Phase IV

(CS 6360) Database Design Term Project

Group 18

A. Problem Description:

Wonder Library is a library for all ages. Wonder Library would like one relational database to be able to smoothly carry out their work in an organized way. The library has following important modules: Person, Employee, Member, Books, Publishers, Authors and Payment.

A Person can be an Employee or a Member. Employee can also be a Member. Details of a person such as Person ID, Name (First, Middle, Last), Address, Gender, Date of Birth, and Phone number (one person can have more than one phone number) are recorded. Employee must be at least 18 years old. The Person ID should have the format "PXXX" where X is a number from 0 to 9 (Hint: you can use regexp_like() function).

Each member is issued a library card. The library card details such as card ID, date of issue, membership level (Silver or Gold) and other information are stored. Library sometimes may provide Promotions associate with library cards. Each Promotion includes a unique Promotion code, and its description.

Employee can be one of three classes: Library Supervisors, Cataloging Managers or Receptionists. The start date of employment is recorded. Receptionist must be trained by a Trainer, a Trainer can be Library Supervisor or a Cataloging Manager. Library Supervisor and Cataloging Manager can train multiple Receptionists.

Each member is classified as a Silver or Gold. A Guest log is maintained for the Gold members, which stores information such as the Gold member's Card ID, guest ID, guest name, guest address, and guest contact information. There are temporary IDs that a person gets when they visit as a guest of a Gold member. Each guest ID is not unique in whole system, and only unique among all guest of a Gold member.

Books details such as book ID, book title and other information are stored. Books are classified as 3 categories: Cate. 1, Cate. 2 and Cate. 3. Each Cataloging Manager is responsible for cataloging one category per day, but may catalog different categories at different days. Person can make comments to the Books. The comments include comment time, rating score (can be 1,2,3,4,5), and comment main contents.

A publisher can publish more than one book, but a book is assumed to be published by a single publisher. The publisher details such as publisher ID and publisher name and other information (you can add assumptions) are stored. Author details such as author ID, author name and other information is stored. One book can have multiple authors and one author can write more than one book.

A receptionist maintains records of borrowing details. Borrowing details are stored containing information about the borrowed book, the date of issue and due date of return, the details about the person borrowing the book, details of the receptionist and payment detail. Borrowed details are stored only when a person borrows a book. Payment detail such as Payment ID, payment method (cash, debit/credit card), payment time and amount are stored.

B. Project Questions

1. **Is the ability to model superclass/subclass relationships likely to be important in the Wonder Library management system like above? Why or why not?**

Ans. Yes, the superclass/ subclass relationships are important in library system as they can store common attributes in a single relation and helps to reduce redundancy of data.

2. **Can you think of 5 more rules (other than those explicitly described above) that are likely to be used in above environment? Please describe how your design would be changed to satisfy your additional rules?**

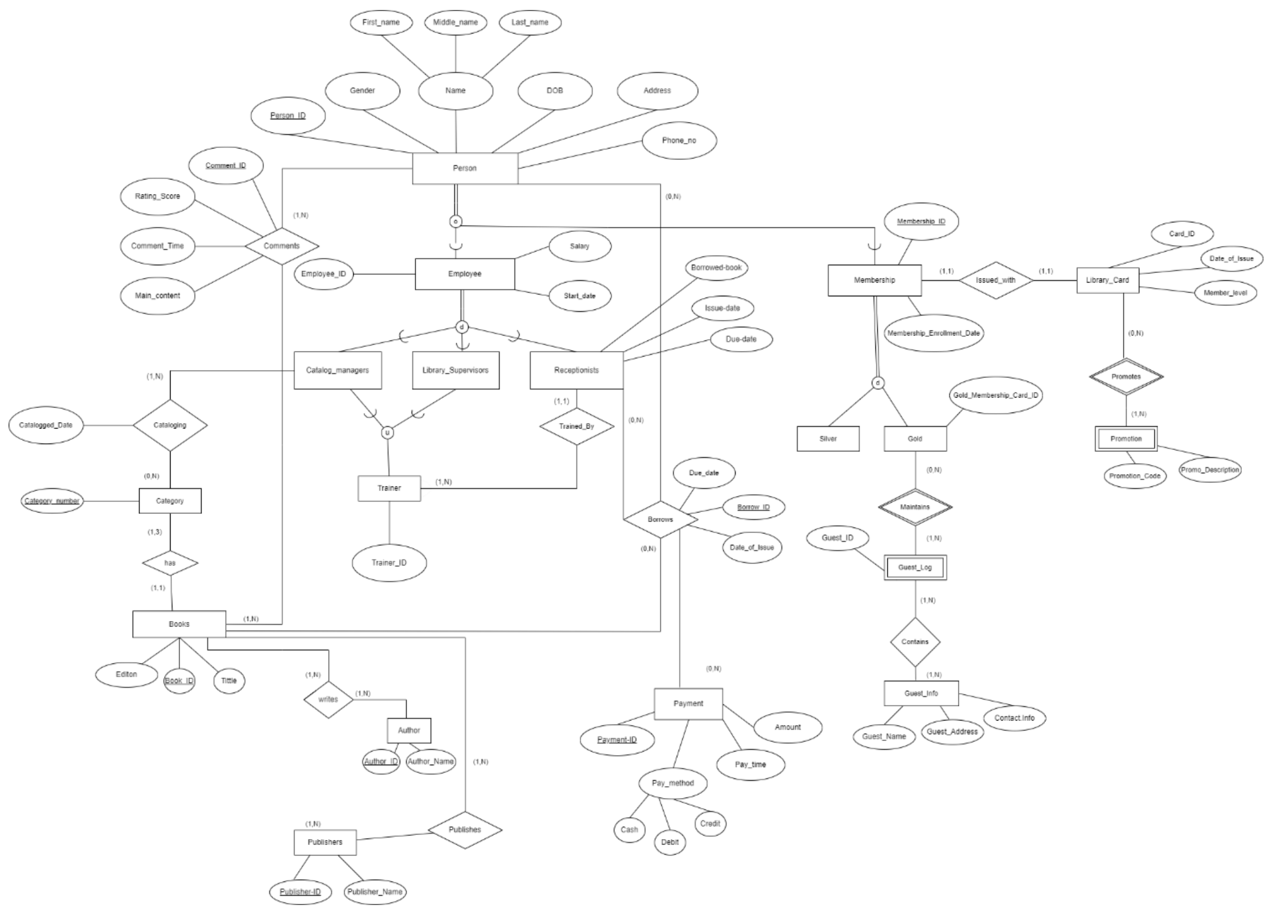
Ans.

A. A person could be given an option to check online for the availability of the books beforehand. This can be connected to the Books entity by creating an attribute for the availability of books
B. The Silver membership can also have a guest log. This can be implemented by updating the guest_log relation.
C. A self-service type of submission can be implemented by giving an option to the person for returning the borrowed books.
D. Books can be further categorized into more categories so that it would be easier for the person to find the book he/she was searching. This can be implemented by increasing the number of categories.
E. A time slot can be provided for the persons to pickup/borrow their books for a seamless experience.

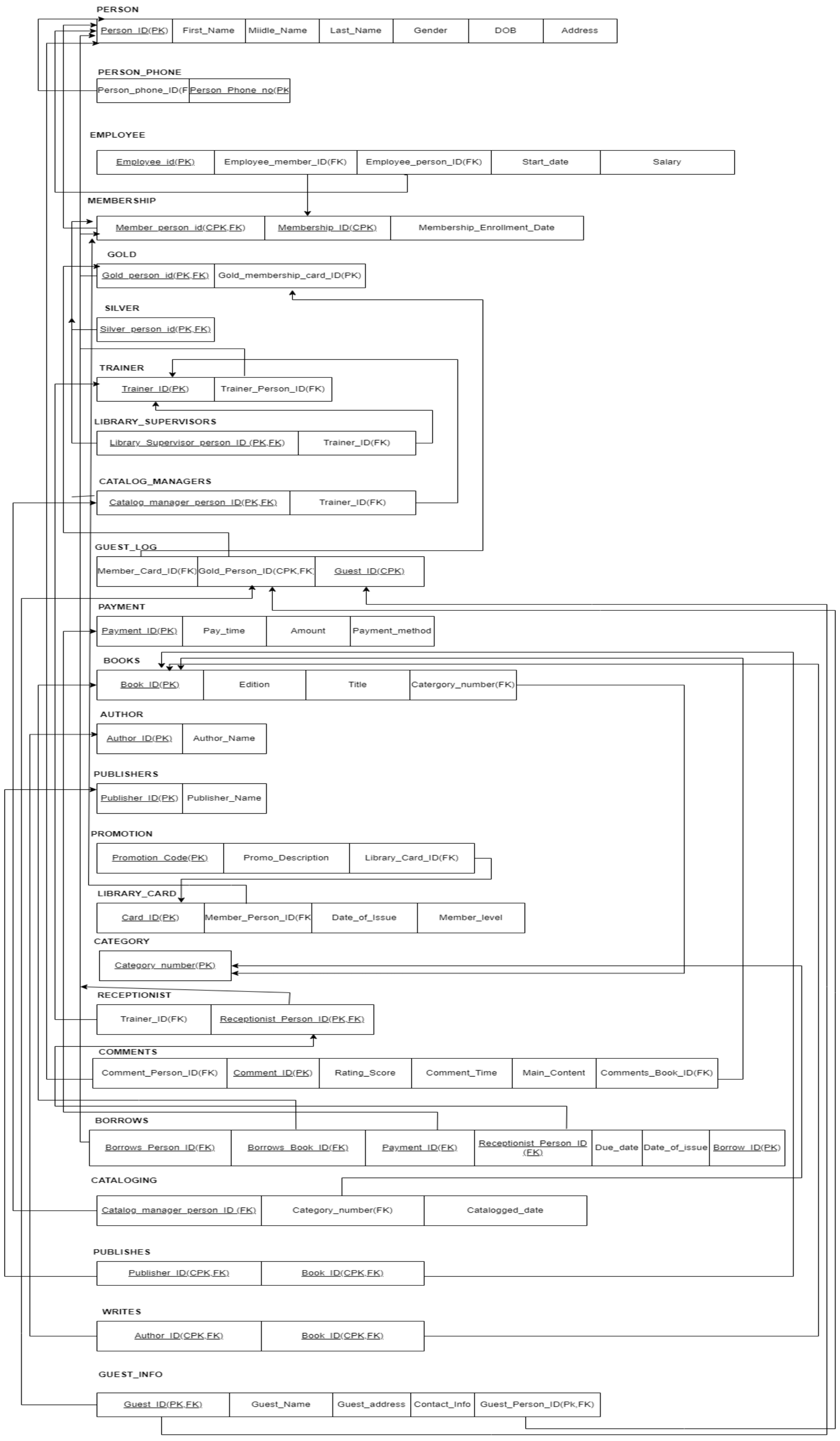
3. **Justify using a Relational DBMS like Oracle for this project (Successfully design a relational database system, please show all the implementation in final report at Phase IV).**

   Ans. For creating the database of this Wonder Library environment, we used MySQL as our RDBMS because it helps to control redundancy of data, security, ease of use, and its flexibility. We can design the database of this system in the third normal form.

C. EER diagram



D. Relation Schema after normalization



E. Dependency Diagram

PERSON

Person_ID(PK,FK)	First_Name	Miidle_Name	Last_Name	Gender	DOB	Address

Person_ID → First_Name, Middle_Name, Last_Name, Gender, DOB, Address

PERSON_PHONE

Phone_Person_ID(FK)	Person_Phone_no(PK)

Person_Phone_no → Phone_Person_ID

EMPLOYEE

Employee_id(PK)	Employee_member_ID(FK)	Employee_person_ID(FK)	Start_date	Salary

Employee_Id → Employee_member_ID, Employee_person_ID, Start_date, Salary

MEMBERSHIP

Member_person_id(CPK,FK)	Membership_ID(CPK)	Membership_Enrollment_Date

Member_person_id, Membesrship_ID → Membership_Enrollment_Date

GOLD

Gold_person_id(PK,FK)	Gold_membership_card_ID(PK)

Gold_person_id →Gold_membership_card

SILVER

Silver_person_id(PK,FK)

Silver_person_id

TRAINER

Trainer_ID(PK)	Trainer_Person_ID(FK)

Trainer_ID → Trainer_Person_ID

LIBRARY_SUPERVISORS

Library_Supervisor_person_ID (PK,FK)	Trainer_ID(FK)

Library_Supervisor_person_ID → Trainer_ID

CATALOG_MANAGERS

Catalog_manager_person_ID(PK,FK)	Trainer_ID(FK)

Catalog_manager_person_ID → Trainer_ID

GUEST_LOG

Member_Card_ID(FK)	Gold_Person_ID(CPK,FK)	Guest_ID(CPK,PK)

Guest_ID → Member_Card_ID, Gold_Person_ID

PAYMENT

Payment_ID(PK)	Pay_time	Amount	Payment_method

Payment_ID → Pay_time, Amount, Payment_method

BOOKS

Book_ID(PK)	Edition	Title	Catergory_number(FK)

Book_ID → Edition, Title, Catergory_number

AUTHORS

Author_ID(PK)	Author_Name

Author_ID → Author_Name

PUBLISHERS

Publisher_ID(PK)	Publisher_Name

Publisher_ID → Publisher_Name

PROMOTION

Promotion_Code(PK)	Promo_Description	Library_Card_ID(FK)

Promotion_Code → Promo_Description, Library_Card_ID

LIBRARY_CARD

Card_ID(PK)	Member_Person_ID(FK	Date_of_Issue	Member_level

Card_ID → Member_Person_ID, Date_of_Issue, Member_level

CATEGORY

Category_number(PK)

Category_number

RECEPTIONIST

Trainer_ID(FK)	Receptionist_Person_ID(PK)

Receptionist_Person_ID → Trainer_ID

COMMENTS

Comment_Person_ID(FK)	Comment_ID(PK)	Rating_Score	Comment_Time	Main_Content	Comments_Book_I

Comment_ID → Comment_Person_ID, Rating_Score, Comment_Time, Main_Content, Comments_Book_ID

BORROWS

Borrows_Person_ID(FK)	Borrows_Book_ID(FK)	Payment_ID(FK)	Receptionist_Person_ID (FK)	Due_date	Date_of_issue	Borrow_ID(PK)

Borrows_Person_ID, Borrows_Book_ID, Payment_ID, Receptionist_Person_ID → Due_date, Date_of_issue

CATALOGING

Catalog_manager_person_ID (FK)	Category_number(FK)	Catalogged_date

Catalog_managers_person_ID → Category_number, Catalogged_Date

PUBLISHES

Publisher_ID(CPK,FK)	Book_ID(CPK,FK)

Publisher_ID, Book_ID

WRITES

Author_ID(CPK,FK)	Book_ID(CPK,FK)

Author_ID, Book_ID

GUEST_INFO

Guest_ID(PK,FK)	Guest_Name	Guest_address	Contact_Info	Guest_Person_ID(Pk,FK)

Guest_ID(PK,FK) → Guest_Name, Guest_address, Contact_Info, Guest_Person_ID

F. All requested SQL statements (Solution for Phase III-c, d and e).

Ans:

- create database dbdesign;
- use dbdesign;

- create table Person(person_id varchar(4),
  first_name varchar(20), middle_name varchar(20), last_name varchar(20),
  gender varchar(20), dob Date, Address varchar(20),
  Primary Key(person_id));

- create table Person_phone(person_phone_id varchar(20),
  person_phone_no varchar(20),
  Primary Key(person_phone_no) ,
  FOREIGN KEY (person_phone_id) REFERENCES Person(person_ID));

- create table membership(member_person_id varchar(20) NOT NULL,
  membership_id varchar(20) NOT NULL,
  membership_enrollment_date date,
  Primary Key(member_person_id,membership_id),
  UNIQUE (membership_id),
  FOREIGN KEY (member_person_id) REFERENCES Person(person_ID));

- create table employee(employee_id varchar(20) NOT NULL,
  employee_person_id varchar(20) NOT NULL,
  employee_member_id varchar(20) NOT NULL,
  start_date date, salary varchar(10) NOT NULL,
  Primary key(employee_id),
  FOREIGN KEY (employee_person_id) REFERENCES Person(person_ID) ,
  FOREIGN KEY (employee_member_id) REFERENCES membership(membership_id));

- create table gold(gold_person_id varchar(20),
  gold_membership_card_id varchar(20),
  Primary key(gold_person_id,gold_membership_card_id),
  UNIQUE(gold_membership_card_id),
  FOREIGN KEY (gold_person_id) REFERENCES Person(person_ID));

- create table silver(silver_person_id varchar(20),
  Primary key(silver_person_id),
  FOREIGN KEY (silver_person_id) REFERENCES Person(person_ID));

- create table trainer(trainer_id varchar(20),
  trainer_person_id varchar(20),
  Primary key(trainer_id),
  FOREIGN KEY (trainer_person_id) REFERENCES Person(person_ID));

- create table library_supervisors(library_supervisor_person_id varchar(20),
  trainer_id varchar(20),
  Primary key(library_supervisor_person_id),

FOREIGN KEY (trainer_id) REFERENCES trainer(trainer_id),
FOREIGN KEY (library_supervisor_person_id) REFERENCES person(person_ID));

- create table catalog_managers(catalog_manager_person_id varchar(20),
trainer_id varchar(20),
Primary key(catalog_manager_person_id),
FOREIGN KEY (trainer_id) REFERENCES trainer(trainer_id),
FOREIGN KEY (catalog_manager_person_id) REFERENCES person(person_ID));

- create table guest_log(guest_id varchar(20),
gold_person_id varchar(20),
member_card_id varchar(20),
Primary key(guest_id),
Foreign Key(gold_person_id) REFERENCES gold(gold_person_id),
Foreign Key(member_card_id) REFERENCES gold(gold_membership_card_id));

- create table payment(payment_id varchar(20),
pay_time time,
amount int ,
payment_method varchar(20),
Primary key(payment_id));

- create table category(category_number varchar(20),
Primary key(category_number));

- create table books(book_id varchar(20),
edition varchar(10),
title varchar(20) ,
category_number varchar(20),
Primary key(book_id),
Foreign key(category_number) REFERENCES CATEGORY(category_number));

- create table author(author_id varchar(20),
author_name varchar(10),
Primary key(author_id));

- create table publisher(publisher_id varchar(20),
publisher_name varchar(20),
Primary key(publisher_id));

- create table library_card(card_id varchar(20),
member_person_id varchar(20),
date_of_issue date,
member_level varchar(10),
Primary key(card_id),
FOREIGN KEY(member_person_id) REFERENCES person(person_id));

- create table promotion(promotion_code varchar(20),
promo_description varchar(20),
library_card_id varchar(20),

```
     Primary key(promotion_code),
     FOREIGN KEY(library_card_id) REFERENCES library_card(card_id));

➢    create table receptionist(receptionist_person_id varchar(20),
     trainer_id varchar(20),
     Primary key(receptionist_person_id),
     FOREIGN KEY(receptionist_person_id) REFERENCES person(person_id),
     FOREIGN KEY(trainer_id) REFERENCES trainer(trainer_id));

➢    create table comments(comment_person_id varchar(20),
     comment_id varchar(20),
     rating_score varchar(20),
     comment_time time,
     main_content varchar(20),
     comments_book_id varchar(20),
     Primary key(comment_id),
     FOREIGN KEY(comment_person_id) REFERENCES person(person_id),
     FOREIGN KEY(comments_book_id) REFERENCES books(book_id));

➢    create table borrows(borrow_id varchar(20),
     borrows_person_id varchar(20),
     borrows_book_id varchar(20),
     Payment_id varchar(20),
     Receptionist_person_id varchar(20),
     due_date date,
     date_of_issue date,
     Primary key(borrow_id,payment_id, borrows_person_id,
     borrows_book_id,Receptionist_person_id),
     FOREIGN KEY(borrows_person_id) REFERENCES person(person_id),
     FOREIGN KEY(borrows_book_id) REFERENCES books(book_id),
     FOREIGN KEY(Receptionist_person_id) REFERENCES person(person_id),
     FOREIGN KEY(Payment_id) REFERENCES payment(payment_id));

➢    create table cataloging(catalog_manager_person_id varchar(20),
     category_number varchar(20),
     Primary key(catalog_manager_person_id),
     FOREIGN KEY(catalog_manager_person_id) REFERENCES person(person_id),
     FOREIGN KEY(category_number) REFERENCES category(category_number));

➢    create table publishes(publisher_id varchar(20),
     book_id varchar(20),
     Primary key(publisher_id, book_id),
     FOREIGN KEY(book_id) REFERENCES books(book_id),
     FOREIGN KEY(publisher_id) REFERENCES publisher(publisher_id));

➢    create table writes(author_id varchar(20),
     book_id varchar(20),
     Primary key(author_id, book_id),
     FOREIGN KEY(book_id) REFERENCES books(book_id),
     FOREIGN KEY(author_id) REFERENCES author(author_id));
➢    create table Guest_info(guest_id varchar(20),
	guest_person_id varchar(20),
	guest_name varchar(20),
	guest_address varchar(20),
	contact_info varchar(20),
	Primary key(guest_id,guest_person_id),
	Foreign key(guest_id) REFERENCES guest_log(guest_id),
	Foreign key(guest_person_id) REFERENCES guest_log(gold_person_id));
```

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.

Ans.

```
CREATE VIEW view_1
As
SELECT first_name ,last_name,membership_enrollment_date
FROM person p
inner join membership mm on mm.member_person_id = p.person_id
where p.person_id in (select distinct borrows_person_id from borrows
where month(date_of_issue) = month(now())-1
group by borrows_person_id
having count(borrows_person_id) > 5
order by count(borrows_person_id) desc) ;
```

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

Ans.

```
CREATE VIEW view_2
AS
SELECT TITLE,EDITION
FROM BOOKS
WHERE BOOK_ID IN (
SELECT DISTINCT BORROWS_BOOK_ID
FROM BORROWS
WHERE YEAR(date_of_issue) = YEAR(now())-1
GROUP BY BORROWS_BOOK_ID
ORDER BY count(BORROWS_BOOK_ID) DESC ) ;
Select * from view_2 LIMIT 5;
```

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

Ans.

```
CREATE VIEW view_3
AS
SELECT pr.publisher_name
```

```
FROM publishes P
INNER JOIN publisher PR
ON P.publisher_id = PR.publisher_id
where P.book_id in (SELECT DISTINCT comments_book_id FROM comments
group by comments_book_id
having avg(rating_score) >= 4);
```

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.

Ans.

```
CREATE view view_4
As
SELECT concat(FIRST_NAME,LAST_NAME) AS person_name ,person_phone_no,p.person_id as ID
FROM person  p
INNER JOIN person_phone pp on p.person_id = pp.person_phone_id
INNER JOIN borrows b on b.borrows_book_id= p.person_id
WHERE b.borrows_person_id IN (SELECT silver_person_id FROM silver);
```

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

Ans.

```
CREATE VIEW view_5
AS
SELECT A.author_id,A.author_name
FROM author A
INNER JOIN writes W on W.author_id = A.author_id
Where W.book_id in (
SELECT DISTINCT borrows_book_id FROM borrows
HAVING COUNT(borrows_book_id) > 1
ORDER BY COUNT(borrows_book_id) DESC ) ;
```

Show the SQL statements 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 library_supervisor_person_id AS ID, CONCAT(first_name,last_name) as
        Name,employee_id,gender,dob,Address
        FROM library_supervisors LS
        INNER JOIN employee E ON LS.library_supervisor_person_id = E.employee_person_id
        INNER JOIN person P ON LS.library_supervisor_person_id = P.person_id
        WHERE MONTH(start_date) BETWEEN (MONTH(NOW())-1) AND (MONTH(NOW())-2);
```

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

```
SELECT CONCAT(first_name,last_name) as Name,BO.title

        FROM person P
```

```
INNER JOIN membership M ON P.person_id = M.member_person_id
INNER JOIN borrows B ON B.borrows_person_id = P.person_id
INNER JOIN books BO ON BO.book_id = B.borrows_book_id
WHERE M.membership_id IN (SELECT DISTINCT employee_member_id FROM employee)
AND  MONTH(date_of_issue) = MONTH(now())-1 ;
```

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

```
SELECT G.gold_person_id as ID,count(borrows_person_id)/count(G.gold_person_id)  AS
NO_OF_BOOKS
FROM BORROWS B INNER JOIN gold G on G.gold_person_id = B.borrows_person_id
group by borrows_person_id
ORDER BY count(borrows_person_id)/count(G.gold_person_id) DESC
LIMIT 5;
```

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

```
SELECT DISTINCT publisher_name , title
FROM publisher P
INNER JOIN publishes ps ON ps.publisher_id = P.publisher_id
INNER JOIN comments C ON C.comments_book_id = ps.book_id
INNER JOIN books B ON B.book_id = ps.book_id ;
```

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

```
SELECT TITLE  FROM books B
INNER JOIN borrows BS ON BS.borrows_book_id = B.book_id
WHERE MONTH(date_of_issue) not BETWEEN (MONTH(NOW())-1) AND (MONTH(NOW())-5);
```

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

```
Select author_id from
(select Distinct comments_book_id as
Book_id,sum(rating_score)/count(comments_book_id) as popular
from comments
order by sum(rating_score)/count(comments_book_id) desc
limit 1) as derived_3
inner join writes w on w.book_id = derived_3.Book_id;
```

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

```
Select concat(first_name,last_name) as name from
(
Select distinct gold_person_id,count(*) from guest_log
order by count(*) desc
limit 1) as derived_5
inner join person p on p.person_id = derived_5.gold_person_id;
```

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

```
SELECT Year FROM
(
SELECT distinct year(date_of_issue) As Year,count(*)
FROM borrows
group by date_of_issue
order by count(*) desc
) as Derived
Limit 1;
```

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

```
Select concat(first_name,last_name) as name
from borrows
Inner join person p on p.person_id = borrows_person_id
where borrows_book_id in ( Select Book_id from
                              (select Distinct comments_book_id as
Book_id,sum(rating_score)/count(comments_book_id) as popular
                              from comments
                              order by sum(rating_score)/count(comments_book_id) desc
                              limit 1)
                              as derived_6 ) ;
```

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

```
select concat(first_name,last_name) as Name ,
e.employee_member_id,e.employee_person_id from membership m
inner join gold g on g.gold_person_id=m.member_person_id
inner join employee e on e.employee_member_id=m.membership_id
inner join person p on p.person_id=e.employee_person_id
where m.membership_enrollment_date-e.start_date<31;
```

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

```
Select concat(first_name,last_name) As Name  from membership mm
Inner join silver s ON s.silver_person_id = mm.member_person_id
Inner join person p on p.person_id = s.silver_person_id
where year(now())-Year(mm.membership_enrollment_date)  > 5;
```

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

```
SELECT person_name, Count(*) as No_of_books
FROM view_4 p, BORROWS b
          WHERE p.person_id = b.borrows_person_id
                    GROUP BY p.person_id, b.borrows_book_id;
```

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

```
Select Name from
(
select distinct r.trainer_id,concat(first_name,last_name) as Name, count(*)
from receptionist r
inner join trainer t on t.trainer_id=r.trainer_id
inner join person p on  p.person_id = t.trainer_person_id
) as Derive
Limit 1;
```

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

```
Select name from
(
Select distinct
catalog_manager_person_id,c.category_number,concat(first_name,last_name) as name
from cataloging ct
inner join category c on ct.category_number = c.category_number
inner join person p on p.person_id = ct.catalog_manager_person_id
where week(ct.catalogged_date) between week(now()) and (week(now())-4)
group by catalog_manager_person_id,category_number
) as derived_2;
```