**PROJECT REPORT PHASE – 3**

**Phase III. Now, you are ready for implementation. Use appropriate naming conventions for all your tables and attributes.**

**a. Normalize all your tables to third normal form**.

Normalized tables with the Functional dependencies are shown below :

A picture containing table

Description automatically generated

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated with medium confidence

Diagram

Description automatically generated

**Normal forms :**

* In the above functional dependencies, all the non-key attributes are fully functional dependent on the primary key.
* All the above relation tables are in **1 NF**. As all the relation tables are in atomic form.
* A relation schema R is in second normal form (2NF) if every nonprime attribute A in R is not partially dependent on any key of R. (or) Every non prime attribute A in R is fully functionally dependent on every key of R.
* Therefore, all the relation tables are in **2NF**. Since all the non prime attributes are fully functionally dependent on every key of the table.
* A relation schema R is in third normal form (3NF) if, whenever a nontrivial functional dependency X → A holds in R, either (a) X is a superkey of R, or (b) A is a prime attribute of R.
* All the non trivial functional dependencies in the tables have the superkey in the left side of the dependency. Therefore, all the tables are in **3NF** as well.
* BCNF can be directly tested by using all of the given dependencies and finding out if the left-hand side of each is a super key. Therefore all the relation tables are in BCNF as well.

Therefore, all the relational tables are normalized and can be implemented into the database.

**b. Draw a dependency diagram for each table from Phase III a.**

Graphical user interface, application, table, Excel

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

**c. Write SQL statements to create database, tables and all other structures. Primary key and foreign keys must be defined as appropriate. Also specify data type and constraints for each attribute and in addition to specify the referential integrity.**

Creating the database tables and the specifying the primary keys, foreign keys and constraints as well in the SQL statements .

CREATE DATABASE wonder\_library;

USE wonder\_library;

CREATE TABLE PERSON (

PID varchar(10) NOT NULL,

FName varchar(250) NOT NULL,

Middle\_Name varchar(250) NOT NULL,

Lname varchar(250) NOT NULL,

DOB DATE NOT NULL,

Gender varchar(20) NOT NULL,

Address varchar(300) NOT NULL,

PRIMARY KEY (PID),

CONSTRAINT Check\_person CHECK (PID not like 'P[0-9][0-9][0-9]')

);

CREATE TABLE MEMBER (

Member\_ID varchar(10) NOT NULL,

Enrollment\_date date NOT NULL,

Membership\_type varchar(300) NOT NULL,

PRIMARY KEY (Member\_ID),

FOREIGN KEY(Member\_ID) REFERENCES PERSON(PID) ON DELETE CASCADE

);

CREATE TABLE GOLD (

M\_ID varchar(10) NOT NULL,

PRIMARY KEY (M\_ID),

FOREIGN KEY(M\_ID) REFERENCES MEMBER(Member\_ID) ON DELETE CASCADE

);

CREATE TABLE SILVER (

M\_ID varchar(10) NOT NULL,

PRIMARY KEY (M\_ID),

FOREIGN KEY(M\_ID) REFERENCES MEMBER(Member\_ID) ON DELETE CASCADE

);

CREATE TABLE EMPLOYEE (

Employee\_ID varchar(10) NOT NULL,

Start\_Date date NOT NULL,

Age integer NOT NULL,

EType varchar(300) NOT NULL,

PRIMARY KEY (Employee\_ID),

FOREIGN KEY(Employee\_ID) REFERENCES PERSON(PID) ON DELETE CASCADE,

CONSTRAINT Check\_age CHECK (Age >= 18)

);

CREATE TABLE TRAINER (

Trainer\_ID varchar(10) NOT NULL,

PRIMARY KEY (Trainer\_ID)

);

CREATE TABLE RECEPTIONIST (

Recep\_ID varchar(10) NOT NULL,

Trainer\_ID varchar(10) NOT NULL,

PRIMARY KEY (Recep\_ID),

FOREIGN KEY(Recep\_ID) REFERENCES EMPLOYEE(Employee\_ID) ON DELETE CASCADE,

FOREIGN KEY(Trainer\_ID) REFERENCES TRAINER(Trainer\_ID) ON DELETE CASCADE

);

CREATE TABLE LIBRARY\_SUPERVISOR (

LibSup\_ID varchar(10) NOT NULL,

Trainer\_ID varchar(10) NOT NULL,

PRIMARY KEY (LibSup\_ID),

FOREIGN KEY(LibSup\_ID) REFERENCES EMPLOYEE(Employee\_ID) ON DELETE CASCADE,

FOREIGN KEY(Trainer\_ID) REFERENCES TRAINER(Trainer\_ID) ON DELETE CASCADE

);

CREATE TABLE CATALOGING\_MANAGER (

CatMang\_ID varchar(10) NOT NULL,

Trainer\_ID varchar(10) NOT NULL,

PRIMARY KEY (CatMang\_ID),

FOREIGN KEY(CatMang\_ID) REFERENCES EMPLOYEE(Employee\_ID) ON DELETE CASCADE,

FOREIGN KEY(Trainer\_id) REFERENCES TRAINER(Trainer\_ID) ON DELETE CASCADE

);

CREATE TABLE CATEGORY (

Category\_Number integer NOT NULL,

PRIMARY KEY(Category\_Number),

CONSTRAINT Check\_Category CHECK (Category\_Number <= 3 AND Category\_Number >= 1)

);

CREATE TABLE BOOK (

Book\_ID varchar(10) NOT NULL,

Title varchar(200) NOT NULL,

Other\_Info varchar(300),

Category\_number integer NOT NULL,

PRIMARY KEY (Book\_ID),

FOREIGN KEY(Category\_number) REFERENCES CATEGORY(Category\_Number) ON DELETE CASCADE

);

CREATE TABLE PUBLISHER (

Publisher\_ID varchar(10) NOT NULL,

Publisher\_Name varchar(200) NOT NULL,

Established\_Year integer,

Address varchar(300),

PRIMARY KEY(Publisher\_ID)

);

CREATE TABLE AUTHOR (

Author\_ID varchar(10) NOT NULL,

Author\_Name varchar(200) NOT NULL,

Style\_of\_Writing varchar(200),

Year\_of\_Operation integer,

PRIMARY KEY(Author\_ID)

);

CREATE TABLE PAYMENT (

Payment\_ID varchar(10) NOT NULL,

Amount integer NOT NULL,

Time timestamp DEFAULT CURRENT\_timestamp,

PMethod varchar(10) NOT NULL,

PRIMARY KEY (Payment\_ID)

);

CREATE TABLE GUEST\_LOG (

Guest\_ID varchar(10) NOT NULL,

M\_ID varchar(10) NOT NULL,

Fname varchar(300) NOT NULL,

Middle\_name varchar(300),

Lname varchar(300) NOT NULL,

Address varchar(300),

Contact varchar(200),

PRIMARY KEY (Guest\_ID, M\_ID),

FOREIGN KEY(M\_ID) REFERENCES GOLD(M\_ID) ON DELETE CASCADE

);

CREATE TABLE LIBRARY\_CARD (

Card\_num integer NOT NULL,

MemberID varchar(10) NOT NULL,

PRIMARY KEY (Card\_num,MemberID),

FOREIGN KEY(MemberID) REFERENCES MEMBER(Member\_ID) ON DELETE CASCADE

);

CREATE TABLE PROMOTION (

Code varchar(10) NOT NULL,

MemberID varchar(10) NOT NULL,

Card\_Num integer NOT NULL,

Description varchar(300),

PRIMARY KEY (Code, MemberID, Card\_Num),

FOREIGN KEY(Card\_num,MemberID) REFERENCES LIBRARY\_CARD(Card\_num,MemberID) ON DELETE CASCADE

);

CREATE TABLE COMMENTS\_ON (

PID varchar(10) NOT NULL,

Book\_ID varchar(10) NOT NULL,

Comment\_Time timestamp,

Rating integer NOT NULL,

PRIMARY KEY (PID, Book\_ID),

FOREIGN KEY(PID) REFERENCES PERSON(PID) ON DELETE CASCADE,

FOREIGN KEY(Book\_ID) REFERENCES BOOK(Book\_ID) ON DELETE CASCADE,

CONSTRAINT Check\_Rating CHECK (Rating <= 5 AND Rating >= 1)

);

CREATE TABLE PUBLISHES (

Book\_ID varchar(10) NOT NULL,

Publisher\_ID varchar(10) NOT NULL,

PRIMARY KEY (Publisher\_ID, Book\_ID),

FOREIGN KEY(Publisher\_ID) REFERENCES PUBLISHER(Publisher\_ID) ON DELETE CASCADE,

FOREIGN KEY(Book\_ID) REFERENCES BOOK(Book\_ID) ON DELETE CASCADE

);

CREATE TABLE WRITES (

Book\_ID varchar(10) NOT NULL,

Author\_ID varchar(10) NOT NULL,

PRIMARY KEY (Author\_ID, Book\_ID),

FOREIGN KEY(Author\_ID) REFERENCES AUTHOR(Author\_ID) ON DELETE CASCADE,

FOREIGN KEY(Book\_ID) REFERENCES BOOK(Book\_ID) ON DELETE CASCADE

);

CREATE TABLE CATELOG\_ACTIVITY (

CatMangID varchar(10) NOT NULL,

Category\_Num integer NOT NULL,

Catalog\_Date date NOT NULL,

PRIMARY KEY (CatMangID, Category\_Num),

FOREIGN KEY(Category\_Num) REFERENCES CATEGORY(Category\_Number) ON DELETE CASCADE,

FOREIGN KEY(CatMangID) REFERENCES CATALOGING\_MANAGER(CatMang\_ID) ON DELETE CASCADE

);

CREATE TABLE PHONE\_NUMBERS (

PID varchar(10) NOT NULL,

Phone\_num integer NOT NULL,

PRIMARY KEY (PID, Phone\_num),

FOREIGN KEY(PID) REFERENCES PERSON(PID) ON DELETE CASCADE,

CONSTRAINT Check\_phone CHECK (Phone\_num <= 9999999999 AND Phone\_num >= 000000000)

);

CREATE TABLE BORROWING (

PID varchar(10) NOT NULL,

Recep\_ID varchar(10) NOT NULL,

BookID varchar(10) NOT NULL,

Payment\_ID varchar(10) NOT NULL,

Date\_of\_issue date NOT NULL,

Due\_date date NOT NULL,

PRIMARY KEY (PID, Recep\_ID, BookID, Payment\_ID),

FOREIGN KEY(PID) REFERENCES PERSON(PID) ON DELETE CASCADE,

FOREIGN KEY(BookID) REFERENCES BOOK(Book\_ID) ON DELETE CASCADE,

FOREIGN KEY(Payment\_ID) REFERENCES PAYMENT(Payment\_ID) ON DELETE CASCADE,

FOREIGN KEY(Recep\_ID) REFERENCES RECEPTIONIST(Recep\_ID) ON DELETE CASCADE

);

**d. Use the Create View statement to create the following views:**

**1. TopGoldMember - This view returns the First Name, Last Name and Date of membership enrollment of those members who have borrowed more than 5 books in past month**

CREATE VIEW TOP\_GOLD\_MEMBER

AS

SELECT m.Member\_ID, p.FName, p.LName, m.Enrollment\_date,COUNT(\*) as Number\_of\_books

FROM PERSON AS p

INNER JOIN MEMBER m

ON p.PID = m.Member\_ID

INNER JOIN GOLD g

ON g.M\_ID = m.Member\_ID

INNER JOIN BORROWING Br

ON Br.PID = p.PID

WHERE br.Date\_of\_issue >= DATE\_SUB(curdate(), INTERVAL 1 MONTH)

GROUP BY m.Member\_ID,p.FName,p.LName,m.Enrollment\_date

HAVING Number\_of\_books > 5

ORDER BY Number\_of\_books DESC;

SELECT \* FROM TOP\_GOLD\_MEMBER;

Graphical user interface, application

Description automatically generated

**2. PopularBooks - This view returns the details of the most borrowed books over the past year**

CREATE VIEW POPULAR\_BOOKS

AS

SELECT br.BookID, bk.Title , COUNT(\*) as No\_of\_books

FROM BOOK bk

INNER JOIN BORROWING br

ON bk.Book\_ID = br.BookID

WHERE br.Date\_of\_issue >= DATE\_SUB(curdate(), INTERVAL 1 YEAR)

GROUP BY br.BookID

ORDER BY No\_of\_books DESC;

SELECT \* FROM POPULAR\_BOOKS;

Graphical user interface, application, table

Description automatically generated

**3. BestRatingPublisher – This view returns the names of publisher whose books are all have at least 4.0 average rating score**

CREATE VIEW BEST\_RATING\_PUBLISHER

AS

SELECT Pr.Publisher\_name, Cr.Average\_rating\_score

FROM (SELECT ps.Book\_ID, ps.Publisher\_ID , AVG(co.Rating) as Average\_rating\_score

FROM PUBLISHES ps

INNER JOIN COMMENTS\_ON co

ON ps.Book\_ID = co.Book\_ID

GROUP BY ps.Book\_ID) as Cr,

PUBLISHER Pr

WHERE Pr.Publisher\_ID = Cr.Publisher\_ID AND Cr.Average\_rating\_score >= 4.0;

SELECT \* FROM BEST\_RATING\_PUBLISHER;

Graphical user interface, text, application

Description automatically generated

**4. PotentialGoldMember - This view returns the name, phone number and ID of the silver members who borrowed books in every month in the past year**

CREATE VIEW POTENTIAL\_GOLD\_MEMBER

AS

SELECT p.PID, p.Fname, p.Middle\_name, p.LName, ph.Phone\_num

FROM (SELECT br.PID, COUNT(br.PID) as Counts

FROM BORROWING br WHERE br.Date\_of\_issue > DATE(curdate() - INTERVAL 1 YEAR)

GROUP BY br.PID) AS ct, SILVER sm, MEMBER m, PERSON p, PHONE\_NUMBERS ph

WHERE ct.Counts = 12 AND

m.Member\_ID = ct.PID AND

p.PID = ct.PID AND

sm.M\_ID = ct.PID;

SELECT \* FROM POTENTIAL\_GOLD\_MEMBER;

Graphical user interface, text, application, email

Description automatically generated

**5. PopularAuthor – This view returns details of authors whose books have been borrowed the most**

CREATE VIEW POPULAR\_AUTHOR

AS

SELECT Au.Author\_ID, Au.Author\_Name ,COUNT(\*) as No\_of\_books

FROM AUTHOR Au, WRITES w ,BOOK bk, BORROWING br

WHERE Au.Author\_ID = w.Author\_ID AND w.Book\_ID = bk.Book\_ID AND bk.Book\_ID = br.BookID

GROUP BY br.BookID

ORDER BY No\_of\_books DESC;

SELECT \* FROM POPULAR\_AUTHOR;

Graphical user interface, application

Description automatically generated

**e. Show the SQL statement of the following Queries. Feel free to use any of the views that you created in part (d.):**

**1.List the details of all the supervisors of the library hired in past two months.**

SELECT ls.LibSup\_ID, p.FName, p.Middle\_Name, p.Lname, emp.Start\_date

FROM PERSON p

JOIN EMPLOYEE emp

ON p.PID = emp.Employee\_ID

JOIN LIBRARY\_SUPERVISOR ls

ON emp.Employee\_ID = ls.LibSup\_ID

WHERE emp.Start\_date > DATE\_SUB(curdate(),INTERVAL '2' MONTH);

Graphical user interface, application

Description automatically generated

**2. Find the names of employees who are also a member and the books they have borrowed in the past month**

SELECT p.FName, p.Middle\_Name, p.Lname, br.BookID

FROM PERSON p

INNER JOIN EMPLOYEE emp

ON p.PID = emp.Employee\_ID

INNER JOIN MEMBER m

ON m.Member\_ID = emp.Employee\_ID,

BORROWING br

WHERE p.PID = br.PID and br.Date\_of\_issue > DATE\_sub(CURDATE(), INTERVAL '1' MONTH);

Graphical user interface, application

Description automatically generated

**3.Find the average number of books borrowed by the top five gold members in the library**

SELECT sq.Member\_ID, AVG(sq.Number\_of\_Books)

FROM (SELECT \*

FROM TOP\_GOLD\_MEMBER tg

LIMIT 5) as sq, BORROWING br

WHERE Br.PID = sq.Member\_ID;

Graphical user interface

Description automatically generated

**4. Find the name of publishers and the title of the most popular book for each publisher.**

SELECT pub.Publisher\_Name, popb.Title

FROM POPULAR\_BOOKS popb, PUBLISHES pb, PUBLISHER pub

WHERE popb.BookID = pb.Book\_ID AND pb.Publisher\_ID = pub.Publisher\_ID;

Graphical user interface, text, application

Description automatically generated

**5. Find names of books that were not borrowed in the last 5 months**.

SELECT Bk.Title

FROM BOOK Bk

WHERE Bk.Title NOT IN (SELECT Bk.Title

FROM BOOK Bk, BORROWING Br

WHERE Bk.Book\_ID = Br.BookID AND Br.Date\_of\_issue >= DATE\_SUB(curdate() , INTERVAL '5' MONTH));

Graphical user interface

Description automatically generated

**6. Find the members who have borrowed all the books wrote by the most popular author.**

SELECT main.PID, p.FName, p.Middle\_name, p.LName

FROM (SELECT ct.PID, SUM(Counts) as Total FROM

(SELECT br.PID, br.BooKID, COUNT(\*) AS Counts FROM BORROWING br

GROUP BY br.PID, br.BookID) AS ct

WHERE ct.BookID IN (SELECT b.Book\_id FROM BOOK b, WRITES w

WHERE b.Book\_id = w.Book\_id AND

w.Author\_id = (SELECT a.Author\_id

FROM AUTHOR a, WRITES w

WHERE a.Author\_id = w.Author\_id

GROUP BY w.Author\_id

ORDER BY count(\*) DESC LIMIT 1))

GROUP BY ct.PID) as main, PERSON p

WHERE main.Total = (SELECT Count(b.Book\_id) FROM BOOK b, WRITES w

WHERE b.Book\_id = w.Book\_id AND

w.Author\_id = (SELECT a.Author\_id

FROM AUTHOR a, WRITES w

WHERE a.Author\_id = w.Author\_id

GROUP BY w.Author\_id

ORDER BY Count(\*) DESC LIMIT 1)) AND

main.PID = p.PID;

**7. Find the Gold Member with the greatest number of guests.**

SELECT g.M\_ID,Count(\*) as Number\_of\_guests

FROM GOLD g, GUEST\_LOG gl

WHERE g.M\_ID = gl.M\_ID

GROUP BY g.M\_ID

ORDER BY COUNT(\*) DESC LIMIT 1;

Graphical user interface

Description automatically generated

**8. Find the year with the maximum number of books borrowed.**

SELECT YEAR(Br.Date\_of\_issue) , Count(\*) as Number\_of\_books

FROM BORROWING Br

GROUP BY YEAR(Br.Date\_of\_issue)

ORDER BY COUNT(\*) DESC LIMIT 1;

Graphical user interface, text, application

Description automatically generated

**9. Find the names of members who borrowed the most popular books.**

SELECT p.FName, p.Middle\_Name, p.Lname

FROM PERSON p, MEMBER m, POPULAR\_BOOKS pb, BORROWING br

WHERE p.PID = m.Member\_ID AND p.PID = br.PID AND pb.BookID = br.BookID;

Graphical user interface, application

Description automatically generated

**10. List all the employees that have enrolled into Gold membership within a month of being employed.**

SELECT emp.Employee\_ID, p.FName, p.Lname

FROM EMPLOYEE emp

JOIN PERSON p

ON emp.Employee\_ID = p.PID

JOIN MEMBER m

ON p.PID = m.Member\_ID

JOIN GOLD g

ON m.Member\_ID = g.M\_ID

WHERE m.Enrollment\_date < DATE(emp.Start\_Date + INTERVAL 1 MONTH);

Graphical user interface, application

Description automatically generated

**11. Find the name of members who have been a silver member for over 5 years.**

SELECT p.Fname, p.Lname

FROM PERSON p

JOIN MEMBER m

ON m.Member\_ID = p.PID

JOIN SILVER s

ON s.M\_ID = m.Member\_ID

WHERE m.Enrollment\_date >= DATE\_SUB(curdate(), INTERVAL 5 YEAR);

Graphical user interface, application

Description automatically generated

**12. Find the names of the potential gold members and number of books they borrowed in the last year.**

SELECT p.FName, p.Middle\_name, p.LName, Count(\*) as No\_of\_books

FROM POTENTIAL\_GOLD\_MEMBER p, BORROWING br

WHERE p.PID = br.PID

GROUP BY p.PID, br.BookID;

Graphical user interface, application

Description automatically generated

**13. List the employee who trained the greatest number of receptionists.**

SELECT p.FName, p.Middle\_name, p.LName, r.Trainer\_id , COUNT(\*)

FROM RECEPTIONIST r, EMPLOYEE emp, PERSON p

WHERE r.Recep\_ID = emp.Employee\_id AND

r.Recep\_ID = p.PID

GROUP BY r.Trainer\_id

ORDER BY COUNT(\*) DESC;

Graphical user interface, application

Description automatically generated

**14. List the Cataloging Managers who cataloged all categories every week in past 4 weeks.**

SELECT p.FName, p.Middle\_name, p.LName, ct.CatMangID

FROM (SELECT c.CatMangID, COUNT(c.CatMangID) as Counts

FROM CATELOG\_ACTIVITY c WHERE c.Catalog\_date > DATE(curdate() - INTERVAL 4 WEEK)

GROUP BY c.CatMangID) AS ct, Employee emp, Person p

WHERE ct.Counts = 4 AND

emp.Employee\_id = ct.CatMangid AND

p.PID = ct.CatMangid;

Graphical user interface

Description automatically generated