Week - 6 Basics of SQL - I

Module 1 : Database Design

Introduction:-

One of the biggest challenges while dealing with data is collating it from various sources and managing its size and complexity. It is risky to manage data from different files and this often leads to inconsistency. An organisation cannot track its business activities efficiently by maintaining a large number of files. Hence, databases are used for accessing and managing data efficiently. Handling a complex and large database has to be planned accordingly especially when the type of data you are managing is expected to grow massively. This is why there's a need for **Database Management System (DBMS)** and industries use this widely to analyse data easily.

In this module, you will learn some important concepts of database design, such as data warehousing, ERDs, star schemas, and DDL and DML statements. It is important to be thorough with these concepts before you go ahead and learn how to create and query a schema. Let's hear from Shreyas in the upcoming segments as he elaborates on the topics mentioned above.

In this session

You will be introduced to the basic concepts of a data warehouse, star schema, OLAP (Online Analytical Processing), OLTP (Online Transactional Processing), ETL (Extraction, Transformation, and Loading), constraints and ERD (Entity Relationship Diagrams).

Data Warehouse:-

Imagine that you are a data analyst at a retail company, such as DMart, and let's consider three departments: Marketing, Sales and Finance. Now, let's assume that each department maintains a separate database which is nothing but an organised collection of data. These databases could lead to a situation where each department has its own version of the facts. For a question like "What is the total revenue of the last quarter?", every department might have a different answer. This occurs because each department draws information from a different database. You can imagine how serious this might get because of the scale at which DMart operates.

This is where a data warehouse can help in creating a single version of the truth and facts. A data warehouse would thus be the central repository of data of the entire enterprise. The video given below will help you start off with an understanding of what a data warehouse is and how it is useful for companies in carrying out data analytics.

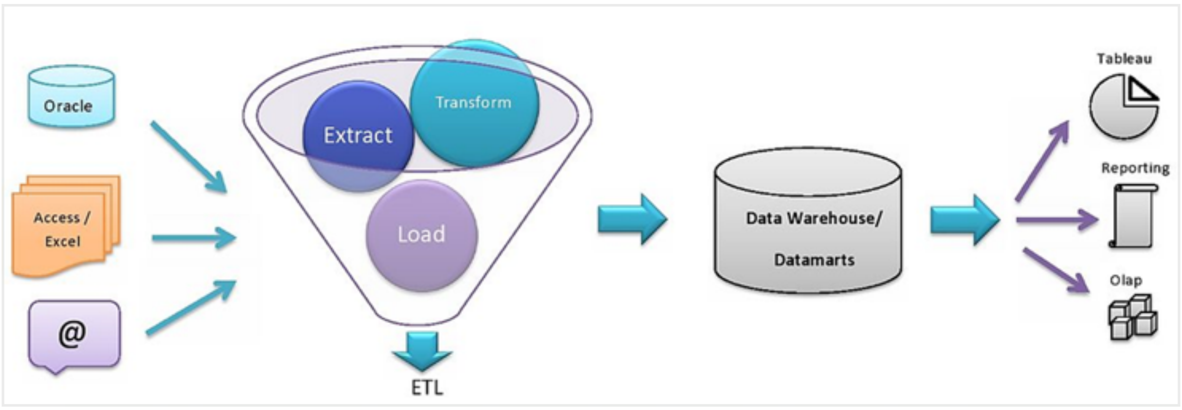
As explained by Shreyas, a data warehouse is conceptually similar to a warehouse. Let's look at the similarities given below:

Warehouse	Data Warehouse
1. It is used by a retail company to store all its goods.	1. It is used by a company to store all its data in databases.
2. Goods are stored such that their retrieval and management is easy.	2. Data is stored such that its retrieval and management is easy.

As you learnt in the video given above, a data warehouse is characterised by the following:

- **Subject oriented:** Built for a specific purpose
- **Integrated:** Collated from multiple sources into one form
- **Non-volatile:** Data does not change over time
- **Time variant:** Capable of capturing information over time

Now, let's understand the journey that any captured data takes in an organisation using the diagram given below.



Journey of Data

As you can understand from the diagram provided above:

1. Data is captured from multiple sources and stored in Excel files or in databases, such as Oracle.
2. The Extract, Transform, Load (ETL) process is performed on the captured data in the following manner:

- **Extract:** Extracting the information from the data sources
- **Transform:** Transforming the values to the required format, such as converting rupees to dollars
- **Load:** Loading the data into a central repository, which is none other than a data warehouse

3. The stored data is either connected to Online Analytical Processing (OLAP) for further processing or used for creating reports in tools, such as Tableau or PowerBI. You will be learning about OLAP in the subsequent segments of this session.

A dimension model is a database structure technique and is used to optimize the database for the fast retrieval of data. The dimensional model contains two entities: Facts and Dimensions. Let's understand this better in the following video.

Facts are the numerical data and dimensions are the metadata (that is, data explaining some other data) attached to the fact variables. Both facts and dimensions are equally important for generating actionable insights from a data set.

Now the precursor to designing a database is drawing a well-defined **Entity-Relationship Diagram**, **or ERD.** It can be thought of as a map of the database schema. We can visualise the structure of the entire schema and answer the following questions just by looking at the ERD:

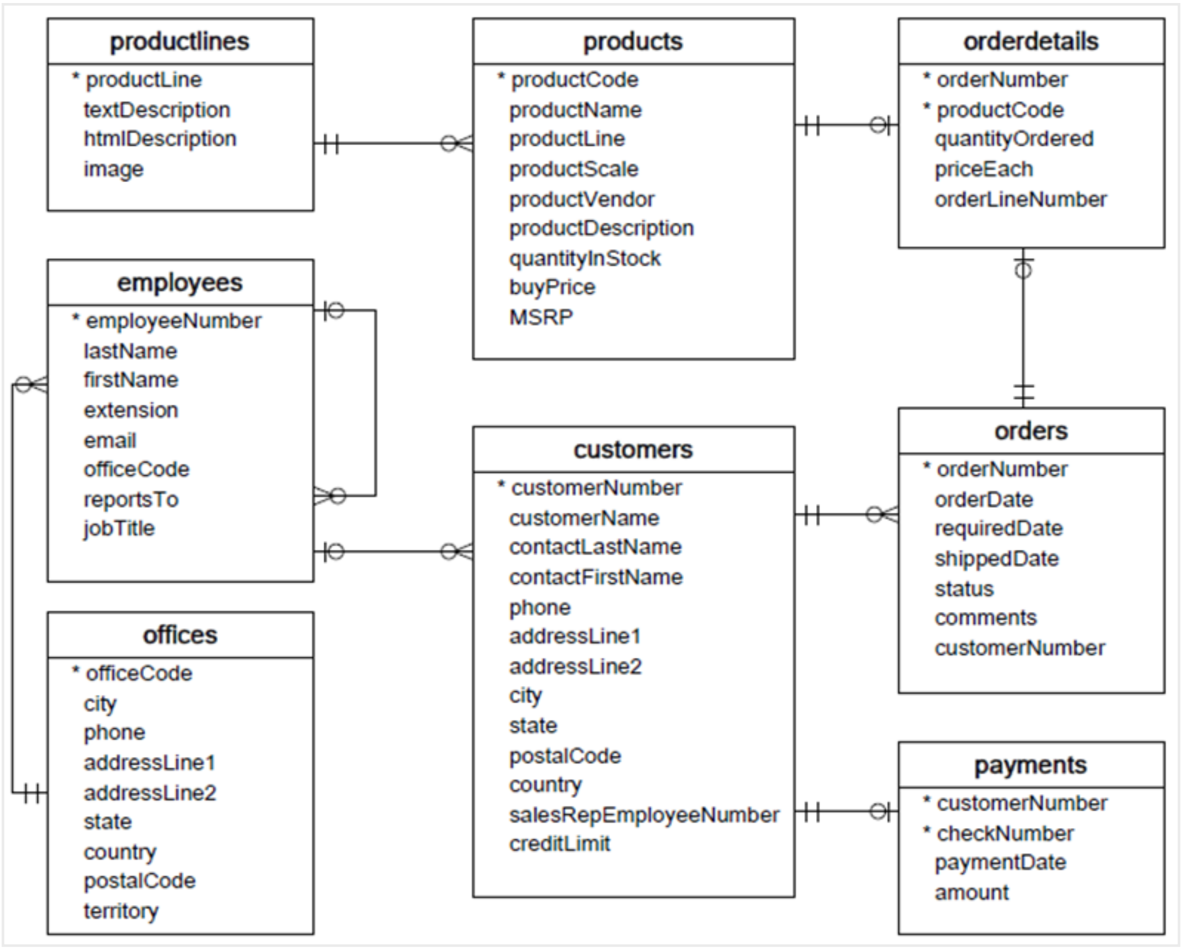
- What are the tables that it contains?
- What are the columns that each table contains?
- What is/are the data types and constraint/s (if any) for each column?
- What are the relationships between the various tables?

ERD:-

Consider a map of India. As you know, it is a visual representation of how far the states, mountains and rivers are from each other. Just by looking at the map, you can determine the layout of India; you need not travel the entire country. Similarly, an Entity Relationship Diagram (ERD) is a representation of the tables in a database. It gives us an idea of how different entities are connected. ERDs are extremely useful to get an overall idea of a database in very less time. In the video provided below, let's find out how Shreyas views the importance of an ERD.

As you saw in the video given above, designing an ERD is a fundamental aspect of data modelling. We always start with an ERD before we start creating the required tables for the schema. In the next video, let's take a look at an ERD and understand how to read it.

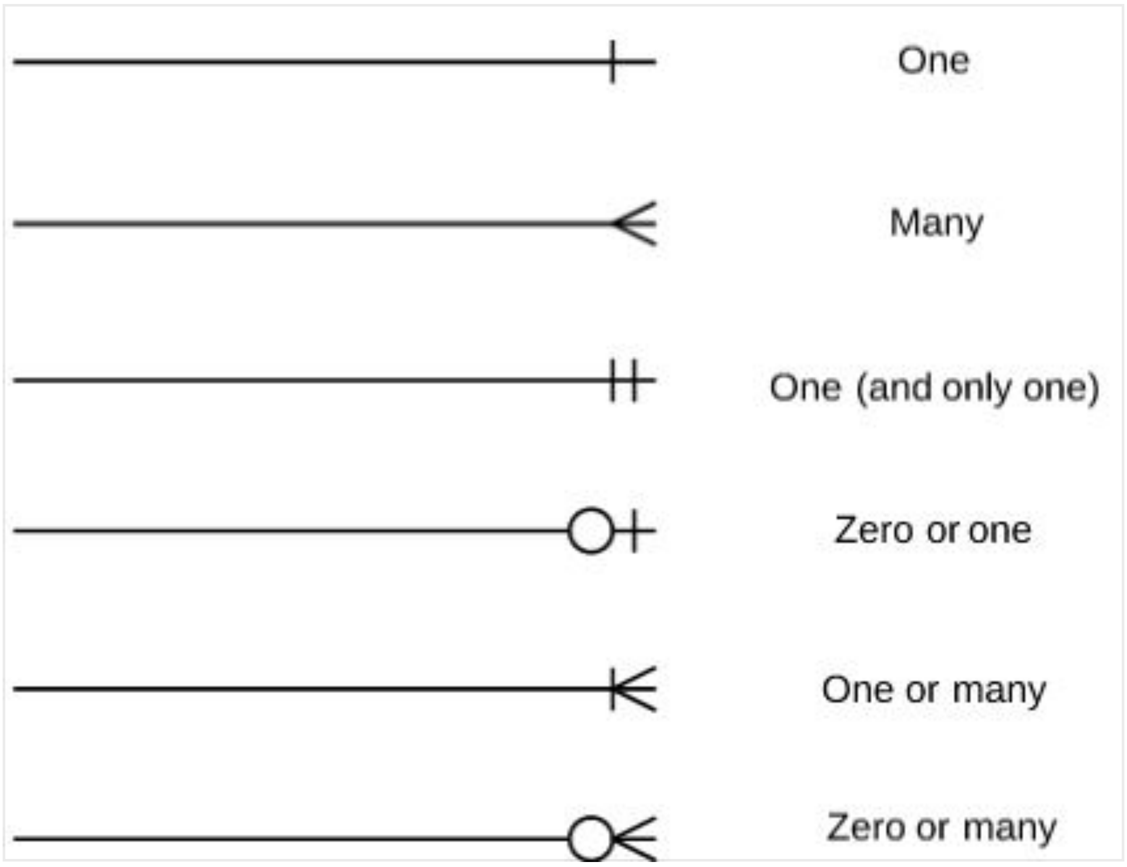
Let's take another look at the ERD used in the video given above and try to understand the notations used. These notations have been detailed in the image given below.



Here is a short summary of what each notation in an ERD signifies:

- Fields marked with a star are the primary keys for each table.
- A line connecting two tables indicates that a relationship exists between them.
- A line connecting a table to itself indicates that the table references itself.

The diagram given below shows the types of cardinalities that can exist in a relationship.



There are four types of relationships that exist in an ERD:

- **One to One:** One to one type of relationship exists, when a single instance of an entity is related to only a single instance of another entity.
- **One to Many:** One to many type of relationship exists, when a single instance of an entity is related to more than one instance of another entity.
- **Many to One:** Many to one type of relationship exists, when more than one instance of an entity is related to only one instance of another entity.
- **Many to Many:** Many to many type of relationship exists, when more than one instance of an entity is related to more than one instance of another entity.

Now that you know how to design and read ERDs, let's continue with the other important database design concepts. You will learn more about star schemas and snowflake schemas in the upcoming segment.

Star and Snowflake Schemas:-

In the previous segments, you learnt about facts and dimensions, which are the two key elements of dimension modelling. Now, a typical problem might involve multiple databases with many different variables, but we may not be interested in all of them. Hence, only some facts and dimensions are combined in a specific manner to build the structure of a data model, called a schema diagram.

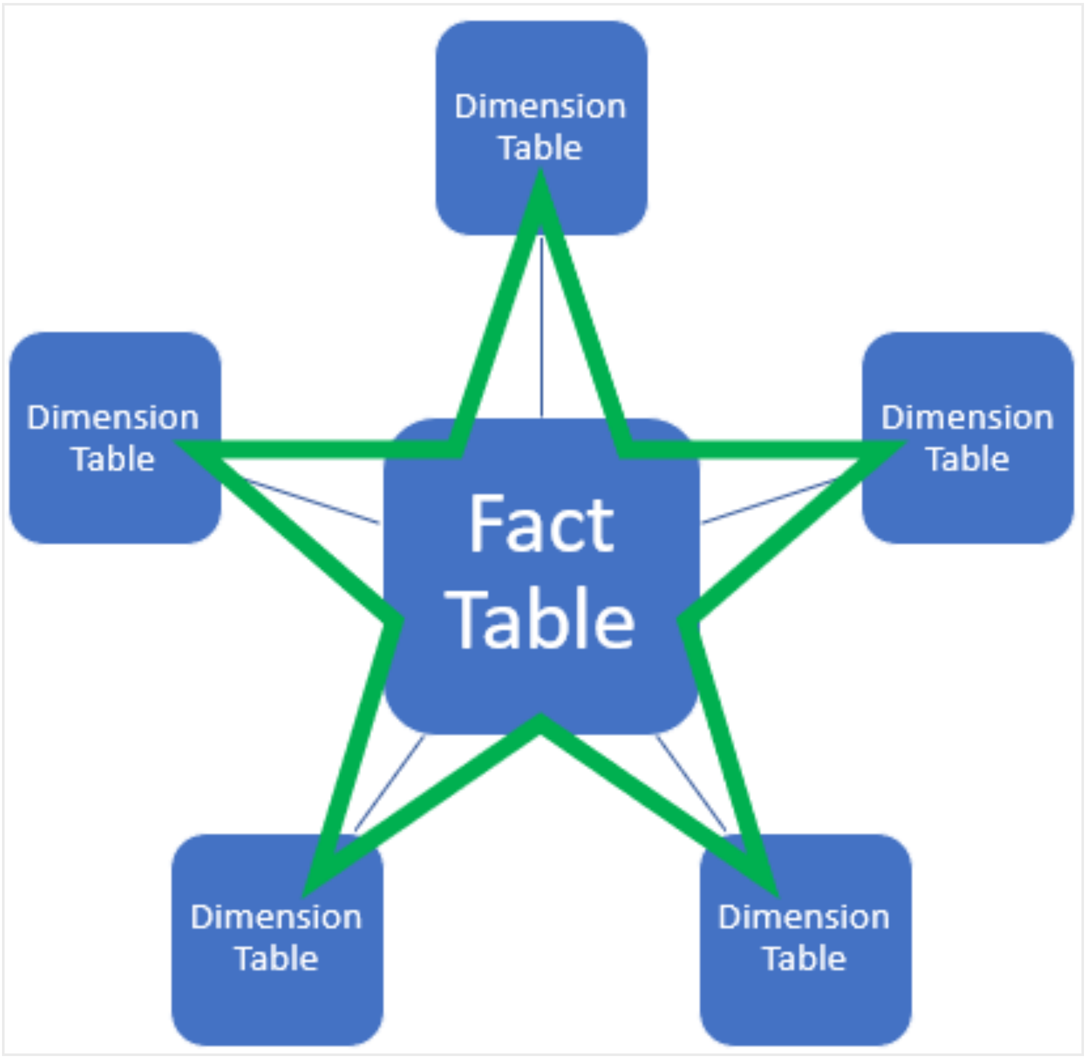
A schema is an outline of the entire data model which shows how different data sets are connected and how the different attributes of each data set are used for the database design.

A star schema is a type of schema design that has one fact table connected to multiple dimension tables. Let’s watch the following video to understand star schema in detail.

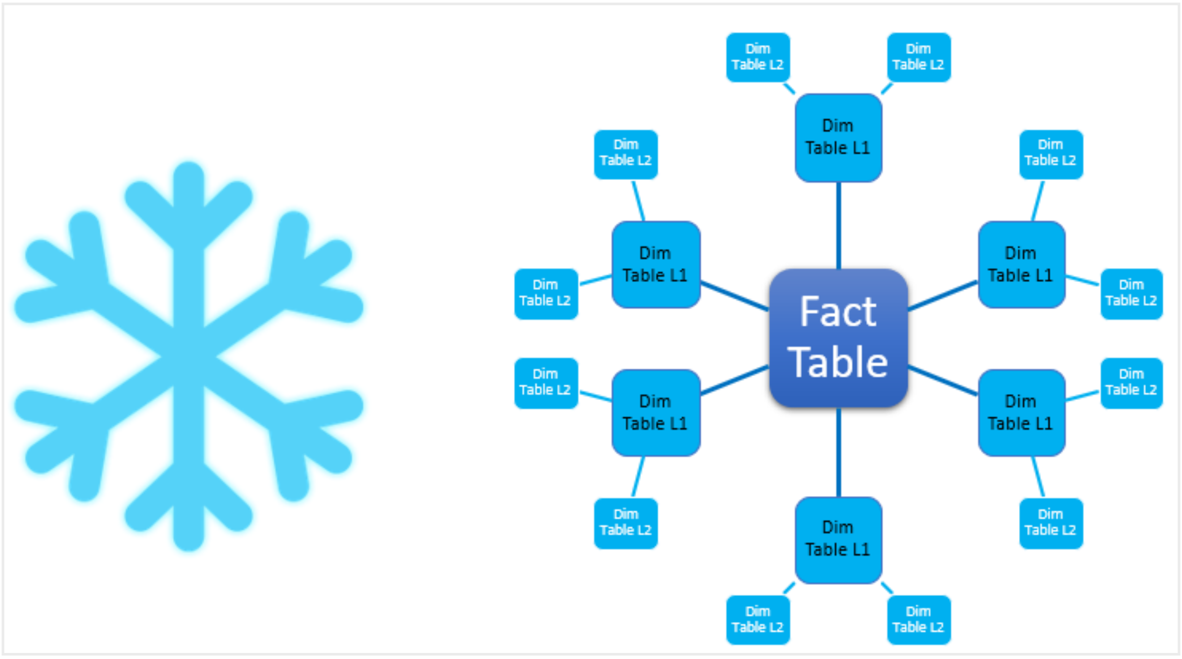
Some of the advantages of star schemas are as follows:

- Easy to understand
- Easy to implement
- Efficient in terms of querying because a maximum of only one join is required

In the next video, let’s take a look at the structure of a star schema and understand it better.
In the video above, you also learnt about another type of schema: the snowflake schema. Let’s look at the diagrams provided below and understand the structure of each. A star schema has a star-like shape, as shown.



A snowflake schema is called so because the dimension tables surrounding the fact table can branch off into more dimension tables. Check out the diagram given below to get a better understanding of the same.



Snowflake schemas are particularly useful when it becomes difficult to manage the size of the dimensional tables. For example, a 'Customer' dimension may have an attribute called 'CityID'. The 'Customer' table would be drawn out and difficult to read if it had all the details of the city to which a customer belongs. Hence, it is a good idea to store the details of the city, such as the city name and the PIN code, in another dimension table.

Note that querying would be more time-consuming for such schemas, as you would require to write multiple join statements. On the other hand, snowflake schemas are efficient in terms of data storage.

You can refer to the links given below to know more about star and snowflake schemas:

- Star and snowflake schemas

OLAP vs OLTP:-

Now that you have developed a fair understanding of databases, you might be wondering how a data warehouse and a database differ from each other.

So, in this segment, you will learn exactly which features of a data warehouse differentiate it from a regular database. In the upcoming video, Shreyas will talk about the differences between a transactional database and a data warehouse.

So, in the video, you learnt about the differences between a transactional database (i.e., **OLTP, or Online Transactional Processing**) and a data warehouse (which is often referred to as **OLAP, or Online Analytical Processing**). Notice that the major difference between OLAP and OLTP is apparent in the names themselves: OLTP is used for day-to-day transactions, whereas OLAP is used for analytical purposes.

Earlier, you were introduced to the terms **dimensional modelling** and **star schema.** They are essential for creating the structure of a data warehouse. These techniques involve finding out the variables on which analysis can be performed and then combining them with the metadata to derive meaningful insights. You will learn more about them in the next session.

Additional Resources:

1. You can go Data Warehousing Concepts to learn more about a data warehouse and its characteristics.
2. You can go OLTP vs. OLAP to learn more about the differences between OLAP and OLTP systems.

In this way, you have understood about OLTP and OLAP systems. In the next segment, you will learn about the important concept of constraints and why they are important while handling a database schema.

Entity Constraints:-

You have already learnt about the ETL process, which is used to ingest data into a schema and perform operations on it. But can we add just any value that we want to a schema or are there any constraints to maintain the sanctity of the schema being defined?

Put yourself in the place of a data analyst at Uber. A data warehouse at Uber has several tables, which record several details, such as details on the rider, the driver, the vehicle used and transaction details. Each such table has several related attributes, which describe it in detail. Consider the 'Rider' table. You go through the values entered in a column that contains the fares for each ride. It would be right for you to expect that the fares can have a maximum of four digits – a fare more than even 5,000 would raise concerns. This is definitely an outlier and needs to be replaced with the right value. Ensuring that you implement the right constraints for the right attributes in a table can help you avoid such fallacies.

So, as you learnt in the video, constraints are the rules that are used in MySQL to restrict the values that can be stored in the columns of a database. This ensures data integrity, which is nothing but the accuracy and consistency of the data stored in the database. Let's continue our discussion on one of the types of constraints called entity constraints in the next video.
So, as you learnt in the video, entity constraints are of the following different types:

- **Unique:** Used for columns that need unique values.
- **Null:** Determines the columns that can have null values.
- **Primary Key:** Determines the column that uniquely identifies a table.

Note that there may be a situation wherein, say, a company wants to store the records of its employees over multiple years. In this case, 'employee id' may not be unique, since an employee will have multiple rows storing details of multiple years. Also, you will need a new column named 'year' in the table.

In this case, a combination of an employee id and a year, i.e., a variable EmpID-Year, can act as a **composite primary key**. To see how this is done in MySQL, you can refer to this Stack Overflow answer.

Referential Constraints:-

In the previous segment, you learnt about the first type of constraints, entity constraints, which pertain to the values in a single table. The second type of constraints is called referential constraints. These are used to restrict the values that are taken by a column in one table based on the values that exist in another table.

Consider the tables given below.

Customers

CustomerID	FirstName	LastName	Age
1	John	Wick	52
2	Sheldon	Cooper	41
3	Charlie	Harper	45

Orders

OrderID	OrderNumber	CustomerID
1	12345	2
2	31343	2
3	12466	3
4	41234	1

CustomerID is a foreign key in the 'Orders' table because it is used to reference the 'Customers' table. You can easily determine which customer placed a particular order and find out all the details of that customer by looking up the corresponding customer id in the 'Customers' table. Watch the upcoming videos to understand how foreign keys make it easier to design and query databases with the help of referential constraints.

So, as you learnt in the video, a referential constraint is a rule between two tables. According to this rule, the value that appears as a foreign key in a table is valid only if it also appears as a primary key in the table to which it refers.

Also, note that a given table has only one primary key but it can have multiple foreign keys. Before you assign a column as a foreign key, you need to ensure that the primary key column of the table that it refers to is present and it does not have null or duplicate values.

Semantic Constraints:-

Suppose you work at Tata Motors and are asked to analyse the data on the prices of different car models. The values can range from 2 lakhs to around 12 lakhs if you are looking to sell the cars to middle-class households. Now, imagine you come across a car model that costs 1 crore. You would need to get rid of this value as it would negatively affect your analysis and generate misleading insights. How can you ensure such values (also known as outliers) are not present in your data?

So, as you learnt in the video, semantic constraints impose additional restrictions on the values in a column. Using a semantic constraint ensures that we do not get incorrect data for any row in the database. For example, a row with an Indian phone number not having 10 digits after the country code would not be allowed to enter the database.

Summary:-

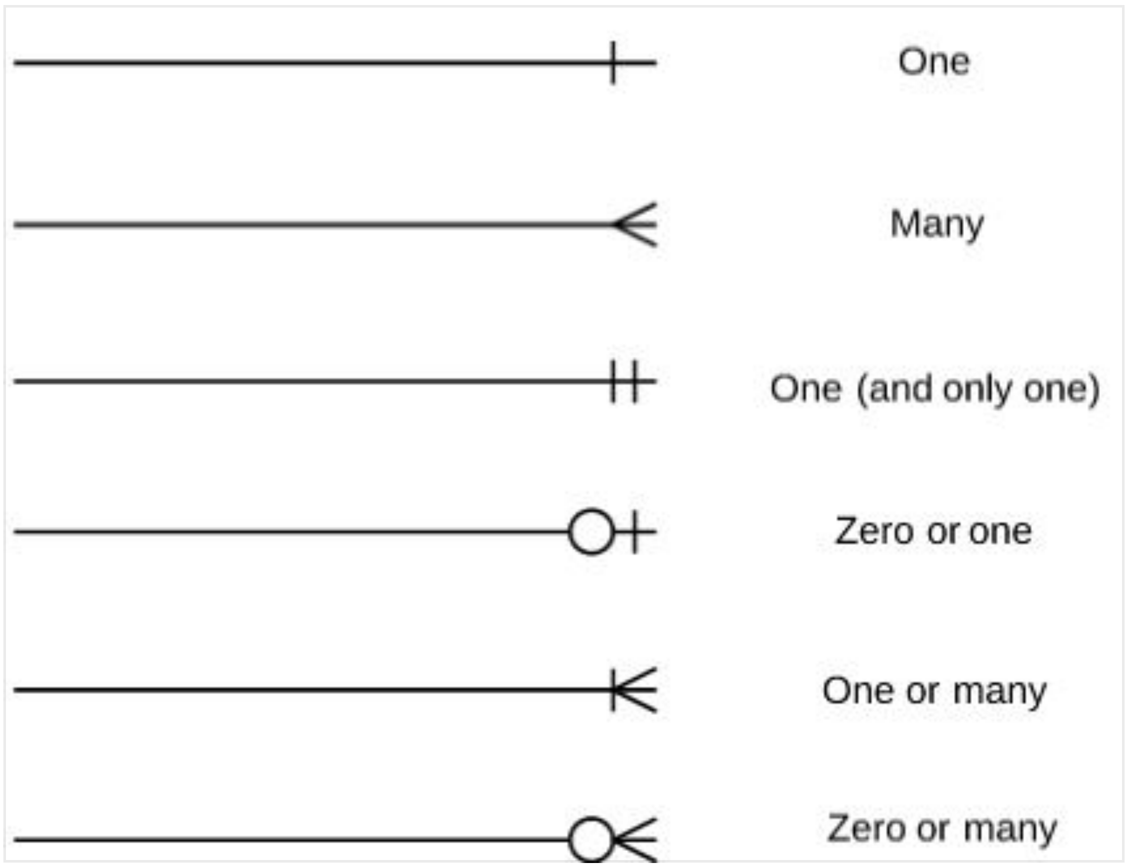
In this session, you learnt some basic concepts of database design. Let's revisit the topics that were covered in this session.

Data warehouse: You learnt that a data warehouse is a collection of tables containing data that is structured according to some pre-decided specifications. It provides an integrated view of the entire organisation and the data is organised for carrying out analysis efficiently. It is:

- Subject oriented,
- Integrated,
- Non-volatile
- Time variant.

Facts and dimensions are the two major components in designing a data warehouse. Dimension tables act as metadata, which is data about data, and enhance the facts table to enhance the insights from the data. You also understood about the Extract, Transform, Load (ETL) process, which is a series of operations that data undergoes in an organisation.

ERD: You learnt how to draw and read Entity Relationship Diagrams (ERDs) according to business requirements, with the help of an example. You also saw the possible cardinalities that can exist in a relationship between two tables. These are mentioned in the diagram given below.



Star and snowflake schemas: You compared a star schema with a snowflake schema. A star schema has one central fact table surrounded by multiple dimension tables, while a snowflake schema can have dimension tables that branch off into more such tables. While a star schema makes querying data easier, a snowflake schema is more efficient in terms of data storage.

Finally, you studied the different types of constraints: Entity, Referential and Semantic constraints which are essential in maintaining the sanctity of a database schema.

In the next session, you will be introduced to some basic concepts on SQL querying involving DDL and DML statements.

Module 2 : Basic SQL Querying

Introduction to SQL:-

As you learnt in the video above, SQL stands for Structured Query Language. In some applications, it is also referred to as DSL, or Domain Specific Language. SQL is used for managing and manipulating data held in a relational database management system (RDBMS).

SQL helps you to:

- Create data,
- Manipulate data,
- Delete data, and
- Share data

Why should y
ou learn SQL?

- SQL is **language-agnostic**, which makes it a language that can be easily learnt and comprehended.
- It is also **supported by all the modern database systems** such as Oracle, MySQL, SQL Server and so on.

Have you ever wondered how data is stored in a database? What if this data is interlinked? Do we have any system in place that can look into such a situation?

RDBMS helps to store your data while preserving the relationships within the data. It is a Database Management System that stores data in the form of tables, where each table is related to the other tables in the database. This kind of model follows a relational model and hence, it is known as a relational database management system.

Before that, here is a list of the ten most popular RDBMS systems presently used:

- MS SQL
- MySQL
- Oracle Database
- IBM Db2
- PostgreSQL
- Amazon Aurora
- Amazon Relational Database Service (RDS)
- SAP HANA
- IBM Informix
- MariaDB

Amongst them, you will be working with MySQL. This is because:

- It is open-source, meaning it is completely free.
- It supports all the features of an RDBMS.
- It has a wide user community. This means that you can easily find communities that are working with MySQL, who can guide you properly, and after this module, you can be part of that community, too.

Here are some of the common industrial applications of RDBMS in our day-to-day lives.

- The banking sector: Managing the flow of money from one account to another is a key operation in the banking sector. Because of its ACID properties, RDBMS is the ideal DBMS to handle complex transactional operations.
- E-commerce websites: Companies such as Amazon and Flipkart are heavily dependent on RDBMS for operations such as product categorisation, wherein there are relations between different products. For example, when you buy a product, your website provides suggestions of other products that may be relevant to your purchase. (You learnt about this under Association Rule Mining.) However, these websites rely heavily on RDBMS to manage and store these products and relations.
- Social networking sites: Every time you log in to your account, Facebook provides suggestions of 'people you may know' by analysing your mutual friends. This is done by maintaining the relationships between the different entities (i.e. the different users on the platform). This kind of relational mapping can be stored only with the help of RDBMS.

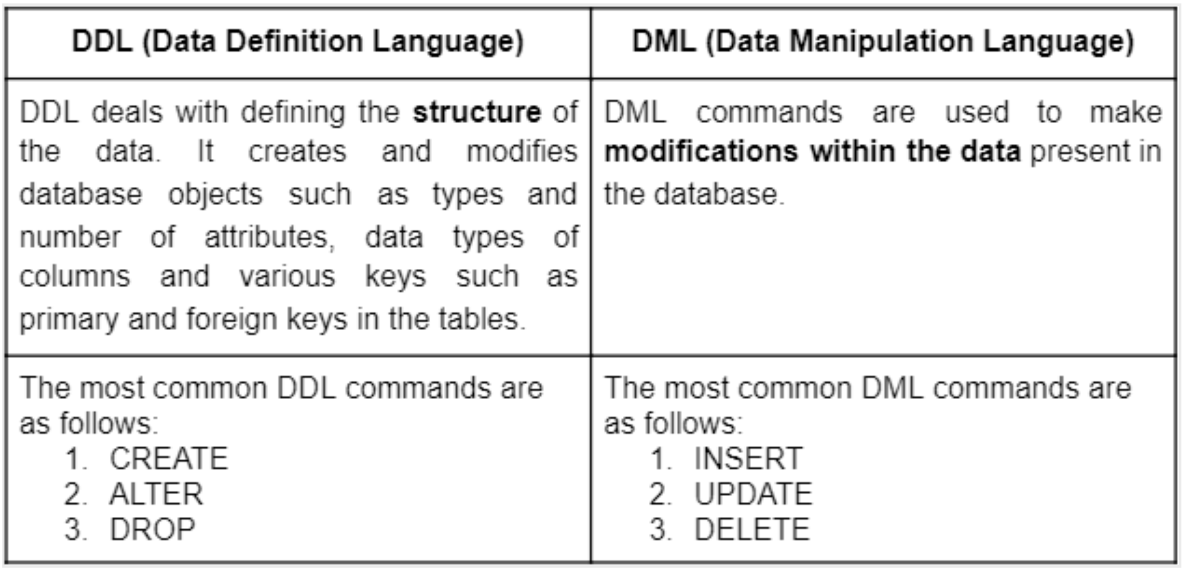
Before going to the next segment let's download MySQL workbench and server in your system to perform the queries.

DDL Statements:-

Now that you have understood that RDBMS plays a vital role in our day-to-day lives, let's learn SQL, starting with some basic syntaxes. In this segment, you will learn about DDL statements. Before we move onto learning these syntaxes, let us first understand the two basic subcategories of SQL commands.

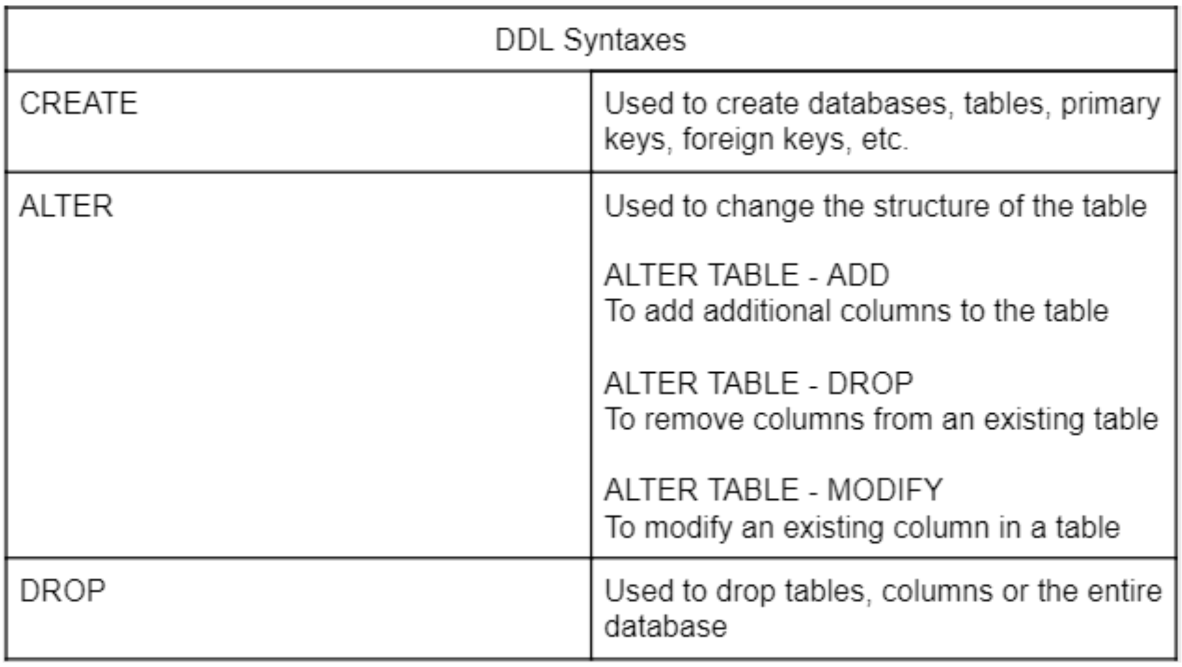
SQL commands are mainly divided into two subcategories:

- DDL (Data Definition Language), and
- DML (Data Manipulation Language)



DDL (Data Definition Language)	**DML (Data Manipulation Language)**
DDL deals with defining the **structure** of the data. It creates and modifies database objects such as types and number of attributes, data types of columns and various keys such as primary and foreign keys in the tables.	DML commands are used to make **modifications within the data** present in the database.
The most common DDL commands are as follows: 1. CREATE 2. ALTER 3. DROP	The most common DML commands are as follows: 1. INSERT 2. UPDATE 3. DELETE

Let's revise them one by one.

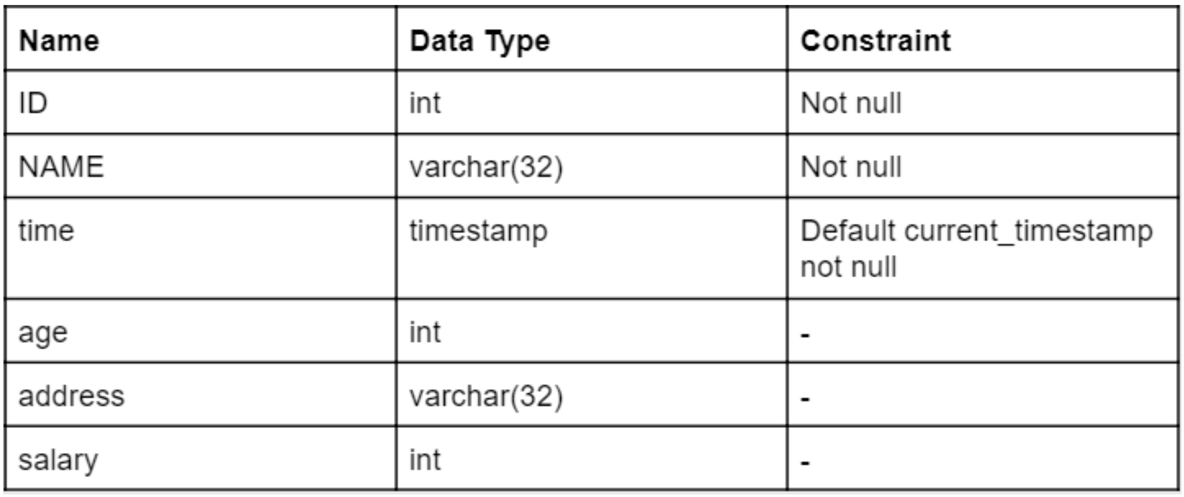


DDL Syntaxes	
CREATE	Used to create databases, tables, primary keys, foreign keys, etc.
ALTER	Used to change the structure of the table ALTER TABLE - ADD To add additional columns to the table ALTER TABLE - DROP To remove columns from an existing table ALTER TABLE - MODIFY To modify an existing column in a table
DROP	Used to drop tables, columns or the entire database

Let's summarise what you learned in the video above.

You learned how to use the CREATE command and executed the following operations using this syntax:

- You created a database with the name 'demonstration'.
- Following this, you created a table named 'customers' with the following attributes, data types, and constraints:



Name	Data Type	Constraint
ID	int	Not null
NAME	varchar(32)	Not null
time	timestamp	Default current_timestamp not null
age	int	-
address	varchar(32)	-
salary	int	-

After creating the table, you used the ALTER command to make the following modifications to the table:

- The ‘ID’ column in the ‘customers’ table was made as the primary key.
- A new column ‘employer’ was added to the ‘customers’ table with VARCHAR(32) as the data type.
- The column ‘employer’ was dropped (deleted) from the ‘customers’ table.

Following this, you learned how to use the DROP command to execute the following operations:

- Dropping the ‘customers’ table, and
- Dropping the database ‘demonstration’.

In the next segment, you will learn about the second set of commands known as DML (Data Manipulative Language).

DML Statements:-

Now you will look at the next set of commands, which is known as DML, or Data Manipulation Language. As you learned, DML commands are mainly used to manipulate data as per our requirements.

DML Commands	
INSERT	This command is used to add data to existing columns in the table
UPDATE	This command is used to update existing rows in the table of RDBMS.
DELETE	This command is used to delete specific rows from the tables of RDBMS.

Now, let’s jump to the code implementation of each of these commands. In order to demonstrate their use cases, a table named ‘transportation’ has already been created with the following attributes:

Name	**Data Type**	**Constraint**
ship_mode	varchar(25)	-
vehicle_comapany	varchar(25)	-
toll_required	boolean	-

Let’s revise the use cases of each of the commands shown in the video above:

- The **INSERT command** was used to add the following records to the table:
  ('DELIVERY TRUCK', 'Ashok Leyland', false)
  ('REGULAR AIR', 'Air India', false)
- The **UPDATE command** was used to modify the attribute ‘toll_required’ to ‘True’ wherever ‘ship_mode’ contained the value ‘DELIVERY TRUCK’.
  Note: We have also used the WHERE command for the purpose of this demonstration, which will be explained in detail in the next segment. The WHERE command is used to filter out specific rows of a table.
- Next, the **DELETE command** was used to delete all the rows where the attribute ‘vehicle_company’ had the value ‘AIR INDIA’.
  Note: Executing the same command without the WHERE clause will result in deleting all the rows of the table.
- Using the **ALTER command**, we added a column known as ‘vehicle_number’ and set its data type to varchar(32). Following this, the UPDATE command was used to update all the records in this column to ‘MH-05-R1234'.

There can be additional subcategories besides DDL and DML. You may refer to this link to explore the different commands within SQL.

Note:

- If you wish to view your table at any point in time, use this command: SELECT * FROM table_name
- The commands in SQL are **not case-sensitive**, I.e., you can use an uppercase or a lowercase to write your SQL queries.

In the next segment, you will learn how to generate an ERD in MySQL once you define your database and its tables. After that, you will learn about some of the common SQL operators and their use cases with the help of a case study.

SQL Basic Statements and Operators:-

In the previous segment, you learned about the various DDL and DML statements. In the coming segments, we are going to work on an e-commerce data set and fetch relevant insights using SQL.

In the next video, Vishwa will begin by introducing you to the e-commerce process. You will also learn how to create the ERD for this database using MySQL.

This case study is about 'market_data', where you are given a total of five tables. The description of the tables is as follows:

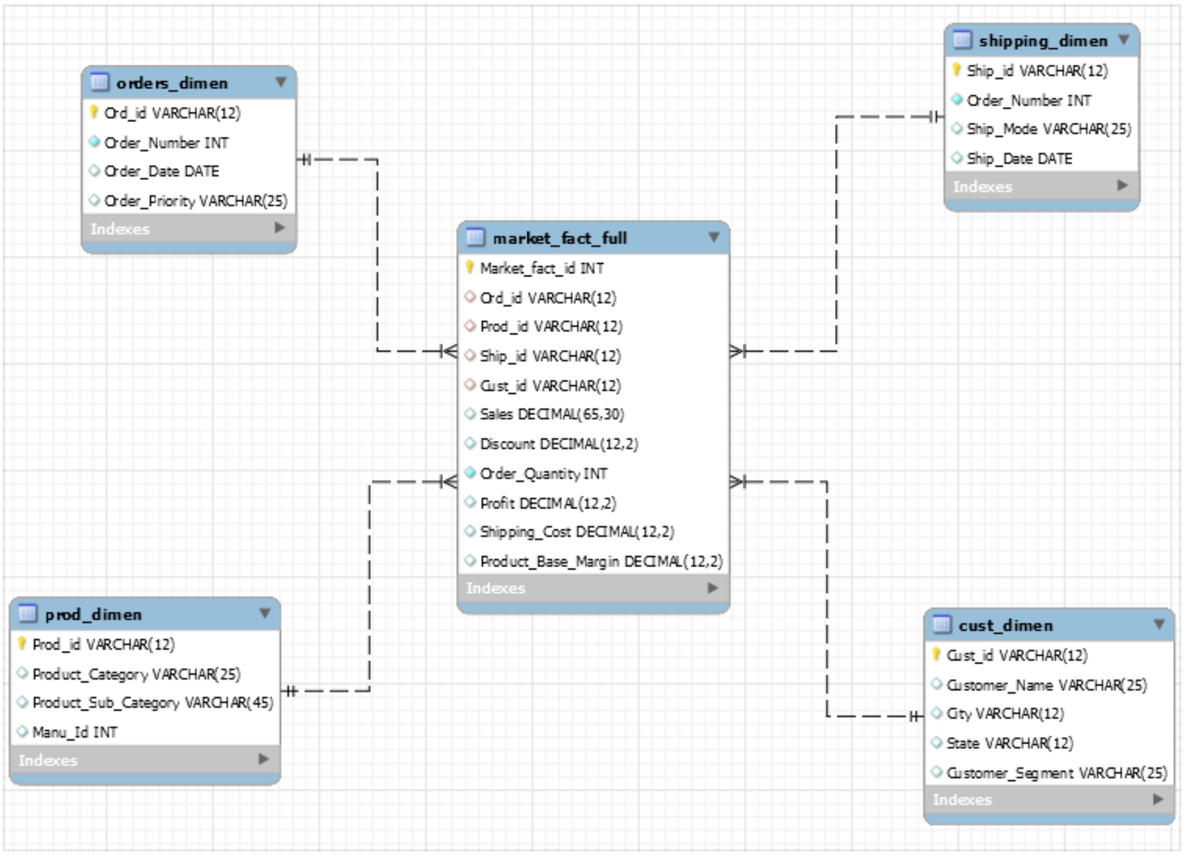
- **cust_dimen**: This table contains information regarding the customers. It defines their customer ID, customer name, city, state and the nature of their business.
- **shipping_dimen**: This table stores the data of the shipping modes for each order ID placed by the customers. Its attributes are order_ID, ship_mode and ship_date.
- **orders_dimen**: This table contains information about the orders placed by customers. It contains the Order_Number, order_ID, order_date and the order_priority. The order ID denotes the ID of the order placed by the customer. In the same order, the customer may order multiple products, where the ID of each product is given by the ‘Order_Number’.
- **prod_dimen**: This table contains information about the products of a company that are going to be sold to the customers. It includes information about product types and product subcategories.
- **Market_fact_full**: This table contains the details of each order, regarding the customer who placed the order, shipping details and the product details. It contains the foreign keys of all four tables described above.

Download the workbench given below which includes the query for table creation and few comments. As you move along the module you are expected to use this workbench to practise hands-on.

you learnt how to perform a case study of an e-commerce process. The database was structured and the data was inserted using various DDL and DML commands that you learnt in the previous segment. Finally, with the help of MySQL, an ERD was constructed depicting the relationship of all the tables within this database as shown below.

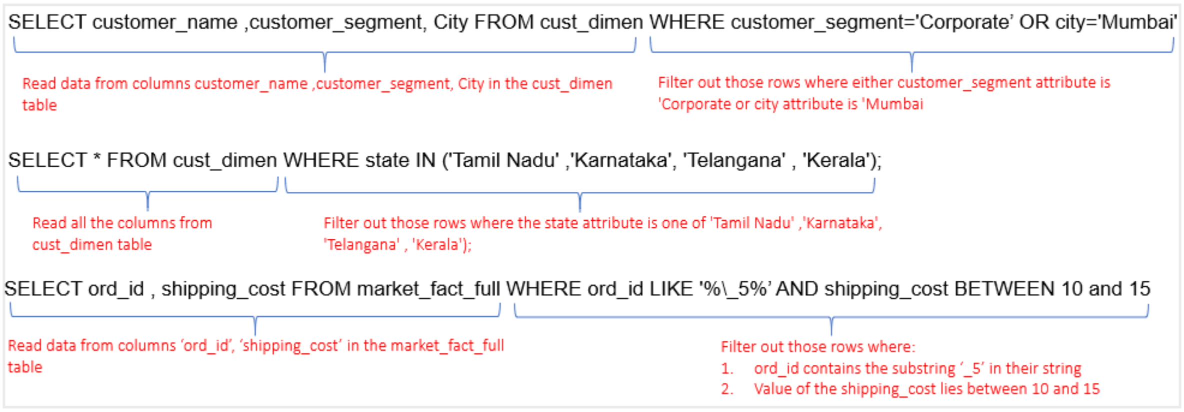
You also learnt about the different data types that can be used in MySQL. To understand further, refer to this link.

Note: In the video above, Vishwa used the keyword 'KEY'. This attribute has been used to apply indexing to the tables. By applying indexing, search queries can be executed faster.



Let us revise the commands that you learned in the earlier videos.

1. **SELECT**: Used to read the data
2. **WHERE**: Used as a conditional statement to filter out data
3. **OPERATORS**: Some examples of operators that can be used along with the WHERE clause are 'OR', 'AND', 'IN', 'BETWEEN'.



```sql
SELECT customer_name ,customer_segment, City FROM cust_dimen WHERE customer_segment='Corporate' OR city='Mumbai'
```

Read data from columns customer_name ,customer_segment, City in the cust_dimen table

Filter out those rows where either customer_segment attribute is 'Corporate or city attribute is 'Mumbai

```sql
SELECT * FROM cust_dimen WHERE state IN ('Tamil Nadu' ,'Karnataka', 'Telangana' , 'Kerala');
```

Read all the columns from cust_dimen table

Filter out those rows where the state attribute is one of 'Tamil Nadu' ,'Karnataka', 'Telangana' , 'Kerala');

```sql
SELECT ord_id , shipping_cost FROM market_fact_full WHERE ord_id LIKE '%\_5%' AND shipping_cost BETWEEN 10 and 15
```

Read data from columns 'ord_id', 'shipping_cost' in the market_fact_full table

Filter out those rows where:
1. ord_id contains the substring '_5' in their string
2. Value of the shipping_cost lies between 10 and 15

The **LIKE** operator is a logical operator that checks whether a string contains a specified pattern or not. The following two wildcards are used to specify the pattern:

1. '%' allows you to check with a string of any length.
2. '_' allows you to check with a single character.

Consider the following examples:

Example	Description
`SELECT Customer_name FROM cust_dimen WHERE customer_name LIKE 'CLA%';`	This will return all the names of the customers whose names start with 'CLA'
`SELECT Customer_name FROM cust_dimen WHERE customer_name LIKE 'CL_IRE GOOD';`	OUTPUT: CLAIRE GOOD Returns those strings that contain a single character between 'CL' and 'IRE GOOD'
`SELECT Customer_name FROM cust_dimen WHERE customer_name LIKE '%IRE%';`	This statement will return all the strings which contain the substring 'IRE'.

Note: In the example demonstrated above, the escape operator ('\') has been used to search for strings containing the substring '_5'. This is because the character '_' is itself an operator. Hence, we need to specify the escape character so that MySQL interprets '_' as a literal character. To understand the LIKE operator in-depth, you can refer to this Link.

Aggregate and Inbuilt Functions:-

The aggregate and inbuilt functions play a crucial role in analyzing the data or extracting relevant insights. Let's learn some of the basic functions from Vishwa before we discuss the code demonstration.

Vishwa discussed the importance of aggregation, grouping and inbuilt functions.

The aggregation functions are as follows:

1. **COUNT()**: Counts the total number of records specified by the given condition
2. **SUM()**: Returns the sum of all the non-NULL values
3. **AVG()**: Returns the average of all the non-NULL values
4. **MIN()**: Returns the minimum of the specified values (NULL is not included)
5. **MAX()**: Returns the maximum of the specified values (NULL is not included)

The data is grouped using the **GROUP BY clause**. It groups rows that have the same values based on the specified condition and performs operations on the individual groups. Besides this, Vishwa also gave us a few examples of inbuilt functions, such as **UPPER** and **LOWER** clauses (converting a string to uppercase and lowercase, respectively). We will cover strings and other functions in more detail in the coming segment. Let’s understand how to use these concepts in our case study to gain some more insights.

you learned how to use the GROUP BY and the HAVING clause. Let’s revise these two clauses. The syntax is as follows:

SELECT column_names

FROM table_name

WHERE condition

GROUP BY column_names

HAVING condition

`select count(Customer_Name)`
`as City_Wise_Customers, City`
`from cust_dimen`
`group by City;`
	Print the total number of customers from each city within the ‘city’ column.

`select count(Customer_Name)`
`as Segment_Wise_Customers, Customer_Segment`
`from cust_dimen`
`where State = 'Bihar'`
`group by Customer_Segment;`
	Print the total number of customers from Bihar in each segment.

`select Cust_id, sum(Shipping_Cost)`
`as Customer_Wise_Shipping_Cost`
`from market_fact_full`
`group by Cust_id`
`having Customer_Wise_Shipping_Cost > 50;`
	1. Find the sum of the shipping cost for each customer.
	2. Print the customer ID of those customers whose total shipping cost is more than 50.

we will add another condition known as the ORDER clause. As the name suggests, the ORDER clause allows us to sort values either numerically or alphabetically. Let’s see how.

String and Date-Time Functions and Ordering:-

In the last segment, you learnt how to use the GROUP BY and HAVING clauses. In this segment, you will learn how to use the ORDER and LIMIT clauses. You will also gain an understanding of some of the common string and date-time functions that are very useful for analysis.

As the name suggests, the ORDER clause is used to sort the values in the columns in ascending or descending order. The order statement must be followed by ‘asc’ or ‘desc’ in order to specify an ascending order or descending order, respectively. In case nothing is mentioned, the default sort is ascending.

In the previous segment, you were introduced to inbuilt functions such as UPPER and LOWER and many others. Similarly, there are many other useful String and Date-Time functions that can prove to be useful for extracting relevant insights. Let’s look at some of these functions.

STRING FUNCTIONS			
Function	**Example**	**Description**	**Output**
UPPER	SELECT UPPER('upgrad');	Converts to uppercase	UPGRAD
LOWER	SELECT LOWER('UPGRAD');	Convert to lower case	upgrad
SUBSTRING	SELECT SUBSTRING('upgradeability', 1, 6)	SUBSTRING(text, start_index, length of substring)	upgrad
SUBSTRING_INDEX	SELECT SUBSTRING_INDEX('www.upgrad.com', '.', 1)	Returns the substring before the mentioned delimiter	www
	SUBSTRING_INDEX('www.upgrad.com', '.', 2)	SUBSTRING_INDEX(text, delimiter, occurence number of delimiter)	www.upgrad
LENGTH	SELECT LENGTH('upgrad');	Gives the number of characters in the input	6
REVERSE	SELECT REVERSE('upgrad');	Reverses the string	dargpu

DATE-TIME FUNCTIONS		
Syntax	**Description**	**My output**
SELECT CURTIME()	Returns the current time	17:11:12
SELECT MONTH('2020-03-10')	Returns the month of the date	03
SELECT MONTHNAME('2020-03-10')	Returns the name of the month of the given date	March
SELECT STR_TO_DATE('10,03,2020','%d,%m,%Y')	Converts the string to date type	2020-03-10
SELECT YEAR('2020-03-10')	Extracts the year from the date	2020
SELECT DAYNAME('2020-03-10')	Returns the day of the week for the given date	Tuesday
SELECT DAYOFMONTH('2020-03-10')	Extracts the day of the month from the given date	10

There are many such functions that you might find useful. You can refer to the following links to learn more about them:

1. String functions
2. Date-time functions

In the next segment, you will learn about a pattern matching capability available in MySQL known as regular expressions and understand how it can be implemented in MySQL workbench.

Regular Expressions:-

You have already learnt how to perform pattern matching using the **like** operator along with **wildcards**. Now, the 'like' operator and the wildcards may fall short for some advanced use cases. One such example that you can consider is email validation. Given a string, how can you determine whether an email is valid or not? This may not even be possible to achieve using 'like'. In fact, this is a slightly complicated requirement to meet even for regular expressions. It definitely makes the work of an analyst much easier, though.

A regular expression is a pattern matching operation available in MySQL with the help of REGEXP operator. It is case-insensitive and provides powerful and flexible pattern match capabilities. It also supports a number of metacharacters giving more control while performing pattern matching.

Some special characters used in REGEXP operator in SQL:—>

1.) '^' - simply means begins with and if you want to match multiple then use square brackets '[]'.
2.) '.' - simply means any kind of character.
3.) '*' - simply means one or many occurrence of character present just Prior to it.
4.)'$' - simply means the end of the String , so any thing present prior to this symbols will be checked as last entry of the string.

Remember if you want to reverse the search like for e.g. (something that doesn't start with). For that scenario we have to use that particular symbol twice , one outside and one inside the square brackets. For e.g. (^[^OCT] - this simply means that something that doesn't begins with O,C and T .).

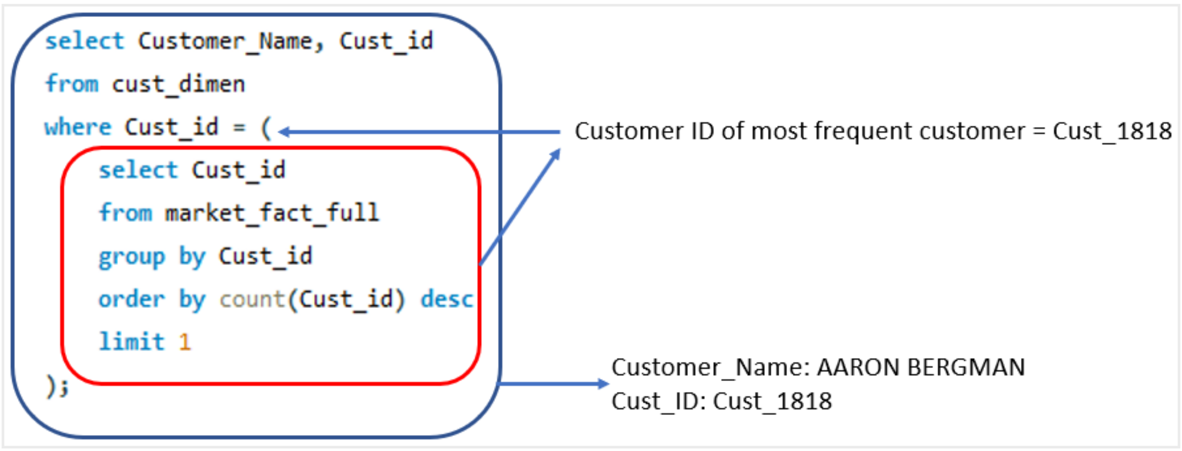
Now you will learn about nested queries. As you may already know, a nested loop is a loop within a loop. Similarly, a nested query is a way of writing a query inside another query. In the next segment, we will see how we can implement nested queries

Additional Resources:
You can refer to this MySQL | Regular expressions (Regexp) link to get an idea about some of the pattern matching that can be achieved using regular expressions.

Nested Queries:-

In this segment, you will learn about nested queries. In simple terms, a **nested query is a query within another query**. The inner query is known as the subquery; it performs an operation, the output of which is used by the main query to perform additional operations. Let's look at some use cases...

Let's revise some of the use cases of nested queries.

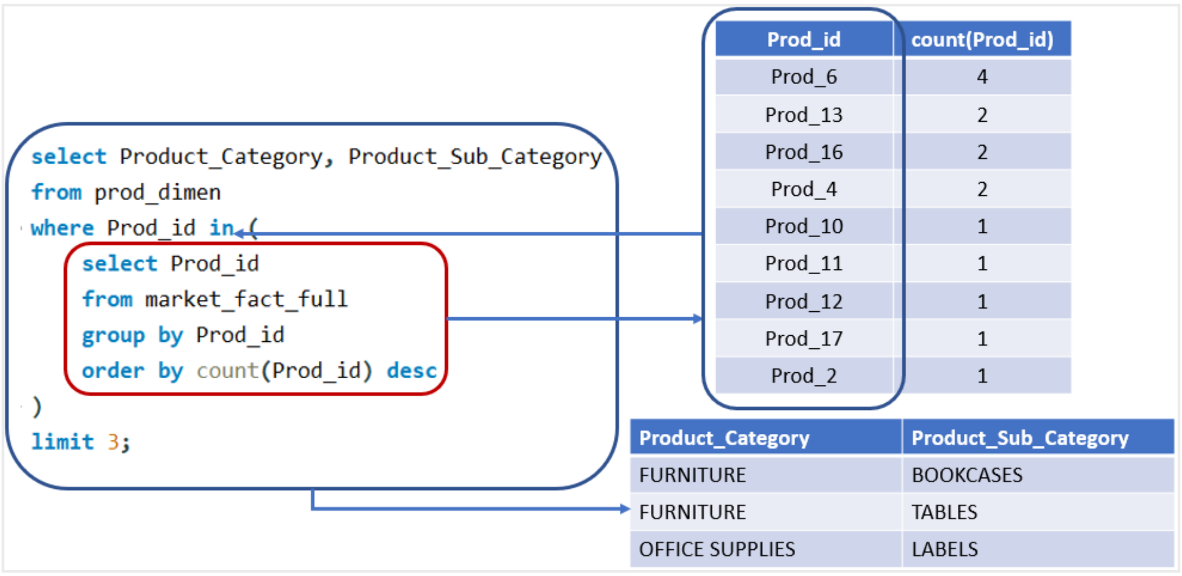


```
select Customer_Name, Cust_id
from cust_dimen
where Cust_id = (
    select Cust_id
    from market_fact_full
    group by Cust_id
    order by count(Cust_id) desc
    limit 1
);
```

Customer ID of most frequent customer = Cust_1818

Customer_Name: AARON BERGMAN
Cust_ID: Cust_1818

In the image above, the inner query groups each customer ID and sorts them by the count of each customer in descending order. Finally, it returns the customer ID of the most frequent customer. This value is used to find the customer in the cust_dimen table whose Cust_id is equal to that value.



```
select Product_Category, Product_Sub_Category
from prod_dimen
where Prod_id in(
    select Prod_id
    from market_fact_full
    group by Prod_id
    order by count(Prod_id) desc
)
limit 3;
```

Prod_id	count(Prod_id)
Prod_6	4
Prod_13	2
Prod_16	2
Prod_4	2
Prod_10	1
Prod_11	1
Prod_12	1
Prod_17	1
Prod_2	1

Product_Category	Product_Sub_Category
FURNITURE	BOOKCASES
FURNITURE	TABLES
OFFICE SUPPLIES	LABELS

In this image, the inner query returns the Prod_id sorted by their count. Note that their corresponding count has only been shown for your understanding and is not returned as part of the inner query. In the outer query, we find the Product_Category and Product_Sub_Category for the corresponding Prod_id values and limit them to 3.

With this, we conclude our segment on 'Nested Queries'. The next segment will be about Views. Take the example that we used in nested queries itself. Suppose the result from the inner query would have to be used on multiple occasions. It would be a cumbersome task to write this code again and again. Instead, would it not be more efficient to store this data in some virtual table, which can be used simply by calling the name of the table? That is what Views are all about.

Views:-

In the previous segment, you learned about nested queries, which help us use the output from a query to apply further restrictions to get the desired output. In this segment, we will discuss the Views. So what are views? A view is a **virtual table** created as a result of operations on a single table or multiple tables. A view can be treated like any other table and can be further updated or deleted.

the concept of views. As he mentioned, when you create a view, there is no separate storage layer for storing this table. One major advantage of creating a view is data security. Suppose you have confidential data and you need to make specific columns available to a user who is denied access to the rest of the data. In such cases, views can be used to extract the specific data, which is then shared with the user while keeping the main table hidden. As a result, the end-user can only see the view.

Example of views:—>

Creation of View from an Existing Table : create view order_info as select Ord_id, Order_Quantity , Profit, Shipping_Cost from market_fact_full;

Querying on View : select Ord_id, Profit from order_info where Profit >1000;

you learnt how to work with Views. With this, we have come to the end of the second session. In the next session, you will learn how to work with joins. Joins help you merge different tables together. You will learn about the different types of joins and some of their use cases.

Summary:-

Let's summarise your learnings from this session.

SQL stands for Structured Query Language. In some applications, it is also referred to as DSL, or Domain Specific Language. SQL is used for managing and manipulating data held in a relational database management system (RDBMS).

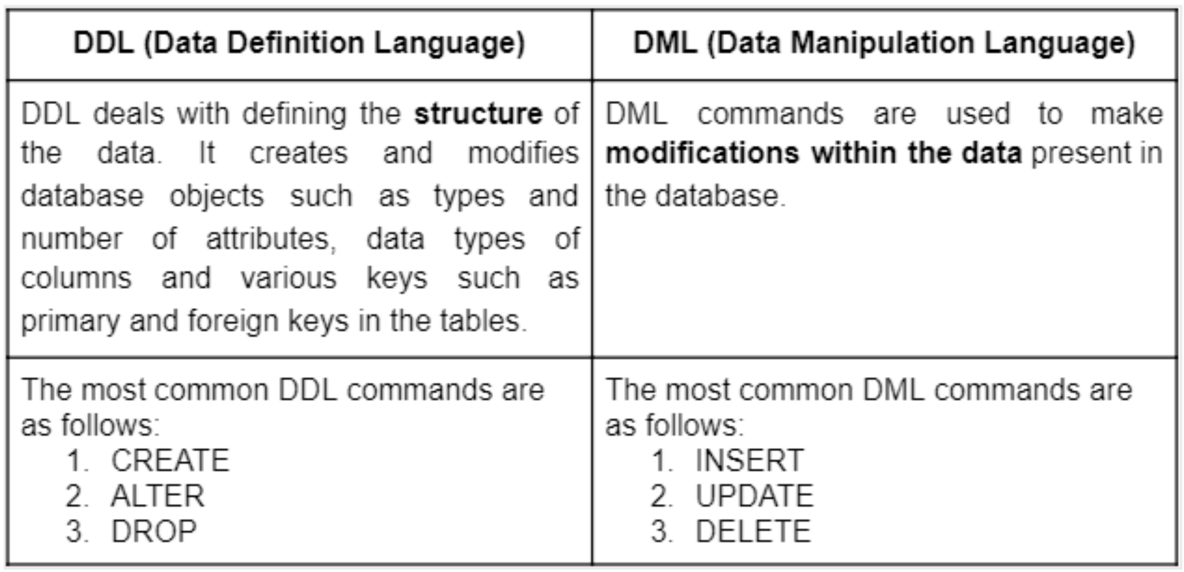
SQL helps you:
- Create data

- Manipulate data
- Delete data
- Share data

Features of SQL:
- SQL is language-agnostic, which makes it a language that can be easily learnt and comprehended.
- It is also supported by all the modern database systems such as Oracle, MySQL, SQL Server, and so on.
- The SQL syntax can be used for Hive, which is used for big data processing

Among the existing RDBMS systems, you learned how to work with MySQL. RDBMS commonly finds application in the following:
- The banking sector
- E-commerce websites
- Social networking sites

SQL commands are mainly divided into two subcategories:
- DDL (Data Definition Language)
- DML (Data Manipulation Language)



DDL (Data Definition Language)	**DML (Data Manipulation Language)**
DDL deals with defining the **structure** of the data. It creates and modifies database objects such as types and number of attributes, data types of columns and various keys such as primary and foreign keys in the tables.	DML commands are used to make **modifications within the data** present in the database.
The most common DDL commands are as follows: 1. CREATE 2. ALTER 3. DROP	The most common DML commands are as follows: 1. INSERT 2. UPDATE 3. DELETE

Following this, we worked on a case study where we learnt how to work with SQL operators and inbuilt functions. You also learnt how to create an ERD on MySQL. Now, let's look at some of the common syntaxes:

- SELECT: Used to read the data
- COUNT: Used to count the data
- WHERE: Used as a conditional statement to filter out data
- OPERATORS: Some examples of operators that can be used along with the WHERE clause are OR, AND, IN, BETWEEN.
- LIKE: Used to search for a specific pattern

- GROUP BY: Groups data that have the same values based on the specified condition
- HAVING: Used as a WHERE clause in conjunction with grouping statements
- ORDER: Used to sort the data

You also learnt about aggregation functions as shown below:

- COUNT(): Counts the total number of records specified by the given condition
- SUM(): Returns the sum of all the non-NULL values
- AVG(): Returns the average of all the non-NULL values
- MIN(): Returns the minimum of the specified values (NULL is not included)
- MAX(): Returns the maximum of the specified values (NULL is not included)

SQL has several String and Date-time inbuilt functions.
Common string functions are LOWER, UPPER, SUBSTRING, SUBSTRING_INDEX, LENGTH.
Common date-time functions are CURTIME, MONTH, YEAR, DAYNAME.

You also learnt about the pattern matching capability of MySQL known as regular expressions.

After this, you learnt about nested queries. A nested query is a query within another query. You also learnt about views. A view is a virtual table created as a result of operations on a single or multiple tables. A view can be treated like any other table and can be further updated or deleted. One major advantage of creating a view is the data security. They can be used to extract the specific data from the database while keeping the main table hidden from the restricted user. As a result, the restricted user is only permitted to work with the view.

In the next session, you will learn about joins and the different types of joins. You will also work with joins in conjunction with views.