Samriddhi College

Lokanthali-1, Bhaktapur

B.Sc. CSIT Fourth Semester

2077 Batch

Lab Report

on

Database Management System (DBMS)

Submitted by

Nirdeshika Chuhan

Symbol No.: **26808**

TU Regd.: **5-2-1113-17-2020**

JUNE, 2023

Index

Lab 1

Basic Introduction to SQL

SQL, which stands for Structured Query Language, is a programming language designed for managing and manipulating relational databases. It provides a standard way to interact with databases and perform operations such as querying, updating, and managing data.

Here are some key concepts and components of SQL:

Database:

A database is an organized collection of data stored and accessed electronically. It consists of tables that hold related data.

Table:

A table is a structured representation of data in a database. It consists of rows and columns. Each column represents a specific attribute or field, while each row represents a record or data entry.

Query:

A query is a request for data from a database. SQL allows you to write queries to retrieve specific information from one or more tables using keywords like SELECT, FROM, WHERE, and more.

WHAT CAN SQL DO?

- SQL can **execute queries** against a database.
- SQL can **retrieve data** from a database.
- SQL can **insert records** in a database.
- SQL can **update records** in a database.
- SQL can **delete records** from a database.
- SQL can **create** new **databases**.
- SQL can **create** new **tables** in a **database**.
- SQL can **create** stored **procedures** in a **database**.
- SQL can **create views** in a **database**.
- SQL can **set permissions** on tables, procedures, and views.

CLASSIFICATION OF SQL STATEMENTS:

SQL commands can be mainly divided into following categories:

1.Data Definition Language (DDL) Commands:

Commands that allow you to perform task, related to data definition

For e.g:

- Creating, altering and dropping.
- Granting and revoking privileges and roles.
- Maintenance commands.

2.Data Manipulation Language (DML) Commands:

Commands that allow you to perform data manipulation

For e.g: retrieval, insertion, deletion and modification of data stored in a database.

3.Transaction Control Language(TCL) Commands:

Commands that allow you to manage and control the transactions

For e.g:

- Making changes to database, permanent
- Undoing changes to database, permanent
- Creating savepoints
- Setting properties for current transactions

4.Data Control Language (DCL) Commands:

Commands that are used to manage access rights and permissions within the database.

Key DCL statements include:

- **GRANT:** Gives specific privileges to database users.
- **REVOKE:** Removes specific privileges from database users.

DATABASE COMMANDS:

1. VIEW EXISTING DATABASE:

To view existing database names, the command is: SHOW DATABASES;

2. CREATING DATABASE IN MYSQL:

For creating the database in MySQL, we write the following command:

CREATE DATABASE <databasename>;

For e.g:

In order to create a database Student, command is: CREATE DATABASE Student;

3. ACCESSING DATABASE:

For accessing already existing database, we write:

USE <databasename>;

For e.g:

In order to access a database named Student, we write command as: USE Student;

4. DELETING DATABASE:

For deleting any existing database, the command is:

DROP DATABASE <databasename>;

For e.g:

In order to delete a database, say student, we write command as:

DROP DATABASE Student;

5. VIEWING TABLE IN DATABASE:

In order to view tables present in currently accessed database, command is:

SHOW TABLES;

CREATING TABLES IN MYSQL

Tables are created with the CREATE TABLE command. When a table is created, its columns are named, datatypes and sizes are supplied for each column.

Syntax of CREATE TABLE command is:

```
CREATE TABLE <table-name>(
<column name> <data type> ,
<column name> <data type> ,
 ......... ) ;
```

For e.g:

In order to create table EMPLOYEE given below:

ECODE	ENAME	GENDER	GRADE	GROSS

We write the following command:

```
CREATE TABLE EMPLOYEE
(
ECODE integer,
ENAME varchar (20),
 GENDER char (1),
GRADE char (2),
GROSS integer
);
```

INSERTING DATA INTO TABLE:

The rows are added to relations (table) using INSERT command of SQL.

Syntax of INSERT is:

```
INSERT INTO [table-name] VALUE ( , , …..);
```

For e.g:

In order to enter a row into EMPLOYEE table (created above), we write command as:

```
INSERT INTO EMPLOYEE VALUES (1001, ‘Ravi’, ‘M’, ‘E4’, 50000);
```

OR

```
INSERT INTO employee (ECODE, ENAME, GENDER, GRADE, GROSS)
VALUES (1001, ‘Ravi’, ‘M’, ‘E4’, 50000);
```

Database Normalization:

Database normalization is a technique used to organize data efficiently and eliminate redundancy. It involves splitting a large table into smaller, related tables and establishing relationships between them. Normalization helps improve data integrity and reduces data redundancy, making the database more efficient and easier to maintain.

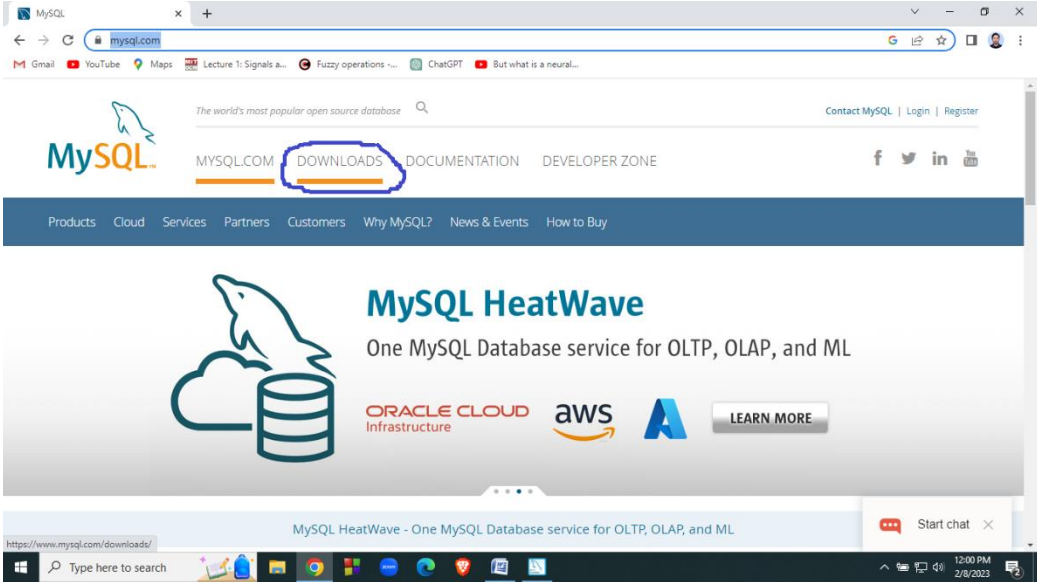
Lab 2

Installation of MySQL Community Edition (GPL) on Windows

Steps for downloading MySQL Community Edition:

Step 1. Go to https://www.mysql.com/

Step 2. Click on Downloads

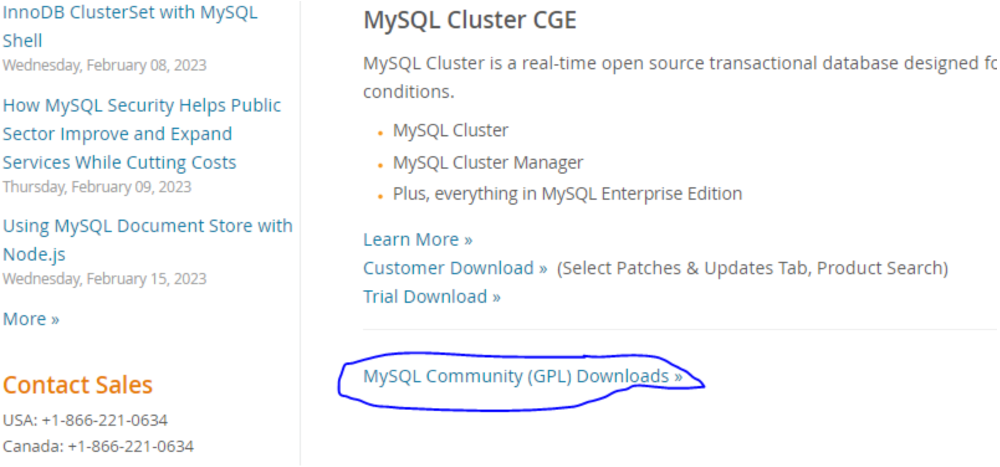


MySQL HeatWave

One MySQL Database service for OLTP, OLAP, and ML

MySQL HeatWave - One MySQL Database service for OLTP, OLAP, and ML

Step 3. After clicking on Downloads, scroll the webpage and locate the link for MySQL Community (GPL) Downloads



InnoDB ClusterSet with MySQL Shell
Wednesday, February 08, 2023

How MySQL Security Helps Public Sector Improve and Expand Services While Cutting Costs
Thursday, February 09, 2023

Using MySQL Document Store with Node.js
Wednesday, February 15, 2023

More »

Contact Sales

USA: +1-866-221-0634
Canada: +1-866-221-0634

MySQL Cluster CGE

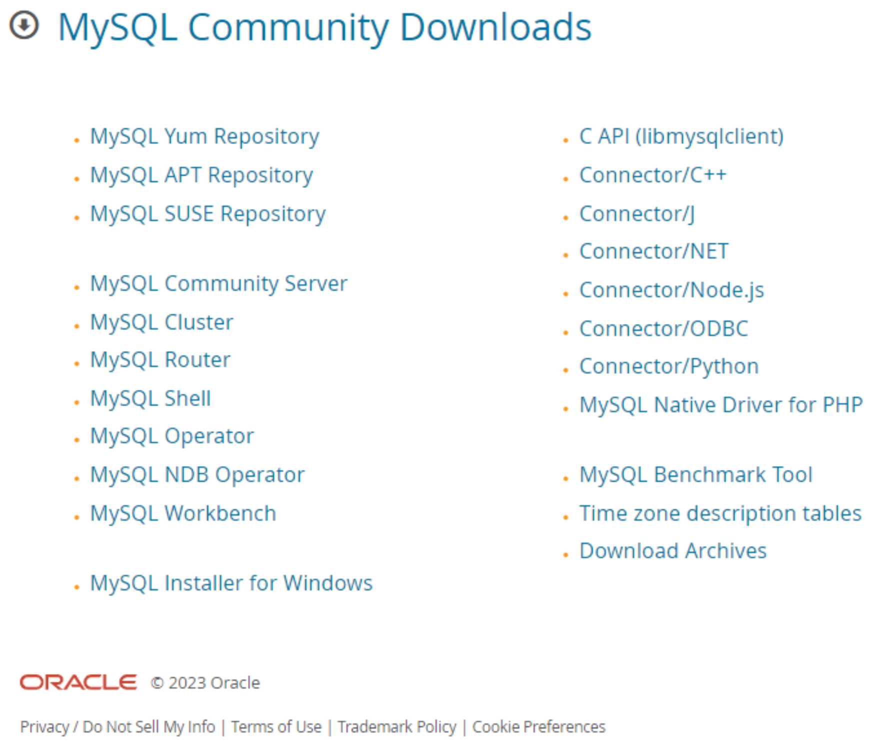
MySQL Cluster is a real-time open source transactional database designed fc conditions.

- MySQL Cluster
- MySQL Cluster Manager
- Plus, everything in MySQL Enterprise Edition

Learn More »
Customer Download » (Select Patches & Updates Tab, Product Search)
Trial Download »

MySQL Community (GPL) Downloads »

Step 4. Upon click on the aforementioned link, following list of downloads will be shown:



MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench

- MySQL Installer for Windows

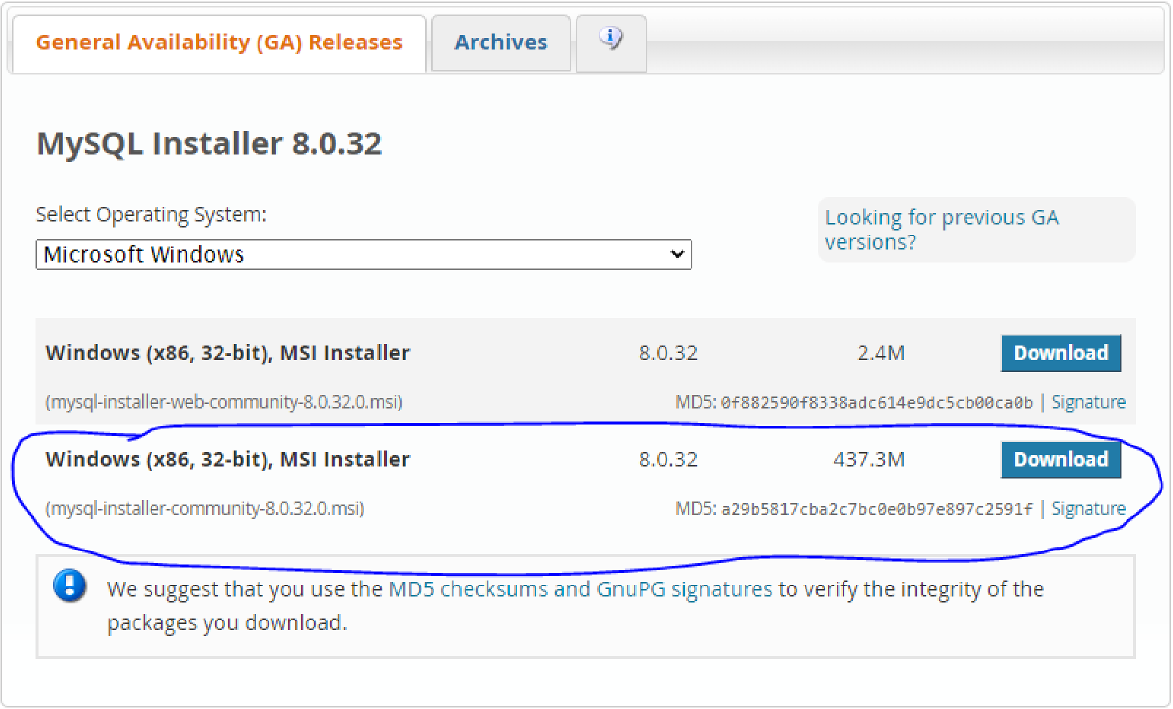
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

ORACLE © 2023 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie Preferences

Step 5. Click on MySQL Installer for Windows. Following download links will be displayed:



MySQL Installer 8.0.32

Select Operating System:
Microsoft Windows

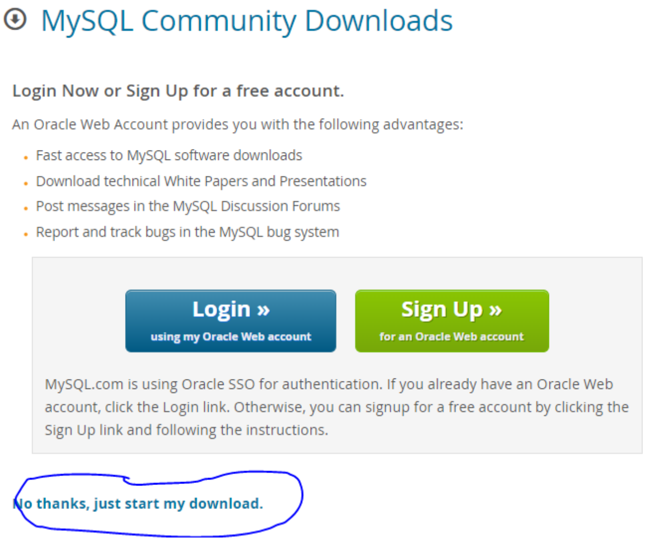
Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer	8.0.32	2.4M	Download
(mysql-installer-web-community-8.0.32.0.msi)	MD5: 0f882590f8338adc614e9dc5cb00ca0b | Signature		
Windows (x86, 32-bit), MSI Installer	8.0.32	437.3M	Download
(mysql-installer-community-8.0.32.0.msi)	MD5: a29b5817cba2c7bc0e0b97e897c2591f | Signature		

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

Step 6. Click on Windows(x86, 32bit), MSI Installer for offline installation.

Step 7. Before starting download, the web page suggests you to login or signup for oracle web account. If you like, you can open one. However, if you want to download directly, click on **No thanks, just start my download** as shown in the following image.



MySQL Community Downloads

Login Now or Sign Up for a free account.

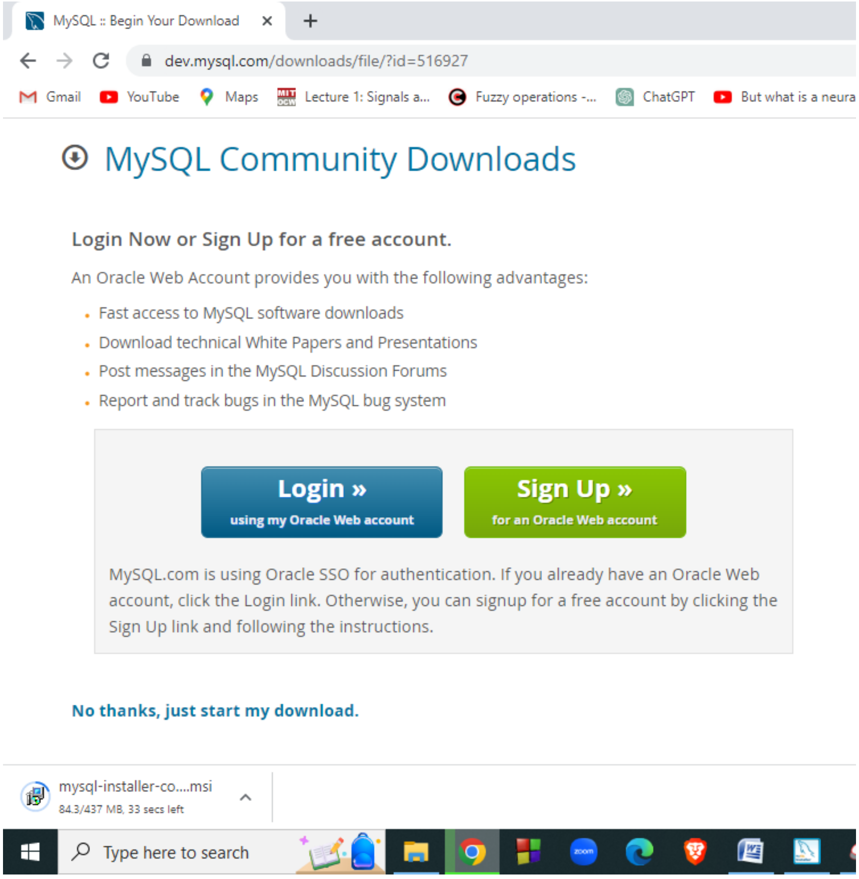
An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

Step 8. Now, the browser will start the download of installer file.



MySQL :: Begin Your Download
dev.mysql.com/downloads/file/?id=516927
Gmail
YouTube
Maps
Lecture 1: Signals a...
Fuzzy operations -...
ChatGPT
But what is a neura

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account
Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

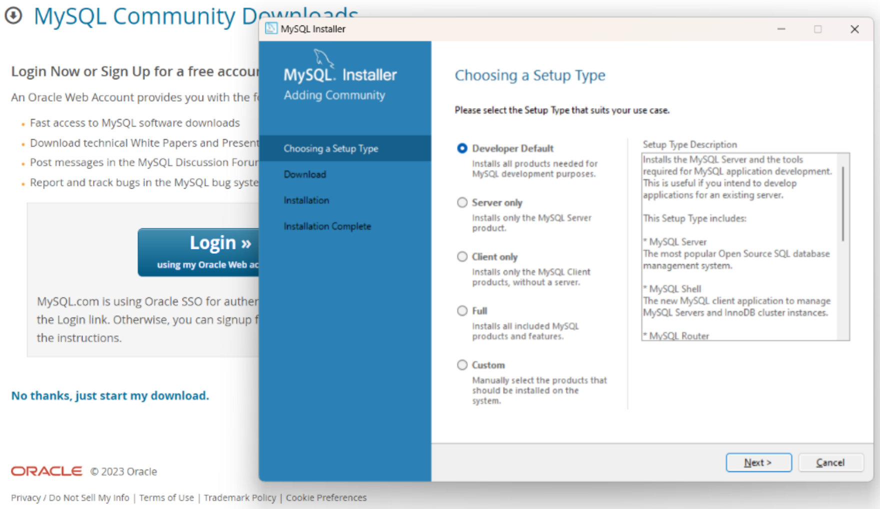
mysql-installer-co....msi
84.3/437 MB, 33 secs left

This completes the download procedure.

Steps for installing MySQL Community Edition:

Step 1. Double click on the downloaded MSI installer file for MySQL community edition.

Step 2. Choose the "Developer Default" option, the installer will automatically select the necessary components for a basic MySQL installation. Click next to proceed.



MySQL Community Downloads

Login Now or Sign Up for a free accou

An Oracle Web Account provides you with the f

- Fast access to MySQL software downloads
- Download technical White Papers and Present
- Post messages in the MySQL Discussion Forur
- Report and track bugs in the MySQL bug syste

MySQL.com is using Oracle SSO for auther the Login link. Otherwise, you can signup f the instructions.

No thanks, just start my download.

ORACLE © 2023 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie Preferences

MySQL Installer

Choosing a Setup Type

Please select the Setup Type that suits your use case.

● Developer Default
Installs all products needed for MySQL development purposes.

○ Server only
Installs only the MySQL Server product.

○ Client only
Installs only the MySQL Client products, without a server.

○ Full
Installs all included MySQL products and features.

○ Custom
Manually select the products that should be installed on the system.

Setup Type Description

Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.

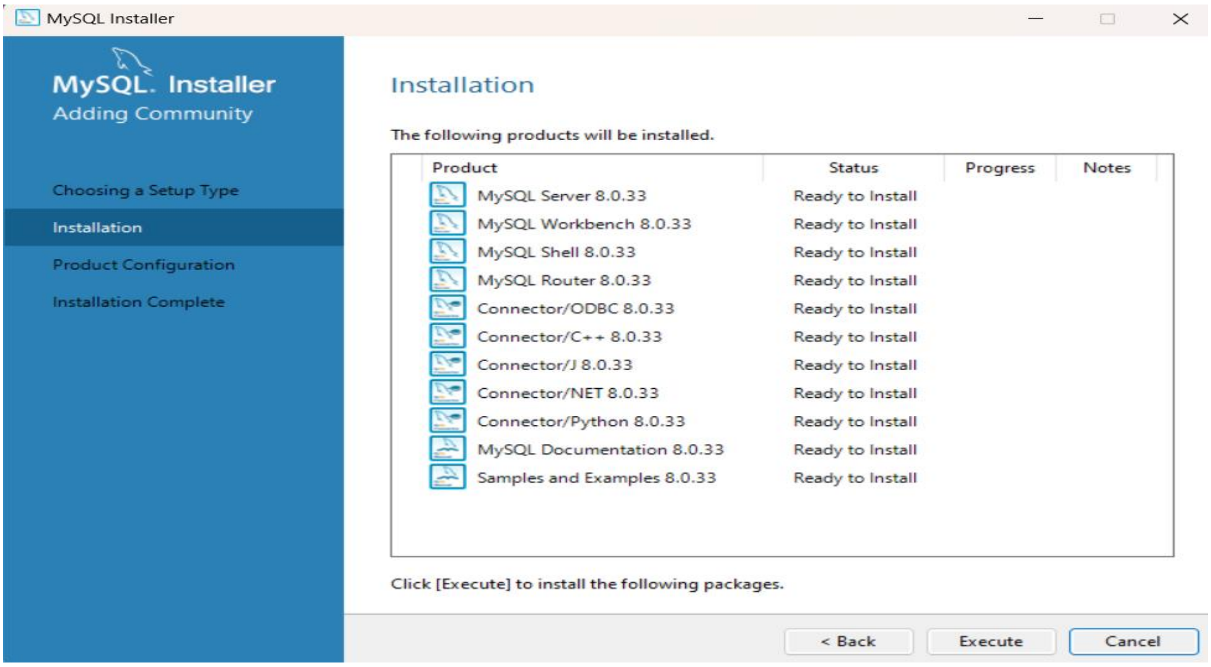
This Setup Type includes:

* MySQL Server
The most popular Open Source SQL database management system.

* MySQL Shell
The new MySQL client application to manage MySQL Servers and InnoDB cluster instances.

* MySQL Router

Step 3. After the system requirements check, the installer will display a summary of the selected products to be installed. Click the "Execute" button to start the installation process.



MySQL Installer

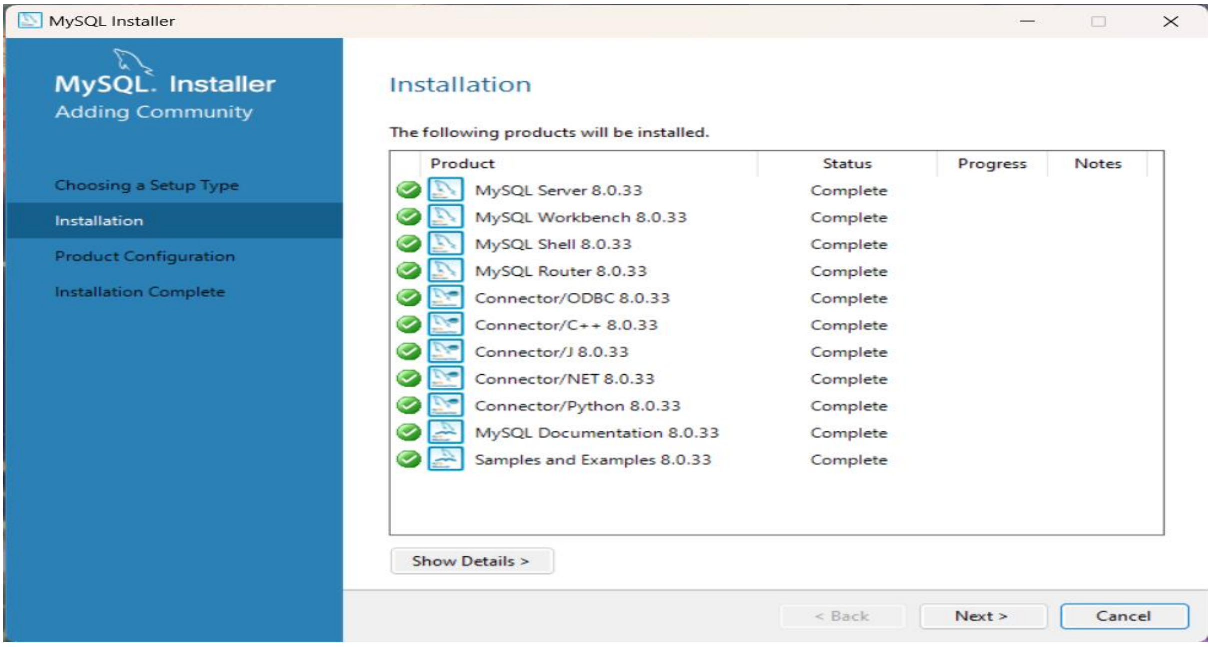
Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.33	Ready to Install		
MySQL Workbench 8.0.33	Ready to Install		
MySQL Shell 8.0.33	Ready to Install		
MySQL Router 8.0.33	Ready to Install		
Connector/ODBC 8.0.33	Ready to Install		
Connector/C++ 8.0.33	Ready to Install		
Connector/J 8.0.33	Ready to Install		
Connector/NET 8.0.33	Ready to Install		
Connector/Python 8.0.33	Ready to Install		
MySQL Documentation 8.0.33	Ready to Install		
Samples and Examples 8.0.33	Ready to Install		

Click [Execute] to install the following packages.

Step 4. After successful installation, you can click the "Next" button to proceed to the configuration step.

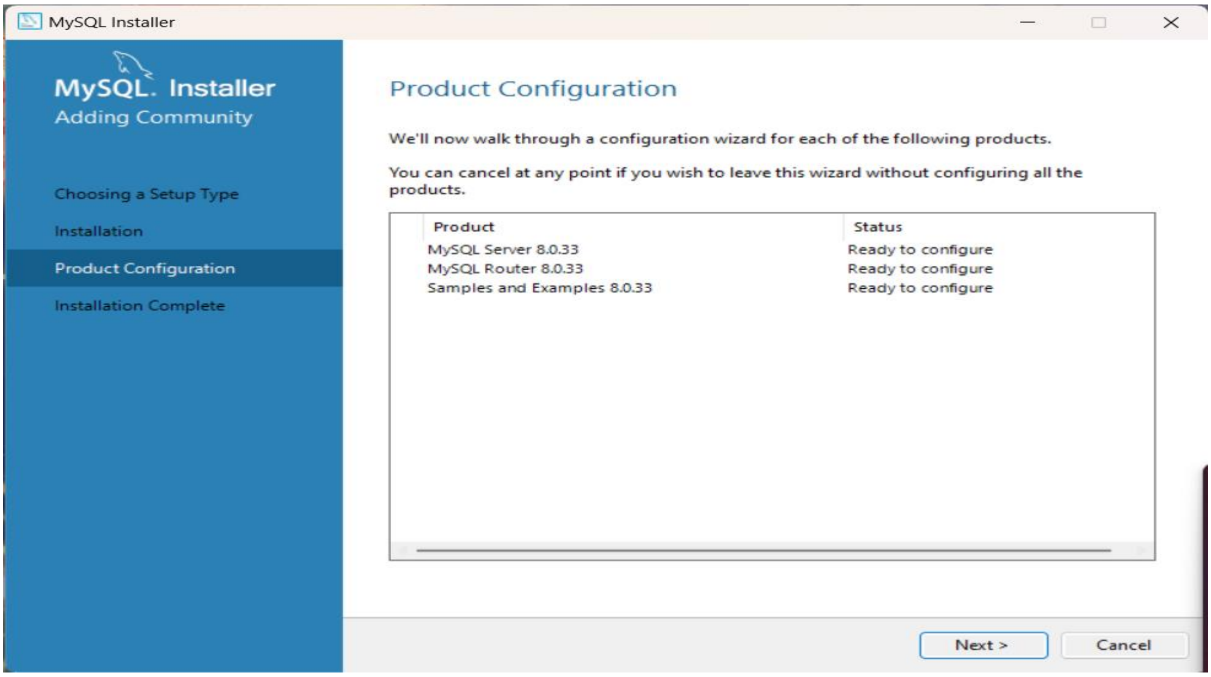


Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.33	Complete		
MySQL Workbench 8.0.33	Complete		
MySQL Shell 8.0.33	Complete		
MySQL Router 8.0.33	Complete		
Connector/ODBC 8.0.33	Complete		
Connector/C++ 8.0.33	Complete		
Connector/J 8.0.33	Complete		
Connector/NET 8.0.33	Complete		
Connector/Python 8.0.33	Complete		
MySQL Documentation 8.0.33	Complete		
Samples and Examples 8.0.33	Complete		

Step 5. Now, installer will walk us through a product configuration. Click the “Next” button.



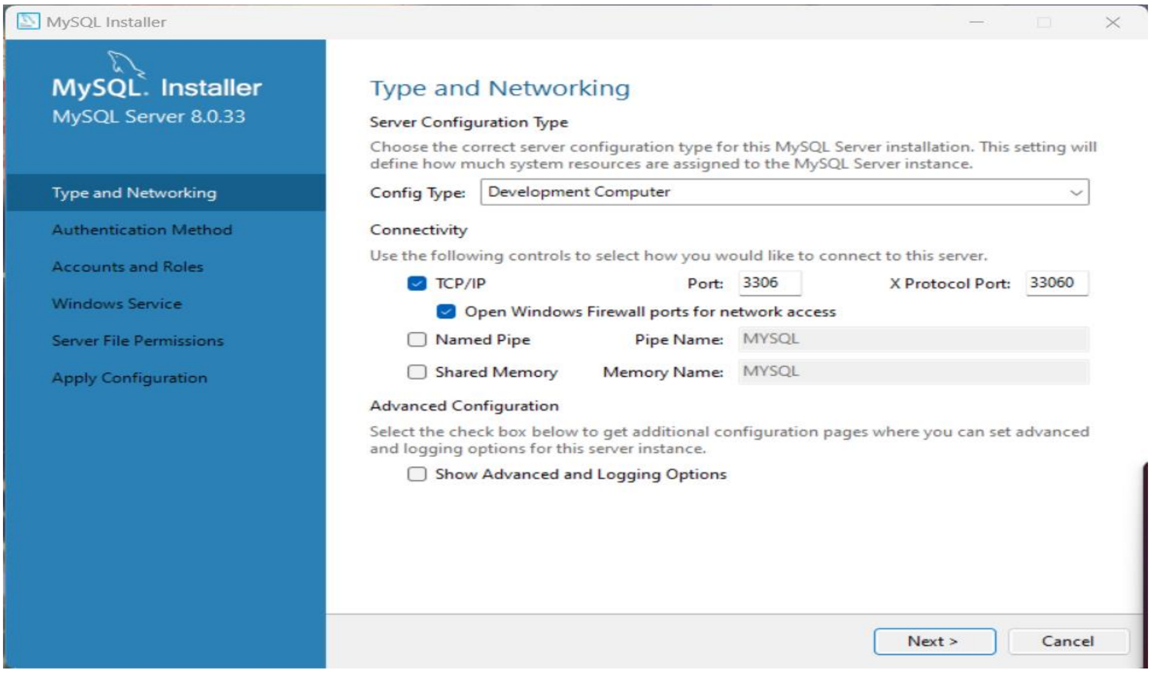
Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.33	Ready to configure
MySQL Router 8.0.33	Ready to configure
Samples and Examples 8.0.33	Ready to configure

Step 6. Provide the necessary details like port number, root password, window service name, etc. You can also choose to enable or disable various options as per your needs. Then proceed further by clicking "Next" button.



Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

☑ TCP/IP Port: 3306 X Protocol Port: 33060

☑ Open Windows Firewall ports for network access

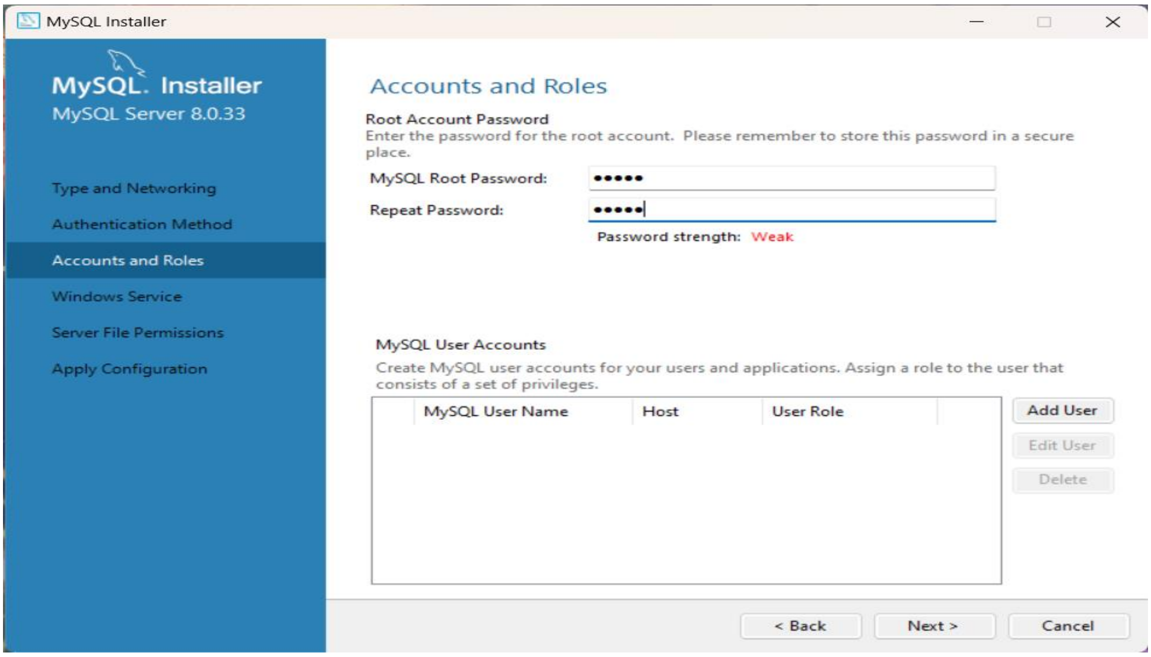
□ Named Pipe Pipe Name: MYSQL

□ Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

□ Show Advanced and Logging Options



Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: •••••

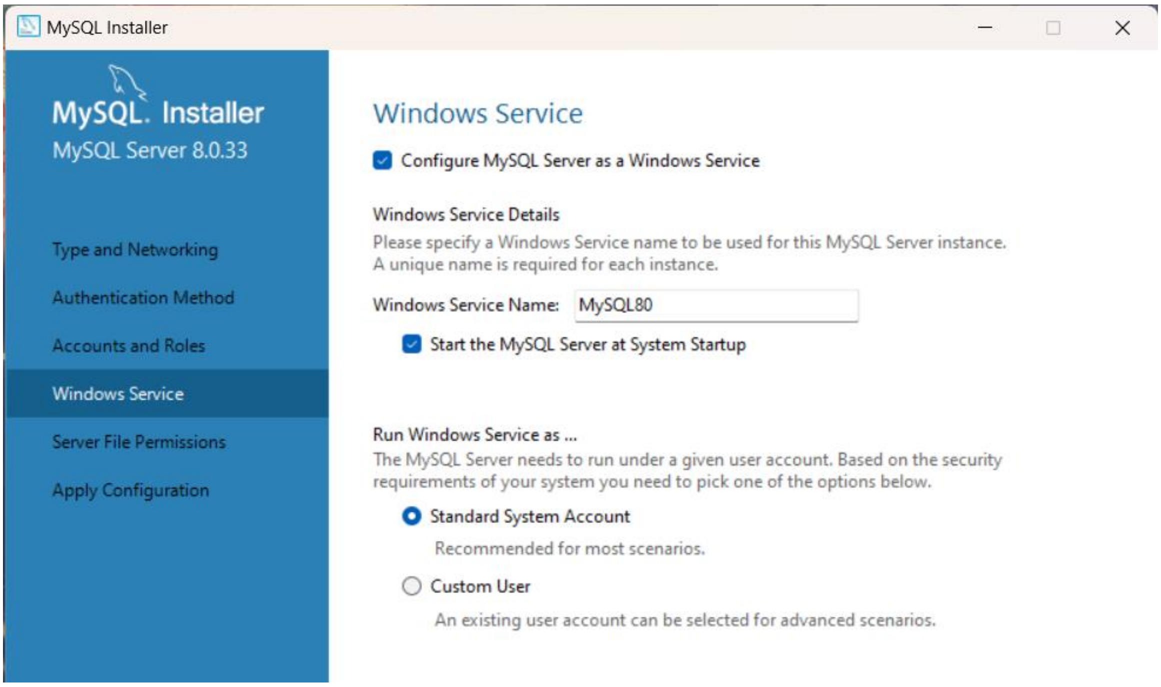
Repeat Password: •••••

Password strength: Weak

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role



Windows Service

☑ Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name: MySQL80

☑ Start the MySQL Server at System Startup

Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

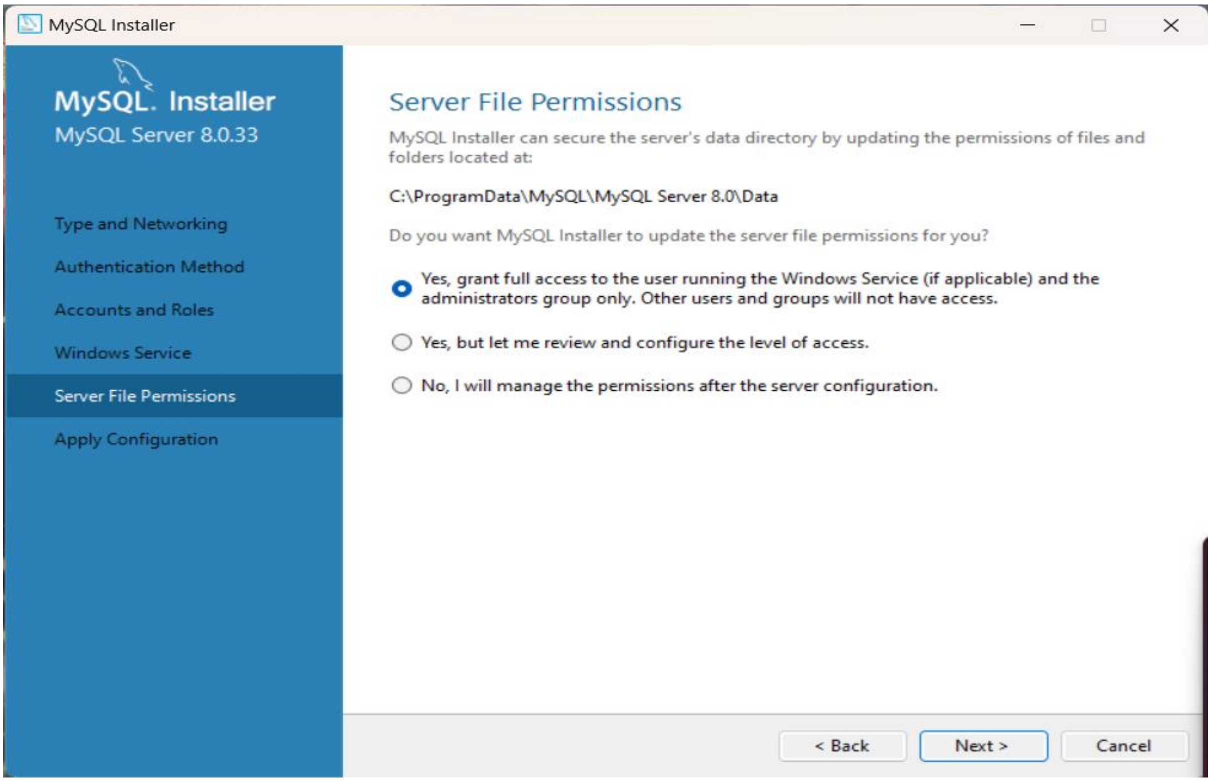
● Standard System Account

Recommended for most scenarios.

○ Custom User

An existing user account can be selected for advanced scenarios.

Step 7. Update server file permissions by allowing the MySQL installer.



Server File Permissions

MySQL Installer can secure the server's data directory by updating the permissions of files and folders located at:

C:\ProgramData\MySQL\MySQL Server 8.0\Data

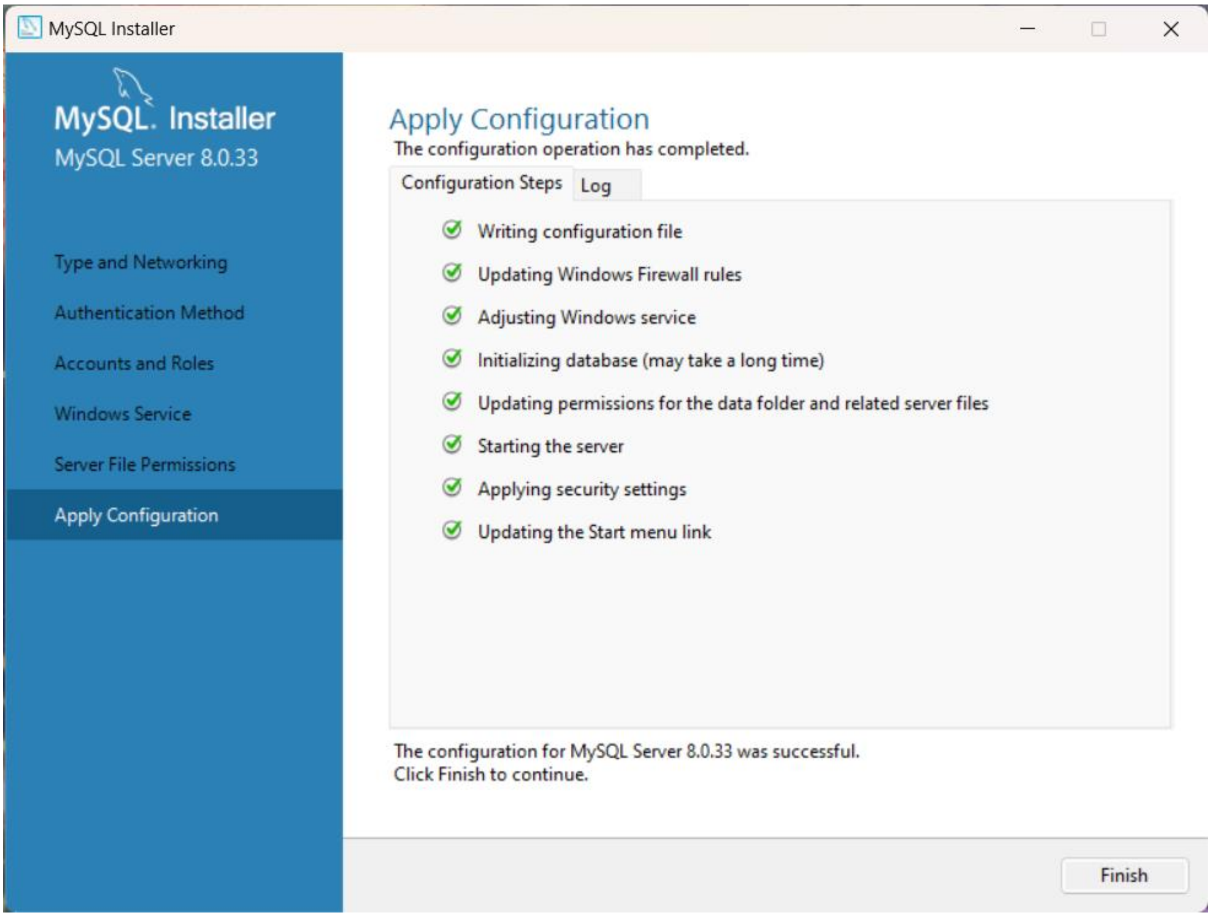
Do you want MySQL Installer to update the server file permissions for you?

● Yes, grant full access to the user running the Windows Service (if applicable) and the administrators group only. Other users and groups will not have access.

○ Yes, but let me review and configure the level of access.

○ No, I will manage the permissions after the server configuration.

Step 8. Execute the configuration steps and after its completion click the "Finish" button.



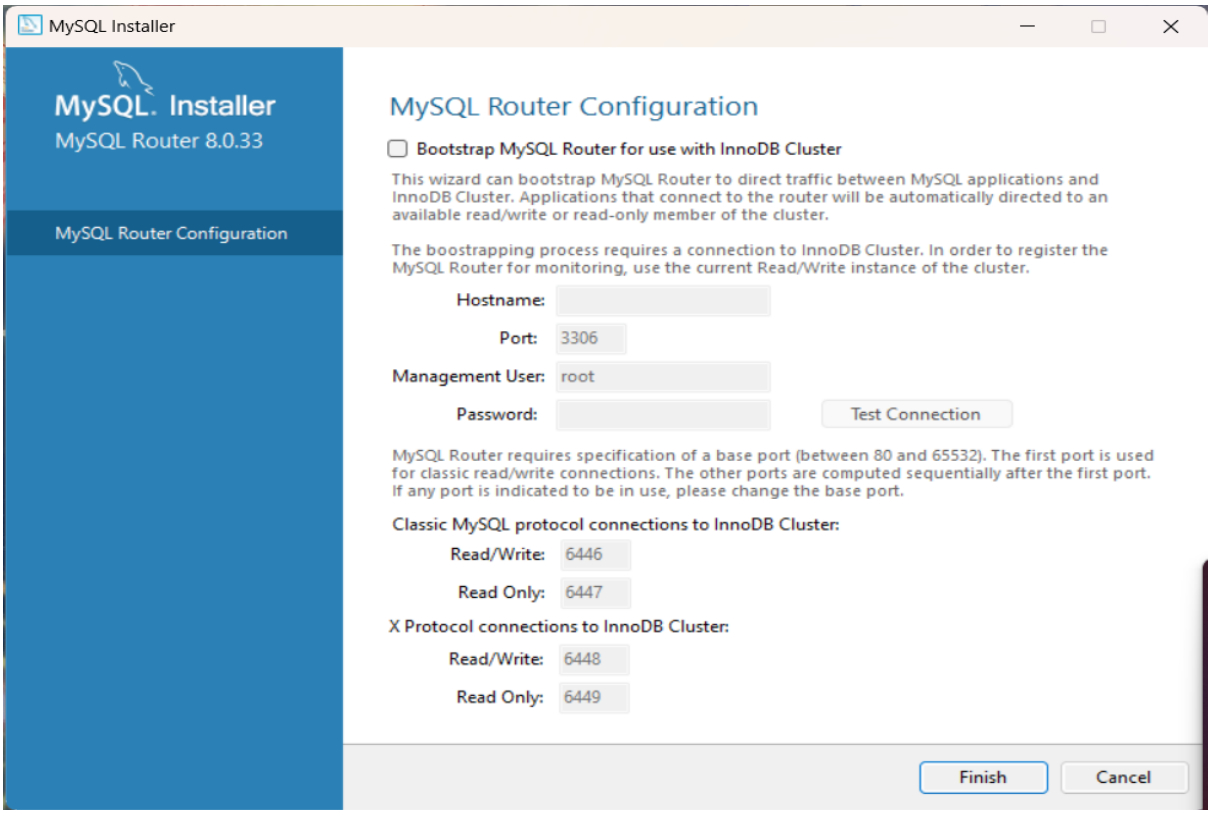
Apply Configuration

The configuration operation has completed.

- ✓ Writing configuration file
- ✓ Updating Windows Firewall rules
- ✓ Adjusting Windows service
- ✓ Initializing database (may take a long time)
- ✓ Updating permissions for the data folder and related server files
- ✓ Starting the server
- ✓ Applying security settings
- ✓ Updating the Start menu link

The configuration for MySQL Server 8.0.33 was successful.
Click Finish to continue.

Step 9. MySQL Router Configuration will be displayed click on "Finish button".



MySQL Router Configuration

☐ Bootstrap MySQL Router for use with InnoDB Cluster

This wizard can bootstrap MySQL Router to direct traffic between MySQL applications and InnoDB Cluster. Applications that connect to the router will be automatically directed to an available read/write or read-only member of the cluster.

The boostrapping process requires a connection to InnoDB Cluster. In order to register the MySQL Router for monitoring, use the current Read/Write instance of the cluster.

Hostname:

Port: 3306

Management User: root

Password:

MySQL Router requires specification of a base port (between 80 and 65532). The first port is used for classic read/write connections. The other ports are computed sequentially after the first port. If any port is indicated to be in use, please change the base port.

Classic MySQL protocol connections to InnoDB Cluster:

Read/Write: 6446

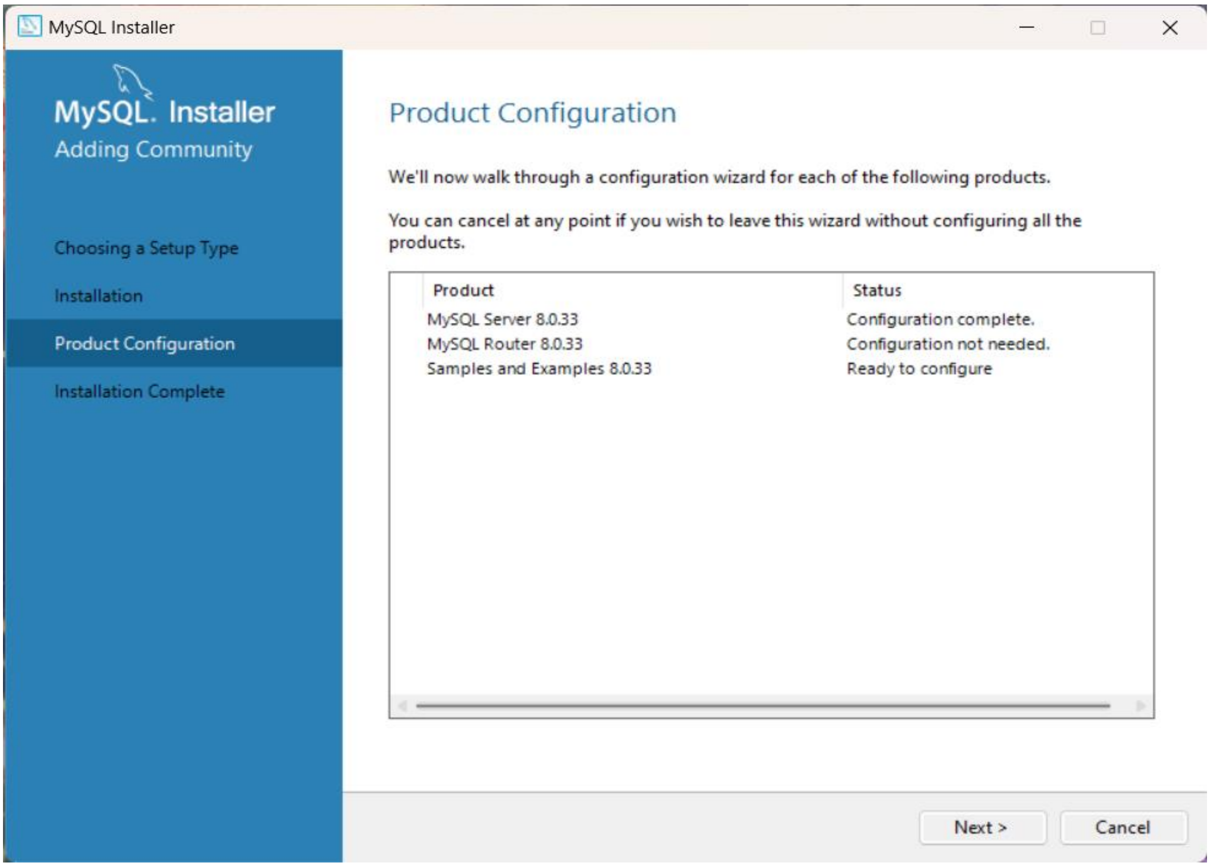
Read Only: 6447

X Protocol connections to InnoDB Cluster:

Read/Write: 6448

Read Only: 6449

Step 10. Again, installer will walk us through a product configuration. Click the “Next” button.



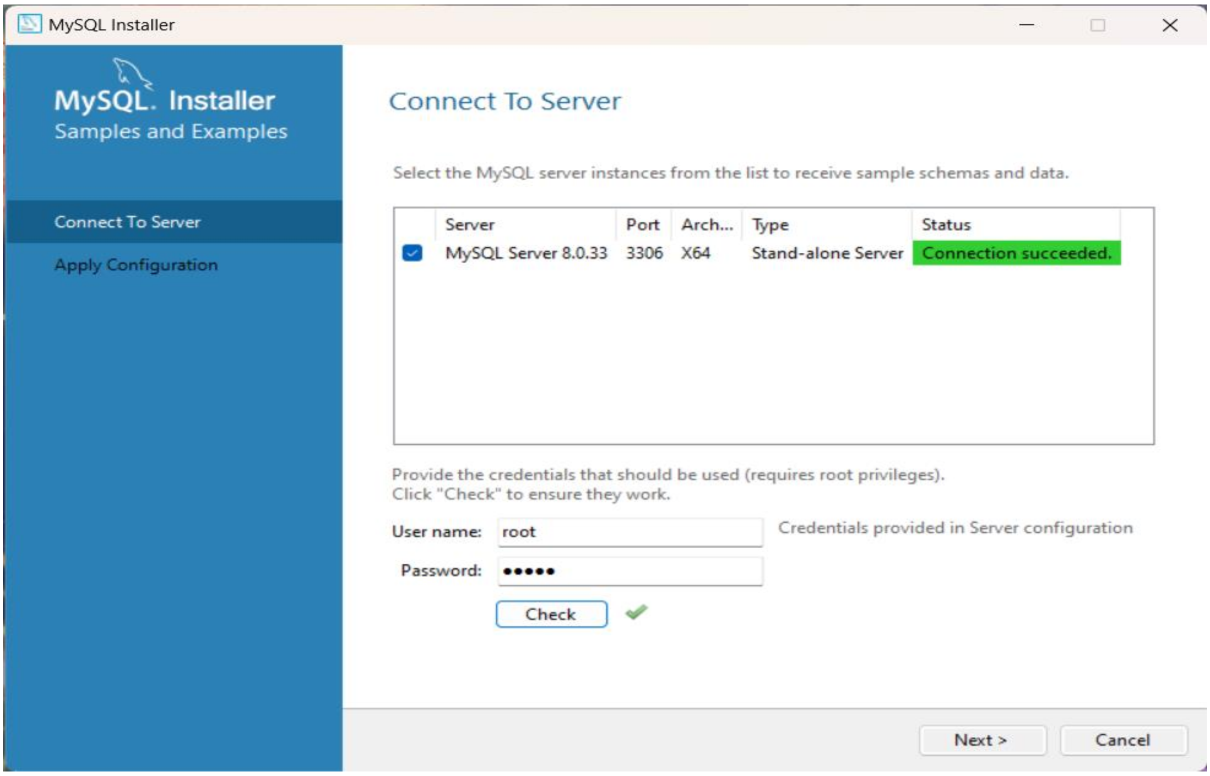
Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.33	Configuration complete.
MySQL Router 8.0.33	Configuration not needed.
Samples and Examples 8.0.33	Ready to configure

Step 11. Connect to the server by providing the username and password which have been set on step 6 and step 9.



Connect To Server

Select the MySQL server instances from the list to receive sample schemas and data.

	Server	Port	Arch...	Type	Status
☑	MySQL Server 8.0.33	3306	X64	Stand-alone Server	Connection succeeded.

Provide the credentials that should be used (requires root privileges).
Click "Check" to ensure they work.

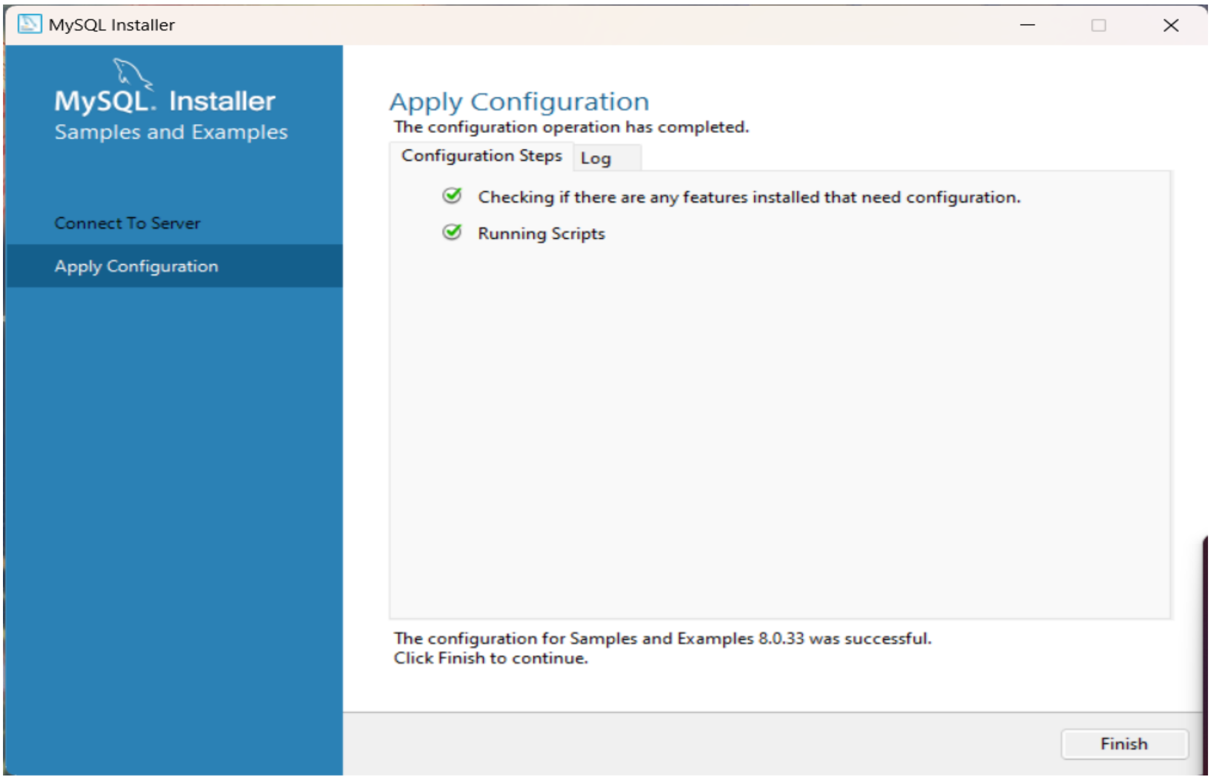
User name: root

Password: •••••

Credentials provided in Server configuration

Step 12. Once you have configured the server, click the "Next" button to proceed.

Step 13. The installer will apply the configuration settings and start the MySQL Server.



MySQL Installer

MySQL. Installer
Samples and Examples

Connect To Server

Apply Configuration

Apply Configuration

The configuration operation has completed.

Configuration Steps | Log

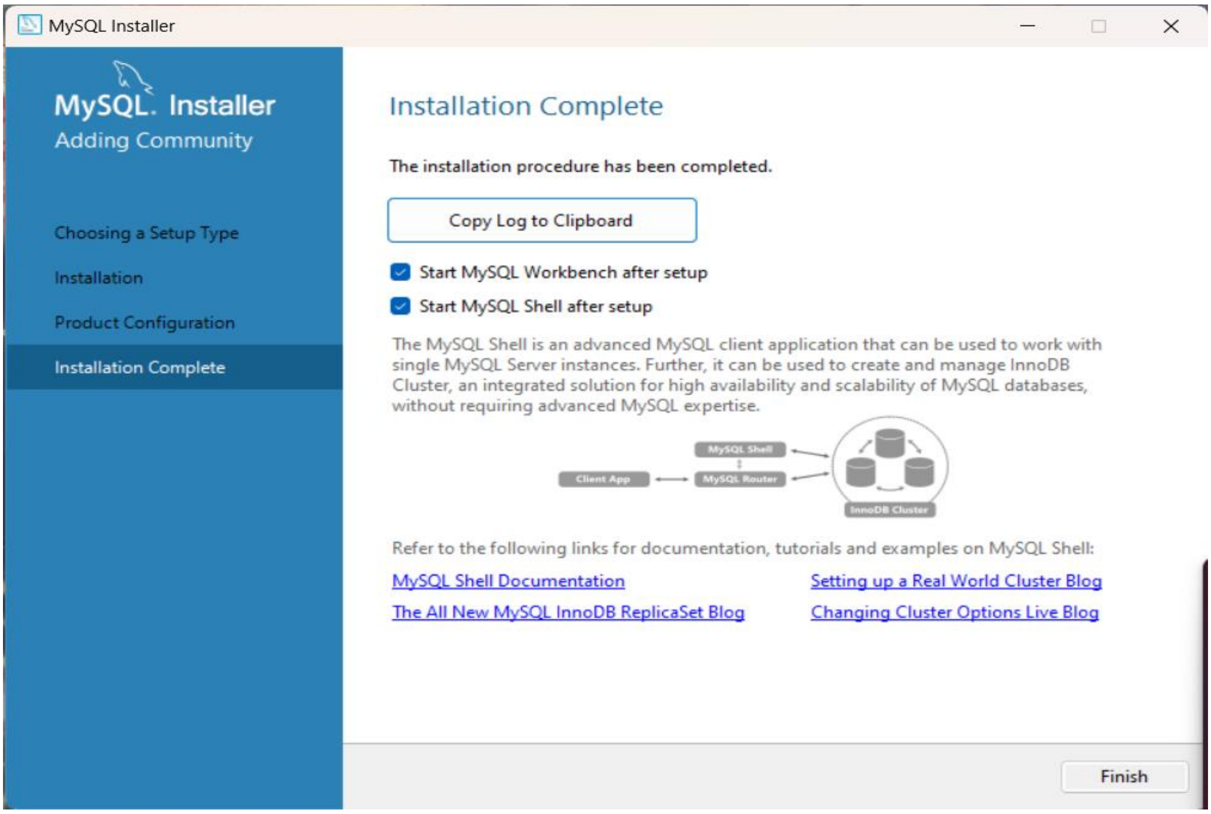
✓ Checking if there are any features installed that need configuration.

✓ Running Scripts

The configuration for Samples and Examples 8.0.33 was successful.
Click Finish to continue.

Finish

Step 14. After the server configuration is complete, you will see a screen with the option to launch the MySQL Shell or MySQL Workbench. You can choose to launch them or click the "Finish" button to exit the installer.



MySQL Installer

MySQL. Installer
Adding Community

Choosing a Setup Type

Installation

Product Configuration

Installation Complete

Installation Complete

The installation procedure has been completed.

Copy Log to Clipboard

☑ Start MySQL Workbench after setup

☑ Start MySQL Shell after setup

The MySQL Shell is an advanced MySQL client application that can be used to work with single MySQL Server instances. Further, it can be used to create and manage InnoDB Cluster, an integrated solution for high availability and scalability of MySQL databases, without requiring advanced MySQL expertise.

Refer to the following links for documentation, tutorials and examples on MySQL Shell:

MySQL Shell Documentation
Setting up a Real World Cluster Blog
The All New MySQL InnoDB ReplicaSet Blog
Changing Cluster Options Live Blog

Finish

Lab 3
SQL Queries Set 1

Q₁. Perform the following task:

Task #1: Create a database called **`DBMS_CSIT.`**

Task #2: Use the database **`DBMS_CSIT.`**

Task #3: Create a table **`student`** with following schema:

`Student (name, roll, marks, address)`

Task #4: Populate the table with following data:

name	*roll*	*marks*	*address*
Ram	12	98	KTM
Hari	13	77	BKT
Shyam	14	78	PKR
Gita	15	79	KTM
Rita	16	80	BKT

Task #5: Write SQL queries to display the records of students in the order of marks (both ascending and descending).

Task #6: Write SQL query to display the records of students in alphabetical order (both forward and reverse alphabetical order).

Task #7: Write SQL query to display details of a student with roll no 12.

Task #8: Write SQL query to display details of students whose name is "Ram".

Task #9: Write SQL query to add an attribute phone_no.

Task #10: Write SQL query to drop the attribute address.

Solution:

Task #1 Solution:

```
create database DBMS_CSIT;
```

Task #2 Solution:

```
use DBMS_CSIT;
```

Task #3 Solution:

```
create table student
(
name varchar(50),
roll int,
marks int,
address varchar(50)
);
```

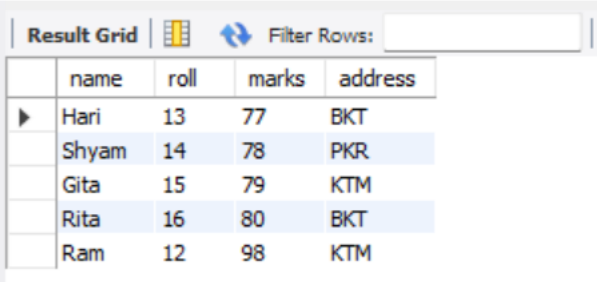
Task #4 Solution:

```
insert into student values("Ram",12,98,"KTM");
insert into student values("Hari",13,77,"BKT");
insert into student values("Shyam",14,78,"PKR");
insert into student values("Gita",15,79,"KTM");
insert into student values("Rita",16,80,"BKT");
```

Task #5 Solution:

```
select * from student
order by marks;
```

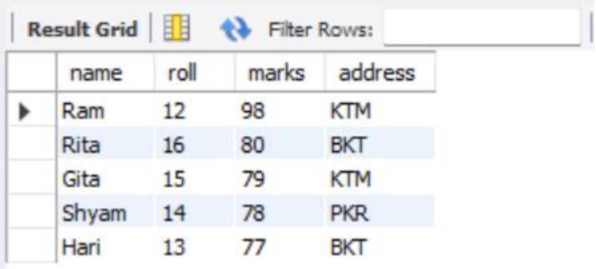
Output:



name	roll	marks	address
Hari	13	77	BKT
Shyam	14	78	PKR
Gita	15	79	KTM
Rita	16	80	BKT
Ram	12	98	KTM

```
select * from student
order by marks desc;
```

Output:

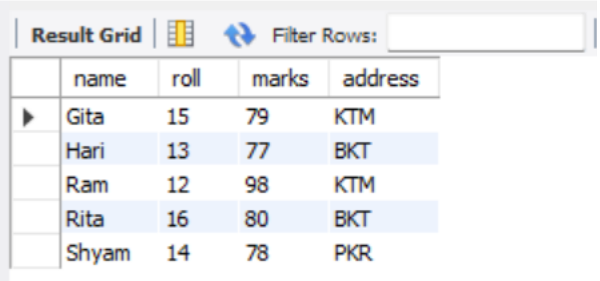


name	roll	marks	address
Ram	12	98	KTM
Rita	16	80	BKT
Gita	15	79	KTM
Shyam	14	78	PKR
Hari	13	77	BKT

Task #6 Solution:

```
select * from student
order by name;
```

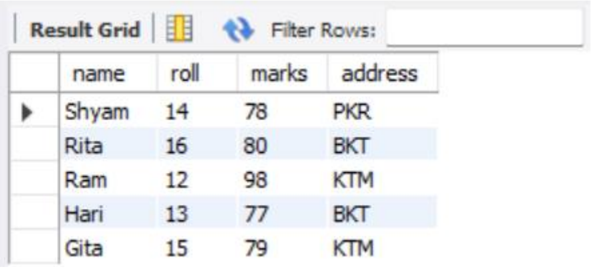
Output:



name	roll	marks	address
Gita	15	79	KTM
Hari	13	77	BKT
Ram	12	98	KTM
Rita	16	80	BKT
Shyam	14	78	PKR

```
select * from student
order by name desc;
```

Output:

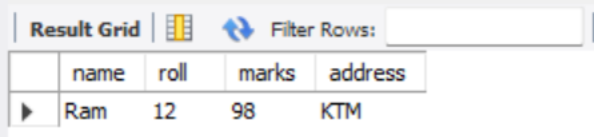


name	roll	marks	address
Shyam	14	78	PKR
Rita	16	80	BKT
Ram	12	98	KTM
Hari	13	77	BKT
Gita	15	79	KTM

Task #7 Solution:

```
select * from student
where roll = 12;
```

Output:

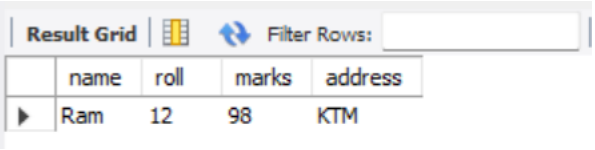


name	roll	marks	address
Ram	12	98	KTM

Task #8 Solution:

```
select * from student
where name = 'Ram';
```

Output:

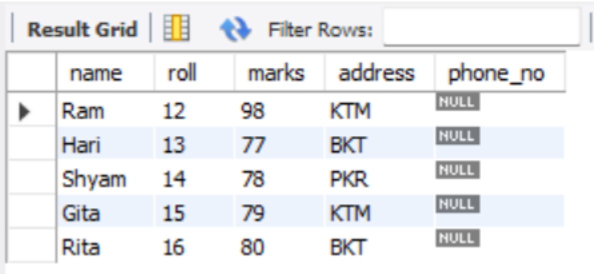


name	roll	marks	address
Ram	12	98	KTM

Task #9 Solution:

```
alter table student
add column phone_no varchar(10) default NULL;
select * from student;
```

Output:

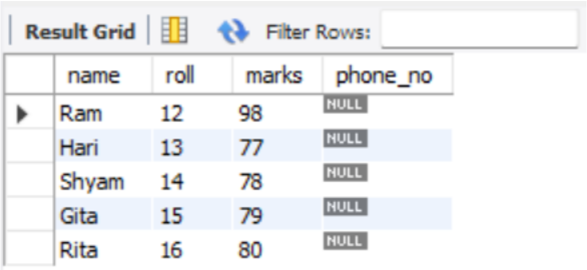


name	roll	marks	address	phone_no
Ram	12	98	KTM	NULL
Hari	13	77	BKT	NULL
Shyam	14	78	PKR	NULL
Gita	15	79	KTM	NULL
Rita	16	80	BKT	NULL

Task #10 Solution:

```
alter table student
drop column address;
```

Output:



name	roll	marks	phone_no
Ram	12	98	NULL
Hari	13	77	NULL
Shyam	14	78	NULL
Gita	15	79	NULL
Rita	16	80	NULL

Q$_2$. Perform the following task:

Task #1: Create a database called **`DBMS`.**

Task #2: Use the database **`DBMS`.**

Task #3: Create a table **`Hospital`** with following schema:

`Hospital (no, patientname, age, department, dateofadm, charges, sex)`

Task #4: Populate the table with following data:

no	*patientname*	*age*	*department*	*dateofadm*	*charges*	*sex*
1	Ram	65	Surgery	22/02/23	300	M
2	Shyam	24	Orthopedic	20/01/23	200	M
3	Hari	45	Orthopedic	19/02/23	200	M
4	Rita	12	Surgery	01/01/23	300	F
5	Sita	36	ENT	12/01/23	250	F

Task #5: Write SQL queries to display patient's name, charges and age for male patients only.

Task #6: Write SQL query to display the name of all patients with their date of admission in reverse alphabetical order.

Task #7: Write SQL query to display details of the patients of orthopedic department.

Solution:

Task #1 Solution:

```
create database DBMS;
```

Task #2 Solution:

```
use DBMS;
```

Task #3 Solution:

```
create table Hospital(
no int,
patient_name varchar(50),
age int,
department varchar(50),
dateofadm date,
charges int,
sex varchar(10));
```

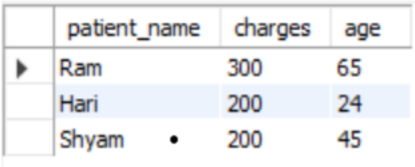
Task #4 Solution:

```
insert into Hospital values(1,"Ram",65,"Surgery","22/02/23",300,"M");
insert into Hospital values(2,"Hari",24,"Orthopedic","20/01/23",200,"M");
insert into Hospital values(3,"Shyam",45,"Orthopedic","19/02/23",200,"M");
insert into Hospital values(4,"Rita",12,"Surgery","01/01/23",300,"F");
insert into Hospital values(5,"Sita",36,"ENT","12/01/23",250,"F");
```

Task #5 Solution:

```
select patient_name,charges,age from Hospital
where sex="M";
```

Output:

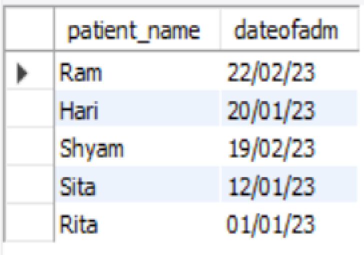


	patient_name	charges	age
▶	Ram	300	65
	Hari	200	24
	Shyam •	200	45

Task #6 Solution:

```
select patient_name,dateofadm from Hospital
order by dateofadm desc;
```

Output:

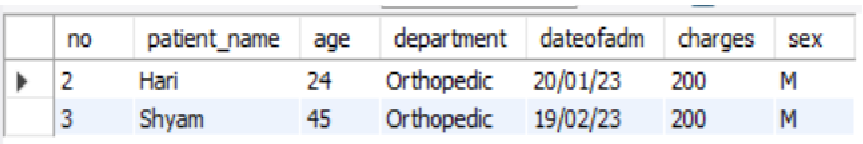


	patient_name	dateofadm
▶	Ram	22/02/23
	Hari	20/01/23
	Shyam	19/02/23
	Sita	12/01/23
	Rita	01/01/23

Task #7 Solution:

```
select * from Hospital
where department="Orthopedic";
```

Output:



	no	patient_name	age	department	dateofadm	charges	sex
▶	2	Hari	24	Orthopedic	20/01/23	200	M
	3	Shyam	45	Orthopedic	19/02/23	200	M

Q3. Perform the following task:

Task #1: Create a database called **CSIT.**

Task #2: Use the database **CSIT**.

Task #3: Create a table **`Student`** with following schema:

```
      Student (student_id, name, marks, subject, grade)
```

Task #4: Populate the table with your own records.

Task #5: Write SQL queries to display the records of students who have scored more than 80 marks in science subject.

Task #6: Write SQL query to count the total number of students who have passed assuming more than 50 is pass mark.

Task #7: Write SQL query to find the average marks of all students.

Solution:

Task #1 Solution:

```
create database CSIT;
```

Task #2 Solution:

```
use CSIT;
```

Task #3 Solution:

```
create table Student(
      student_id int,
      name varchar(50),
      marks int,
      subject varchar(50),
      grade varchar(20)
);
```

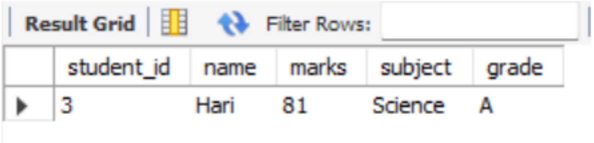
Task #4 Solution:

```
insert into student values(1,"Ram",95,"Math","A+");
insert into student values(2,"Shyam",70,"Science","B+");
insert into student values(3,"Hari",81,"Science","A");
insert into student values(4,"Gita",78,"Math","B+");
insert into student values(5,"Sita",45,"Science","C");
```

Task #5 Solution:

```
select * from student where marks>80 and subject="Science";
```

Output:

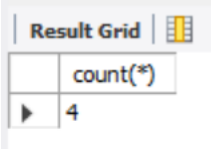


student_id	name	marks	subject	grade
3	Hari	81	Science	A

Task #6 Solution:

```
select count(*) from student where marks>50;
```

Output:

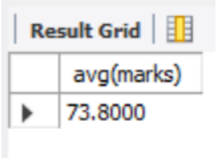


count(*)
4

Task #7 Solution:

```
select avg(marks) from student;
```

Output:



avg(marks)
73.8000

Lab 4

SQL Queries Set 2

Q$_1$. Create a table `employee` with following schema:

```
employee(name, eid, designation, salary)
```

Perform following tasks:

Task #1. Populate employee table with 10 or more records

Task #2. Write SQL query to retrieve all records from the table.

Task #3. Write SQL query to set salary of all employees whose designation is "Supervisor"

Task #4. Write SQL query to change the name of employee with eid=50 to "Hari"

Task #5. Write SQL query to delete the record of a employee with eid=10

Task #6. Write SQL query to display average salary of employees

Task #7. Write SQL query to display the no. of employees

Task #8. Write SQL query to display the total salary paid by the company.

Task #9. Write SQL query to increase the salary of all employees by 10%.

***Solution*:**

```
create table employee(
eid int not null primary key,
name varchar(50),
salary int,
designation varchar(50));
```

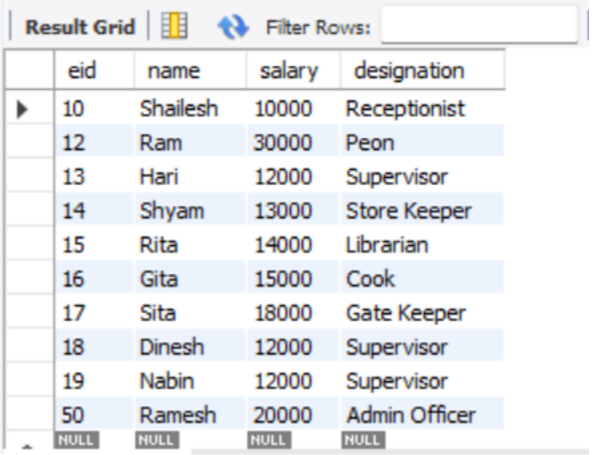
Task #1 Solution:

```
insert into employee values(12,"Ram",30000,"Peon");
insert into employee values(13,"Hari",12000,"Supervisor");
insert into employee values(14,"Shyam",13000,"Store Keeper");
insert into employee values(15,"Rita",14000,"Librarian");
insert into employee values(16,"Gita",15000,"Cook");
insert into employee values(17,"Sita",18000,"Gate Keeper");
insert into employee values(18,"Dinesh",12000,"Supervisor");
insert into employee values(19,"Nabin",12000,"Supervisor");
insert into employee values(50,"Ramesh",20000,"Admin Officer");
insert into employee values(10,"Shailesh",10000,"Receptionist");
```

Task #2 Solution:

```
select *from employee;
```

Output:

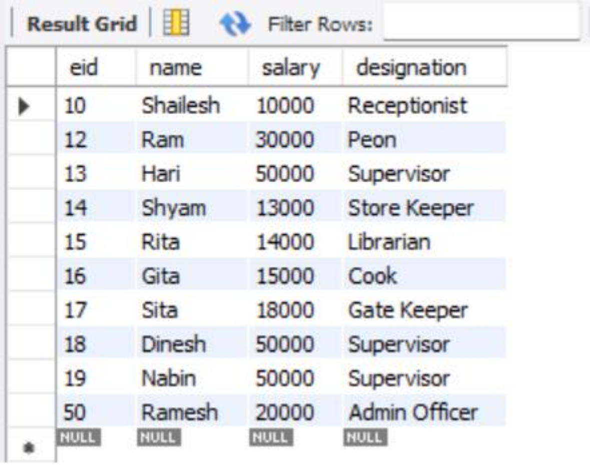


eid	name	salary	designation
10	Shailesh	10000	Receptionist
12	Ram	30000	Peon
13	Hari	12000	Supervisor
14	Shyam	13000	Store Keeper
15	Rita	14000	Librarian
16	Gita	15000	Cook
17	Sita	18000	Gate Keeper
18	Dinesh	12000	Supervisor
19	Nabin	12000	Supervisor
50	Ramesh	20000	Admin Officer
NULL	NULL	NULL	NULL

Task #3 Solution:

```
SET SQL_SAFE_UPDATES = 0;
update employee
set salary=50000
where designation="Supervisor";
select *from employee;
```

Output:

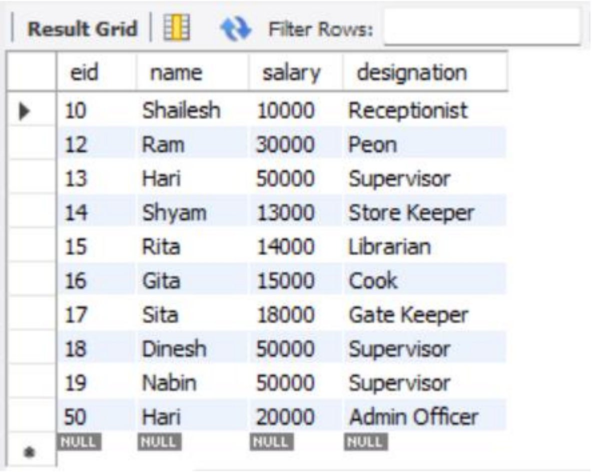


eid	name	salary	designation
10	Shailesh	10000	Receptionist
12	Ram	30000	Peon
13	Hari	50000	Supervisor
14	Shyam	13000	Store Keeper
15	Rita	14000	Librarian
16	Gita	15000	Cook
17	Sita	18000	Gate Keeper
18	Dinesh	50000	Supervisor
19	Nabin	50000	Supervisor
50	Ramesh	20000	Admin Officer
NULL	NULL	NULL	NULL

Task #4 Solution:

```
update employee
set name="Hari"
where eid=50;
select *from employee;
```

Output:

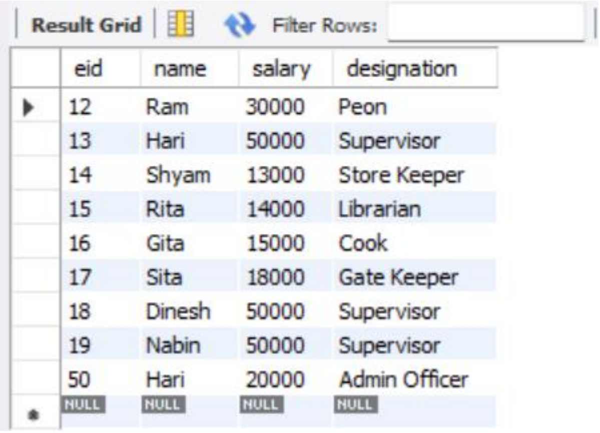


eid	name	salary	designation
10	Shailesh	10000	Receptionist
12	Ram	30000	Peon
13	Hari	50000	Supervisor
14	Shyam	13000	Store Keeper
15	Rita	14000	Librarian
16	Gita	15000	Cook
17	Sita	18000	Gate Keeper
18	Dinesh	50000	Supervisor
19	Nabin	50000	Supervisor
50	Hari	20000	Admin Officer
NULL	NULL	NULL	NULL

Task #5 Solution:

```
delete from employee
where eid=10;
select *from employee;
```

Output:

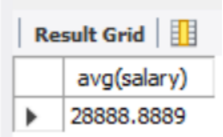


eid	name	salary	designation
12	Ram	30000	Peon
13	Hari	50000	Supervisor
14	Shyam	13000	Store Keeper
15	Rita	14000	Librarian
16	Gita	15000	Cook
17	Sita	18000	Gate Keeper
18	Dinesh	50000	Supervisor
19	Nabin	50000	Supervisor
50	Hari	20000	Admin Officer
NULL	NULL	NULL	NULL

Task #6 Solution:

```
select avg(salary) from employee;
```

Output:

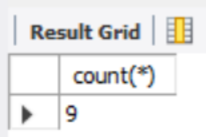


avg(salary)
28888.8889

Task #7 Solution:

```
select count(*) from employee;
```

Output:

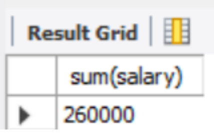


count(*)
9

Task #8 Solution:

```
select sum(salary) from employee;
```

Output:

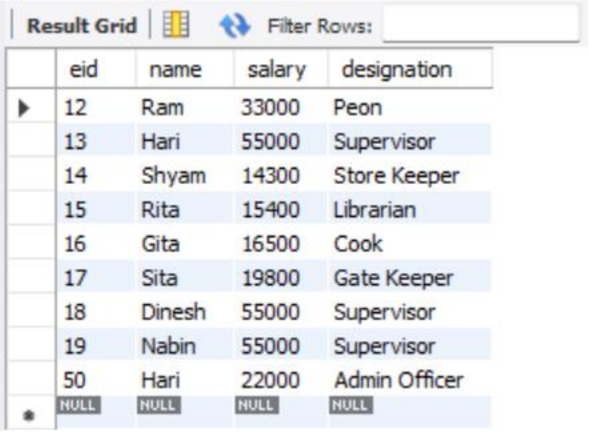


sum(salary)
260000

Task #9 Solution:

```
update employee
set salary=1.1*salary;
```

Output:



eid	name	salary	designation
12	Ram	33000	Peon
13	Hari	55000	Supervisor
14	Shyam	14300	Store Keeper
15	Rita	15400	Librarian
16	Gita	16500	Cook
17	Sita	19800	Gate Keeper
18	Dinesh	55000	Supervisor
19	Nabin	55000	Supervisor
50	Hari	22000	Admin Officer
NULL	NULL	NULL	NULL

Q2. Consider the following tables:

Student

Name	***Roll***	*CID*
Ram	1	S001
Shyam	2	S002
Hari	3	S003
Rita	4	S001
Sita	5	S002
Gita	6	S003

Course

CID	*Cname*
S001	DBMS
S002	TOC
S003	CN
S004	OS
S005	Extra
S006	AI

Perform the following tasks:

1 Create two tables with following schema:

`Student(Name, Roll, CID) and Course(CID, Cname)`

2 Set CID of relation Student as foreign key which references CID of relation Course.

3 Populate the tables with records.

4 Write SQL query to retrieve records of all students along with course they took.

5 Write SQL query to display details of all students who took DBMS course.

6 Write SQL query to delete the table Course and comment on the result.

7 Write SQL query to insert a record ('Kartik', 7,'S007') into student table and comment on the result.

Solution:

Task #1 and 2 Solution:

```
create database dbms_csit;
use dbms_csit;
create table course
(
CID varchar(10),
Cname varchar(50),
primary key(CID)
);
create table student(
name varchar(50),
roll int primary key,
CID varchar(10) ,
foreign key(CID) references Course(CID)
);
```

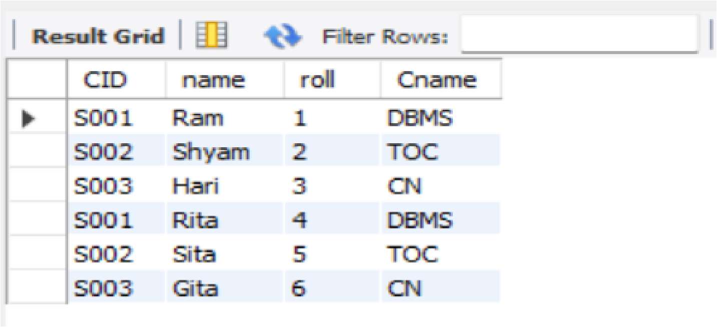
Task #3 Solution:

```
insert into course values('S001','DBMS');
insert into course values('S002','TOC');
insert into course values('S003','CN');
insert into course values('S004','OS');
insert into course values('S005','Extra');
insert into course values('S006','AI');

insert into student values('Ram',1,'S001');
insert into student values('Shyam',2,'S002');
insert into student values('Hari',3,'S003');
insert into student values('Rita',4,'S001');
insert into student values('Sita',5,'S002');
insert into student values('Gita',6,'S003');
```

Task # 4 Solution:

```
select * from student natural join course;
```

Output:

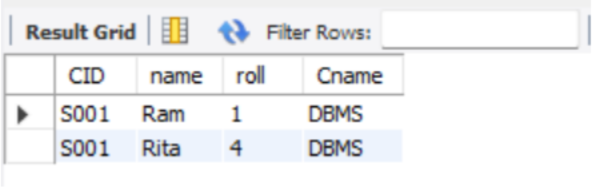


	CID	name	roll	Cname
▶	S001	Ram	1	DBMS
	S002	Shyam	2	TOC
	S003	Hari	3	CN
	S001	Rita	4	DBMS
	S002	Sita	5	TOC
	S003	Gita	6	CN

Task #5 Solution:

```
select * from student natural join course
where cname='DBMS';
```

Output:



	CID	name	roll	Cname
▶	S001	Ram	1	DBMS
	S001	Rita	4	DBMS

Task #6 Solution:

```
drop table course;
```

Output:

```
# generates following error message
Cannot drop table 'course' referenced by a foreign key constraint
'student_ibfk_1' on table 'student'.
```

Task #7 Solution:

```
insert into student values('Kartik',7,'S007');
```

Output:

```
# generates following error message
 Foreign key constraint fails
```

Lab 5

SQL Queries Set 3

Q1. Consider the following COURSE table given below:

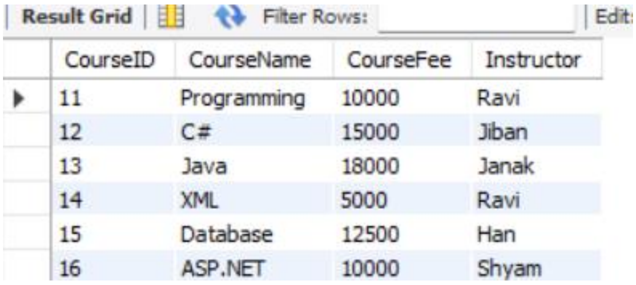
CourseID	CourseName	CourseFee	Instructor
11	Programming	10000	Ravi
12	C#	15000	Jiban
13	Java	18000	Janak
14	XML	5000	Ravi
15	Database	12500	Han
16	ASP.net	10000	Shyam

Now answer the following questions:

a) Write SQL syntax to create the given table and insert few records in it.

```
create table COURSE
(
CourseID integer primary key,
CourseName varchar(50),
CourseFee integer,
Instructor varchar(50)
);
insert into COURSE values(11,'Programming',10000,'Ravi');
insert into COURSE values(12,'C#',15000,'Jiban');
insert into COURSE values(13,'Java',18000,'Janak');
insert into COURSE values(14,'XML',5000,'Ravi');
insert into COURSE values(15,'Database',12500,'Han');
insert into COURSE values(16,'ASP.NET',10000,'Shyam');
select * from COURSE;
```

Output:



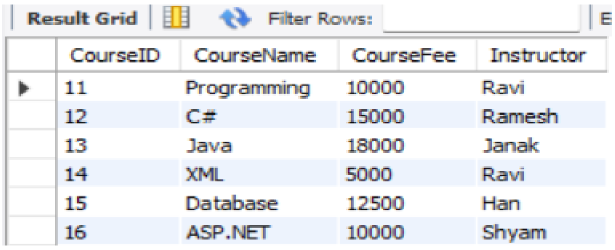
CourseID	CourseName	CourseFee	Instructor
11	Programming	10000	Ravi
12	C#	15000	Jiban
13	Java	18000	Janak
14	XML	5000	Ravi
15	Database	12500	Han
16	ASP.NET	10000	Shyam

b) **Write SQL syntax to update the instructor to Ramesh whose CourseID is 12.**

Solution:

```
update COURSE
set Instructor='Ramesh'
where CourseID=12;
```

Output:



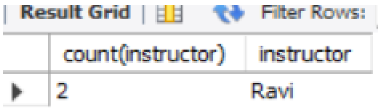
CourseID	CourseName	CourseFee	Instructor
11	Programming	10000	Ravi
12	C#	15000	Ramesh
13	Java	18000	Janak
14	XML	5000	Ravi
15	Database	12500	Han
16	ASP.NET	10000	Shyam

c) **Write SQL query to retrieve all information of courses that have more than one instructor.**

Solution:

```
select count(instructor), instructor from Course
group by instructor
having count(instructor)>1;
```

Output:



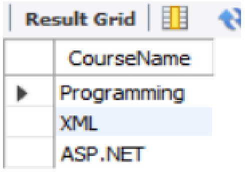
count(instructor)	instructor
2	Ravi

d) **Write SQL query to find the name of course whose fee is less than the average fee of all the courses.**

Solution:

```
select CourseName from COURSE
where CourseFee<(Select avg(CourseFee) from COURSE);
```

Output:



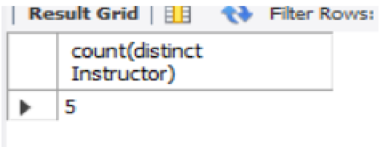
CourseName
Programming
XML
ASP.NET

e) **Write SQL query to count distinct number of instructors in the course table.**

Solution:

```
select count(distinct Instructor) from COURSE;
```

Output:



count(distinct Instructor)
5

Q2. Consider the following relation and attributes:

PRODUCT

PID	Varchar(5)
ProductName	Varchar(40)
Unit Price	number(5)

a) Develop DDL in SQL to implement above schema.

Solution:

```
create table PRODUCT
(
PID Varchar(5),
ProductName Varchar(40),
UnitPrice numeric(5)
);
insert into PRODUCT values(45678, ‘monitor’,15000);
insert into PRODUCT values(34567, ‘laptop’,30000);
insert into PRODUCT values(62345, ‘keyboard’,3000);
insert into PRODUCT values(56789, ‘mouse’,100);
insert into PRODUCT values(23456, ‘landline’,190);
```

b) Develop SQL Queries to insert a new product named Smartphone with PID 12345 and price of 25000.

Solution:

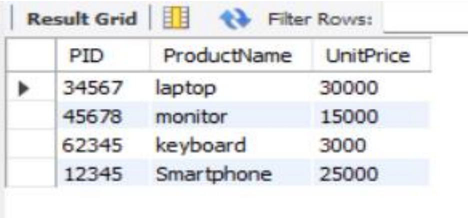
```
insert into PRODUCT(PID,ProductName,UnitPrice)
values(12345,'Smartphone',25000);
```

c) Develop SQL queries to list product with unit price greater than 200.

Solution:

```
select * from Product
where UNITPRICE>200
```

Output:



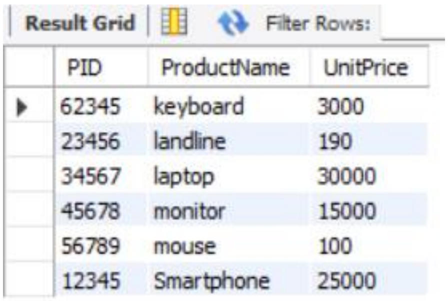
PID	ProductName	UnitPrice
34567	laptop	30000
45678	monitor	15000
62345	keyboard	3000
12345	Smartphone	25000

d) **Develop SQL queries to list products sorted by the “ProductName” column.**

Solution:

```
select * from Product
order by ProductName;
```

Output:



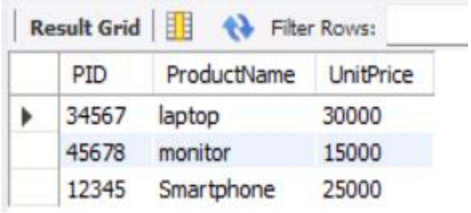
PID	ProductName	UnitPrice
62345	keyboard	3000
23456	landline	190
34567	laptop	30000
45678	monitor	15000
56789	mouse	100
12345	Smartphone	25000

e) **Develop SQL queries to list details of product whose price is greater than the average price of all products.**

Solution:

```
select * from PRODUCT
where UNITPRICE>(select avg(unitprice) from PRODUCT);
```

Output:



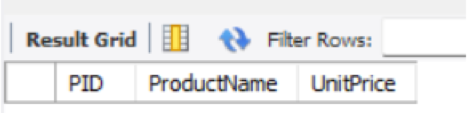
PID	ProductName	UnitPrice
34567	laptop	30000
45678	monitor	15000
12345	Smartphone	25000

f) **Develop SQL queries to delete all rows in a table without deleting the table.**

Solution:

```
delete from Product;
```

Output:



PID	ProductName	UnitPrice

g) **Develop SQL queries to delete the table named product from the database.**

Solution:

```
drop table PRODUCT;
```

Output:

```
Table ‘dbms.product’ doesn’t exist
```

Q$_3$. Consider the relational database where the primary keys are highlighted. Give an expression in SQL for each of the following queries:

Employee(**person_name**, street, city)

Works(**person_name**, company_name, salary)

Company(**company_name**, city)

Manages(**person_name**,manager_name)

a) Implement DDL for the given relation.

Solution:

```
create database DBMS_CSIT;
use DBMS_CSIT;
create table Employee
(
person_name varchar(30) primary key,
street varchar (50),
city varchar(30)
);
insert into Employee values("Raju","L","KTM");
insert into Employee values("Shyam","M","BKT");
insert into Employee values("Gita","N","PKR");
insert into Employee values("Ram","L","KTM");
create table Works
(
person_name varchar(30) primary key,
company_name varchar (50),
salary numeric);
insert into Works values("Raju","First Bank Corporation","15000");
insert into Works values("Shyam","Small Bank Corporation","10000");
insert into Works values("Gita","Last Bank Corporation","18000");
create table Company
(
company_name varchar(50) primary key,
city varchar(50)
);
```

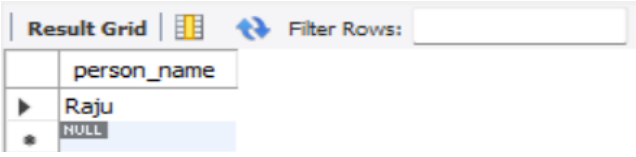
```
insert into Company values("First Bank Corporation","KTM");
insert into Company values("Small Bank Corporation","BKT");
insert into Company values("Last Bank Corporation","PKR");
create table Manages
(
person_name varchar(30) primary key,
manager_name varchar(30),
foreign key (person_name) references employee(person_name)
);
insert into Manages values("Raju","John");
insert into Manages values("Shyam","John");
insert into Manages values("Gita","Raju");
insert into Manages values("Ram","Raju");
```

b) Find the names of all employees who work for the First Bank Corporation.

Solution:

```
select person_name from WORKS where
company_name='First Bank Corporation';
```

Output:



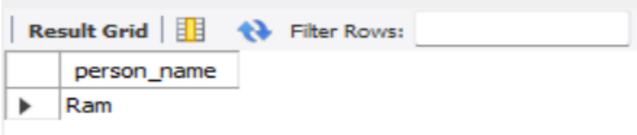
```
person_name
Raju
NULL
```

c) Find the names of all employees who live in the same city and on the same street as do their managers.

Solution:

```
Select E1.person_name
From Employee as E1, Employee as E2, Manages as M
Where E1.person_name=M.person_name and
E2.person_name=M.manager_name
and E1.stree=E2.street and E1.city=E2.city
```

Output:



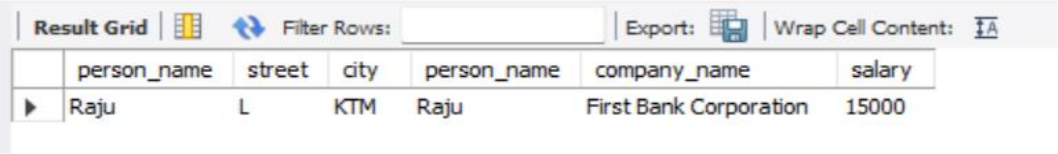
```
person_name
Ram
```

d) **Find the names, street address and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.**

Solution:

```
select *from Employee
inner join WORKS
on Employee.person_name=Works.PERSON_NAME
where Works.COMPANY_NAME='First Bank Corporation' and
Works.salary>10000;
```

Output:



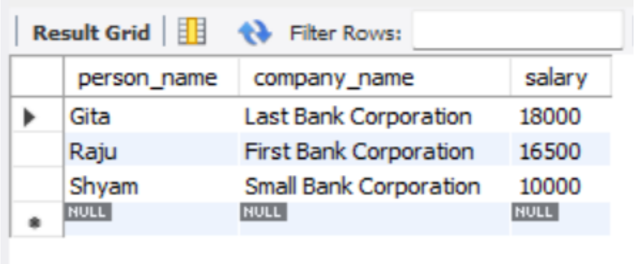
	person_name	street	city	person_name	company_name	salary
▶	Raju	L	KTM	Raju	First Bank Corporation	15000

e) **Give all employees of First Bank Corporation a 10 percent salary raise.**

Solution:

```
SET SQL_SAFE_UPDATES=0;
update WORKS SET SALARY=1.1*SALARY
WHERE COMPANY_NAME='First Bank Corporation';
select * from works;
```

Output:



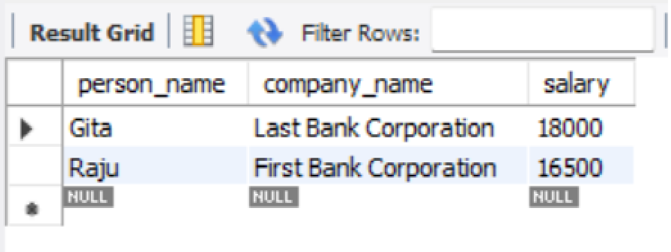
	person_name	company_name	salary
▶	Gita	Last Bank Corporation	18000
	Raju	First Bank Corporation	16500
	Shyam	Small Bank Corporation	10000
*	NULL	NULL	NULL

f) **Delete all the tuples in the works relation for employees of Small Bank Corporation**

Solution:

```
delete from Works where COMPANY_NAME='Small Bank Corporation';
```

Output:



	person_name	company_name	salary
▶	Gita	Last Bank Corporation	18000
	Raju	First Bank Corporation	16500
*	NULL	NULL	NULL

Q4. Create a `student` table with following schema

```
STUDENT (name, roll, marks, address);
```

a) Write SQL query to create the table.

Solution:

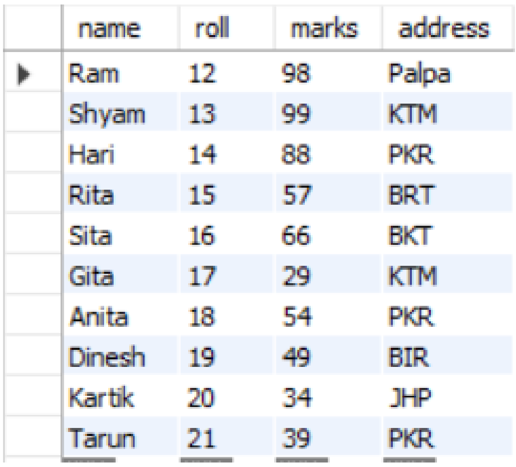
```
create table student
(
name varchar(20),
roll integer primary key,
marks integer,
address varchar(50)
);
```

b) Write SQL queries to populate the table with 10 records.

Solution:

```
insert into student values('Ram',12,98,'Palpa');
insert into student values('Shyam',13,99,'KTM');
insert into student values('Hari',14,88,'PKR');
insert into student values('Rita',15,57,'BRT');
insert into student values('Sita',16,66,'BKT');
insert into student values('Gita',17,29,'KTM');
insert into student values('Anita',18,54,'PKR');
insert into student values('Dinesh',19,49,'BIR');
insert into student values('Kartik',20,34,'JHP');
insert into student values('Tarun',21,39,'PKR');
```

After the execution of above commands the state of the database is:



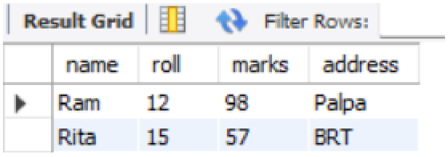
	name	roll	marks	address
▶	Ram	12	98	Palpa
	Shyam	13	99	KTM
	Hari	14	88	PKR
	Rita	15	57	BRT
	Sita	16	66	BKT
	Gita	17	29	KTM
	Anita	18	54	PKR
	Dinesh	19	49	BIR
	Kartik	20	34	JHP
	Tarun	21	39	PKR

c) Write SQL queries to list the details of all the student whose name starts with ‘R’

Solution:

```
select * from student
where name like 'R%';
```

Output:



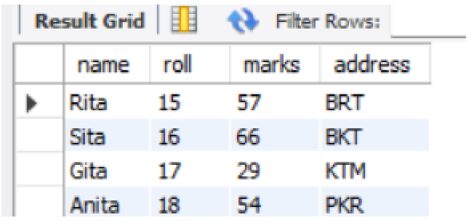
name	roll	marks	address
Ram	12	98	Palpa
Rita	15	57	BRT

d) Write SQL queries to display the details of all the student whose name ends with ‘ita’

Solution:

```
select * from student
where name like '%ita';
```

Output:



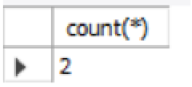
name	roll	marks	address
Rita	15	57	BRT
Sita	16	66	BKT
Gita	17	29	KTM
Anita	18	54	PKR

e) Write SQL queries to count the no. of students whose name starts with ‘R’

Solution:

```
select count(*) from student
where name like 'R%';
```

Output:



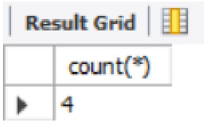
count(*)
2

f) Write SQL queries to count the no. of students whose name ends with ‘ita’

Solution:

```
select count(*) from student
where name like '%ita';
```

Output:



count(*)
4

Lab 6

SQL Queries Set 4

Q$_1$. Consider following tables:

Students

stud#	name	course
100	Fred	PH
200	Dave	CM
300	Bob	CM

Courses

course#	name
PH	Pharmacy
CM	Computing

a. Create the schema for the tables Students and Courses.

Solution:

```
Create table Students(
studno integer primary key,
name varchar(50),
course varchar(4)
);

create table Courses(
courseno varchar(4) primary key,
name varchar(50)
);
```

b. Populate the tables with above indicated values.

Solution:

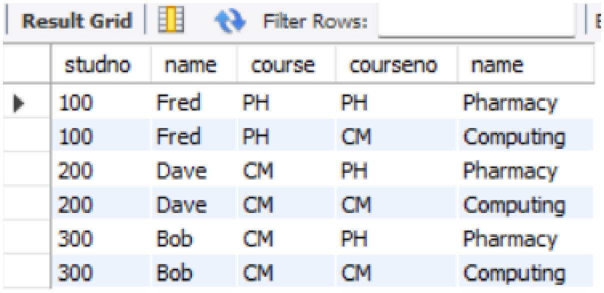
```
insert into Students values(100,'Fred','PH');
insert into Students values(200,'Dave','CM');
insert into Students values(300,'Bob','CM');

insert into Courses values('PH','Pharmacy');
insert into Courses values('CM','Computing');
```

c. **Write SQL query to display the Cartesian product of two tables.**

Solution:

```
select * from Students, Courses;
```

Output:



studno	name	course	courseno	name
100	Fred	PH	PH	Pharmacy
100	Fred	PH	CM	Computing
200	Dave	CM	PH	Pharmacy
200	Dave	CM	CM	Computing
300	Bob	CM	PH	Pharmacy
300	Bob	CM	CM	Computing

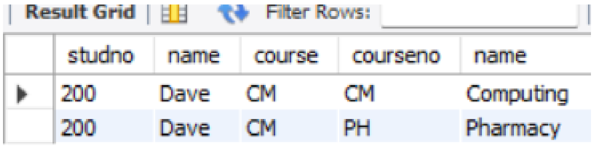
d. **Write SQL query to display the result of the theta join operation**

$$\textit{Students} \bowtie_{\text{stud\#=200}} \text{Courses}$$

Solution:

```
select * from Students, Courses
where studno = 200;
```

Output:



studno	name	course	courseno	name
200	Dave	CM	CM	Computing
200	Dave	CM	PH	Pharmacy

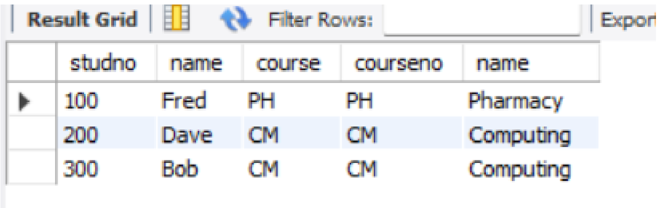
e. **Write SQL query to display the result of the equi join operation**

$$\textit{Students} \bowtie_{\text{course=course\#}} \text{Courses}$$

Solution:

```
select * from Student, Courses
where course=courseno;
```

Output:



studno	name	course	courseno	name
100	Fred	PH	PH	Pharmacy
200	Dave	CM	CM	Computing
300	Bob	CM	CM	Computing

Q2. Consider following tables:

r

a	***b***
a1	b1
a2	b2
a3	b3

s

b	***c***
b1	c1
b2	c2
b4	c4

a. Write SQL query to create the schemas for tables *r* and *s* and to populate the indicated values.

Solution:

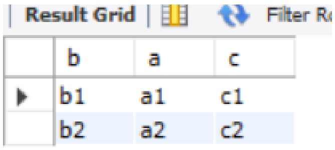
```
create table r(
a varchar(4),
b varchar(4)
);
create table s(
b varchar(4),
c varchar(4)
);
insert into r values('a1','b1');
insert into r values('a2','b2');
insert into r values('a3','b3');
insert into s values('b1','c1');
insert into s values('b2','c2');
insert into s values('b4','c4');
```

b. Write SQL query to display the result of the natural join operation r ⋈ s

Solution:

```
select * from r natural join s;
```

Output:



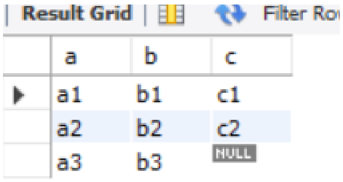
	b	a	c
▶	b1	a1	c1
	b2	a2	c2

c. **Write SQL query to display the result of the left outer join operation r ⟕ s**

Solution:

```
select a,r.b,s.c from r left join s on r.b=s.b;
```

Output:



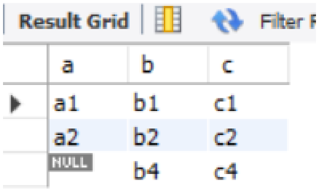
a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	NULL

d. **Write SQL query to display the result of the right outer join operation r ⟖ s**

Solution:

```
select a,s.b,c from r right join s on r.b=s.b;
```

Output:



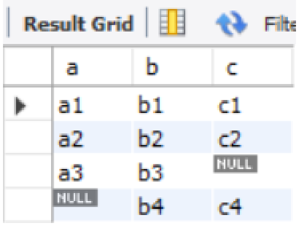
a	b	c
a1	b1	c1
a2	b2	c2
NULL	b4	c4

e. **Write SQL query to display the result of the full outer join operation r ⟗ s**

Solution:

```
create view g as
(select a,r.b,s.c from r left join s on r.b=s.b);

create view h as
(select a,s.b,c from r right join s on r.b=s.b);

select * from g union select * from h;
```

Output:



a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	NULL
NULL	b4	c4

Q3. Consider following tables:

First

id	*name*
1	A
2	B
3	C
4	D

Second

id	*Name*
2	B
3	C
5	E
6	F

a. Write SQL query to create schema for the tables First and Second.

Solution:

```
create table First(
id integer,
name varchar(4)
);

create table Second(
id integer,
name varchar(4)
);
```

b. Write SQL query to populate the indicated values.

Solution:

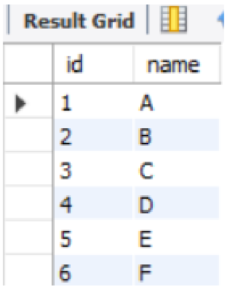
```
insert into First values(1,'A');
insert into First values(2,'B');
insert into First values(3,'C');
insert into First values(4,'D');

insert into Second values(2,'B');
insert into Second values(3,'C');
insert into Second values(5,'E');
insert into Second values(6,'F');
```

c. Write SQL query to find the union of two tables.

Solution:

```
select * from First union Select * from Second;
```

Output:



id	name
1	A
2	B
3	C
4	D
5	E
6	F

d. Write SQL query to find the intersection of the two tables.

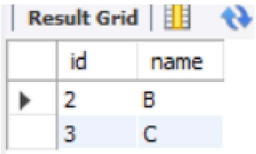
Solution:

```
select * from First intersect Select * from Second;
```

or alternatively following query can be written to generate same output:

```
select * from First where id in(Select id from Second);
```

Output:



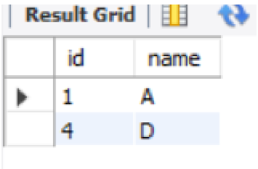
id	name
2	B
3	C

e. Write SQL query to find the difference of the two tables.

Solution:

```
select * from First where id not in(Select id from Second);
```

Output:



id	name
1	A
4	D

Lab 7
SQL Queries Set 5

Q1. Consider following table:

Customers:

ID	Name	Age	Address	Salary
1	Ramesh	32	Kathmandu	20000
2	Kaushik	25	Bhaktapur	15000
3	Chaitali	23	Lalitpur	20000
4	Hardik	25	Pokhara	65000
5	Komal	22	Jhapa	60000
6	Muffy	24	Kathmandu	10000

a. Create the schema for the table `Customers`.

Solution:

```
create database dbms;
use dbms;
create table Customers(
     ID int primary key,
     Name varchar(50),
     Age int,
     Address varchar(50),
     Salary int
);
```

b. Populate the table with above indicated values.

Solution:

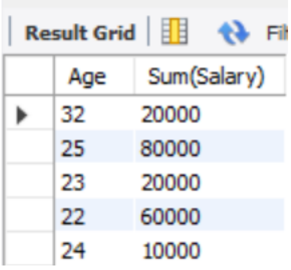
```
insert into Customers values(1,"Ramesh",32,"Kathmandu",20000);
insert into Customers values (2, "Kaushik", 25, "Bhaktapur", 15000);
insert into Customers values (3, "Chaitali", 23,"Lalitpur",20000);
insert into Customers values(4,"Hardik",25,"Pokhata",65000);
insert into Customers values(5,"Komal",22,"Jhapa",60000);
insert into Customers values(6,"Muffy",24,"Kathmandu",10000);
```

c. Write SQL query to display age and salary from Customers of every age.

Solution:

```
select Age, sum(Salary) from Customers
group by Age;
```

Output:



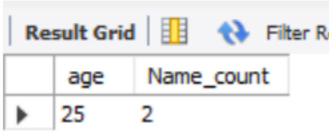
Age	Sum(Salary)
32	20000
25	80000
23	20000
22	60000
24	10000

d. Write SQL query to filter the result using HAVING clause.

Solution:

```
select age,count(*) as Name_count from Customers group by age
having count(*)>1;
```

Output:



age	Name_count
25	2

Q$_2$. Consider following table:

Employee:

EMPNO	**ENAME**	**JOB**	**SAL**	**DEPTNO**
8369	Ram	Clerk	2985	10
8499	Sita	Manager	9870	20
8566	Hari	Salesman	8760	30
8698	Shyam	Salesman	5643	20
8912	Gita	Null	3000	10

a.Create the schema for the table `Employee`.

Solution:

```
create database Csit;
use Csit;
create table Employee(
EMPNO integer,
ENAME varchar(20),
JOB varchar(20),
SAL integer,
DEPTNO integer
);
```

b. Populate the table with above indicated values.

Solution:

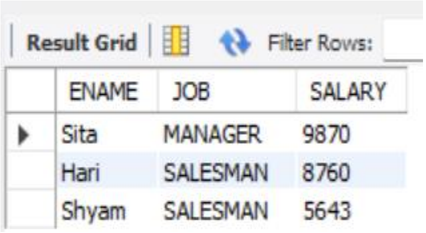
```
insert into Employee values(8369,"Ram","CLERK",2985,10);
insert into Employee values(8499,"Sita","MANAGER",9870,20);
insert into Employee values(8566,"Hari","SALESMAN",8760,30);
insert into Employee values(8698,"Shyam","SALESMAN",5643,20);
insert into Employee values(8912,"Gita","NULL",3000,10);
```

c. Write SQL query using AS clause.

Solution:

```
select ENAME,JOB, SAL as SALARY from Employee where SAL>5000;
```

Output:



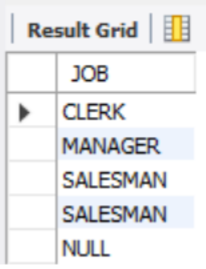
ENAME	JOB	SALARY
Sita	MANAGER	9870
Hari	SALESMAN	8760
Shyam	SALESMAN	5643

d. Write SQL queries using EXISTS clause.

Solution:

```
select JOB from employee where EXISTS(select JOB from Employee
where SAL>1000);
```

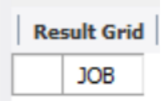
Output:



JOB
CLERK
MANAGER
SALESMAN
SALESMAN
NULL

```
select JOB from employee where EXISTS(select JOB from Employee
where SAL>80000);
```

Output:



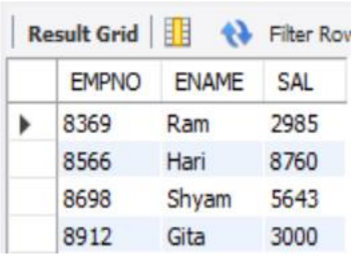
JOB

e. **Write SQL query using SOME clause.**

Solution:

```
select EMPNO, ENAME, SAL
from Employee
where SAL < SOME (
  select SAL
  from Employee
  where DEPTNO = 20
);
```

Output:



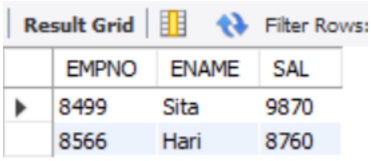
EMPNO	ENAME	SAL
8369	Ram	2985
8566	Hari	8760
8698	Shyam	5643
8912	Gita	3000

f. **Write SQL query using ALL clause.**

Solution:

```
select EMPNO, ENAME, SAL from Employee
where SAL > ALL (
  select AVG(SAL)
  from Employee
);
```

Output:



EMPNO	ENAME	SAL
8499	Sita	9870
8566	Hari	8760

Q3. Consider following table:

Department:

Dept_id	**Dept_name**
A2	Anatomy
B2	Micro-Biology
P1	Pathology
D1	Dermatology
A1	Administration

a.Create the schema for the table `Department`.

Solution:

```
create database DBMS;
use DBMS;
create table Department
(
dept_id varchar(10),
dept_name varchar(50)
);
```

b. Populate the table with above indicated values.

Solution:

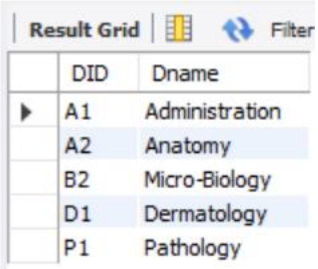
```
insert into Department values("A2","Anatomy");
insert into Department values("B2","Micro-Biology");
insert into Department values("P1","Pathology");
insert into Department values("D1","Dermatology");
insert into Department values("A1","Administration");
```

c.Write SQL query using AS clause.

Solution:

```
select dept_id as DID,dept_name as Dname from Department
order by dept_id;
```

Output:



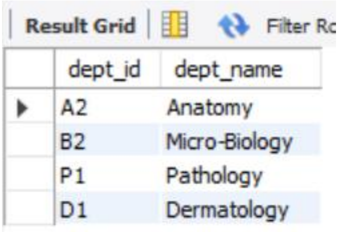
	DID	Dname
▶	A1	Administration
	A2	Anatomy
	B2	Micro-Biology
	D1	Dermatology
	P1	Pathology

d. Write SQL query using ALL clause.

Solution:

```
select * from Department
where dept_id > ALL (
  select dept_id from Employee
  where dept_name = “Administration”
);
```

Ouput:



	dept_id	dept_name
▶	A2	Anatomy
	B2	Micro-Biology
	P1	Pathology
	D1	Dermatology

Q4. Consider following table:

Project:

proj_id	**proj_name**	**start_date**	**deadline**
8000	White Rays	2022-07-07	2022-08-05
8002	Liquid Sky	2022-06-15	2022-12-16
7099	Black Marvel	2022-02-22	2022-06-22
8255	Distruptors	2023-01-23	2023-09-07
8030	Art of Winning	2023-01-01	2023-02-20

a. Create the schema for the table `Project`.

Solution:

```
create database Csit;
use Csit;
create table Project(
 proj_id int,
 proj_name varchar(50),
 start_date date,
 deadline date
 );
```

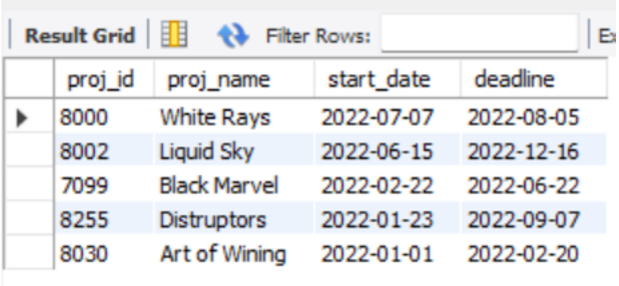
b. Populate the table with above indicated values and display them.

Solution:

```
insert into Project values(8000, "White Rays","2022-07-07",
"2022-08-05");
insert into Project values(8002, "Liquid Sky","2022-06-15",
"2022-12-16");
insert into Project values(7099, "Black Marvel","2022-02-22",
"2022-06-22");
insert into Project values(8255, "Distruptors","2022-01-23",
"2022-09-07");
```

```
insert into Project values(8030, "Art of Wining","2022-01-01",
"2022-02-20");
select * from Project;
```

Output:

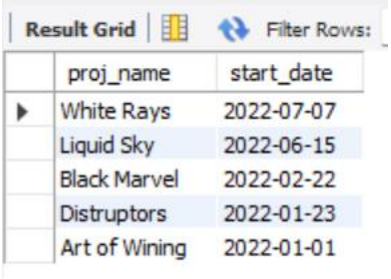


proj_id	proj_name	start_date	deadline
8000	White Rays	2022-07-07	2022-08-05
8002	Liquid Sky	2022-06-15	2022-12-16
7099	Black Marvel	2022-02-22	2022-06-22
8255	Distruptors	2022-01-23	2022-09-07
8030	Art of Wining	2022-01-01	2022-02-20

c. **Write SQL queries using EXISTS clause.**

Solution:

```
select proj_name,start_date from Project where EXISTS
(select proj_name from Project where proj_id>8000);
```



proj_name	start_date
White Rays	2022-07-07
Liquid Sky	2022-06-15
Black Marvel	2022-02-22
Distruptors	2022-01-23
Art of Wining	2022-01-01

Q5. Consider the following database schema:

```
Products (product_id, product_name, category_id, price)
Categories (category_id, category_name)
Orders (order_id,product_id,quantity,order_date)
```

a) Write SQL query to create three tables for given schemas.

Solution:

```
create table Products (
  product_id int primary key,
  product_name varchar(50),
  category_id int,
  price decimal(10, 2)
);

create table Categories (
  category_id int primary key,
  category_name varchar(50)
);

create table Orders (
  order_id int primary key,
  product_id int,
  quantity int,
  order_date date,
  foreign key (product_id) references Products(product_id)
);
```

b) Write SQL queries to populate all the tables.

Solution:

```
insert into Products (product_id, product_name, category_id, price)
values (1, 'Product A', 1, 10.99),
       (2, 'Product B', 1, 15.99),
       (3, 'Product C', 2, 5.99),
       (4, 'Product D', 2, 8.99);
```

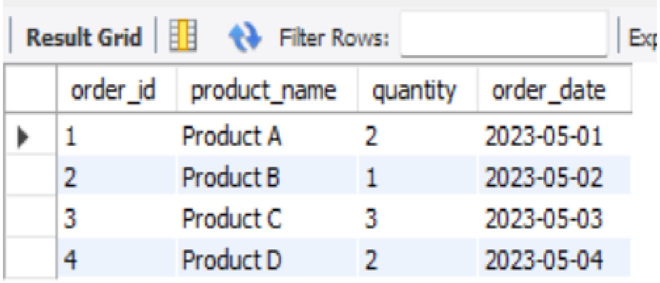
```
insert into Categories (category_id, category_name)
values (1, 'Category X'),
       (2, 'Category Y');
insert into Orders (order_id, product_id, quantity, order_date)
values (1, 1, 2, '2023-05-01'),
       (2, 2, 1, '2023-05-02'),
       (3, 3, 3, '2023-05-03'),
       (4, 4, 2, '2023-05-04');
```

c) Create view `OrderDetails` which includes order_id, product_name, quantity and order_date and display it.

Solution:

```
create view OrderDetails as
select o.order_id, p.product_name, o.quantity, o.order_date
from Orders o join Products p on o.product_id = p.product_id;
select * from OrderDetails;
```

Output:



order_id	product_name	quantity	order_date
1	Product A	2	2023-05-01
2	Product B	1	2023-05-02
3	Product C	3	2023-05-03
4	Product D	2	2023-05-04

d) Develop SQL queries to delete the created view `OrderDetails` from the database.

Solution:

```
drop view if exists OrderDetails;
```

Lab 8
Relational Database Design using ER diagram

Task #1: Draw an ER diagram for **COMPANY** database.

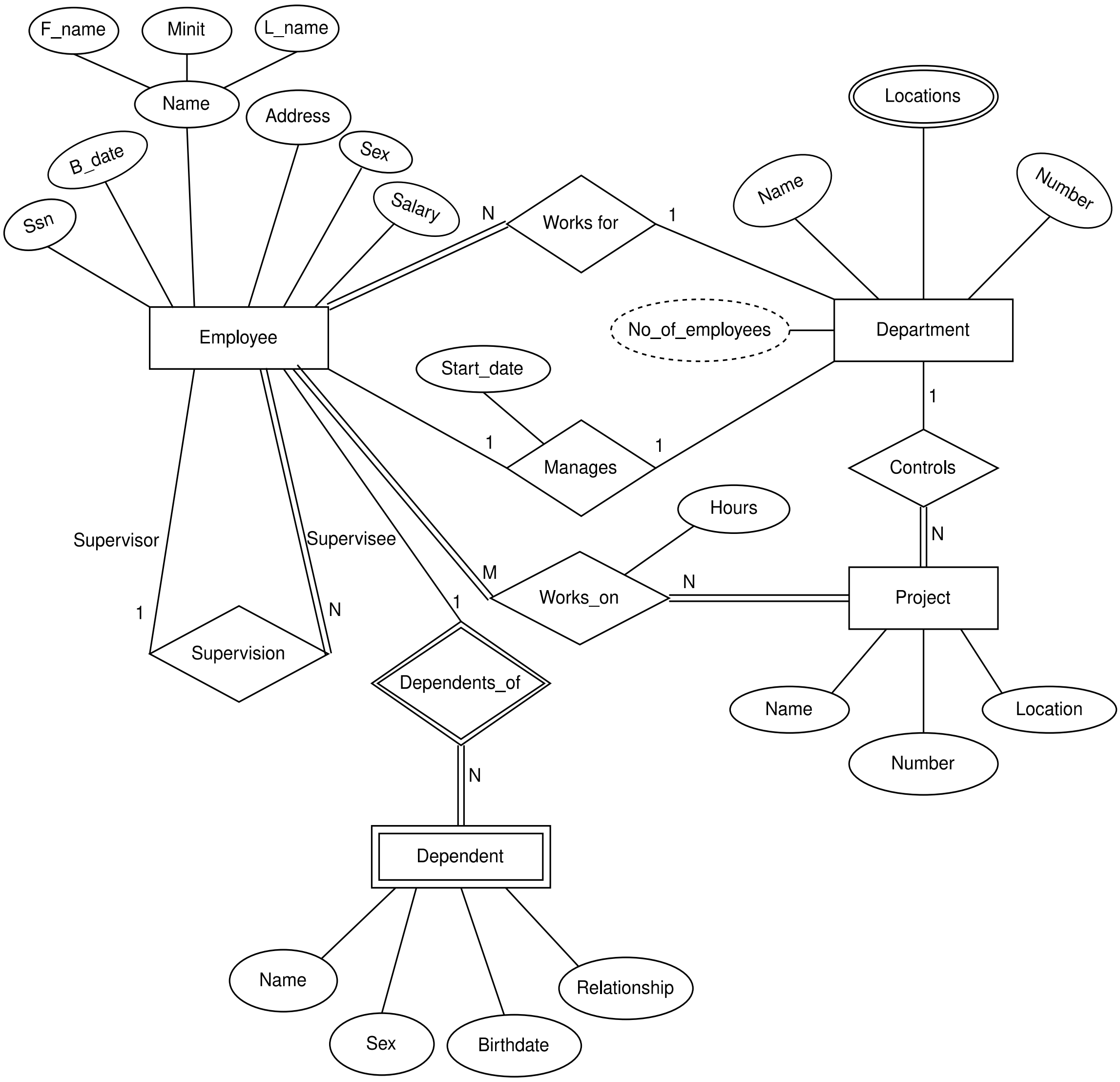


Fig i): ER diagram for COMPANY database

Task #2: Draw an ER diagram for **MOVIE** database.

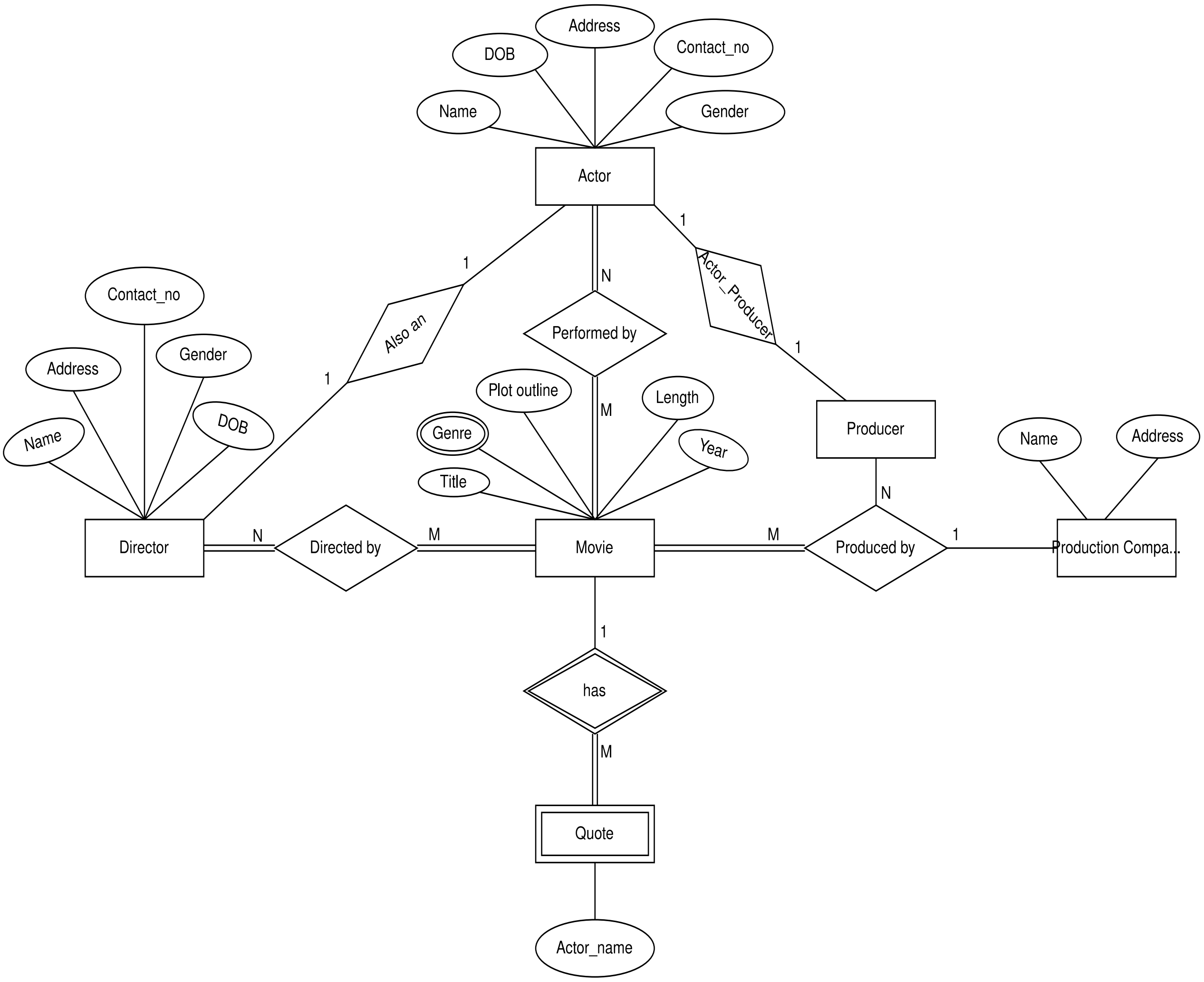


Fig ii): ER diagram for MOVIE database

Task #3: Draw an ER diagram for **AIRLINE RESERVATION SYSTEM** database.

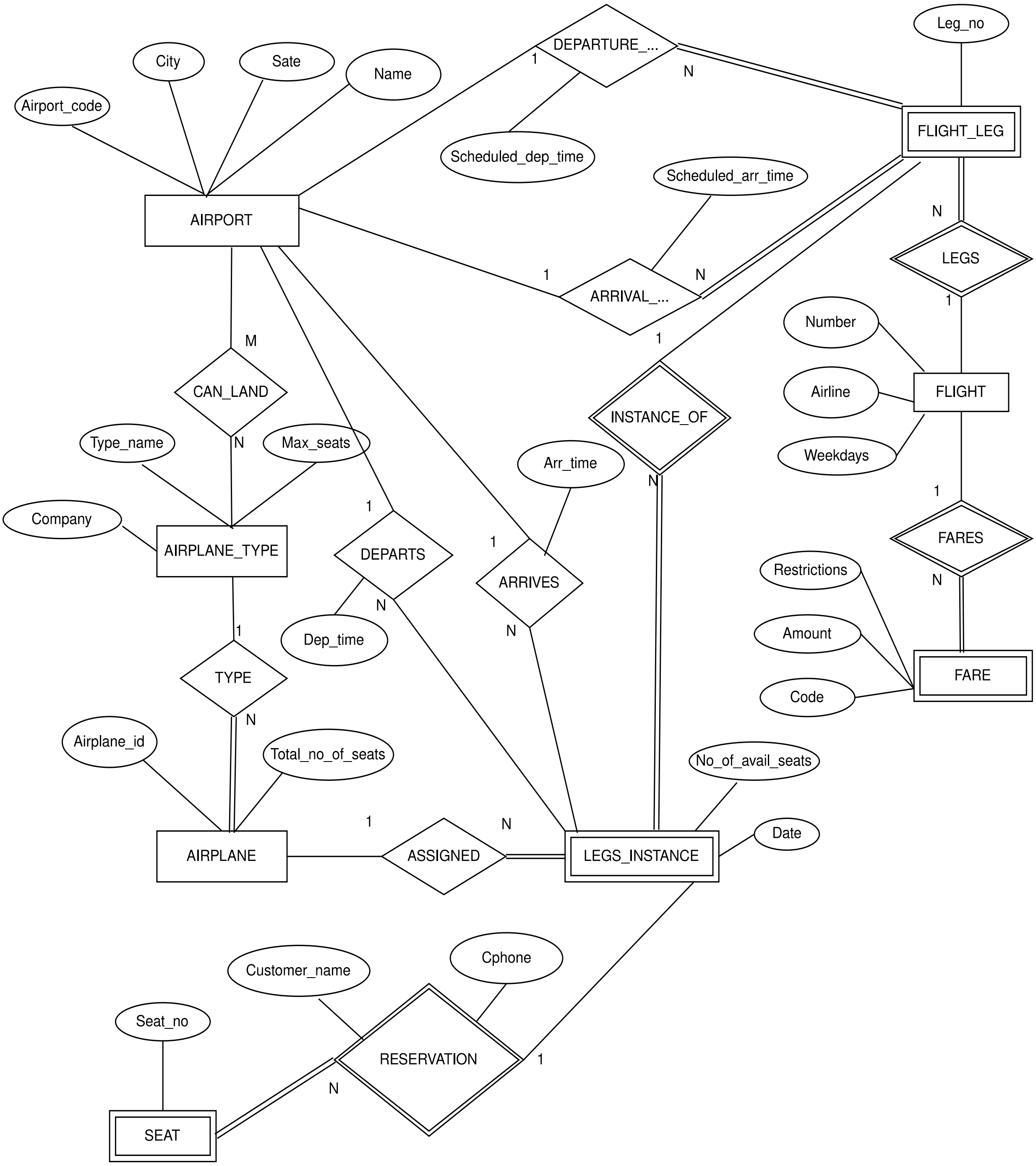


Fig iii): ER diagram for AIRLINE RESERVATION SYSTEM database

Task #4: Draw an ER diagram for **HOSPITAL MANAGEMENT SYSTEM** database.

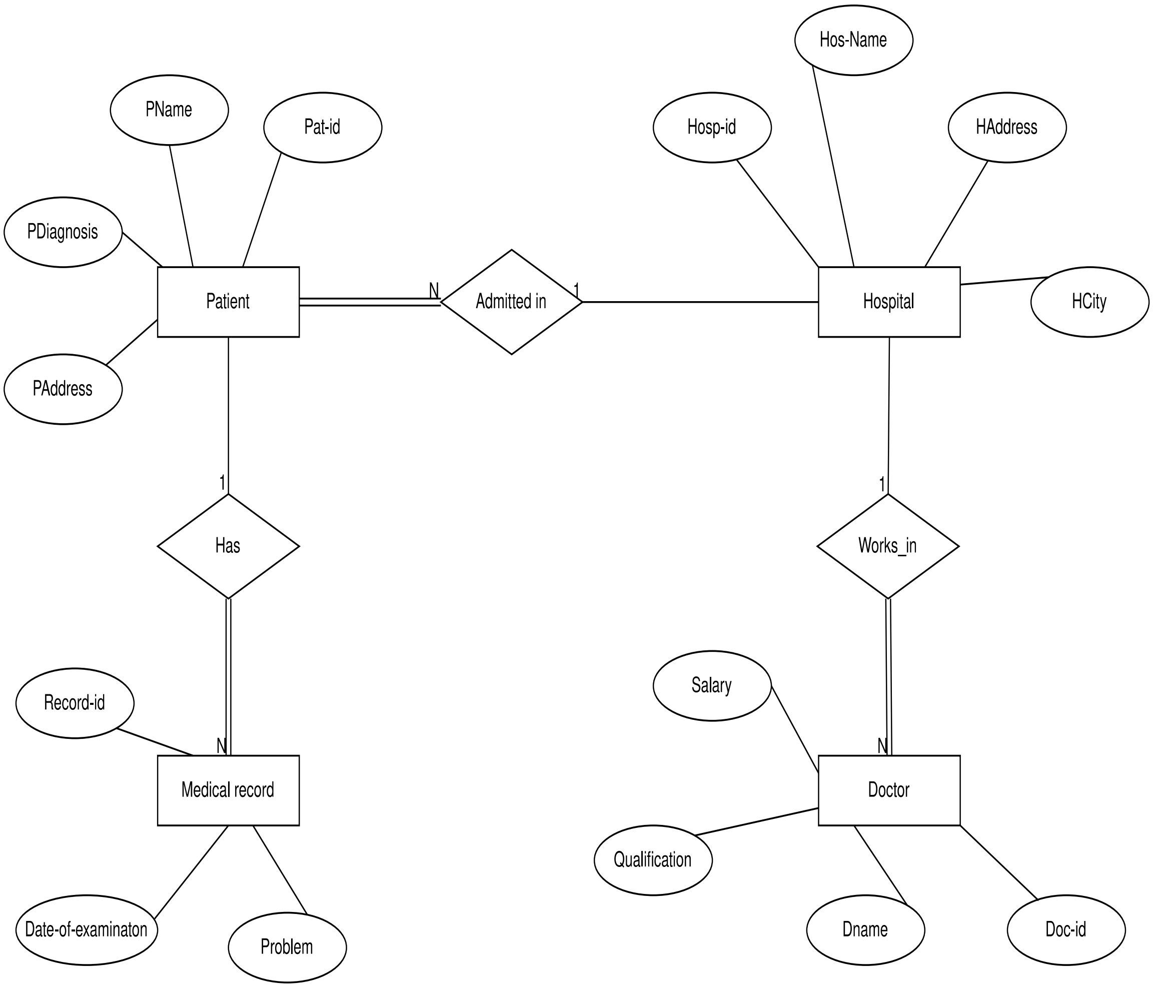


Fig iv): ER diagram for HOSPITAL MANAGEMENT SYSTEM database

Task #5: Draw an ER diagram for **LIBRARY MANAGEMENT SYSTEM** database.

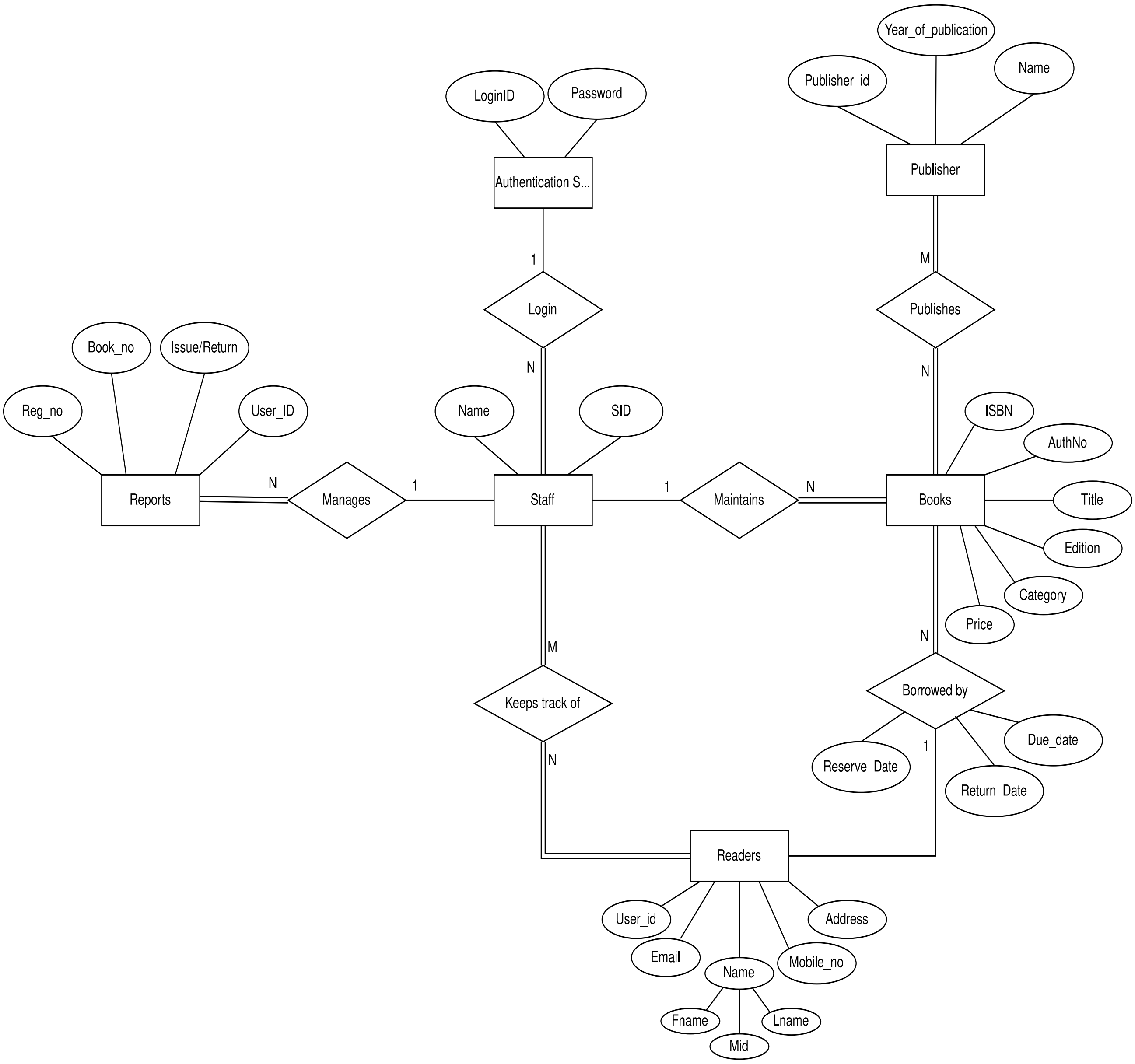


Fig v): ER diagram for LIBRARY MANAGEMENT SYSTEM database

Task #6: Draw an ER diagram for **UNIVERSITY** database.

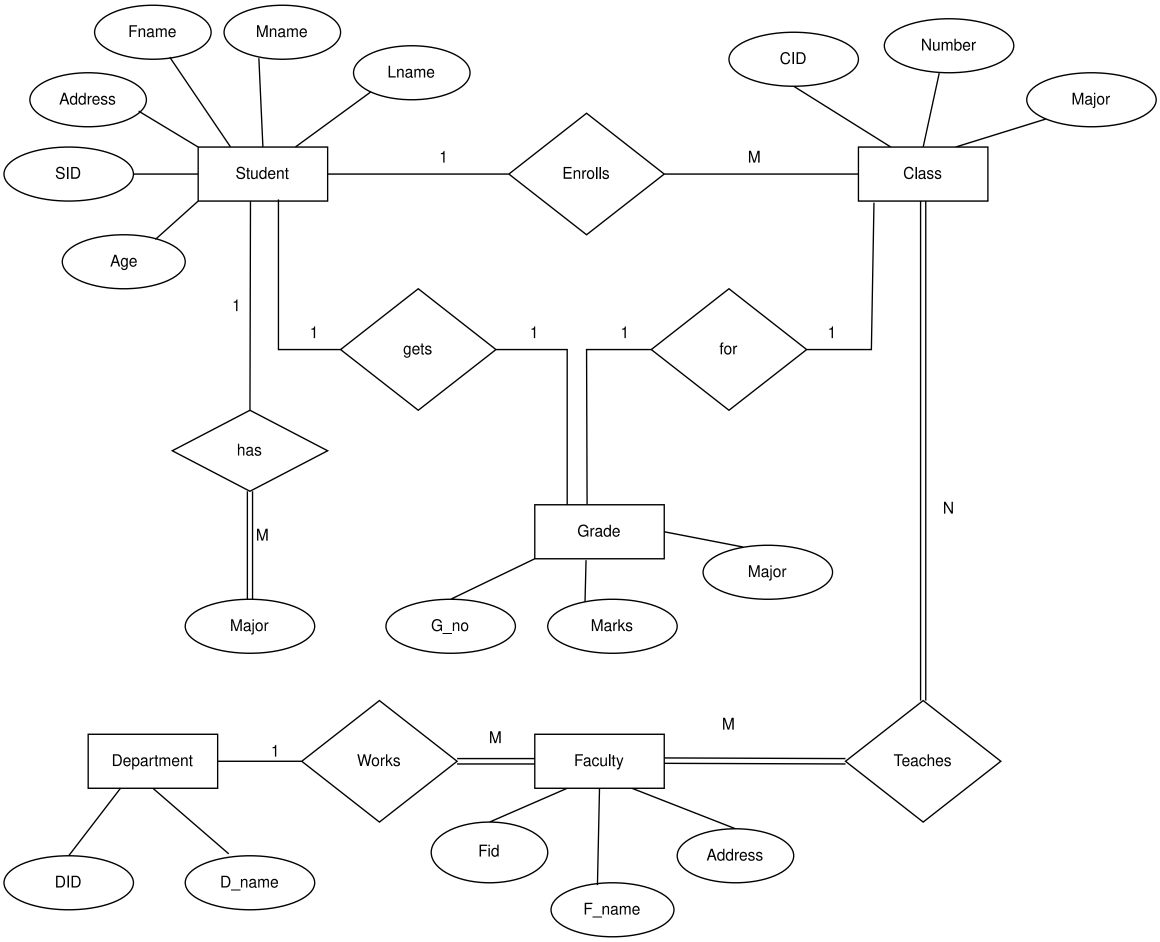


Fig vi): ER diagram for UNIVERSITY database

Task #7: Draw an ER diagram for **BANK** database.

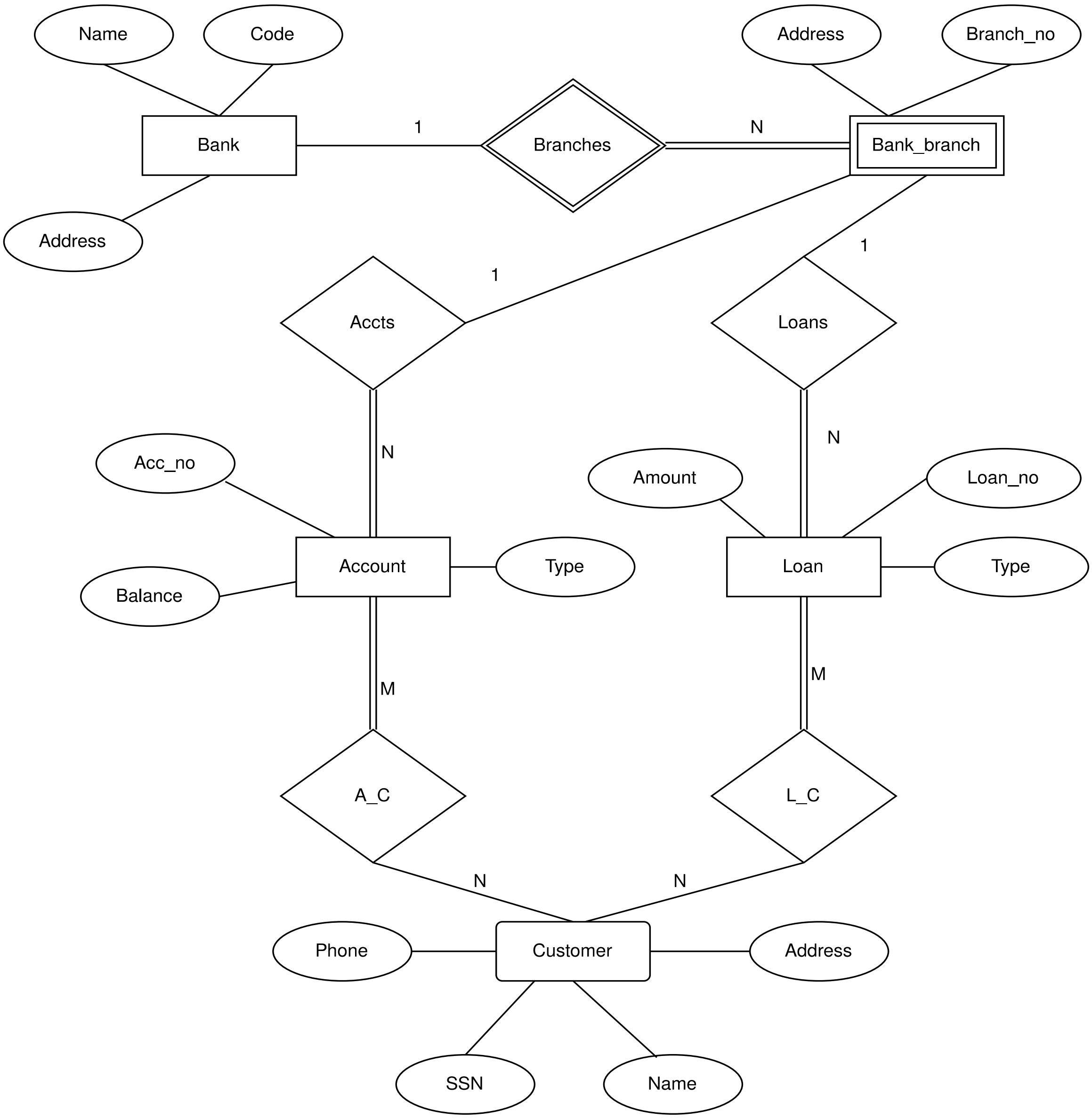


Fig vii): ER diagram for BANK database

Task #8: Draw an ER diagram of any system involving **Specialization and Generalization**.

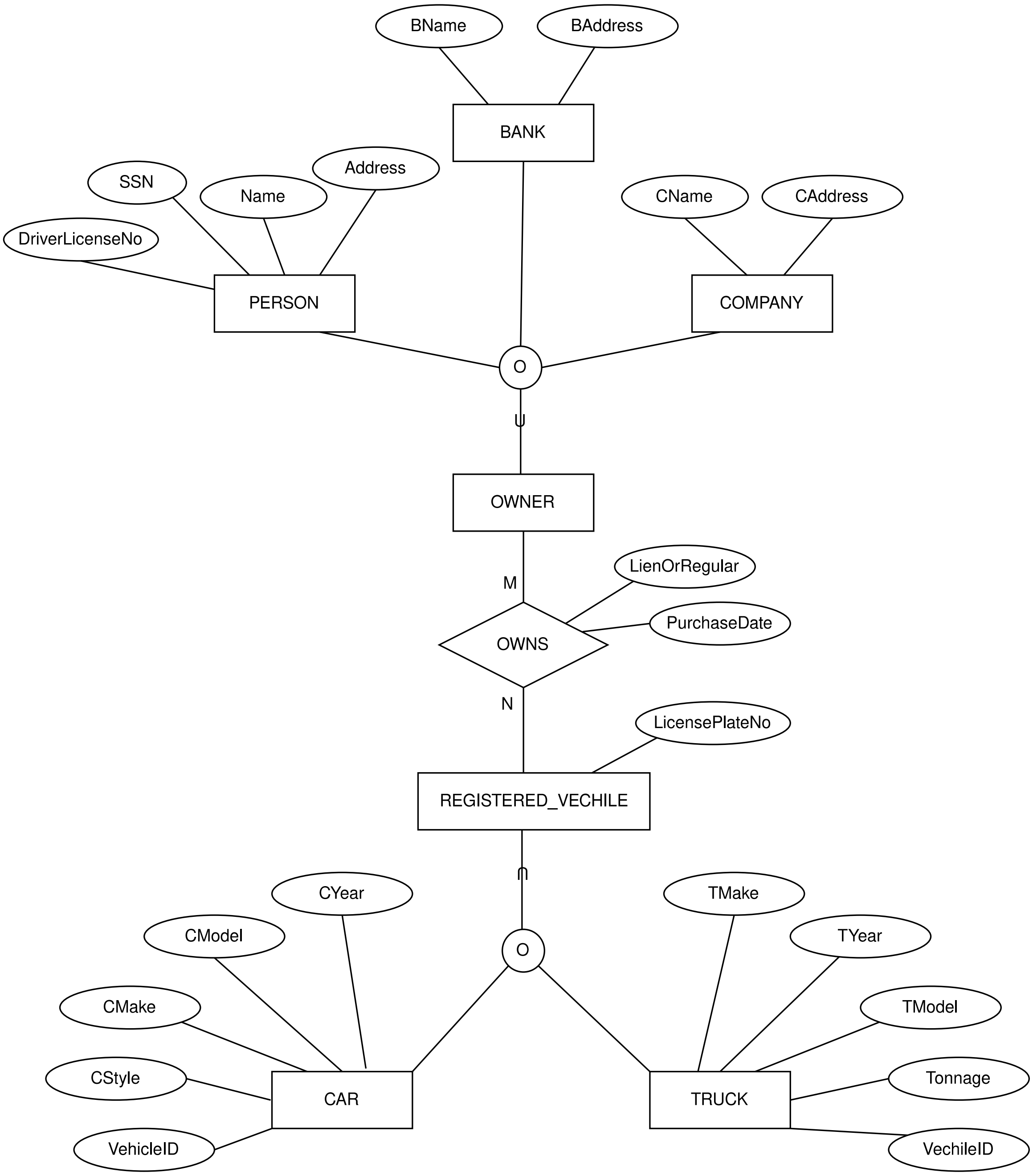


Fig viii): ER diagram of VEHICLE REGISTRATION SYSTEM database involving both Specialization and Generalization

Lab 9

Mini Project

ER-DIAGRAM FOR FOOD BUSINESS:

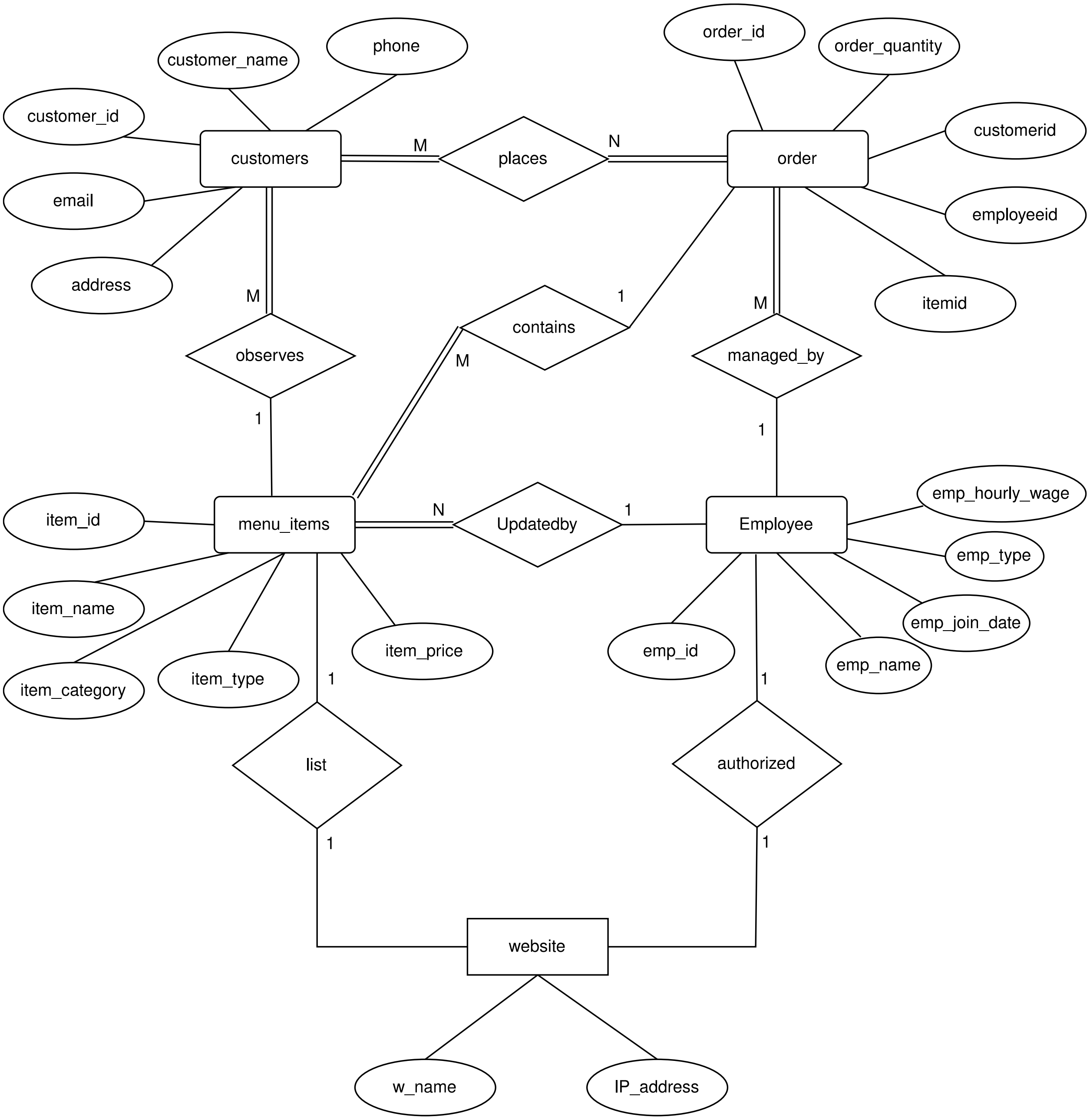


Fig: ER-DIAGRAM FOR FOOD BUSINESS Customer

SCHEMA DIAGRAM:

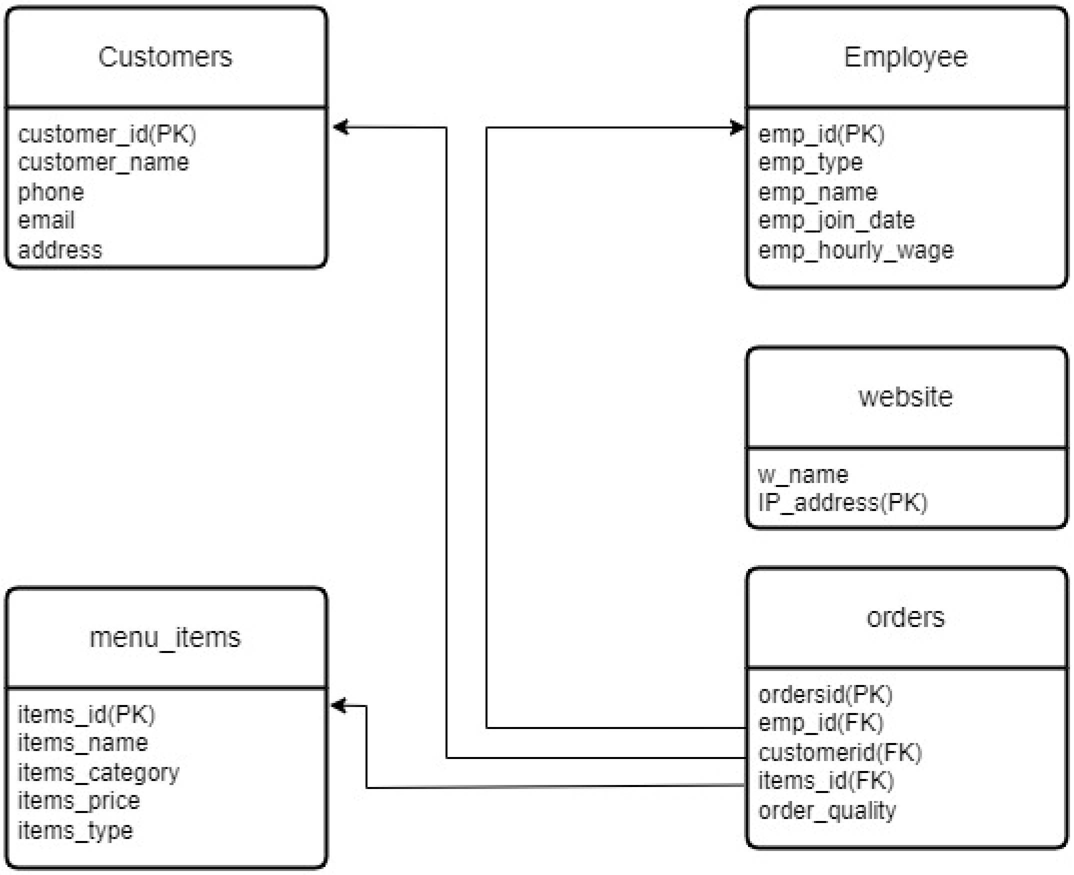


Fig: SCHEMA DIAGRAM of FOOD BUSINESS

RELATIONAL DATABASE MODEL:

Entities and their attributes:

(customer_id, customer_name, address, phone, email)

Order (order_id, itemsid, customerid, emp_id, order_quantity, order_description, price)

Employee (emp_id, emp_name, emp_type, emp_join_date, emp_hourly_wage)

Menu_Items (items_id, items_name, items_category, items_price, items_type)

Website (w_name, IP_address)

RELATIONSHIPS:

1. **Customer-Employee Relationship:**
2. **Customer-Order Relationship:**
3. **Customer-Order Relationship:**
4. **Order-Menu_Item Relationship:**
5. **5Order-Order_Item Relationship:**
6. **Menu_Item-Order_Item Relationship:**