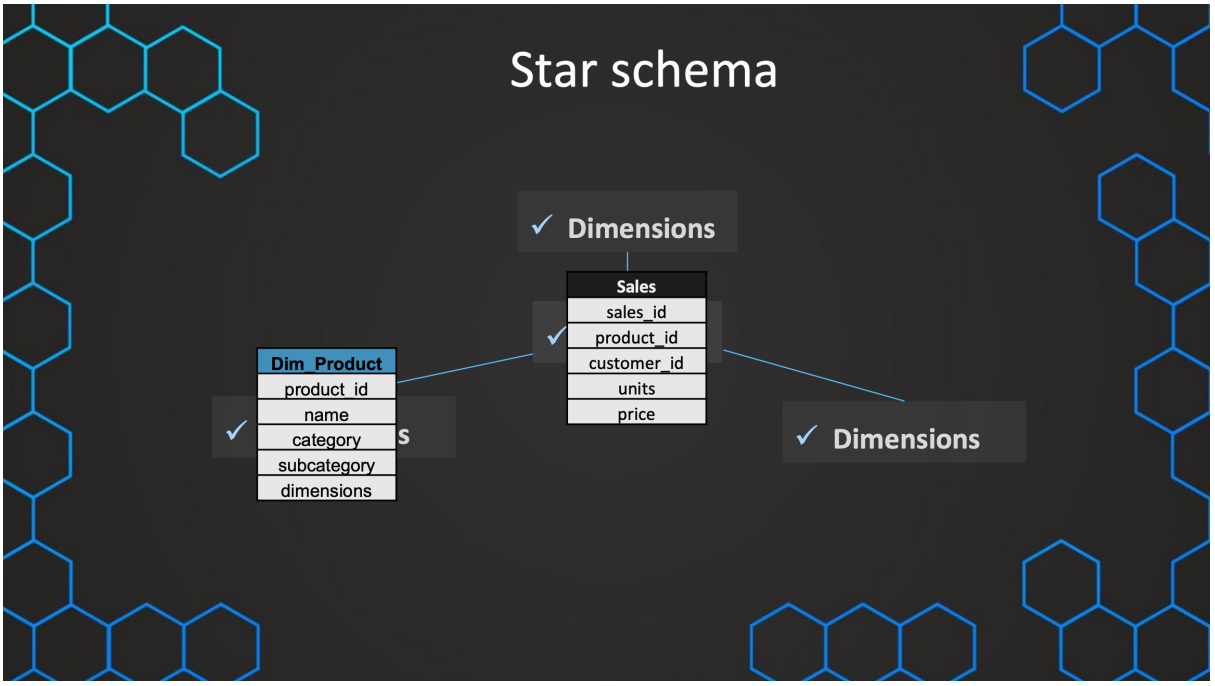
- So for example, product name, product category, this is something that is usually not changing or there's not coming anything new, there's nothing happening.
- Of course, we can have changes in dimensions too but in general this data is more static.
- The dimension table consists of Primary key to uniquely identify the rows, additionally we can have a foreign key in our dimension and of course the dimensions itself like people, products, places, time, employees, managers, companies, brands, product categories, regions, cities, countries, address
- Below is a customer dimension, this can also change also called slowly changing dimension but is generally static



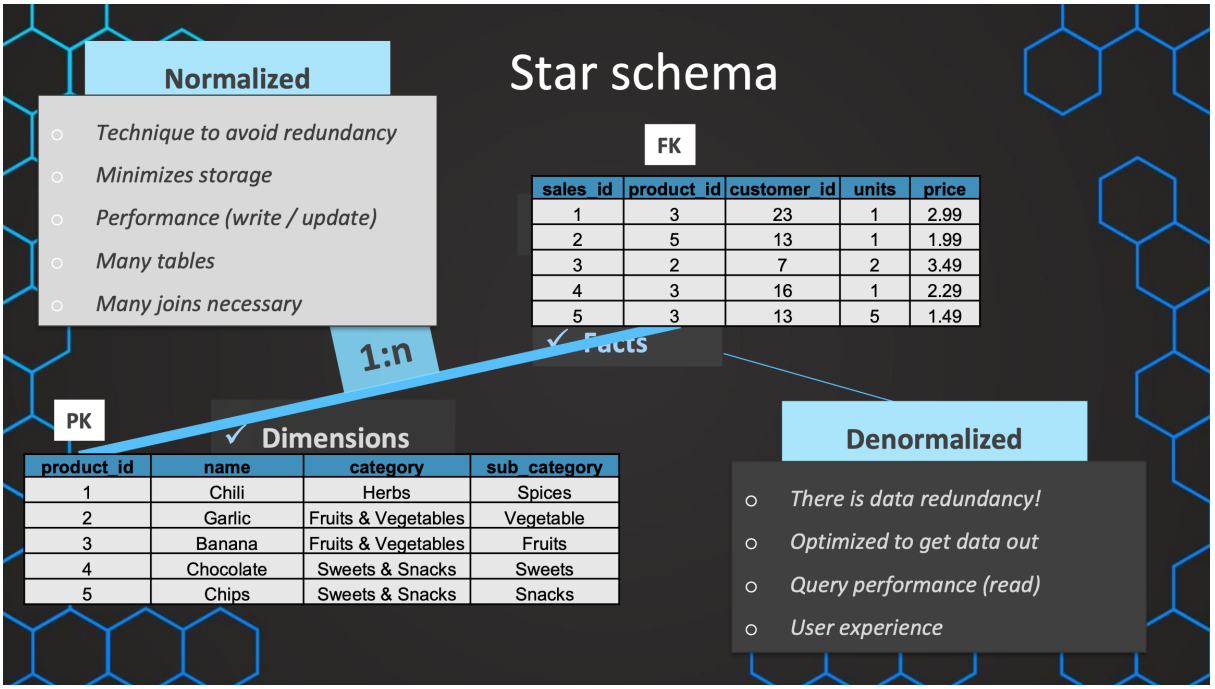
Star schema

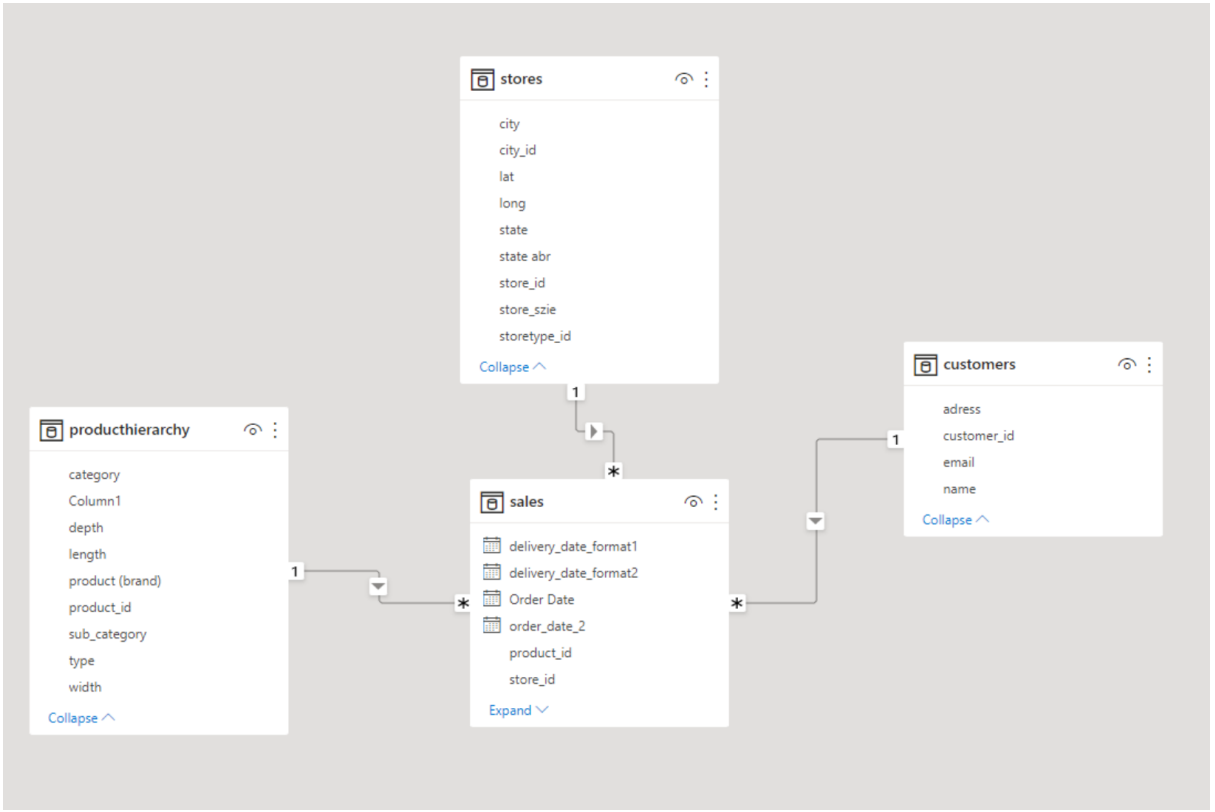
- Most important schema in our data warehouse
- We structure our data into facts and dimension tables
- We have the sales table, which contains all of our important facts, and we can create relationships. So using the foreign key and the primary key to create those connections or joins to our dimension tables in this case our product table
- We usually have 1:n relationship between dimension and fact table

- If we have 1 it means those values are occuring only once and if we mention ‘n’ then we saying values are being repeated



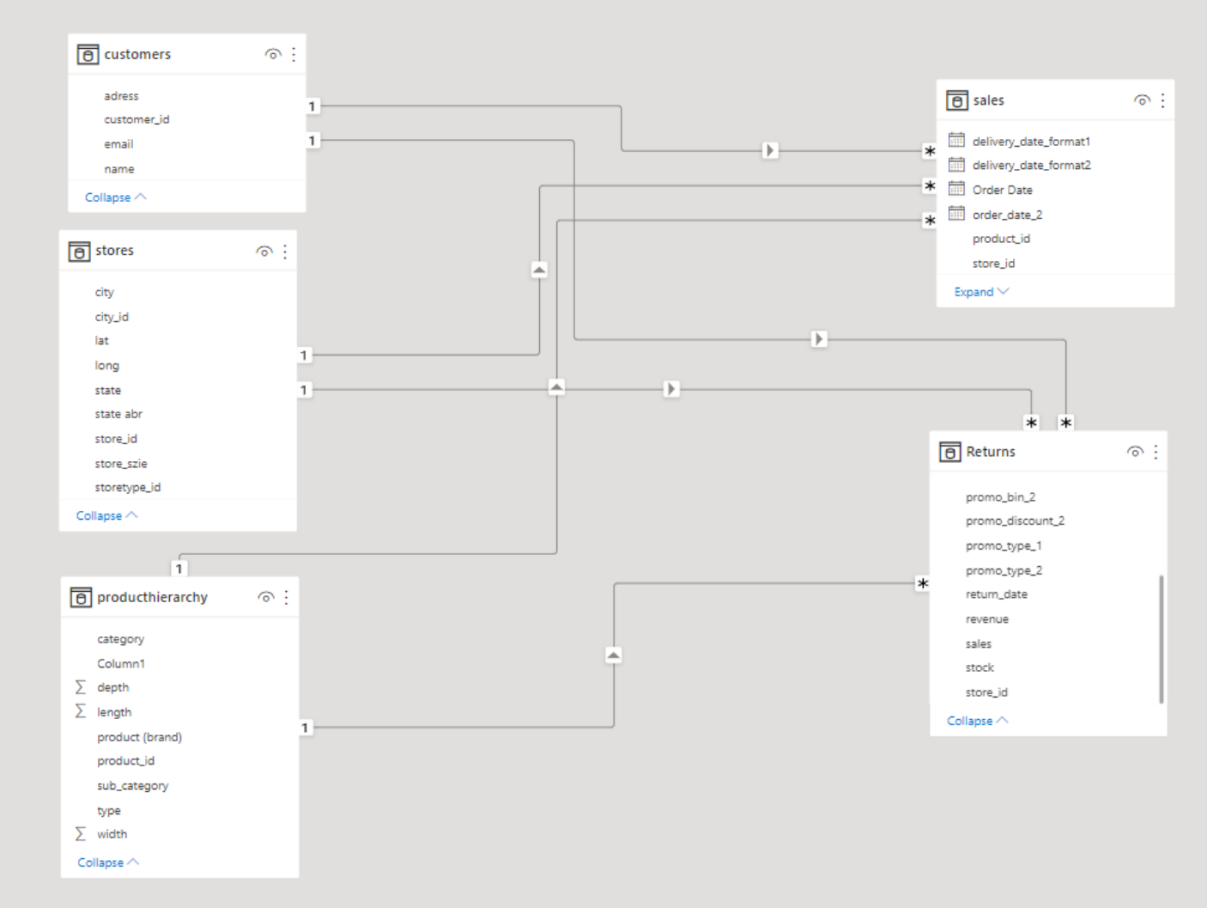
- If we have only 1 level of hierarchy there will be only one connection and there are no further hierarchy from the dimension table, this leads to data redundancy like the category columns of dimension table have repeated values. In SnowFlake schema we are reducing the data redundancy
- This reducing of data redundancy is called normalization
- But normalization is not good to get the data out and have good read performance on our operations and usability
- With normalization we reduce redundancy and reduce storage costs, it is efficient to update the data and good write performance
- But for reporting and other data warehouse activities we usually read the data more than write to it, hence we can accept the data redundancy for better read performance
- In our star schema, the data is in de-normalized upto certain degree due to data redundancy





- In the above diagram we can see the 1:n relationship which is mentioned as 1:* where * is representing many relationship
- Sometimes we can also have more than 1 fact tables, but we should try to define the GRAIN and try to get all the facts to 1 fact table but if it is not

possible we can have multiple fact tables and many dimensions tables which can be connected to more than 1 fact table

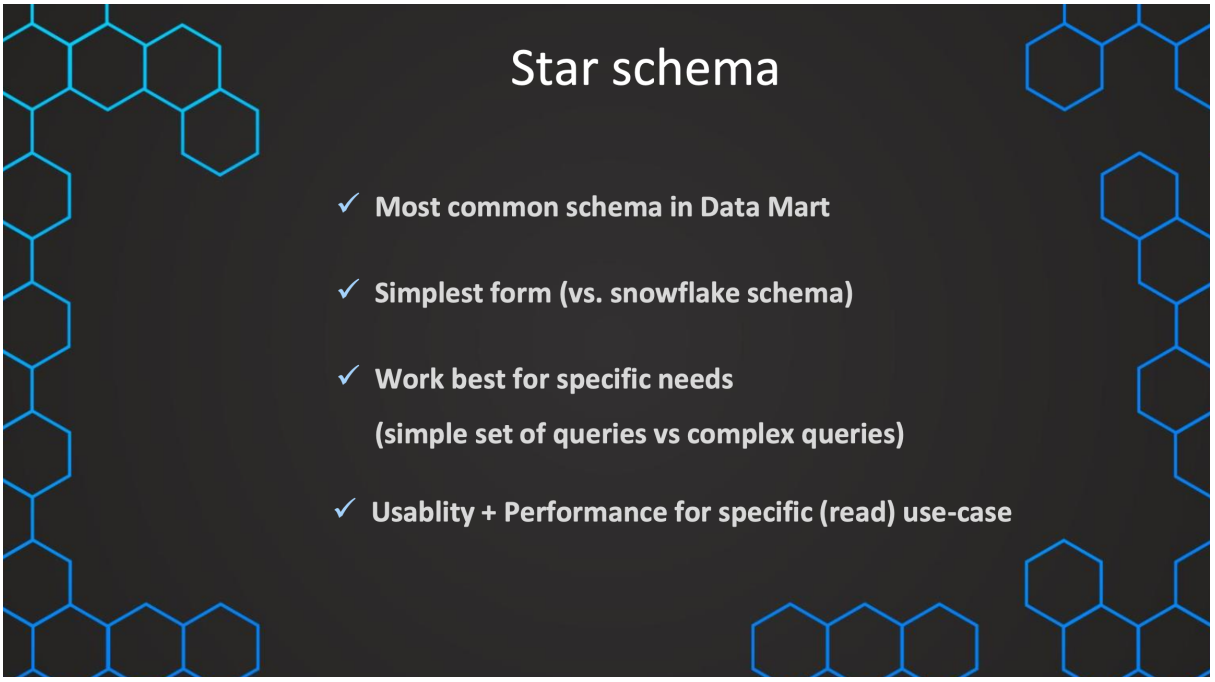


- Here we have 2 fact table and we have 3 dimension table, and all dimension tables are connected to both fact tablesusing 1: n relationship
- Fact tables do not have connection with each other

Summary

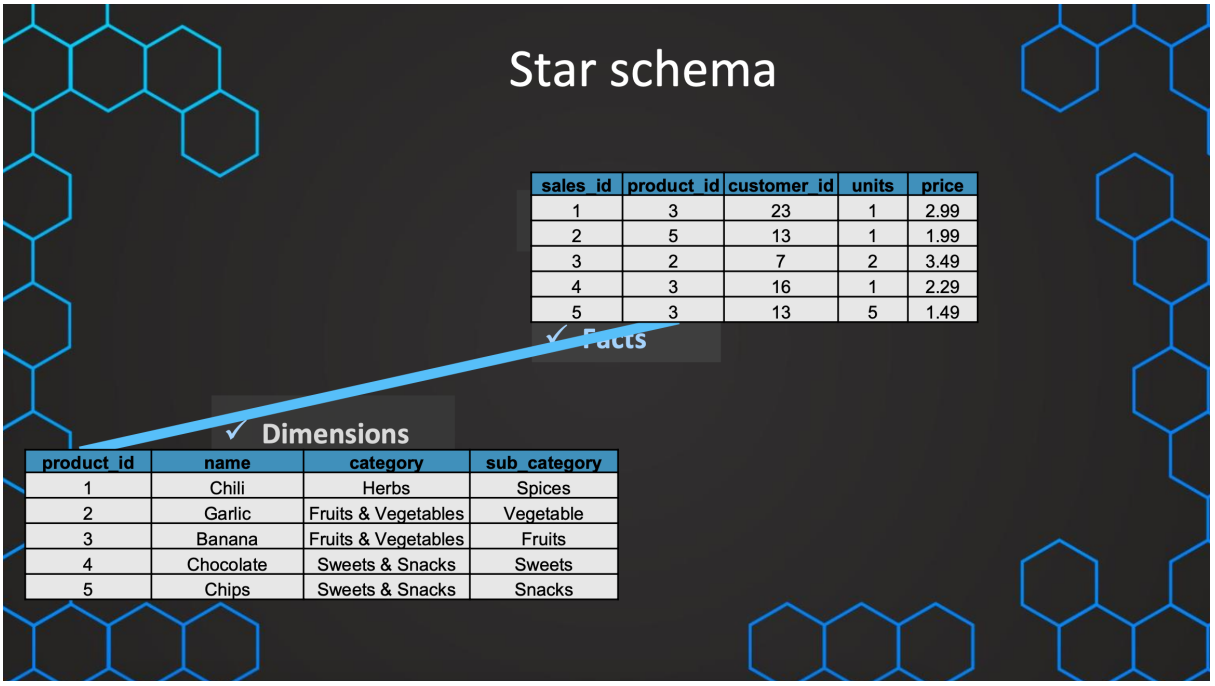
- Star schema is most common schema in the data marts and it combines user friendliness and usability
- Star schema is most simplest form compared to Snowflake schema
- Star Schema, especially optimized and ideal, if we have specific needs, that means we know what set of queries are usually run.
- So for example, we know it's very common that we want to visualize just the profit by the year. The profit by the categories and we don't need super complex queries and in such cases, if we have those more simple, and

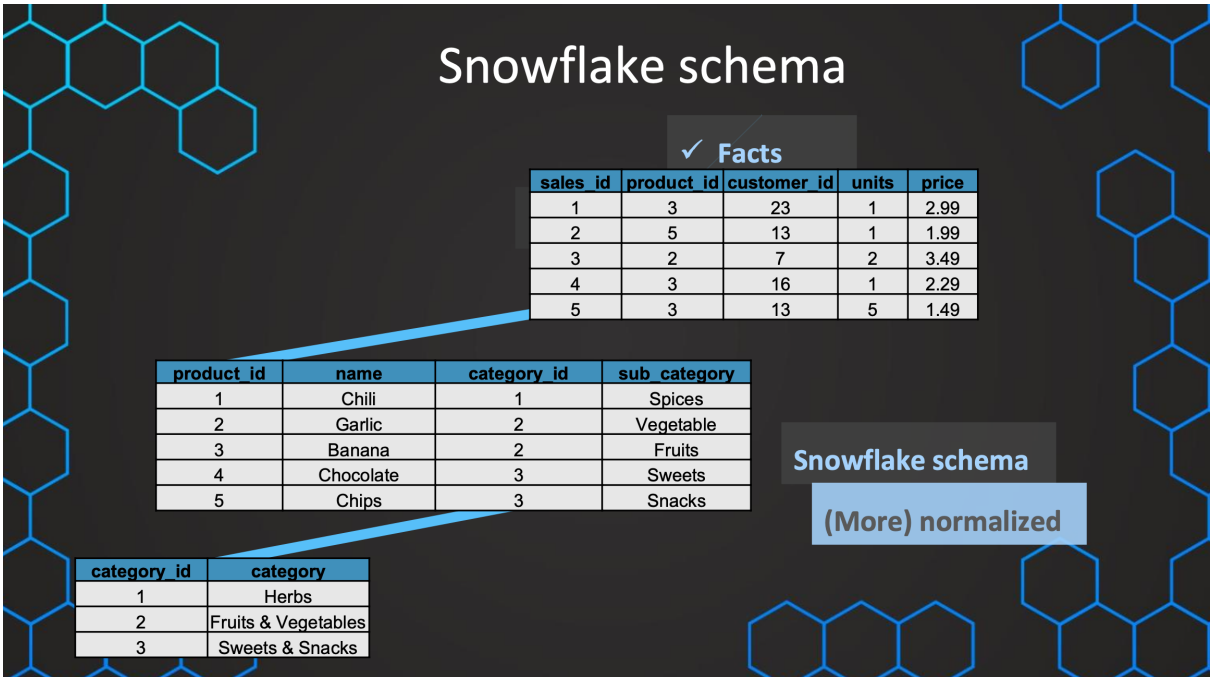
more defined set of queries, set of needs, we have the best performance, and the best usability with our Star Schema.

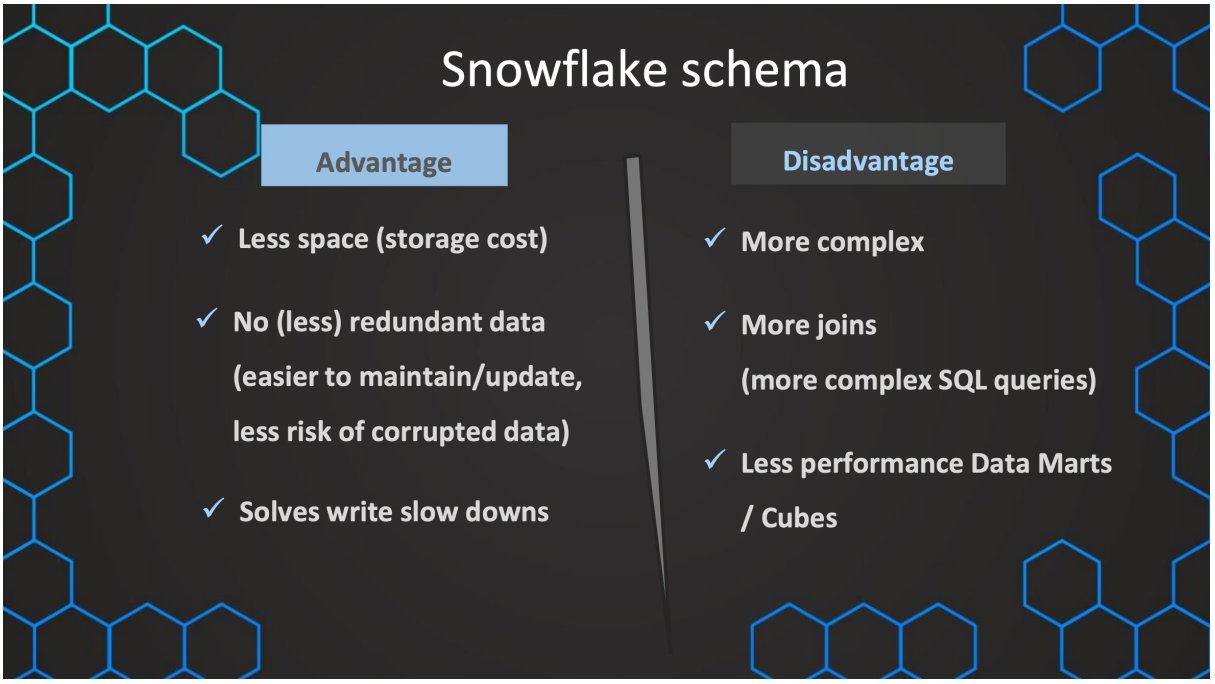


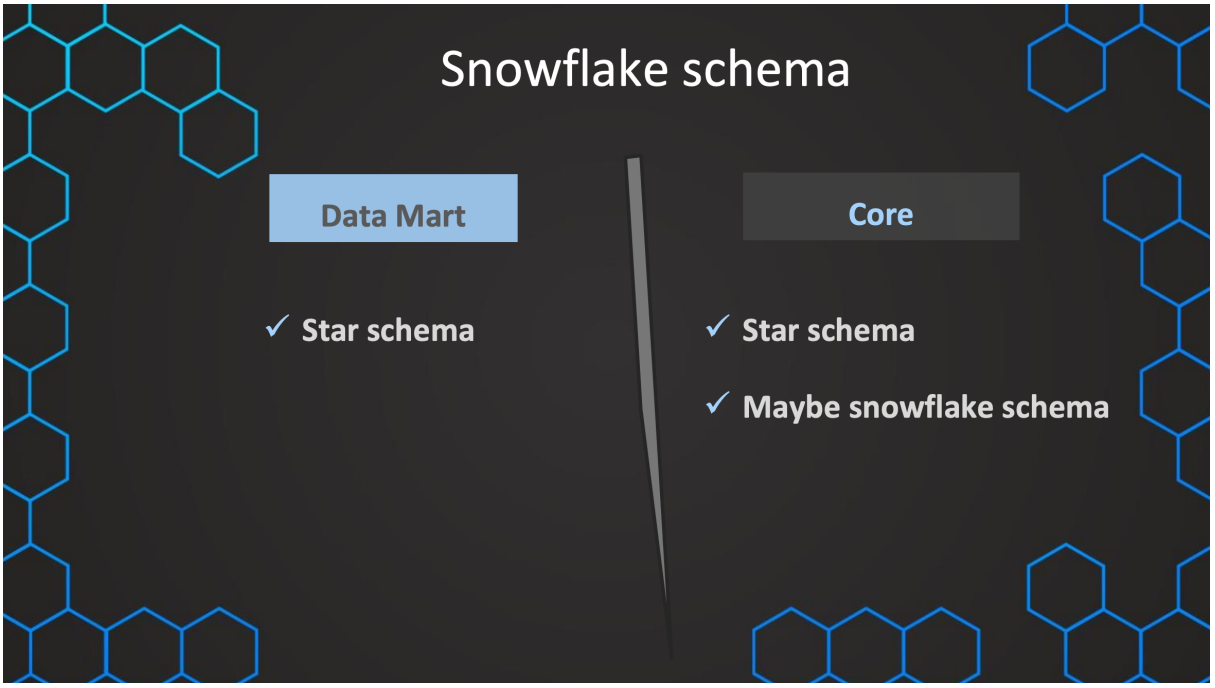
Snowflake schema

- Star schema is a special case of snowflake schema
- Snowflake schema will have multiple hierarchy, so we can say Star schema is just Snowflake schema with 1 level hierarchy
- With Star schema we have data redundancy which we accept over higher read performance and usability
- Snowflake schema is just completely normalizing the data at each hierarchy level so there won't be any data redundancy


Star schema
sales_id product_id customer_id units price
1 3 23 1 2.99
2 5 13 1 1.99
3 2 7 2 3.49
4 3 16 1 2.29
5 3 13 5 1.49
✓ Facts
✓ Dimensions
product_id name category sub_category
1 Chili Herbs Spices
2 Garlic Fruits & Vegetables Vegetable
3 Banana Fruits & Vegetables Fruits
4 Chocolate Sweets & Snacks Sweets
5 Chips Sweets & Snacks Snacks


Snowflake schema
✓ Facts
sales_id product_id customer_id units price
1 3 23 1 2.99
2 5 13 1 1.99
3 2 7 2 3.49
4 3 16 1 2.29
5 3 13 5 1.49
product_id name category_id sub_category
1 Chili 1 Spices
2 Garlic 2 Vegetable
3 Banana 2 Fruits
4 Chocolate 3 Sweets
5 Chips 3 Snacks
Snowflake schema
(More) normalized
category_id category
1 Herbs
2 Fruits & Vegetables
3 Sweets & Snacks





- But if we have really some problems with some write operations, which makes it difficult to maintain, or if the storage cost in some very rare cases can be a challenge, then we can also use the Snowflake schema in the core.

Quiz

Question 1:

What can reduce the query performance in a snowflake schema?

- (●) **Many tables and with that more joins**
- ( ) **Less data redundancy**
- ( ) **More data redundancy**
- ( ) **Few tables with less joins**

Question 2:

What is usually not a reason to create a snowflake schema?

- ( ) **Less data storage**
- (●) **More usability**
- ( ) **Can be easier to maintain and update data in some cases**

Question 3:

Which of the two schema types is more normalized?

- (●) **Snowflake schema**
- ( ) **Star schema**
- ( ) **There is no difference in terms of normalization**

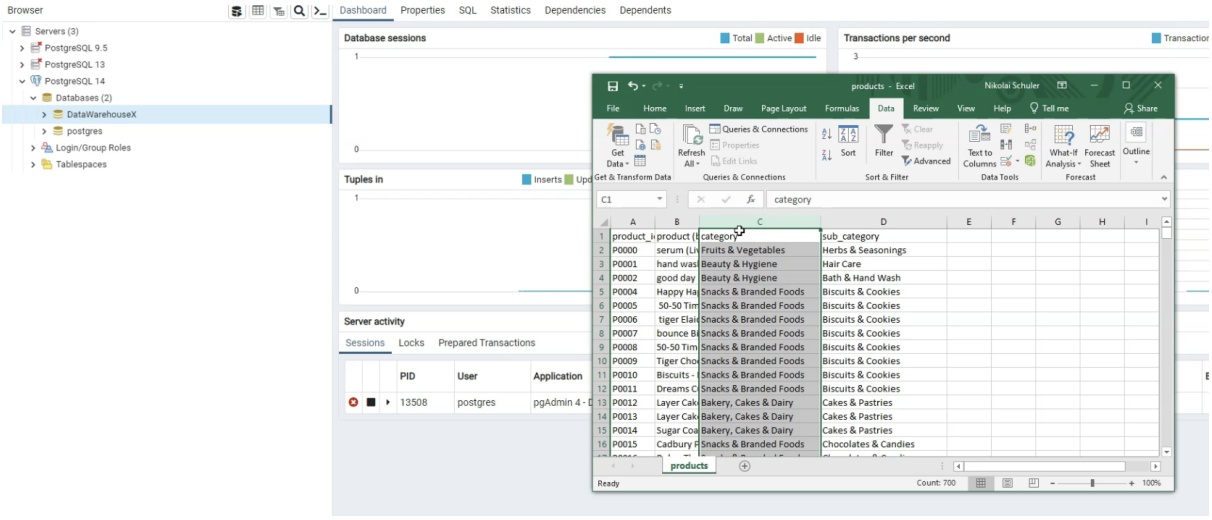
Question 4:

What is not a goal when creating a star schema?

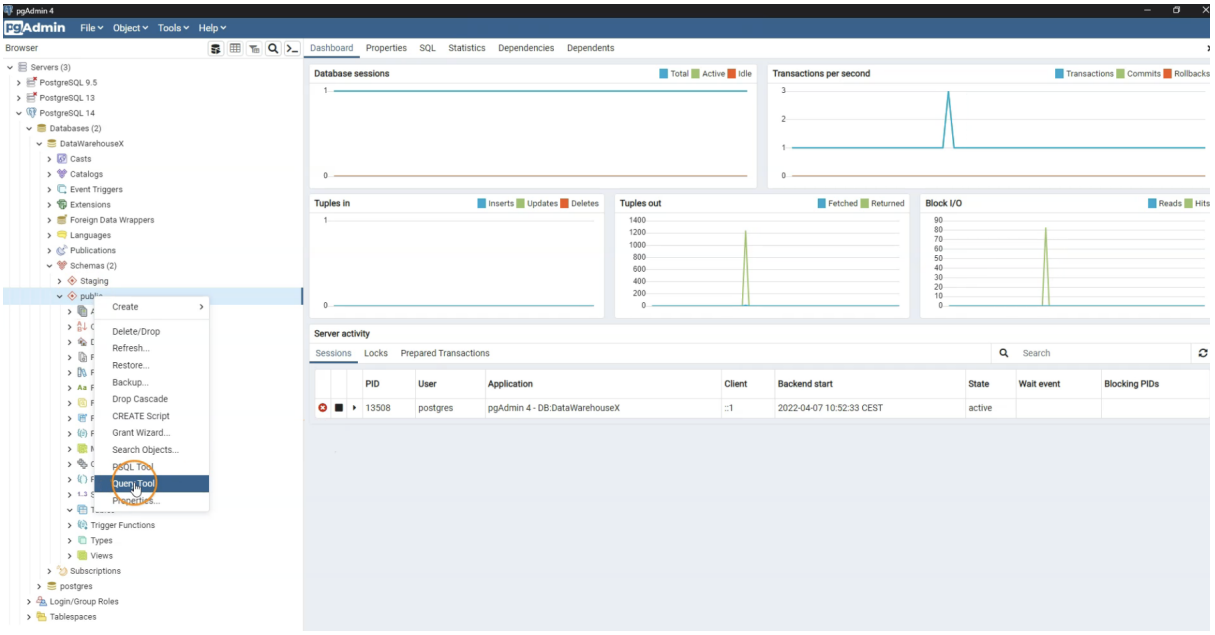
- ○ Improving performance with read queries
- ◉ **Improving performance write operations**
- ○ Improving usuablity

Demo: Product and category dimension

- Let’s see an example of how we can create a snowflake dimension out of product table



- First we will import the data into Product table and then we will create a second table for Category
- Open Query tool from pgAdmin

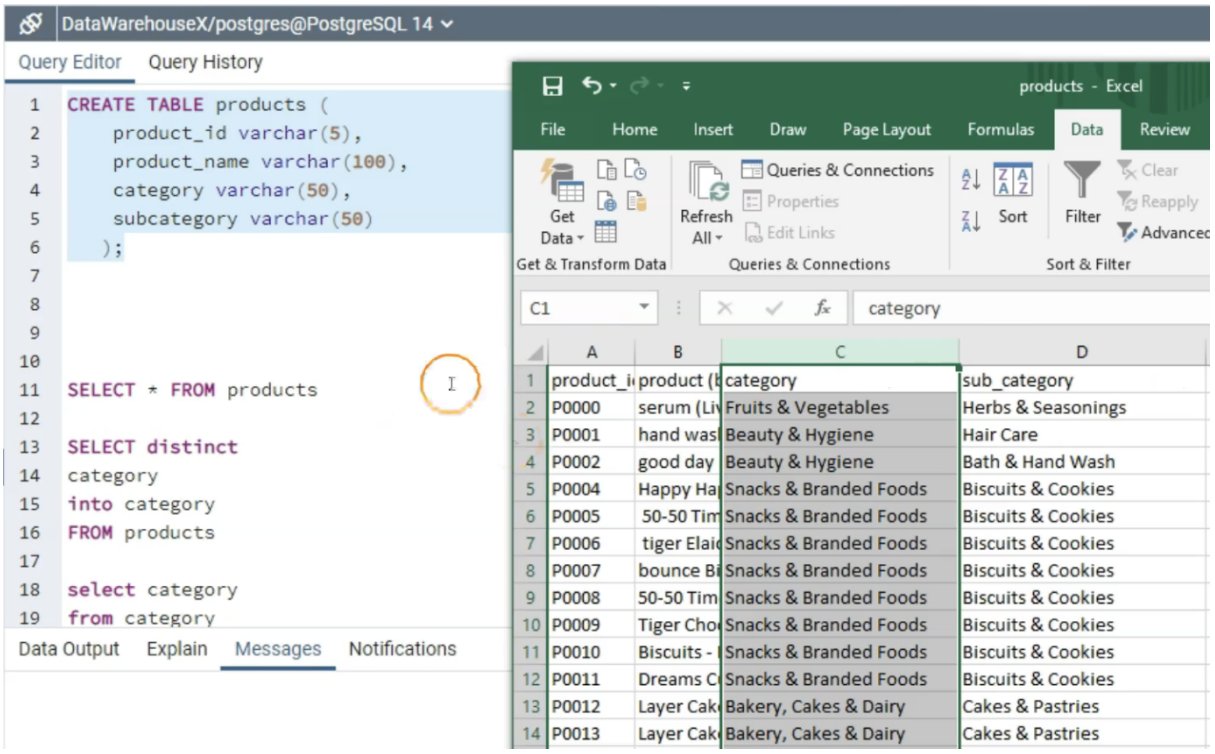


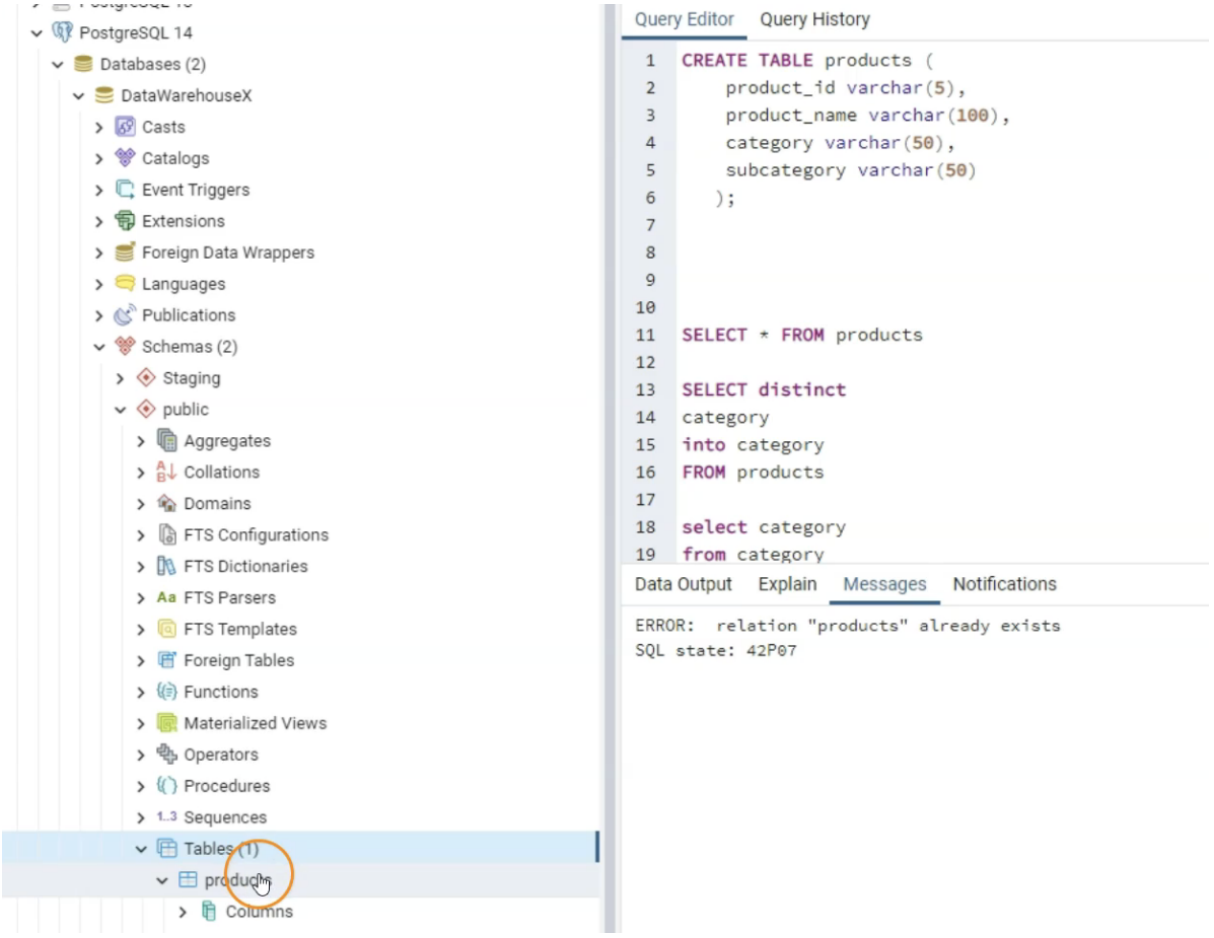
- Use below code to paste in Query editor

```
CREATE TABLE products (
    product_id varchar(5),
    product_name varchar(100),
    category varchar(50),
    subcategory varchar(50)
   );

SELECT * FROM products

SELECT distinct
category
FROM products

SELECT distinct
category
into category
FROM products

select category
from category
```

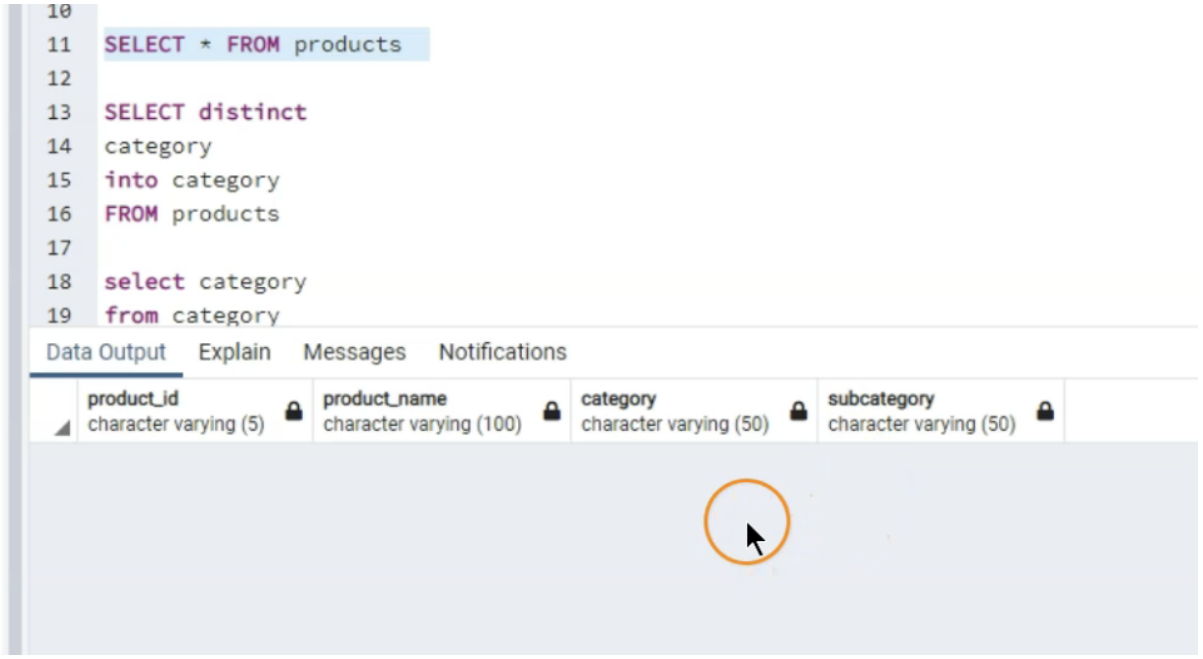
```
select
category
,row_number() over (order by category)
from category


select
row_number() over (order by category) as category_id
,category
from category

select
row_number() over (order by category) as category_id
,category
into category
from category


select
row_number() over (order by category) as category_id
,category
into category_table
from category


select * from
category_table
```

- Create Products with 4 columns

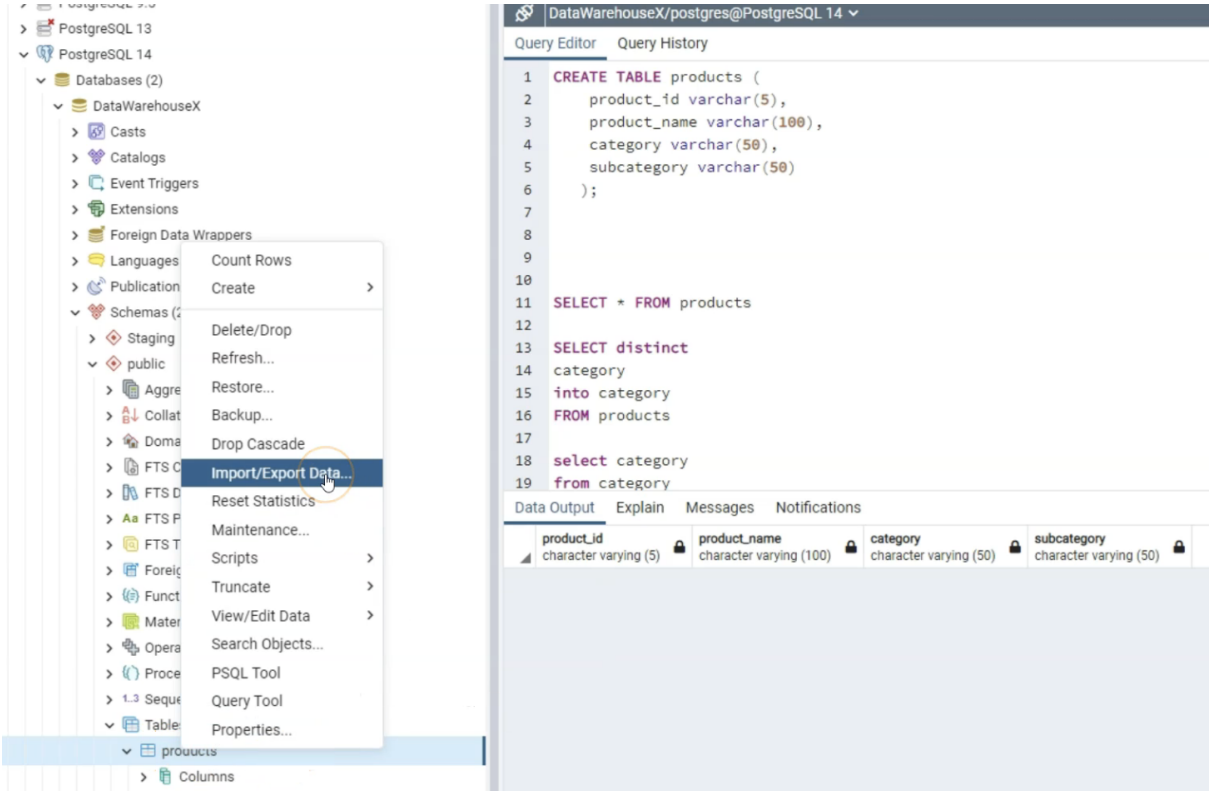


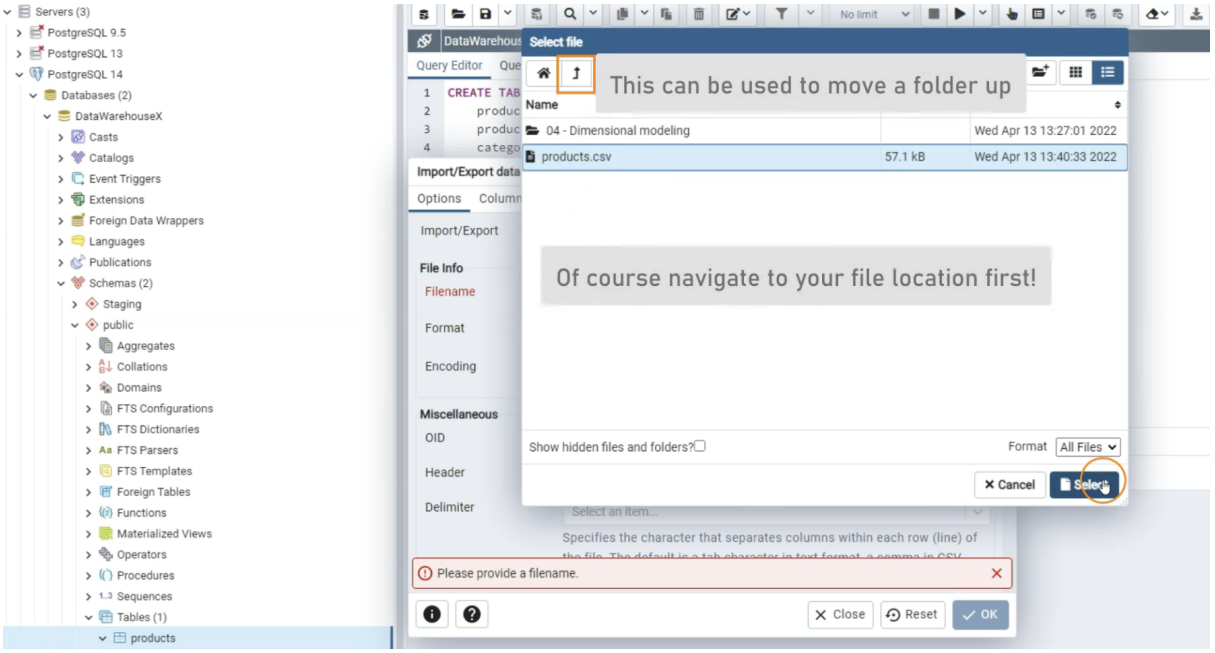


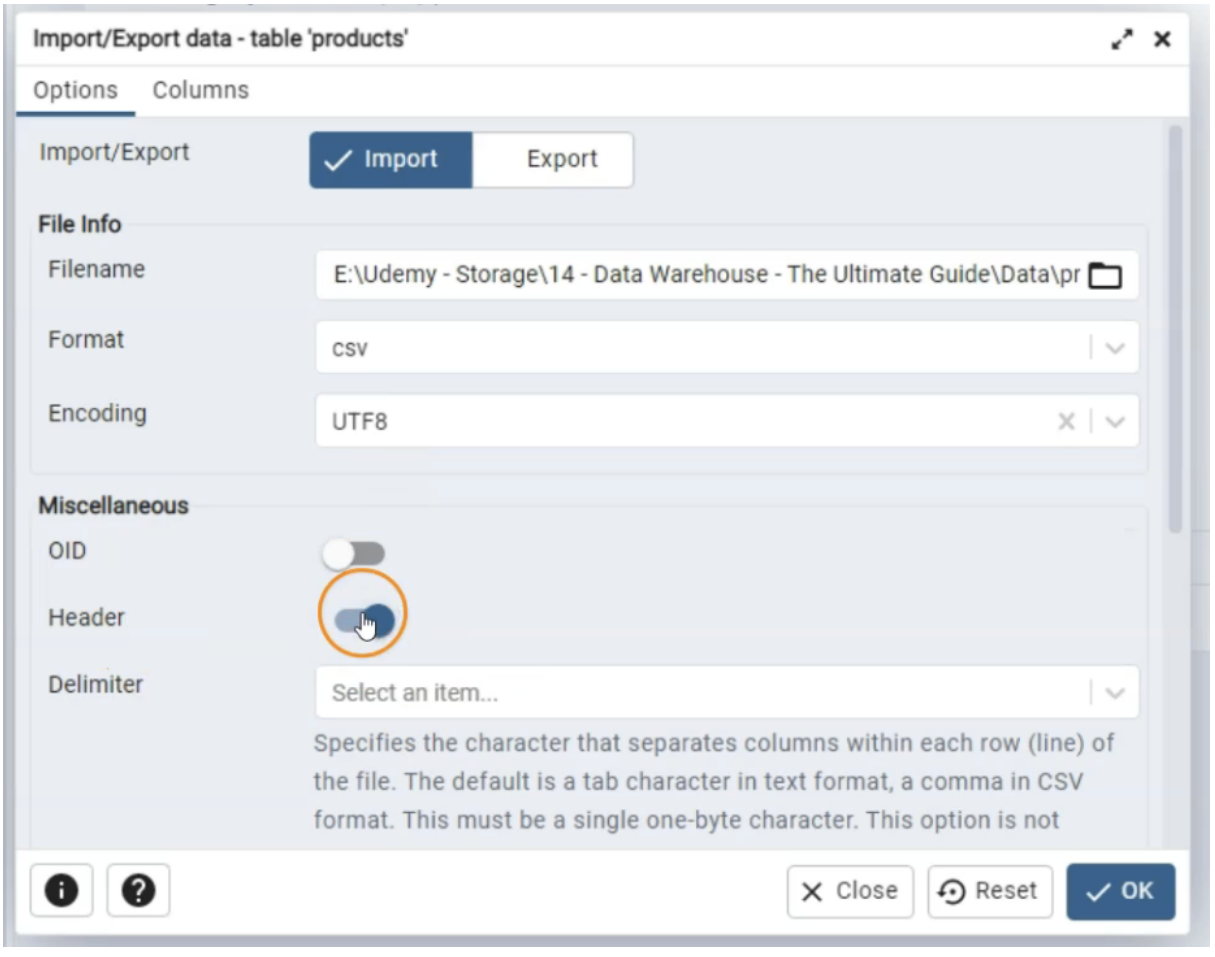
- Query the table and ofcourse there is no data in the table



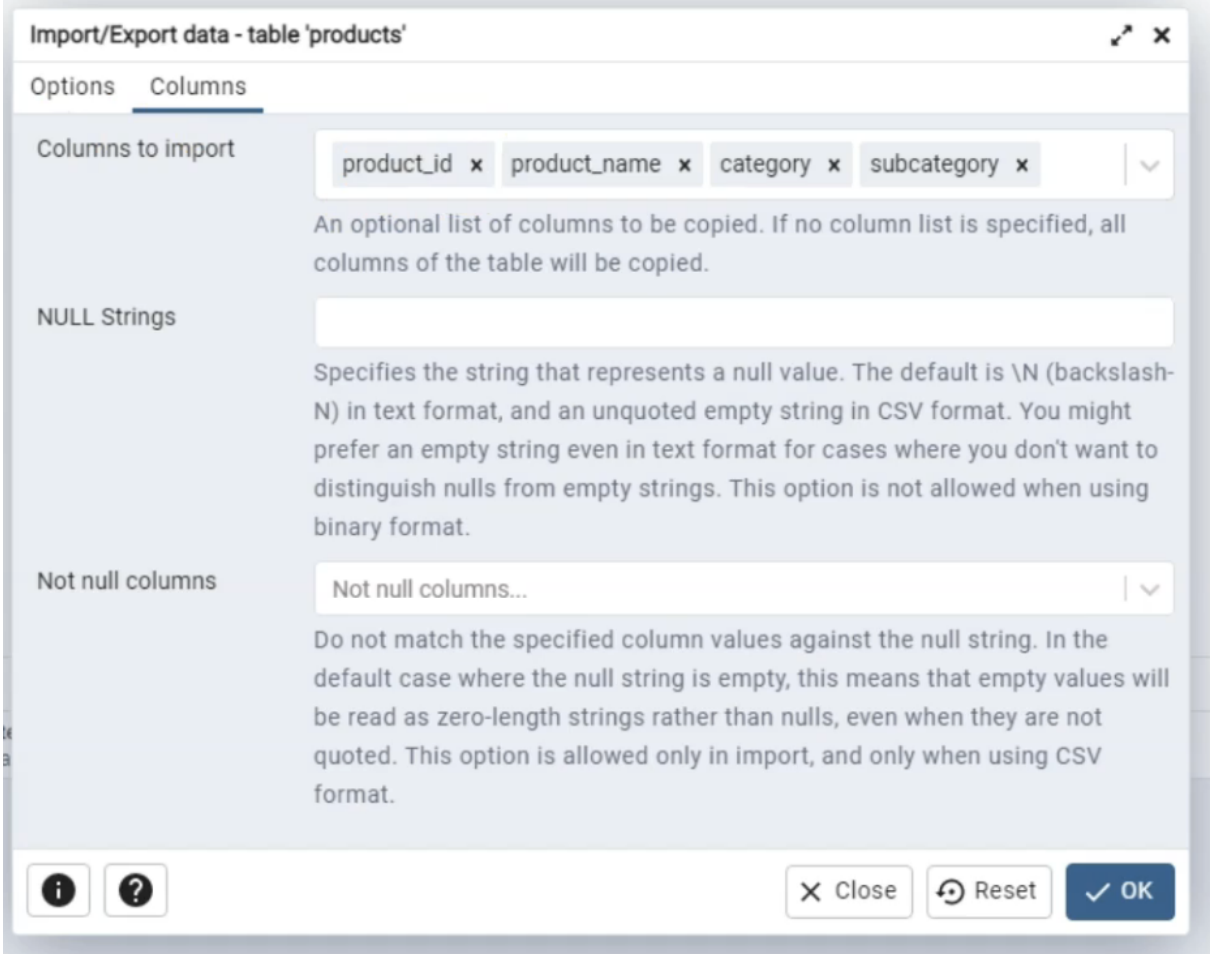
- Import the csv file

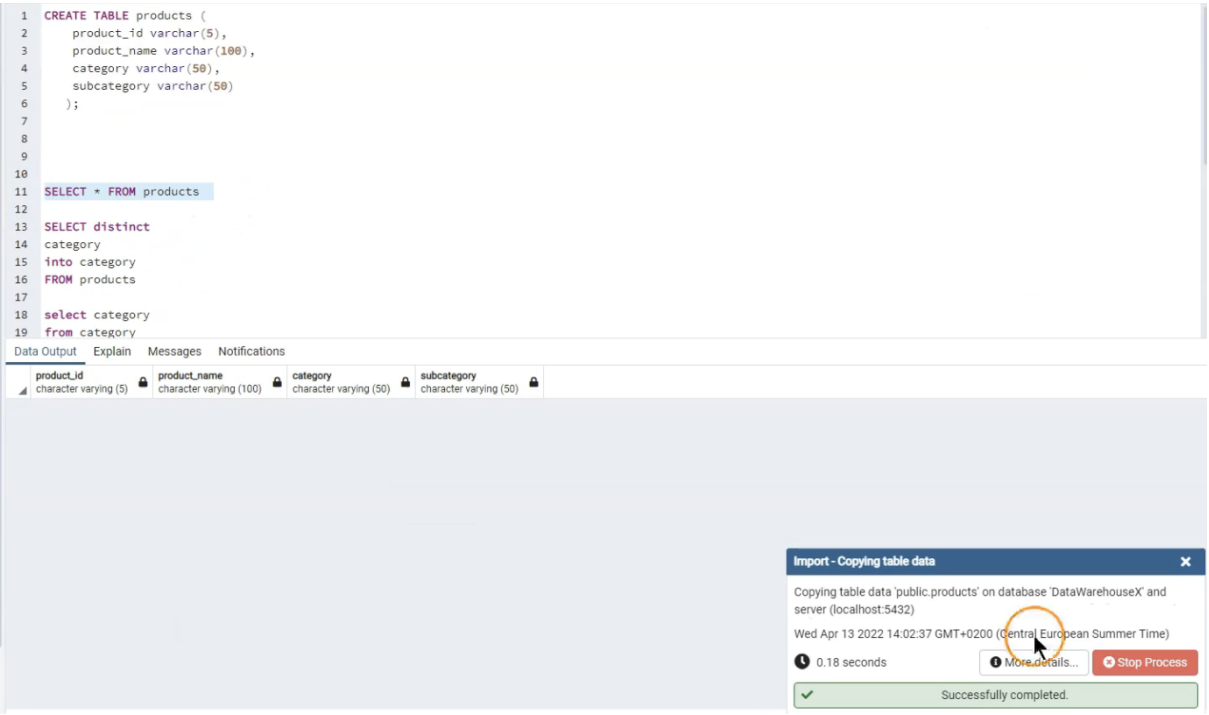






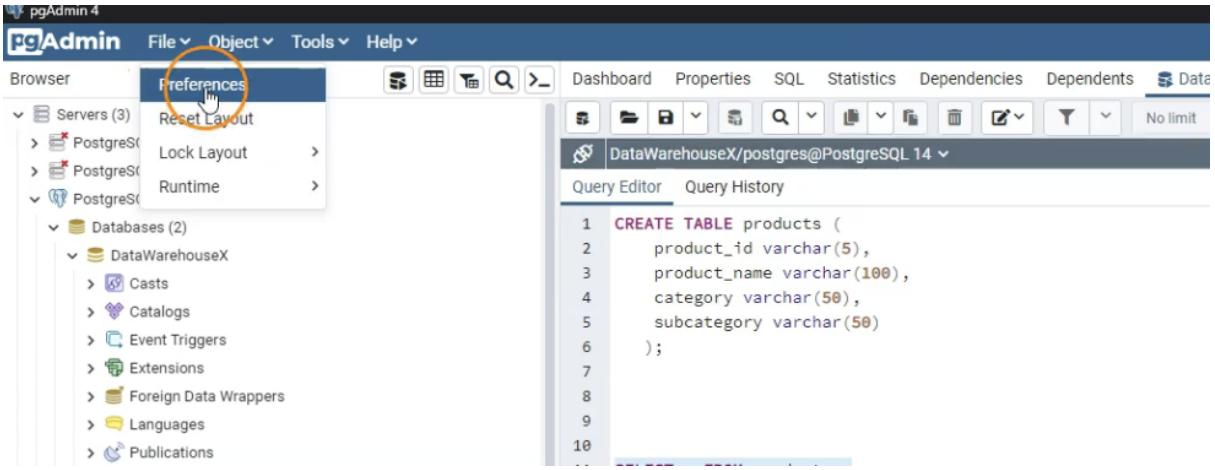
- Select the required columns

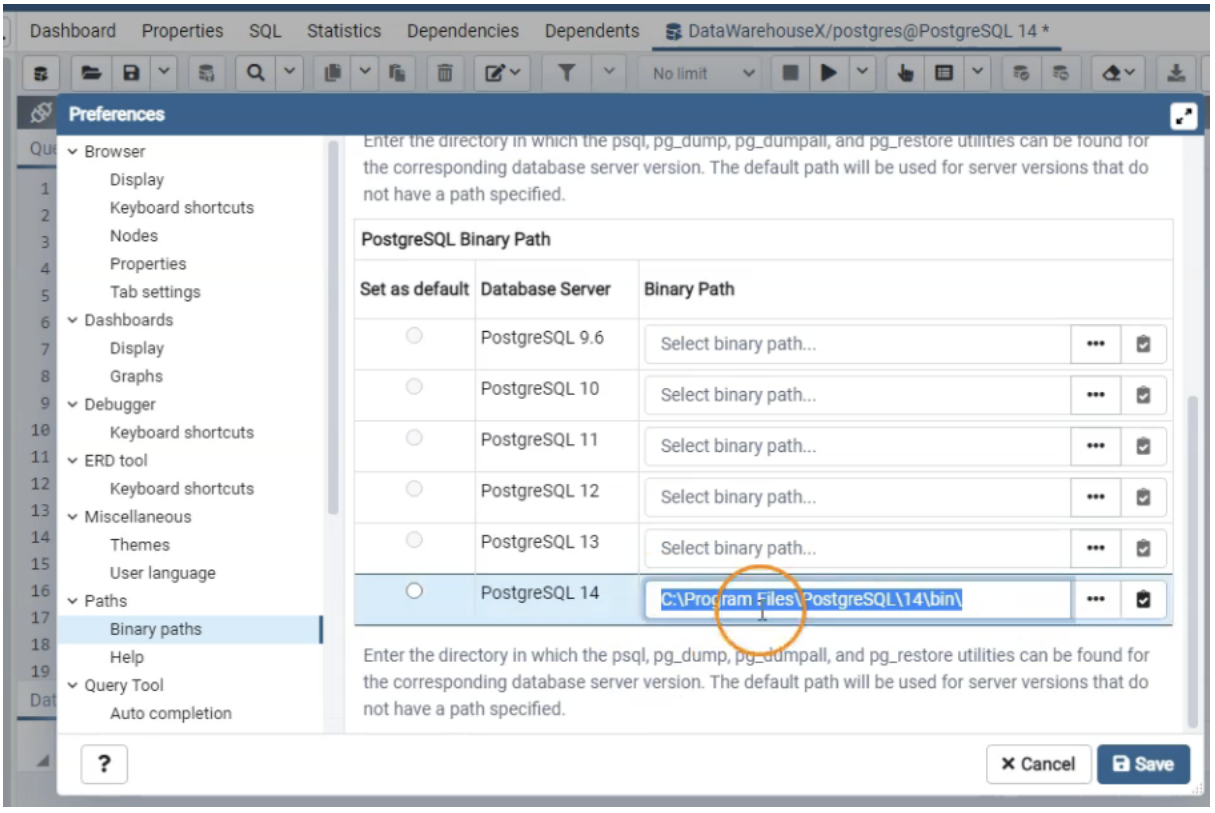




- Sometimes we get error during import usually it will be due to binary path of the file

- To solve that go to File > Preferences





- Under Binary Path, under Postgresql binary path, select the bin path of Postgresql and Save
- Now query again after refresh

```
11  SELECT * FROM products
12
13  SELECT distinct
14  category
15  into category
16  FROM products
17
18  select category
19  from category
```

Data Output | Explain | Messages | Notifications

	product_id character varying (5)	product_name character varying (100)	category character varying (50)	subcategory character varying (50)
1	P0000	[...] serum (Livon)	Fruits & Vegetables	Herbs & Seasonings
2	P0001	hand wash - moisture Shi...	Beauty & Hygiene	Hair Care
3	P0002	good day butter C...	Beauty & Hygiene	Bath & Hand Wash
4	P0004	Happy Happy Choc...	Snacks & Branded Foods	Biscuits & Cookies
5	P0005	[...] 50-50 Timepass salted...	Snacks & Branded Foods	Biscuits & Cookies
6	P0006	[...] tiger Elaichi Cream Bis...	Snacks & Branded Foods	Biscuits & Cookies
7	P0007	bounce Biscuits - Choco ...	Snacks & Branded Foods	Biscuits & Cookies
8	P0008	50-50 Timepass Biscuits (...	Snacks & Branded Foods	Biscuits & Cookies
9	P0009	Tiger Chocolate Cream Bi...	Snacks & Branded Foods	Biscuits & Cookies
10	P0010	Biscuits - Magix Kreams C...	Snacks & Branded Foods	Biscuits & Cookies
11	P0011	Dreams Cup Cake - Choco ...	Snacks & Branded Foods	Biscuits & Cookies

```
12
13  SELECT
14  category
15  FROM products
16
17  SELECT distinct
18  category
19  into category
```

Data Output | Explain | Messages | Notifications

	category character varying (50)
1	Fruits & Vegetables
2	Beauty & Hygiene
3	Beauty & Hygiene
4	Snacks & Branded Foods
5	Snacks & Branded Foods
6	Snacks & Branded Foods
7	Snacks & Branded Foods
8	Snacks & Branded Foods
9	Snacks & Branded Foods
10	Snacks & Branded Foods
11	Snacks & Branded Foods

- Use Distinct to view all records only once

```
12
13  SELECT distinct
14  category
15  FROM products
16
17  SELECT distinct
18  category
19  into category
```

Data Output | Explain | Messages | Notifications

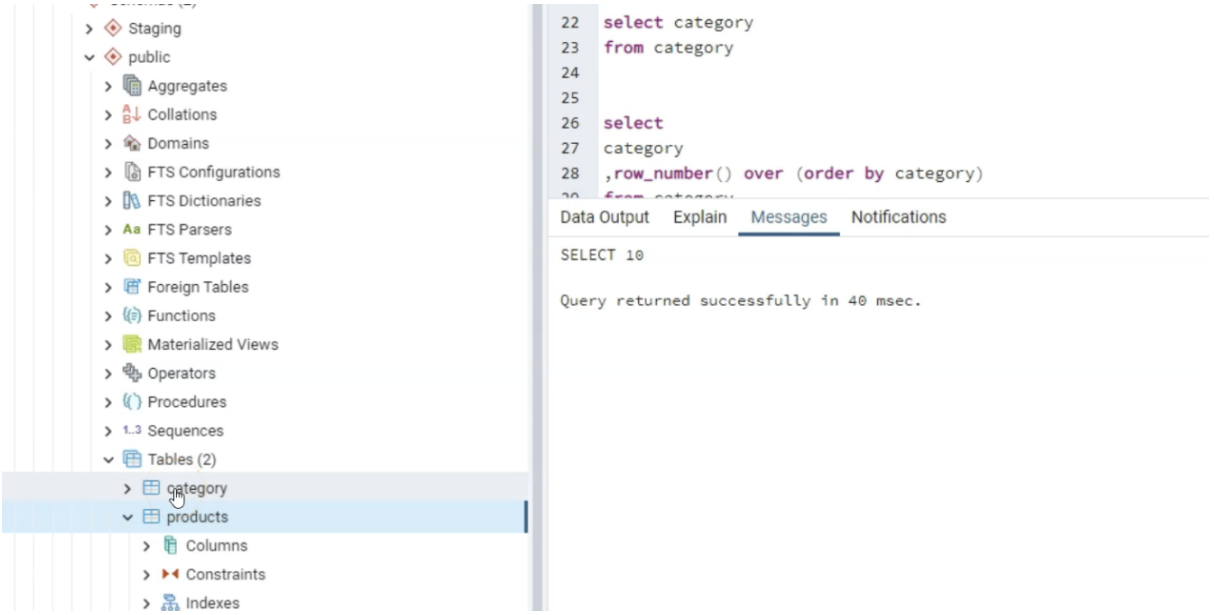
	category character varying (50)
1	Baby Care
2	Fruits & Vegetables
3	Kitchen, Garden & Pets
4	Snacks & Branded Foods
5	Gourmet & World Food
6	Cleaning & Household
7	Beauty & Hygiene
8	Bakery, Cakes & Dairy
9	Foodgrains, Oil & Masala
10	Beverages

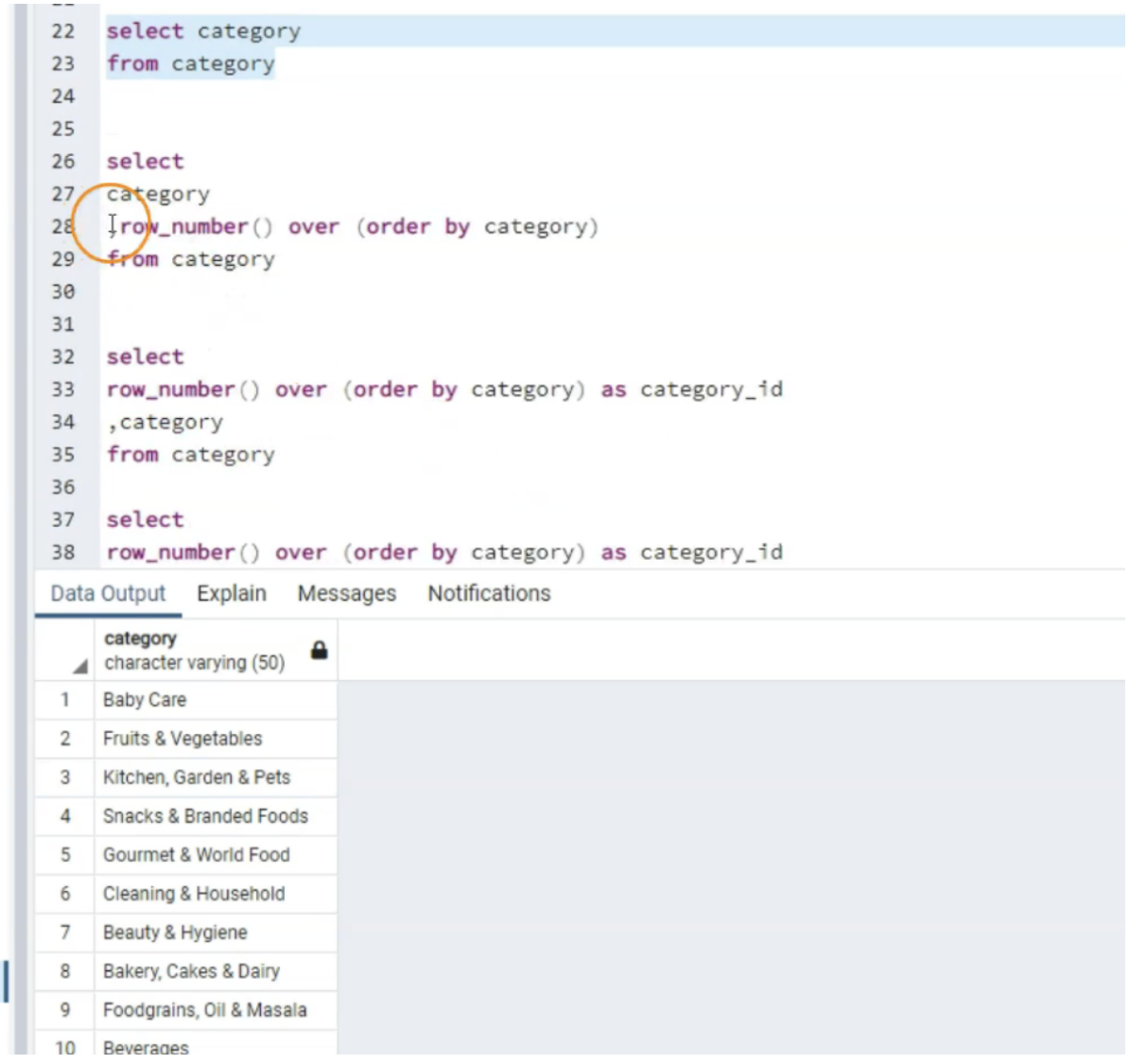
- Insert the distinct values of category into Category table

```
16
17  SELECT distinct
18  category
19  into category
20  FROM products
21
22  select category
23  from category
24
25
26  select
27  category
28  ,row_number() over (order by category)
```

Data Output | Explain | Messages | Notifications

```
SELECT 10

Query returned successfully in 40 msec.
```





- We should have a ID column

```
26  select
27  category
28  ,row_number() over (order by category)
29  from category
30
31
32  select
33  row_number() over (order by category) as category_id
34  ,category
35  from category
36
37  select
38  row_number() over (order by category) as category_id
```

Data Output | Explain | Messages | Notifications

	category character varying (50)	row_number bigint
1	Baby Care	1
2	Bakery, Cakes & Dairy	2
3	Beauty & Hygiene	3
4	Beverages	4
5	Cleaning & Household	5
6	Foodgrains, Oil & Masala	6
7	Fruits & Vegetables	7
8	Gourmet & World Food	8
9	Kitchen, Garden & Pets	9
10	Snacks & Branded Foods	10

```
32  select
33  row_number() over (order by category) as category_id
34  ,category
35  from category
36
37  select
38  row_number() over (order by category) as category_id
39  ,category
40  into category
41  from category
42
43
44  drop table category
```

Data Output | Explain | Messages | Notifications

	category_id bigint	category character varying (50)
1	1	Baby Care
2	2	Bakery, Cakes & Dairy
3	3	Beauty & Hygiene
4	4	Beverages
5	5	Cleaning & Household
6	6	Foodgrains, Oil & Masala
7	7	Fruits & Vegetables
8	8	Gourmet & World Food
9	9	Kitchen, Garden & Pets
10	10	Snacks & Branded Foods

```
37  select
38  row_number() over (order by category) as category_id
39  ,category
40  into category
41  from category
42
43
44  select
45  row_number() over (order by category) as category_id
```

Data Output | Explain | Messages | Notifications

```
ERROR:  relation "category" already exists
SQL state: 42P07
```

```sql
select
row_number() over (order by category) as category_id
,category
into category_table
from category

select * from
category_table
```

Data Output | Explain | Messages | Notifications

	category_id bigint	category character varying (50)
1	1	Baby Care
2	2	Bakery, Cakes & Dairy
3	3	Beauty & Hygiene
4	4	Beverages
5	5	Cleaning & Household
6	6	Foodgrains, Oil & Masala
7	7	Fruits & Vegetables
8	8	Gourmet & World Food
9	9	Kitchen, Garden & Pets
10	10	Snacks & Branded Foods