Introduction

**MySQL** is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

It allows us to use tables, rows, columns, and indexes and to perform database operations on them.

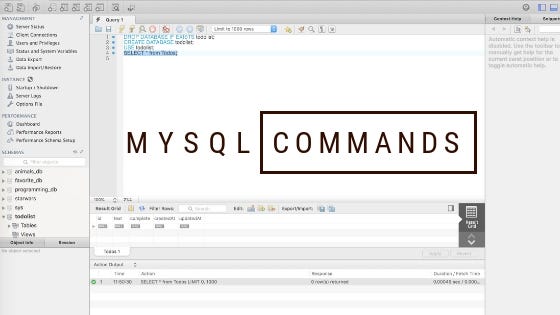
Tables (collection of rows and columns), also known as relations, are used to construct database relationships.

It ensures that the Referential Integrity (referential integrity is an RDBMS concept which states that any foreign key field must agree with the primary key that is referenced by the foreign key) between rows or columns of different tables is maintained.

It automatically updates the table indexes as soon as a record is changed.

It employs a number of SQL queries and integrates useful data from many columns and rows for end-users.

## Some of The Most Important SQL Commands:



**i.SELECT**-statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

SELECT column\_name

FROM table\_name;

### **ii.UPDATE**- updates data in a database.

UPDATE table\_name  
SET some\_column = some\_value  
WHERE some\_column = some\_value;

**iii.DELETE -** deletes data from a database.

DELETE FROM table\_name  
WHERE some\_column = some\_value;

**iv.INSERT INTO -** inserts new data into a database

INSERT INTO table\_name (column\_1, column\_2, column\_3)  
VALUES (value\_1, ‘value\_2’, value\_3);

**v.CREATE DATABASE** - creates a new database.

CREATE DATABASE databasename;

**vi.ALTER DATABASE** - modifies a database.

ALTER DATABASE database\_name  
[COLLATE collation\_name ]

**vii.CREATE TABLE** - creates a new table.

CREATE TABLE table\_name (  
column\_1 datatype,  
column\_2 datatype,column\_3 datatype  
);

**viii.ALTER TABLE** - ALTER TABLE statements add, delete, or modify columns in an existing table.

ALTER TABLE table\_name  
ADD column\_name datatype;

**ix.DROP TABLE** - deletes a table.

DROP TABLE table\_name;

DROP TABLE statements drop an existing table in a database.

**Advanced MySQL** **:**It helps new administrators with manipulating data either through inserting records from other queries or CSV files. CSV files are flat, comma-delimited files that contain data that can be read across numerous different platforms. If you need to transfer data to or from a MySQL database, you can do so without worry of unsupported platforms using CSV files. You also need to know how to replace data or delete records across multiple tables. This article focuses on these advanced concepts.

**MySQL JOINS:** MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.There are three types of MySQL joins:- **i.MySQL INNER JOIN (or sometimes called simple join),ii.MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN),iii.MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)**

## MySQL Inner JOIN (Simple Join)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

**Syntax:**

**SELECT** columns

**FROM** table1

INNER JOIN table2

**ON** table1.**column** = table2.**column**;

**MySQL Left Outer Join**

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

**Syntax:**

**SELECT** columns

**FROM** table1

LEFT [OUTER] JOIN table2

**ON** table1.**column** = table2.**column**;

**MySQL Right Outer Join**

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

**Syntax:**

**SELECT** columns

**FROM** table1

RIGHT [OUTER] JOIN table2

**ON** table1.**column** = table2.**column**;

**SQL UNION Operator:** The UNION operator is used to combine the result-set of two or more SELECT statements.

**UNION Syntax**

SELECT column\_name(s) FROM table1UNION  
SELECT column\_name(s) FROM table2*;*

### **UNION ALL Syntax:-**The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

SELECT column\_name(s) FROM table1  
UNION ALL  
SELECT column\_name(s) FROM table2*;*

**MySQL workbench- Modeling and Design Tool**

MySQL Workbench is a unified visual database designing or graphical user interface tool used for working with database architects, developers, and Database Administrators. It is developed and maintained by Oracle. It provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. We can use this Server Administration for creating new physical data models, E-R diagrams, and for SQL development (run queries, etc.). It is available for all major operating systems like Mac OS, Windows, and Linux.

**Five main functionalities:- i.SQL Development,ii.Data Modelling (Design),iii.** **Server Administration,iv.Data Migration,v.** **MySQL Enterprise Supports.**

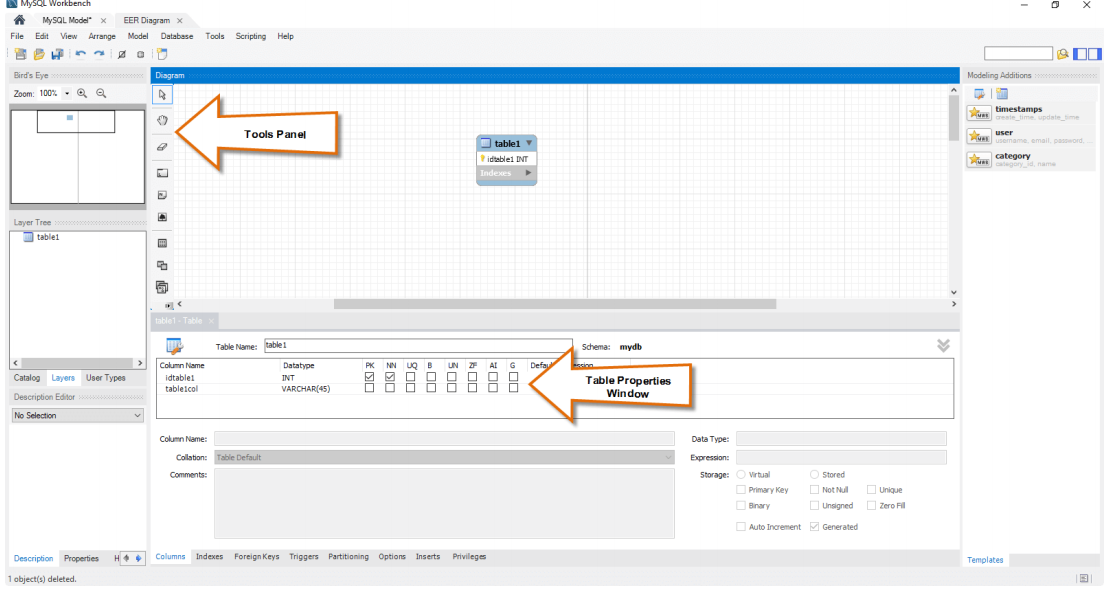
## MySQL Workbench Editions

MySQL Workbench is mainly available in three editions**-i.Community Edition (Open Source, GPL)ii.Standard Edition (Commercial)iii.Enterprise Edition (Commercial).**

**MySQL workbench – SQL development tool**

Structured Query Language (SQL) allows us to manipulate our relational databases. SQL is at the heart of all relational databases.

**The figure shown below shows the modeling window for MySQLWorkbench**



**How To Use MySQL Workbench – Administration tool :**

Server administration plays a critical role in securing the data of the company. The major issues concerning server administration are users’ management, server configuration, server logs and many more. Workbench MySQL has the following features that simplify the process of MySQL server administration;

**i.User administration** – visual utility for managing users that lets database administrators easily add new and remove existing users if need arises, grant and drop privileges and view user profiles.

**ii.Server configuration** – allows for advanced configuration of the server and fine tuning for optimal performance.

**iii.Database backup and restorations** – visual tool for exporting/importing MySQL dump files. MySQL dump files contain SQL scripts for creating databases, tables, views, stored procedures and insertion of data.

**iv.Server logs** – visual tool for viewing MySQL server logs. The logs include error logs, binary logs and InnodDB logs. These logs come in handy when performing diagnosis on the server. The figure shown below shows the modeling window for MySQL Workbench.

**WEEK-1**

**How to Install MySQL workbench (for Windows) :**

Install MySQL workbench is a 2 step process.

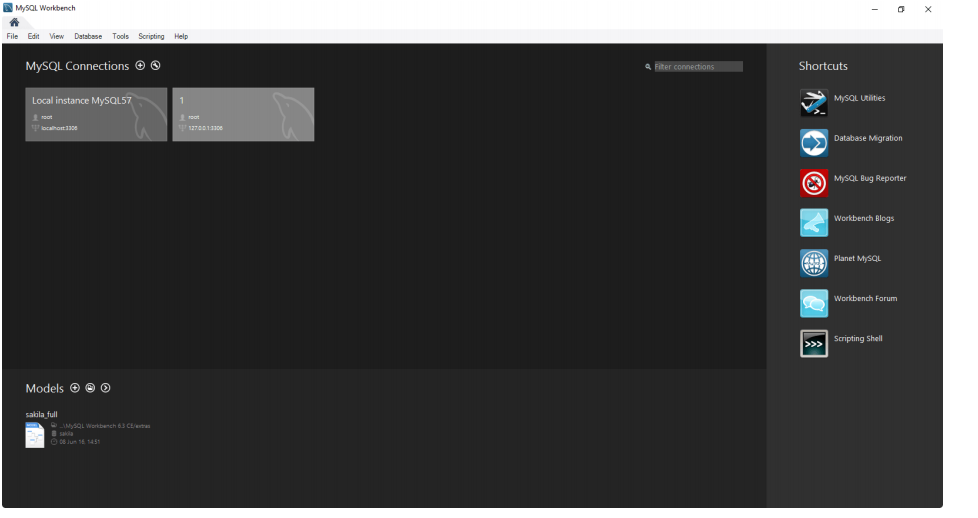
1) Install MySQL Community Server

2) Install MySQL workbench

After finished installing above you need to set up MySQL Workbench as shown below Following is a step by step process on How to Install MySQL workbench.

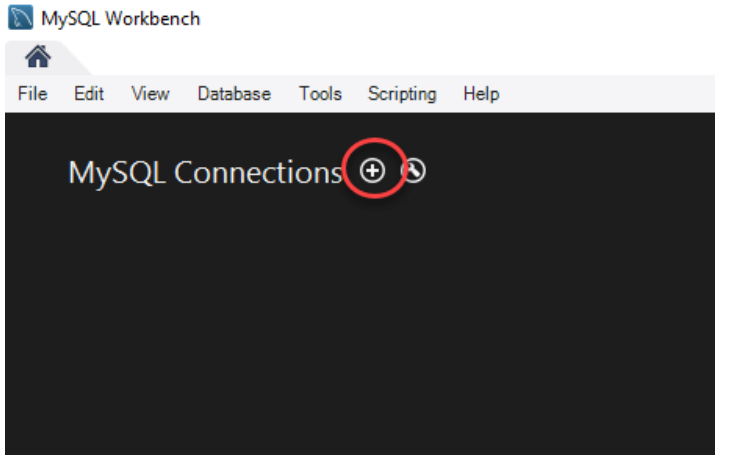
**Step 1)** Open Home Window

First step is launching the Workbench MySQL.It is called **Home Window.**



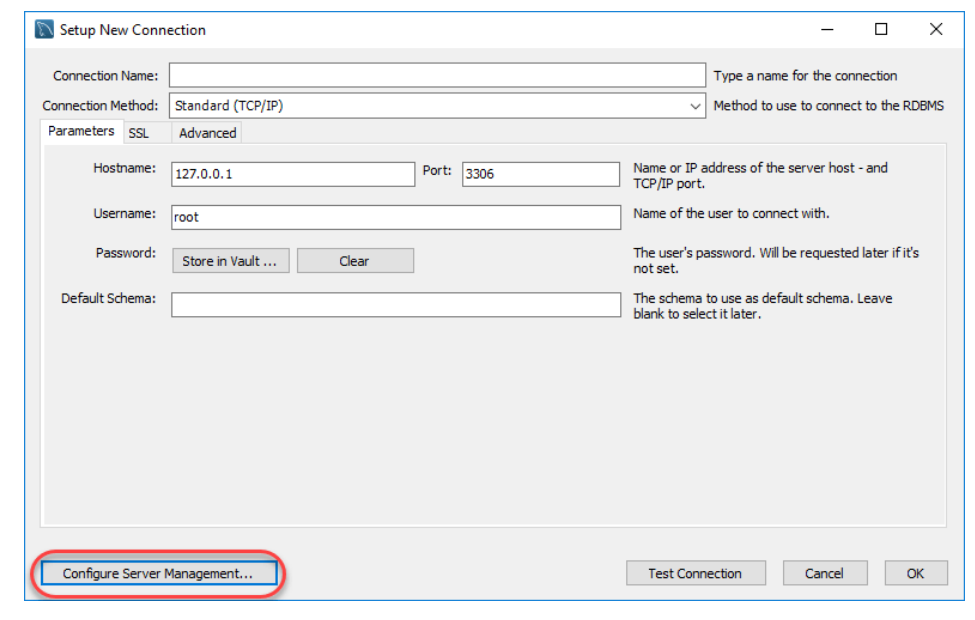
**Step 2)** Open New Connection Wizard

Next need to create MySQL Server Connection which contains details about target database server including how to connect to it. Click **” + “** in MySQL Workbench Home Window. This will open **Setup New Connection**.



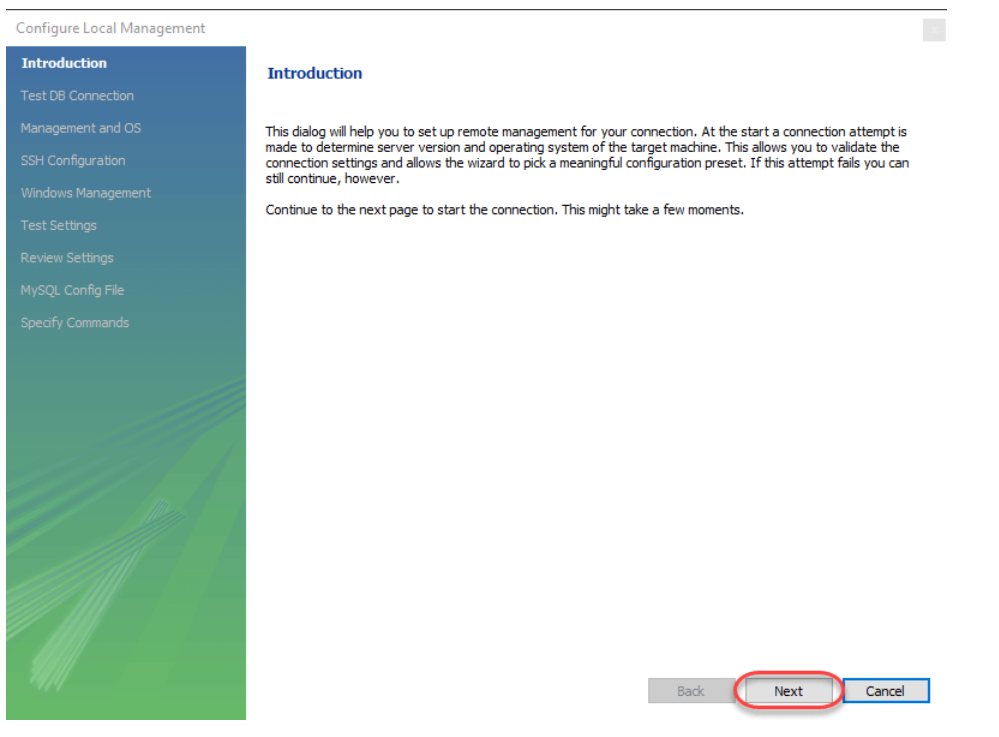
**Step 3)** Click Configure Server Management button

As a beginner you can create a connection for a locally installed server. Click **Configure Server Management** button in **Setup New Connection** window to check the cofiguration of the MySQL server.



**Step 4)** Click Next button to continue

A new window opens named **Configure Local Management**. Click Next button to continue.

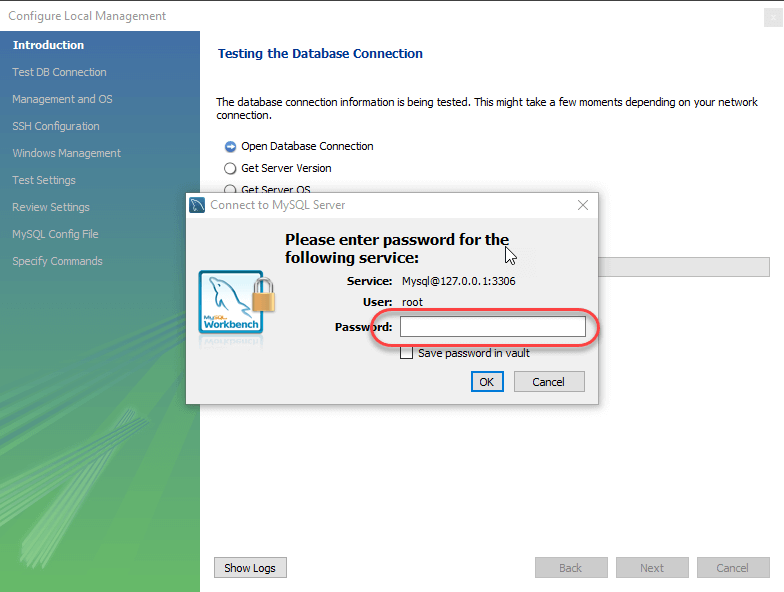


**Step 5)** Enter your password and press OK

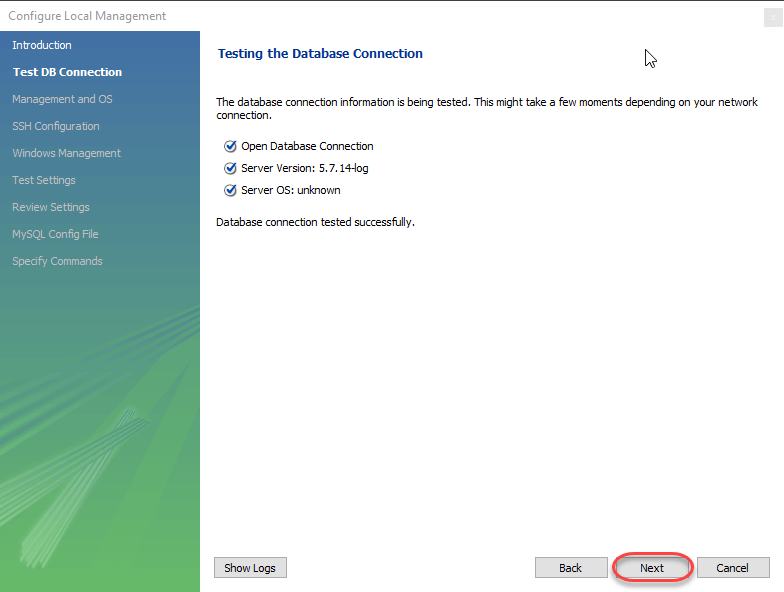
Next the Wizard will test connections to database. If test fails, go back and correct database connection parameters.5.

Next it will open a pop up window asking your root password to test your connection with the local mysql server

instance. The password is the one you set during installation of MySQL Workbench. Enter your password and press **OK.**

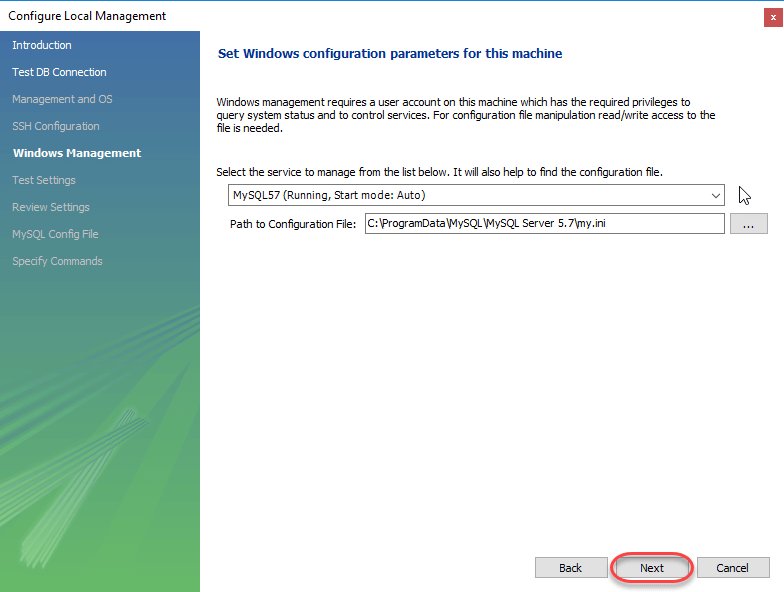


Step 6) Click Next to continue Next the Wizard will test connections to database. If test fails, go back and correct database connection parameters. Else if all tests are sucessful click Next to continue.



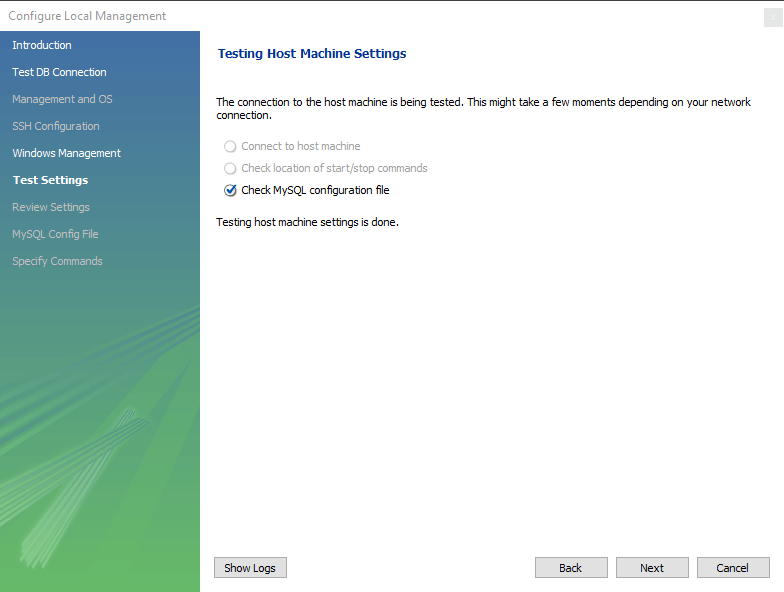
**Step 7)** Click Next

After that a new wizard will open about Local Service Management – It lets you switch between multiple mysql severs installed on one machines. As a beginner you can bypass this and click **Next** to continue.

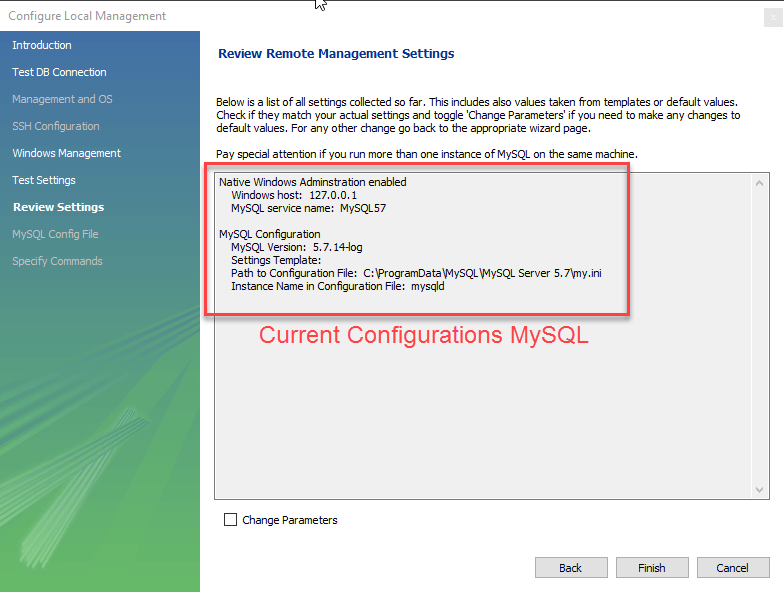


**Step 8)** Select MySQL Server Configuration File

The Wizard will then check ability to access MySQL Server Configuration File, and test start/stop commands.



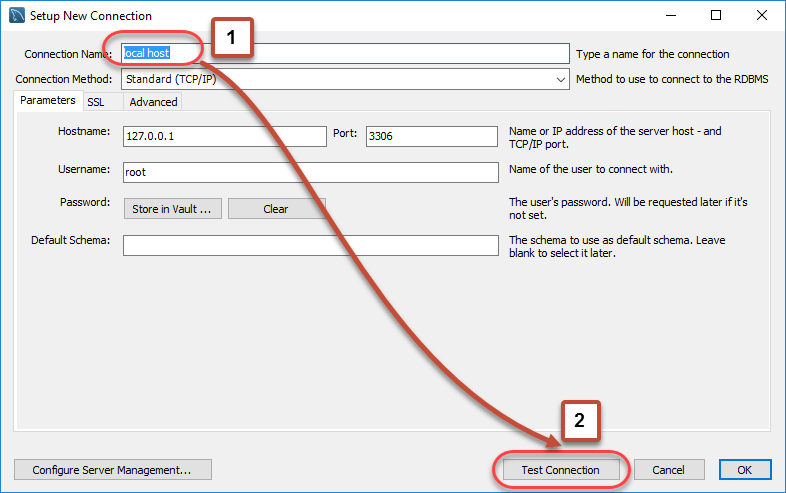
**Step 9)** Click Finish to finsh server cofiguration Next you can review current configurations. After reviewing the configurations, Click Finish to finsh server configuration.



**10)** Click on Test Connection

Next **Step** Step is to setup a connection, which can be used to connect to server. If you have not created a connection already,

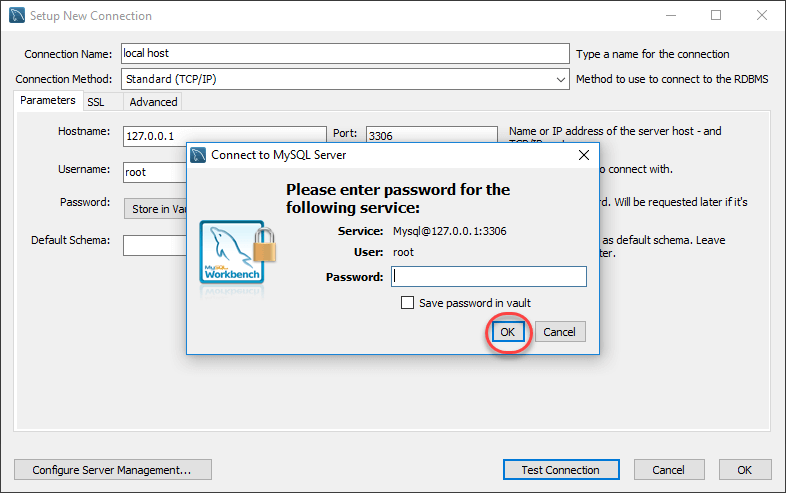
you can use the default values given. Click on Test Connection [ 2 ] after entering the Connection Name [ 1 ].



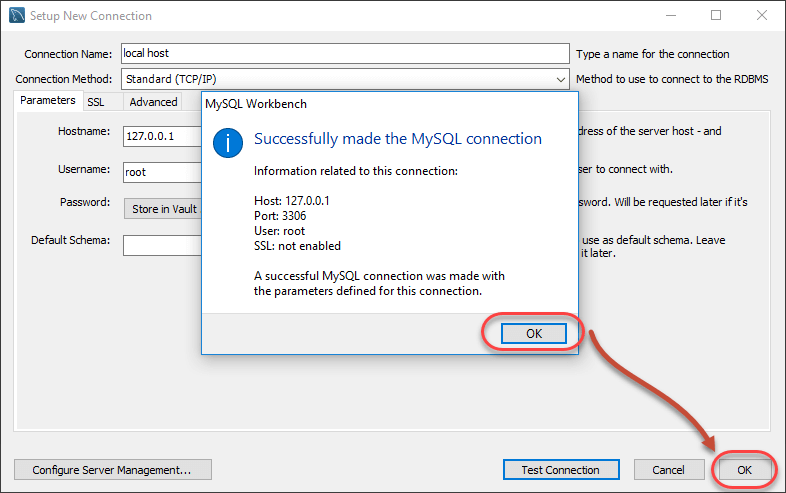
**Step 11)** Click OK

A new dialog box will open asking you password to root/selected user. If your MySQL root user has a password, you can

enter that using Store in Vault feature. Click OK.



If the entered password for the user is correct then the following screen will show. Click on **both OK** buttons and you will be good to go.

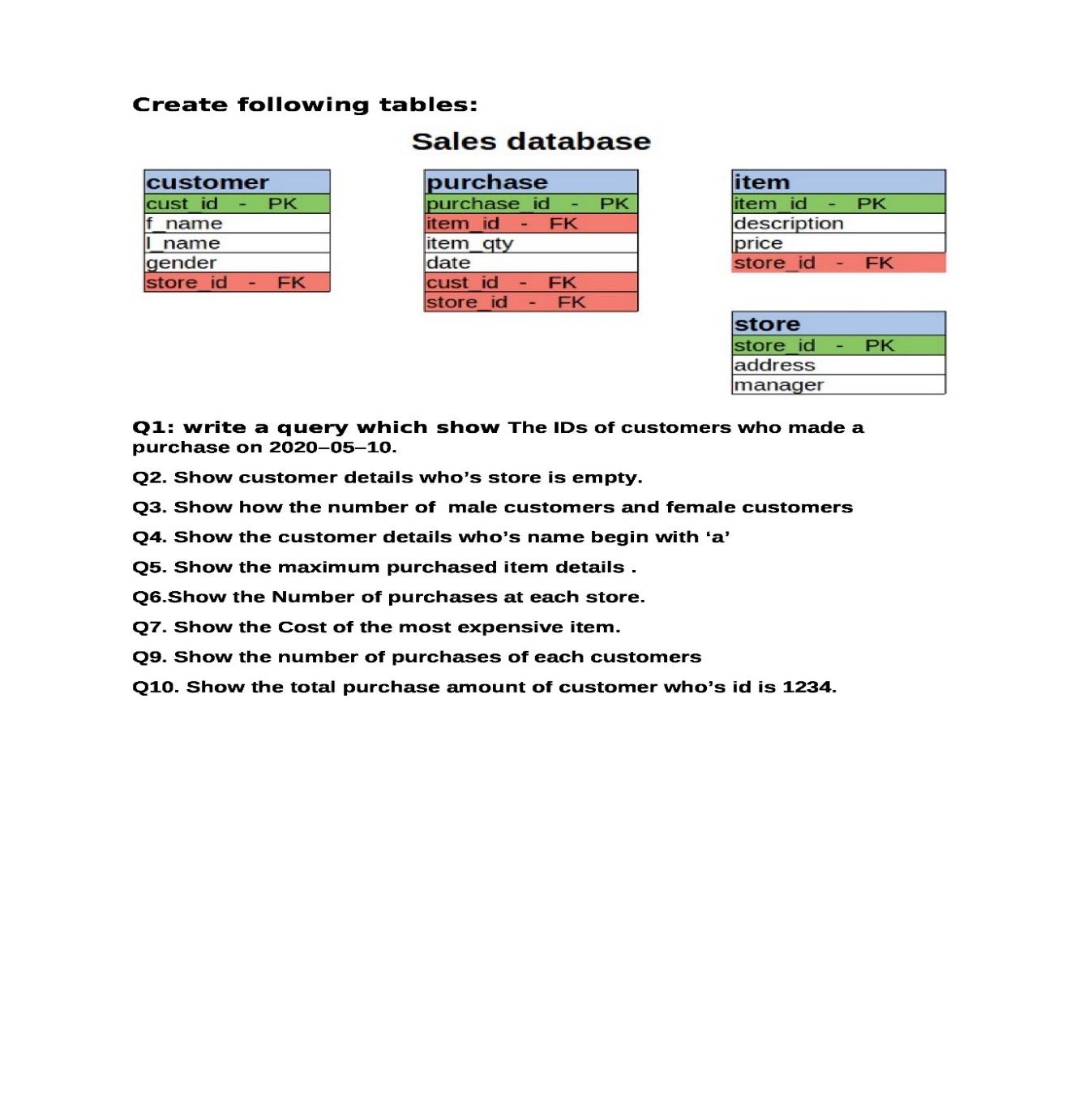


A new instance is shown in the homepage.



Double click and start querying.

**Hands On Sql**



**Creating Sales Database**

create database Sales;

**Creating store Table**

create table store (store\_id varchar(5), address varchar(100), manager varchar(50));

**Insertion in store Table**

insert into store values ("s01","Kolkata","Darshan Sahani");

insert into store values ("s02","Baruipur","Jibon Das");

insert into store values ("s03","Katwa","Alo Dey");

insert into store values ("s04","Malda","Prakriti Mondal");

insert into store values ("s05","Jalpaiguri","Dristi Bose");

**Creating customer Table**

create table customer(cust\_id varchar(5) primary key, fname varchar(50), lname varchar(50), gender varchar(10), store\_id varchar(5) references store);

**Insertion in customer Table**

insert into customer values("c01","Amit","Das","Male","s01");

insert into customer values("c02","Kamal","Pal","Male","s02");

insert into customer values("c03","Argha","Saha","Male","s03");

insert into customer values("c04","Rani","Kayal","Female","s04");

insert into customer values("c05","Sohini","Saha","Female","s05");

**Creating item Table**

create table item (item\_id varchar(5) primary key, description varchar(50), price int, store\_id varchar(5) references store);

**Insertion in item Table**

insert into item values ("i01", "Perfumes",25000,"s01")

insert into item values ("i02", "Laptops",40000,"s02");

insert into item values ("i03", "Cricket Kit",20000,"s03");

insert into item values ("i04", "Electric Scootie",15000,"s04");

insert into item values ("i05", "Makeup Kit",35000,"s05");

**Creating purchase Table**

create table purchase (purchase\_id varchar(5) primary key, item\_id varchar(5) references item, item\_qty int, date date, cust\_id varchar(5) references customer, store\_id varchar(5) references store);

**Insertion in purchase Table**

insert into purchase values ("p01", "i01", 10, "2019-10-05","c01","s01");

insert into purchase values ("p02", "i02", 12, 2020-01-09","c02","s02");

insert into purchase values ("p03", "i03", 19,"2020-12-30","c03","s03");

insert into purchase values ("p04", "i04", 32,"2021-08-30","c04","s04");

insert into purchase values ("p05", "i05", 23, "2022-11-20","c05","s05");

**:: Queries ::**

1. select cust\_id from customer where cust\_id in (select cust\_id from purchase where to\_char(date, ‘yyyy-mm-dd’) = '2020-05-10');

1. select \* from customer where cust\_id in (select cust\_id from purchase where item\_qty=0);

* 1. select count(gender) "No. of Male Customers" from customer where gender='Male';
  2. select count(gender) "No. of Female Customers" from customer where gender='Female';

1. select \* from customer where f\_name like 'a%';

1. select \* from item where item\_id in (select item\_id from purchase where item\_qty in (select max(item\_qty) from purchase));

1. select store\_id, item\_qty from purchase;

1. select max(p[rice) from item;

1. select cust\_id, item\_qty from purchase;

1. select price from item where item\_id in (select item\_id from purchase where cust\_id='c01');

**WEEK-2**

**Create Schema and get required information from it through Queries:**

-- MySQL dump 10.13 Distrib 8.0.32, for Win64 (x86\_64)

--

-- Host: localhost Database: sarn

-- ------------------------------------------------------

-- Server version 8.0.32

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `class`

--

DROP TABLE IF EXISTS `class`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `class` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(1000) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `class`

--

LOCK TABLES `class` WRITE;

/\*!40000 ALTER TABLE `class` DISABLE KEYS \*/;

INSERT INTO `class` VALUES (1,'1st'),(2,'2nd'),(3,'3rd'),(4,'4th');

/\*!40000 ALTER TABLE `class` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `student`

--

DROP TABLE IF EXISTS `student`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `student` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(1000) DEFAULT NULL,

`address` varchar(1000) DEFAULT NULL,

`class` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY ` classIdClass\_idx` (`class`),

CONSTRAINT ` classIdClass` FOREIGN KEY (`class`) REFERENCES `class` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `student`

--

LOCK TABLES `student` WRITE;

/\*!40000 ALTER TABLE `student` DISABLE KEYS \*/;

INSERT INTO `student` VALUES (1,'Rana Das','Bihar',3),(2,'Amit Saha','Birbhum',1),(3,'Aniket Ghosh','Bankura',2),(4,'Sanjay Mondal','Bardhaman',4),(5,'Rahul Barman','Malda',2),(6,'Sarno Maity','Kolkata',4);

/\*!40000 ALTER TABLE `student` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `studentclass`

--

DROP TABLE IF EXISTS `studentclass`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `studentclass` (

`StudentID` int NOT NULL DEFAULT '0',

`StudentName` varchar(1000) DEFAULT NULL,

`Address` varchar(1000) DEFAULT NULL,

`Class` int DEFAULT NULL,

`ClassID` int NOT NULL DEFAULT '0',

`ClassName` varchar(1000) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `studentclass`

--

LOCK TABLES `studentclass` WRITE;

/\*!40000 ALTER TABLE `studentclass` DISABLE KEYS \*/;

INSERT INTO `studentclass` VALUES (2,'Amit Saha','Birbhum',1,1,'1st'),(3,'Aniket Ghosh','Bankura',2,2,'2nd'),(5,'Rahul Barman','Malda',2,2,'2nd'),(1,'Rana Das','Bihar',3,3,'3rd'),(4,'Sanjay Mondal','Bardhaman',4,4,'4th'),(6,'Sarno Maity','Kolkata',4,4,'4th');

/\*!40000 ALTER TABLE `studentclass` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2023-03-02 21:55:27

**:: Queries ::**

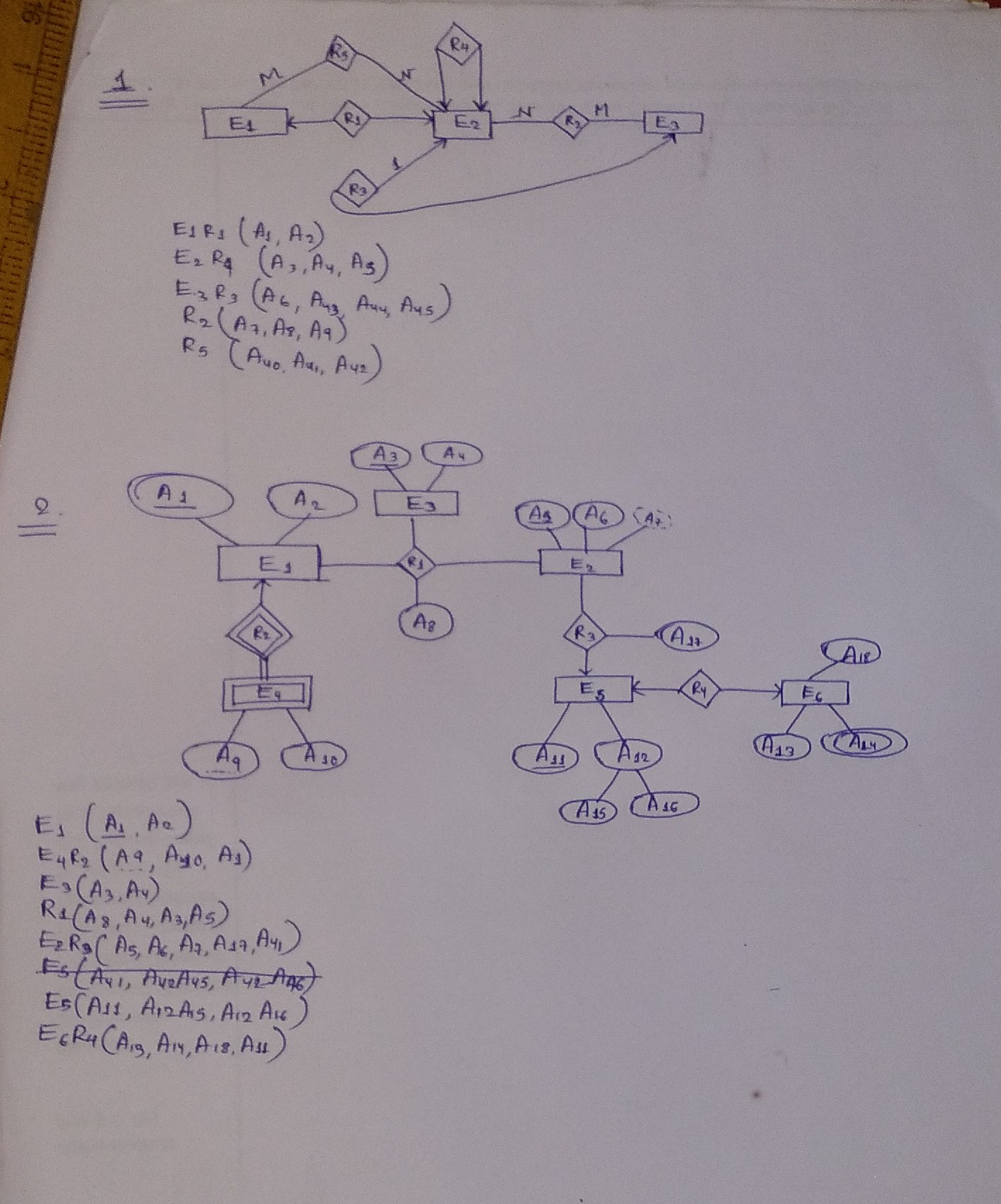
use sneh;

* + - * 1. select \* from student, class where student.class = class.id;
        2. select \* from student as s inner join class as c on s.class = c.id;
        3. select \* from student as s left join class as c on s.class = c.id;
        4. select \* from student as s right join class as c on s.class = c.id;

1. create table StudentClass select s.id as StudentID, s.name as StudentName, Address, Class, c.id as ClassID, c.name as ClassName from student as s inner join class as c on s.class = c.id;
   * + - 1. select \* from studentclass as s1 inner join student as s2 on s1.StudentID = s2.id;
         2. update studentclass as s1 inner join student as s2 on s1.StudentID = s2.id set s1.class = s2.class;
         3. update studentclass as s1 inner join student as s2 on s1.StudentID = s2.id set s1.class = s2.class where s1.StudentID in (1,2);

**WEEK-3**

**Conversion of ER Diagram to Table:**



**Get required information from a given Schema through Queries:**

Given Schema = “sjsms”

use sjsms;

1. select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

inner join clas as c on c.id = cw.clasId

where se.name like '2020-2021' and c.name not like 'class 1';

1. select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

where se.name like '2020-2021' and cw.clasId in (select id from clas where name not like '%1' and id != 1);

1. select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

inner join clas as c on c.id = cw.clasId

where se.name like '2020-2021' and

not exists (select id, name from clas where name not like '%1' and id != 1 and c.id = id and c.name = name);

1. select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

inner join clas as c on c.id = cw.clasId

where se.name like '2020-2021' and (c.id, c.name) in (select id, name from clas where name not like '%1');

1. select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

inner join clas as c on c.id = cw.clasId

where se.name like '2020-2021' and (c.id, c.name) in (select id, name from clas where name like '%1')

union all select \* from student as s

inner join studentdetels as sd on s.studentDetelsId = sd.id

inner join session as se on s.sessionId = se.id

inner join cwst cw on s.id = cw.studentId

inner join clas as c on c.id = cw.clasId

where se.name like '2020-2021' and (c.id, c.name) in (select id, name from clas where name like '%2');

**WEEK-4**

**Create stored procedures and retrieve data from it**

# **Syntax of Procedure**

CREATE PROCEDURE `new\_procedure` ((in, out, inout) parameter\_name

data\_type, (in, out, inout) parameter\_name data\_type) BEGIN

declaration section executable section

END

# **Syntax of Cursor**

CREATE PROCEDURE `cursorsyntax` (num1 int, num2 int) BEGIN

declare curser\_name cursor for select id, name from table\_name; declare num1 int;

declare num2 int; open cursor\_name;

fetch cursor\_name into num1, num2;

close cursor\_name;

END

# **Examples:-**

CREATE DEFINER=`root`@`localhost` PROCEDURE `p1`(in a int, out b int, inout c int)

BEGIN

set b := a; set c := a\*c; #select a;

END

set @num2 = 6;

call p1(5, @num, @num2); select @num;

select @num2;

CREATE DEFINER=`root`@`localhost` PROCEDURE `studentcountshow10`(out count int)

BEGIN

select \* from studentdetels limit 10;

select count(\*) as totalstudent into count from studentdetels; update student set id = count where id = 5;

END

call studentcountshow10 (@num1); select @num1;

CREATE DEFINER=`root`@`localhost` PROCEDURE `ifelse`(num1 int, num2 int) BEGIN

declare cnt int;

/\*if num1 > num2 then

select "No.1 is Greater";

else

select "No.2 is Greater"; end if;\*/

select count(\*) into cnt from studentdetels; if cnt > 80 then

select "Sufficient Students"; elseif cnt > 60 then

select "Moderate Students";

elseif cnt > 40 then

select "Average Students";

else

select"Less Students";

end if;

END

call ifelse(5, 9);

CREATE DEFINER=`root`@`localhost` PROCEDURE `loopexample`(count int) BEGIN

declare i int; set i=1;

loop1:loop

select i;

set i = i+1;

if i > count then

leave loop1;

end if; end loop;

END

call loopexample(10);

CREATE DEFINER=`root`@`localhost` PROCEDURE `cursorexample`() BEGIN

declare idv int;

declare namev varchar(100); declare done int default 0;

declare studentcursor cursor for select id, name from studentdetels

limit 10;

declare continue handler for not found set done = 1;

open studentcursor; loop1:loop

fetch studentcursor into idv, namev; if done = 1 then

leave loop1;

end if;

select idv, namev; end loop loop1;

close studentcursor;

END

call cursorexample();

CREATE DEFINER=`root`@`localhost` PROCEDURE `prepareexample`(vlimit int)

BEGIN

set @query = concat('select \* from studentdetels limit', vlimit); prepare stmt from @query;

execute stmt;

END

call prepareexample(10);