

# COMP- 8115-M50 Database systems

## Assignment – 1

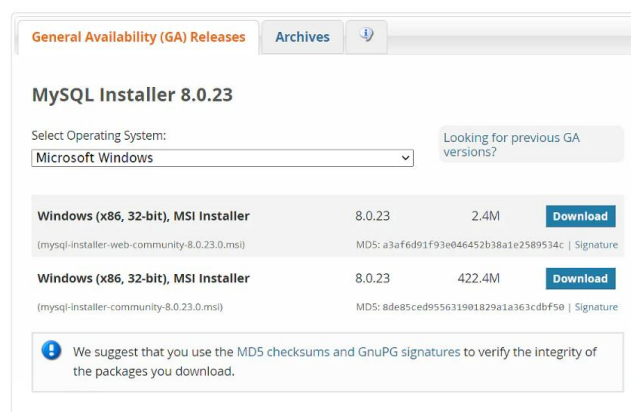
- 1) **Download/install the MySQL Workbench. If you already have it, that is great. Just include the screenshot(s) as proof. [20 pts]**

I have already mentioned in the previous assignment about the installation of the MySQL workbench, but I would attach the screenshots of the installation for your reference.

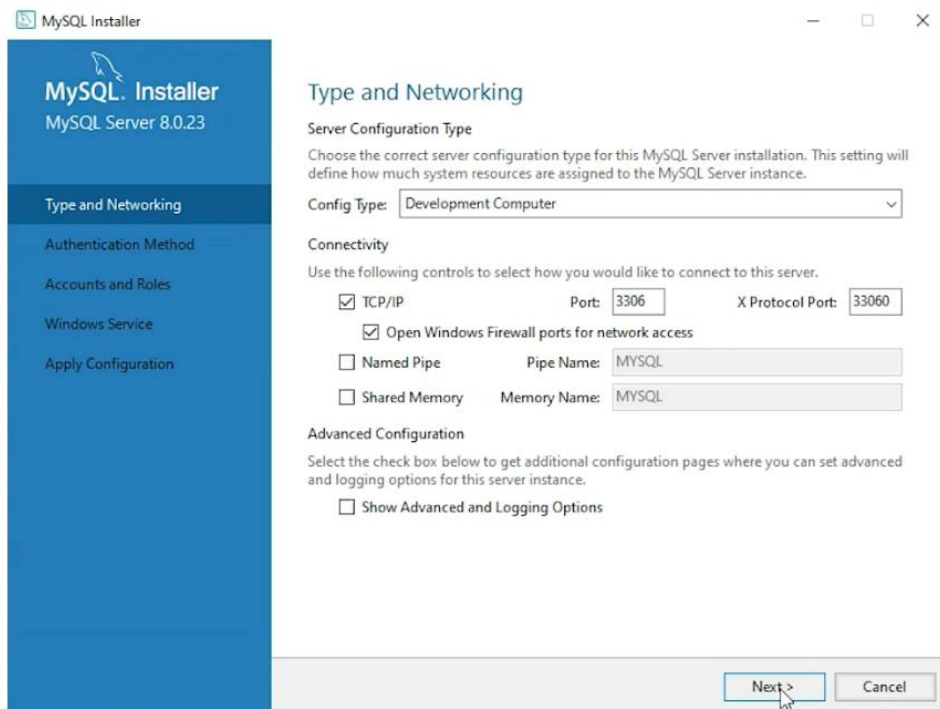
MySQL is a workbench is software used to add functionality and ease to SQL development work. It provides data modelling, SQL development, and various administration tools for configuration.

To install the MYSQL workbench we need to do certain steps

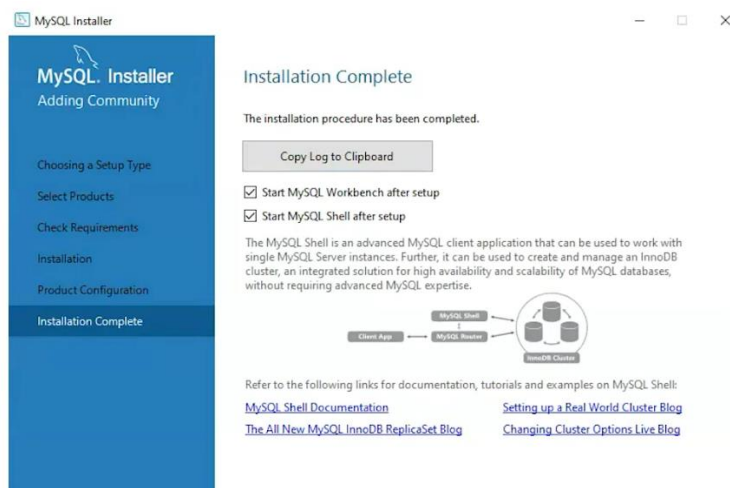
1. Open the MySQL website on a browser
2. Select the downloads option
3. Select MySQL installer for windows
4. Choose the desired installer and click on the download



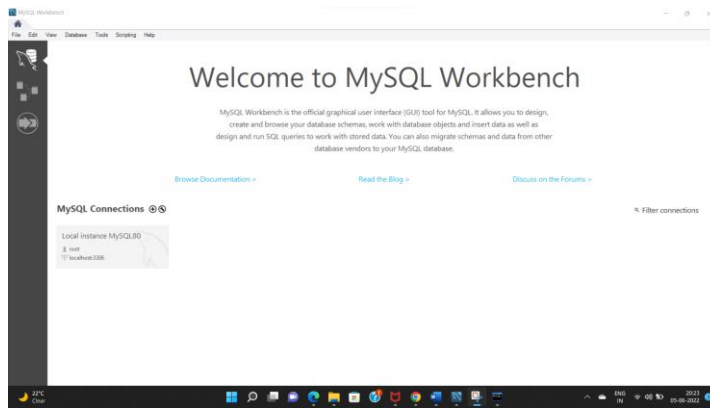
5. After successful completion of downloading, open the installer
6. It will ask for permission; when it does, click Yes. The installer will then open. Now, it will ask to choose the setup type. Here, select Custom.
7. Click on Next. With this, you will install MySQL server, MySQL Workbench, and MySQL shell.
8. Open MySQL Servers, select the server you want to install and move it to the Products/Features to be installed window section. Now, expand Applications, choose MySQL Workbench and MySQL shell. Move both to Products/Features to be installed.
9. Click on the Next button. Now, click on the Execute button to download and install the MySQL server, MySQL Workbench, and the MySQL shell.
10. Once the product is ready to configure, click on Next. Under Type and Networking, go with the default settings and select Next.



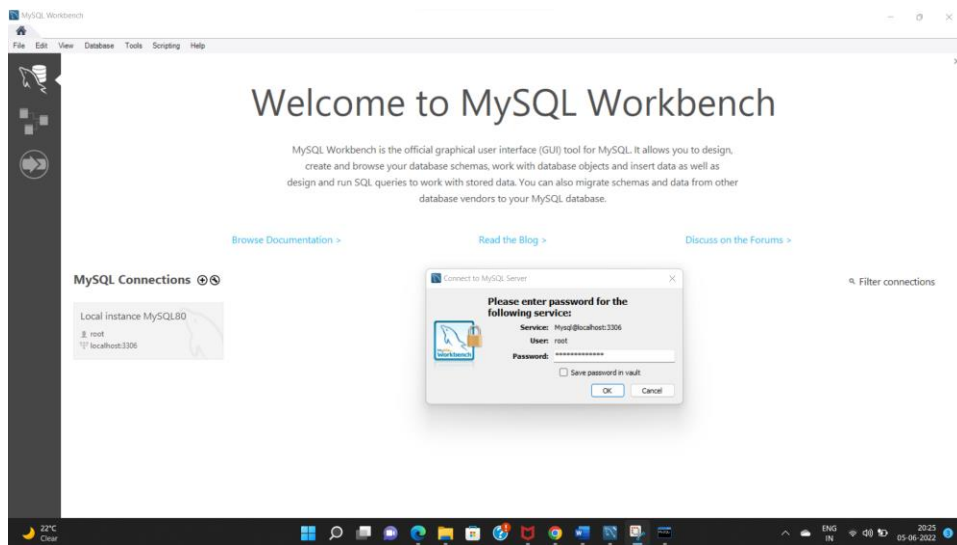
11. For authentication, use the recommended strong password encryption.
12. Set your MySQL Root password and click on next.
13. Go for the default windows service settings and under apply configuration, click on execute. Once the configuration is complete, click on finish.
14. Complete the installation. This will now launch the MySQL Workbench and the MySQL Shell



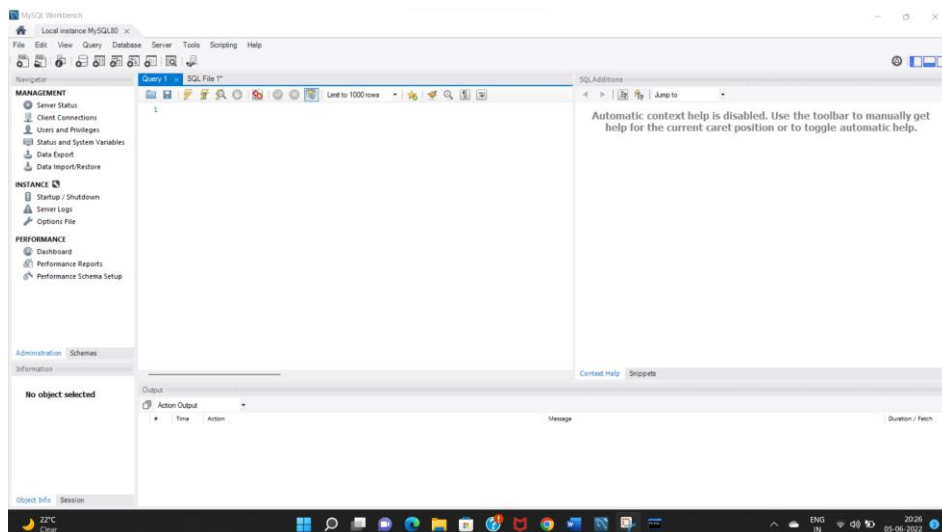
Once MySQL Workbench is installed, select the Local instance, and enter the password.



Enter your password in the blank and store it in the vault for future purposes.



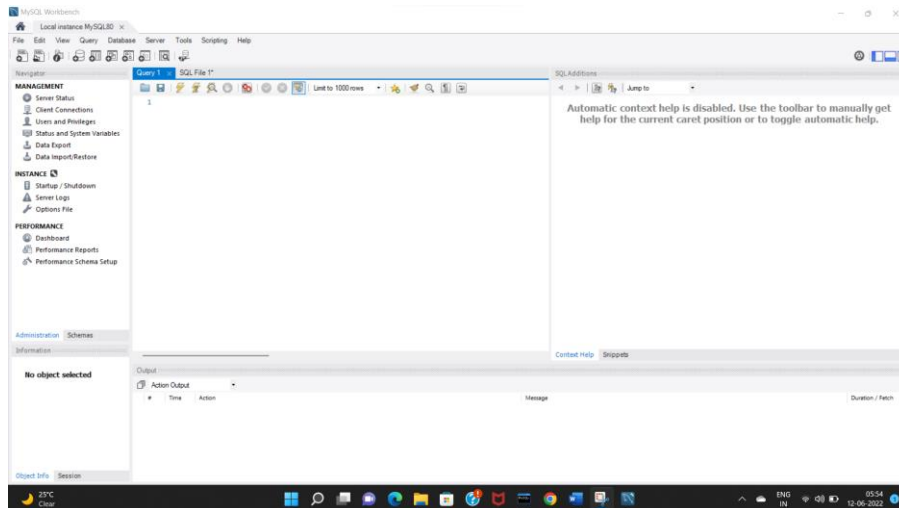
Now, you can use the MySQL query tab to write your SQL queries.



- 2) Using the MySQL Workbench, draw the ER diagram (Figure 3.2) of the COMPANY database. Include the screenshot of your whole ER diagram. [20 pts]

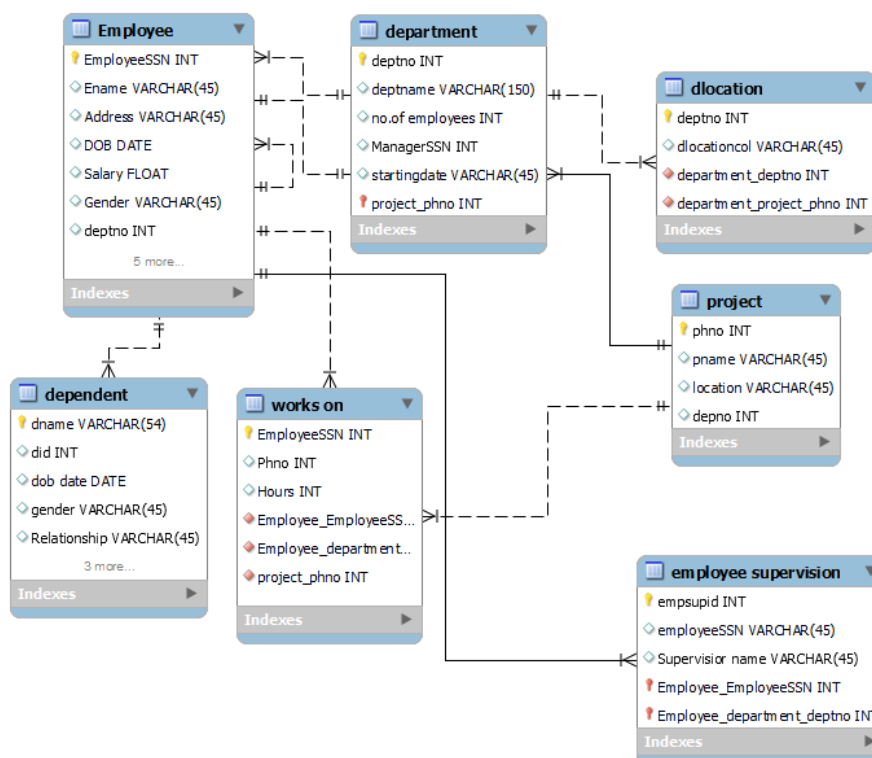
To make the ER diagram for the company database. We need to do and follow certain things.

After you start your localhost by giving your username and password. It pops a window



Now, go to File and select the new model to create a model for the company database.

After successful creation, it would show you something like this



The tables in the company database are

1. Employee
2. Department
3. Dlocation
4. Project
5. Employee supervision
6. Works on
7. Dependent

Explanation:

Here Every table has a relation to other tables. In relation to the employee and the department there are chances that an employee can work in one department. Also, one department can have more than one employee.

Also, a department can have more than one location of working. Also, one project can be handled by more than one department

One employee can work on more than one project. Also, every employee working is dependent on more than one department

The manager controls the department, One department can have more than one project

One employee works on many projects and the number of hours worked by the employee is recorded on a single project

Employees working on projects and their hours are stored on the table

This is some explanation related to the ER diagram.

- 3) **MySQL Workbench provides a powerful feature called forward engineering that allows the database designer to convert the ER design into a schema generation SQL script for one or more target relational databases. Using forward engineering and generating the SQL script. Include screenshots of each step of the forward engineering process. For more information about forwarding engineering, please take a look at this video: <https://www.youtube.com/watch?v=1sqhDJae-xY> [20 pts]**

To do forward engineering

Go to file -> Export -> Forward engineer SQL create script

Click on it




### SQL Export Options

Output SQL Script File:

 ...  
Leave blank to view generated script but not save to a file.

#### SQL Options

- ☐ Generate DROP Statements Before Each CREATE Statement
- ☐ Generate DROP SCHEMA
- ☐ Sort Tables Alphabetically 
- ☐ Skip Creation of FOREIGN KEYS
- ☐ Skip creation of FK Indexes as well
- ☐ Omit Schema Qualifier in Object Names
- ☐ Generate USE statements
- ☐ Generate Separate CREATE INDEX Statements
- ☐ Add SHOW WARNINGS After Every DDL Statement
- ☐ Do Not Create Users. Only Export Privileges
- ☐ Don't create view placeholder tables.
- ☐ Generate INSERT Statements for Tables
- ☐ Disable FK checks for inserts
- ☐ Create triggers after inserts

Back

Next

Cancel

Now tick the below items




### SQL Export Options

Output SQL Script File:

 ...  
Leave blank to view generated script but not save to a file.

#### SQL Options

- ☒ Generate DROP Statements Before Each CREATE Statement
- ☒ Generate DROP SCHEMA
- ☐ Sort Tables Alphabetically 
- ☐ Skip Creation of FOREIGN KEYS
- ☐ Skip creation of FK Indexes as well
- ☐ Omit Schema Qualifier in Object Names
- ☐ Generate USE statements
- ☐ Generate Separate CREATE INDEX Statements
- ☐ Add SHOW WARNINGS After Every DDL Statement
- ☐ Do Not Create Users. Only Export Privileges
- ☐ Don't create view placeholder tables.
- ☒ Generate INSERT Statements for Tables
- ☐ Disable FK checks for inserts
- ☐ Create triggers after inserts

Back

Next

Cancel

Forward Engineer SQL Script


SQL Export Options

Filter Objects

Review SQL Script

### SQL Object Export Filter


To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.



☒ Export MySQL Table Objects
 

Show Filter


7 Total Objects, 7 Selected



☐ Export MySQL View Objects
 

Show Filter

0 Total Objects, 0 Selected



☐ Export MySQL Routine Objects
 

Show Filter

0 Total Objects, 0 Selected

☐ Export MySQL Trigger Objects

Show Filter

0 Total Objects, 0 Selected

☐ Export User Objects

Show Filter

0 Total Objects, 0 Selected

Back

Next

Cancel

Forward Engineer SQL Script

SQL Export Options

Filter Objects

Review SQL Script

### Review Generated Script

Review the generated script.

```

1  -- MySQL Script generated by MySQL Workbench
2  -- Sun Jun 12 12:05:58 2022
3  -- Model: New Model   Version: 1.0
4  -- MySQL Workbench Forward Engineering
5
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CH
8  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
9
10 -----
11 -- Schema companydata
12 -----
13 DROP SCHEMA IF EXISTS `companydata` ;
14
15 -----
16 -- Schema companydata
17 -----
18 CREATE SCHEMA IF NOT EXISTS `companydata` DEFAULT CHARACTER SET utf8 ;
19 USE `companydata` ;
20

```

Save to Other File...

Copy to Clipboard

Back

Finish

Cancel

Now saving the Script to other file.

## SQL Script

```
-- MySQL Script generated by MySQL Workbench
-- Sun Jun 12 12:05:58 2022
-- Model: New Model   Version: 1.0
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO
_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-- -----
-- Schema companydata
-- -----
```

```
DROP SCHEMA IF EXISTS `companydata` ;
```

```
-- -----
-- Schema companydata
-- -----
```

```
CREATE SCHEMA IF NOT EXISTS `companydata` DEFAULT CHARACTER SET utf8 ;
USE `companydata` ;
```

```
-- -----
-- Table `companydata`.`project`
-- -----
```

```
DROP TABLE IF EXISTS `companydata`.`project` ;
```

```
CREATE TABLE IF NOT EXISTS `companydata`.`project` (
  `phno` INT NOT NULL,
  `pname` VARCHAR(45) NULL,
  `location` VARCHAR(45) GENERATED ALWAYS AS () VIRTUAL,
  `depno` INT NULL,
  PRIMARY KEY (`phno`))
ENGINE = InnoDB;
```

```
-- -----
-- Table `companydata`.`department`
-- -----
```

```
DROP TABLE IF EXISTS `companydata`.`department` ;
```

```
CREATE TABLE IF NOT EXISTS `companydata`.`department` (
  `deptno` INT NOT NULL,
  `deptname` VARCHAR(150) NULL,
  `no.of employees` INT NULL,
```



```

`ManagerSSN` INT NULL,
`startingdate` VARCHAR(45) NULL,
`project_phno` INT NOT NULL,
PRIMARY KEY (`deptno`, `project_phno`),
INDEX `fk_department_project1_idx` (`project_phno` ASC) VISIBLE,
CONSTRAINT `fk_department_project1`
  FOREIGN KEY (`project_phno`)
    REFERENCES `companydata`.`project` (`phno`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `companydata`.`Employee`
-----

```

```

DROP TABLE IF EXISTS `companydata`.`Employee` ;

```

```

CREATE TABLE IF NOT EXISTS `companydata`.`Employee` (
  `EmployeeSSN` INT NOT NULL,
  `Ename` VARCHAR(45) NULL,
  `Address` VARCHAR(45) NULL,
  `DOB` DATE NULL,
  `Salary` FLOAT NULL,
  `Gender` VARCHAR(45) NULL,
  `deptno` INT NULL,
  `department_deptno` INT NOT NULL,
  `Employee_EmployeeSSN` INT NOT NULL,
  `Employee_department_deptno` INT NOT NULL,
  `department_deptno1` INT NOT NULL,
  `department_deptno2` INT NOT NULL,
  PRIMARY KEY (`EmployeeSSN`, `department_deptno`),
  INDEX `fk_Employee_Employee1_idx` (`Employee_EmployeeSSN` ASC,
`Employee_department_deptno` ASC) VISIBLE,
  INDEX `fk_Employee_department1_idx` (`department_deptno1` ASC) VISIBLE,
  INDEX `fk_Employee_department2_idx` (`department_deptno2` ASC) VISIBLE,
  CONSTRAINT `fk_Employee_Employee1`
    FOREIGN KEY (`Employee_EmployeeSSN`, `Employee_department_deptno`)
      REFERENCES `companydata`.`Employee` (`EmployeeSSN`, `department_deptno`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Employee_department1`
    FOREIGN KEY (`department_deptno1`)
      REFERENCES `companydata`.`department` (`deptno`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Employee_department2`
    FOREIGN KEY (`department_deptno2`)

```

```

REFERENCES `companydata`.`department` (`deptno`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `companydata`.`dlocation`
-----

```

```

DROP TABLE IF EXISTS `companydata`.`dlocation` ;

```

```

CREATE TABLE IF NOT EXISTS `companydata`.`dlocation` (
  `deptno` INT NOT NULL,
  `dlocationcol` VARCHAR(45) NULL,
  `department_deptno` INT NOT NULL,
  `department_project_phno` INT NOT NULL,
  PRIMARY KEY (`deptno`),
  INDEX `fk_dlocation_department1_idx` (`department_deptno` ASC,
  `department_project_phno` ASC) VISIBLE,
  CONSTRAINT `fk_dlocation_department1`
    FOREIGN KEY (`department_deptno`, `department_project_phno`)
    REFERENCES `companydata`.`department` (`deptno`, `project_phno`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `companydata`.`employee supervision`
-----

```

```

DROP TABLE IF EXISTS `companydata`.`employee supervision` ;

```

```

CREATE TABLE IF NOT EXISTS `companydata`.`employee supervision` (
  `empsupid` INT NOT NULL,
  `employeeSSN` VARCHAR(45) NULL,
  `Supervisor name` VARCHAR(45) NULL,
  `Employee_EmployeeSSN` INT NOT NULL,
  `Employee_department_deptno` INT NOT NULL,
  PRIMARY KEY (`empsupid`, `Employee_EmployeeSSN`, `Employee_department_deptno`),
  INDEX `fk_employee supervision_Employee1_idx` (`Employee_EmployeeSSN` ASC,
  `Employee_department_deptno` ASC) VISIBLE,
  CONSTRAINT `fk_employee supervision_Employee1`
    FOREIGN KEY (`Employee_EmployeeSSN`, `Employee_department_deptno`)
    REFERENCES `companydata`.`Employee` (`EmployeeSSN`, `department_deptno`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

-----  
-- Table `companydata`.`works on`  
-----

DROP TABLE IF EXISTS `companydata`.`works on` ;

CREATE TABLE IF NOT EXISTS `companydata`.`works on` (  
 `EmployeeSSN` INT NOT NULL,  
 `Phno` INT NULL,  
 `Hours` INT NULL,  
 `Employee\_EmployeeSSN` INT NOT NULL,  
 `Employee\_department\_deptno` INT NOT NULL,  
 `project\_phno` INT NOT NULL,  
 PRIMARY KEY (`EmployeeSSN`),  
 INDEX `fk\_works on\_Employee1\_idx` (`Employee\_EmployeeSSN` ASC,  
 `Employee\_department\_deptno` ASC) VISIBLE,  
 INDEX `fk\_works on\_project1\_idx` (`project\_phno` ASC) VISIBLE,  
 CONSTRAINT `fk\_works on\_Employee1`  
 FOREIGN KEY (`Employee\_EmployeeSSN`, `Employee\_department\_deptno`)  
 REFERENCES `companydata`.`Employee` (`EmployeeSSN`, `department\_deptno`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION,  
 CONSTRAINT `fk\_works on\_project1`  
 FOREIGN KEY (`project\_phno`)  
 REFERENCES `companydata`.`project` (`phno`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION)  
ENGINE = InnoDB;

-----  
-- Table `companydata`.`dependent`  
-----

DROP TABLE IF EXISTS `companydata`.`dependent` ;

CREATE TABLE IF NOT EXISTS `companydata`.`dependent` (  
 `dname` VARCHAR(54) NOT NULL,  
 `did` INT NULL,  
 `dob date` DATE NULL,  
 `gender` VARCHAR(45) NULL,  
 `Relationship` VARCHAR(45) NULL,  
 `EmployeeSSN` INT NULL,  
 `Employee\_EmployeeSSN` INT NOT NULL,  
 `Employee\_department\_deptno` INT NOT NULL,  
 PRIMARY KEY (`dname`),  
 INDEX `fk\_dependent\_Employee1\_idx` (`Employee\_EmployeeSSN` ASC,  
 `Employee\_department\_deptno` ASC) VISIBLE,  
 CONSTRAINT `fk\_dependent\_Employee1`

```

FOREIGN KEY (`Employee_EmployeeSSN`, `Employee_department_deptno`)
REFERENCES `companydata`.`Employee` (`EmployeeSSN`, `department_deptno`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

- 4) Cardinality ratios often dictate the detailed design of a database. The cardinality ratio depends on the real-world meaning of the entity types involved and is defined by the specific application. For the binary relationships below, suggest cardinality ratios based on the common-sense meaning of the entity types. Clearly state any assumptions you make.

	Entity 1	Cardinality Ratio	Entity 2
1.	Student		SocialSecurityCard
2.	Student		Teacher
3.	ClassRoom		Wall
4.	Country		CurrentPresident
5.	Course		TextBook
6.	Item (that can be found in an order)		Order
7.	Student		Class
8.	Class		Instructor
9.	Instructor		Office
10.	E-bay Auction item		E-bay bid

So, now let's get the Cardinality ratio for the two entities

- 1) Entity 1: Student  
Entity 2: Social Security Card  
Cardinality Ratio: 1 to Many  
Reason: Here one student has more than one social security card legally with the same unique social security number and every social security number belongs to a student uniquely
- 2) Entity 1: Student  
Entity 2: Teacher  
Cardinality Ratio: Many to Many  
  
Reason: It's obvious that in general Students are taught by many teachers and a teacher teaches many students. It's like a vice versa.
- 3) Entity 1: Classroom

Entity 2: Wall

Cardinality Ratio: Many to Many

Reason: It's important to remember that the wall is usually and always shared by adjacent rooms.

4) Entity 1: Country

Entity 2: Current President

Cardinality Ratio: 1 to 1

Reason: Under normal situations, one country has only one president at a time.

5) Entity 1: Course

Entity 2: Textbook

Cardinality Ratio: Many to Many

Reason: A course might have many textbooks and also one textbook can be prescribed for different courses

6) Entity1: Item

Entity 2: Order

Cardinality Ratio: Many to Many

Reason: Assuming that the same item can appear in different orders

7) Entity1: Student

Entity 2: Classes

Cardinality Ratio: Many to Many

Reason: One student may take several classes and every class usually has several students.

8) Entity1: Class

Entity 2: Instructor

Cardinality Ratio: Many-to-1

Reason: Every class has a unique instructor. Here instructors were allowed to team to teach. this will be many-many

9) Entity1: Instructor

Entity2: Office

Cardinality Ration: 1 to 1

Reason: Assume every instructor has only one office and it is not shared. In the case of offices that are being shared by 2, the relationship would be 2 to 1 whereas if the instructor has a joint appointment and offices in both the department, then the relationship will be 1-2. In the general case, it would be many to many

10) Entity1: E-bay Auction item

Entity2: E-bay bid

Cardinality Ratio: one to one

Reason: One item has many bids and bid is unique to an item