MYSQL - Basic Guide!

Contents

[**1.What is SQL?** 3](#_Toc99141084)

[**2. ‘SELECT’ statement?** 3](#_Toc99141085)

[SELECT DISTINCT -statement 5](#_Toc99141086)

[**3.‘WHERE’ statement?** 5](#_Toc99141087)

[**Operator** 5](#_Toc99141088)

[**BETWEEN**  - 6](#_Toc99141089)

[Between a certain range 6](#_Toc99141090)

[**LIKE** – 6](#_Toc99141091)

[search for pattern 6](#_Toc99141092)

[ **IN** – 6](#_Toc99141093)

[to specify multiple possible values for a column 6](#_Toc99141094)

[ **AND –** 7](#_Toc99141095)

[ **OR -** 7](#_Toc99141096)

[ **NOT -** 7](#_Toc99141097)

[ **REGEXP –** 7](#_Toc99141098)

[**4.‘ORDER BY’ statement?** 8](#_Toc99141099)

[**5.‘LIMIT’ statement?** 9](#_Toc99141100)

[**6.‘INSERT INTO ’ statement?** 10](#_Toc99141101)

[**7.‘UPDATE’ statement?** 11](#_Toc99141102)

[**8.‘DELETE’ statement?** 12](#_Toc99141103)

[**9.‘CASE’ statement?** 12](#_Toc99141104)

[**10.‘DATABASE’ statement?** 13](#_Toc99141105)

[**•** **‘CREATE DATABASE’ :** 13](#_Toc99141106)

[**•** **’DROP DATABASE’ :** 13](#_Toc99141107)

[**• DEFFERENTIAL ‘BACKUP DATABASE’ :** 14](#_Toc99141108)

[**11.CREATE TABLE statement?** 14](#_Toc99141109)

[**•** **‘CREATE table’ :** 14](#_Toc99141110)

[**•** **’ALTER TABLE’ :** 15](#_Toc99141111)

[**12.FUNCTIONS in sql !** 15](#_Toc99141112)

[**13. ‘GROUP BY’ in sql !** 16](#_Toc99141113)

[**14. ‘HAVING’ clause !** 16](#_Toc99141114)

[**15. ‘EXISTS’ operator !** 16](#_Toc99141115)

[**16. stored procedure!** 17](#_Toc99141116)

[**17. SQL constraints!** 18](#_Toc99141117)

[**18. SQL joins!** 18](#_Toc99141118)

[**• (INNER) JOIN :** 18](#_Toc99141119)

[**•** **Self joins** 19](#_Toc99141120)

[**•** **LEFT (OUTER) JOIN :** 20](#_Toc99141121)

[**•** **RIGHT (OUTER) JOIN :** 20](#_Toc99141122)

[**•** **FULL (OUTER) JOIN :** 20](#_Toc99141123)

[**19. SQL UNIONS!** 21](#_Toc99141124)

[Step by step procedures: 21](#_Toc99141125)

[**1 .Download MySQL** 21](#_Toc99141126)

[**2.Connecting to mysql database in python** 22](#_Toc99141127)

[ **Using MySQLdb** 23](#_Toc99141128)

[ **Using windows command prompt** 23](#_Toc99141129)

[ **Using MySql shell** 24](#_Toc99141130)

[**3.Create Database** 24](#_Toc99141131)

[**4.Create Table** 25](#_Toc99141132)

[**5.Insert data in table** 25](#_Toc99141133)

[**6.Select data from table** 26](#_Toc99141134)

[**7.Update data in the table** 26](#_Toc99141135)

[**8.Delete data from the table** 27](#_Toc99141136)

# **1.What is SQL?**

SQL stands for structured Query language. Used to store,access and manipulate databases. It’s a relational database management system (RDBMS).

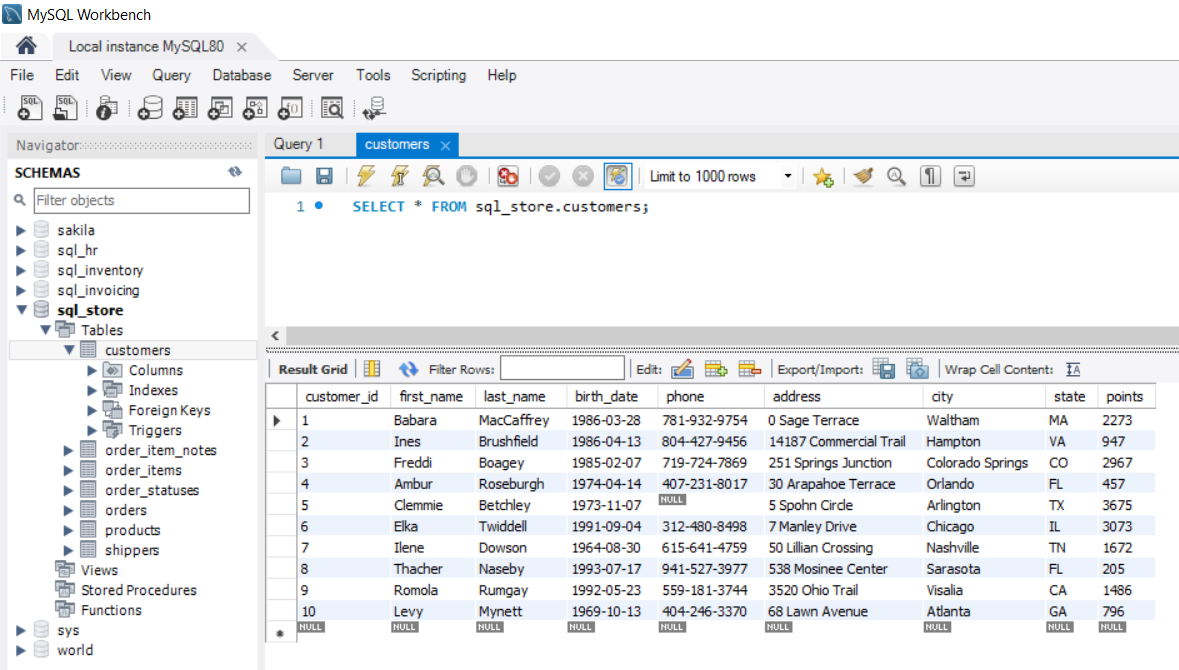
The database mainly consist of TABLES which consist of various rows(records) and columns(fields)

SQL keywords are not case sensitive and require a semicolon after the query in case of having multiple lines present.

We can comment using ‘--’ in starting of SQL line

# **2. ‘SELECT’ statement?**

* Used to select the data from databases



**USE sql\_store;**

**SELECT \* FROM customers;**

**‘**sql\_store’ - database name & ‘customers’ - table name

* We can specifically mention the field names

**SELECT customer\_id,first\_name,points FROM customers;**

* We can select a different data from available details in the data

**SELECT customer\_id,**

**first\_name,**

**Points,**

**Points+10 AS special\_points**

**(points+10)\*100 AS ‘discount factor’**

**FROM customers;**

Here we are creating two fields (special\_points and discount factor) with data we already have. We use an ‘AS’ statement to give them a new field name.

When we give names to new fields, when they have space b/w words give in single quotes ,when single word -no need of quotes.

## SELECT DISTINCT -statement

Inorder to get distinct data by removing duplicates.

**SELECT DISTINCT states FROM customers;**

Inorder to get count of distinct data by removing duplicates.

**SELECT COUNT(DISTINCT states) FROM customers;**

# **3.‘WHERE’ statement?**

* Used to extract only those records that fulfill a special condition

**SELECT \***

**FROM customers**

**WHERE condition;**

**SELECT points**

**FROM customers**

**WHERE points>500;**

**SELECT customer\_id**

**FROM customers**

**WHERE customer\_id=1;**

## **Operator**

= equal

> Greater than

< Less than

>= Greater than or equals

<= Less than or equals

<> or != Not equal

## **BETWEEN** -

## Between a certain range

**SELECT points**

**FROM customers**

**WHERE points BETWEEN 1000 AND 3000;**

## **LIKE** –

## search for pattern

**SELECT \***

**FROM customers**

**WHERE last\_name LIKE ‘b%’;**

‘%’- number of characters not fixed. Meaning based on where it is placed.

‘b%’ - select records starting with word b

‘%b’ - select records ending with word b

‘%b%’ - select records with word b anywhere.

‘-’ - used to mention specific no of characters.

‘-b’ One ‘-’ indicate one character.two only characters with second character b

‘----b’ - ‘----’ indicates 4 characters. 5 characters only with fifth character b

## **IN** –

## to specify multiple possible values for a column

**SELECT points ,states**

**FROM customers**

**WHERE states IN (‘va’,‘fl’,‘ga’);**

**SELECT points ,states**

**FROM customers**

**WHERE states NOT IN (‘va’,‘fl’,‘ga’);**

## **AND –**

displays records when all conditions separated by AND is True

**SELECT points ,states**

**FROM customers**

**WHERE points >3000 AND states= ‘va’;**

## **OR -**

displays records when any one conditions separated by OR is True

**SELECT points,states**

**FROM customers**

**WHERE points >3000 OR states= ‘va’;**

## **NOT -**

displays records when all conditions are not True

**SELECT points,states**

**FROM customers**

**WHERE NOT (points >3000 OR states= ‘va’)**

## **REGEXP –**

**Regular expression**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘AVENUE;**

**Select records with ‘avenue’ in address anywhere.**

**^ - indicate starting**

**$ - indicate end of string**

**| - indicate multiple conditions**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘^AVENUE’;**

**Select records with starting ‘avenue’.**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘AVENUE$’;**

**Select records with ending ‘avenue’.**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘TRAIL|AVENUE’;**

**Select records with multiple matches anywhere in string.**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘[GNF]E’;**

**Select records with any matches like GE or NE or FE**

**SELECT address**

**FROM customers**

**WHERE address REGEXP ‘[a-d]E’;**

**Select records with any matches like AE or BE or CE or DE.**

**IS NULL - check whether record is a null value**

**SELECT address**

**FROM customers**

**WHERE address IS NULL;**

**SELECT address**

**FROM customers**

**WHERE address IS NOT NULL;**

# **4.‘ORDER BY’ statement?**

**• Used to sort the result in ascending or descending order.**

**• Ascending**

**SELECT \***

**FROM customers**

**ORDER BY states;**

**• Descending**

**SELECT \***

**FROM customers**

**ORDER BY states DESC;**

**• Sorting by two fields first states descending then first\_name ascending**

**SELECT \***

**FROM customers**

**ORDER BY states DESC,first\_name;**

# **5.‘LIMIT’ statement?**

**• Used to limit the no of selecting records**

**SELECT \***

**FROM customers**

**LIMIT 3**

**Record no is limited to 3 .**

**SELECT \***

**FROM customers**

**LIMIT 6,3**

**Skip first 6 (since offset given as 6) and take the next 3 records .**

**5.General syntax of MySQL Query?**

**• Order of statements**

**SELECT> FROM>WHERE>ORDER BY>LIMIT**

**SELECT \***

**FROM customers**

**WHERE points>3000**

**ORDER BY First\_name**

**LIMIT 3**

# **6.‘INSERT INTO ’ statement?**

**• Used to insert new records in a table**

**• Insert one record**

**INSERT INTO customers**

**VALUES (Default, ‘bhagya’, ‘suresh’, NULL, Default, ‘kadavil’, ‘thrissur’ , ‘kerala’,0);**

**either we can use Default or values that we give as default.**

**There is no need to fill all fields. But while inserting we have to fill ‘NN’ -not null values. We can skip other fields.**

**INSERT INTO customers (first\_name,last\_name,address,city,state)**

**VALUES (‘bhagya’, ‘suresh’, ‘kadavil’, ‘thrissur’ , ‘kerala’);**

**• Insert multiple tables:**

**INSERT INTO customers (first\_name,last\_name,address,city,state)**

**VALUES (‘bhagya’, ‘suresh’, ‘address1’, ‘city1’, ‘state1’),**

**(‘bhavya’, ‘suresh’, ‘address2’, ‘city2’, ‘state2’),**

**(‘baghu’, ‘suresh’, ‘address3’, ‘city3’, ‘state3’)**

**• Insert hierarchical rows:**

**INSERT INTO orders (customer\_id,order\_date,status)**

**VALUES (1, ‘2022-01-01’, 1);**

**INSERT INTO order\_items(order\_id,order\_item\_id,quantity,price)**

**VALUES (LAST\_INSERT\_ID(),1,1,2.95)**

**(LAST\_INSERT\_ID(),2,1,7.95);**

# **7.‘UPDATE’ statement?**

**• Used to modify existing record in a table**

**UPDATE customers**

**SET points=points+50,**

**City= ‘VA’**

**WHERE birth\_date< ‘1990-01-01’;**

**• Using subqueries in Update statement**

**UPDATE customers**

**SET points=points+50,**

**City= ‘VA’**

**WHERE customer\_id=**

**(SELECT customer\_id FROM clients WHERE name= ‘important’**

**UPDATE customers**

**SET points=points+50,**

**City= ‘VA’**

**WHERE states=**

**(SELECT states IN (‘CA’, ‘NY’))**

# **8.‘DELETE’ statement?**

**• Used to delete existing records**

**DELETE FROM customers**

**WHERE address IS NULL;**

**DELETE FROM customers**

**WHERE states=**

**(SELECT states IN (‘CA’, ‘NY’))**

**• Delete complete table**

**DELETE FROM customers**

# **9.‘CASE’ statement?**

**• Goes through conditions and return a value when the first condition is met. If not return value in ‘ELSE’.**

**CASE**

**WHEN condition1 THEN result1**

**WHEN condition2 THEN result2**

**WHEN condition3 THEN result3**

**ELSE result**

**END;**

**SELECT customer\_id,first\_name,**

**CASE**

**WHEN points>3000 THEN “Golden customer”**

**WHEN points>2000 THEN “Silver customer”**

**WHEN points>1000 THEN “Bronze customer”**

**ELSE “New customer”**

**END AS customer\_value**

**FROM customers;**

**SELECT customer\_id,first\_name,**

**FROM customers;**

**ORDER BY**

**(CASE**

**WHEN city IS NULL THEN country**

**ELSE city**

**END);**

# **10.‘DATABASE’ statement?**

## **• ‘CREATE DATABASE’ :**

**Used to create a new database.**

**CREATE DATABASE sql\_store;**

## **• ’DROP DATABASE’ :**

**Used to delete a database.**

**DROP DATABASE customers;**

**• ’BACKUP DATABASE’ :**

**Used to backup a database.**

**BACKUP DATABASE customers**

**TO DISK= “filepath”;**

**BACKUP DATABASE customers**

**TO DISK= “D:\backup\testdb.bak”;**

## **• DEFFERENTIAL ‘BACKUP DATABASE’ :**

**Used to backup database those are changed since last backup**

**BACKUP DATABASE customers**

**TO DISK= “filepath”**

**WITH DIFFERENTIAL;**

# **11.CREATE TABLE statement?**

## **• ‘CREATE table’ :**

**Used to create a new table.**

**CREATE TABLE `customers` (**

**`customer\_id` int(11) NOT NULL AUTO\_INCREMENT,**

**`first\_name` varchar(50) NOT NULL,**

**`last\_name` varchar(50) NOT NULL,**

**`birth\_date` date DEFAULT NULL,**

**`phone` varchar(50) DEFAULT NULL,**

**`address` varchar(50) NOT NULL,**

**`city` varchar(50) NOT NULL,**

**`state` char(2) NOT NULL,**

**`points` int(11) NOT NULL DEFAULT '0',**

**PRIMARY KEY (`customer\_id`))**

**• CREATE a table using another table :**

**CREATE TABLE `new\_customers` AS**

**SELECT \* FROM customers;**

**INSERT INTO new\_customers**

**SELECT \* FROM customers WHERE points<1000;**

**• ’DROP TABLE’ :**

**Used to delete a table.**

**DROP TABLE customers;**

## **• ’ALTER TABLE’ :**

**Used to alter add, delete, modify columns in existing table.**

**ALTER TABLE customers**

**ADD email varchar(50);**

**ALTER TABLE customers**

**DROP COLUMN email ;**

**ALTER TABLE customers**

**MODIFY COLUMN email varchar(100);**

# **12.FUNCTIONS in sql !**

• MAX() : Largest value of selected column

• MIN() : Smallest value of selected column

• COUNT(): No of rows that matches a specific criteria

• AVG(): Returns average value of numeric column

• SUM(): Returns the sum of numeric column

**SELECT**

**MAX(points)**

**MIN(points)**

**COUNT(points)**

**AVG(points)**

**SUM(points**

**FROM customers WHERE points<1000;**

# 

# **13. ‘GROUP BY’ in sql !**

• Group rows that have same value into summary row. Used to along with aggregate functions

**SELECT COUNT(customer\_id),country**

**FROM customers**

**GROUP BY country**

**ORDER BY COUNT(customer\_id) DESC;**

# 

# 

# **14. ‘HAVING’ clause !**

• Replaces ‘WHERE’ when we use aggregate functions (min,max,count).Instead WHERE we use HAVING **.**

**SELECT COUNT(customer\_id),country**

**FROM customers**

**GROUP BY country**

**HAVING COUNT(customer\_id)>5**

**ORDER BY COUNT(customer\_id) DESC;**

# **15. ‘EXISTS’ operator !**

• Use to test existence of any record in subquery. Returns True when record found.

**SELECT supplier\_name**

**FROM suppliers**

**WHERE EXISTS( SELECT product\_name FROM products**

**WHERE products.supplierid=suppliers.supplierid AND price=22);**

# **16. stored procedure!**

• It’s a prepared SQL code that you can save, so code can be reused over and over again.we can also pass parameters to this procedures.

**CREATE PROCEDURE procedure\_name**

**AS**

**Sql\_statement**

**GO;**

**EXEC procedure\_name**

**CREATE PROCEDURE selectallcustomers**

**AS**

**SELECT \* FROM customers**

**GO;**

**EXEC selectallcustomers**

**CREATE PROCEDURE selectallcustomers @city nvarchar(30),@postalcode nvarchar(30)**

**AS**

**SELECT \* FROM customers WHERE city=@city**

**AND postalcode=@postalcode**

**GO;**

**EXEC selectallcustomers @city= ‘London’, @postalcode= ‘yjjkjo’**

# **17. SQL constraints!**

• Used to specify rules for the data in a table.

• NOT NULL : Ensures the column cannot have a null value

• UNIQUE : Ensures all value in column are different

• PRIMARY KEY: Combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

• FOREIGN KEY : Prevent actions that would destroy links between table.

• CHECK : Ensures that the value in a column satisfies a specific condition

• DEFAULT: Sets a default value for a column if no value is specified.

• CREATE INDEX: Used to create and retrieve data from database very quickly.

# **18. SQL joins!**

• Used to combine rows from two or more tables,based on related columns between them

## **• (INNER) JOIN :**

Return records that have matching value in both table**.**

**SELECT \* FROM orders**

**JOIN customers**

**ON orders.customer\_id=customers.customer\_id**

We can simplify table name by giving abbreviations

**SELECT \* FROM orders o**

**JOIN customers c**

**ON o.customer\_id=c.customer\_id**

• Or we can replace same field name using ‘USING’ clause

**SELECT \* FROM orders o**

**JOIN customers c**

**ON USING(customer\_id)**

## **• Self joins**

**SELECT \* FROM employee e**

**JOIN employee m**

**ON e.reports\_to =m.employee\_id**

• Joining across multiple databases

**SELECT \* FROM order\_items oi**

**JOIN sql\_inventory.products p**

**ON oi.product\_id =p.product\_id**

• Joining multiple tables

**USE sql\_store;**

**SELECT \* FROM orders o**

**JOIN customer c**

**ON o.customer\_id =c.customer\_id**

**JOIN order\_statuses os**

**ON o.status=os.order\_status\_id**

• Compound joining conditions

**SELECT \* FROM order\_items oi**

**JOIN order\_item\_notes oin**

**ON oi.order\_id=oin.order\_id**

**AND oi.product\_id=oin.product\_id**

• Implicit join syntax

**SELECT \* FROM orders o**

**JOIN customers c**

**ON o.customer\_id=c.customer\_id**

## **• LEFT (OUTER) JOIN :**

Return all records from the left table and matched record from right table

**SELECT c.customer\_id,c.first\_name,o.orderid**

**FROM customers c**

**LEFT JOIN orders o**

**ON c.customer\_id=c.customer\_id**

## **• RIGHT (OUTER) JOIN :**

Return all records from the right table and matched record from left table

**SELECT c.customer\_id,c.first\_name,o.orderid**

**FROM customers c**

**RIGHT JOIN orders o**

**ON c.customer\_id=c.customer\_id**

## **• FULL (OUTER) JOIN :**

Return all records from the left and right table.

# **19. SQL UNIONS!**

**•** Used to combine result set of two or more SELECT statement. Column name is taken from first select statement

**SELECT city FROM customers**

**UNION**

**SELECT city FROM suppliers**

**ORDER BY city;**

# 

**Implentation of Mysql in python**

# Step by step procedures:

# **1 .Download MySQL**

1. Mysql.com -- > Downloads ..> community downloads..> mysql community server ..> mysqlinstaller for windows ..>Start my download

Work in mysql workbench (ui) and mysql shell

1. Download Xampp control panel and use phpmy admin as UI ( port 3308 default 3306. But here 3306 already taken by mysql.so changed port to 3306)

* Setting up password for root in phpMyAdmin

Click admin, it will open phpMyAdmin. In user\_accounts 🡪 edit privileages in 3 root 🡪 give password 🡪 generate 🡪 Go

In local storage:

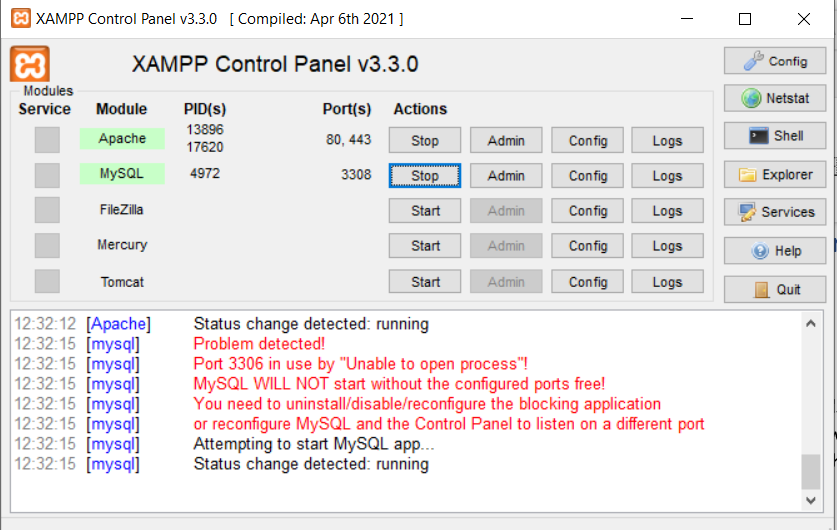
Xampp ..> phpMyAdmin..> config.inc.php

Add password b/w ‘ ’

Xampp ..> mysql ..>bin ..>my.ini

After[mysql d]

Add one more line skip-grant-tables



# **2.Connecting to mysql database in python**

## **Using MySQLdb**

It is an interface for connecting to a MySQL database server from python.

For that we have to install two packages

1. mysql-connector-python (pip install mysql-connector-python)
2. mysql-python (pip install mysql-python)

now import MySQLdb

import MySQLdb

# for phpMyAdmin in 3308 port

conn=MySQLdb.connect(host="127.0.0.1",

user="root",

password="bhagya",

db="bank",

port=3308)

# for workbench in 3306 port

conn=MySQLdb.connect(host="localhost",

user="root",

password="bhagya",

db="bank")

cursor=conn.cursor()

## **Using windows command prompt**

Open the command prompt

#mysql

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p

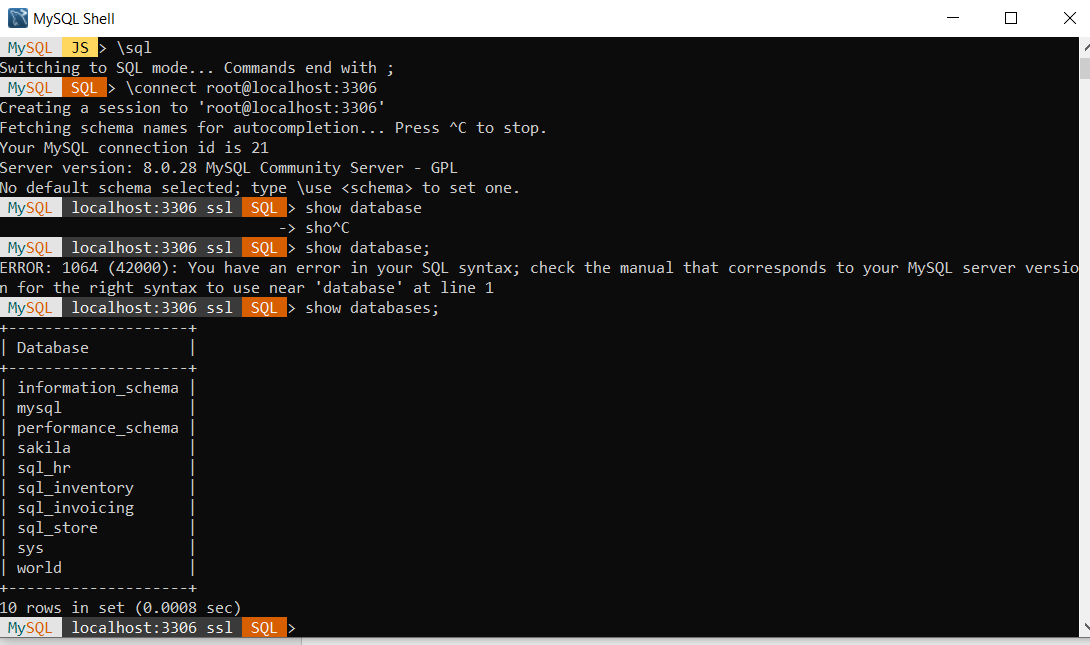
Password :

Show databases;

# xampp

C:\Xampp\mysql\bin>mysql -u root -p -h localhost -P 3308

# **Using MySql shell**



\sql

\connect root@localhost:3306 - mysql

\connect root@localhost:3308 -phpmyadmin

Show databases;

# **3.Create Database**

conn=MySQLdb.connect(

host="127.0.0.1",

user="root",

password="bhagya",

database="bankdb",

port=3308)

cursor=conn.cursor()

cursor.excecute("CREATE DATABASE bankdb")

# **4.Create Table**

cursor.execute("CREATE TABLE users (\

account\_no INT NOT NULL AUTO\_INCREMENT,\

Name VARCHAR(40) NOT NULL,\

Mobile\_no VARCHAR(40) NOT NULL,\

Address VARCHAR(100) NOT NULL,\

Country VARCHAR(100) NOT NULL,\

Email VARCHAR(100) NOT NULL,\

Balance INT DEFAULT 0,\

PRIMARY KEY(account\_no))")

cursor.execute("CREATE TABLE transactions (\

trans\_no INT NOT NULL AUTO\_INCREMENT,\

user\_id INT,\

trans\_type VARCHAR(40) NOT NULL,\

Trans\_date TIMESTAMP DEFAULT current\_timestamp,\

Trans\_amount INT NOT NULL\

Current\_balance INT NOT NULL,\

PRIMARY KEY(trans\_no),\

FOREIGN KEY(user\_id)\

REFERENCES users(account\_no))")

# **5.Insert data in table**

* cursor.execute("INSERT INTO users(Name,Mobile\_no,Address,Country,Email)\

VALUES('bhagya','5566778009','address city','india','bhagya@gmail.com'),\

('bhavya','1166888009','add city','india','bhavya@gmail.com')")

conn.commit()

* records=['bhagya','5566778009','address city','india','bhagya@gmail.com']

cursor.execute("INSERT INTO users(Name,Mobile\_no,Address,Country,Email)\

VALUES(%s,%s,%s,%s,%s)",records)

conn.commit()

* cursor.execute("INSERT INTO transactions(user\_id,trans\_type,Trans\_amount,Current\_balance)\

VALUES({},'{}',{},{})".format(acc\_no,trans\_type,trans\_amt,curr\_balance))

conn.commit()

# **6.Select data from table**

* data=cursor.execute("SELECT \* FROM users ORDER BY {}".format(sort\_by))

data=cursor.fetchall

* data=cursor.execute("SELECT \* FROM users WHERE account\_no={} ".format(search\_by))

data=cursor.fetchall()

# **7.Update data in the table**

* data=cursor.execute("UPDATE users SET {}='{}' WHERE account\_no={}".format(update\_by,set\_inp,acc\_no))

conn.commit()

* cursor.execute("UPDATE users SET Balance={} WHERE account\_no={}".format(curr\_balance,acc\_no))

conn.commit()

# **8.Delete data from the table**

* data=cursor.execute("DELETE FROM users WHERE account\_no= '%s'"%(acc\_no))

conn.commit()