Database Design and Introduction to MySQL

Module 1 : Database Design

Introduction:-

So far, you have learnt how to perform basic data analysis using Python and Excel. You have also learnt that it is risky to manage data from different files and that it may lead 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. A software known as **Database Management System (DBMS)** is used widely in the industry to analyse data easily.

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), SETL (Select, Extraction, Transformation, and Loading), constraints and ERD (Entity Relationship Diagrams).

What is a Data Warehouse?:-

Think of yourself as a data analyst working for a company that has the following three departments: Marketing, Sales and Finance. Now, let's assume that each department maintains a separate database.

This could lead to a situation wherein each department has its own version of the facts. For a question such as 'What is the total revenue of the last quarter?', every department might have a different answer. This is because each department draws information from a different database.

This is where a data warehouse can prove to be useful. It can help with creating a single version of the truth and the facts. A data warehouse would thus be the central repository of data of the entire enterprise.

Now, in the upcoming video, you will learn what a data warehouse is and how it is useful for companies in carrying out data analytics. You will also learn about **OLAP**, which stands for **Online Analytical Processing systems**. OLAP is used to extract business-relevant information and perform analysis on the data stored in data warehouses. In the next video, you will understand the data warehouse.

So, a data warehouse is a collection of data. It has the following properties:

- **Subject-oriented:** A data warehouse should contain information

about a few well-defined subjects rather than the enterprise.
- **Integrated:** A data warehouse is an integrated repository of data. It contains information from various systems within an organisation.
- **Non-volatile:** The data values in a database cannot be changed without a valid reason.
- **Time-variant:** A data warehouse contains historical data for analysis.

In the next segment, you will learn about the structure of the data warehouse.

Structure of a Data Warehouse:-

In the previous segment, you learnt about the basic concepts of data warehousing. Now, one of the primary methods of designing a data warehouse is **dimensional modelling**.

The two key elements of dimensional modelling include **facts** and **dimensions**, which are basically the different types of variables that are used to design a data warehouse. They are arranged in a specific manner, known as a **schema diagram**. So, in the following video, you will learn more about facts and dimensions.

So, essentially, facts are the numerical data in a data warehouse 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.

So, you have learnt about the structure of the data warehouse. In the next segment, you will get the idea about the star schema but before that let's solve some topic related MCQs.

Star Schema:-

In the previous segment, 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 warehouse, called a schema diagram.

A schema is an outline of the entire data warehouse. It shows how different data sets are connected and how the different attributes of each data set are used for the data warehouse.

In the upcoming video, you will see an example of how an e-commerce company can design a data warehouse.

So, in the video, you learnt about **star schema**. Although there are other types of schemas available for a data warehouse, the star schema is the most widely used. You can visit the link provided under Additional Reference to learn more about the different types of schemas.

Additional Reference:

Click Data Warehouse Dimensional Modelling (Types of Schemas) to learn more about the different types of schemas.

In the next segment, you will learn about OLTP and OLAP systems.

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, our expert 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 SETL (Select, Extract, Transform and Load).

SETL:-

So far in this module, you have learnt about the different types of databases that are available for solving different problems. Now, what is the next step in the process? As you can see, we now need to get the data into the schema to perform relevant operations on it. You might be wondering how that is done. In the upcoming video, Professor Chandrashekhar R will give you the answer to this question.

So, in the video, you were introduced to the SETL process, as follows:

SETL: Select, Extract, Transform and Load.

- Select: Identification of the data that you want to analyse
- Extract: Connecting to the particular data source and pulling out the data
- Transform: Modifying the extracted data to standardise it
- Load: Pushing the data into the data warehouse

This process includes the typical operations that are involved in selecting the required data, extracting data from multiple sources, operating on the data so that data from multiple sources is compatible, and loading this data into a data warehouse for analytical purposes.

In this way, you have understood the SETL. In the upcoming segments, you will understand the various constraints.

Entity Constraints:-

You have already learnt about the SETL 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 database schema?

Put yourself in the place of a data analyst at Uber. A database 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.

In the next video, you will learn about the relational data model.

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 the relational model in the next video.

So, as you learnt in the video, entity constraints are of the following different types:

- **Unique:** This constraint is used for columns that need unique values. For example, 'employee ids' should be unique in an 'employees' table.
- **Null:** This constraint is used to determine the columns that can have null values. For example, an employee may not need to specify their location, which means the 'location' column can have null values in an 'employees' table.
- **Primary Key:** This constraint is used to determine the column that uniquely identifies a table. For example, 'employee ids' uniquely identify every employee. Two employees may have the same name or the same salary, but not the same employee id.

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 constraint, entity constraints, which pertain to the values in a single table. The second type of constraint 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.

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.

To sum up, 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? Let's find out in the upcoming video.

Note: At [02:33], the professor misspeaks that NOT NULL is a semantic constraint. This is, however, not true. You have already learnt that it is an entity constraint.

So, as you learnt in the video, semantic constraints impose additional restrictions on the values in a column. For example, all the mobile numbers in India start with '+91', followed by 10 digits. Using a semantic constraint for this requirement ensures that we do not get incorrect data for any row. For example, a row with a phone number +91987654321 would not be allowed to enter the database as it has only 9 digits after the country code.

Now, let's have practice questions on the above-learnt topics.

In the next segmrnt, we will look into ERDs.

ERDs:-

Now that you have learnt about the various types of constraints that are used to restrict the values in a database, the precursor to designing a database is drawing a well-defined ERD.

An **Entity-Relationship Diagram**, **or ERD**, 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?

So, as you can see, ERDs are extremely useful to get an overall idea of a database in very less time. Now, in the upcoming video, you will learn more about them from our expert.

So, as you learnt in the video, the constituents of an ERD are as follows:

- **Entity Type/Entity:** It is nothing but a table in the schema. For example, 'orders' and 'payments' are both entity types.
- **Attribute:** It is a column in an entity type. For example, 'orderNumber' is an attribute in the 'orders' entity type.
- **Relationship Types:** They are the lines between the tables. They define the relationships among the tables. These can be of various types based on their cardinalities, i.e., **one-to-one**, **one-to-many**, **many-to-many**, etc.

In the next video, let's understand the self-referential relationship.

So, as you learnt in the video, there is another type of relationship, called **self-referential relationship**. Here, the table refers to itself.

Take a look at the table given below.

Team India

Player Name	Player Role	Captain	Vice-Captain
K. L. Rahul	Batsman	Kohli	Sharma
Rohit Sharma	Batsman	Kohli	Sharma
Virat Kohli	Batsman	Kohli	Sharma
Shreyas Iyer	Batsman	Kohli	Sharma
Rishabh Pant	Batsman	Kohli	Sharma

Ravindra Jadeja	All-rounder	Kohli	Sharma
R. Ashwin	All-rounder	Kohli	Sharma
Hardik Pandya	All-rounder	Kohli	Sharma
Bhuvneshwar Kumar	Bowler	Kohli	Sharma
Mohammed Shami	Bowler	Kohli	Sharma
Jasprit Bumrah	Bowler	Kohli	Sharma

The table given above has a self-referential relationship because Virat Kohli is the captain of every player, but he is a player himself. The same applies to Rohit Sharma, who is the vice-captain of the team.

Comprehension
Suppose you are given a data set of a college. This data set mainly includes the following four tables:

- **Student:** This table contains information about the students, such as student id, name, range of marks scored in the exam, year of graduation and branch id.

Student
student_id
student_name
marks_range
year
branch_id

- **Branch:** This table includes the different branches present in the college, such as Electrical, Mechanical, Civil, Computer Science, Chemical, along with branch ids.

Branch
branch_id
branch_name

- **HoD:** This table contains information about the HoDs, i.e., Head of the

Departments of the different branches. This information includes HoD name, duration of service as HoD, branch id of the branch in which a particular person is/was HoD and their contact information.

HoD
branch_HoD_name
year_of_service_as_HoD
branch_id
HoD_contact

-

- **Marks:** This table contains the mapping of the range of marks with the grades awarded.

Marks
marks_range
grades

You can consider the following entries as examples of each of the table formats above.

Student

student_id	student_name	marks_range	year	branch_id
S-04	Ashish Mehra	50-60	1st	EE
S-05	Chetan Singh	70-90	3rd	ME

Branch

branch_id	branch_name
EE	Electrical Engineering
ME	Mechanical Engineering

HoD

branch_HoD_name	year_of_service_as_HoD	branch_id	HoD_contact
Rakesh Sharma	2008	EE	9876543123
Vinay Rathod	2009	EE	9567456321
Rudra Singh	2010	CS	9876789044

Marks

marks_range	grades
90-100	A
70-90	B

Answer the following questions based on the information in the tables given above.

Star Schema: A Demonstration:-

By now, you must have a fair idea about star schemas, constraints and ERDs. Now, we will go through an example of a star schema and understand how it makes database designing simpler and more intuitive. So, let's watch the upcoming video to learn more about it from our expert.

So, as you learnt in the video, the market star schema consists of five tables: 1 central fact table and 4 dimension tables, which describe the fact table. This structure of one fact table with multiple dimension tables makes data analysis easier because it reduces the number of joins required for querying data. When you learn about joins later in this module, you will better understand the reason why Professor Ramanathan insists that you design a star schema during the database creation phase.

In this way, you have completed the session on database design. let's summarise the session in the next session.

Summary:-

In this session, you learnt the following:

- What a data warehouse is
- The differences between a data warehouse and a transactional database
- OLAP (Online Analytical Processing) systems are Subject-oriented, Integrated, Non-volatile and Time-variant
- A data warehouse provides an integrated view of the entire organisation and the data is organised for carrying out analysis

efficiently
- Facts and dimensions and how to arrange them in order to design 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
- What a star schema is
- The advantages of having a star schema in a database
- The SETL process: Select, Extract, Transform and Load
- The different types of constraints: Entity, Referential and Semantic constraints
- ERDs and their constituents

Module 2 : Database Creation in the MySQL Workbench

Introduction:-

We will begin this session with an example that is fairly commonplace in the lives of many: Hospital visits. Recall the last time you went to the hospital as a patient. You may have observed that the receptionist asks for your details. The basic details may include name, age, gender, address, phone number, etc. Some additional details could include information on vaccinations, medical history, details of family illnesses and so on. Based on the information that you provide, the receptionist may assign you to a specific doctor, ward and nurse for treatment. Now, how are these decisions made in almost real-time? Well, this is how it all happens: All the data that you provide, along with the data pertaining to the hospital, is stored in some database. Operations are performed on the database to return relevant results so that the authorities involved can act accordingly.

But how is all of this exactly done? What are the tools and languages that are involved in the process? In this session, Professor Ramanathan will walk you through the data retrieval language that is used in RDBMS (Relational Database Management System): **SQL (Structured Query Language)**.

There are many RDBMS (Relational Database Management System) available, for instance, MySQL, Oracle, MS Access, SQL Server and Postgres. You will be introduced to MySQL, and the platform that you will use in order to retrieve the required data is **MySQL Workbench**.

In this session

You will be introduced to certain basic SQL commands. SQL is one of the strongest tools for data manipulation, and it is used heavily across all industries. By the end of this session, you should be able to use MySQL databases and write basic SQL queries to retrieve relevant information from typical databases.

Introduction to DDL and DML Statements:-

In this segment, Professor Ramanathan will demonstrate the MySQL Workbench commands and tools that are used to create the schema and tables.

You should be able to install the MySQL Workbench on your machines. Please follow the installation instructions provided in the installation guides attached below.

Note that 32-bit installation is available for Windows. Irrespective of your system architecture (whether it is 32/64 bit), please install the 32-bit version.

So, after installing SQL on your machine and learning about the basic components and terms associated with SQL, you will next learn more about the entire set of commands that are available in SQL. So, let's watch the upcoming video and learn about it from our expert.

So, the commands available in SQL can be broadly categorised as follows:

- **Data Definition Language (DDL)**
- **Data Manipulation Language (DML)**

DDL, as the name suggests, is used to create a new schema as well as modify an existing schema. The typical commands available in DDL include **CREATE, ALTER** and **DROP**. Now, as a data analyst, the majority of your work will focus on insight generation, and you will be working with DML commands, specifically, the **SELECT** command. In the next segment, Professor Ramanathan will discuss this in detail.

DDL Statements: A Demonstration:-

In the first video of this segment, you will learn how to create and use a database (also known as a schema) in the MySQL Workbench. You will need to specify the database that you need to use. This is because even after creating a database, the MySQL Workbench does not select your newly created database automatically. The syntax for these operations is given below:

- create database <database_name>
- use <database_name>

So, you have now created a database. However, it holds no meaning without its constituent tables. Now, in the next video, you will learn about the **create table** command and create a table containing three columns along with their respective data types.

So, you now have a table ready at your disposal. Recall that every table should

have a primary key to identify it uniquely. To fulfil this requirement, in the next video, you will learn how to use the **alter table** command for adding a primary key constraint to the table that you created in the previous video.

In this way, you have learnt about the DDL statements. In the next segment, you will have hands-on of DML statement.

DML Statements: A Demonstration:-

In the previous segment, you learnt how to create a table and add the necessary columns with the required data types. However, to start analysing data, you need the data in the first place. This is where Data Manipulation Language statements come into the picture. In the first video of this segment, you will learn how to insert values in a table using the **insert into** and **values** keywords in a query.

So, now that you have inserted values into the table, you may feel the need to update or delete them (in case of incorrect entries). You can do this using the **update - set - where** and the **delete - from - where** commands. So, let us go ahead and learn about these in the next video.

Now, do you remember what you learnt about primary and foreign keys? A value that is a primary key in a table cannot be deleted if another table has that same value as a foreign key. Let's watch this in action in the next video.

In this way, you have learnt about the DML statements. In the next segment, we will look into modifying the columns.

Modifying Columns:-

Now that you have a complete table with values and constraints, in this segment, you will learn how to:

- Add another column using the **alter table** and **add** commands,
- Update a value in a column using the **update** and **set** commands, and
- Delete a column using the **alter table** and **drop column** commands.

So, in the upcoming video, you will learn about these commands from our expert.

It is important that you know how to modify columns so that you can enter the right data immediately if you notice that some value has been incorrectly inserted into a particular column. Doing this will ensure that you get the right insights and make the right business decisions after analysing the correct data.

You have got the understanding of modifying the columns. In the next segment, you will be provided with some of the practice exercises.

Summary:-

In this session, you learnt how to set up a schema and make it ripe for analysis using various DDL (Data Definition Language) and DML (Data Manipulation Language) statements in MySQL.

The DDL statements are:

- CREATE
- ALTER
- DROP

The DML statements are:

- INSERT
- UPDATE
- DELETE
- SELECT

While Data Definition Language statements are used to alter the structure of a database, Data Manipulation Language statements are used to change the data itself that is present inside the database.

So, now that you are ready to use MySQL for its primary intended purpose, that is, data analysis, it's time to write useful queries to derive actionable insights from the ocean of data that lies in front of you.

Module 3 : Querying in MYSQL

Introduction:-

In the previous sessions, you learnt what databases are, what they are composed of, and how to create a database along with its tables, constraints and values. In this session, we will dive into the core of SQL and you will learn all about the statements that are used for querying data. You will also develop the thought process that is required to extract the data needed for analysis and to drive business decisions.

In this session

You will learn how to write queries using the **select**, **from**, **where**, **group by**, **having**, **order by** and **limit** statements. A query may generally contain all or some of these keywords. You will also get an understanding of the need for the various types of operators and functions, which make it easy to address seemingly complex data requirements.

Eventually, Professor Ramanathan will talk about subqueries, CTEs (Common Table Expressions) and views, which help with solving even more complex

problem statements.

SQL Statements and Operators:-

Consider a data set of the Indian census containing at least a billion rows, with data on every individual's height, weight, monthly income, and hundreds of more such attributes. How do you even start with the analysis of such an extensive data set to capture any trends in the data? The SQL clauses that you will learn in this segment and throughout this session will help you with this.

So, in this segment, you will learn about the basic constructs of the SELECT query. You will specifically learn about the following:

- 'Select' Clause
- 'From' Clause
- 'Where' Clause
- Operators: Arithmetic, Comparison and Logical

Now, it is important that you go through the next videos carefully, as they will form the foundation for the entire SQL course.

Before proceeding to the operations, download the following SQL script and set up the schema on which all the operations are performed.
Also, please download the attached script file, which contains the questions that are solved in the upcoming videos.

So, in the video, you learnt about a few basic queries using the **select** and **from** statements. After watching the next couple of videos, you will be able to filter data based on certain given criteria and group it into categories, if required. You will also learn how to use operators in queries. In the next video, you will have hands-on using the basic SQL commands.

you will see more SQL operators' applications.

Operators can be used for various functions: pattern matching, returning a subset of the data based on its properties or comparing different values to get a sense of the distribution of the data. Pattern matching is an important concept in text-based processing. As you learnt, in SQL, certain characters are reserved as wildcards, which can match any number of preceding or trailing characters.

Additional Resources:

1. SQL Operators is an excellent resource where you can have a quick glance at the various operators that are supported in SQL.

Now let's try to solve some MCQs as well as SQL querying questions below:

Aggregate Functions:-

As you go about examining and analysing data of varying magnitudes, you will quickly realise the need for grouping similar types of values together and looking at them as one group. For example, consider a table that has data, which consists of the marks scored by students in their 12th board exams. While you would want to know how the students performed in all the subjects put together, it is equally important to see how they performed in each subject. You can derive even further insights if you group these students by state. Hence, it is imperative that you learn the usage of aggregate functions in your queries. So, let's go ahead and learn about it from our expert in the upcoming video.

So, in this segment, you learnt how to use the **group by** clause. In brief, we use 'group by' when we need to find the aggregate values of a column C1 'grouped by' a certain column C2.

count() is just one of many aggregate functions that are available in the MySQL Workbench. Feel free to explore other functions, such as **min()**, **max()** and **avg()**. Try using them in queries and examine the results that you obtain. You will realise that aggregate functions play a crucial role in determining outliers in the data and analysing any existing trends in it.

So, as Professor Ramanathan mentioned in the video, please keep this in mind whenever you write any query because not all versions of the MySQL Workbench in all operating systems allow non-aggregate columns in the 'select' clause whenever you use a 'group by' statement.

Now based on the above learnings, let's try to solve the below MCQs.

You can refer to the link given below to learn more about some other types of aggregate functions that are used in MySQL:

Now, let's try to solve the following coding question.

Ordering:-

Quite often, you would want to display the retrieved records in a particular order, for example, in increasing order of income, joining date or alphabetical order. This is commonly useful when you are making a report or presenting the data to someone else. Let's try to understand this in the next video.

The Having Clause:-

You have already learnt how to filter individual values based on a given condition. But how do you do this for grouped values? Suppose your manager asks you to count all the employees whose salaries are more than the average

salary in their respective departments. Now, intuitively, you know that two aggregate functions would be used here: **count()** and **avg().** You decide to apply the 'where' condition on the average salary of the department, but to your surprise, the query fails. This is exactly what the '**having**' clause is for. So, in the upcoming video, you will learn how the 'having' clause is used in a query.

To summarise, you are now aware of all the SQL clauses: 'select', 'from', 'where', 'group by', 'order by' and 'having'.

The 'having' clause is typically used when you have to apply a filter condition on an 'aggregated value'. This is because the 'where' clause is applied before aggregation takes place and, thus, it is not useful when you want to apply a filter on an aggregated value.

In other words, the 'having' clause is equivalent to a 'where' clause after the 'group by' clause has been executed but before the 'select' clause is executed. You can read this StackOverflow answer to understand the 'having' clause better.

It is important to not get confused between the 'having' and 'where' clauses. For example, if you want to display the list of all the employees whose salary >= 30,000, then you can use the 'where' clause, since there is no aggregation taking place in this query. But if you want to display the list of all the employees whose salary <= the average salary, where avg() is the aggregation function, then you will have to use the 'having' clause.

In the next segment, you will learn the strings and the date-time functions.

String and Date–Time Functions:-

In the video below, you will learn all about string functions. They are used for manipulating string data and make it more understandable for analysis. For example, given two strings 'robertdowneyjr' and 'Robert Downey Jr.', which one is more readable? Of course, the latter one.

Similar to string functions, SQL also has functions to manipulate the date and time values in a database. These are quite useful for analysing time-series data. For example, consider a database pertaining to the students of the university from which you graduated. Now, suppose you have data on the total number of students who used the library every day. You can use date functions to determine whether the library was busier on any particular day of the week as compared with the other days, or maybe a particular month (for example, exam time). In the next video, Professor Ramanathan will take you through queries using such functions.

As you must have realised, functions are extremely important in SQL. They help us extract appropriate information from a single value or a group of values, and

even present the information in a better format. Data in a more readable format can make the work of an analyst much easier in the long run.

Now, let's try to solve the following MCQs and coding questions.

In this way, you have completed the segment on String, date-time functions. In the next segment, you will learn about Regular Expressions.

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.

Additional Resources:
1. 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:-

By now, you know that a database is a collection of multiple related tables. Now, while generating insights from data, you may need to refer to these multiple tables in a query. There are two ways to deal with such types of queries:

1. **Joins**
2. **Nested queries/Subqueries**

In the upcoming videos, you will learn about nested queries/subqueries. You will learn about joins in the next session.

Subqueries are a generic form of writing queries wherein you do not need to specify some value explicitly (either minimum or maximum, or some other value). If any update is made to a table, then the subquery would still return the required output as it is independent of any particular value in the table. Now, let's move on to the next and learn more about nested queries.

To summarise, in this segment, you learnt about nested queries, which are typically used when you have to select columns from one table based on the filter conditions from another table. In such cases, you write a subquery inside the 'where' clause, instead of a specific value.

CTEs:-

Imagine you are working as an analyst at a large bank (if you aren't already, that is). Let's say you are required to find out your top 10 clients out of thousands. Of these clients, your company wants to give some cash handouts to the one that has the least turnover in the current year. You can definitely use a simple query to achieve this [by using the min() function and the 'where' clause], but what if you want to derive multiple insights from the data of your top 10 clients? In this case, it is better to use a **Common Table Expression (CTE)**, so that you can get the data of only those clients and perform all of your analysis on a subset of the entire data. In the upcoming video, you will learn more about CTEs from our expert

A CTE is used to create a temporary table, which is smaller than the existing table. This smaller table cannot be individually queried, that returns an error. The CTE has to be used as part of the main query.

In this way, you have learnt about the CTEs, let's try to understand views in the next segment.

Views:-

In the previous segment, you learnt that CTEs are temporary tables that you can use in order to query data from a part of a huge data set. Now, what if you want to use the same subset of data for multiple queries? In such a case, instead of writing a CTE over and over again, you can store the required data on which you want to perform your analysis. This is what views are used for. You will understand the importance of using views better after watching the upcoming video.

Note that it is not necessary that views would always be preferred to CTEs. When you know for sure that you need to subset data from a table only once, you should use a CTE to avoid extra memory and space usage.

In the next segment, we will summarise our learnings about querying in MySQL.

Summary:-

In this session, you understood the crux of SQL queries. These are the basic concepts of SQL, which will be tested rigorously in your interviews. Hence, we urge you to practise all you can and develop a deep understanding of the various SQL statements.

Now, you learnt that a complete query (in a sense) has the following basic outline:

select (attributes)
from (table)
where (filter_condition)
group by (attributes_to_be_grouped_upon)
having (filter_condition_on_grouped_values)
order by (values)
limit (no_of_values_to_display);

You also learnt about the following important concepts related to basic SQL querying:

- Relational, arithmetic and logical operators
- Aggregate functions
- Regular expressions
- Nested queries
- Common Table Expressions
- Views
- Advantages of views over CTEs

In the next session, you will finally learn about joins and set operations.

Module 4 : Joins and Sets Operations

Introduction:-

Previously, you learnt about nested queries, which are used to retrieve data from multiple tables. However, as you must have noticed, a nested query refers to only one table at a time. What if you want to refer to multiple tables in a single query? In such a case, you can use joins. Joins are a handy tool for outputting data from multiple tables in a single table.

You will also understand the set theory and some types of set operations. After this session, you will be able to perform set operations on your data using some SQL keywords. For example, you will be able to get the output of two separate queries in a single query using the 'union' operator.

In this session

You will learn the basics of set theory. The basic set operations are as follows:

- **Union**
- **Intersection**
- **Difference**

You will also learn about various types of **joins**, such as follows:

- **Inner join**
- **Left join**
- **Right join**

Set Theory:-

There are many set operations in practice, but in this segment, you will learn about **union**, **intersection** and **difference**. A set is a collection of distinct values. It is denoted by comma-separated values enclosed by curly braces.

Consider two sets A and B containing even numbers and prime numbers, respectively. For ease of understanding, let's consider only values less than 10. Therefore, the sets will be A = { 2, 4, 6, 8} and B = { 2, 3, 5, 7 }.

The union of A and B (denoted by **A** ∪ **B**) will contain all the values that are present in either A or B. Therefore, A ∪ B = { 2, 3, 4, 5, 6, 7, 8 }.

The intersection of A and B (also denoted by **A** ∩ **B**) will contain all the values that are present in both A and B. Therefore, A ∩ B = { 2 }.

The set difference of A and B (also denoted by **A - B**) will contain all the values that are present in A but not in B. Therefore, A - B = { 4, 6, 8 }.

Types of Joins:-

Consider a database that contains data about a music festival that you are organising. Now, while designing the database, you may have tables pertaining to the following:

- Data about the artists who would be performing at the festival
- Details of the stages on which the artists would perform
- Timings for each performance on each stage
- Details of organisers and other details of security guards, volunteers, photographers, etc.
- Details of attendees, including name, age, address, ticket category, etc.

As you can see, the design of such a database can become quite complicated. How would you analyse the data of the artists and attendees together? This information may be required to calculate the feasibility of the event in terms of cost. This is where joins can be used to combine multiple tables and make the analysis of such data easier for you.

Let's understand the joins in our market example in the next two videos.

As you learnt in the two videos given above, there are mainly two types of joins: inner joins and outer joins. While an inner join searches tables for matching or overlapping data, an outer join also returns rows for which there is no match between the tables.

Types of Joins: A Demonstration:-

Now that you have a theoretical understanding of joins in MySQL, next, Professor Ramanathan will walk you through various examples where you would need to use inner joins. The use case may be as simple as just returning some data from two tables. It may even be as advanced as analysing multiple tables together in a single query and identifying any trends that emerge.

The general syntax of a query using an inner join statement is as follows:

select <column_1>, <column_2>
from table_1 a
inner join table_2 b
on a.<common_column> = b.<common_column>;

Earlier in this session, you learnt about inner joins and writing join queries. You must have noticed that only two tables were referred to in the join. But what if you have a complicated problem statement that needs referencing three, four, five or more tables? In such situations, you can write join queries using **multi-joins**.

In the next video, Professor Ramanathan will demonstrate this critical concept, which you will be using frequently as a data analyst.

In brief, you can use multi-joins to join multiple tables using common attributes between pairs of tables. This is possible because the result of a join is also a table, which you can join further to another table (with a common attribute).

ERDs can be useful for understanding the links between tables, which, in turn, can be quite helpful in writing multi-way join queries.

Outer Joins: A Demonstration:-

Now that you know when to use an outer join in a query, Professor Ramanathan will take you through some example queries using left and right joins. Using a left join will return all values from the left table and matching values from the right table. If there is no match for any value, the right table will return null for that value.

To summarise, an outer join is used when you want to display the rows in one table even if they do not have a corresponding entry in the other table. Also, an outer join is of two types: left outer join and right outer join. It does not really matter which table you treat as left or right, i.e., you can choose the table that you are more comfortable with.

Views with Joins:-

Imagine you have two tables Menu and Ingredients. The Menu table contains data about different food items and their prices, while the Ingredients table contains data about the ingredients as well as their quantities used in each food item. It turns out that you need to analyse food products that cost above? 200 repeatedly to understand if there is any way to drive down their costs. Is there a way to avoid writing join statements every single time for each query? Well, it turns out there is. Watch the video given below and find out how you can do this.

Views with joins are especially useful if you need to store data from multiple columns in a single place for ready reference. You can even use subqueries and CTEs in views. Feel free to explore such use cases and try writing queries to get the exact results for further analysis.

By now, you have mastered the most important aspect of writing SQL queries, joins. Do keep practising different queries involving multiple joins, as joins form a majority of the questions that are generally asked in interviews.

In the next segment, you will learn about the different sets operation with SQL.

Set Operations with SQL:-

Can you recall the types of set operations that you came across at the beginning of this session? They were union, intersection and difference. In MySQL, these operations are achieved using the keywords 'union' and 'union all'. Unfortunately, MySQL Workbench does not support 'intersect' and 'minus' keywords; these operations are performed with a combination of other SQL clauses that you are familiar with by now. You may find the two links provided below helpful for reading more about these two operations.

From the next video, you will get an understanding of the difference between using the 'union' and 'union all' operators and learn how they are implemented in queries.

As Professor Ramanathan explained in the video given above, always make sure that the tables or the results of your queries are union-compatible before you perform a union operation on them. Two tables are union-compatible if:

1. They have the same number of attributes, and
2. The attribute types are compatible, i.e., the corresponding attributes have the same data type.

Additional resources

1. MySQL INTERSECT is a useful link to help you understand how you can emulate intersect operations in MySQL.

2. Similarly, you can implement minus operations in MySQL. You can go to this MySQL MINUS to know more about it.

Summary:-

You learnt the following topics in this session:

- Some basic set operations such as union, intersection and difference
- Types of joins: inner joins, left joins and right joins
- Writing simple and complex queries involving multiple joins and views
- Using 'union' and 'union all' in queries

Armed with all this knowledge and experience of writing numerous queries, you are now primed to dive into some of the more advanced features in MySQL. In the next module, you will learn about many of them in great detail.

The SQL codes used in this segment can be found in the file below.

Module 5 : SQL Practical Case Study

SQL Practice Case Study:-

You went through the following sessions in this module:

1. Database Design
2. Database Creation in MySQL Workbench
3. Querying in MySQL
4. Joins and Set Operations

You now have a fair understanding of databases, OLAP vs OLTP systems, basic querying commands in the database using MySQL. You also learnt about some advanced SQL concepts such as joins and set operations.

In this session, we have created an end-to-end case study for you to practise and get a more practical and industrial perspective of RDBMS and SQL querying.

Please refer to the following PDF document, where you will find a problem statement along with the questions.

Please refer to the below spreadsheet for the attribute description regarding the tables that have been used in the above case study

The data set and the required tables have been provided in the well-commented MySQL Workbench given below. You need to run this Workbench to create the database, tables and insert the data into the tables before answering the questions. You are expected to follow each of the instructions that are mentioned in this Workbench.

This is going to be a great hands-on experience on basic SQL querying.

Here, you can find the solution document of the case study. **You need to run each step either by yourself or using the answer feedback that is given in the solution document to move further in the case study; otherwise, you will not be able to proceed further in the case study or answer some of the questions if you do not run the feedback command before moving to the next question.**

Please note the correction in the solution for Q.21 "**where country = 'France'** ", instead of " **where country = 'USA'** "

Note: This case study is **NOT graded;** our main motive to add this case study is to give you a better hands-on experience in SQL programming language.