```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;

SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

EASY WAY TO LEARN MYSQL

BY -
GEOFFREY JORDAN

CONTENT

Chapter 1: Introduction to MySQL

Chapter 2: MySQL syntax

2.1 Data Types in MySQL

- 2.1.1 Numeric data type
- 2.1.2 String data type
- 2.1.3 Date and Time data type

2.2 Operators in MySQL

- 2.2.1 Arithmetic operators
- 2.2.2 Logical operators
- 2.2.3 Comparison operators
- 2.2.4 Bitwise operators
- 2.2.5 Set operators

Chapter 4: Data Manipulation Language (DML)

4.1 Introduction to MySQL DML

4.2 DML command execution

Chapter 7: Data Query Language (DQL)

7.1 Introduction to MySQL DQL

7.2 Current Date and Time using MySQL

Chapter 9: Functions and Logical operators in MySQL

9.1 Functions in MySQL

9.2 Logical operators in MySQL

PREFACE

SQL is one of the most powerful programming language which handles the tasks of RDBMS (Relational Data base Management System). It uses a compact-English-statements to create, define, insert, update, and control the relational database.

SQL is used to store and process information in a relational database whereas MySQL is used in open-source relational database. In this eBook we are going discuss about both SQL and MySQL.

MySQL is best suitable for client-Server Environment. It provides a flexible transaction management. It serves as the primary relational database for many applications, websites as well as commercial products.

As an eBook, this would be a good reference on how to use MySQL to extract, organize, manage, and manipulate data stored in relational databases. We have discussed in details with several examples about DDL, DQL, DML, DCL and TCL.

This eBook covers a good range of Database tasks in a way that is more accessible, painless and effective. You will not be bored. If you find most programming books to be too dry, this could be an excellent book for you to get started in MySQL.

OVERVIEW OF MYSQL

SQL (Structured query language) is initially created by IBM and first commercially implemented by Oracle corporation in 1979 but there were no official standards so later 1986 it was improved and jointly published by ANSI and ISO. In 1992, SQL'92' was published. It is organised in much better way, Data is contained in table, tables are grouped into schemas and schemas are grouped into catalogues and catalogues are grouped into clusters. The first version of MySQL was developed by Monty Widenius, David Axmark and Allan. It was appeared in 1995. Many versions of the MySQL were released by 2000. These versions were compatible with almost all the major platforms.

MySQL is a RDBMS (Relational Database management system) which is based on structured query language (SQL). A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values.

We use commands to create, define, Insert, extract, update and control the database. Following are the types of SQL commands:

- **DDL (Data Definition Language)**
- **DQL (Data Query Language)**
- **DML (Data Manipulation Language)**
- **DCL (Data Control Language)**
- **TCL (Transaction Control Language)**

We will discuss about all these in upcoming chapters.

Following are the Top 10 popular applications using MySQL

1. **online gaming,**
2. **digital marketing,**

3. **retail point-of-sale systems,**
4. **Internet of Things monitoring systems.**
5. **SaaS applications**
6. **Facebook,**
7. **Flickr,**
8. **Web development**
9. **Twitter**
10. **YouTube.**

CHAPTER 1: INTRODUCTION TO MYSQL

1.1 Introduction

MySQL is a software which employs SQL (Structured query language). It is very fast, reliable, scalable, and easy to use for both small and large applications. The name MySQL suggested by the co-founder Widenius's daughter.

What is Database?

Data is a raw fact and figure that is used everywhere in our real world. A group of data makes a specific information or database. In our day-to-day life if we visit a mall, we see that a shopkeeper is handling a product database using a scanner. Another example, let us consider a bank, they are handling customer database. Third example is in schools they are handling student's database. Henceforth the database can be handled either manually in a register or through computer software such as MySQL.

What is relational database?

It is a collection of organized data stored in one or more tables. We can easily see and understand how different data structures relate to each other.

Difference between MySQL and SQL

We can create, read, update, and delete the database using MySQL. But SQL (structured query language) is a Compact-English-Statement used to access, manipulate, and coordinate the database. In this eBook we are going to discuss about interactive environment that is how to work on entered database.

Types of Command used in MySQL

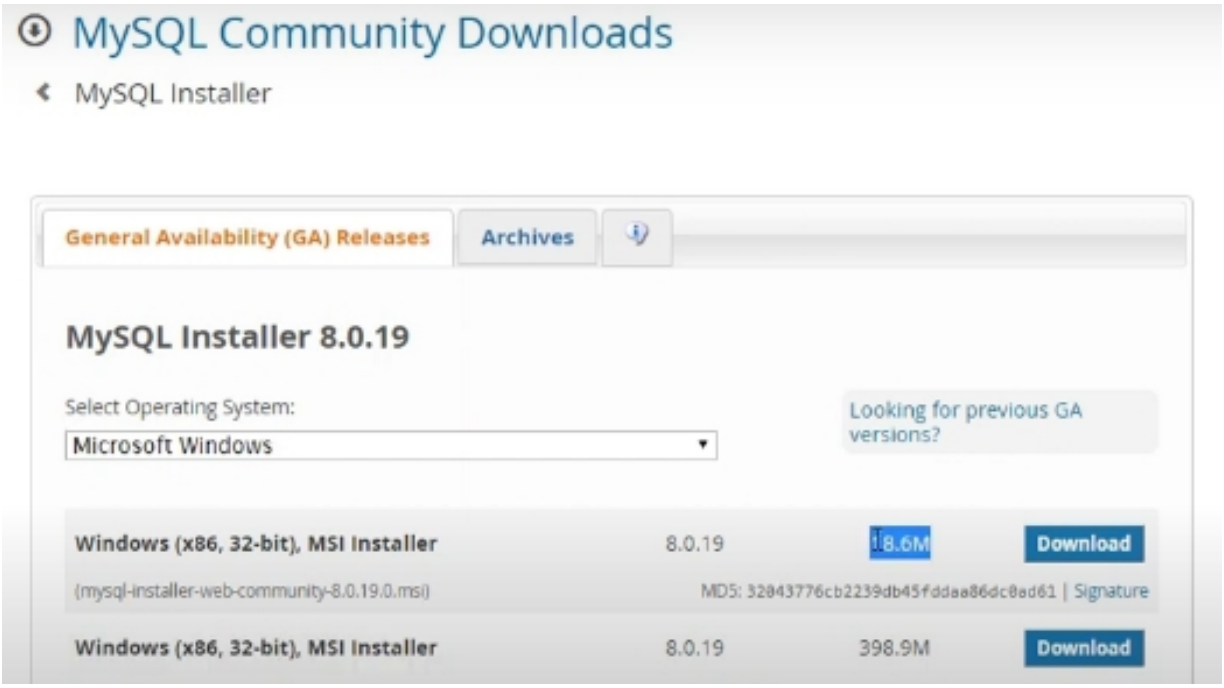
Following are the types of commands used in MySQL

- DDL (Data definition language)- These commands are used to create, drop, alter and describe the Table.
- DML (Data manipulation language) -These commands are used to manipulate data such sum, average and many more
- DCL (Data control language) – It provides security to the database such as Grant and Revoke.
- DQL (Data query language) – It allows user to pass query to the relational database.
- TCL (Transaction Control Language) – It allows user to save and restore databases.

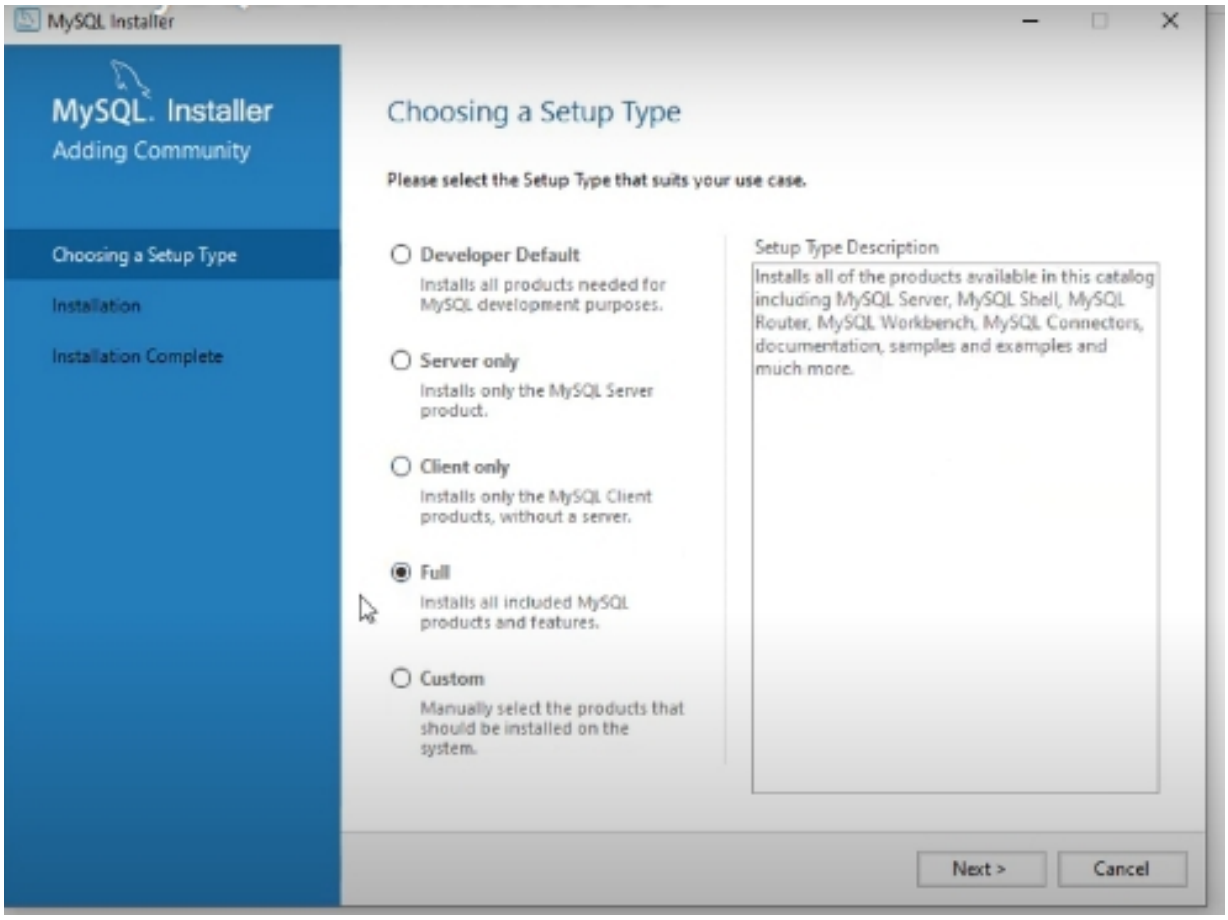
Note: We are about to discuss these commands in upcoming chapters.

1.2 Installation of MySQL

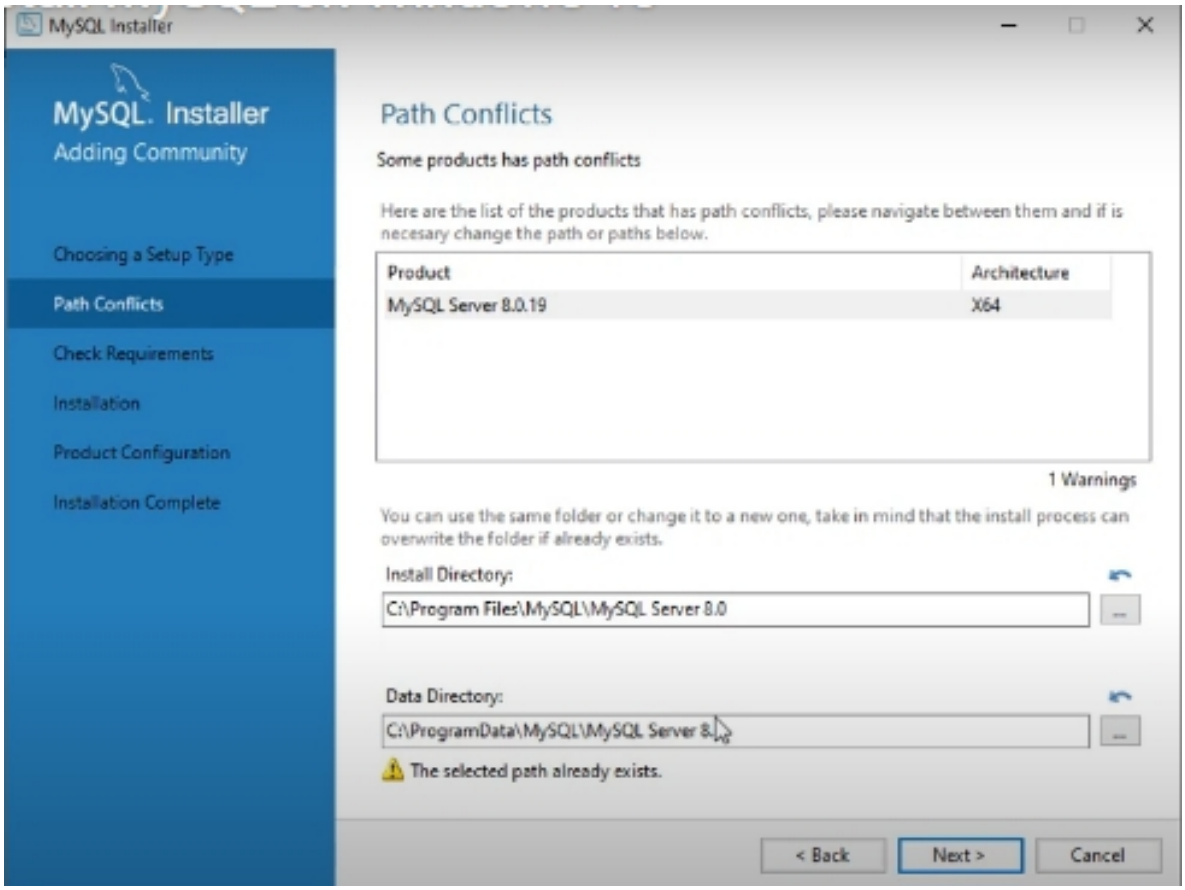
In this section you will learn the installation of MySQL work bench. Let's get started by downloading MySQL from dev.mysql.com. Select the operating system and download MySQL installer 8.0.19. Save it in appropriate location for e.g. Desktop.



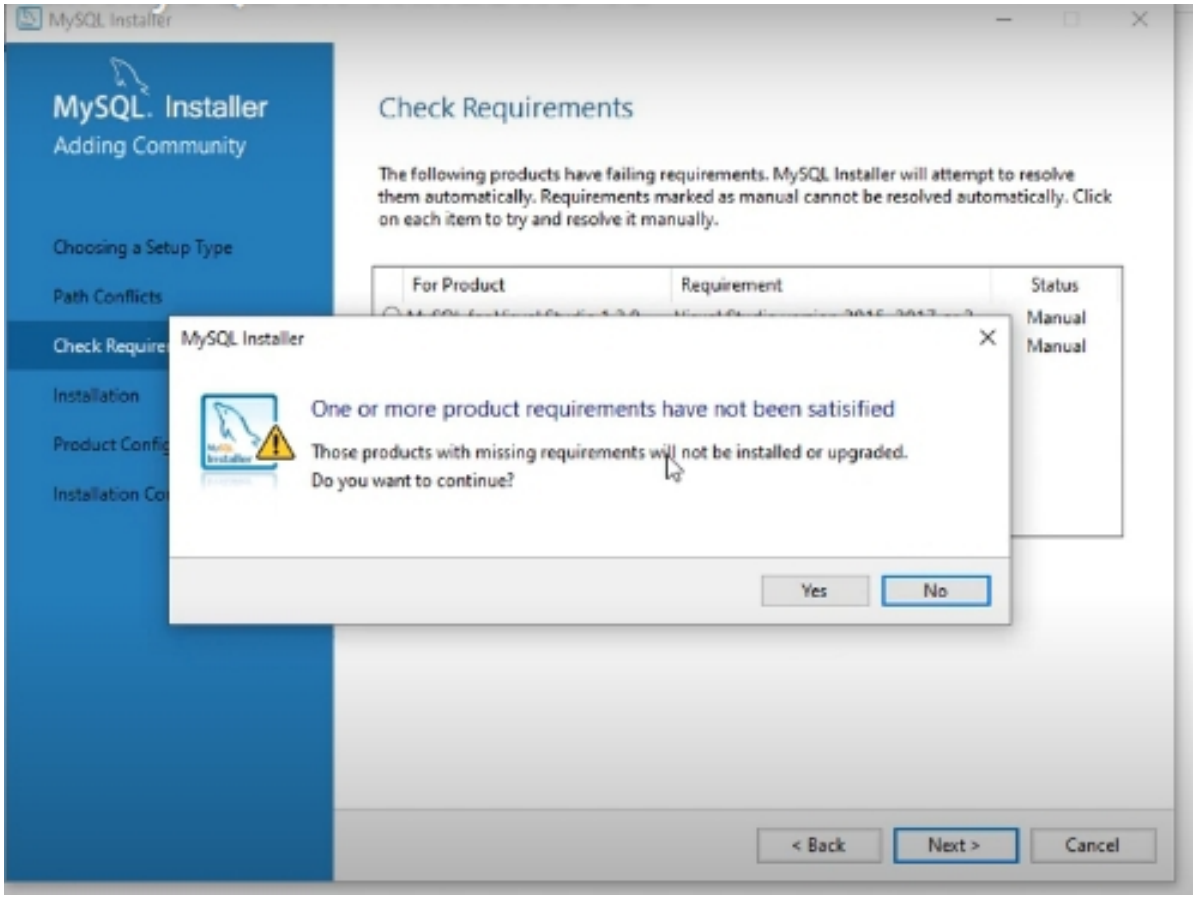
Click on saved file, now choose a setup type, I am choosing according to my requirement – Full than click on next button



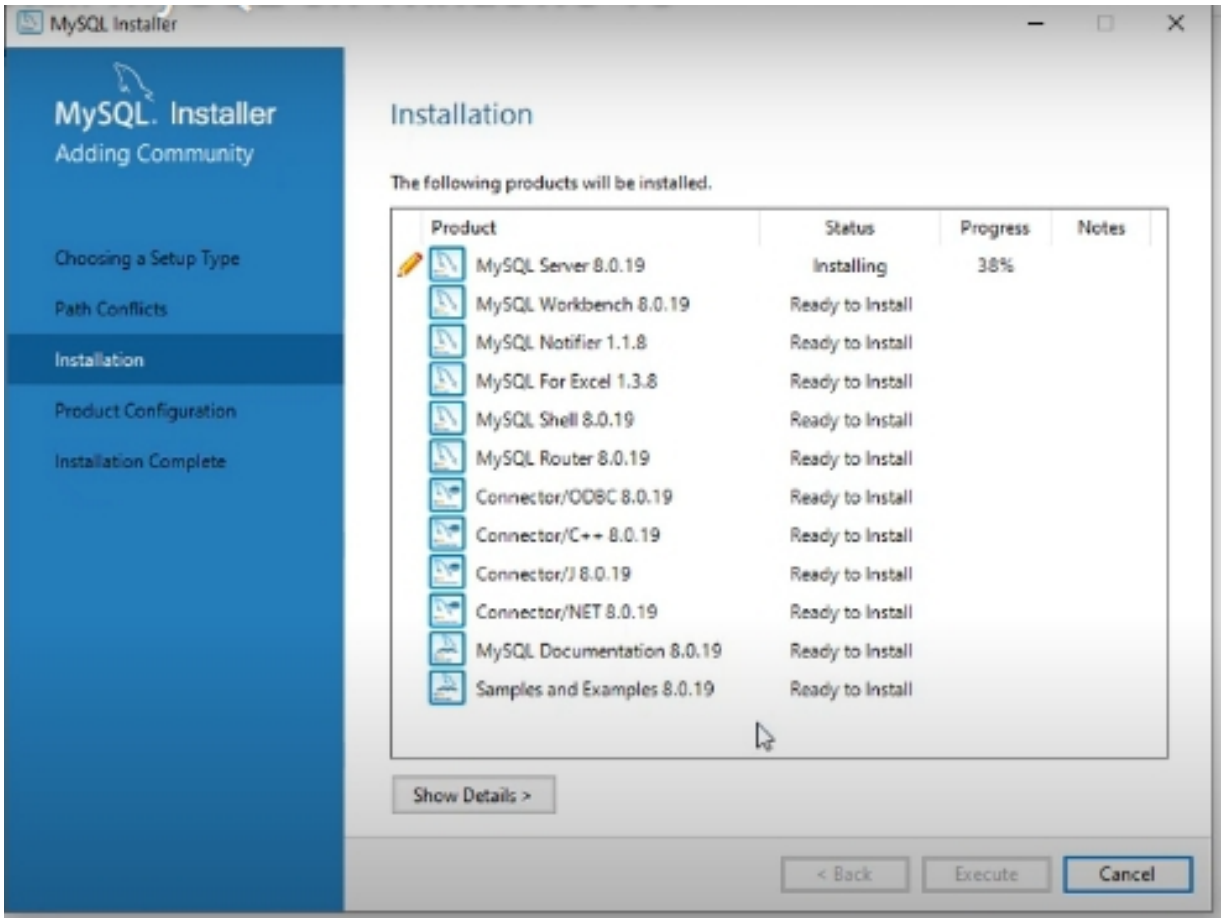
Select the location for MySQL and data. Click on next button, check requirement screen appears you can select the requirement or just click next.



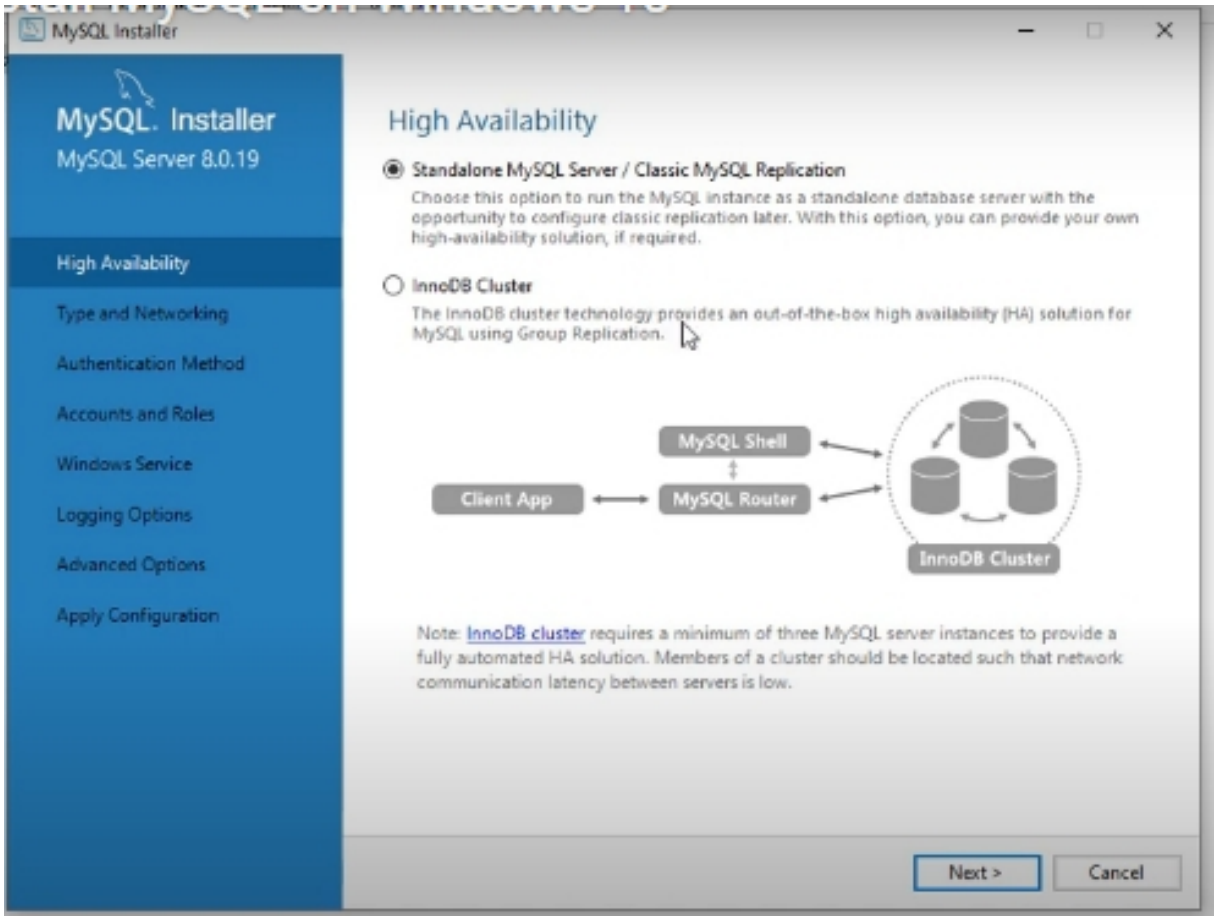
If we do not select the requirements than screen shown below will appear, just click on “Yes” than next



Next features of MySQL screen will appear, click on Execute and wait for few minutes

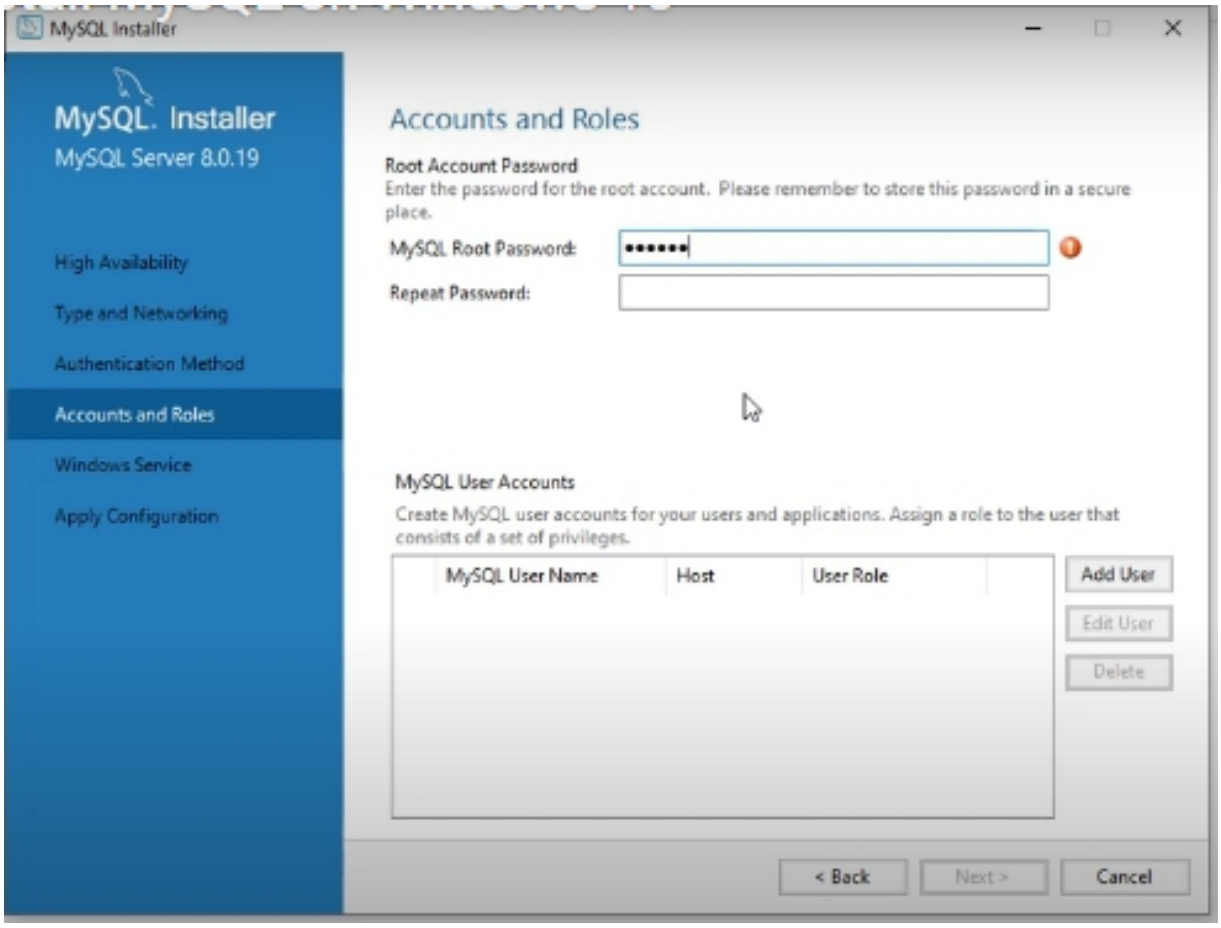


All products installed successfully on our PC, now click on next button twice, select standalone MySQL server than click on next button.

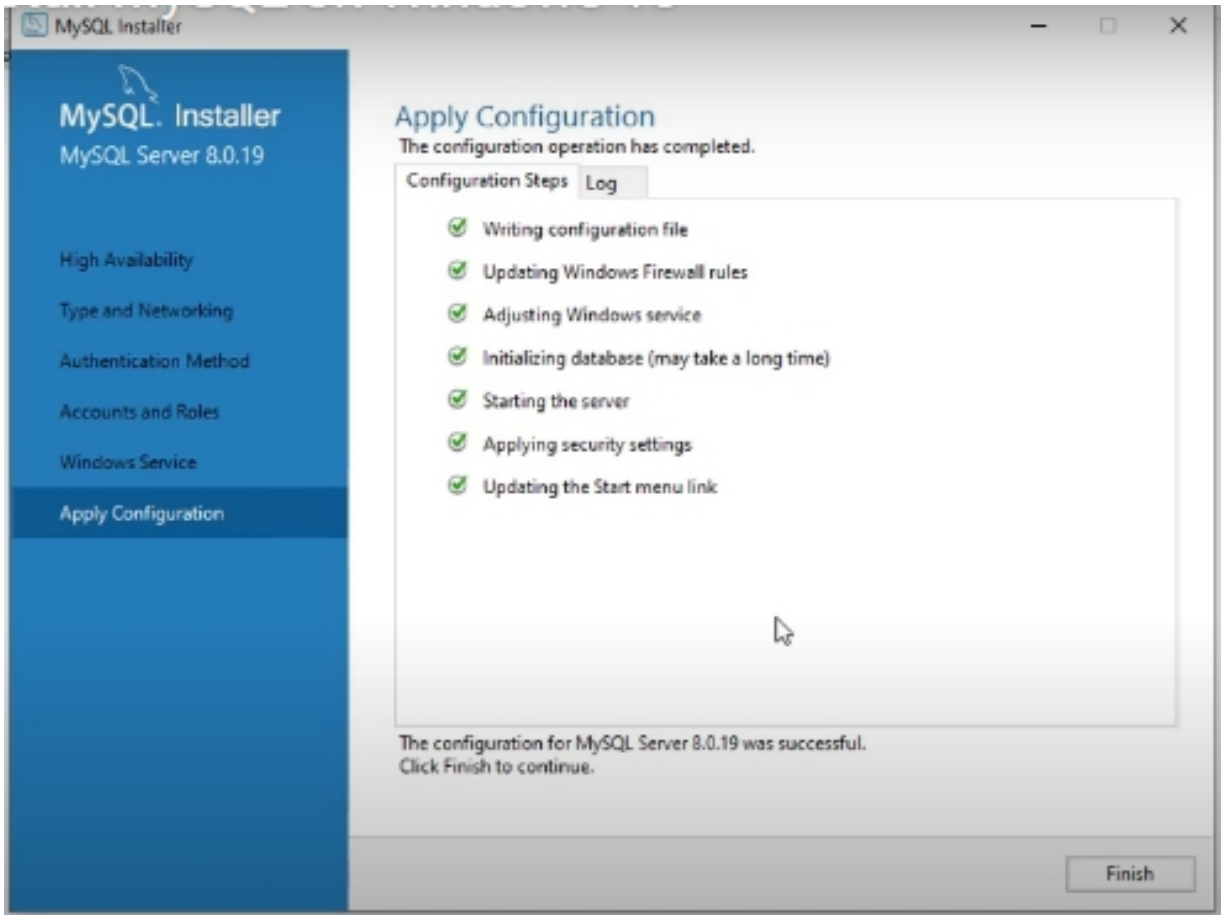


Skip type and networking/Authentication by clicking on next button

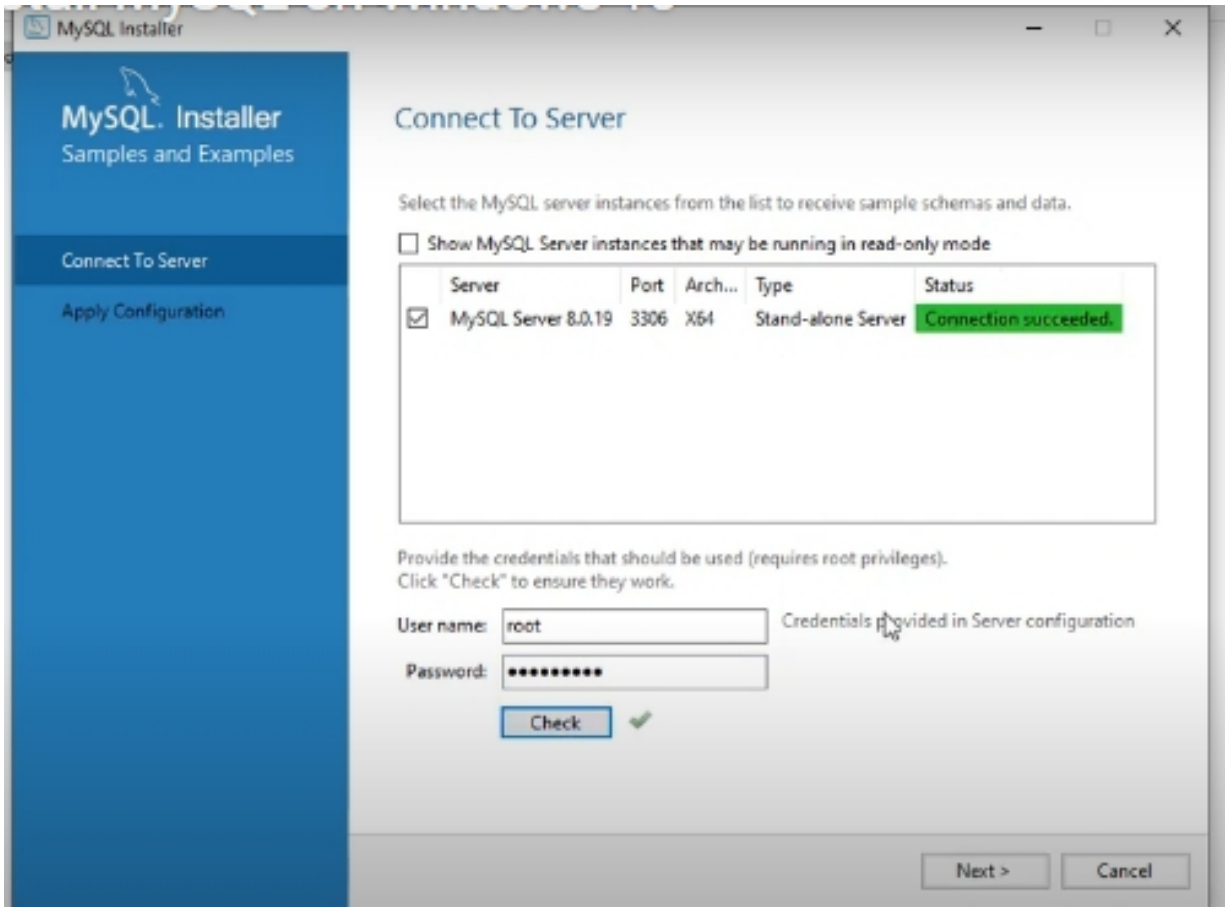
Now set password for your account, click on next button



Click next and reach to Apply configuration screen, click on execute than hit Finish. We have successfully configured the MySQL. Next skip the product configuration



Next Connect to the server screen will appear, enter your account password and check whether you have connected to the server or not – Connection succeeded, click on next button



Go to Apply configuration click on execute than click on Finish. We are successfully configured and installed MySQL in our PC.

MySQL work bench is successfully installed, connect to the server using your password and start working on MySQL.

CHAPTER 2: MYSQL SYNTAX

Just like other programming language MySQL follows the unique set of rules and guidelines known as MySQL syntax. It is very easy to run and execute as well.

2.1 Data Types in MySQL

MySQL data types specify the type of value that can be store in database table. Data types are mainly classified into three categories.

- **Numeric**
- **String**
- **Date and Time**

2.1.1 Numeric data type

It refers to the data that is in the form of numbers such as Integer, Float and Boolean. The following table lists the MySQL Numeric data type.

Numeric data Type	**Range**
BIT	0-1
TINYINT	0-255
SMALLINT	-32,768 - 32,767
BIGINT	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807
FLOAT	-1.79E+308 - 1.79E+308
REAL	-3.40E+38 - 3.40E+38

2.1.2 String data type

It allows us to store fixed or variable character values such as char, varchar, and text. The following table lists the MySQL String data type.

String data Type	**Description**
CHAR	Fixed length of 8000 characters
VARCHAR	Variable- length of 8000 characters
TEXT	Variable- length of 2GB data

2.1.3 Date and Time data type

It is used for storing values that can contain both date as well as time. The following table lists the MySQL Date and Time data type.

Date and Time data Type	**Description**
DATE	It stores date in the format (YYYY-MM-DD)
TIME	It stores time in the format (HH:MI:SS)
DATETIME	It stores date and time information (YYYY-MM-DD HH:MI:SS)

2.2 Operators in MySQL

MySQL operators use to specify certain condition in statement. Operators are broadly classified in five categories -

- **Arithmetic operator**
- **Logical operator**
- **Comparison operator**
- **Bitwise operator**
- **Set operator**

2.2.1 Arithmetic operators

It performs the arithmetical operation on numerical data in a table as well as SQL queries. The following table lists the MySQL Arithmetic operators.

Arithmetic operators	**Description**
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

2.2.2 Logical operators

The Logical operator performs Boolean operation which gives two results either true or false. The following table lists the MySQL Logical operators.

Logical operators	**Description**
AND	True if both expressions are true
IN	TRUE if the operand is equal to any one from the list of expression
OR	One expression must be true to get TRUE result
NOT	Reverses the value
Like	if the operand matches a pattern than true
BETWEEN	if operand is within a range than true

2.2.3 Comparison operators

It is also known as relational operator which compare values in a Database table. The following table lists the MySQL Comparison operators.

Comparison operators	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

2.2.4 Bitwise operators

It converts two integer values to binary bits, perform the AND, OR, and NOT operation on each bit. The following table lists the MySQL Bitwise operators.

Bitwise operators	**Description**
AND	True if both values are true
OR	True if any one value is true
NOT	Reverses the value

2.2.5 Set operators

Set operators used to combine rows from two or more table. The following table lists the MySQL Set operators.

Set operators	**Description**
UNION	Combines two or more result set without duplicating values
UNION ALL	Combines two or more result set including duplicating values
INTERSECT	It includes only the common values present in two or more result set
EXCEPT	It includes only results from first result set that are not included in second result set

CHAPTER 3: DATA DEFINITION LANGUAGE (DDL)

In previous chapter we have seen the syntax of MySQL which includes datatypes and operators used in MySQL, now we will learn about integrity constraints used in MySQL than we will learn about DDL and create a table using DDL commands.

3.1 Integrity constraints in MySQL

Integrity constraints are set of rules that a table's data or column must follow. We can use integrity Constraints with commands at the column or table level. The table-level Integrity constraints will be applied to the entire table, but the column level constraints are only applied to one column. Table shows the list of Integrity constraints used in MySQL.

Integrity constraints	**Description**
NOT NULL	It prevents a column from having a NULL value
DEFAULT	When no value is defined for a column, the DEFAULT Constraint provides a default value.
UNIQUE	It ensures that any value in a column is unique.
PRIMARY KEY	Uniquely identifies each row/record.
FOREIGN KEY	It recognizes a row/record in any database table uniquely.

3.2 Introduction to MySQL DDL

MySQL DDL statements are used to define and manipulate data base structure. It allows us to create a table, modify and maintain databases using commands.

Following are the commands of DDL.

- **CREATE- It is used to create a table and schema.**
- **ALTER - It is used to add, modify or delete columns or constraints from the database table.**
- **TRUNCATE - It is used to delete the data present in the table but this will not delete the table.**
- **DROP – It is used to delete the entire Table.**
- **RENAME – It is used to rename the Table.**

3.2.1 CREATE command

MySQL allows us to create a database table using CREATE command. Let's create a table of employees. If we execute the sequence of statement in MySQL workbench as shown below than a table of employees can be created.

```
1 CREATE TABLE employees (
2       employee_id INT,
3       first_name VARCHAR (50),
4       last_name VARCHAR (50),
5       hourly_pay DECIMAL (5,2),
6       hire_date DATE
7       );
```

I am going to show you how we can make some tables in MySQL. A table in a relational database they consist of rows and columns kind of like an Excel spreadsheet. In this topic we are going to create the table and the column.

Just type CREATE TABLE than the name of table, I am going to create a table named employees then add a set of parentheses semicolon at the end within the set of parentheses we will list the columns for employees, let's have an employee_id, first_name, hourly_pay, hire_date. Now each column we need to set the data type of what we are storing within each column

Let's set the data type for each and every column:

Employee_id INT – Data going to be whole integer

First_name VARCHAR (50) – series of max 50 characters

Last_name VARCHAR (50) – series of max 50 characters

Hourly_pay Decimal (5,2) – max 5 before decimal and 2 after decimal point (99999.99)

Hire_data DATE – date format (dd-mm-yyyy)

If you need to select the table than type

Syntax: SELECT * FROM table name;

Implementation: SELECT * FROM employees;

employee_id	first_name	last_name	hourly_pay	hire_date

We have successfully created a table of Employees with five columns.

3.2.2 RENAME command

If we need to rename the above table than type

Syntax: RENAME TABLE original name TO new name;

Implementation: RENAME TABLE employees TO workers;

Execute and get the result in MYSQL workbench.

3.2.3 DROP command

If we need to drop the above table than type

Syntax: DROP TABLE name of table;

Implementation: DROP TABLE employees;

Execute and get the result in MYSQL workbench.

3.2.4 ALTER command

If we need to alter the table, let's add the new column phone_number, type

ALTER TABLE employees

ADD phone_number VARCHAR (15);

Execute and get the result in MYSQL workbench.

Let's rename the phone_number to email , type

ALTER TABLE employees

RENAME COLUMN phone_number to email;

Execute and get the result in MYSQL workbench.

Let's add data type to email column, here we have to use MODIFY command, type

ALTER TABLE employees

MODIFY COLUMN email VARCHAR (100)

Execute and get the result in MYSQL workbench.

Let's add email after last_name, type

ALTER TABLE employees

MODIFY email VARCHAR (100)

AFTER last_name;

Execute and get the result in MYSQL workbench.

Let’s drop the column email, type

ALTER TABLE employees

DROP COLUMN email;

Execute and get the result in MYSQL workbench.

CHAPTER 4: DATA MANIPULATION LANGUAGE (DML)

4.1 Introduction to MySQL DML

MySQL DML deals with the modification of data within the tables of a database. DML statements used for adding(inserting), deleting, and modifying (updating) data in a database.

Following are the commands of DML

- **INSERT – It is used to add rows to a table manually.**
- **SELECT – We can access and manipulate data using Select command.**
- **DELETE – We can remove rows of existing table using Delete command.**
- **UPDATE - To change the actual data in a table.**

4.2 DML command execution

In previous chapter we have created an employees table, now let’s manipulate the table using DML commands.

4.2.1 INSERT command

In this topic, I am going to discuss about how we can insert rows into a table. We have a table named employees. I will select everything from my table employees.

```
SELECT * FROM employees;
```

employee_id	first_name	last_name	hourly_pay	hire_date

Let's insert a row of data using INSERT command.

To insert a row into a table we would type INSERT INTO the name of the table followed by VALUES parentheses semicolon, between this set of parentheses we will add all of the data for a row. We will follow this order beginning with employee ID, first name, last name so on but we do have to pay attention to the data types. Each piece of data will be separated with a comma.

Let's insert a row of data in Employees table

```
INSERT INTO employees

VALUES (1, "Eugene", "Krabs", 25.50, "2023-01-02");
```

Now we can execute these statements and get the result as shown below.

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02

Let's insert multiple rows at once, to do this MySQL allows us to add another set of parentheses each separated with a comma as shown below.

```
1 INSERT INTO employees
2 VALUES    (2, "Squidward", "Tentacles", 15.00, "2023-01-03"),
3           (3, "Spongebob", "Squarepants", 12.50, "2023-01-04"),
4           (4, "Patrick", "Star", 12.50, "2023-01-05"),
5           (5, "Sandy", "Cheeks", 17.25, "2023-01-06");
6
```

7 SELECT * FROM employees;

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06

4.2.2 SELECT command

In this section, we are going to discuss how to select data in a table. To select entire rows and columns in employees table, type

SELECT * FROM employees;

Sometimes you may not want all of the data. We can select specific columns.

To retrieve full name of employees, type

SELECT first_name, last_name FROM employees;

Output

first_name	last_name
Eugene	Krabs
Squidward	Tentacles
Spongebob	Squarepants
Patrick	Star
Sandy	Cheeks

We can also change the order of column, type

SELECT last_name, first_name FROM employees;

Execute and get the result in MYSQL workbench.

There is a WHERE clause if we are looking for something specific.

Let’s select all from employees where employee ID equals one.

```
1 SELECT *
2 FROM employees
3 WHERE employee_id =1;
```

This will give us a specific employee, the employee that has an ID of one.

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02

Note: Similarly, we can take reference of any column and their corresponding data such as first name, last name and so on.

We can also add conditions using comparison operator to WHERE clause.

Let’s find all employees that have an hourly pay greater than or equal to 15.

```
1 SELECT *
2 FROM employees
3 WHERE hourly_pay >= 15;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
5	Sandy	Cheeks	17.25	2023-01-06

We can also add integrity constraints to WHERE clause.

NULL means no data, let’s display the data of every employee that does have a hire date.

```
SELECT *FROM employees

WHERE hire_date IS NOT NULL;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06

4.2.3 UPDATE command

In this section, we are going discuss how to update data from a table. In our table employee Id 6 missing some data.

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton		

Let’s update single column hourly_pay = 10.25, type

```
1 UPDATE employees
2 SET hourly_pay = 10.25
3 WHERE employee_id = 6;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	

Let’s update multiple columns hourly_pay and hire_date, type

```
1 UPDATE employees
2 SET hourly_pay = 10.25,
3       hire_date = “2023-01-07”
4 WHERE employee_id = 6;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

We can UPDATE all of the rows within a column.

Let’s update hourly_pay = 10.25 to all employees, type

```
1 UPDATE employees
2 SET hourly_pay = 10.25;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	10.25	2023-01-02
2	Squidward	Tentacles	10.25	2023-01-03
3	Spongebob	Squarepants	10.25	2023-01-04

4	Patrick	Star	10.25	2023-01-05
5	Sandy	Cheeks	10.25	2023-01-06
6	Sheldon	Plankton	10.25	

We can also use integrity constraints with UPDATE command

Let’ set a field to NULL meaning no value, as in above example Sheldon Plankton’s hire_date is blank, type

```
1 UPDATE employees
2 SET hire_date = NULL
3 WHERE employee_id = 6;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	10.25	2023-01-02
2	Squidward	Tentacles	10.25	2023-01-03
3	Spongebob	Squarepants	10.25	2023-01-04
4	Patrick	Star	10.25	2023-01-05
5	Sandy	Cheeks	10.25	2023-01-06
6	Sheldon	Plankton	10.25	

Note: Plankton’s hire_date is now NULL.

4.2.4 DELETE command

To delete all rows in employees table, type

```
1 DELETE FROM employees;
```

Note: Be alert! entire rows in a table will be deleted.

To delete 6th row in employees table, type

```
1 DELETE FROM employees;
2 WHERE employee_id = 6;
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	10.25	2023-01-02
2	Squidward	Tentacles	10.25	2023-01-03
3	Spongebob	Squarepants	10.25	2023-01-04
4	Patrick	Star	10.25	2023-01-05
5	Sandy	Cheeks	10.25	2023-01-06

CHAPTER 5: DATA CONTROL LANGUAGE (DCL)

5.1 Introduction to MySQL DCL

MySQL DCL used to implement security on database objects. GRANT command gives the permission to access the database and REVOKE command is used to remove previously granted access privileges from a user.

The primary DCL commands in MySQL include:

GRANT: It allows users to perform specified tasks.

REVOKE: To remove user access rights.

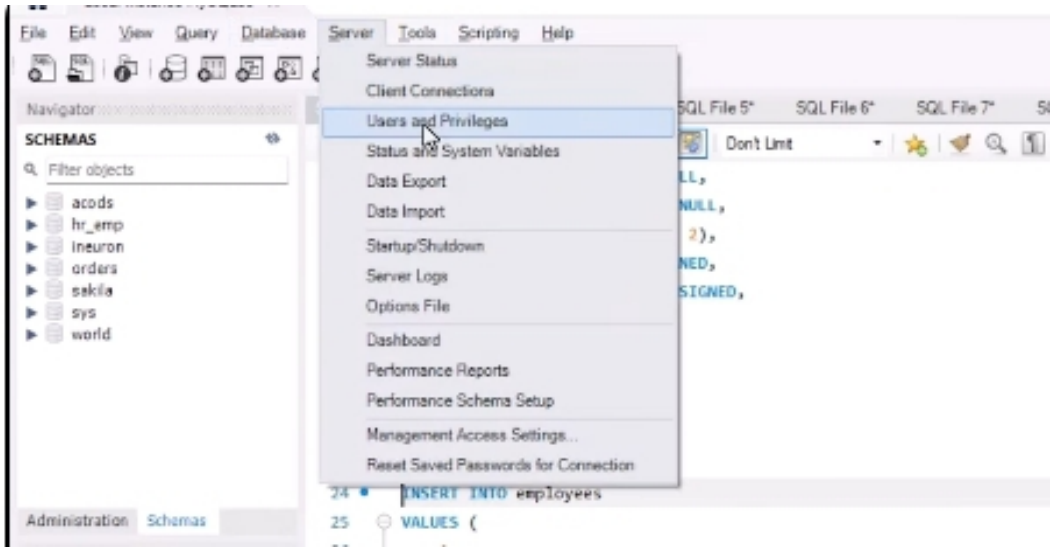
5.2 DCL command execution

Let us control the root database by giving access permission. DCL gives us facility to implement security on created database.

5.2.1 GRANT command

We have already created a root database (employees table), now Let's give access privileges to a new user(user1) using GRANT command.

To create new dummy user, let's move on to the MySQL workbench. We can click on one connection, we have already created a data base (employees table), to create a new user click on server and go to Users and Privileges



In “User and privileges” section we have all the users already created, now I will add an account, enter login Name – user_1 and password – 12345, confirm the password and apply.

Now go to Home of MySQL workbench, click on data base, and connect to database, enter user- User_1 and password – 12345 a blank screen will appear. There is no database over here because there is no access which is given to the user, we must close this application and go to the root data base(hr_emp) which is already created by us (employees table).

Enter the Grant command, type

```
1 USE hr_emp;
2 GRANT SELECT, INSERT, DELETE
3 On hr_emp
4 TO “User_1”;
```

Execute the sequence of statements

```
1 USE hr_emp;
2 GRANT SELECT
3 On employee
4 TO “User_1”;
```

Execute the sequence of statements

Now, Logout from MySQL workbench,

Login and reconnect with User- User_1 and password- 12345.

User_1 got the access to the employee data, Type

```
1 USE hr_emp
2 SELECT * FROM employees
```

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

Note: New user (User_1) having permission to access, insert and delete data from root database.

5.2.2 REVOKE command

Let's remove the permission of inserting data in to a employees table, login to the root database, press ctrl+V and type

USE hr_emp;

REVOKE INSERT

ON employees

FROM 'User_1';

Execute and get the result

User access rights for INSERT command is successfully removed.

Note: Now, the new user (User_1) is unable to insert data in root database (employees table).

CHAPTER 6: TRANSACTION CONTROL LANGUAGE (TCL)

6.1 Introduction to MySQL TCL

MySQL TCL is a subset of SQL commands used to manage transactions in a database. It allows us to roll back the changes made in database during transaction.

Following are the applications of MySQL TCL:

1. **Committing Transactions: TCL allows us to save the changes made in database.**
2. **Rolling Back Transactions: TCL used to undo the changes made during the transaction and restores the database to its previous state.**
3. **Setting Transaction Isolation Levels: TCL used to set the transaction isolation level. We can determine the level of concurrency and consistency.**
4. **Savepoints: TCL allows us to set savepoints within a transaction, allowing for partial rollback if needed.**
5. **Managing Transactions in Stored Procedures: We can use manage transactions within the scope of the procedure.**

6.2 TCL command execution

In this section we are going to explain auto commit, commit and rollback. Auto commit is a mode by default, Auto commit is set to on whenever you execute a transaction within MySQL that transaction is saved what if we were to make a transaction and we need to undo a transaction, for example what if we accidently delete all of the rows of the table.

Let's take an example of employees table which we have already created, I accidently deleted all rows because I forgot to add a WHERE clause so all rows are now gone, how to undo these changes, type

```
1 DELETE FROM employees;
```

To undo follow these steps, type

```
1 SET AUTOCOMMIT = OFF;
```

Now, transaction will not save automatically, we would need to manually save each transaction, type

```
1 COMMIT;
```

Execute it, Now I am going to select my table, type

```
1 SELECT * FROM employees;
```

Now if I delete the rows and to undo the changes, type

```
1 ROLLBACK
```

This will restore my current transaction back to previous save point where we used commit, type

```
1 SELECT * FROM employees;
```

Our table is back to what it previously was

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03

3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

Note: If we want to save the change, use COMMIT command.

CHAPTER 7: DATA QUERY LANGUAGE (DQL)

7.1 Introduction to MySQL DQL

DQL in MySQL is a part of the base grouping of SQL sub-languages. It is used to pass query to the data base schema. It offers maximum flexibility of using SELECT command to retrieve the information from database according to requirement of user. We are going to discuss in details about SELECT command.

When a SELECT statement is fired against a table the result is compiled into a further temporary table according to requirements.

We can use SELECT commands only if table is created and values are inserted. Let’s take an example of employees table which is already created.

7.1.1 Syntax1

SELECT * FROM table_name;

Implementation: To retrieve all information from employees table.

SELECT * FROM employees;

Execute and get the result

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

7.1.2 Syntax 2

SELECT Col1, Col2,…. FROM table_name;

Implementation: To retrieve specific column of table.

SELECT employee_id, first_name, last_name FROM employees;

Execute and get the result

employee_id	first_name	last_name
1	Eugene	Krabs
2	Squidward	Tentacles
3	Spongebob	Squarepants
4	Patrick	Star
5	Sandy	Cheeks
6	Sheldon	Plankton

Note: The order of column will be decided by the user.

7.1.3 Syntax 3

SELECT * FROM table_name WHERE condition;

Implementation: It will retrieve all the rows of data related to WHERE clause

SELECT * FROM employees WHERE employee_id< 4;

Execute and get the result

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04

Note: WHERE clause use to filter the rows depending upon condition.

7.1.4 Syntax 4

SELECT col1, col2, …… FROM table_name WHERE condition;

Implementation: Here first_name, last_name and hourly_pay of employees will be retrieved, depending upon WHERE clause.

SELECT first_name, last_name, hourly_pay FROM employees WHERE hourly_pay<15;

Execute and get the result

first_name	Last_name	hourly_pay
Spongebob	Squarepants	12.50
Patrick	Star	12.50
Sheldon	Plankton	10.25

7.2 Current date and time using MySQL

Let's create a temporary demo table, type

```
1 CREATE TABLE demo(
2 my_date DATE,
3 my_time TIME,
4 my_datetime DATETIME
5 );
```

Execute it, we have successfully created a table with three columns – my_date, my_time, my_datetime.

If we want to get the current date than we need to create a timestamp

Let's insert data into a DEMO table, type

```
1 INSERT INTO demo
2 VALUE(CURRENT_DATE(), CURRENT_TIME(), NOW());
3
4 SELECT * FROM demo;
```

Execute it and get the current date and time.

my_date	my_time	my_datetime
2024-07-23	07:27:58	2024-07-23 07:27:58

Let's add a day to current date, type

```
1 INSERT INTO demo
2 VALUES(CURRENT_DATE() + 1, NULL, NULL)
3
4 SELECT * FROM demo;
```

Execute and get the result

my_date	my_time	my_datetime
2024-07-23	07:27:58	2024-07-23 07:27:58

2024-07-24		

Note: Here time and datetime set to NULL.

CHAPTER 8: ROLE OF CONSTRAINTS IN MYSQL

In this section we will discuss about the constraints in MySQL.

8.1 UNIQUE constraint in MySQL

The unique constraint shows the values of a column are all different. We add this constraint when we create a table, so let's a create a new table

```
1 CREATE TABLE products (
2       product_id INT,
3       product_name VARCHAR (25) UNIQUE,
4       price DECIMAL (4,2)
5 );
```

Since we have used UNIQUE constraint so we cannot insert any product names that are the same they all have to be unique.

In case if we forgot to add UNIQUE constraint during table creation than we can use the following sequence of commands to add UNIQUE constraint

```
1 ALTER TABLE products
2 ADD CONSTRAINT
3 UNIQUE (product_name);
```

Execute and get the result.

UNIQUE constraint is successfully added to column product_name.

If we insert the same product twice in product_name column than an error message will be displayed. So, all the values in product_name need to be different.

8.2 NOT NULL constraint

If we use NOT NULL constraint than the value within a specific column should not be null. If we forgot to add value than error will occur. We cannot enter a null value. We can set to be zero that’s acceptable. It’s a useful constraint to verify input if there’s any column that you don’t want to have any null values.

8.3 CHECK constraint

The CHECK constraint is used to limit what values can be placed in a column, for example I live in a United States depending on which state you live in there is a minimum hourly wage that employers must pay in this example let's set an hourly pay to our employees table, every employee needs to be paid at least minimum wage in that region, we can do that with the CHECK constraint.

Let's create employees table with constraint CHECK

```
1 CREATE TABLE employees (
2       employee_id INT,
3       first_name VARCHAR (50),
4       last_name VARCHAR (50),
5       hourly_pay DECIMAL (5,2),
6       hire_date DATE
7       CONSTRAINT chk_hourly_pay CHECK (hourly_pay >= 10.00)
8       );
```

Here, check constraint check hourly pay will be violated if hourly_pay is less than 10.00. so after executing the above sequence of statement a user must be alert not to enter any value less than 10 in hourly_pay column.

We can also remove the CHECK constraint, type

```
1 ALTER TABLE employees
2 DROP CHECK chk_hourly_pay;
```

Execute and get the result

8.4 DEFAULT constraint

When we are inserting a new row if we do not specify a value for a column by default, we can add some value that we set here's an example. Let's create products table and set the price default value to 0. Type

```
1 CREATE TABLE products (
2       product_id INT,
3       product_name VARCHAR(25),
4       price DECIMAL(4,2) DEFAULT 0
5 );
```

Here, if we insert product id and product name in the table than the price 0 will be automatically included due to DEFAULT constraint.

Let's insert some values in to the products table, type

```
1 INSERT INTO products (product_id, product_name)
2 VALUES (10, "spoon"),
3       (11, "fork");
4 SELECT * FROM products;
```

Output

product_id	product_name	price
10	spoon	0.00
11	fork	0.00

8.5 PRIMARY KEY constraint

The PRIMARY KEY is applied to a column where each value in that column must both be unique and not NULL. It is typically used as a unique identifier, for example I live in United States each citizen within the United States has a unique social security number. There is a strong possibility that two citizens in the United States shares the same first name and last name. Let us take an example "John Smith", if we search for a citizen with a unique social security number than we know for sure we have the right person. A table can have only one primary key constraint.

Let's create a table of transactions with two columns and PRIMARY KEY constraint, type

```
1 CREATE TABLE transactions(
2       transaction_id INT PRIMARY KEY,
3       amount DECIMAL(5,2)
4 );
```

Execute and get the result

We can also add primary key to any table, type

```
1 ALTER TABLE products
2 ADD CONSTRAINT
3 PRIMARY KEY (transaction_id);
```

8.6 Auto increment attribute in MYSQL

The auto increment attribute can be applied to a column that is set as a key whenever we insert a new row our primary key can be populated automatically then each subsequent row is auto incremented.

Let's drop the transaction table and recreate with auto increment feature, type

```
1 CREATE TABLE transactions (
2       transaction_id INT PRIMARY KEY AUTO_INCREMENT,
3       amount DECIMAL (5,2)
4 );
```

Note: By default, auto increment will set to 1.

Let's insert some values in transaction table, no need to insert transaction_id, it will be automatically assigned.

```
1 INSERT INTO transactions (amount)
2 VALUES (2.56);
3 SELECT * FROM transactions;
```

transaction_id	amount
1	2.56

Note: Here transaction id automatically inserted.

We could set our primary key to begin at a different value, type

```
1 ALTER TABLE transactions
2 AUTO_INCREMENT = 1000;
```

Execute and get the result, transaction ID set to 1000 onwards. We must delete entire rows and insert new values.

8.7 FOREIGN KEY constraint

Foreign key is a PRIMARY KEY from one table that can be found within a different table using a foreign key we can establish a link between two tables. There are two primary benefits to this in our transaction table if I were to take a look at the customer ID of who initiated this transaction, I could refer to the customers table then find the first and last name of that customer.

Let's create transaction and customer table, establish a link between them using foreign key

I am going to create a new table of customers, type

```
1 CREATE TABLE customers (
2       customer_id INT PRIMARY KEY AUTO_INCREMENT.
3       first_name VARCHAR (50),
4       last_name VARCHAR (50)
5 );
```

Execute and get the result

Now insert some values in to the table, type

```
1 INSERT INTO customers (first_name, last_name)
2 VALUES ("Fred", "Fish"),
3       ("Larry", "Lobster"),
4       ("Bubble", "Bass");
5
6 SELECT * FROM customers;
```

Output

customer_id	first_name	last_name
1	Fred	Fish
2	Larry	Lobster
3	Bubble	Bass

Now, we are going to create a link between our customers table and our transactions table via customer ID. Let's create a transaction table with FOREIGN KEY constraint, type

```
1 CREATE TABLE transactions (
2       transaction_id INT PRIMARY KEY AUTO_INCREMENT,
3       amount DECIMAL (5,2),
4       customer_id INT,
5       FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
6 );
```

Execute and get the result

transaction_id	amount	customer_id

Note: Here customer_id column is FOREIGN KEY.

Link between two table using foreign key is shown below

customer_id	first_name	last_name
1	Fred	Fish
2	Larry	Lobster
3	Bubble	Bass

transaction_id	amount	customer_id
1000	4.89	2
1001	2.68	3
1002	3.32	1
1003	4.99	3

8.8 JOIN clause in MySQL

A JOIN is a clause that is used to combine rows from two or more tables based on a related column between them such as a foreign key. Let's join two previously created table. We have two tables a table of transaction and a table of customers.

1. Table of Transaction

transaction_id	amount	customer_id
1000	4.89	2
1001	2.68	3
1002	3.32	1
1003	4.99	3

2. Table of Customers

customer_id	first_name	last_name
1	Fred	Fish
2	Larry	Lobster
3	Bubble	Bass

Let's inner join both table, type

```
1 SELECT *
2 FROM transactions INNER JOIN customers
3 ON transaction.customer_id = customer.customer_id;
```

Execute and get the result

Output

transaction_id	amoun	customer_id	customer_id	first_name	last_nam

	t				e
1000	4.89	2	2	Larry	Lobster
1001	2.68	3	3	Bubble	Bass
1002	3.32	1	1	Fred	Fish
1003	4.99	3	3	Bubble	Bass

CHAPTER 9: FUNCTIONS AND LOGICAL OPERATORS IN MYSQL

9.1 Functions in MySQL

A stored program that accepts parameter from the user and returns the value known as Function . MySQL provides lot of functions. Some of them shown below.

Let's count number of rows in amount column of transaction table, type

```
1 SELECT COUNT (amount)
2 FROM transaction;
```

Execute and get the result

Let's find the maximum value in amount column, type

```
1 SELECT MAX(amount) AS maximum
2 FROM transaction;
```

Execute and get the result

Note: Here column name will be changed in output "maximum".

Let's find the minimum value in amount column, type

```
1 SELECT MIN(amount) AS minimum
2 FROM transaction;
```

Execute and get the result

Let's find average value of amount column, type

```
1 SELECT AVG(amount) AS average
2 FROM transaction;
```

Execute and get the result

Let’s find sum of amount column, type

```
1 SELECT SUM(amount) AS sum
2 FROM transaction;
```

Execute and get the result

Let’s concatenate the first and last name of employees, type

```
1 SELECT CONCAT (first_name, last_name) AS full_name
2 FROM employees;
```

Execute and get the result

Note: Here, I have explained the process of using functions in MySQL.

9.2 Logical operators in MySQL

The Logical operator is a keyword use to combine more than one condition.

Let’s use our employees table to demonstrate logical operators.

SELECT * FROM employees;

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

9.2.1 AND operator

```
1 SELECT *
2 FROM employees
3 WHERE hire_date< “2023-01-6” AND hourly_pay<15;
```

It will return any results that match these two criteria.

employee_id	first_name	last_name	hourly_pay	hire_date
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05

9.2.2 OR operator

```
1 SELECT *
2 FROM employees
3 WHERE hire_date< “2023-01-6” OR hourly_pay<15;
```

It will return any results that match one of these two criteria.

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05

9.2.3 NOT operator

```
1 SELECT *
2 FROM employees
3 WHERE NOT hourly_pay = 12.50;
```

Here, NOT basically reverses anything you say.

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03

9.2.4 Between operator

```
1 SELECT *
2 FROM employees
3 WHERE hire_date BETWEEN “2023-01-04” AND “2023-01-07”;
```

Execute and get the result according to criteria.

employee_id	first_name	last_name	hourly_pay	hire_date
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

9.2.5 IN operator

```
1 SELECT *
2 FROM employees
3 WHERE hourly_pay IN (17.25,10.25);
```

We can find any values that are within a set.

employee_id	first_name	last_name	hourly_pay	hire_date
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

CHAPTER 10: SOME IMPORTANT MYSQL COMMANDS

In this section, we are going to discuss about some important clauses used in MySQL to handle databases or schema object.

10.1 Wild card character

Wild card is a special character used to handle the database. There are two wild card character – precent and underscore.

Let's use % in our employees table to retrieve first name begins with S, type

```
1 SELECT * FROM employees
2 WHERE first_name LIKE "s%";
```

Execute and get the result

employee_id	first_name	last_name	hourly_pay	hire_date
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

Let's find the employee hired in January from employees table, type

```
1 SELECT * FROM employees
2 WHERE hire_date LIKE "____-01-__";
```

Note: Year uses four underscore and month uses two underscore.

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02

2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

10.2 Order by clause

The ORDER BY clause sorts the results of a query in either ascending or descending order based on which column we list.

Let's change the order of our employees table's first name, type

```
1 SELECT * FROM employees
2 ORDER BY first_name DESC;
```

Execute and get the result

```
1 SELECT * FROM employees
2 ORDER BY first_name ASC;
```

Execute and get the result

10.3 Limit clause

The LIMIT clause use to limit the number of records. It is very useful if you are working with a large database.

Let's limit the number of employees that are displayed from employees table, type

```
1 SELECT * FROM employees
2 LIMIT 2;
```

Note: Only first two employee's details will be displayed.

Output

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03

10.4 Union operator

The UNION operator combines the results of two or more select statements.

Let's combine the first_name and last_name of our two tables- Employees table and Customers table.

Employees table

employee_id	first_name	last_name	hourly_pay	hire_date
1	Eugene	Krabs	25.50	2023-01-02
2	Squidward	Tentacles	15.00	2023-01-03
3	Spongebob	Squarepants	12.50	2023-01-04
4	Patrick	Star	12.50	2023-01-05
5	Sandy	Cheeks	17.25	2023-01-06
6	Sheldon	Plankton	10.25	2023-01-07

Customers table

customer_id	first_name	last_name
1	Fred	Fish
2	Larry	Lobster
3	Bubble	Bass

To combine using UNION, type

```
1 SELECT first_name, last_name FROM employees
2 UNION
3 SELECT first_name, last_name FROM customers
```

Output

first_name	last_name
Eugene	Krabs
Squidward	Tentacles
Spongebob	Squarepants
Patrick	Star
Sandy	Cheeks
Sheldon	Plankton
Fred	Fish
Larry	Lobster
Bubble	Bass

Note: If we want to combine duplicate values than use UNION ALL instead of UNION.

10.5 Views in MySQL

We can create a virtual table using VIEWS in MySQL. The fields in a view are fields from one or more real tables in the database.

Let’s view our employees table as employees_attendance table, type

```
1 CREATE employees_attendance AS
2 SELECT first_name, last_name
3 FROM employees;
```

So everything was successful, we have a new view, type

```
1 SELECT * FROM employees_attendance;
```

Output

first_name	last_name
Eugene	Krabs
Squidward	Tentacles
Spongebob	Squarepants
Patrick	Star
Sandy	Cheeks
Sheldon	Plankton

10.6 Indexes in MySQL

Indexes are used to find the values within a specific column more quickly. MySQL normally searches sequentially through a column.

Customer's table

customer_id	first_name	last_name
1	Fred	Fish
2	Larry	Lobster
3	Bubble	Bass

Let's create index in our customer table's last name, type

```
1 CREATE INDEX last_name_idx
2 ON customers(last_name);
```

Execute and get the result

If I want to search for a customer by last name, type

```
1 SELECT * FROM customers
2 WHERE last_name = "Bass";
```

Output

customer_id	first_name	last_name
3	Bubble	Bass

A query within the query is called the subquery.

Let's display the name of employees whose hourly pay is equal to average hourly_pay from our employees table, type

```
1 SELECT first_name, last_name, hourly_pay,
2       (SELECT AVG (hourly_pay) FROM employees) AS avg_pay
3 FROM employees;
```

Execute and get the result

10.8 Group by clause

A GROUP BY clause will aggregate all rows by a specific column often used with aggregate functions such as SUM(), MAX(), MIN(), AVG(), COUNT() etc.

Transaction table

transaction_id	amount	Order_date
1000	4.89	2023-01-01
1001	2.68	2023-01-01
1002	3.32	2023-01-02
1003	4.99	2023-01-02

Let’s apply a group by clause in transaction table to calculate transaction per day, type

```
1 SELECT SUM (amount), order_date
2 FROM transactions
3 GROUP BY order_date;
```

Output

SUM(amount)	order_date
7.57	2023-01-01
8.31	2023-01-02

10.9 Roll up clause

It is an extension of GROUP BY clause which produces another row and shows the super aggregate values.

Let’s calculate the grand total of transaction made in transaction table using ROLLUP, type

```
1 SELECT SUM (amount), order_date
2 FROM transactions
3 GROUP BY order_date WITH ROLLUP;
```

Output

SUM(amount)	order_date
7.57	2023-01-01
8.31	2023-01-02
15.88	

10.10 On delete clause

There are two versions of ON DELETE clause

1. ON DELETE SET NULL

2. ON DELETE CASCADE

When a foreign key is deleted than we can replace the foreign key with the value 0.

Let's take an example of customer table where customer id is a foreign key, so to delete the entire selected row, type

```
1 SET foreign_key_checks = 0;

2 DELETE FROM customers

3 WHERE customer_id = 4;

4 SELECT * FROM customers;
```

Execute and get the result

THE END