**CMR TECHNICAL CAMPUS**

**UGC AUTONOMOUS**

**B. Tech - IV Semester, II-Lab Internal Examinations – June-2022**

**Database Management System [20CS407PC]**

**CSE(AI&ML)**

1 a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

**Create table sailors(sid int PRIMARY KEY,sname varchar(20),rating int,age int);**

Boats(bid,bname,color);

**Create table boats(bid int PRIMARY KEY,bname varchar(30),color varchar(10));**

Reserves(sid,bid,day);

**Create table reserves(sid int,bid int,day date,FOREIGN KEY(sid) REFERENCE sailors(sid), FOREIGN KEY(bid) REFERENCES boats(bid));**

b) Write a SQL Query to find name of the sailors who have reserved both red or green

boat

**select s.sname from sailors s ,reserves r,boats b where s.sid=r.sid and r.bid=b.bid and b.color='red' union select s2.sname from sailors s2,boats b2,reserves r2 where s2.sid=r2.sid and r2.bid=b2.bid and b2.color='green';**

c) Write a SQL Query to find the name of the sailors who have reserved boat bid 103

**select s.sname from sailors s where s.sid in(select r.sid from reserves r where r.bid=103);**

d) Write a SQL Query to find the name of the sailors who have not reserved boat bid

104

**select s.sname from sailors s where not exists (select \* from reserves r where r.bid=104 and r.sid=s.sid);**

e) Write a SQL Query to find all SIDs of sailors who have a rating of 10 or reserved

boat 104

**select s.sid from sailors s where s.rating=10 union select r.sid from reserves r where r.bid=104;**

2 a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

Boats(bid,bname,color);

Reserves(sid,bid,day);

b) Write a SQL Query to display average age of all sailors

**select avg (s.age) from sailors s;**

c) Write a SQL Query to display number of sailors in the sailors table

**select count(\*) from sailors s;**

d) Write a SQL Query to display name of the sailor who is older than all sailors

**select s.sname,max(s.age) from sailors s;**

e) Write a SQL Query to find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two sailors.

**select s.rating,avg(s.age) as average from sailors s where s.age>=18 group by s.rating having 1<(select count(\*) from sailors s2 where s.rating = s2.rating);**

3. a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

Boats(bid,bname,color);

Reserves(sid,bid,day);

b) Write a SQL query to create a view with the name young\_sailors (age <30)

Young\_sailors(sid,sname,age)

**Create view young-sailors As select sname from sailors where age<30;**

c) Write a SQL query to create a view with the name old\_sailors (age >30)

old\_sailors(sid,sname,age)

**create view old-sailors As,select sname from sailors where age>30;**

d) Write a SQL query to delete young\_sailors view.

**drop view young\_sailors view;**

e) Write a SQL query to delete old\_sailors view.

**drop view old\_sailors view;**

4. a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

Boats(bid,bname,color);

Reserves(sid,bid,day);

b) What are triggers? Explain with detailed syntax for creating triggers?

**Trigger:** A Trigger is a named database object which defines some action that the database should take when some databases related event occurs. Triggers are executed when you issues a data manipulation command like INSERT, DELETE, UPDATE on a table for which the trigger has been created. They are automatically executed and also transparent to the user. But for creating the trigger the user must have the CREATE TRIGGER privilege. In this section we will describe you about the syntax to create and drop the triggers and describe you some examples of how to use them.

**Create Trigger:** The general syntax of CREATE TRIGGER is:

CREATE TRIGGER trigger name trigger time trigger event ON tbl\_name FOR EACH ROW trigger statement.

By using above statement we can create the new trigger. The trigger can associate only with the table name and that must be refer to a permanent table. Trigger time means trigger action time. It can be BEFORE or AFTER. It is used to define that the trigger fires before or after the statement that executed it. Trigger event specifies the statement that executes the trigger. The trigger event can be any of the DML Statement: INSERT, UPDATE,DELETE.   
 We can not have the two trigger for a given table, which have the same trigger action time and event. For Instance: we cannot have two BEFORE INSERT triggers for same table. But we can have a BEFORE INSERT and BEFORE UPDATE trigger for a same table. Trigger statement have the statement that executes when the trigger fires but if you want to execute multiple statement the you have to use the BEGIN…END compound statement.We can refer the columns of the table that associated with trigger by using the OLD and NEW keyword. OLD.column\_name is used to refer the column of an existing row before it is deleted or updated and NEW.column\_name is used to refer the column of a new row that is inserted.

In INSERT trigger we can use only NEW.column\_name because there is no old row and in a DELETE trigger we can use only OLD.column\_name because there is no new row. But in UPDATE trigger we can use both, OLD.column\_name is used to refer the columns of a row before it is updated and NEW.Column\_name is used to refer the column of the row after it is updated.

c) Demonstrate creating trigger with an example.

**Insert**

**MySQL>** Create trigger t1

    -> before insert on passenger

    -> for each row

    -> begin

    -> if new.age>30 then

    -> set new.pnrno=3333;

    -> else

    -> set new.pnrno=' ';

    -> end if;

    -> end

    -> //

Query OK, 0 rows affected (0.20 sec)

**MySQL>** Insert into Passenger Values(555,'lata',30,'F','Nacharam',’9999999234’,5439)

    -> //

Query OK, 1 row affected (0.16 sec)

**MySQL>** Select \* from Passenger;

**-> //**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PNR no** | **Pname** | **Age** | **Gender** | **Addr** | **Phno** | **Tno** |
| 456 | Avinash | 45 | M | Hyd | 9008745625, 9000567834 | 6754 |
| 746 | Vani | 34 | F | Delhi | 8146372897, 9856437893 | 2087 |
| 129 | Meena | 24 | F | Kurnool | 9989764534 | 3475 |
| 231 | Shashi | 38 | M | B’lore | 9248654891, 8145637689 | 2298 |
| 901 | Navya | 22 | F | Chennai | 9867549072 | 6057 |
| 555 | Lata | 30 | F | Nacharam | 9999999234 | 5439 |

6 rows in set (0.00 sec)

**Update:**

**MySQL>**  Create trigger t1 before update on reservation

    ->  for each row

    ->  begin

    ->  if new.noofseats>30 then

    ->  set new.noofseats=old.noofseats;

    ->  else

    ->  set new.noofseats=new.noofseats;

    ->  end if;

    ->  end//

Query OK, 0 rows affected (0.03 sec)

**MySQL>** Update Reservation set No-of-seats=20 where PNRno=231;

    -> //

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

**MySQL>** Select \* from Reservation;

    -> //

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PNR no** | **Journey Date** | **No-of-seats** | **Addr** | **Phno** | **Status** |
| 456 | 10/08/10 | 40 | Hyd | 9008745625  9000567834 | Yes |
| 129 | 20/08/10 | 40 | Kurnool | 9989764534 | No |
| 901 | 25/08/10 | 40 | Chennai | 9867549072 | Yes |
| 746 | 14/08/10 | 40 | Delhi | 8146372897  9856437893 | No |
| 231 | 18/08/10 | 20 | B’lore | 9248654891  8145637689 | Yes |

6 rows in set (0.00 sec)

**Delete:**

**MySQL>** delimiter //

**MySQL>** Create trigger rc before delete on cancellation

    -> for each row

    -> begin

    -> Insert into Reservation Values (old.PNRno, old.No-of-seats, old.Addr, old.phno,old.status);

    -> end//

Query OK, 0 rows affected (0.05 sec)

**MySQL>** Delete from cancellation where PNRno=129//

Query OK, 1 row affected (0.08 sec)

**MySQL>** Select \* from Reservation//

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PNR no** | **Journey Date** | **No-of-seats** | **Addr** | **Phno** | **Status** |
| 456 | 10/08/10 | 40 | Hyd | 9008745625  9000567834 | Yes |
| 129 | 20/08/10 | 40 | Kurnool | 9989764534 | No |
| 901 | 25/08/10 | 40 | Chennai | 9867549072 | Yes |
| 746 | 14/08/10 | 40 | Delhi | 8146372897  9856437893 | No |
| 231 | 18/08/10 | 40 | B’lore | 9248654891  8145637689 | Yes |

**MySQL>** Select \* from cancellation//

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PNR no** | **Journey Date** | **No-of-seats** | **Addr** | **Phno** | **Status** |
| 456 | 10/08/10 | 40 | Hyd | 9008745625  9000567834 | No |
| 901 | 25/08/10 | 40 | Chennai | 9867549072 | No |
| 746 | 14/08/10 | 40 | Delhi | 8146372897  9856437893 | Yes |
| 231 | 18/08/10 | 40 | B’lore | 9248654891  8145637689 | No |

5. a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

Boats(bid,bname,color);

Reserves(sid,bid,day);

b) Write a procedure to retrieve name and age of sailors whose age is more than 30.

**Select s.name,s.age from sailors s where s.age>30;**

c) Write a procedure to display names of the boat with red color.

**Select B.bname from boats as B where b.colur =’red’;**

d) Write a procedure to display names of the sailors who reserved green color boat.

**Select R.sid from boats b,rserves R where B.bid=R.bid And b.colot=’green’;**

6. What is the need of normalization?

Normalization is used **to minimize the redundancy from a relation or set of relations**. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies. Normalization divides the larger table into smaller and links them using relationships.

Explain 1st, 2nd and 3rd normal forms with suitable examples?

**The normalization forms are:**

1. **First Normal Form**: Eliminate repeating groups: Make a separate each set of related attributes,and give each table a primary key.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tickt\_no | Age | Gen | Journeydate | Src | Dest | Dep\_  time | Booking  date | Bus  \_type | Boarding\_  at | Seat\_no | Total  seats | Amt | Bus\_no |
| 100 | 23 | M | 12-08-2010 | HYD | MUM | 10:00 | 01-08-2010 | A/C | L.B Nagar | 20 | 1 | 800 | 1001 |
| 101 | 24 | F | 14-08-2010 | HYD | CHN | 12:00 | 30-0-2010 | Non-A/C | IBP | 22,23 | 2 | 500 | 1002 |
| 102 | 29 | M | 13-08-2010 | HYD | VJA | 19:00 | 01-08-2010 | A/C | JBS | 24,25 | 2 | 300 | 1003 |
| 103 | 30 | F | 22-08-2010 | HYD | GNT | 13:00 | 10-08-2010 | Non-A/c | DSNR | 28 | 1 | 250 | 1004 |



|  |  |  |
| --- | --- | --- |
| Tkt\_id | Ticket\_no | Seat\_no |
| 1 | 100 | 20 |
| 2 | 101 | 22,23 |
| 3 | 102 | 24,25 |

|  |  |  |
| --- | --- | --- |
| Total\_seats | Amount | Bus\_no |
| 1 | 800 | 1001 |
| 2 | 1000 | 1002 |
| 2 | 600 | 1003 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bus\_no | Ticket  \_no | Journey  \_date | Age | Gen | Src | Dest | Dep\_  time | Booking  date | Boarding  at | Bus\_  type | Total\_  seats | amt |
| 1001 | 100 | 12-08-2010 | 23 | M | HYD | MUM | 10:00 | 01-08-2010 | lbnagar | A/C | 1 | 800 |
| 1002 | 101 | 14-08-2010 | 24 | F | HYD | CHN | 12:00 | 30-08-2010 | IBP | Non-A/C | 2 | 1000 |
| 1003 | 102 | 13-08-2010 | 9 | M | HYD | VJA | 19:00 | 01-08-2010 | JBS | A/C | 2 | 600 |

1. **Second Normal Form**: Eliminate Redundant data-If an attribute depends on only part of a multivalued key remove it to a separate table.

In this example branch entity is having a column which has redundant data that is location.

|  |  |  |  |
| --- | --- | --- | --- |
| Branch\_id | B\_name | manager | Location |
| A1b1 | Sv travels | Rao | Hyd |
| B2c4 | Sv travels | Ram | Goa |
| C3d4 | Sv travels | Ramesh | Chennai |
| D4e5 | Sv travels | Rakesh | Mumbai |



|  |  |
| --- | --- |
| Loc\_id | Loc\_name |
| 10 | HYD |
| 20 | Goa |
| 20 | Chennai |
| 40 | Mumbai |

|  |  |  |
| --- | --- | --- |
| Branch\_id | B\_name | manager |
| A1b1 | Svtravels | Rao |
| B2c4 | Sv travels | Ram |
| C3d4 | Sv travels | Ramesh |
| D4e5 | Sv travels | Rakesh |

|  |  |
| --- | --- |
| Branch\_id | Loc\_id |
| A1b1 | 10 |
| B2c4 | 20 |
| C3d4 | 30 |
| D4e5 | 40 |

1. **Third Normal Form**: Eliminates columns not dependent on key: If attributes do not contribute to a description of the key, remove them out into distinct tables.

In branch entity, branch\_id and b\_name describe only a branch, not a manager. To achieve 3NF, manager must be moved into a separate table.

|  |  |  |
| --- | --- | --- |
| Branch\_id | B\_name | manager |
| A1b1 | Svtravels | Rao |
| B2c4 | Sv travels | Ram |
| C3d4 | Sv travels | Ramesh |
| D4e5 | Sv travels | Rakesh |



|  |  |  |  |
| --- | --- | --- | --- |
| Branch\_id | B\_name | Mgr\_Id |  |
| A1b1 | Svtravels | 01 |  |
| B2c4 | Sv travels | 02 |  |
| C3d4 | Sv travels | 03 |  |
| D4e5 | Sv travels | 04 |  |

|  |  |  |
| --- | --- | --- |
| Mgr\_Id | Mgr\_name | Branch\_Id |
| 01 | Svtravels | A1b1 |
| 02 | Sv travels | B2c4 |
| 03 | Sv travels | C3d4 |
| 04 | Sv travels | D4e5 |

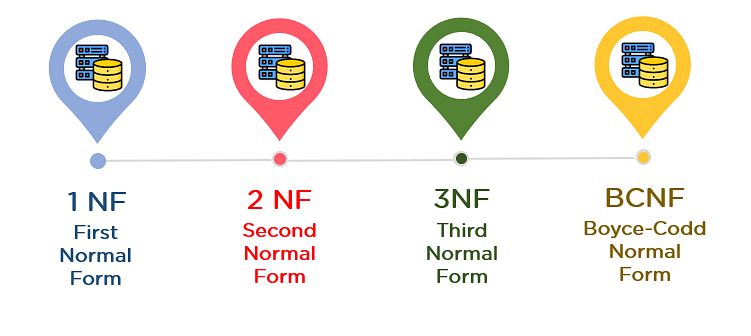
**7. What is normalization?**

**Normalization is the process to eliminate**[**data**](https://www.simplilearn.com/what-is-data-article)**redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.**

**Normalization organizes the columns and tables of a database to ensure that database integrity constraints properly execute their dependencies. It is a systematic technique of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update, and Deletion anomalies.**

**In 1970 Edgar F. Codd defined the First Normal Form.**

**Now let's understand the types of Normal forms with the help of examples.**



Explain BCNF,

**BCNF (Boyce Codd Normal Form) is the advanced version of 3NF. A table is in BCNF if every functional dependency X->Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD. LHS is super key.**

**Example:** Let's assume there is a company where employees work in more than one department.

**EMPLOYEE table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_COUNTRY** | **EMP\_DEPT** | **DEPT\_TYPE** | **EMP\_DEPT\_NO** |
| 264 | India | Designing | D394 | 283 |
| 264 | India | Testing | D394 | 300 |
| 364 | UK | Stores | D283 | 232 |
| 364 | UK | Developing | D283 | 549 |

**In the above table Functional dependencies are as follows:**

1. EMP\_ID  →  EMP\_COUNTRY
2. EMP\_DEPT  →   {DEPT\_TYPE, EMP\_DEPT\_NO}

Explain 4th and 5th normal forms with suitable examples?

**Fourth normal form (4NF):**

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

**Properties –** A relation R is in 4NF if and only if the following conditions are satisfied: 

1. It should be in the Boyce-Codd Normal Form (BCNF).
2. the table should not have any Multi-valued Dependency.

A table with a multivalued dependency violates the normalization standard of Fourth Normal Form (4NK) because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break this information into two tables.

**Example –** Consider the database table of a class which has two relations R1 contains student ID(SID) and student name (SNAME) and R2 contains course id(CID) and course name (CNAME).

**Table –** R1(SID, SNAME)

| SID | SNAME |
| --- | --- |
| S1 | A |
| S2 | B |

**Table –** R2(CID, CNAME) 

| CID | CNAME |
| --- | --- |
| C1 | C |
| C2 | D |

When there cross product is done it resulted in multivalued dependencies:

**Table –** R1 X R2

| SID | SNAME | CID | CNAME |
| --- | --- | --- | --- |
| S1 | A | C1 | C |
| S1 | A | C2 | D |
| S2 | B | C1 | C |
| S2 | B | C2 | D |

**Fifth Normal Form / Projected Normal Form (5NF):**

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated, when relations are reunited through a natural join.

**Properties –** A relation R is in 5NF if and only if it satisfies following conditions: 

1. R should be already in 4NF.
2. It cannot be further non loss decomposed (join dependency)

**Example –** Consider the above schema, with a case as “if a company makes a product and an agent is an agent for that company, then he always sells that product for the company”. Under these circumstances, the ACP table is shown as:

**Table –** ACP

| Agent | Company | Product |
| --- | --- | --- |
| A1 | PQR | Nut |
| A1 | PQR | Bolt |
| A1 | XYZ | Nut |
| A1 | XYZ | Bolt |
| A2 | PQR | Nut |

The relation ACP is again decompose into 3 relations. Now, the natural Join of all the three relations will be shown as:

**Table –** R1

| Agent | Company |
| --- | --- |
| A1 | PQR |
| A1 | XYZ |
| A2 | PQR |

**Table –** R2 

| Agent | Product |
| --- | --- |
| A1 | Nut |
| A1 | Bolt |
| A2 | Nut |

**Table –** R3 

| Company | Product |
| --- | --- |
| PQR | Nut |
| PQR | Bolt |
| XYZ | Nut |
| XYZ | Bolt |
| Result of Natural Join of R1 and R3 over ‘Company’ and then Natural Join of R13 and R2 over ‘Agent’and ‘Product’ will be table **ACP**.  Hence, in this example, all the redundancies are eliminated, and the decomposition of ACP is a lossless join decomposition. Therefore, the relation is in 5NF as it does not violate the property of [lossless join](https://www.geeksforgeeks.org/database-management-system-lossless-decomposition/). |  |

8) a) create sailors, boats, reserves tables using the following fields.

Note: sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

Boats(bid,bname,color);

Reserves(sid,bid,day);

b) What are cursors?

**Cursors are used when the SQL Select statement is expected to return more than one row. Cursors are supported inside procedures and functions. Cursors must be declared and its definition contains the query. The cursor must be defined in the DECLARE section of the program. A cursor must be opened before processing and close after processing.**

1. **Implicit Cursors:**  
   Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
2. **Explicit Cursors :**  
   Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

Write syntax for:

1. Declaring a cursor

**DECLARE <cursor\_name> CURSOR FOR <select\_statement>**

1. Opening a cursor

**OPEN <cursor\_name>**

1. Fetching a cursor

**FETCH <cursor\_name> INTO <var1>,<var2>…….**

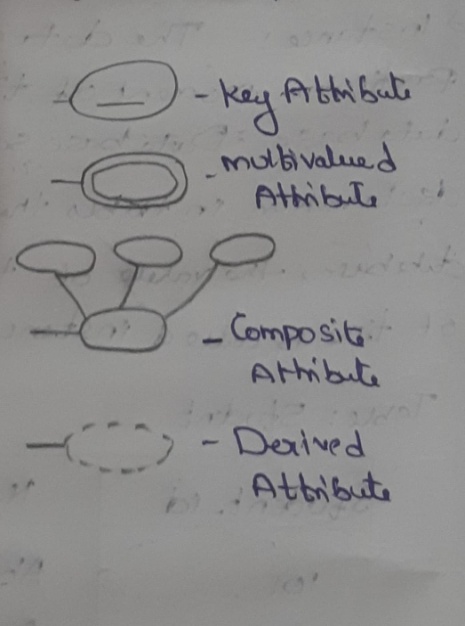
iv ) closing a cursor

**CLOSE <cursor\_name>**

1. a) Explain about different notations used in E-R Model.

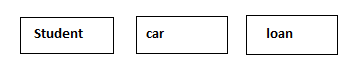
**Entity Relationship Diagram (ERD) Symbols and Notations**





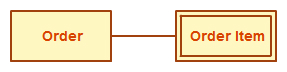
#### Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in [ER diagrams](https://creately.com/lp/er-diagram-tool-online/) by a rectangle and named using singular nouns.



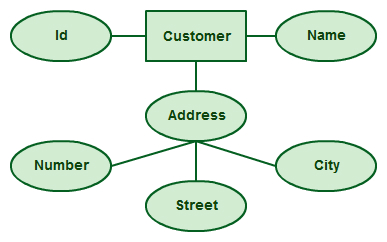
#### Weak Entity

A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a [foreign key](https://creately.com/diagram/example/jhix8fel1/Foreign%20Key%20in%20ER%20Diagram) combined with its attributed to form the primary key.



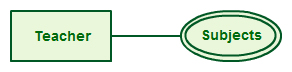
**Attribute**

An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes.



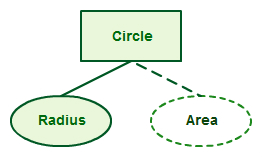
#### Multivalued Attribute

If an attribute can have more than one value it is called a [multi-valued attribute.](https://creately.com/diagram-type/objects/database-design/multivalued-attribute) It is important to note that this is different from an attribute having its own attributes. For example, a teacher entity can have multiple subject values.

[](https://creately.com/demo-start?tempId=NJHJqfIcQP8)

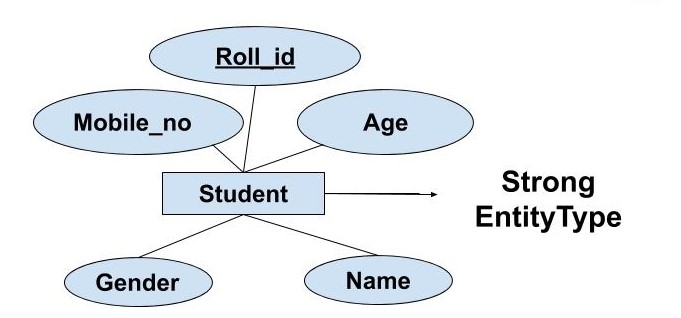
**Derived Attribute**

An attribute based on another attribute. This is found rarely in ER diagrams. For example, for a circle, the area can be derived from the radius.

[](https://creately.com/demo-start?tempId=pEj9ElcQZ5W)

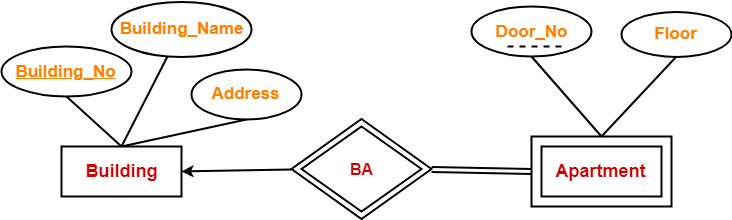
**Key attribute**

Key attributes are special types of attributes that act as the primary key for an entity and they can uniquely identify an entity from an entity set. The values that key attributes store must be unique and non-repeating.



## Weak Entity

The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.



#### ****Relationship****

A relationship describes how entities interact. For example, the entity “Carpenter” may be related to the entity “table” by the relationship “builds” or “makes”. Relationships are represented by diamond shapes and are labeled using verbs.

[](https://creately.com/demo-start?tempId=I2z3FuXrAgO)

Using Relationships in Entity Relationship Diagrams

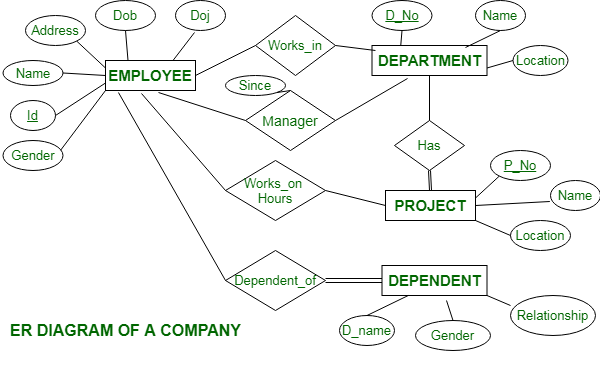
**weak relation*ship***

#### A weak relation arises between two types of entities in the case when there is no direct relationship between them. In this case, the relation is optional, that is, there may be situations where the relation between entities is not needed. Entity types are able to exist autonomously without each other.

#### Weak Entity Set in ER diagrams - GeeksforGeeks

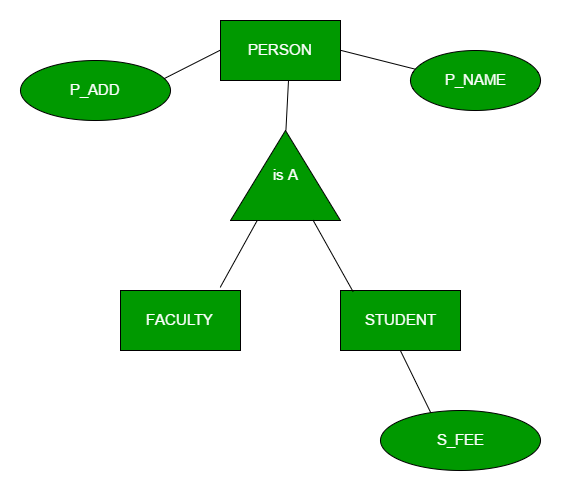
b) Design ER diagram for EMPLOYEE – DEPARTMENT database

.

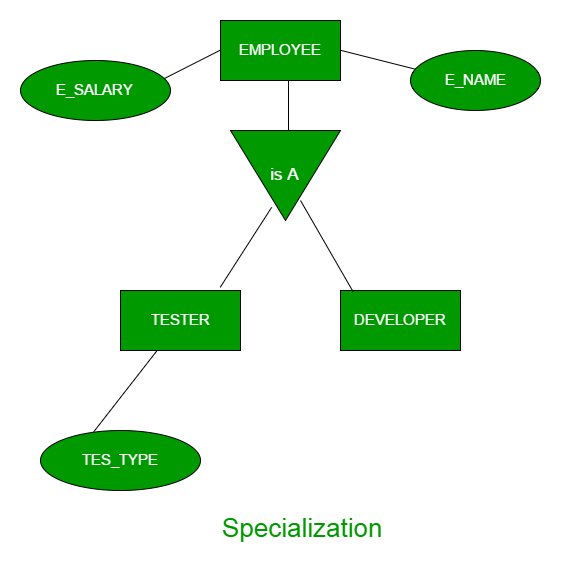


1. a) Explain about additional features of E-R model.

**Generalization –**   
Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P\_NAME, P\_ADD become part of higher entity (PERSON) and specialized attributes like S\_FEE become part of specialized entity (STUDENT).

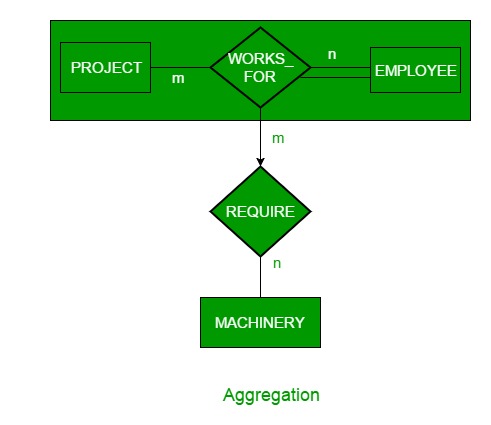


**Specialization –**   
In specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E\_NAME, E\_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES\_TYPE become part of specialized entity (TESTER).

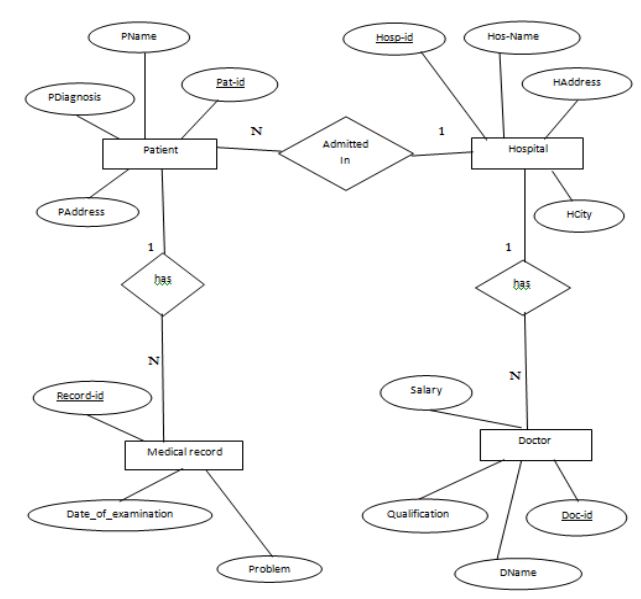


**Aggregation –**   
An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

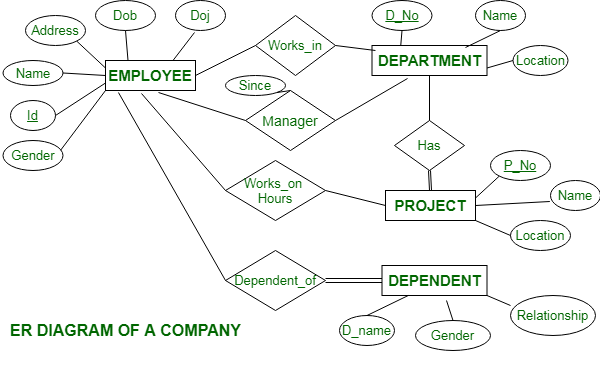
For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS\_FOR and entity MACHINERY. Using aggregation, WORKS\_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



b) Design ER diagram for HOSPITAL database.

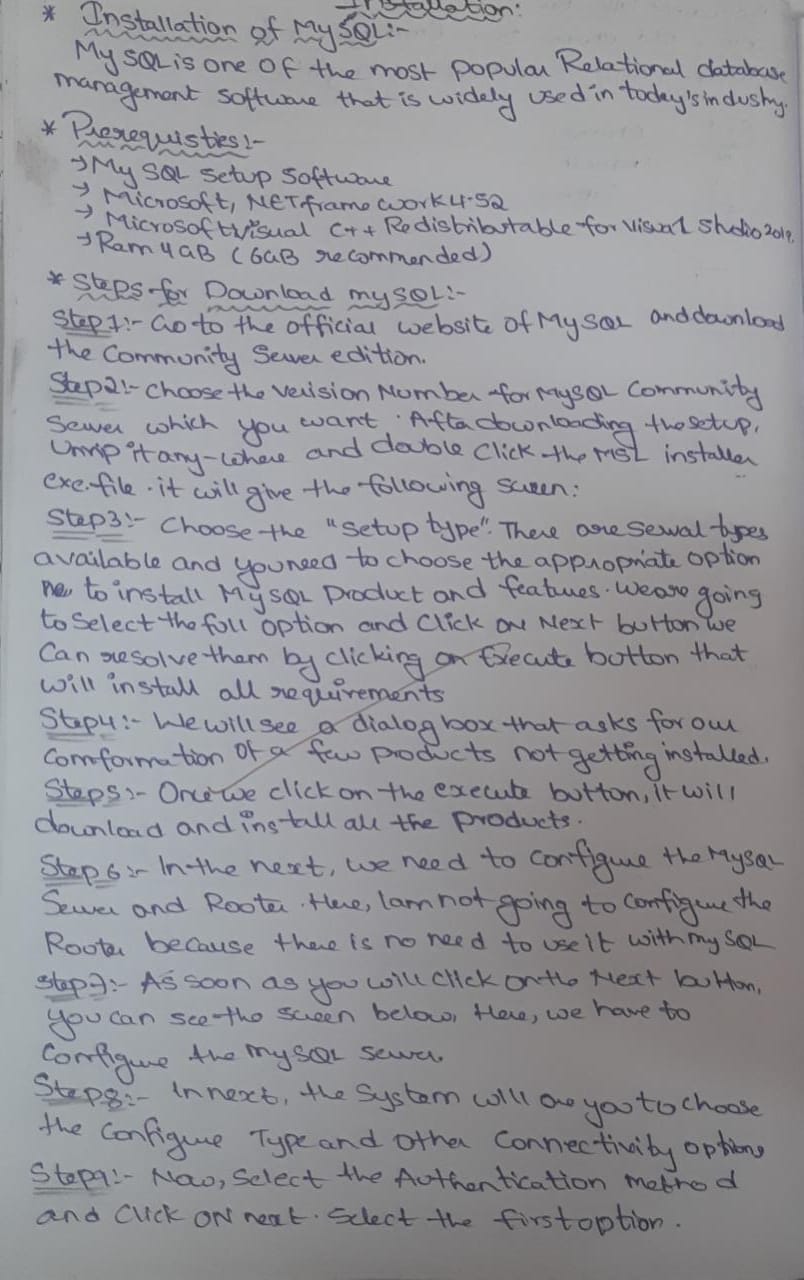


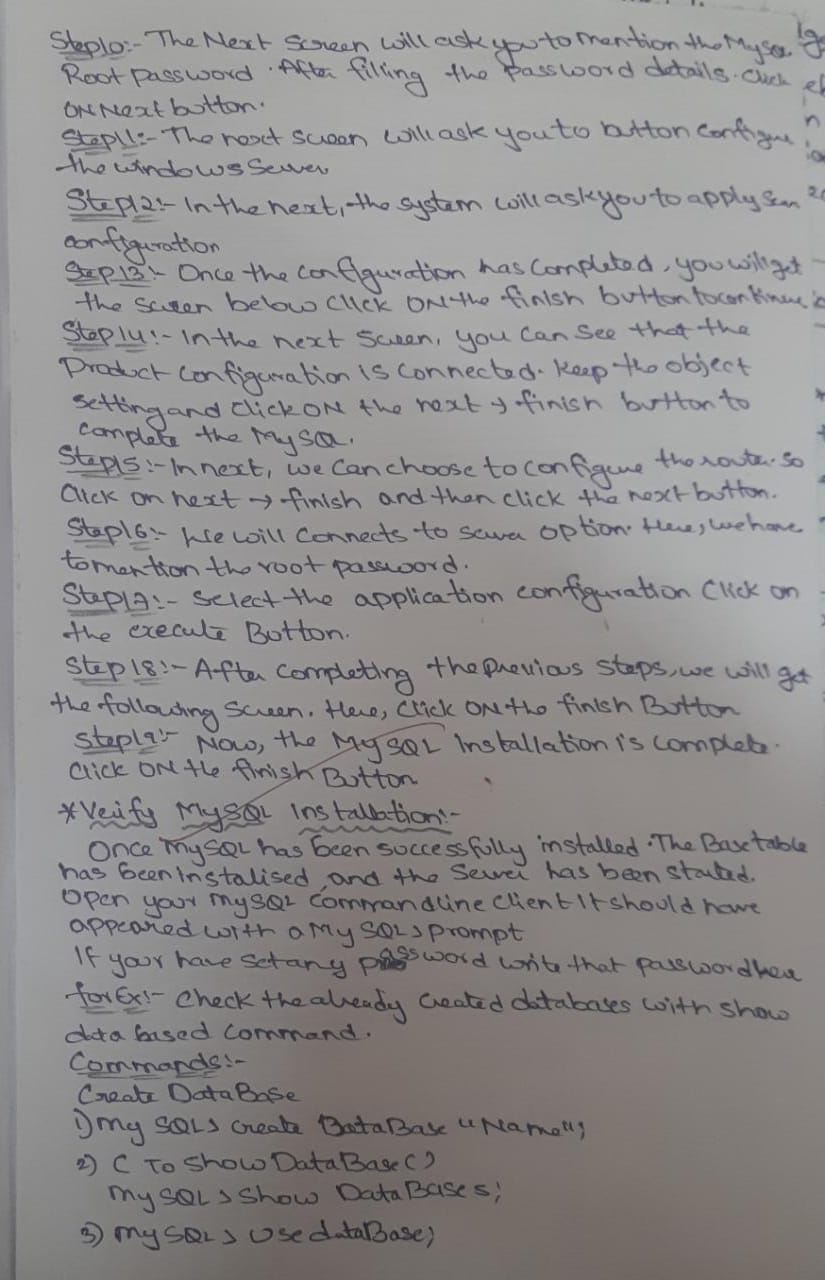
1. a) Explain logical database design for Employee-Department database.

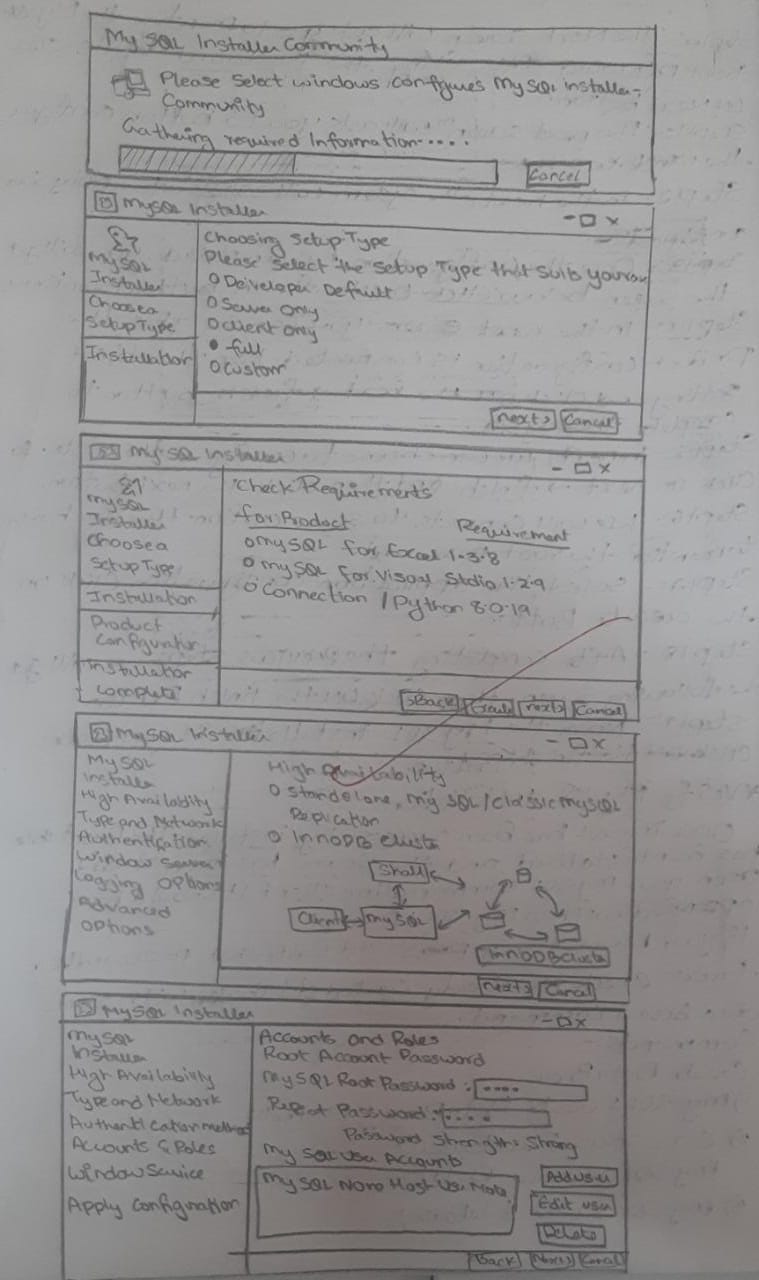


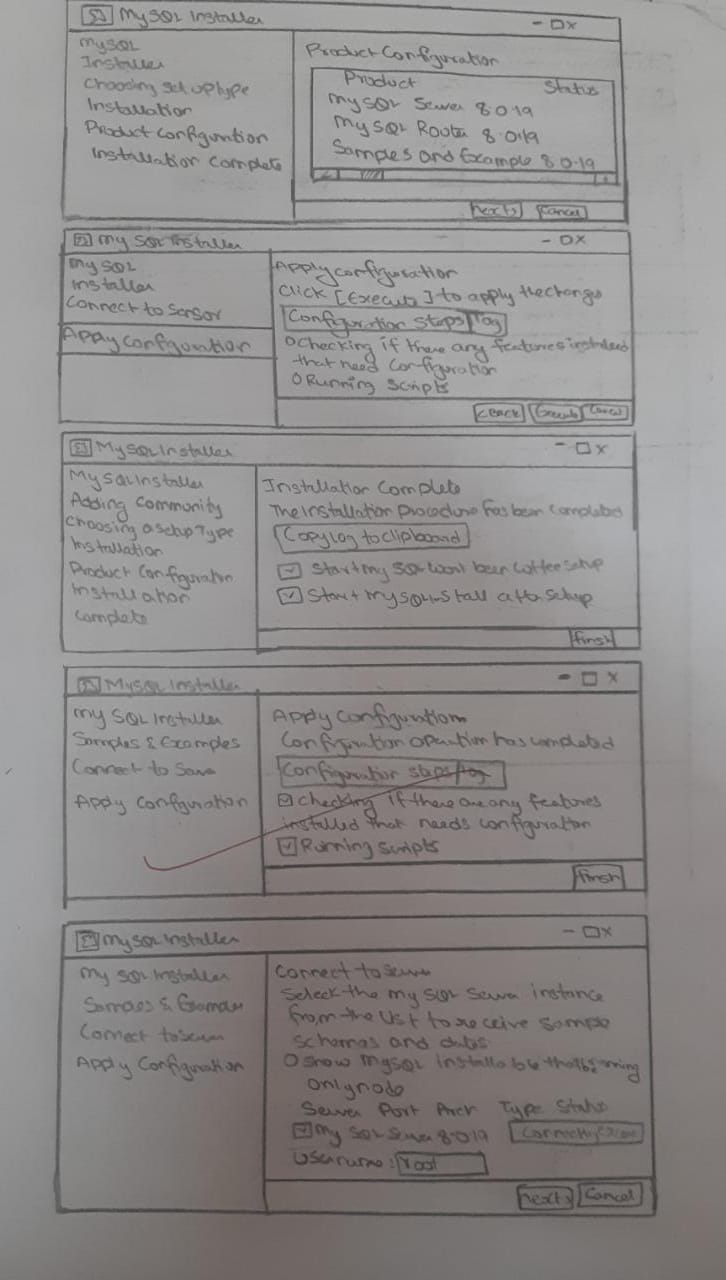
b) Write step-by-step procedure for MySQL Software installation and Demonstrate it

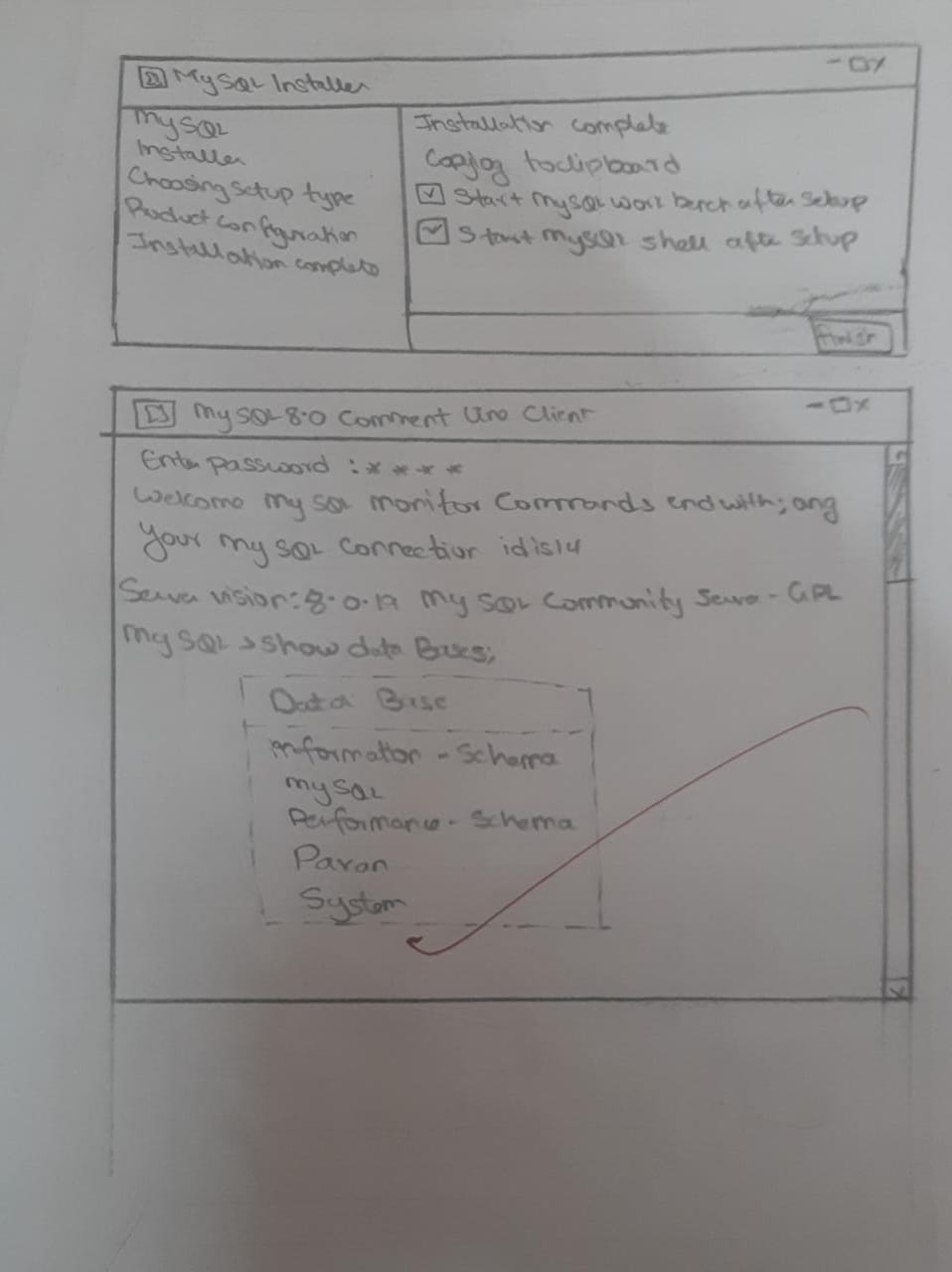
on a Desktop.











1. a) Create the employee table using following schema.

Employee(eid int,name varchar(10),salary float);

**create table Employee(eid int ,name varchar(10),salary float);**

b) write an SQL query to add a new column gender.

**alter table Employee add column gender varchar(10) after name;**

c)write an SQL query to change the name of the salary column as ‘esal’

**alter table Employee rename column salary to esal;**

d)write an SQL query to modify data type of the name column to varchar(30)

**alter table Employee modify name varchar(30);**

e)Write an SQL query to change the name of the table from employee to employee\_details.

**alter table Employee rename to employee\_details;**

f) Write an SQL query to remove all the rows from the employee table using DDL command.

**truncate employee\_details;**

g)Write an SQL query to remove the employee table from the database.

**delete from employee\_details;**

1. a) Create the Book table using following schema.

Book(bid int, title varchar(10),pages int);

**create table Book(bid int,title varchar(10),pages int);**

**insert into Book values(101,'us',10),(102,'is',29),(103,'hs',28);**

b) write an SQL query to add a new column price.

**alter table Book add column price float after title;**

c)write an SQL query to change the name of the title column as ‘btitle’

**alter table Book rename column title to btitle;**

d)write an SQL query to modify data type of the title column to varchar(30)

**alter table Book modify btitle varchar(30);**

e)Write an SQL query to change the name of the table from Book to Book\_details.

**alter table Book rename to Book\_details;**

f) Write an SQL query to remove all the rows from the Book table using DDL command.

**truncate Book\_details;**

g)Write an SQL query to remove the Book table from the database

**delete from Book\_details;**

14a) Create the employee table using following schema. Make sure that eid column doesnot allow null values.

Employee(eid int,name varchar(20),salary float);

**Create table employee(eid int NOT NULL,name varchar(20),salary float);**

b) Create the department table using following schema. Make sure that budget should be more than 50000 for all records.

department(did int,dname varchar(20),budget float);

create table

**create table department(did int,dname varchar(20),budget float check(budget>=50000);**

c) Create the student table using following schema. Make sure that sid column doesnot allow duplicate values.

student(sid int,name varchar(20),age int);

**Create table student(sid int PRIMARY KEY,name varchar(20),age int);**

d) create sailors, boats, reserves tables using the following fields.

Note :sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

**Create table sailors(sid int PRIMARY KEY,sname varchar(20),rating int,age int);**

Boats(bid,bname,color);

**Create table boats(bid int PRIMARY KEY,bname varchar(30),color varchar(10));**

Reserves(sid,bid,day);

**Create table reserves(sid int,bid int,day date,FOREIGN KEY(sid) REFERENCE sailors(sid), FOREIGN KEY(bid) REFERENCES boats(bid));**

15a) Create the student table using following schema. Make sure that sid column doesnot allow null values.

student(sid int,name varchar(20),age int);

**Create table student(sid int NOT NULL,name varchar(20),age int);**

b) Create the voters table using following schema. Make sure that only records with more than 18 years age stored in the table.

Voters(voter\_id int, voter\_name varchar(20),voter\_age int);

**Create table voters(voter\_id,voter\_name varchar(20),voter\_age int check(age>=18));**

c) Create the employee table using following schema. Make sure that if the user does not provide a value for salary, 20000 need to be stored in the table.

Employee(eid int,name varchar(20),salary float);

**Create table employee(eid int,name varchar(20),salary float default 20000);**

d) create sailors, boats, reserves tables using the following fields.

Note :sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

**Create table sailors(sid int PRIMARY KEY,sname varchar(20),rating int,age int);**

Boats(bid,bname,color);

**Create table boats(bid int PRIMARY KEY,bname varchar(30),color varchar(10));**

Reserves(sid,bid,day);

**Create table reserves(sid int,bid int,day date,FOREIGN KEY(sid) REFERENCE sailors(sid), FOREIGN KEY(bid) REFERENCES boats(bid));**

16)a) Create the department table using following schema. Make sure that did column doesnot allow null values.

department(did int,dname varchar(20),budget float);

**create table department(did int NOT NULL,dname varchar(20),budget float);**

b) Create the SSC\_Student table using following schema. Make sure that age of every SSC student more than or equals to 15 years.

SSC\_Student(sid int,name varchar(20),age int);

**Create table student(sid int,name varchar(20),age int check(age>=18));**

c) Create the employee table using following schema. Make sure that if the user does not provide a value for salary, 10000 need to be stored in the table.

Employee(eid int,name varchar(20),salary float);

**Create table employee(eid int,name varchar(20),float default 10000);**

d) create sailors, boats, reserves tables using the following fields.

Note :sid is the primary key for sailors table

bid is the primary key for boats table

sid,bid is foreign keys in reserves table

Sailors(sid,sname,rating,age);

**Create table sailors(sid int PRIMARY KEY,sname varchar(20),rating int,age int);**

Boats(bid,bname,color);

**Create table boats(bid int PRIMARY KEY,bname varchar(30),color varchar(10));**

Reserves(sid,bid,day);

**Create table reserves(sid int,bid int,day date,FOREIGN KEY(sid) REFERENCE sailors(sid), FOREIGN KEY(bid) REFERENCES boats(bid));**

17)a) Create the employee table using following schema and data.

Employee(eid int,name varchar(20),salary float);

|  |  |  |
| --- | --- | --- |
| eid | name | salary |
| 101 | Mr.X | 120000 |
| 102 | Mr.Y | 150000 |
| 103 | Mr.Z | 90000 |
| 104 | Mr.P | 80000 |

**create table Employee(eid int,name varchar(20),salary float);**

**insert into Employee values(101,'x',120000),(102,'y',150000),(103,'z',90000),(104,'p',80000);**

**select \* from Employee;**

b) write an SQL query to display the names and salary of all employees in the organization.

**select name,salary from Employee;**

c) write an SQL query to change the salary of Mr.Z from 90000 to 95000.

**update Employee set salary=95000 where name='z';**

**select \* from Employee;**

d) write an SQL query to display employee names whose salary is more than 100000

**select \* from Employee where salary>100000;**

e) write an SQL query to remove the record of Mr.P from the table.

**delete from Employee where name='p';select \* from Employee;**

f) write an SQL query to remove all the records from the table using DML command.

**delete from Employee;**

18)a) Create the student table using following schema. Make sure that sid column doesnot allow null values.

student(sid int,name varchar(20),age int);

|  |  |  |
| --- | --- | --- |
| sid | name | age |
| 101 | Mr.X | 15 |
| 102 | Mr.Y | 14 |
| 103 | Mr.Z | 19 |
| 104 | Mr.P | 20 |

**create table student(sid int not null,name varchar(20),age int);**

**insert into student values(101,'x',15),(102,'y',14),(103,'z',19),(104,'p',20);**

**select \* from student;**

b) write an SQL query to display the names and age of all students in the college.

**select name,age from student;**

c) write an SQL query to change the age of Mr.Z from 19 to 20.

**update student set age=20 where age=19;**

**select \* from student;**

d) write an SQL query to display student names whose age is more than 18

**select \* from student where age>18;**

e) write an SQL query to remove the record of Mr.P from the table.

**delete from student where name='p';**

**select \* from student;**

f) write an SQL query to remove all the records from the table using DML command.

**delete from student;**