Project: Wonder Library

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 Employee has Employee ID, which should have the format "EXXX" where X is a number from 0 to 9 (Hint: you can use regexp_like() function). Note: "E000", "E999" are all valid employee IDs, "E01" or "E0001" are invalid employee IDs.

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. The 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. Receptionists must be trained by a Trainer, a Trainer can be Library Supervisor or a Cataloging Manager. Each Trainer has a trainer certificate with unique certificate number. The certificate issuing date is also recorded. A Trainer can train multiple Receptionists.

Each member is classified as a Silver member or Gold member. A Guest log is maintained for the Gold members, which stores information such as the Gold member's library card ID, guest ID, guest name, guest address, and guest contact information. Guest IDs are temporary IDs that a person gets when they visit as a guest of a Gold member. Each guest ID is not unique in the 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 only 3 categories: Cate. 1, Cate. 2 and Cate. 3. Each category has a description of what this category is about (e.g. topics, contents, types of the books in the category). Each Cataloging Manager is responsible for cataloging books. They can only catalog one category per day, but may catalog different categories on 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 are 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.

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?**
   The ability to model superclass/subclass relationships is crucial to wonder library project because it facilitated the creation of entities that fell under same category. For example, Employee is the superclass in our EER diagram and its subclasses are receptionist, library supervisor, catalog manager. While we are able to capture employee level details, we are also able to capture designation level details because of subclasses that we created.
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?**
   Additional Rules:
   A. An Author can be a member of the library.
   B. A Publisher can be a member of the library.
   C. An Author can't comment or rate on his own books.
   D. A Publisher can't comment or rate on his own books.
   E. Both Author and Publisher can comment on books that are not published or authored by them.

   Changes in design:

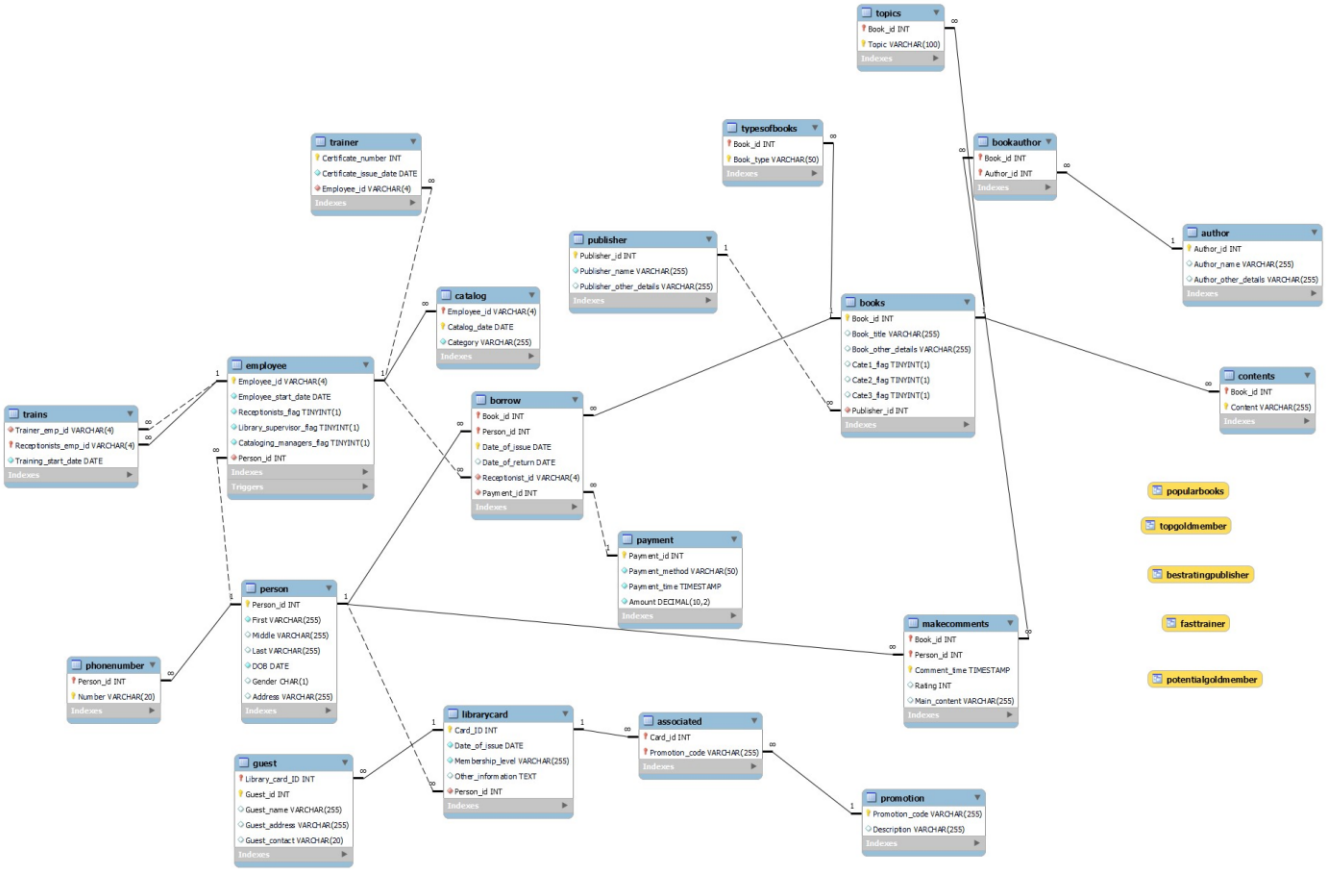
   I. Author and Publisher would be become subclass of Person entity.
   II. Person entity will hold two extra flags, one for Author and another for publisher.
   III. There would be addition of check constraint for Person to verify if the person is author or publisher of the book who is commenting. This has to be prevented.
3. **Justify using a Relational DBMS like Oracle for this project.**
   We have used relational DBMS because a project like wonder share would be better maintained, access and updated on a regular basis if it is using RDBMS. This project had lot of constraints, stakeholders and scope of scaling up. Hence, doing it in RDBMS is the better solution instead of going for file system-based approach or any other data format.

EER Diagram

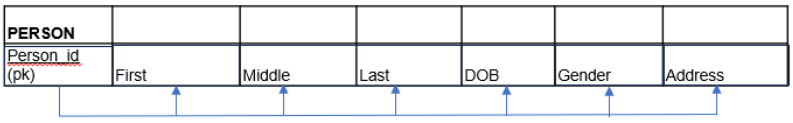
As the diagram is huge, we have embedded our diagram in a link which could be viewed.

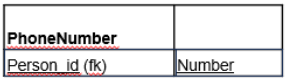
Click this link

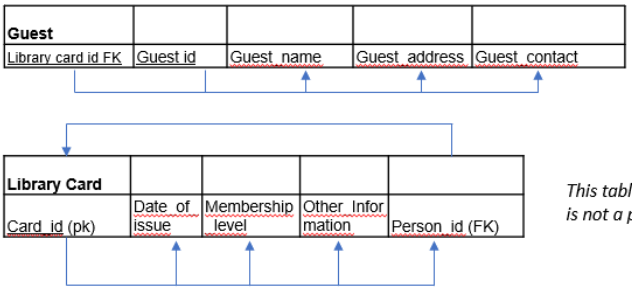
Relational Schema



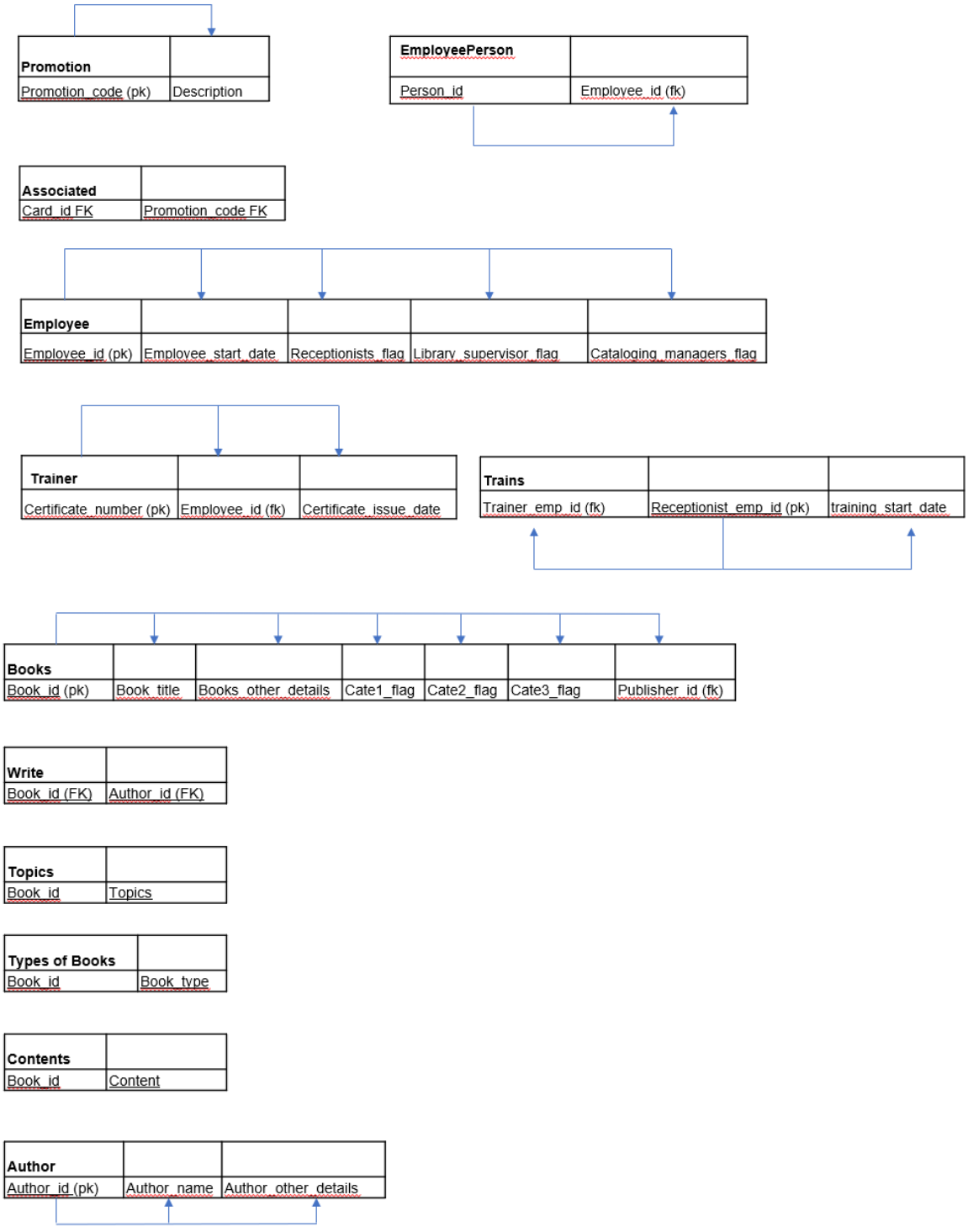
Dependency diagram







This table is in 3NF because Person_ID -> Card_ID is not a problem because both are CKs



Promotion	
Promotion_code (pk)	Description

EmployeePerson	
Person_id	Employee_id (fk)

Associated	
Card_id FK	Promotion_code FK

Employee				
Employee_id (pk)	Employee_start_date	Receptionists_flag	Library_supervisor_flag	Cataloging_managers_flag

Trainer		
Certificate_number (pk)	Employee_id (fk)	Certificate_issue_date

Trains		
Trainer_emp_id (fk)	Receptionist_emp_id (pk)	training_start_date

Books						
Book_id (pk)	Book_title	Books_other_details	Cate1_flag	Cate2_flag	Cate3_flag	Publisher_id (fk)

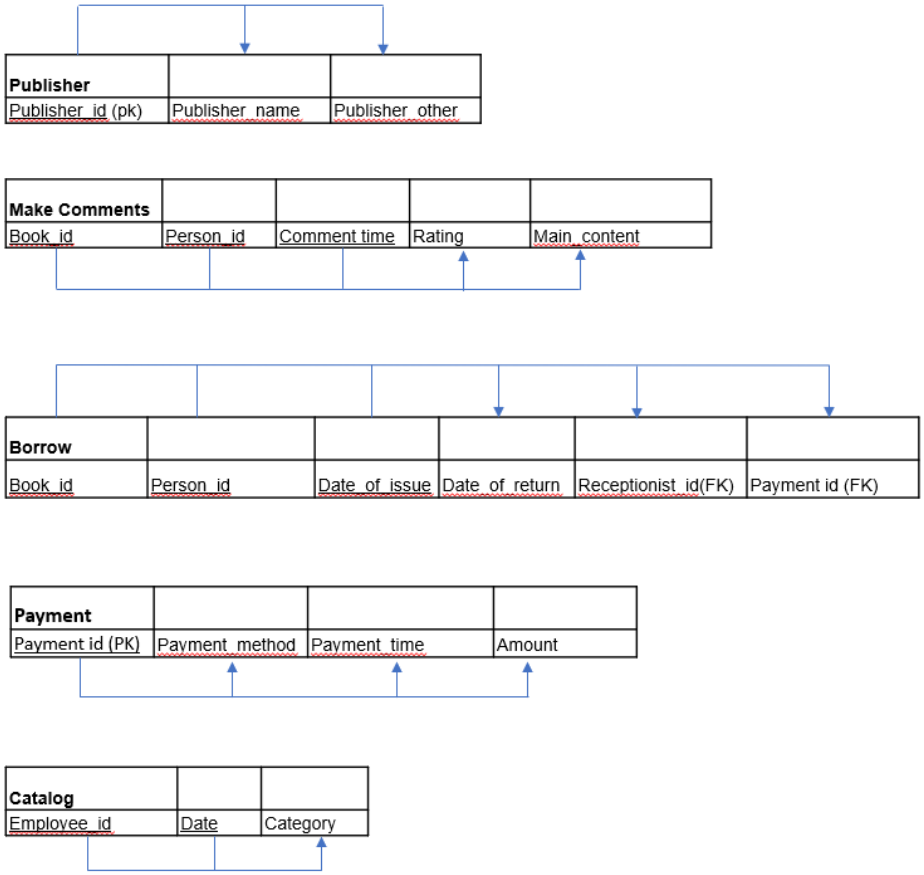
Write	
Book_id (FK)	Author_id (FK)

Topics	
Book_id	Topics

Types of Books	
Book_id	Book_type

Contents	
Book_id	Content

Author		
Author_id (pk)	Author_name	Author_other_details



All requested SQL statements

Create tables commands (Foreign keys and Check constraints where added via the UI)

```
CREATE DATABASE wonder_lib;

USE wonder_lib;

CREATE TABLE Person (
    Person_id INT PRIMARY KEY,
    First VARCHAR(255),
    Middle VARCHAR(255),
    Last VARCHAR(255),
    DOB DATE,
    Gender CHAR(1),
    Address VARCHAR(255)
);

CREATE TABLE PhoneNumber (
    Person_id INT,
    Number VARCHAR(20),
    PRIMARY KEY (Person_id, Number)
```

```
);


CREATE TABLE Guest (
    Library_card_ID INT,
    Guest_id INT,
    Guest_name VARCHAR(255),
    Guest_address VARCHAR(255),
    Guest_contact VARCHAR(20),
    PRIMARY KEY (Library_card_ID, Guest_id)
);

CREATE TABLE LibraryCard (
    Card_ID INT PRIMARY KEY,
    Date_of_issue DATE,
    Membership_level VARCHAR(255),
    Other_information TEXT,
    Person_id INT
);


CREATE TABLE Promotion (
    Promotion_code VARCHAR(255) PRIMARY KEY,
    Description VARCHAR(255)
);

CREATE TABLE Associated (
    Card_id INT,
    Promotion_code VARCHAR(255),
    PRIMARY KEY (Card_id, Promotion_code)
);
CREATE TABLE Employee (
    Employee_id INT PRIMARY KEY,
    Person_id INT,
    Employee_start_date DATE,
    Receptionists_flag BOOLEAN,
    Trainer_flag BOOLEAN,
    Certificate_number VARCHAR(255),
    Certificate_issue_date DATE,
    Library_supervisor_flag BOOLEAN,
    Cataloging_managers_flag BOOLEAN
);

CREATE TABLE Trains (
    Trainer_emp_id INT,
    Receptionists_emp_id INT,
    PRIMARY KEY (Receptionists_emp_id)
```

```
);

CREATE TABLE Catalog (
    Employee_id INT,
    Catalog_date DATE,
    Category VARCHAR(255),
    PRIMARY KEY (Employee_id, Catalog_date)
);

CREATE TABLE Books (
    Book_id INT PRIMARY KEY,
    Book_title VARCHAR(255),
    Book_other_details TEXT,
    Cate1_flag BOOLEAN,
    Cate2_flag BOOLEAN,
    Cate3_flag BOOLEAN,
    Publisher_id INT
);

-- BookAuthor is the Write table
CREATE TABLE BookAuthor (
    Book_id INT PRIMARY KEY,
    Author_id INT
);

CREATE TABLE Topics (
    Book_id INT,
    Topic VARCHAR(100),
    PRIMARY KEY (Book_id, Topic)
);

CREATE TABLE TypesOfBooks (
    Book_id INT,
    Book_type VARCHAR(50),
    PRIMARY KEY (Book_id, Book_type)
);

CREATE TABLE Contents (
    Book_id INT,
    Content VARCHAR(255),
    PRIMARY KEY (Book_id, Content)
);

CREATE TABLE Publish (
    Book_id INT,
    Publisher_id INT,
    PRIMARY KEY (Book_id)
```

```
);

CREATE TABLE Author (
   Author_id INT PRIMARY KEY,
   Author_name VARCHAR(255),
   Author_other_details VARCHAR(255)
);

CREATE TABLE Publisher (
   Publisher_id INT PRIMARY KEY,
   Publisher_name VARCHAR(255),
   Publisher_other_details VARCHAR(255)
);

CREATE TABLE MakeComments (
   Book_id INT,
   Person_id INT,
   Comment_time TIMESTAMP,
   Rating INT,
   Main_content VARCHAR(255),
   PRIMARY KEY (Book_id, Person_id, Comment_time)
);

drop table borrow;
CREATE TABLE Borrow (
   Book_id INT,
   Person_id INT,
   Date_of_issue DATE,
   Date_of_return DATE,
   Date_of_reception DATE,
   Payment_id INT,
   PRIMARY KEY (Book_id, Person_id)
);

CREATE TABLE Payment (
   Payment_id INT PRIMARY KEY,
   Payment_method VARCHAR(50),
   Payment_time TIMESTAMP,
   Amount DECIMAL(10, 2)
);
```

Create 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 TopGoldMember as
   select mem.person_id, mem.first first, mem.last last, l.date_of_issue enrollment_date from

```
        (select b.person_id, p.first first, p.last last
    from (select count(book_id) nbk, Person_id from (select * from borrow where Date_of_issue
BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 MONTH) AND CURDATE()) prevB
group by person_id having nbk > 5)
         b join person p
         on b.person_id = p.person_id)
mem join librarycard l
on mem.person_id = l.person_id;
```

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

```
create view popularbooks as
select books.* from
        (select book_id, count(1) used from
                (select book_id from borrow
                where Date_of_issue between DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
AND CURDATE()) PastYrBooks
        group by book_id
   order by used desc
   limit 1)
pop left join books
on pop.book_id = books.book_id;
```

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

```
create view BestRatingPublisher as
select distinct p.* from
        (select b.publisher_id from
                (select book_id, avg(rating) avg_rating from makecomments
                group by book_id
                having avg_rating >= 4)
        rating join books b
        on rating.book_id = b.book_id) -- highRatingPublisherIDs (HRP)
HRP left join publisher p
on HRP.publisher_id = p.publisher_id;
```

4. **PotentialGoldMember - This view returns the name, phone number and ID of the people who are not Gold member but borrowed books in every month in the past year.**

```
create view PotentialGoldMember as
select m.*, ph.number from
        (select members.person_id, p.first, p.middle, p.last from
                (select distinct b.person_id from
                        (select * from borrow
```

```
                    where Date_of_issue between DATE_SUB(CURDATE(), INTERVAL 1
YEAR) AND CURDATE()) b join librarycard l
                on b.Person_id = l.Person_id
                where l.Membership_level = 'Silver'
                group by b.person_id
                having COUNT(DISTINCT MONTH(b.Date_of_issue)) = 12) members
        join person p
        on p.person_id = members.person_id) m left join phonenumber ph
on ph.person_id = m.person_id;
```

5. **Fast Trainer – This view returns details of trainers who trains receptionist within 1 week after the trainer issued a certificate.**

```
create view fasttrainer as
select trnr.*, trn.training_start_date from trainer trnr join trains trn
on trn.Trainer_emp_id = trnr.Employee_id
where trn.training_start_date between trnr.certificate_issue_date AND
DATE_ADD(trnr.certificate_issue_date, INTERVAL 7 DAY);
```

Show the SQL statements of the following Queries

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

```
select * from employee where Library_supervisor_flag=1 and
Employee_start_date between date_sub(Curdate(), interval 2 month) and Curdate();
```

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

```
select distinct bb.first, bb.last, Books.book_title from
        (select b.person_id, b.first, b.last, brw.Book_id from
                (select e.person_id, p.first, p.last from
                        (select emp.*, lib.card_id from employee emp join librarycard lib
                        on emp.person_id = lib.person_id) e
                join person p
                on p.person_id=e.person_id) b
        join borrow brw
        on brw.person_id = b.person_id and
        brw.Date_of_issue between date_sub(Curdate(), interval 1 month) and Curdate()) bb
join books
on bb.book_id = books.book_id;
```

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

```
select avg(tgb.book_count) avg_book_borrowed_by_top_gold_members from
        (select tg.person_id, count(b.book_id) book_count from
        topgoldmember tg join borrow b
        on tg.person_id = b.Person_id
        group by tg.person_id) tgb;
```

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

```
select p.publisher_name, pb.book_title from
```

```
popularbooks pb join publisher p
on pb.publisher_id = p.Publisher_id;
```

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

```
select books.book_title
from books
left join borrow on books.Book_id = borrow.Book_id
        and borrow.Date_of_issue >= DATE_SUB(CURDATE(), interval 5 month)
where borrow.Book_id is null;
```

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

```
select distinct p.* from
        (select b.* from
        borrow b join popularbooks pb
        on b.book_id = pb.book_id) pbb
join person p
on p.Person_id = pbb.person_id;
```

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

```
select p.* from
        (select lib.card_id, lib.person_id from
                (select library_card_id, count(guest_id) as g
                from guest
                group by library_card_id
                having g = (select max(g_count) from
                (select library_card_id, count(guest_id) as g_count from guest group by
library_card_id) as max_guests)) gst
        join librarycard lib
        on lib.card_id = gst.library_card_id) lg
join person p
on p.person_id = lg.person_id;
```

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

```
select year(date_of_issue) as borrowyear, count(*) as numberofbooksborrowed
from borrow
group by borrowyear
order by numberofbooksborrowed desc
limit 1;
```

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

```
select distinct p.* from
        (select b.* from
        borrow b join popularbooks pb
```

```
        on b.book_id = pb.book_id) pbb
join person p
on p.Person_id = pbb.person_id;
```

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

```
select e.employee_id, e.employee_start_date, lc.date_of_issue
from employee e
join librarycard lc on e.person_id = lc.person_id
where lc.membership_level = 'gold'
and lc.date_of_issue between e.employee_start_date and date_add(e.employee_start_date,
interval 1 month);
```

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

```
select p.first, p.last
from person p
join librarycard lc on p.person_id = lc.person_id
where lc.membership_level = 'silver'
and datediff(curdate(), lc.date_of_issue) > 5 * 365; -- assuming a year has 365 days
```

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

```
select p.person_id, p.first, p.last, count(1) borrowed_last_year from potentialgoldmember p join
        (select * from borrow
        where Date_of_issue between date_sub(curdate(), interval 1 year) and curdate())
pastYrBorrow
on pastYrBorrow.person_id = p.person_id
group by p.person_id, p.first, p.last;
```

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

```
select Trainer_emp_id, count(1) Number_of_receptionists from trains
        group by Trainer_emp_id
having Number_of_receptionists = (select max(Number_of_receptionists) from
(select Trainer_emp_id, count(1) Number_of_receptionists from trains
        group by Trainer_emp_id) t);
```

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

```
select employee_id
from catalog
where catalog_date >= current_date - interval 4 week
group by employee_id
having count(distinct category) = 3
  and count(distinct catalog_date) = 4;
```