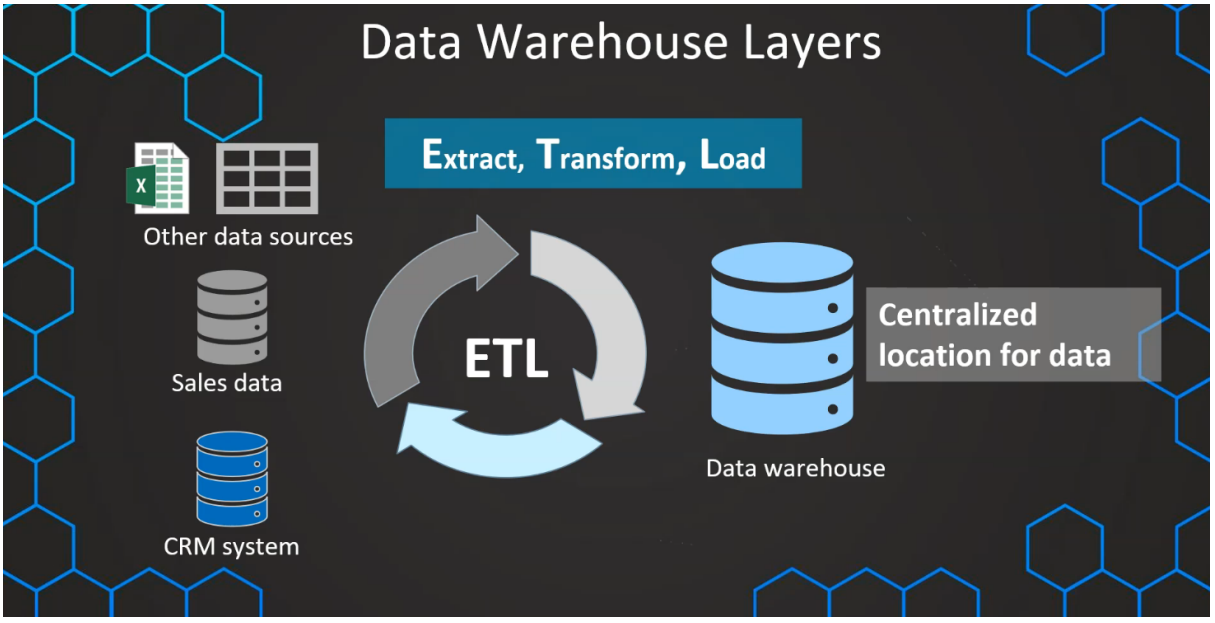
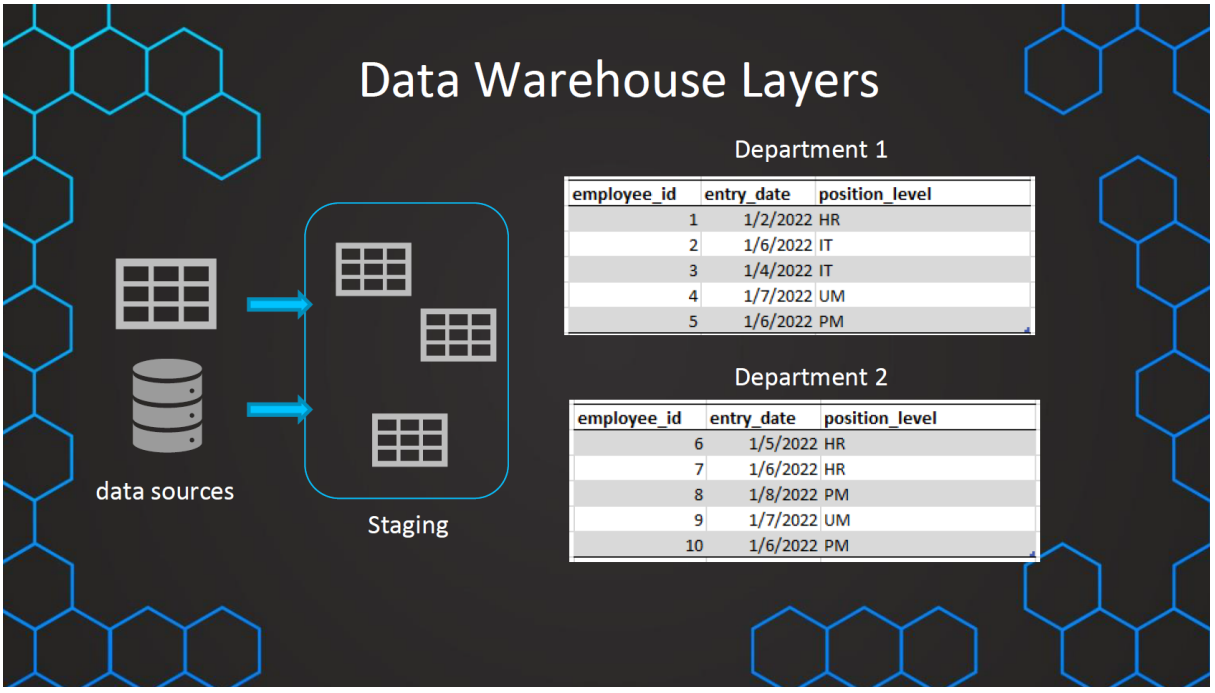
04. Data warehouse architecture

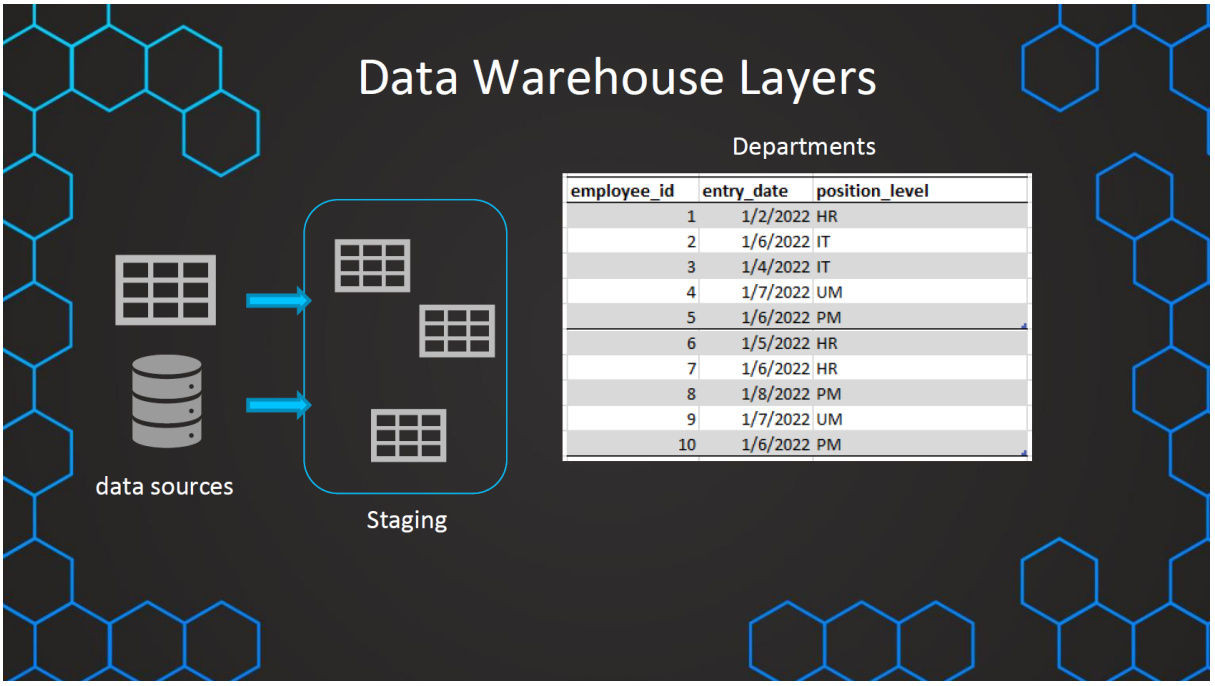
- Data warehouse consists of multiple layers, we have our data sources, through an ETL process we transform these data and load it into data warehouses



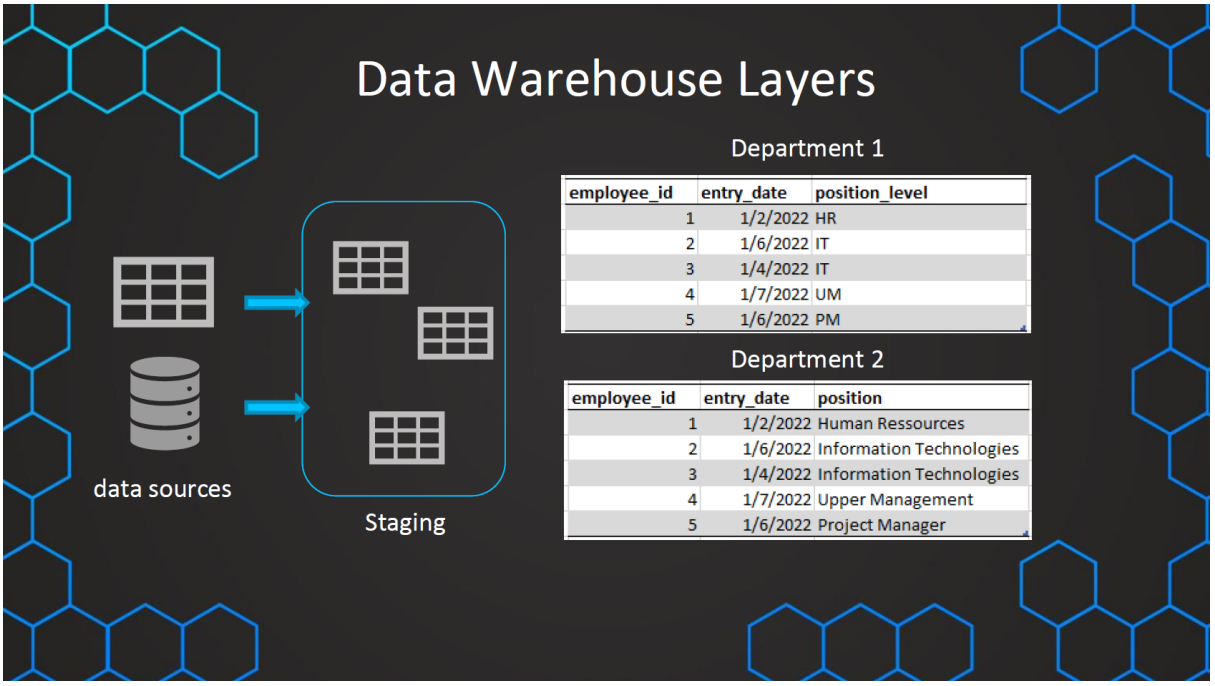
- In beginning of ETL process we have source data and use the ETL tool to extract these different formats and files



- We use our ETL tool to extract these different formats and these different files into our first layer of data warehouse and this is the so-called "staging layer".
- Staging layer is there just to extract the tables as they are into this first layer and we want to place it into tables. So, they are now all in tables and we didn't do any data transformation.
- We want to leave it as untouched as possible.
- So, for example, let's say we have employee tables in different departments and they are all in different formats, like CSV files, some are in databases, and so on and we want to extract them all into tables, into our staging area.
- Staging layer is just raw data from different file formats/different sources into tables



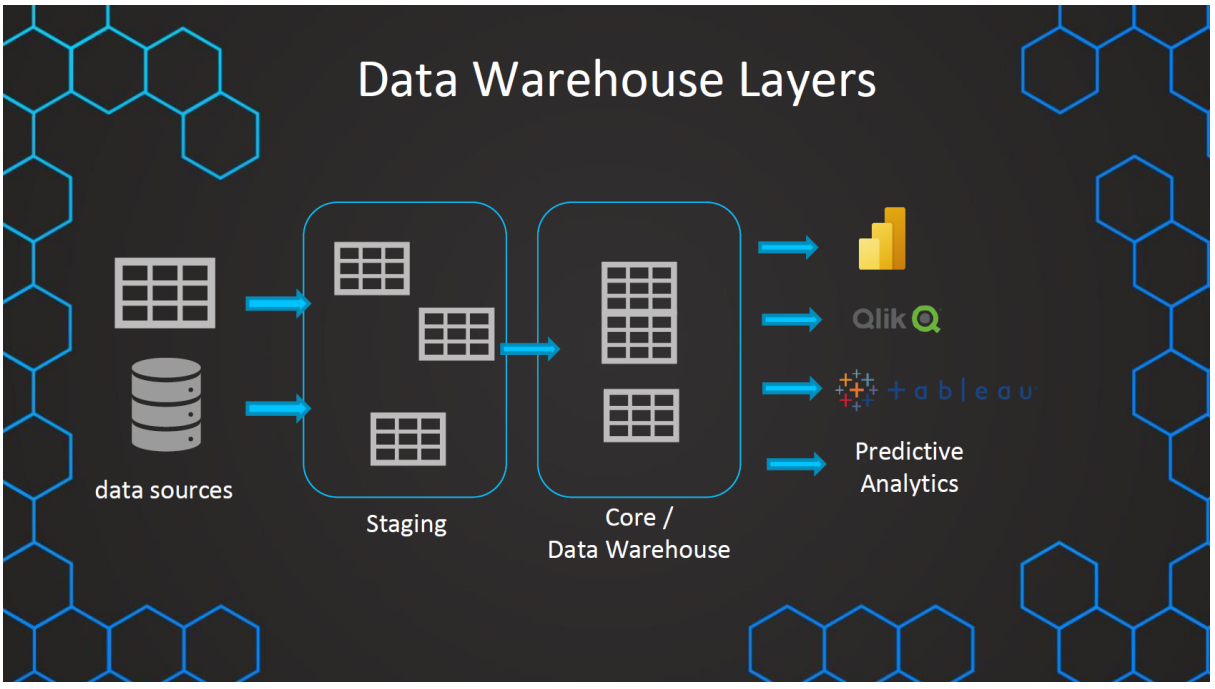
- We can have the data into multiple tables in our staging area or we can do some light transformation and logic to put the data into 1 table as the schema looks same for the data coming from 2 sources
- There can be some small transformation steps such as combining data (appending) from multiple tables



- The data can be little different coming from different sources like these 2 tables having position_level column with abbreviation data and another

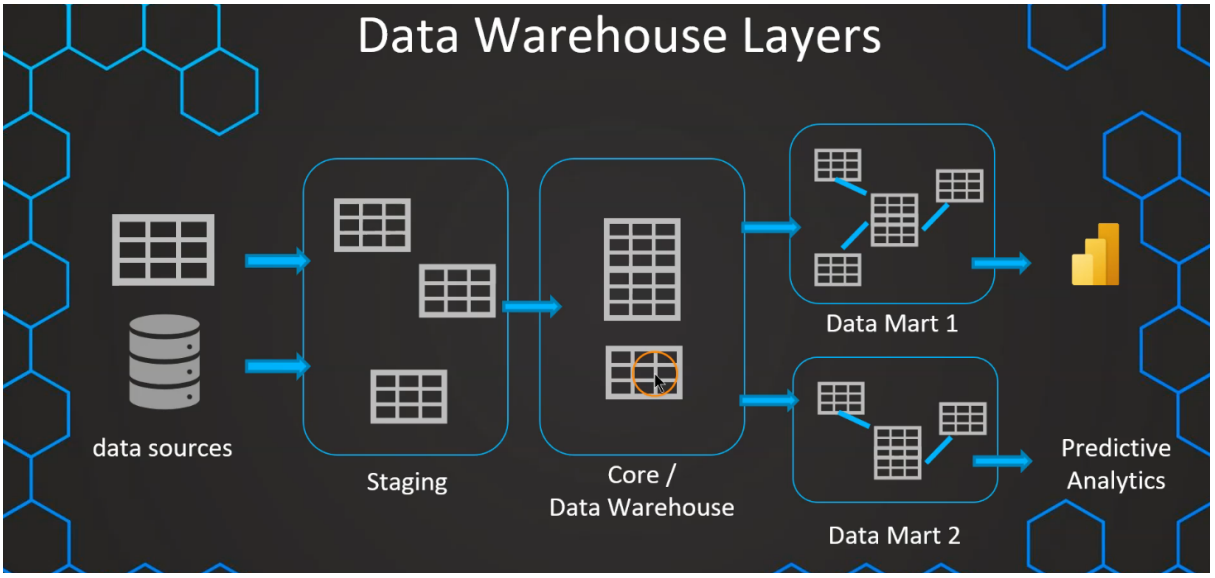
table having full form values column called position, and the ID column have similar IDs instead of consecutive like the one above image

- Here we can do some small transformations to integrate the data into good tables for our staging layer of data warehouse
- We want to model the data as we want to have it. So, oftentimes, we want to restructure that data which we can do using our ETL tool which we can do while copying the data from Staging to Core layer of data warehouse.
- The Core layer is sometimes considered as the data warehouse itself as we do some transformations while we copy the data from staging layer to core layer
- The Core layer can also be used by end user to do reporting and predictive analysis

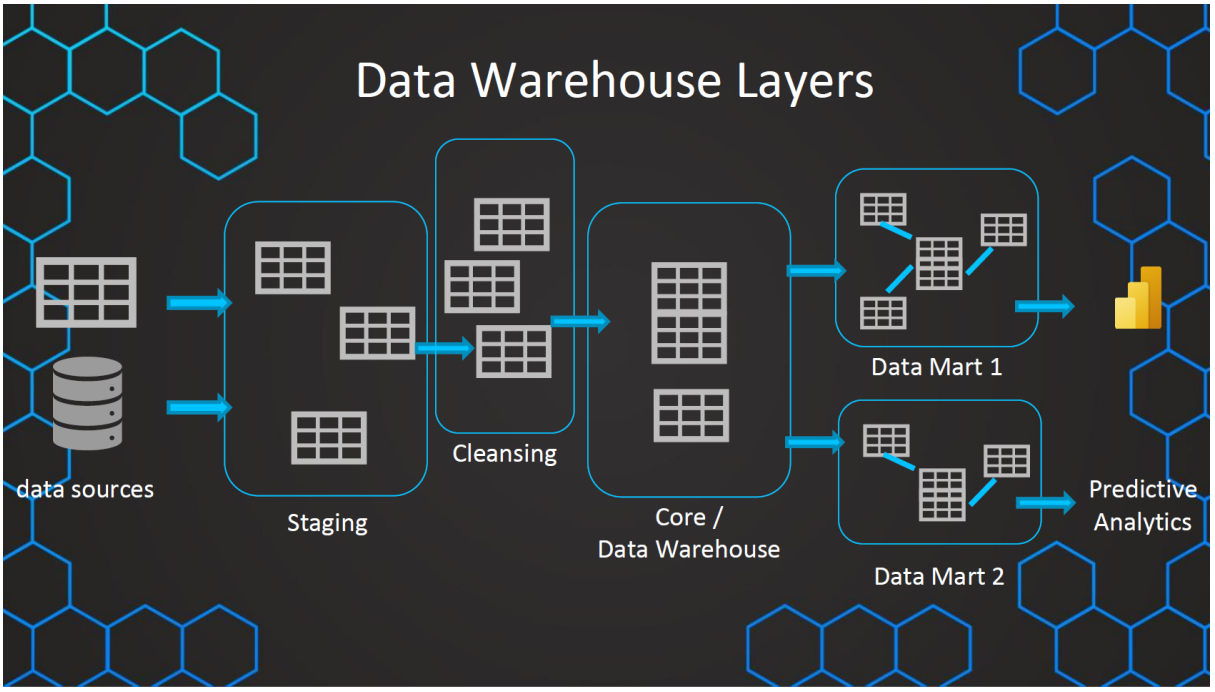


- Sometimes if we have large data warehouse which consists of many tables and we have many different use cases we can build **data marts** on top the core layer
- Data mart just takes the tables which are relevant to one specific use case like reporting or predictive analysis or data mining
- Data mart increases user friendliness as the user is not overwhelmed with so many tables and can also help with query performance as we only the tables which we need

- We can use specialised data bases like in-memory databases, cubes to increase the performance
- Data marts are optional and not always necessary

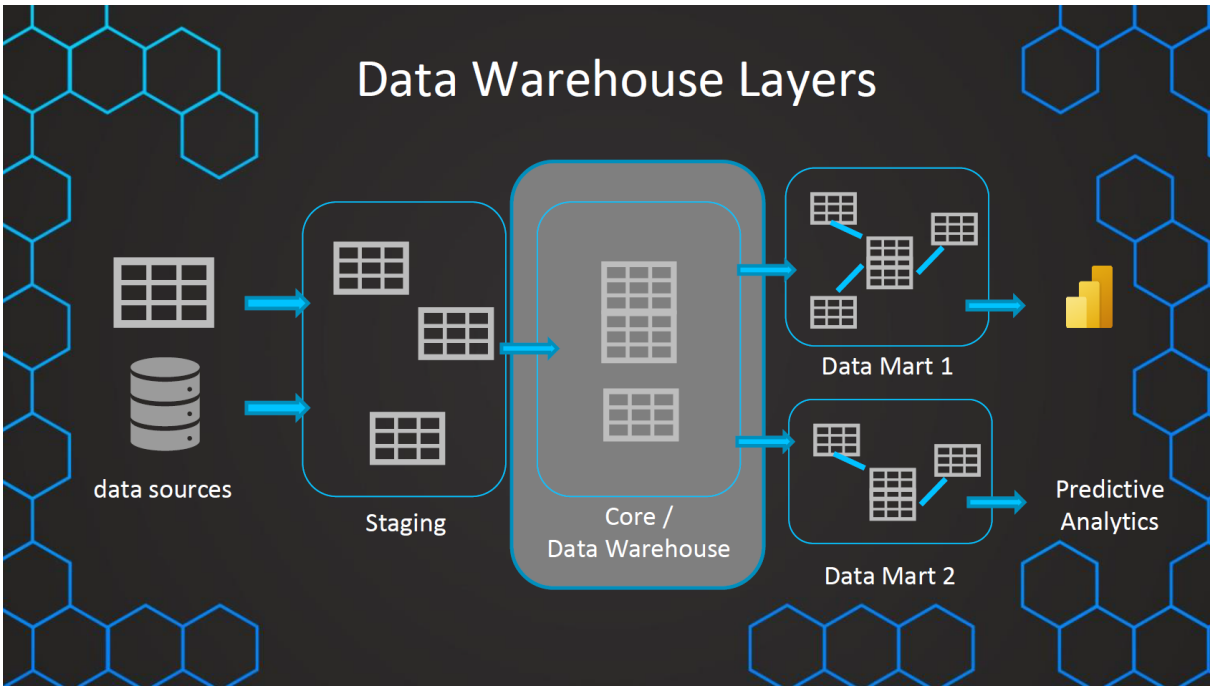


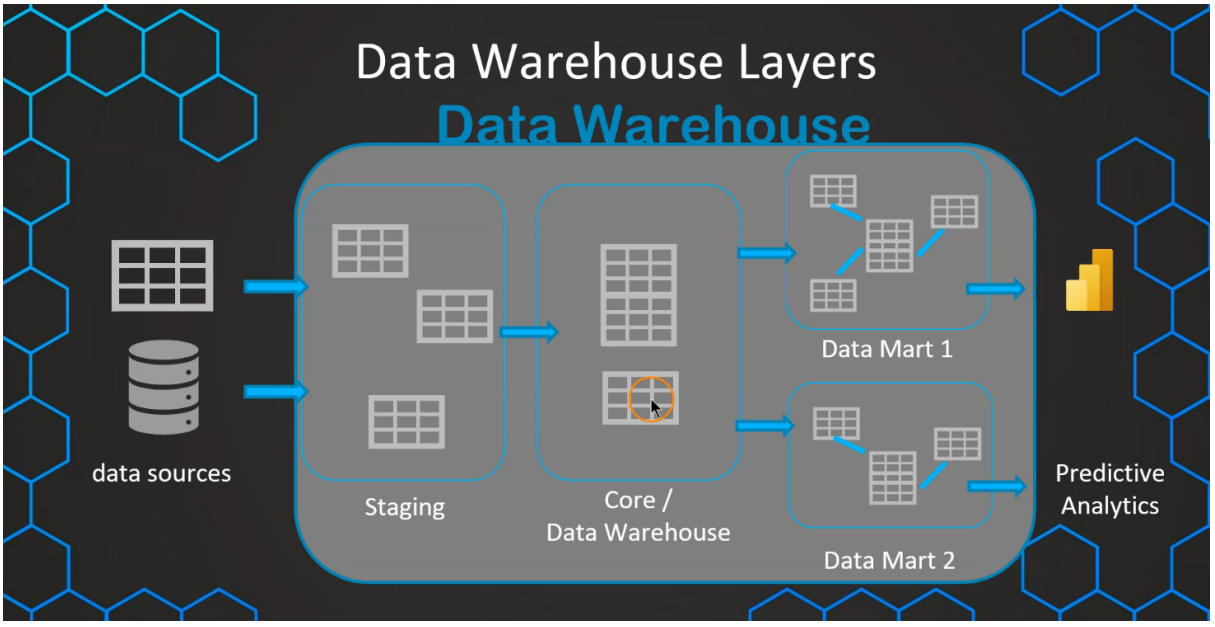
- Another optional layer is cleansing area, if we need to do lot of data cleaning and do some validations before pushing the data into core data warehouse layer



- Just the core layer can be called data warehouse or the layer from staging, core layer and data marts can be considered as the complete data

warehouse

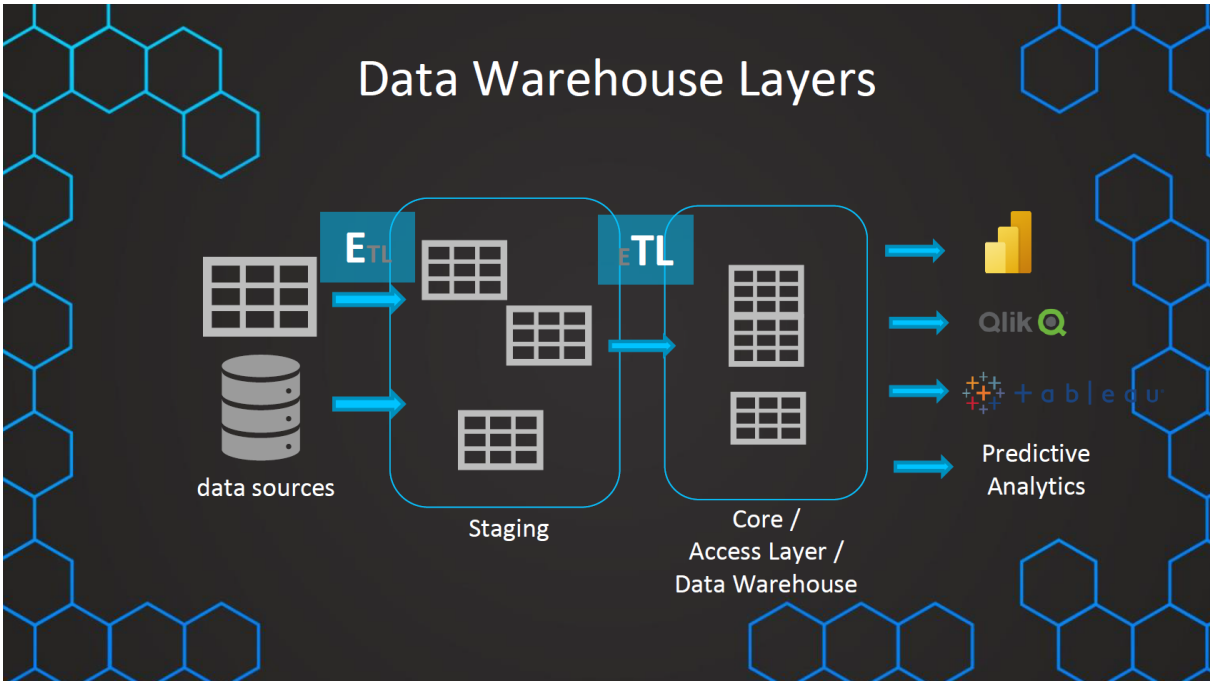




Staging area

- The goal of staging area is just to extract the data from our data sources, we just need read access to our data sources to fetch the data and push it into our staging layer of data warehouse
- Staging layer will usually be nice and neat database tables unlike source which can be csv, json, text, another database, etc

- After the data reaches staging layer, we can safely do our transformations using our ETL tool and load the transformed data into our core layer of data warehouse
- The core layer is accessible to end users and sometimes perceived as data warehouse

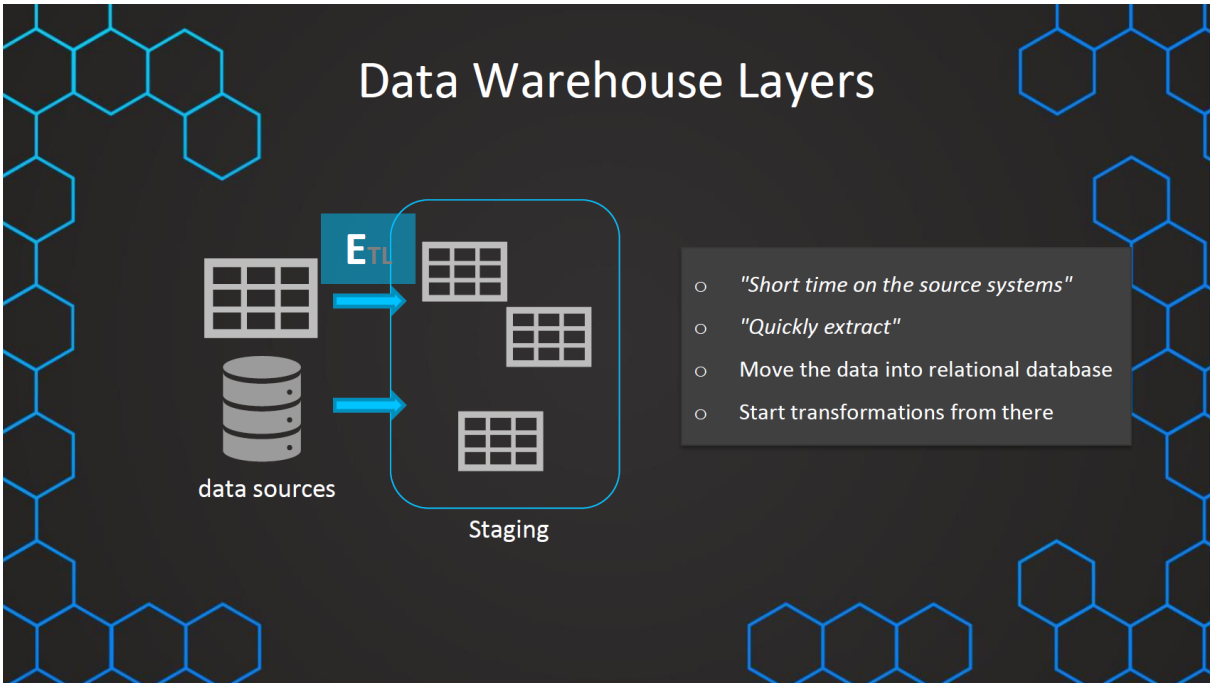


Why do we need a staging layer?

- Doesn't it create redundant data?
- Doesn't it make thing more complicated by adding additional layer to the data warehouse

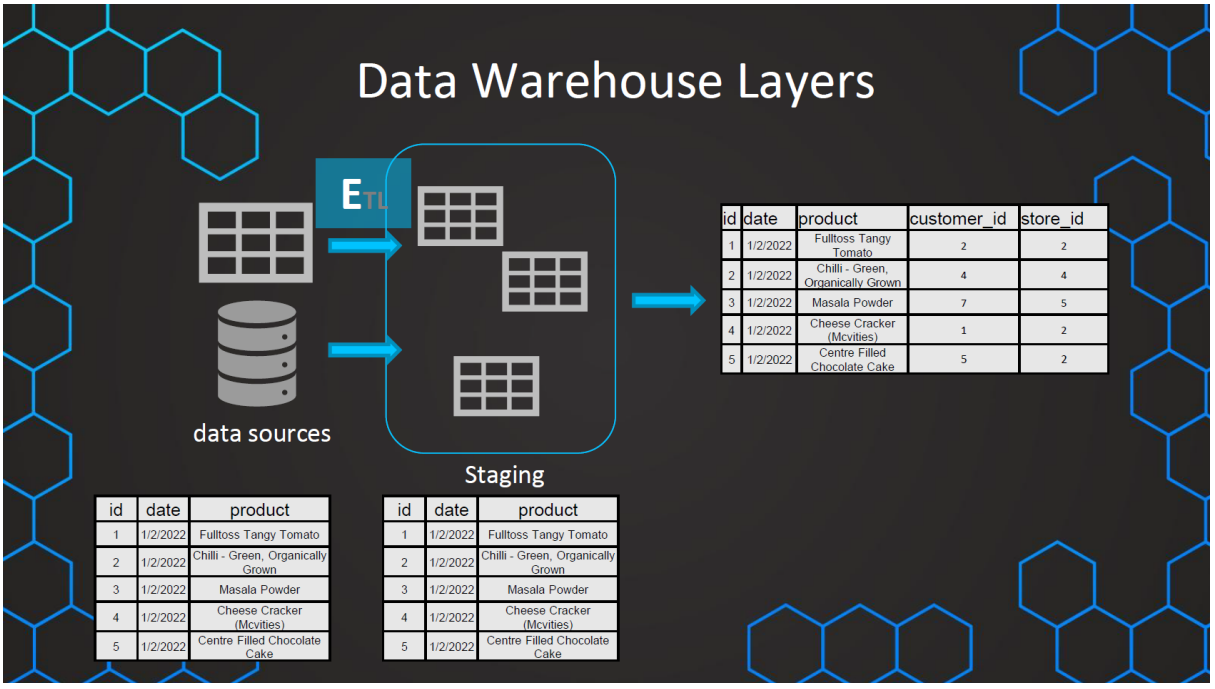
Advantages

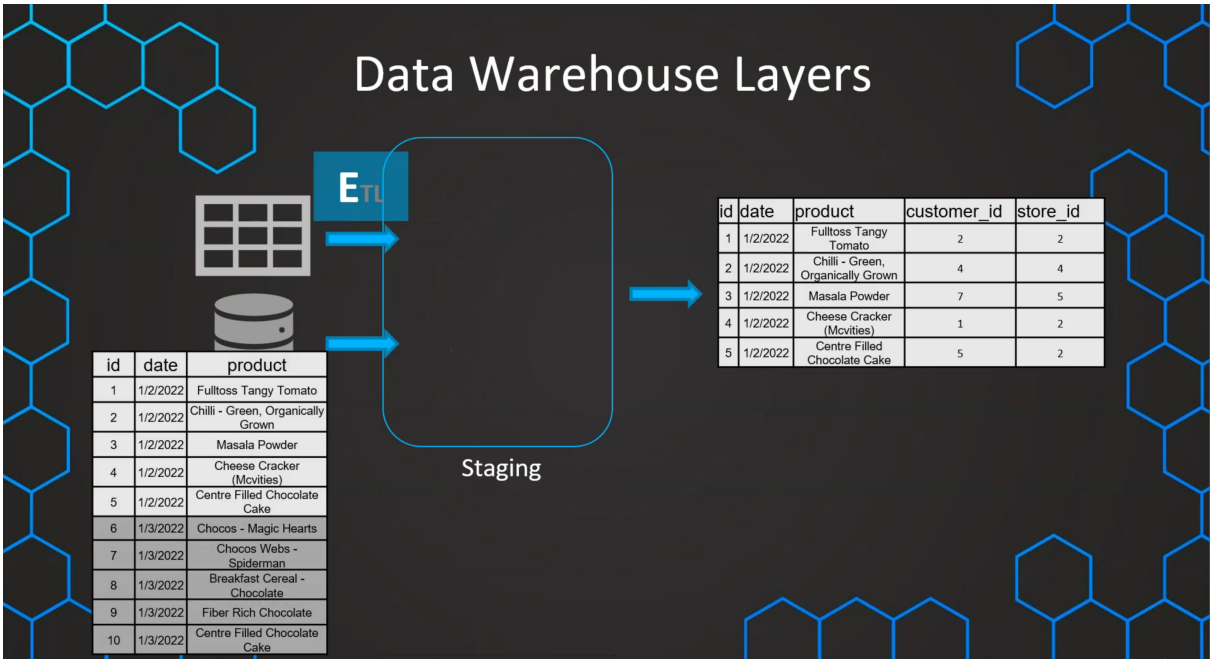
- We are pulling the data of operational systems, which is actually keeping the wheels turning by running up the business, hence we do not want to spend too much time with these systems which could affect the performance of these operational systems
- We also need the data in table format, the data in data sources can be coming from various data sources types
- Once we have the data in safe staging area and all in tables format we can then start to apply business transformation



Example

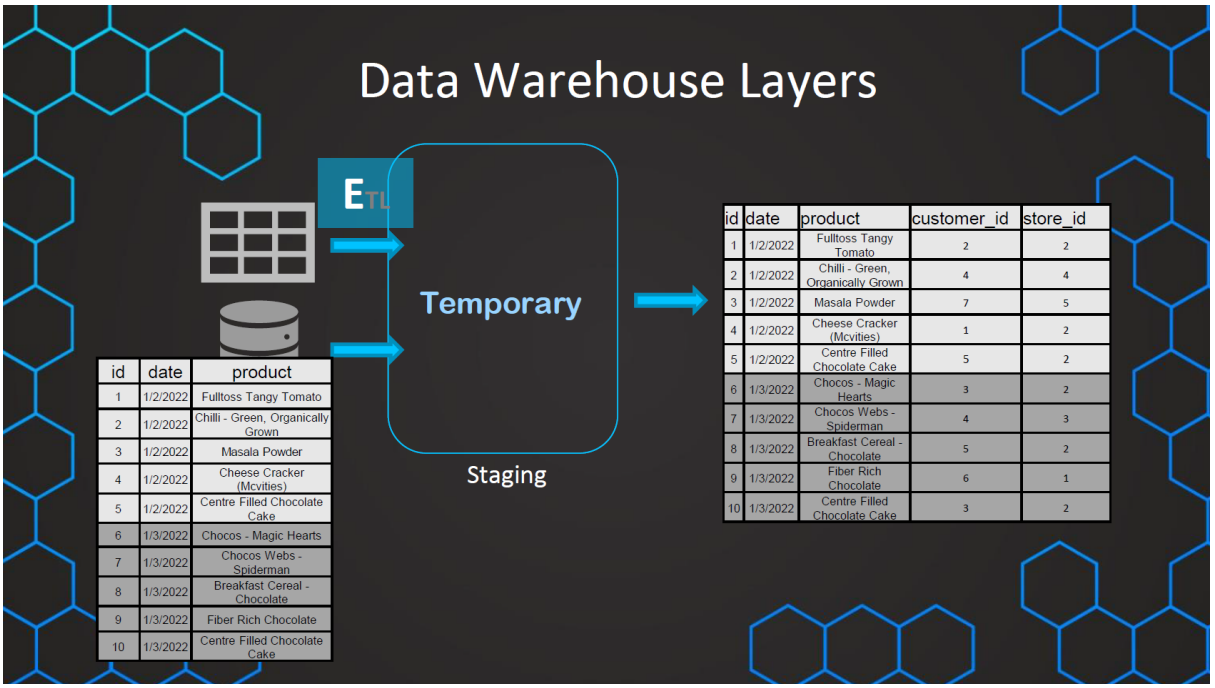
- After every cycle we truncate the staging layer and load the data from data sources
- To identify the new data in the data sources, we need to have delta logic in place for this we need to find a column which tells if this data is new like an ID column
- With ID column we have to have a strict ascending numerical column to identify proper delta records
- Usually a date column is delta column, we can use the last date in our system and load the data after that date

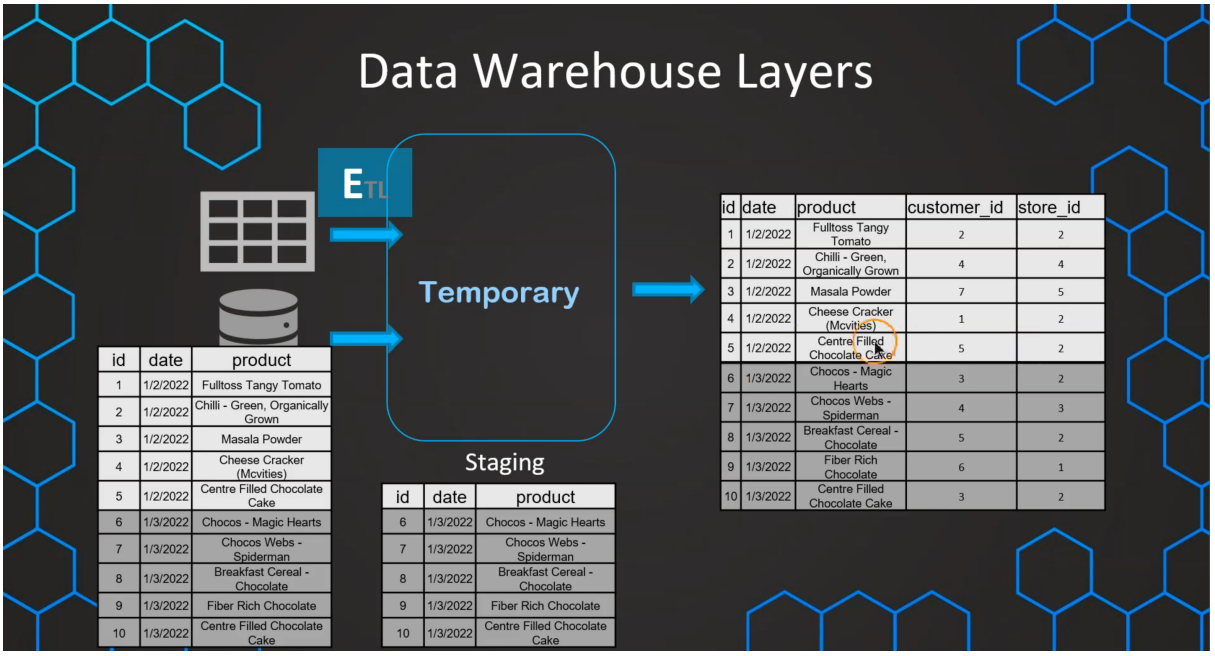




- Staging layer is made empty(truncated) before loading the new data from source
- Staging layer is a temporary layer because we truncate after each load
- We look at the max id in the data we have already loaded id=5, and we will write logic to load data from new data where id>5 and append the new data to existing tables that is if we use id column to identify delta records and id here is strictly ascending numerical column

- Usually date column is used to identify delta records

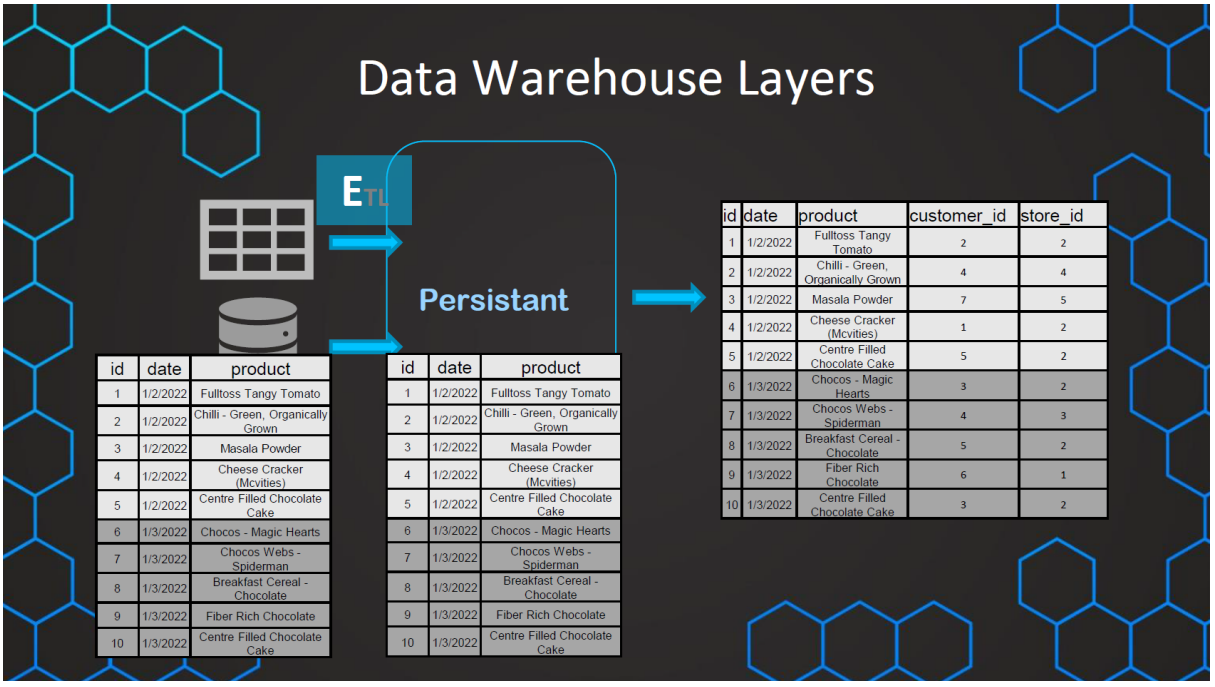


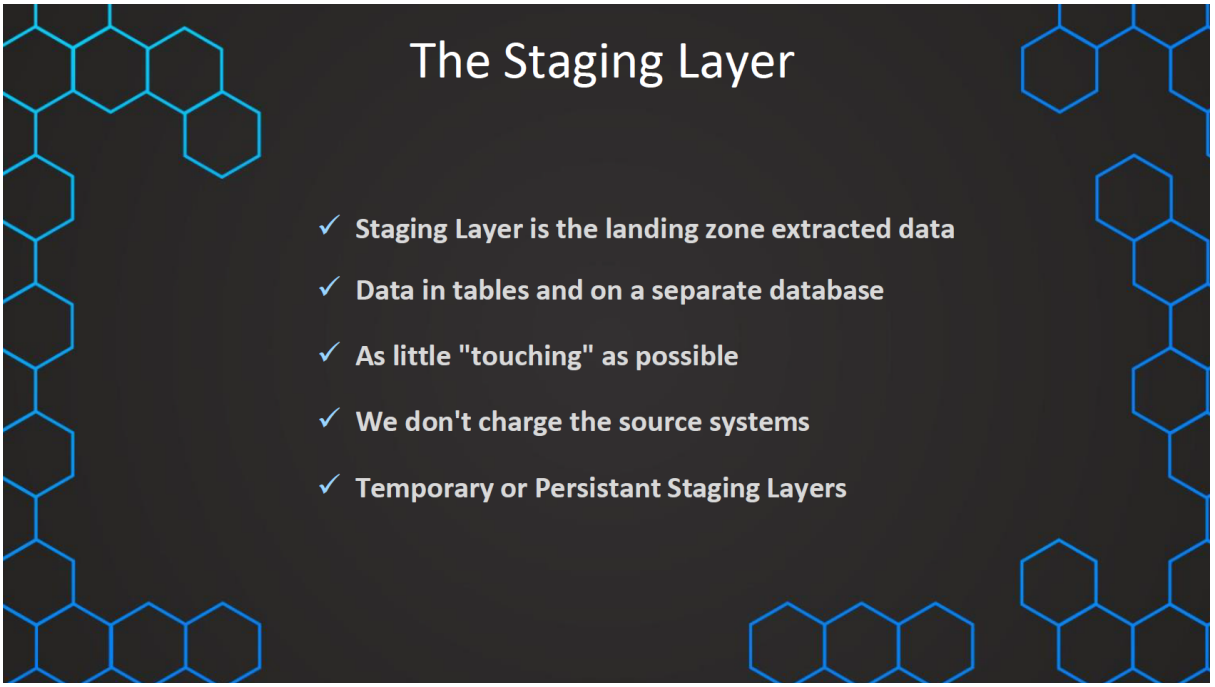


- One downside with this approach is, the transfomation can sometimes be problamatic and we might have to go back and start extracting and transforming from previous day/week for example and to overcome this we have persistant staging layer

Persistant Staging Layer

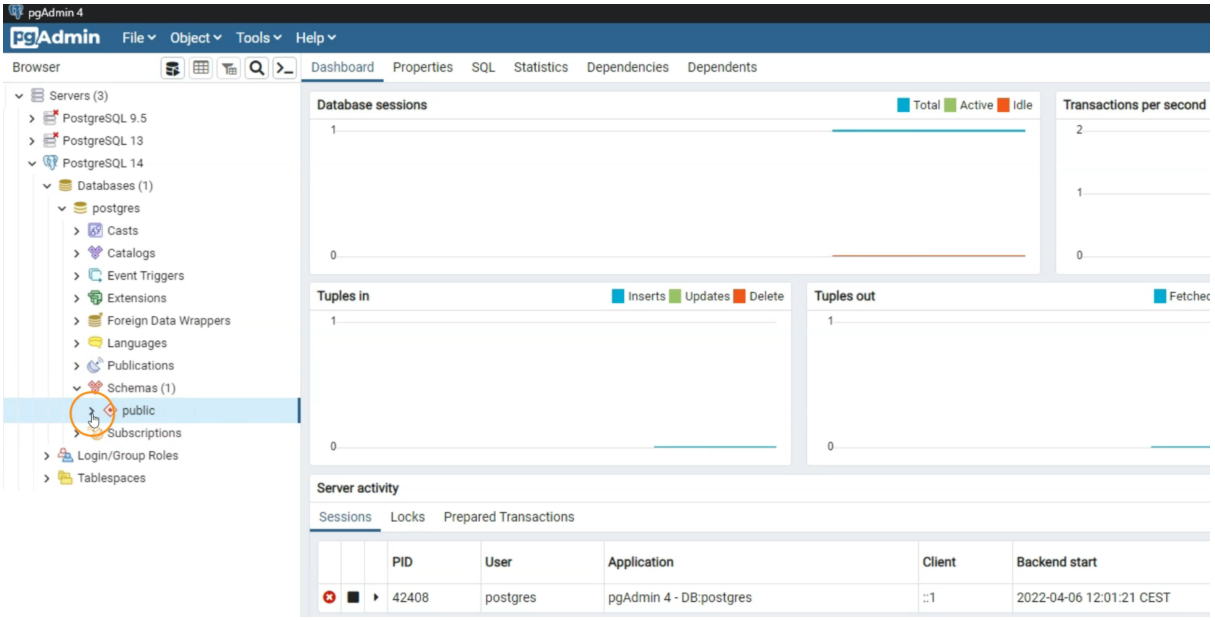
- There can be some cases where we have to roll back and start again from the previous load data
- For this scenario we can have something called a persistant staging layer where we do not truncate the staging layer, it can be used to roll back as we do not want to spend more time with the actual source systems for avoiding the reducing performance of operational systems
- But this is optional and rare

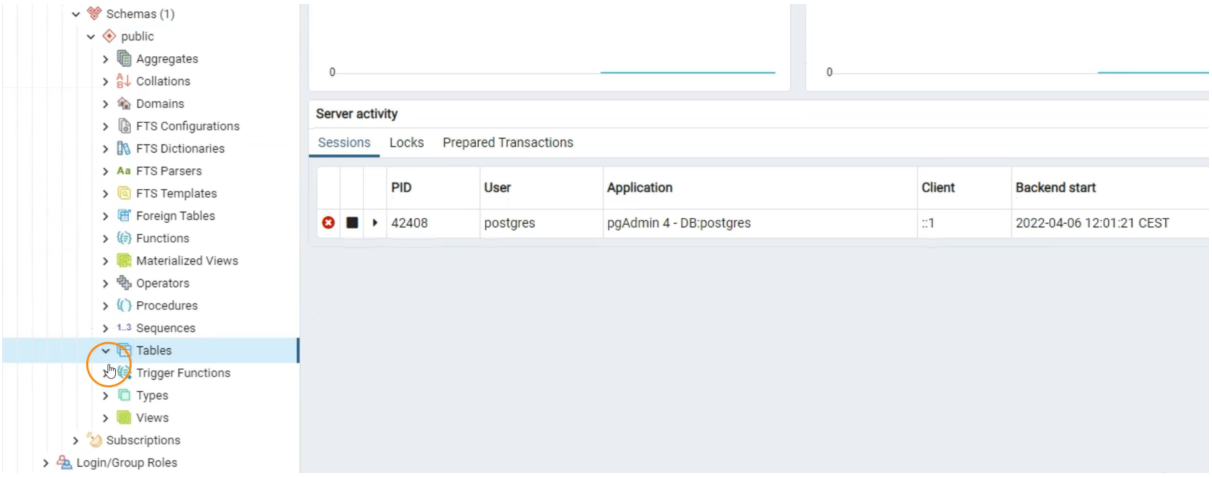




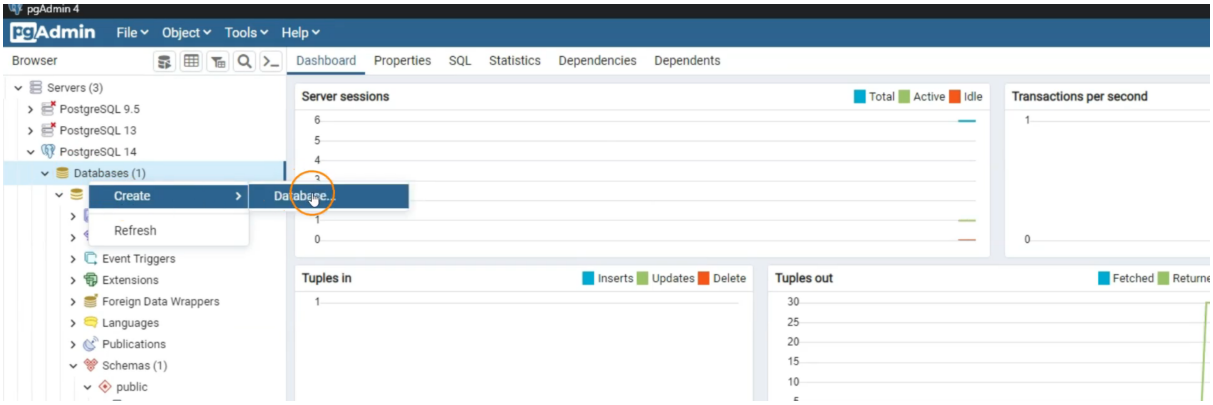
Demo of Staging layer

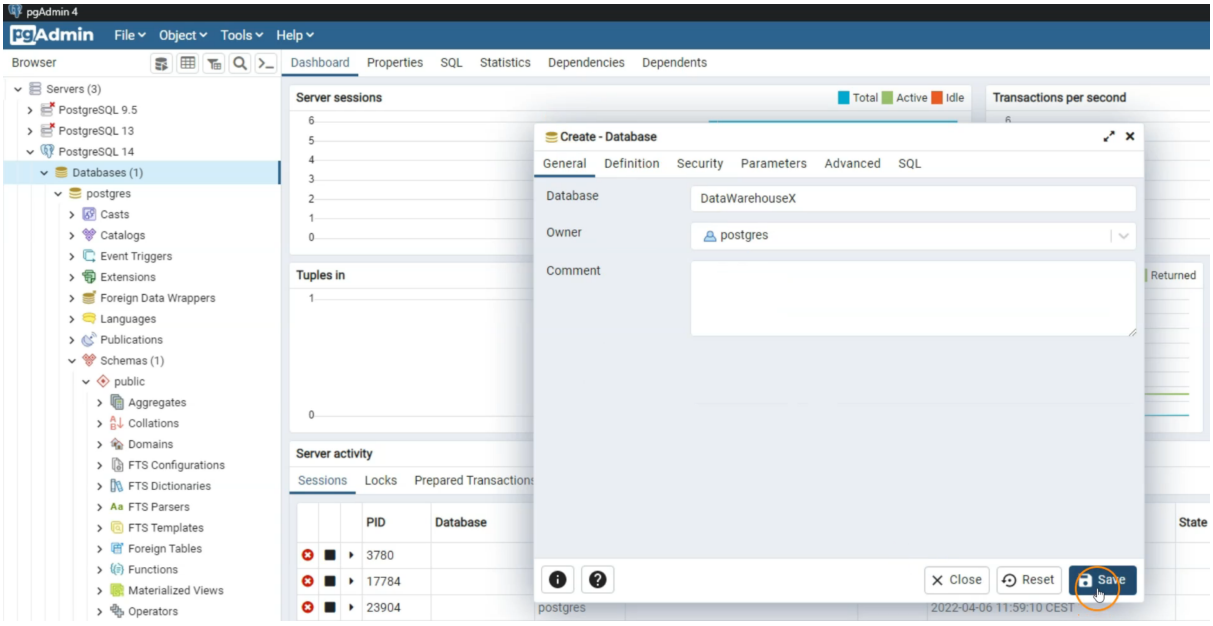
- Data warehouse is usually in one database.
- So we have all of these layers, usually in one database. We can have, depending on the company, if we want to define it like that, also have them spread across multiple databases.
- So one database is the staging layer, another database is the core layer or any other layers that we want to have.
- It's more common that we use so-called ***schemas*** for that. So a schema is basically a bucket within a database
- We have *Databases>postgres* (default database)*>schemas>public*(default schema)
- Inside a schema we can have multiple tables

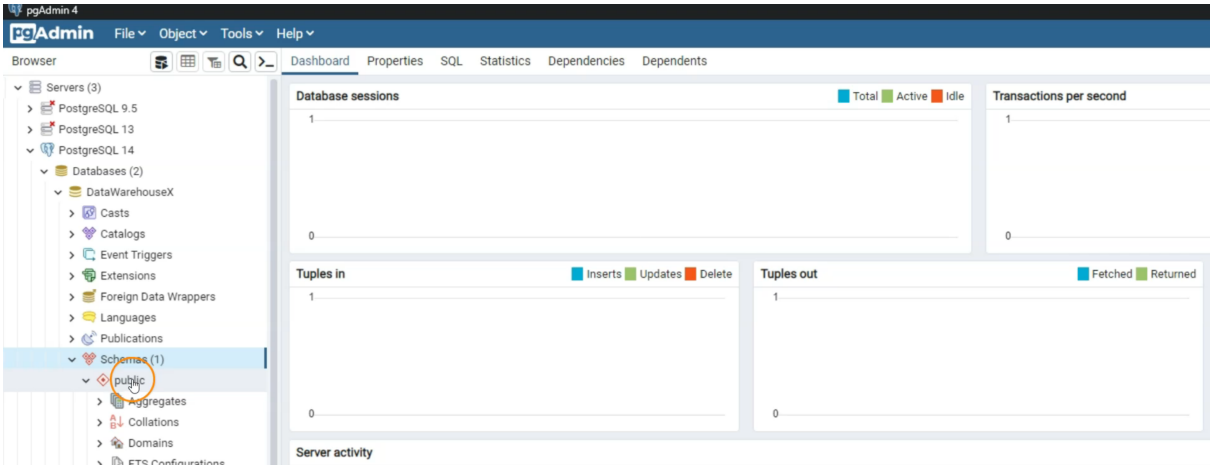




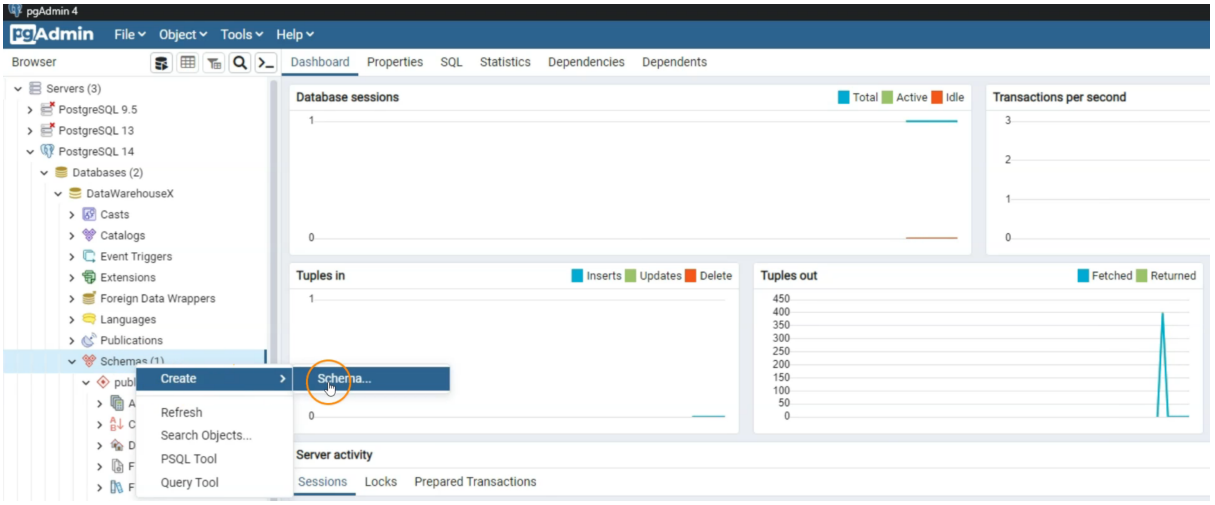
- Right click on Database > Click Create Database for our data warehouse

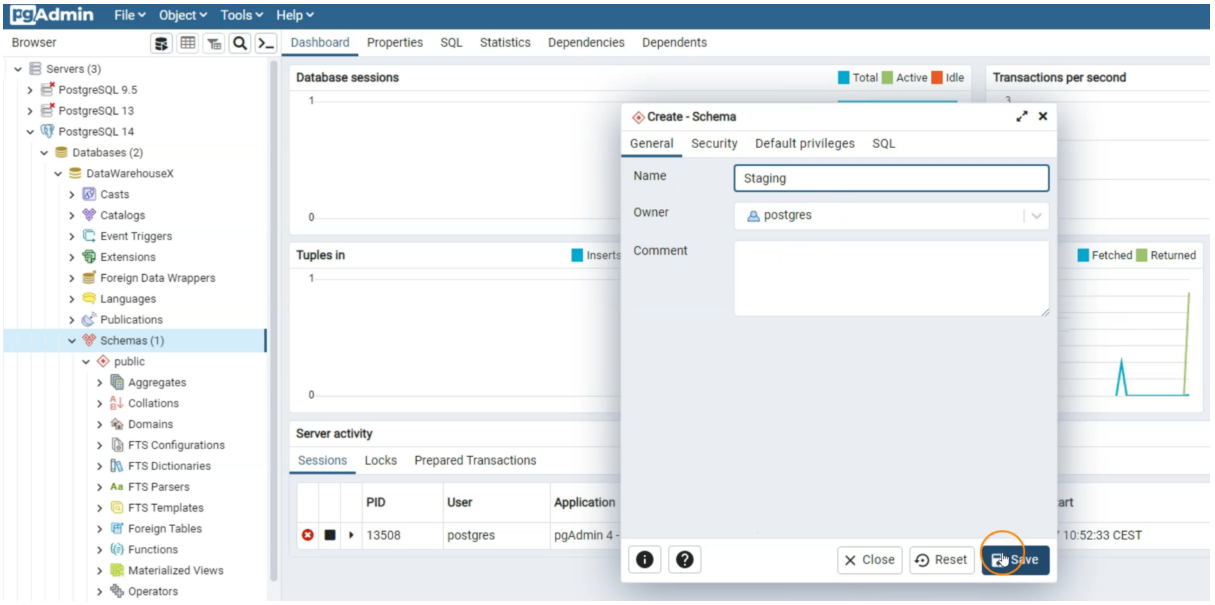




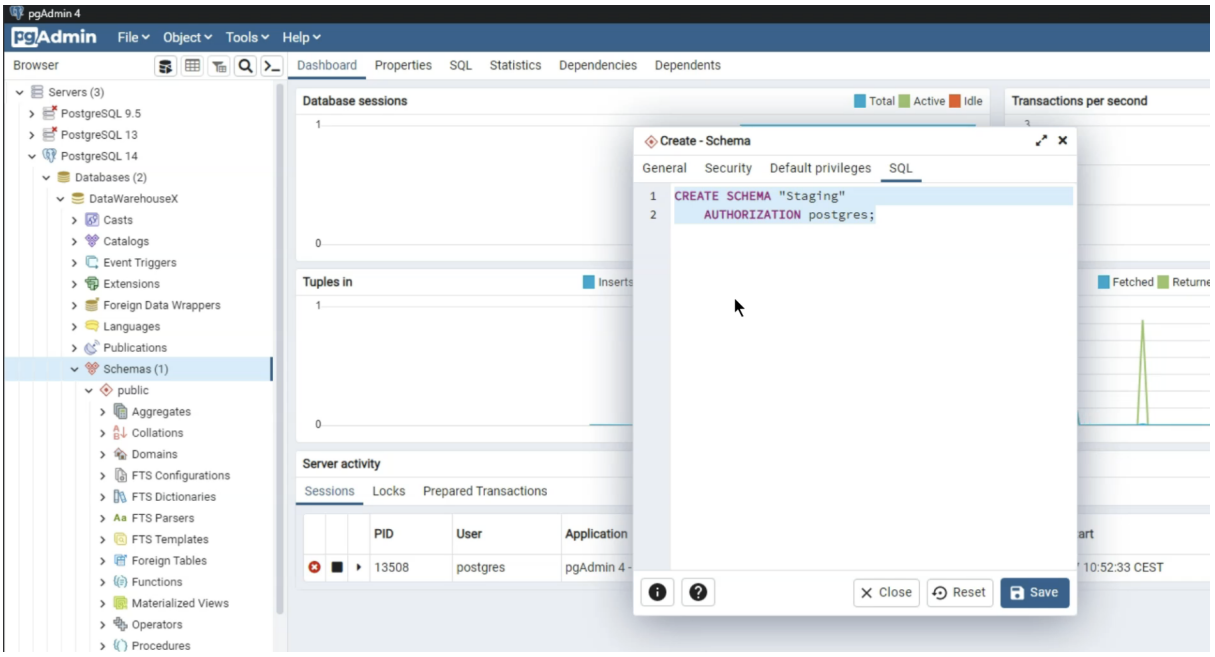


- Now in our new DatabaseX, we again have a default schema public that is available

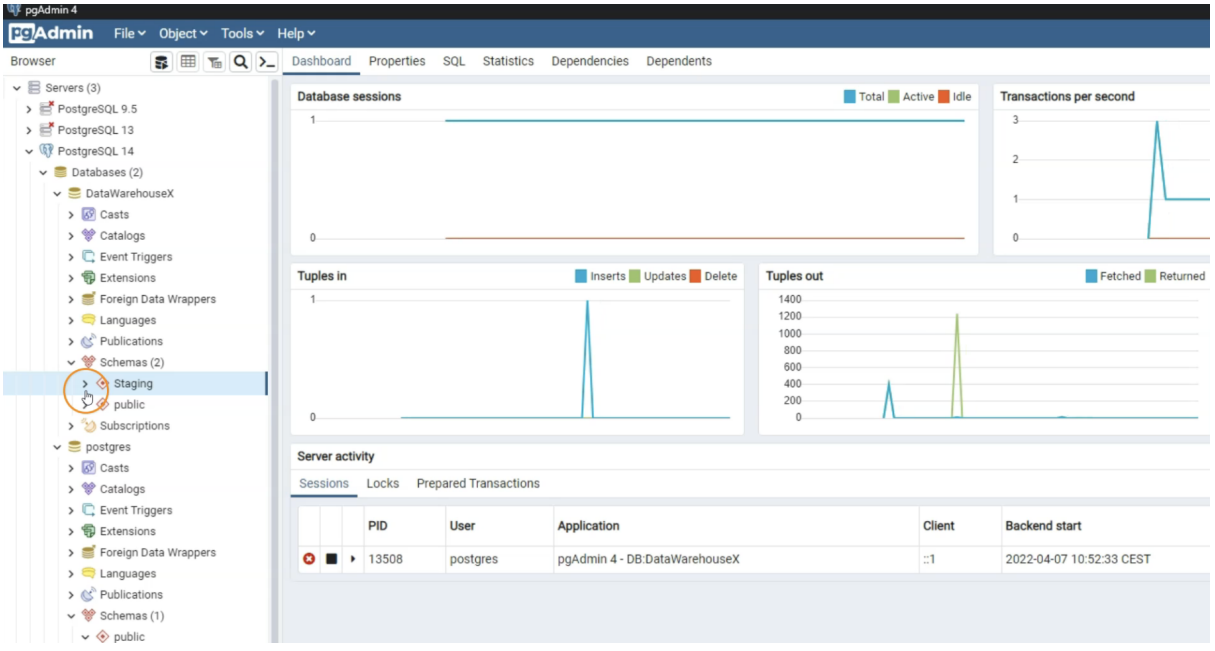




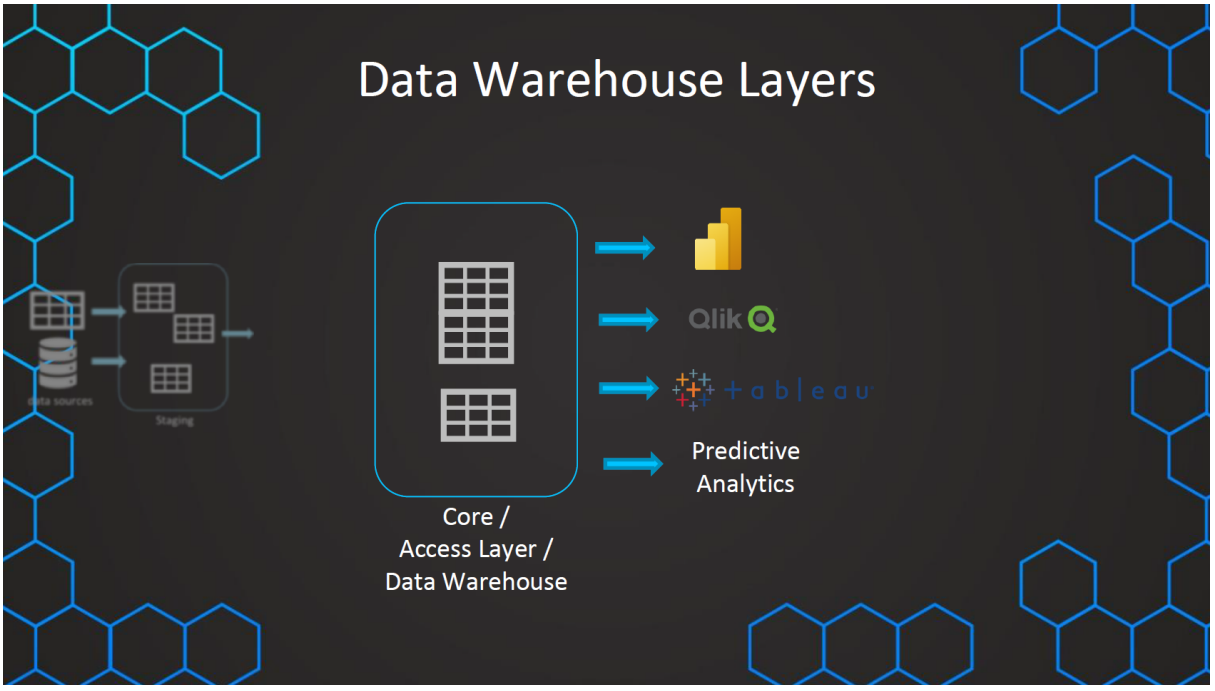
- We can also use SQL command to create the database/ schema / tables

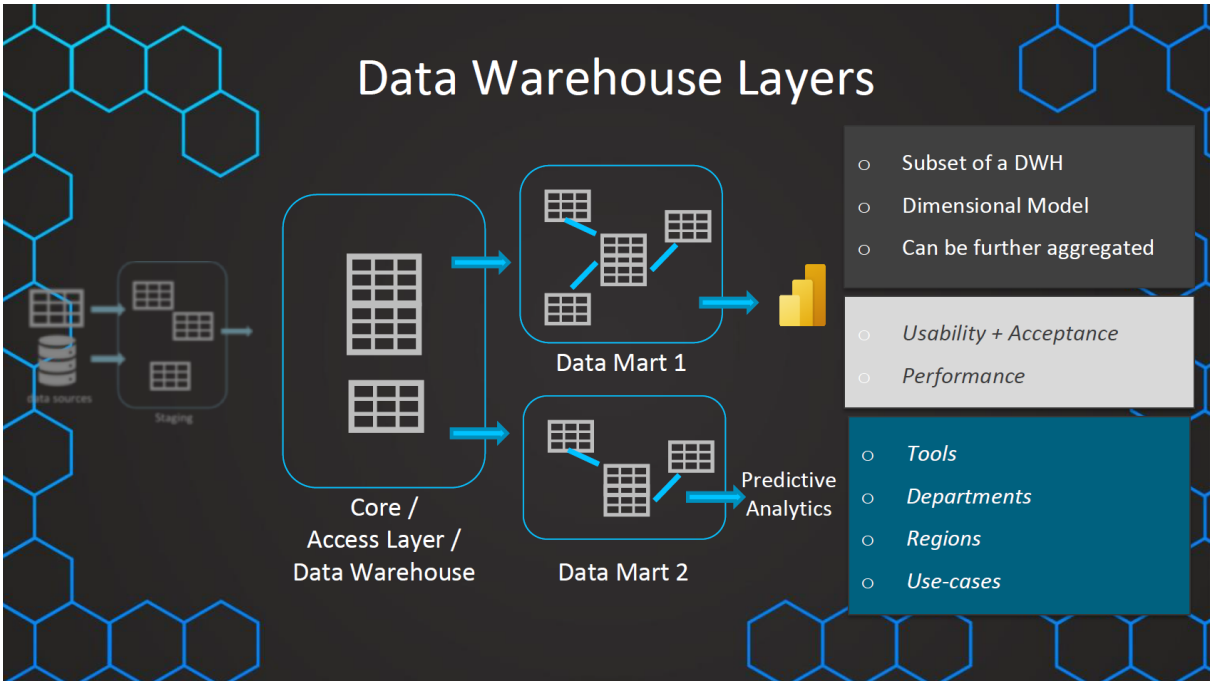


- Go to our Datawarehouse database which we created and go to schema and create a new schema called Staging inside this staging schema we will create staging tables



Data Marts





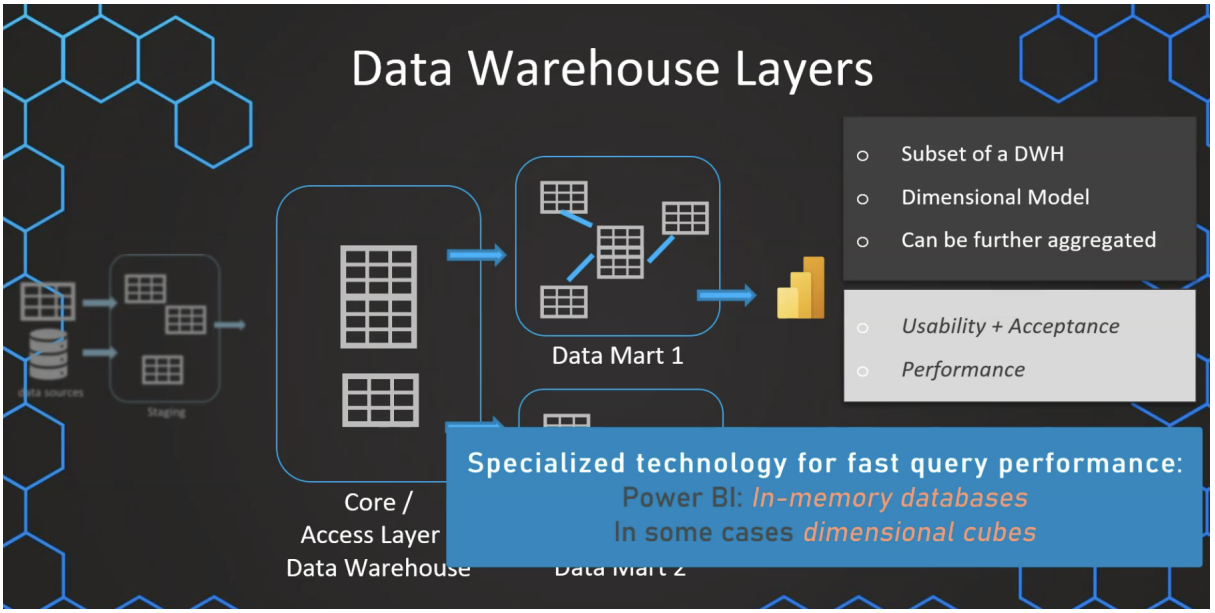
- We have staging layer and core layer in our data warehouse
- Usually the core layer of data warehouse can be used as a access layer
- Sometimes if the company is large and the data warehouse is built for a large purpose, we may use many different tools with this Data Warehouse. All of the user groups, all of the departments, all of the regions, are all using the same Data Warehouse then this sometimes create complications and

for this reason we might need to have additional layer on top of core layer called data marts

- Data marts are just a subset of our data warehouse/core layer and these data marts have fact tables in the middle and have dimension table around the fact table
- Even in core layer if there are no data marts present, this core layer also can be modelled in a way to have a fact table and dimension tables around the fact table
- We can aggregate the data in data marts

Advantages

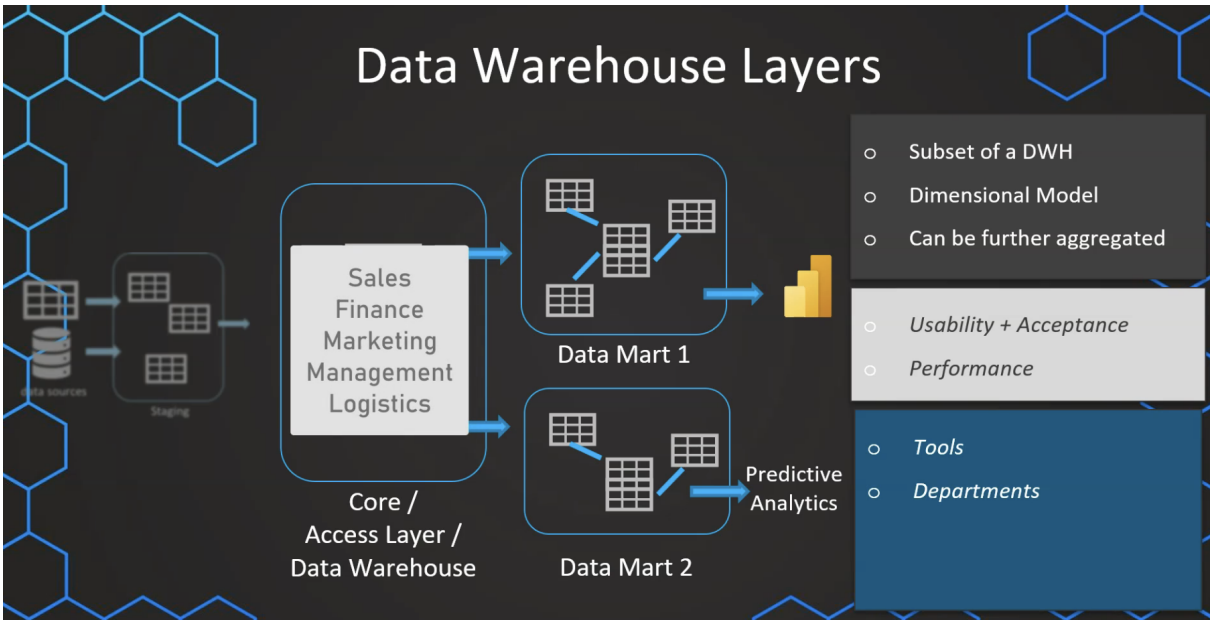
- Increases usablility and easier to focus on the relevant data. There can be too many tables in the core layer and we do not want to overwhelm the users
- Can be used to increase the performance because the data is modelled in a dimenional way like we can use in-memory database which have fast query performance or we can build cubes

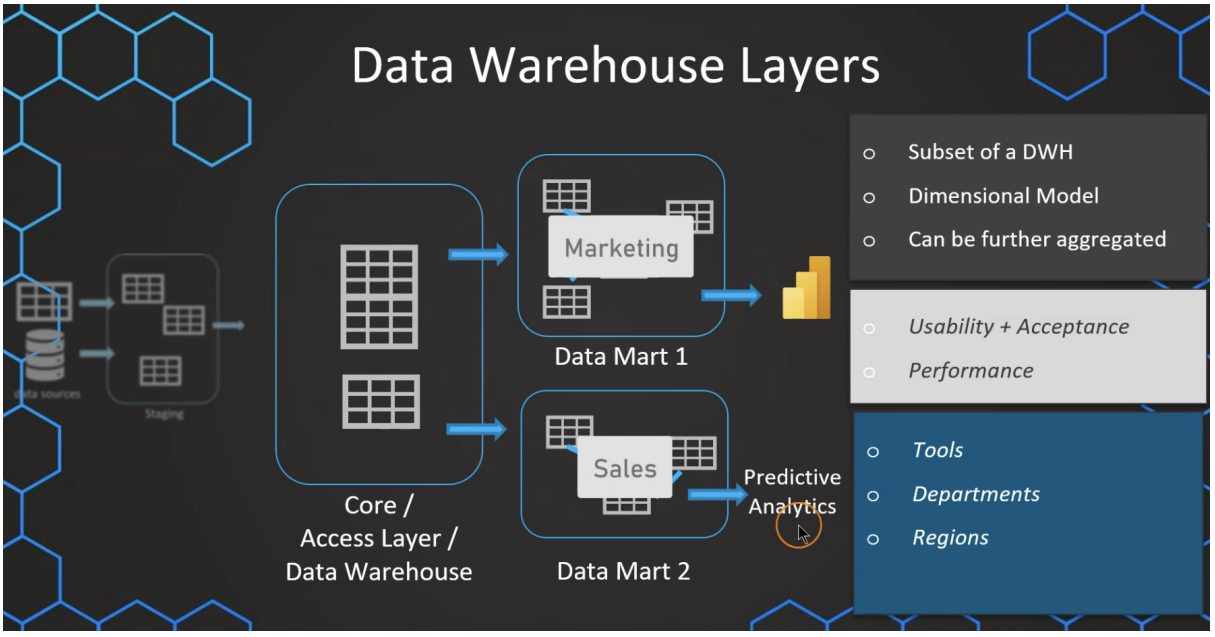


Use cases

- We can have different tool such as visualization with Power BI, using in-memory database as data marts can improve the performance significantly
- Each department data can be seperated out into it’s own data marts also can be done for different regions or for different tools, one data mart for

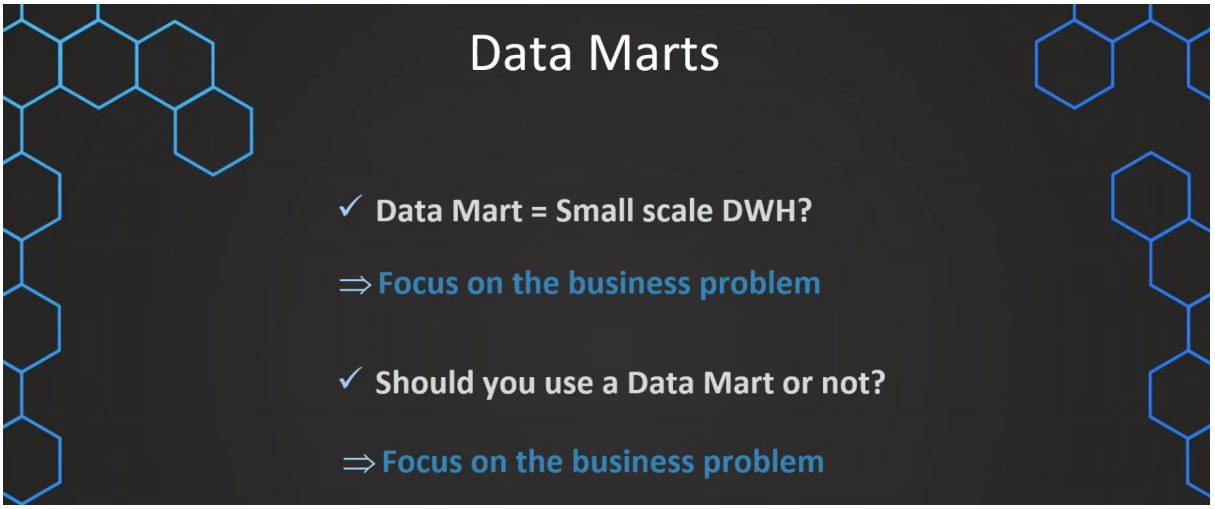
PowerBI and another data mart for Predictive Analysis





- *Data marts can be treated as small scale datawarehouse.*
- Should you use a data mart or no?

  This depends on how centralized your core is or our Data Warehouse is, and how many use cases we have. Focus on the business problem and if this is a good solution in our case, we should absolutely use a Data Mart.



Quiz

Question 1:

What is not a reason for a staging layer?

- (●) **Increasing the performance of the access layer**
- ( ) Spending short time on productive systems
- ( ) Getting data into tables in a relational database

Good job!
Exactly, this is where we apply the transformation logic and load the transformed data.

Question 2:

Where do we apply most of the transformations including data cleaning in the ETL process?

- ( ) From the data sources to the staging layer
- (●) **From the staging layer to the core layer**
- ( ) From the core layer to the data marts

Good job!
Exactly, this is usually not a good reason to built a dedicated data warehouse. We usually want to analyze data over time and different periods usually don't constitute a different use-case.

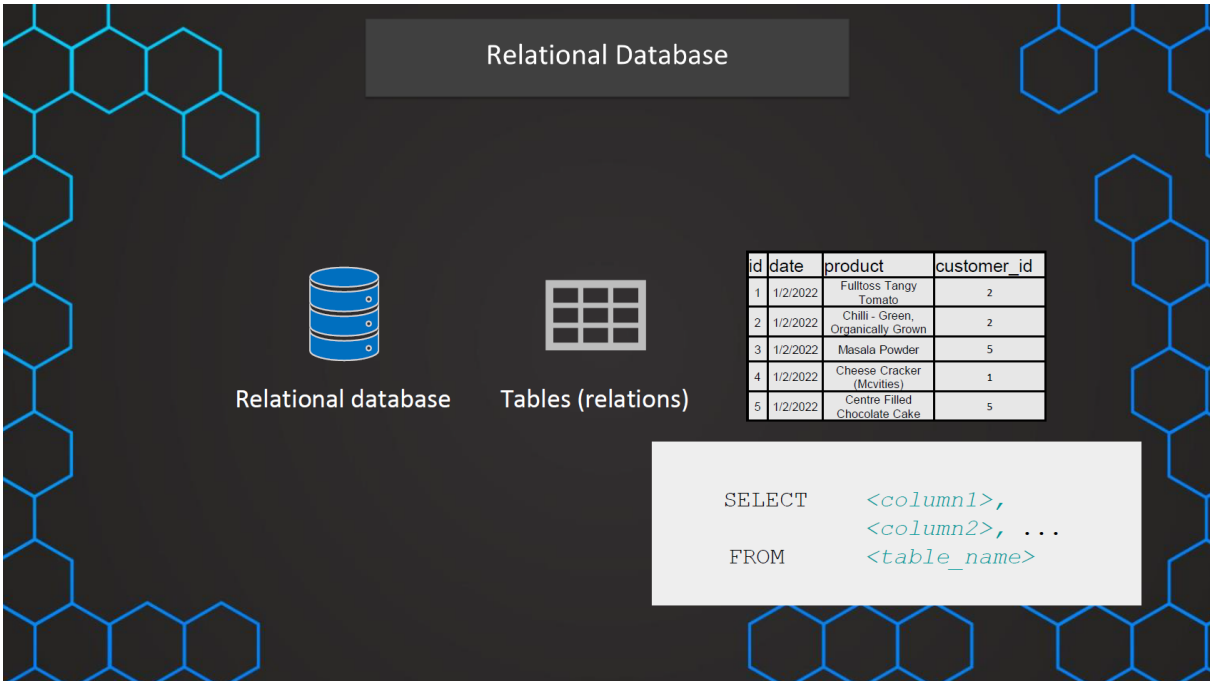
Question 3:

What is not a good reason to create a dedicated data mart?

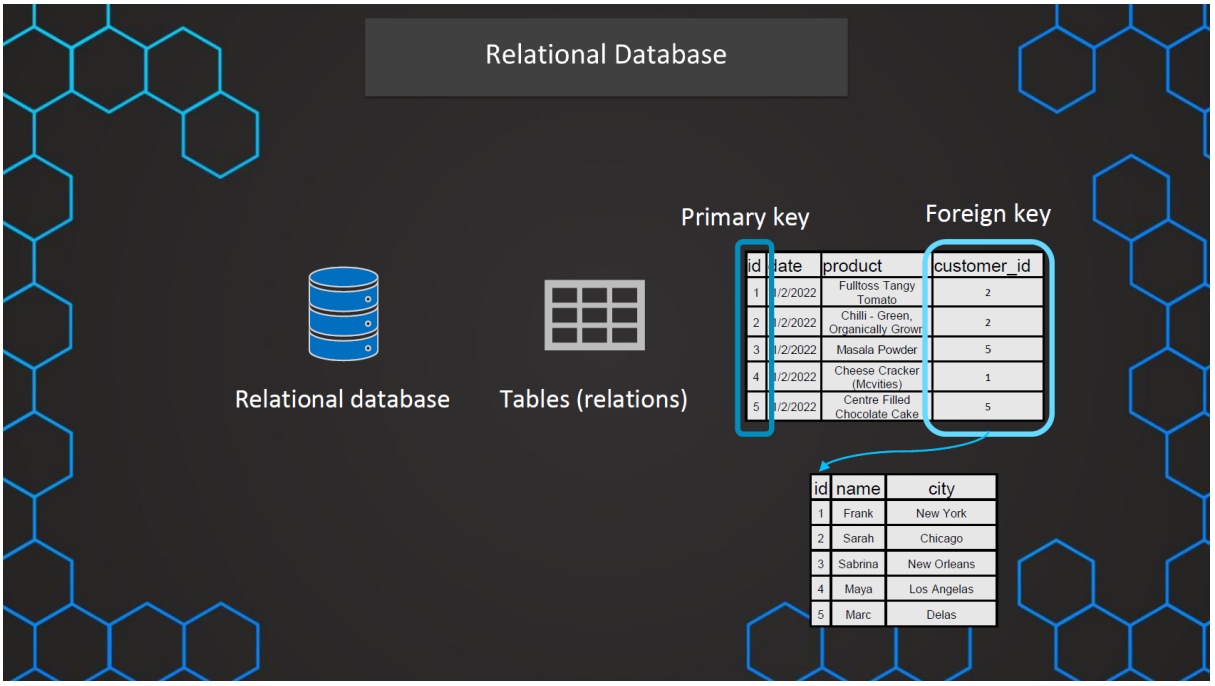
- ○ Different user groups
- ○ Different tools
- ◉ **Different periods**
- ○ Different departments

Relational Database

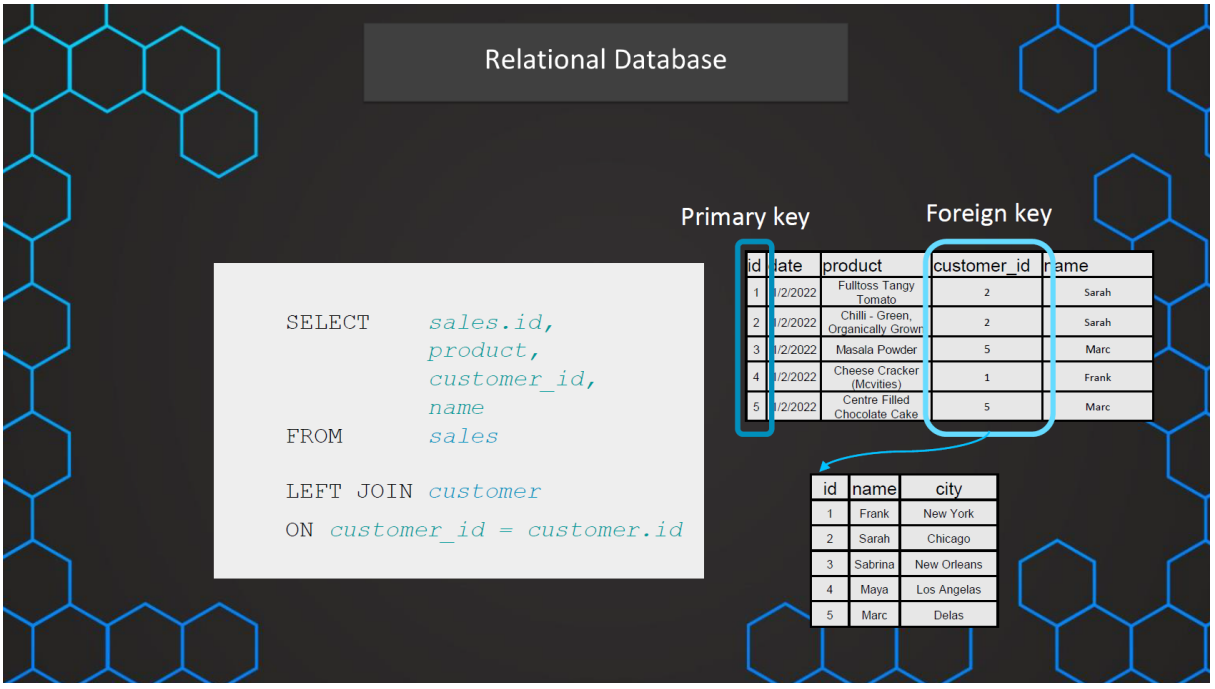
- In Relational database we store the data in tables and the tables have relations between them and hence called relational database
- Data in Relational database is in form of columns and rows
- We use SQL to query the data

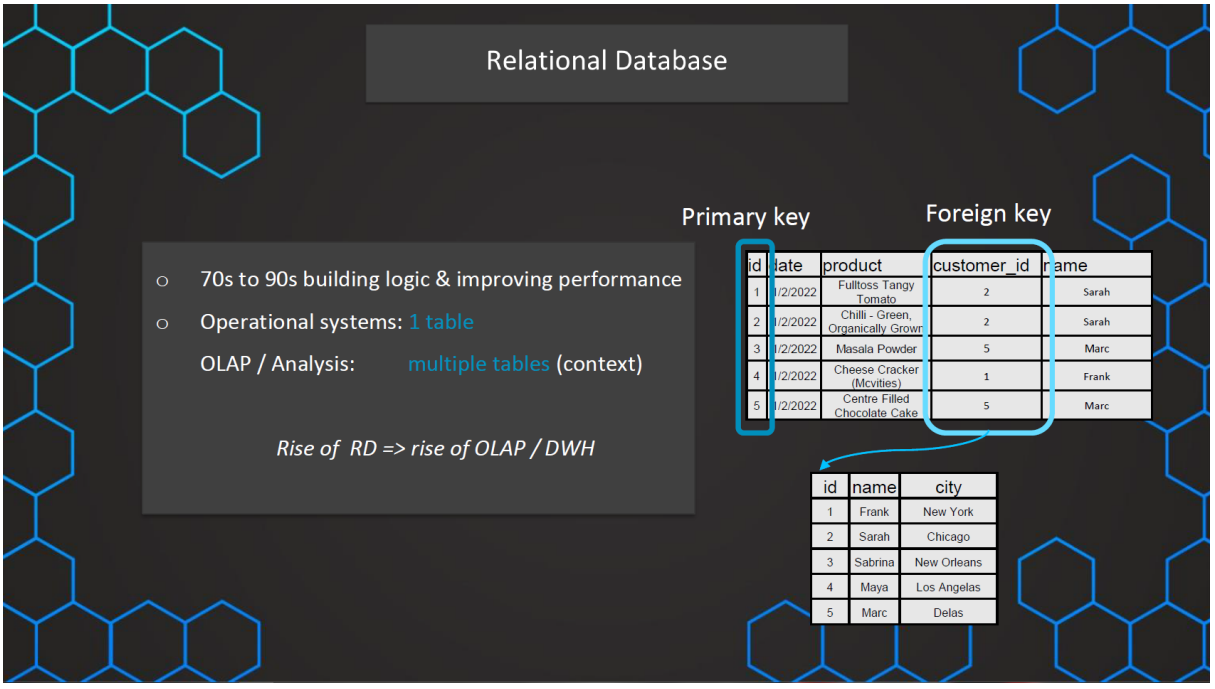


- We can use primary keys and foreign keys to create relation between tables
- Primary key is used to uniquely identify a record (row) in a table, primary key columns should contain only non-null unique values
- In each table we can have one or many foreign keys, purpose of foreign key is to reference another table unique record i.e. foreign key is primary key of another table



- We can do joins between tables to combine 2 or more tables

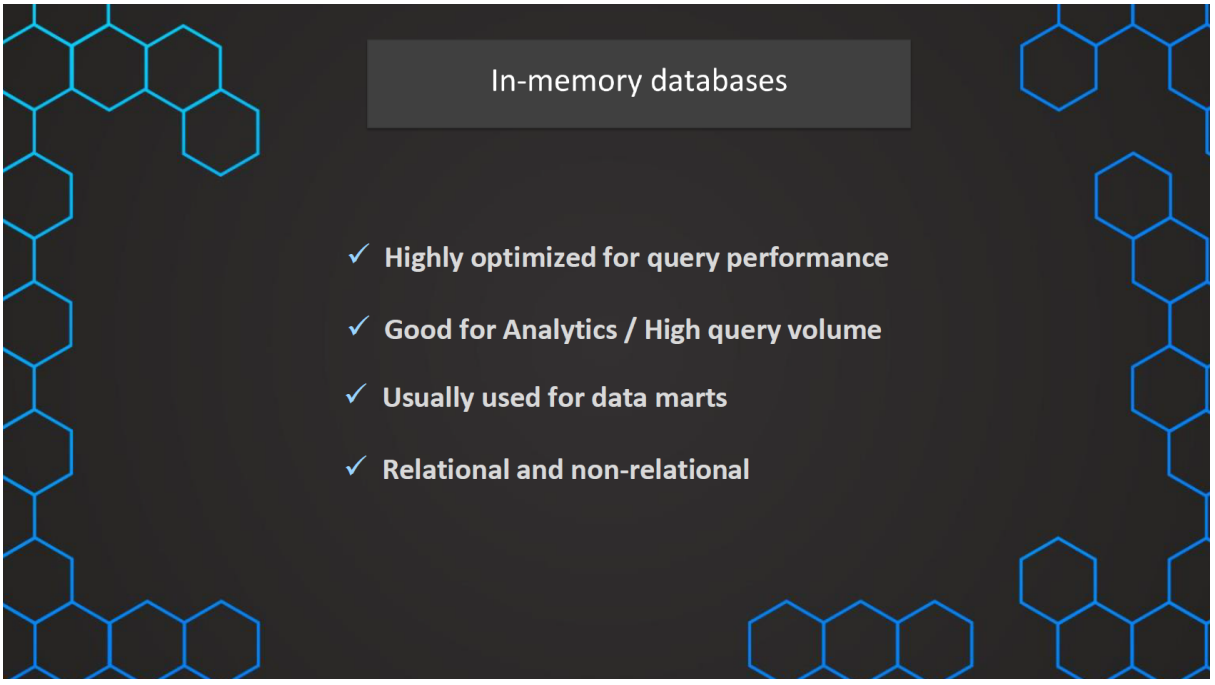




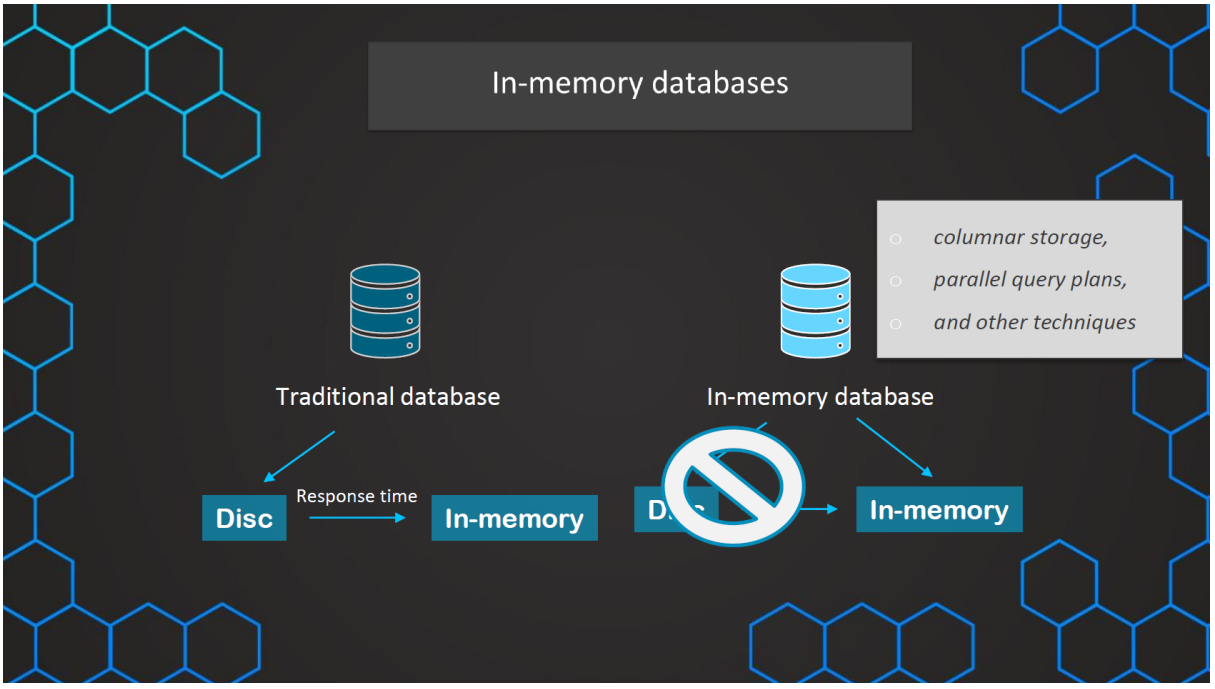
- Some examples of relational database
  - Oracle
  - Microsoft SQL server
  - PostgreSQL
  - MySQL
  - Amazon Relational Database service (RDS)

- Azure SQL databases

In-memory databases



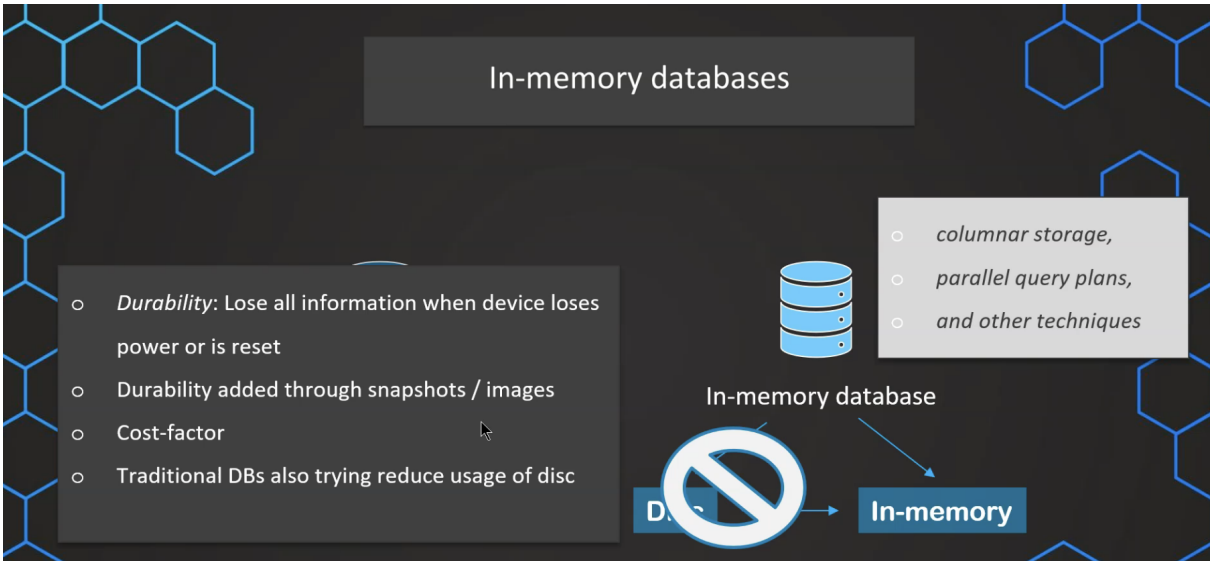
- Traditional database store the data in disks like HDD/SSD, when the data is queried and data is processed and then the data is loaded into memory, this response time is lot, hence this is not optimal if we need high query performance
- Directly we will store the data in-memory, we eliminate the reponse time coming from disk which takes up more time



- Columnar storage - Here the data is scanned through columns and not by rows
- Parallel query plans - Larger queries to take up a lot of time can be broken down into multiple parts and then be processed parallel by different threads.
- We eliminate the response time of loading the data from the disk into memory and hence achieve high query performance

Disadvantages

- Durability: Lose all data when devices power or restart
- Durability added through snapshots/images
- cost-factor
- Traditional database are also trying to reduce disk usage



Examples

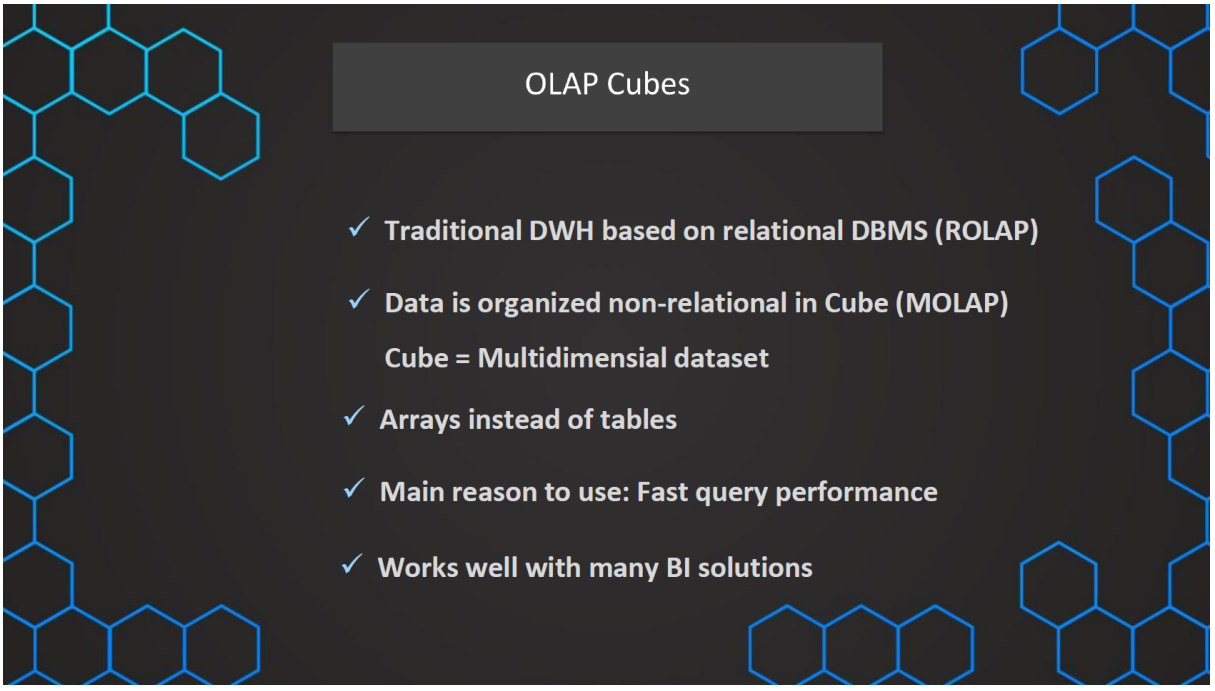
- SAP HANA
- MS SQL Server In-memory tables
- Oracle in-memory
- Amazon MemoryDB

  We would usually use these wisely where data is less like data marts and not for entire data warehouse as the cost of this technology is very high

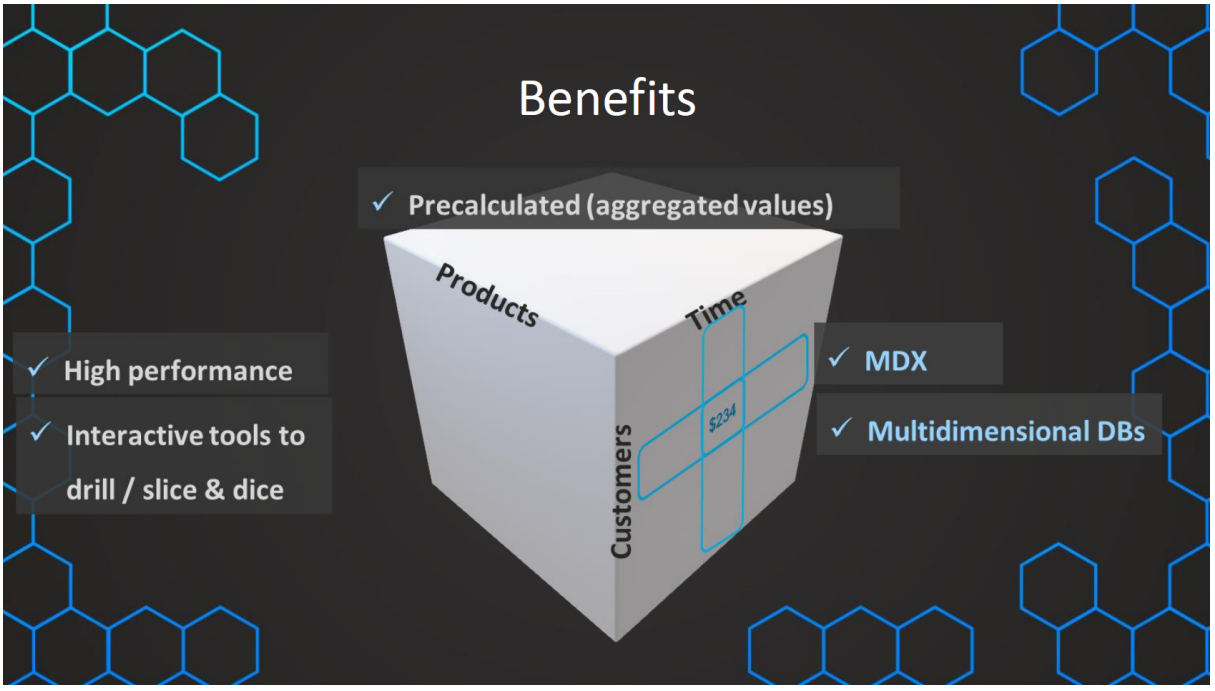
OLAP Cubes

- This is an alternative method to increase the performance of data marts
- In traditional data warehouse we know the data is stored in relational database and data is organized into tables which can have relations between them
- But now in a cube, the data is not organized in tables with relations but in a non-relational way and into dimensions. So we have multiple dimensions
- We also refer to this as MOLAP, because the more precise expression for a cube is a multidimensional dataset. And in this multidimensional data set, the data as we mentioned, is not organized into tables with columns and rows, but into so-called arrays.
- Main purpose is fast query performance

- If we have those cubes created with some technology or some software, we can use those cubes in different BI solutions.



- For example, we want to analyze the sales data into multiple dimensions and be aware that we can have more than just three dimensions. But more than three dimensions is difficult to draw, and that's why we use three dimensions.
- So in our case, we have products, time and customers, and we want to analyze the sales. We want to measure sales on multiple dimensions like Products, Time, Customers, etc



- We can use arrays to slice and dice the data, we can use intersection from a certain customer on certain product and for certain time and get the result from pre-calculated data cube
- We can get a specific data point calculated, the benefit of the cube is we have pre-calculated values and can get fast query performance which is beneficial when we want to see the data, visualize it and use it in our tools as the data is already calculated and already available
- To access the data we have to different language than SQL called MDX language (Multidimensional expression developed by Microsoft), MDX is most common language that is used to query the data from cubes
- We get high performance due to this pre-calculated values, we get most benefit when we use it in interactive tools where we slice and dice the data
- Since this is a different technology and data is stored in a different way this is not stored in relational database but we have data in multi-dimensional database so the hardware is also in a differen way

Recommendation

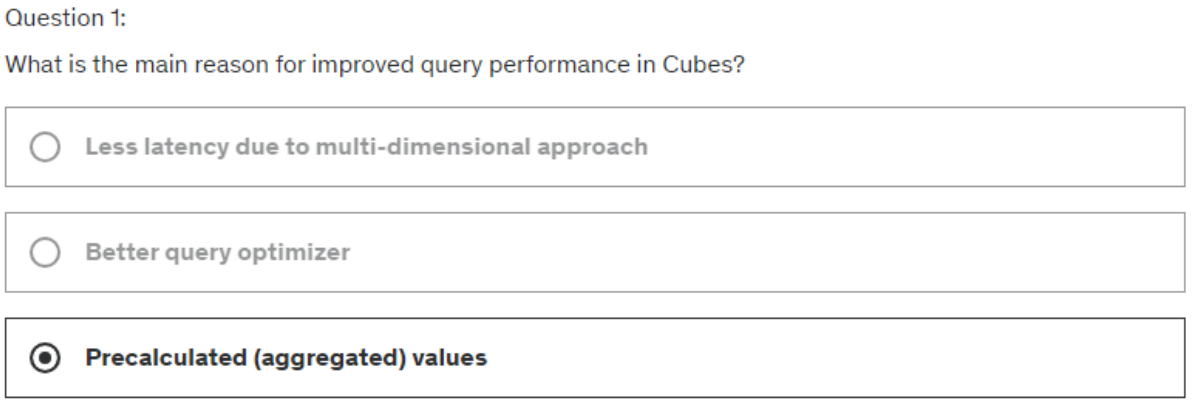
- These multi-dimensional cubes should be built for a specific use case, and that's why we use them in data marts where we have specific tables only those tables that are relevant for a specific use case are loaded into our cube and organized in our cube

- Because the more data we get, the more tables, the more dimensions, the more complex it gets, so the less user-friendly and the performance also gets lower.
- To get most benefit from the cubes we have to use it for a very specific use case of a data mart with only relevant data
- Most advantage is if it is used for interactive queries with hierarchies where the data is sliced and diced, also very beneficial with visualization tools
- This cube technology is optional, we can use a relational database with star schema if we have enough query performance

Alternatives

- With advancement of hardware technology, in-memory database performance is much better and cubes are less relevant
- We also have alternatives methods of storing the data like
  - Tabular models (SASS) by Microsoft
  - ROLAP
  - columnar storage and parallel processing
- Now the performance of relational database is also getting better we can stick with realtional database which is less complicated than muti-dimensional cubes

Quiz



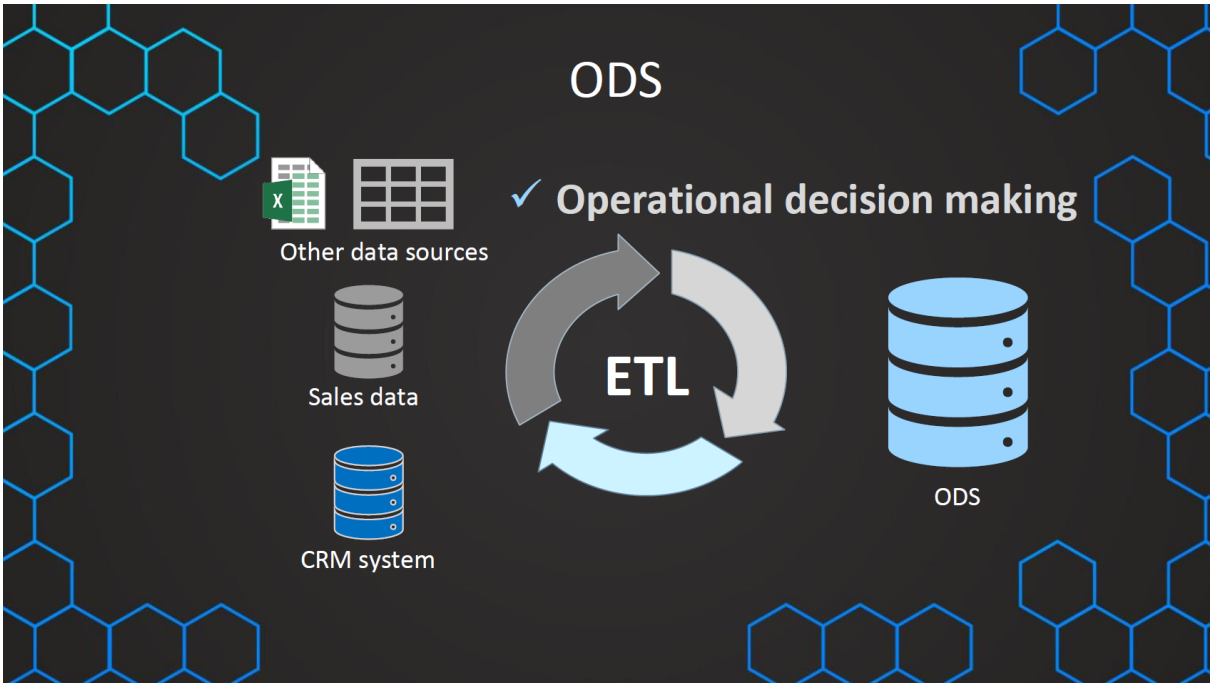
Question 2:

What is the key idea for improving performance with in-memory databases?

- ○ Defining multiple dimension and precalculating the relevant values.
- ◉ **Eliminating response time from disc by processing all data directly in memory.**
- ○ Optimizing queries with better query optimizers

Operational Data Storage (ODS)

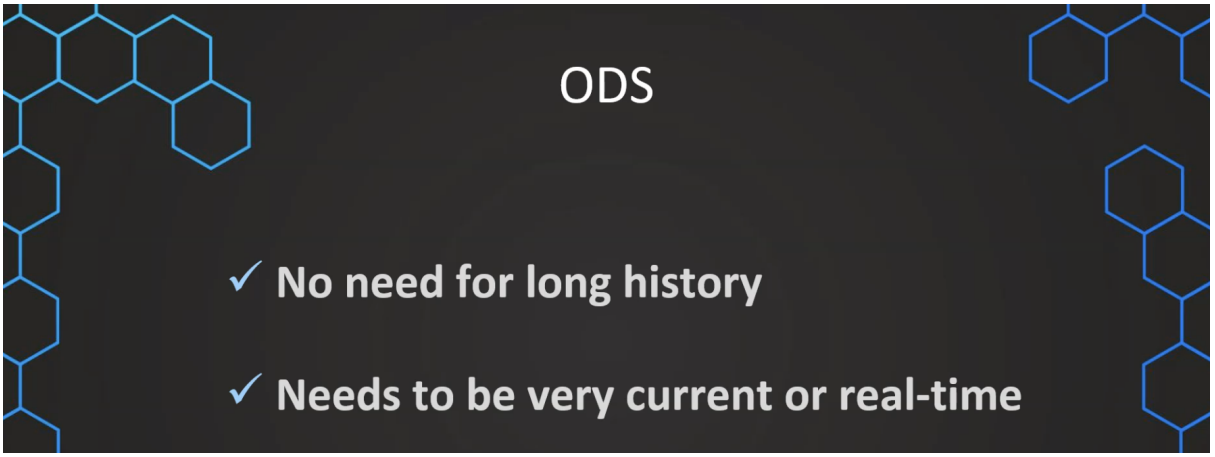
- ODS is similar to data warehouse
- We have different operational systems as our input sources and this can be sometimes where all of the important data is stored. But we want to integrate now these different operational systems and the data of these systems into one single database and we use an ETL process.
- The database in which we have integrated all of this data is our operational data storage.



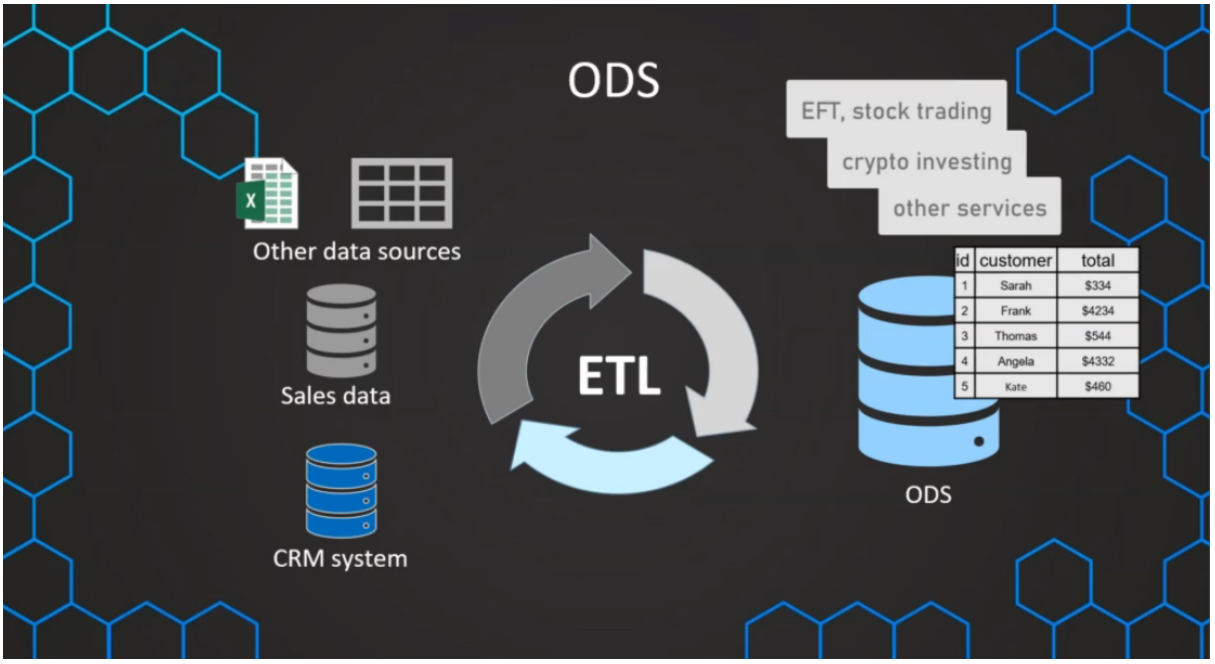
- The key difference is that an ODS is used for operational decision making.
- This makes the process a little bit different and also the requirements of an ODS different because it's not used for analytical or strategic decision

making but for very quick operational decision making.

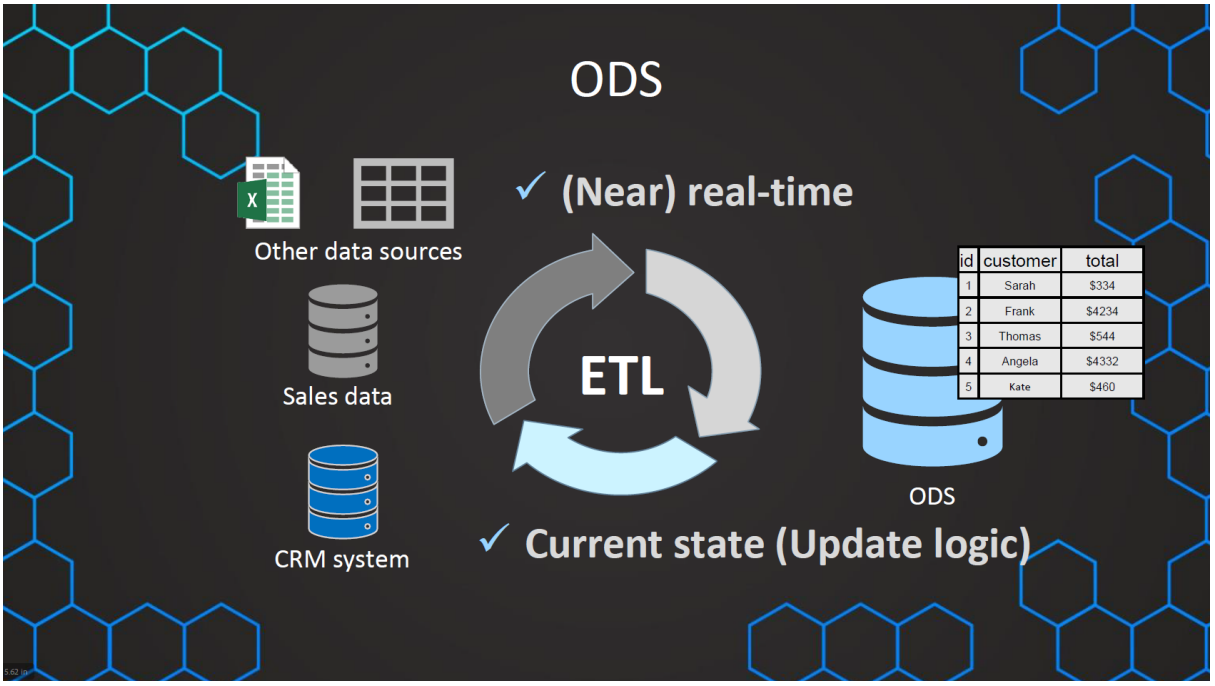
- In ODS we only want to make operational decisions, we usually don’t need very long history, we only need current state of data (needs to be very current or real-time data) as we don’t want to make decisions based on wrong data



- In an ODS, we don't need the history, but we need to have the data integrated as the current state of our operational systems in almost real time and this is why we can have a data warehouse and an ODS in our company.
- Example
  - So an ODS could be, needed in the following case.
  - Assume you are working in a financial service company, and in this financial service company, your customers, they can do different things.
  - They can invest in ETFs, they can invest in stocks, in cryptocurrencies, they can have an account balance and we need to find out from those different systems , we have one system for crypto, one for stock trading, and we need to combine the overall balance the overall amount that they have invested from those different systems and this needs to be reflected immediately so we can make our operational decisions

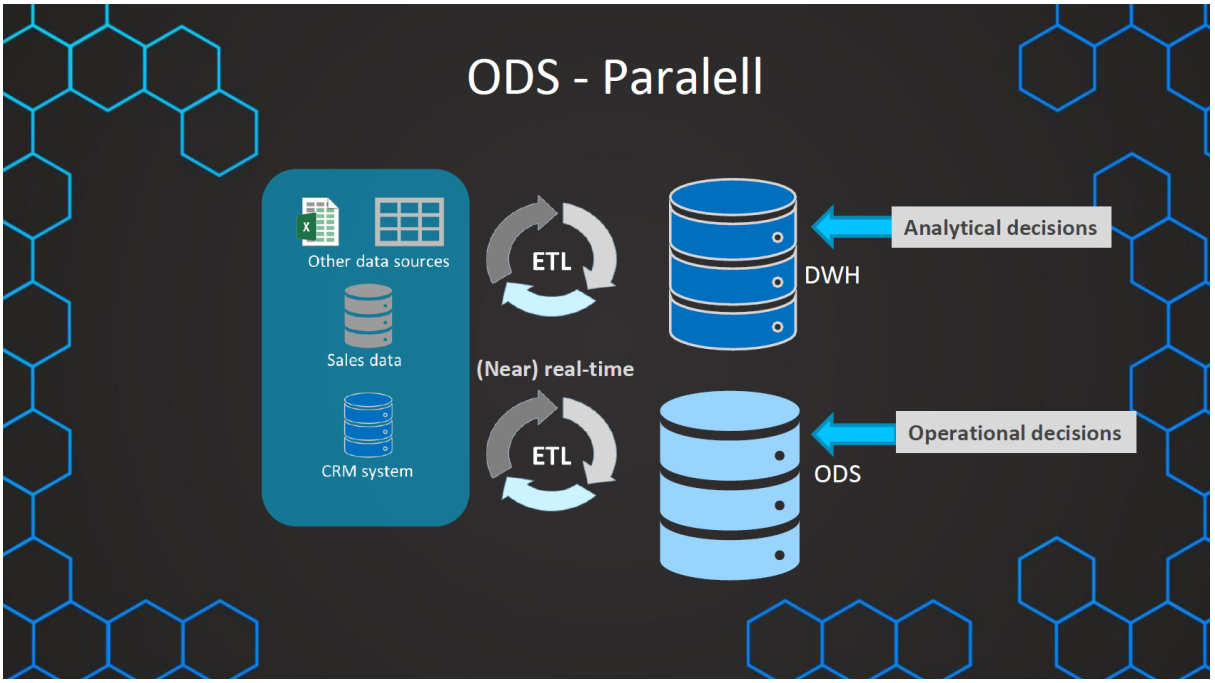


- So this is not anything that we need to analyze and make strategic decisions, but we need to make decisions now for one customer that are operational.
- So, for example, we need to decide can we give this customer a credit or not? and therefore we need to see the data as it is in these operational systems in almost real time so that we can make accurate decisions and also we don't need to have a long history but only how does the data look currently in those source systems.
- And therefore we have usually in this ETL or this real time data feed and update logic in place.
- So we don't append the data usually and keep the history but we just update or replace the data.



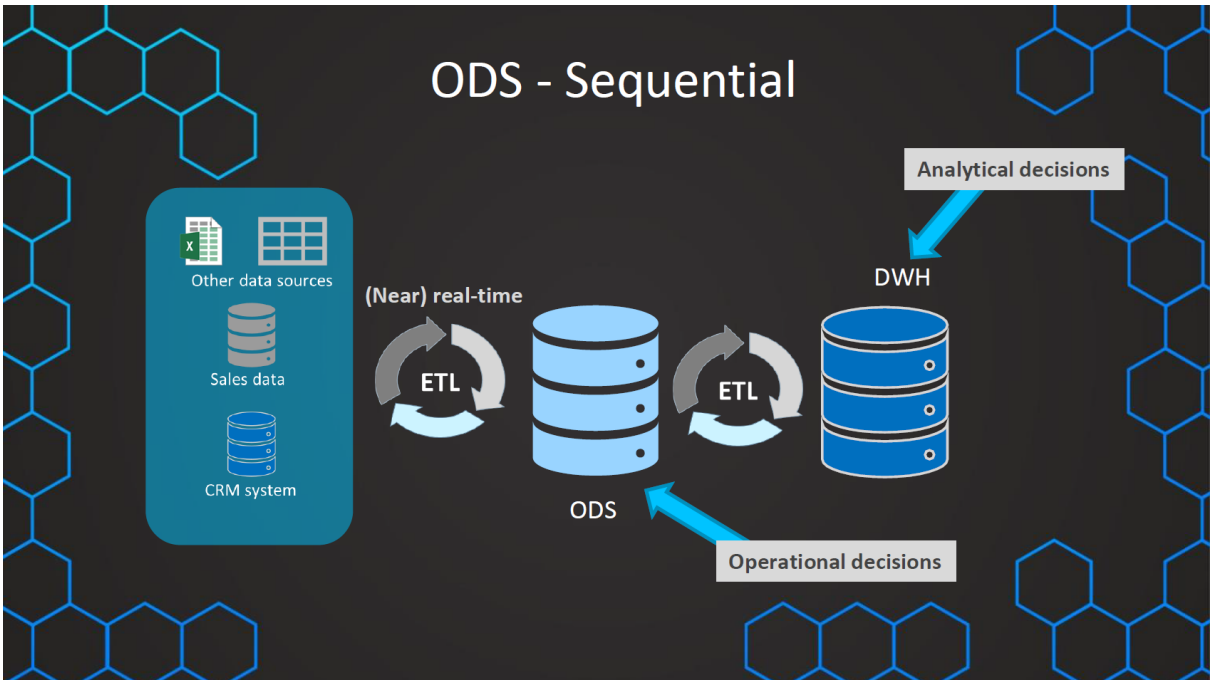
Can we have both ODS and Data Warehouse?

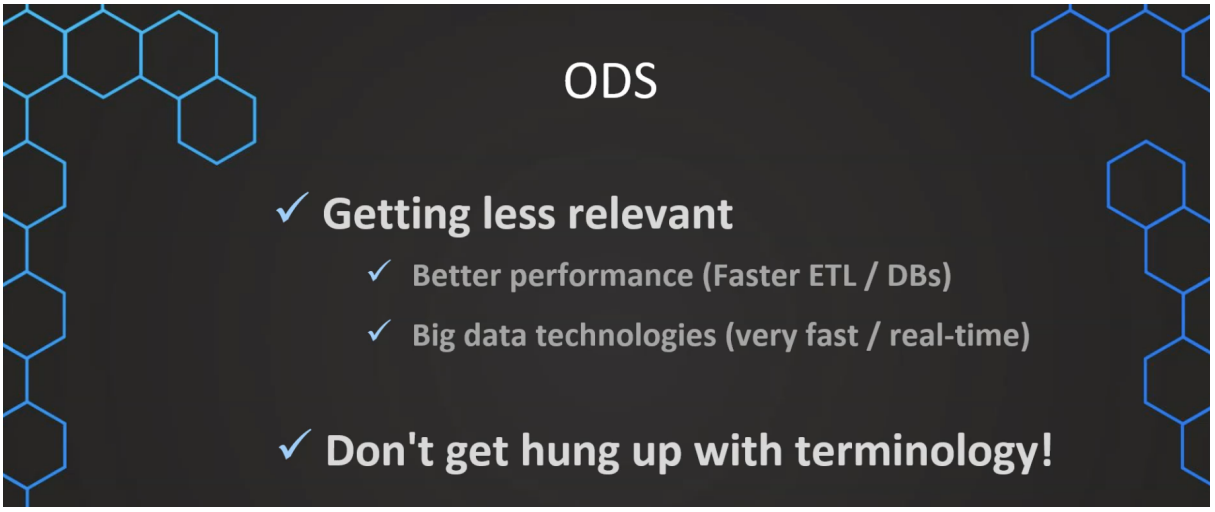
- We can have both ODS and data warehouse system which can have separate requirements and separate ETL process for operational decisions making and analytical decisions making



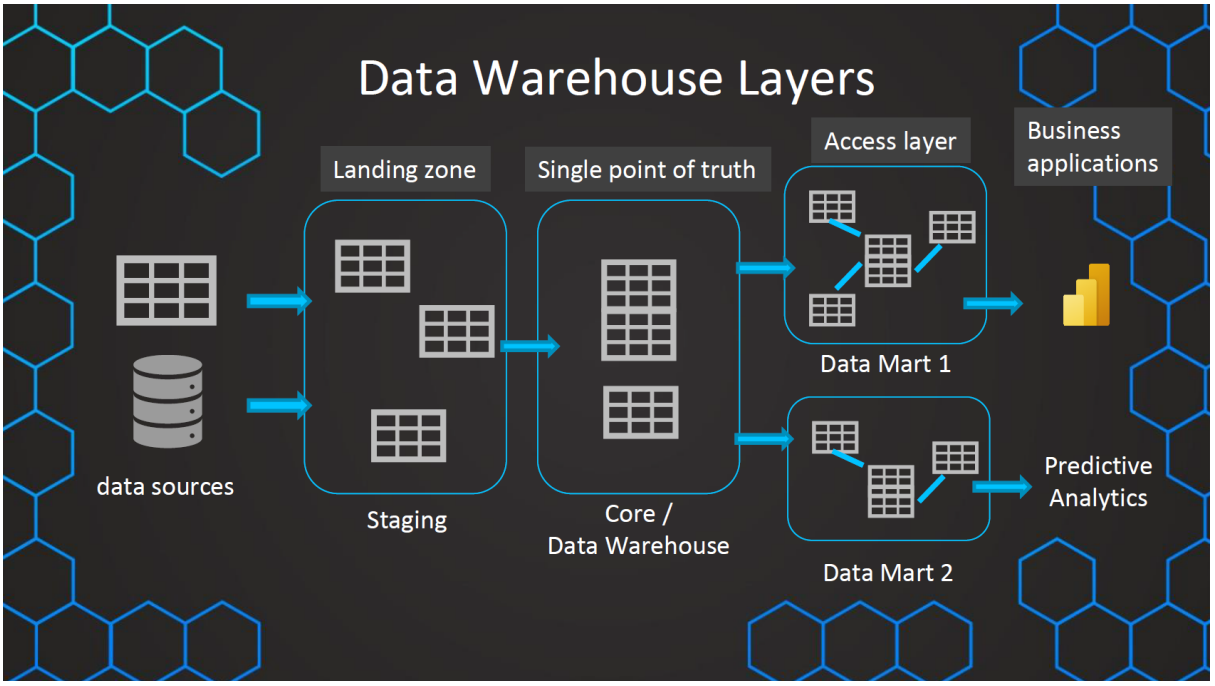
- We can also have sequential integration where we can have ODS layer and then build ETL process for our data warehouse on top of ODS layer , we can use ODS layer as our staging layer for our data warehouse

- We can save lot of work and effort when we use ODS as our staging layer for our data warehouse
- ODS is getting less relevant due to better performance of hardware, we also have big data technologies for updating large data and real time data

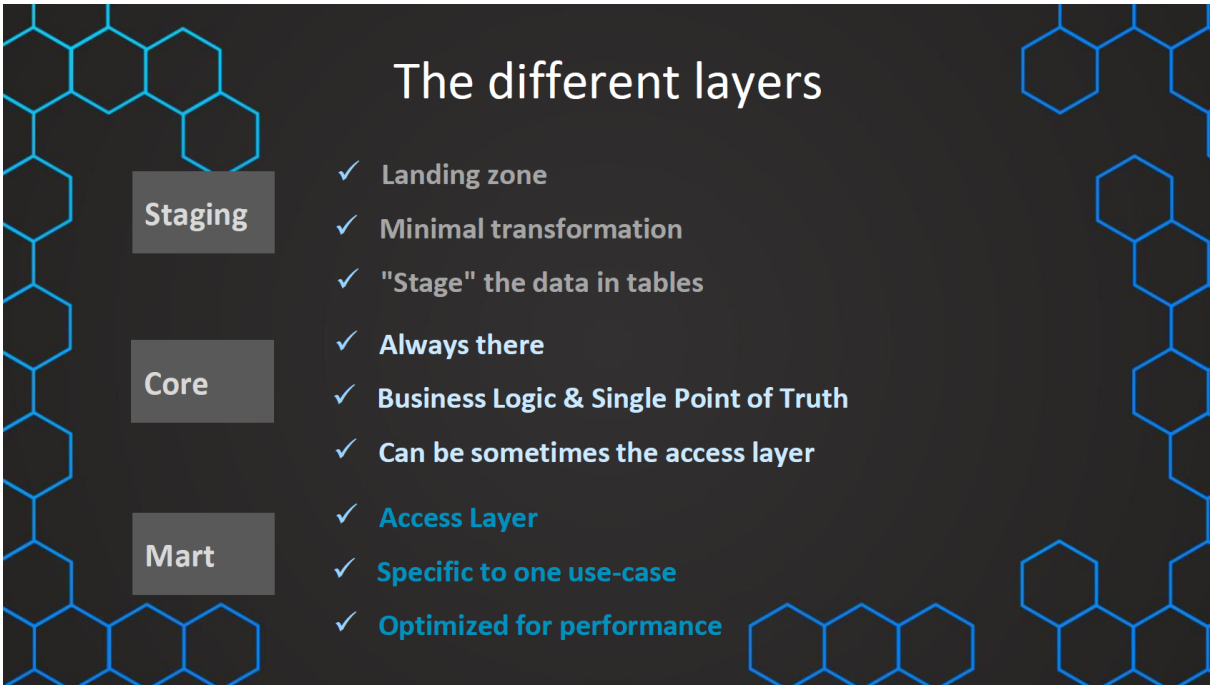




Summary



- Data warehouse consists out of different layers. We have our data sources in the beginning, and then we use our staging layer as kind of a landing zone where we have all of the data loaded into tables in a database and then we have our core layer. This core layer is always available and is serving as our single point of truth. So this means that this is the data source, the only data source for our data marts, which are usually our access layer.
- The core can in some very simple cases, also be used as an access layer, but usually we want to build different data marts with a special purpose or a special use case, and they are then optimized for quick performance and data access and they then are used for different applications or different user groups.



- Staging :
  - This is the landing zone, this is where we will load the data from various data sources into relational database table
  - In this layer we do little transformations as possible
- Core :
  - This layer can be access layer sometimes
  - From landing to core layer we do all the business transformation and the data in this core layer can be considered as single point of truth to make our strategic decisions
  - The data is transformed and modeled in a dimensional way
  - This layer contains all of our data in correct and transformed state
  - This layer is data source for data marts
  - If we have only 1 use case and then this core layer can become the access layer and be called as datawarehouse
- Data mart :
  - Contains data for a specific use case or specific user groups
  - This a access layer

- We also use in-memory databases or cubes to improve the query performance of data marts if our relational database is not fast enough