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 :

PERSON

PID_	FName	Middle Name	Lname	DOB	Gender	Address	Person_type

FD : PID -> FName , Middle Name ,Lname , DOB , Gender, Address, Person_type

MEMBER

Member_ID	Membership_ type	Enrollment_date

FD : Member_ID --> Membership_type , Enrollment_date

GOLD

M_ID

FD : M_ID -> M_ID (Trivial Dependency)

SILVER

M_ID

FD : M_ID -> M_ID (Trivial Dependency)

RECEPTIONIST

Recep_ID	Trainer_ID

FD : Recep_ID -> Trainer _ID

TRAINER

Trainer_ID

FD : Trainer_ID -> Trainer_ID (Trivial Dependency)

LIBRARY_SUPERVISOR

LibSup_ID	Trainer_ID

FD : LibSup_ID -> Trainer_ID

CATALOGING_MANAGER

CatMang_ID	Trainer_ID

FD : CatMang_ID -> Trainer_ID

EMPLOYEE

Employee_ID	Start_Date	Age	Type

FD : Employee_ID -> Start_Date , Age, Type

BOOK

Book_ID	Title	Other_Info	Category_number(FK)

FD : Book_ID -> Title, Other_Info , Category_number

PUBLISHER

Publisher_ID	Publisher_Name	Established_Year	Address

FD : Publisher_ID -> Publisher_Name, Estabished_Year , Address

AUTHOR

Author_ID	Author_Name	Year_of_Operation	Style_of_Writing

FD : Author_ID -> Author_Name, Year_of_Operation , Style_of_Writing

CATEGORY

Category_Number

FD : Category_Number -> Category_Number (Trivial Dependency)

PAYMENT

Payment_ID	Amount	Time	PMethod

FD : Payment_ID -> Amount, Time, PMethod

GUEST_LOG

Guest_ID	M_ID	Fname	Middle_name	Lname	Address	Contact

FD : Guest_ID -> M_ID , Fname, Middle_name, Lname, Address, Contact

LIBRARY_CARD

Member_ID	Card_Num

FD : {Member_ID , Card_Num } -> { Member_ID, Card_Num}

PROMOTION

Code	Member_ID	Card_Num	Description

FD : { Code, Member_ID , Card_Num } -> Description

COMMENTS_ON

PID	Book_ID	Comment_Time	Rating	Content

FD : {PID, Book_ID } -> Comment_Time, Rating , Content

PUBLISHES

Book_ID	Publisher_ID

FD : {Book_ID, Pubisher_ID } -> { Book_ID , Publisher_ID }

WRITES

Book_ID	Author_ID

FD : { Book_ID, Author_ID } -> { Book_ID , Author_ID}

CATALOG_ACTIVITY

CatMangID	Category_Num	Catalog_Date

FD : { CatMangID, Category_Num} -> { Catalog_Date}

BORROWING

Person: PID	Recep_ID	BookID	Payment_ID	Date_of_issue	Due_date

FD : { PID, Recep_ID, Book_ID, Payment_ID} -> Date_of_issue, Due_date

PHONE_NUMBERS

PID	Phone_num

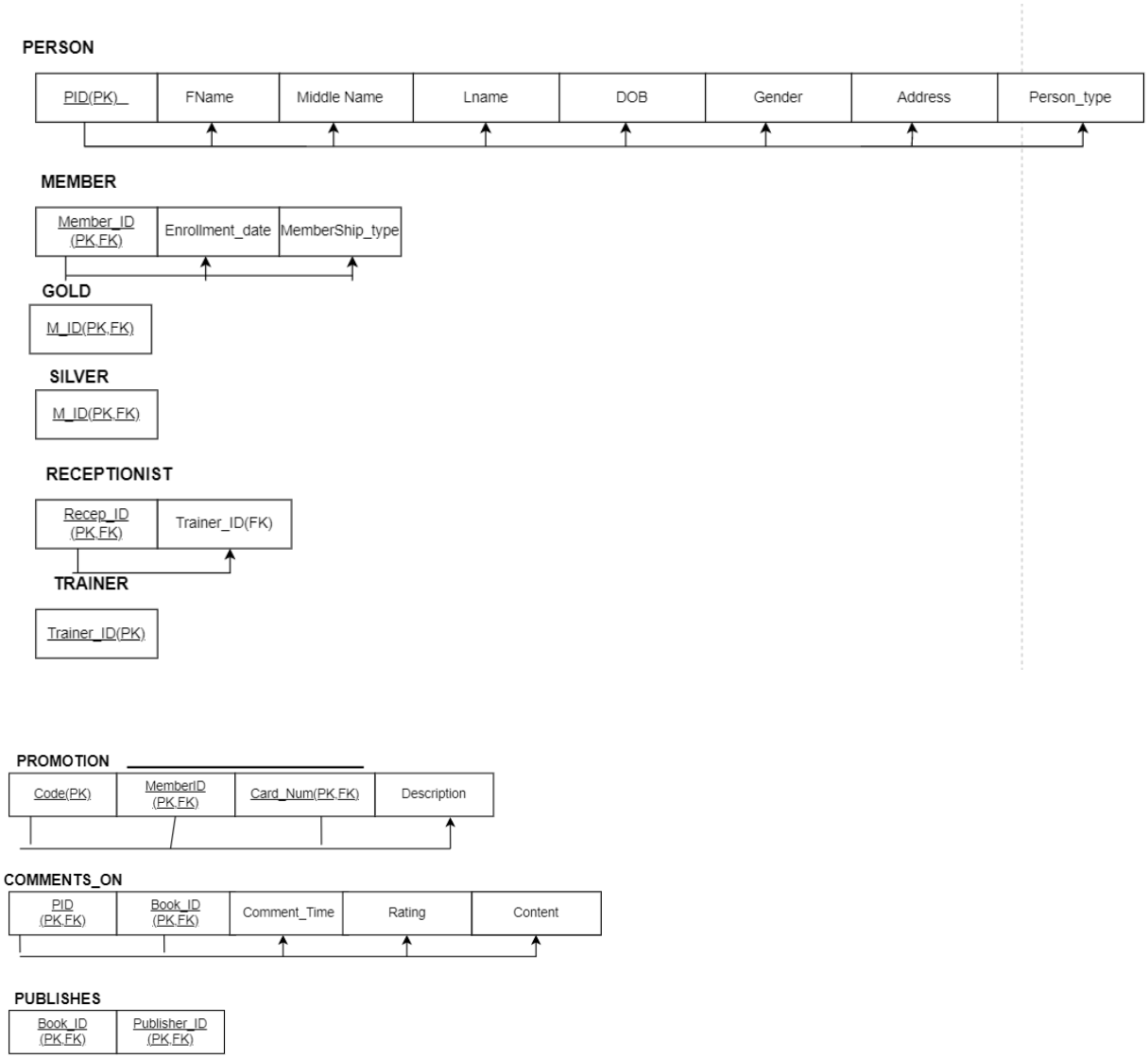
FD : { PID, Phone_num } -> { PID, Phone_num }

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.



LIBRARY_SUPERVISOR

LibSup_ID (PK,FK)	Trainer_ID

CATALOGING_MANAGER

CatMang_ID (PK,FK)	Trainer_ID(FK)

EMPLOYEE

Employee_ID (PK,FK)	Start_Date	Age	Type

BOOK

Book_ID(PK)	Title	Other_Info	Category_number(FK)

PUBLISHER

Publisher_ID(PK)	Publisher_Name	Established_Year	Address

AUTHOR

Author_ID(PK)	Author_Name	Year_of_Operation	Style_of_Writing

CATEGORY

Category_Number (PK)

PAYMENT

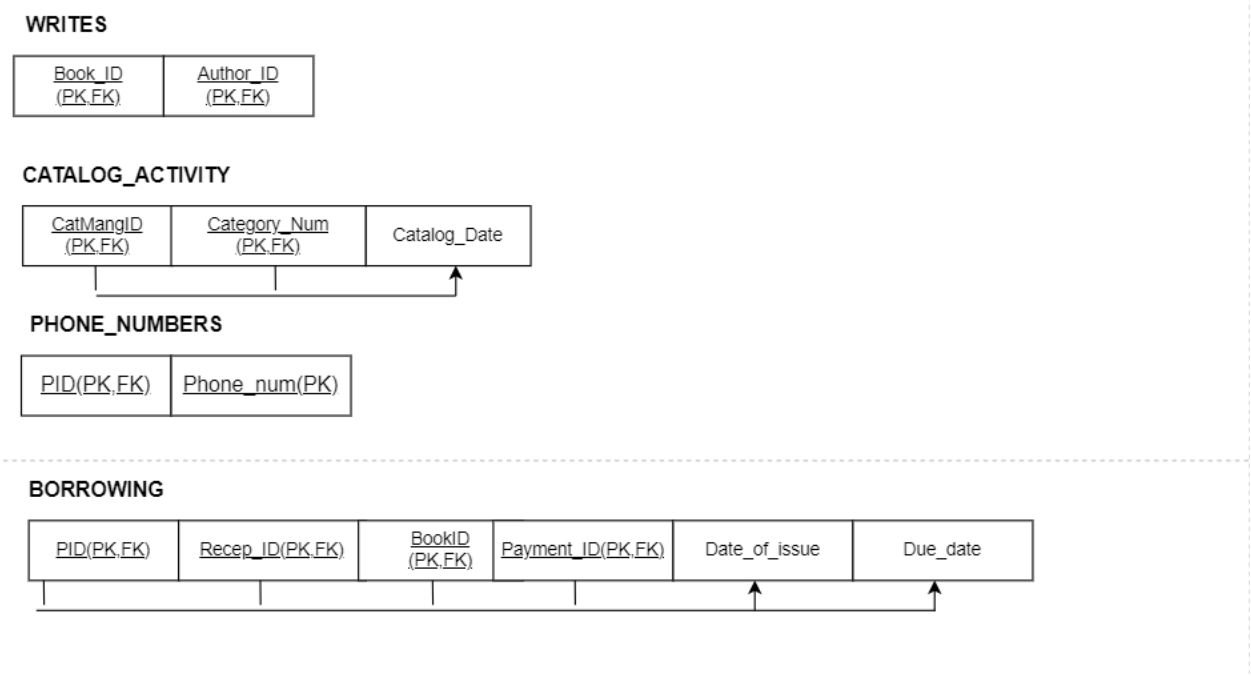
Payment_ID(PK)	Amount	Time	PMethod

GUEST_LOG

Guest_ID (PK)	M_ID(PK,FK)	Fname	Middle_name	Lname	Address	Contact

LIBRARY_CARD

MemberID(PK,FK)	Card_Num(PK)



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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Member_ID	FName	LName	Enrollment_date	Number_of_books
P001	Tottempudi	Chandra	2022-04-25	7

TOP_GOLD_MEMBER 6

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BookID	Title	No_of_books
B001	Harry Potter - 1	2
B002	Harry Potter - 2	2
B003	Harry Potter - 3	2
B009	2 States	2
B010	Amar Chitra Katha	2
B006	Harry Potter - 6	1
B011	Ramayana	1
B008	Half Girlfriend	1
B005	Harry Potter - 5	1
B007	Harry Potter - 7	1

POPULAR_BOOKS 7

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Publisher_name	Average_rating_score
J K Rowling Publishers	4.6667
Chetan Bhagat Publishers	4.0000
Chetan Bhagat Publishers	5.0000
Valmiki Publishers	4.0000

BEST_RATING_PUBLISHER 8

Output

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PID	Fname	Middle_name	LName	Phone_num

POTENTIAL_GOLD_MEMBER 9

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Author_ID	Author_Name	No_of_books
A001	J K Rowling	2
A001	J K Rowling	2
A001	J K Rowling	2
A001	J K Rowling	2
A002	Chetan Bhagat	2
A003	Anant Pai	2
A004	Valmiki	2
A001	J K Rowling	1
A001	J K Rowling	1
A001	J K Rowling	1
A002	Chetan Bhagat	1

POPULAR_AUTHOR 11 ×

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);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

LibSup_ID	FName	Middle_Name	Lname	Start_date
P005	Naga	Sundar	Mysore	2022-04-05

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);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FName	Middle_Name	Lname	BookID
Preethi	L	Reddy	B004

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Member_ID	AVG(sq.Number_of_Books)
P001	7.0000

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Publisher_Name	Title
J K Rowling Publishers	Harry Potter - 1
J K Rowling Publishers	Harry Potter - 2
J K Rowling Publishers	Harry Potter - 3
J K Rowling Publishers	Harry Potter - 4
J K Rowling Publishers	Harry Potter - 5
J K Rowling Publishers	Harry Potter - 6
J K Rowling Publishers	Harry Potter - 7
Chetan Bhagat Publishers	Half Girlfriend
Chetan Bhagat Publishers	2 States
Anant Pai Publishers	Amar Chitra Katha
Valmiki Publishers	Ramayana

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));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title
Half Girlfriend

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

M_ID	Number_of_guests
P222	2

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fet

YEAR(Br.Date_of_issue)	Number_of_books
2022	13

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FName	Middle_Name	Lname
Tottempudi	Sekhar	Chandra
Hima	Sri	Tipirineni
Ram	Koteshwar	Kandimala
Anusha	Sai	Sharma
Harsha	Shri	Gandani
Mani	Sri	Henry
Rama	Rao	M
Hamsini	S	Paladugu
Naga	Sundar	Mysore
Preethi	L	Reddy

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);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Employee_ID	FName	Lname
P005	Naga	Mysore

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);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Fname	Lname
Ram	Kandimala
Rama	M
Harsha	Gandani
Mani	Henry
Anusha	Sharma
Varma	Raj
Vicky	Sharma

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Fname	Middle_name	LName	No_of_books

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FName	Middle_name	LName	Trainer_id	COUNT(*)
Preethi	L	Reddy	TR002	1
Anusha	Sai	Sharma	TR003	1
Hima	Bindu	Gutta	TR007	1

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	FName	Middle_name	LName	CatMangID