Part a)
Original Form:

Person

Person_ID	Name	Address	Gender	Birthday	Card_ID	EMP_Start_Date	EMP_Position

Phone Number

Person_ID	number	phoneType

Member

Card_ID	Person_ID	isSilver

Guest

Member_Card_ID	Guest_ID	Guest_Name	Guest_Address	Guest_Contact_Information

Library Card

Card_ID	Date of issue

Publisher

Publisher_ID	Publisher_Name

Author

Author_ID	Author_Name

Writes

Book_ID	Author_ID

Borrows

Book_ID	Receptionist_ID	Card_ID	Issue_Date	Return_Date	Due_Date	Payment_owed

Book

Book_ID	Publisher_ID	Title	Class_no

Third Normal Form:

Book

Book_ID	Publisher_ID	Title	Class_no

Borrows

Book_ID	Receptionist_ID	Card_ID	Issue_Date	Return_Date	Due_Date	Payment_owed

Writes

Book_ID	Author_ID

Author

Author_ID	Author_Name

Publisher

Publisher_ID	Publisher_name

Guest

Member_Card_ID	Guest_ID	Guest_Name	Guest_Address	Guest_Contact_Information

Library Card/Member
Combine library card/member since both depend only on Card_ID

Card_ID	Date_of_issue	isSilver

Phone Number

Person_ID	Number	phoneType

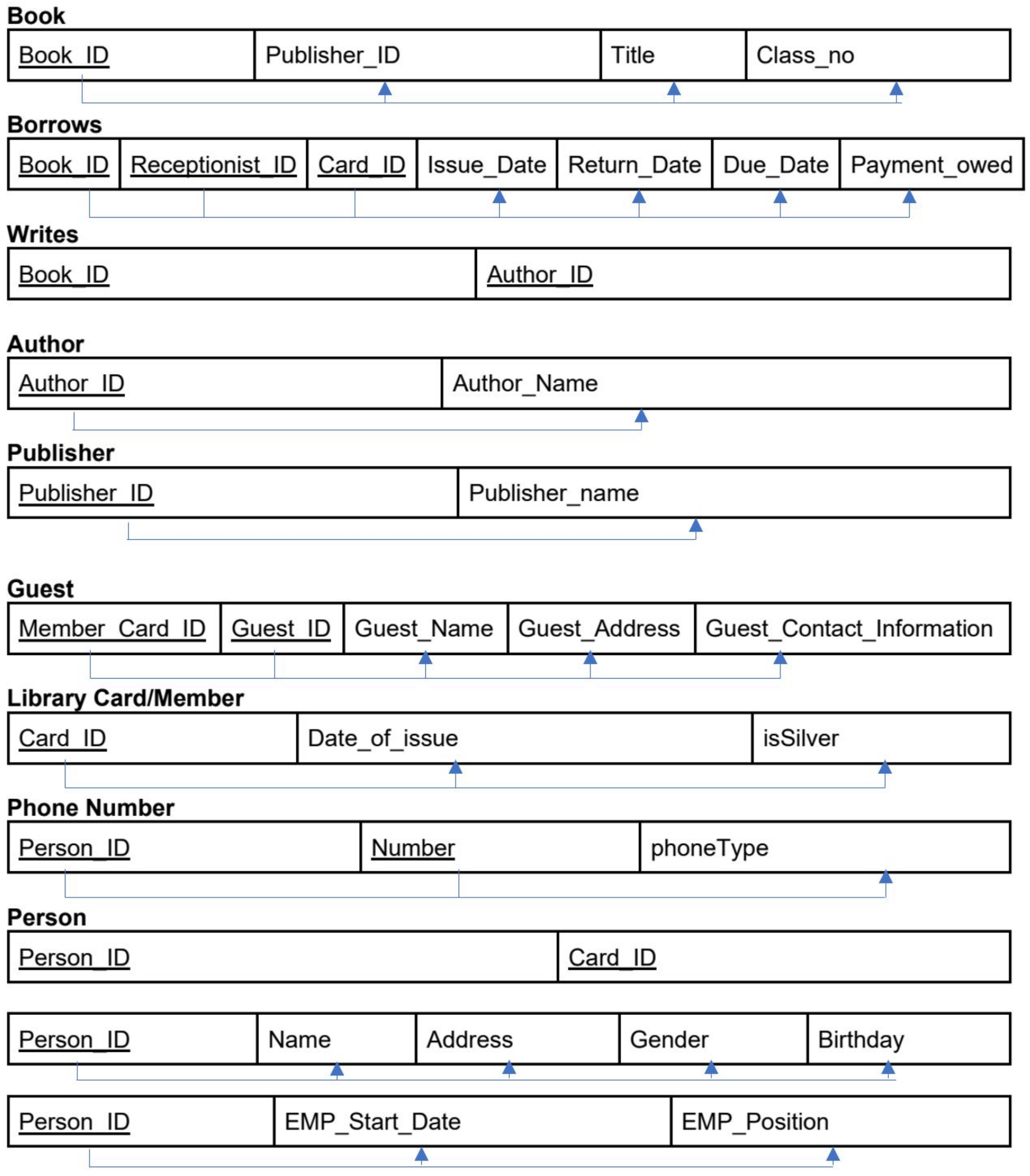
Person
Need to divide Person into 3 tables because Name, Address, Gender, and Birthday are functionally dependent on Person_ID and EMP_Start_Date and EMP_Position are functionally dependent on Person ID but don't need to be together with Name, Address, Gender, and Birthday because they're only valid when the person is an employee.

Person_ID	Card_ID

Person_ID	Name	Address	Gender	Birthday

Person_ID	EMP_Start_Date	EMP_Position

Part b)
Dependency Diagram:



C. SQL Statements

Note: Some of the column names have changed because they conflict with keywords. (Example: NUMBER column in PHONE relation). Others changed for consistency.

Note 2: The order matters. If a foreign key reference has not been defined, then the table creation will fail. Alternative is to create a table without reference and then use an alter table at the end.

Note 3: Oracle DB 18c.

```
alter session applies to queries with it. It does not persist.

create database “CS4347-PROJECT”

alter session set NLS_DATE_FORMAT = 'MM/DD/YYYY' /* allows dates to
insert as MM/DD/YYYY */;

/* PERSON AND DERIVATIVES */

create table PERSON (
  PERSON_ID         VARCHAR(4)      PRIMARY KEY,
  NAME              VARCHAR(100)    NOT NULL,
  ADDRESS           VARCHAR(100)    NOT NULL,
  GENDER            VARCHAR(25)     NOT NULL,
  BIRTHDAY          DATE            NOT NULL CHECK (EXTRACT(YEAR FROM
BIRTHDAY) < 2004)
);

create table PHONE (
  PERSON_ID         VARCHAR(4)      NOT NULL,
  PHONE_NUMBER      VARCHAR(35)     PRIMARY KEY /* store number
unformatted, i.e. 5556667789 */,
  PHONE_TYPE        VARCHAR(10)     NOT NULL,
  FOREIGN KEY (PERSON_ID) REFERENCES PERSON(PERSON_ID)
);

create table EMPLOYEE (
  PERSON_ID         VARCHAR(4)      PRIMARY KEY,
  EMP_START_DATE    DATE            NOT NULL,
  EMP_POSITION      VARCHAR(100)    NOT NULL,
  FOREIGN KEY (PERSON_ID) REFERENCES PERSON(PERSON_ID)
);

create table MEMBER (
  PERSON_ID         VARCHAR(4)          NOT NULL,
  CARD_ID           NUMBER              GENERATED BY DEFAULT ON NULL
AS IDENTITY /* STARTS incrementing ID at 1 */ PRIMARY KEY,
  ISSUE_DATE        DATE                NOT NULL,
  IS_SILVER         CHAR(1) DEFAULT 0   NOT NULL,
  FOREIGN KEY (PERSON_ID) REFERENCES PERSON(PERSON_ID)
);

/* GUEST ID cannot be PK or UNIQUE */
create table GUEST (
  MEMBER_CARD_ID    NUMBER          NOT NULL,
  GUEST_ID          NUMBER          GENERATED BY DEFAULT ON NULL AS
IDENTITY /* STARTS incrementing ID at 1 */,
  GUEST_NAME        VARCHAR(100)    NOT NULL,
  GUEST_ADDRESS     VARCHAR(100)    NOT NULL,
  GUEST_CONTACT     VARCHAR(35)     NOT NULL /* what is the data
stored here? */,
  FOREIGN KEY (MEMBER_CARD_ID) REFERENCES MEMBER(CARD_ID)
);

/* PUBLISHER AND DERIVATIVES */

create table PUBLISHER (
  PUBLISHER_ID      NUMBER          GENERATED BY DEFAULT ON NULL AS
IDENTITY /* STARTS incrementing ID at 1 */ PRIMARY KEY,
  PUBLISHER_NAME    VARCHAR(100)    NOT NULL
);

create table AUTHOR (
  AUTHOR_ID      NUMBER          GENERATED BY DEFAULT ON NULL AS
IDENTITY /* STARTS incrementing ID at 1 */ PRIMARY KEY,
  AUTHOR_NAME    VARCHAR(100)    NOT NULL
);

create table BOOK (
  BOOK_ID       VARCHAR(4)           PRIMARY KEY,
  PUBLISHER_ID  NUMBER               NOT NULL,
  TITLE         VARCHAR(100)         NOT NULL,
  CLASS_NO      CHAR(1) DEFAULT 1    NOT NULL, /* 1 or 2 */
  FOREIGN KEY (PUBLISHER_ID) REFERENCES PUBLISHER(PUBLISHER_ID)
);

create table BORROWS (
  BOOK_ID           VARCHAR(4)  NOT NULL,
  RECEPTIONIST_ID   NUMBER      NOT NULL,
  CARD_ID           NUMBER      NOT NULL,
  ISSUE_DATE        DATE        NOT NULL,
  RETURN_DATE       DATE        /* can be null if book not yet
returned */,
  DUE_DATE          DATE        NOT NULL,
  PAYMENT_OWED      NUMBER      /* can be null if book not late */,
  FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID),
  FOREIGN KEY (CARD_ID) REFERENCES MEMBER(CARD_ID),
  CONSTRAINT PK_BORROWS PRIMARY KEY(BOOK_ID, RECEPTIONIST_ID, CARD_ID)
);

create table WRITES (
  BOOK_ID       VARCHAR(4)  NOT NULL,
  AUTHOR_ID     NUMBER      NOT NULL,
  FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID),
  FOREIGN KEY (AUTHOR_ID) REFERENCES AUTHOR(AUTHOR_ID),
  CONSTRAINT PK_WRITES PRIMARY KEY(BOOK_ID, AUTHOR_ID)
);
```

Before insertion set the date format.

```
alter session set NLS_DATE_FORMAT = 'MM/DD/YYYY' /* allows dates to
insert as MM/DD/YYYY */;
```

Sample insertion statement into PERSON.

```
insert into PERSON values (‘P001’, 'John Doe', '123 Apple Street',
'Male', '04/21/1958');
```

Make John a GOLD member.

```
insert into MEMBER values ((select PERSON_ID from PERSON where NAME =
'John Doe'), NULL, SYSDATE, 1);
```

SYSDATE is the current date/time.

We could also use triggers to automatically populate the member table on the insertion of a new person.

Because of the GENERATED by DEFAULT the NULL turns into an unique id 1,2,3,...,n.

See the shared drive folder for layout and structures creation .sql file. Link provided for quick access. These files are with the documents. You may have to download the svg to see it clearly.
https://drive.google.com/open?id=17qnovXolAY0xlLQUhGI6oScjuDIhhowg

D. Data Dictionary

https://docs.google.com/document/d/1uDBsLDzAWkZ9ScY45dnLfs5qkaV_hcvrUPXjq8fy8YE/edit

PERSON			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
PERSON_ID	VARCHAR(4)	ID for a PERSON	PRIMARY KEY
NAME	VARCHAR(100)	PERSON full name	NOT NULL
ADDRESS	VARCHAR(100)	PERSON address	NOT NULL
GENDER	VARCHAR(25)	PERSON gender	NOT NULL
BIRTHDAY	DATE	PERSON birthdate Ex: 01/01/1995	CHECK AGE_YEAR < 2004 NOT NULL

PHONE			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
PERSON_ID	VARCHAR(4)	ID for a PERSON	FOREIGN KEY REF: PERSON NOT NULL
PHONE_NUMBER	VARCHAR(35)	Unformatted phone number Ex: (5556667789)	PRIMARY KEY
PHONE_TYPE	VARCHAR(10)	Category Ex: Home, Work	NOT NULL

EMPLOYEE

COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
PERSON_ID	VARCHAR(4)	ID for a PERSON	PRIMARY KEY FOREIGN KEY REF: PERSON
EMP_START_DATE	DATE	Start date for an EMPLOYEE	NOT NULL
EMP_POSITION	VARCHAR(100)	Current title Ex: Advisor	NOT NULL

MEMBER			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
PERSON_ID	VARCHAR(4)	ID for a PERSON	FOREIGN KEY REF: PERSON NOT NULL
CARD_ID	NUMBER	ID for a MEMBER’s card	PRIMARY KEY
ISSUE_DATE	DATE	Issue date for a MEMBER card	NOT NULL
IS_SILVER	CHAR(1)	DEFAULT 0 0 = FALSE 1 = TRUE	NOT NULL

GUEST			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
MEMBER_CARD_ID	NUMBER	ID for a MEMBER’s card	FOREIGN KEY REF: MEMBER COL: CARD_ID NOT NULL
GUEST_ID	NUMBER	ID for a GUEST	NOT NULL
GUEST_NAME	VARCHAR(100)	GUEST full name	NOT NULL
GUEST_ADDRESS	VARCHAR(100)	GUEST address	NOT NULL
GUEST_CONTACT	VARCHAR(35)	GUEST contact	NOT NULL

		Ex: Phone	

PUBLISHER			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
PUBLISHER_ID	NUMBER	ID for a PUBLISHER	PRIMARY KEY
PUBLISHER_NAME	VARCHAR(100)	PUBLISHER full name	NOT NULL

AUTHOR			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
AUTHOR_ID	NUMBER	ID for a AUTHOR	PRIMARY KEY
AUTHOR_NAME	VARCHAR(100)	AUTHOR full name	NOT NULL

BOOK			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
BOOK_ID	VARCHAR(4)	ID for a BOOK	PRIMARY KEY
PUBLISHER_ID	NUMBER	ID for a PUBLISHER	FOREIGN KEY REF: PUBLISHER NOT NULL
TITLE	VARCHAR(100)	Book Title Ex: Eragon	NOT NULL
CLASS_NO	VARCHAR(1)	DEFAULT 1 1 = Class 1 2 = Class 2	NOT NULL

BORROWS			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
BOOK_ID	VARCHAR(4)	ID for a BOOK	FOREIGN KEY REF: BOOK

			NOT NULL
RECEPTIONIST_ID	NUMBER	ID for a RECEPTIONIST	NOT NULL
CARD_ID	NUMBER	ID for a MEMBER's card	FOREIGN KEY REF: MEMBER NOT NULL
ISSUE_DATE	DATE	Issue date for a borrowed book	NOT NULL
RETURN_DATE	DATE	Return date for a borrowed book NULL = BORROWED	
DUE_DATE	DATE	Due date for a borrowed book	NOT NULL
PAYMENT_OWED	NUMBER	Payment owed for a late book	
PK_BORROWS			CONSTRAINT PRIMARY KEY(BOOK_ID, RECEPTIONIST_ID, CARD_ID)

WRITES			
COLUMN	TYPE	DESCRIPTION	ATTRIBUTE
BOOK_ID	VARCHAR(4)	ID for a BOOK	FOREIGN KEY REF: BOOK NOT NULL
AUTHOR_ID	NUMBER	ID for a AUTHOR	FOREIGN KEY REF: AUTHOR NOT NULL
PK_WRITES			CONSTRAINT PRIMARY KEY(BOOK_ID, AUTHOR_ID)

```
Example for query number 4
The most popular book is the COUNT of a given BOOK_ID in BORROWS
Use that to find the publisher.
```

E. Views

```
1)
CREATE VIEW TOP_GOLD_MEMBERS
SELECT NAME, M.CARD_ID, M.ISSUE_DATE
FROM PERSON
   JOIN MEMBER M on PERSON.PERSON_ID = M.PERSON_ID
   JOIN BORROWS B on M.CARD_ID = B.CARD_ID
WHERE IS_SILVER = 0 AND B.ISSUE_DATE >= sysdate - 365 AND
     (SELECT COUNT(*)
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND ((B.ISSUE_DATE -
BORROWS.ISSUE_DATE) <= 7) AND ((B.ISSUE_DATE - BORROWS.ISSUE_DATE) >= -7)) >=
5
GROUP BY NAME, M.ISSUE_DATE;


2)
CREATE VIEW POPULAR_BOOKS
AS SELECT COUNT(BORROWS.BOOK_ID) as TIMES_CHECKED_OUT, BORROWS.BOOK_ID,
BOOK.TITLE AS BOOK_TITLE, AUTHOR.AUTHOR_NAME, PUBLISHER.PUBLISHER_NAME
FROM BORROWS
   INNER JOIN WRITES ON BORROWS.BOOK_ID = WRITES.BOOK_ID
   INNER JOIN AUTHOR ON WRITES.AUTHOR_ID = AUTHOR.AUTHOR_ID
   INNER JOIN BOOK ON BOOK.BOOK_ID = BORROWS.BOOK_ID
   INNER JOIN PUBLISHER ON BOOK.PUBLISHER_ID = PUBLISHER.PUBLISHER_ID
GROUP BY BORROWS.BOOK_ID, BOOK.TITLE, AUTHOR.AUTHOR_NAME,
PUBLISHER.PUBLISHER_NAME
ORDER BY TIMES_CHECKED_OUT DESC
FETCH NEXT 3 ROWS ONLY;

3)
CREATE VIEW TOP_LATE_PAYMENT_MEMBERS
AS SELECT SUM(NVL(PAYMENT_OWED, 0)) AS TOTAL_FEES, P.NAME, P.ADDRESS,
P2.PHONE_NUMBER, M.CARD_ID
FROM BORROWS
   JOIN MEMBER M on BORROWS.CARD_ID = M.CARD_ID
   JOIN PERSON P on M.PERSON_ID = P.PERSON_ID
   JOIN PHONE P2 on P.PERSON_ID = P2.PERSON_ID
GROUP BY NAME, ADDRESS, PHONE_NUMBER, M.CARD_ID
ORDER BY TOTAL_FEES DESC
```

```
FETCH NEXT 3 ROWS ONLY;

4)
CREATE VIEW POTENTIAL_GOLD_MEMBERS
AS SELECT Name, PHONE_NUMBER, M.CARD_ID
FROM PERSON
   JOIN MEMBER M on PERSON.PERSON_ID = M.PERSON_ID
   JOIN PHONE P on PERSON.PERSON_ID = P.PERSON_ID
WHERE IS_SILVER = 1 AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -1) AND BORROWS.ISSUE_DATE <= sysdate AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -2) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -1) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -3) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -2) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -4) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -3) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -5) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -4) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -6) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -5) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
            WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -7) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -6) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
            FROM BORROWS
```

```
             WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -8) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -7) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
             FROM BORROWS
             WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -9) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -8) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
             FROM BORROWS
             WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -10) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -9) AND
BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
             FROM BORROWS
             WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -11) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -10)
AND BORROWS.RETURN_DATE <= BORROWS.DUE_DATE) AND
     EXISTS(SELECT BORROWS.CARD_ID
             FROM BORROWS
             WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
add_months(sysdate, -12) AND BORROWS.ISSUE_DATE <= add_months(sysdate, -11)
AND BORROWS.RETURN_DATE <= BORROWS.DUE_DATE)
GROUP BY Name, PHONE_NUMBER, M.CARD_ID;


CREATE VIEW PotentialGoldMember AS SELECT M.CARD_ID
FROM MEMBER M, BORROWS B
WHERE M.IS_SILVER=TRUE AND M.CARD_ID = B.CARD_ID AND NOT EXISTS (
SELECT 1 AS MONTH UNION
SELECT 2 AS MONTH UNION
SELECT 3 AS MONTH UNION
SELECT 4 AS MONTH UNION /* Could also just make a table called MONTH */
SELECT 5 AS MONTH UNION
SELECT 6 AS MONTH UNION
SELECT 7 AS MONTH UNION
SELECT 8 AS MONTH UNION
SELECT 9 AS MONTH UNION
SELECT 10 AS MONTH UNION
SELECT 11 AS MONTH UNION
SELECT 12 AS MONTH
EXCEPT
SELECT MONTH(ISSUE_DATE)
FROM BORROWS
WHERE M.CARD_ID = BORROWS.CARD_ID AND BORROWS.ISSUE_DATE >=
DATE_ADD(DATE(SYSDATE()), INTERVAL -365 DAY)
GROUP BY MONTH(ISSUE_DATE)
)
```

```
GROUP BY B.CARD_ID
HAVING SUM(PAYMENT_OWED) <= 0;
```

5)

```
CREATE VIEW POPULAR_AUTHORS
AS SELECT COUNT(*) AS BOOKS_CHECKED_OUT, AUTHOR_NAME
FROM BORROWS
   JOIN WRITES W on BORROWS.BOOK_ID = W.BOOK_ID
   JOIN AUTHOR A2 on W.AUTHOR_ID = A2.AUTHOR_ID
GROUP BY AUTHOR_NAME
ORDER BY BOOKS_CHECKED_OUT DESC
FETCH NEXT 5 ROWS ONLY;
```

F. Queries

1)

```
SELECT
  PERSON.*
FROM
  PERSON, EMPLOYEE
WHERE
     PERSON.person_id = EMPLOYEE.person_id
  AND UPPER(EMPLOYEE.EMP_POSITION) = UPPER('supervisor')
  AND EMPLOYEE.EMP_START_DATE >= add_months(sysdate, -2);
```

2)

```
SELECT PERSON.NAME, B2.TITLE
FROM PERSON
   JOIN EMPLOYEE E on PERSON.PERSON_ID = E.PERSON_ID
   JOIN MEMBER M on PERSON.PERSON_ID = M.PERSON_ID
   JOIN BORROWS B on M.CARD_ID = B.CARD_ID
   JOIN BOOK B2 on B.BOOK_ID = B2.BOOK_ID
WHERE B.ISSUE_DATE >= add_months(sysdate, -1);
```

3)
```
SELECT COUNT(*) / 5 AS AVERAGE
FROM BORROWS
WHERE CARD_ID = (SELECT TOP_GOLD_MEMBERS.CARD_ID
            FROM TOP_GOLD_MEMBERS_2
            WHERE ROWNUM = 1) OR
    CARD_ID = (SELECT TOP_GOLD_MEMBERS.CARD_ID
            FROM TOP_GOLD_MEMBERS_2
            WHERE ROWNUM = 2) OR
    CARD_ID = (SELECT TOP_GOLD_MEMBERS.CARD_ID
            FROM TOP_GOLD_MEMBERS
            WHERE ROWNUM = 3) OR
    CARD_ID = (SELECT TOP_GOLD_MEMBERS.CARD_ID
            FROM TOP_GOLD_MEMBERS_2
            WHERE ROWNUM = 4) OR
    CARD_ID = (SELECT TOP_GOLD_MEMBERS.CARD_ID
            FROM TOP_GOLD_MEMBERS_2
            WHERE ROWNUM = 5);
```

4)
```
SELECT
          publisher_name, book_title
FROM
          (
          SELECT
          *
          FROM
          popular_books
          ORDER BY
          times_checked_out desc
          )
WHERE
          ROWNUM <= 1;
```

5)
```
SELECT TITLE
FROM BOOK
  JOIN BORROWS B on BOOK.BOOK_ID = B.BOOK_ID
WHERE ISSUE_DATE <= add_months(sysdate, -5) AND
    NOT EXISTS (SELECT B.BOOK_ID
              FROM BORROWS
              WHERE B.BOOK_ID = BORROWS.BOOK_ID AND BORROWS.ISSUE_DATE >
add_months(sysdate, -5));
```

6)
```
SELECT * FROM MEMBER, PERSON WHERE MEMBER.PERSON_ID =
PERSON.PERSON_ID AND NOT EXISTS(
SELECT BOOK_ID FROM BOOK WHERE AUTHOR_ID=(SELECT AUTHOR_ID FROM
PopularAuthor LIMIT 1)
EXCEPT
SELECT BOOK_ID FROM BORROWS WHERE CARD_ID=MEMBER.CARD_ID);
```

7)
```
SELECT name -- this is just name (not person.name since it references the subquery).
FROM (
SELECT COUNT(person.name) as guest_number, person.name
FROM person
INNER JOIN member on person.person_id = member.person_id
INNER JOIN guest on member.card_id = guest.member_card_id
WHERE is_silver = 0 --checks for gold member
group by person.name
order by guest_number desc)
fetch next 1 row only;
```

8)
```
SELECT
        COUNT(*)                                AS books_borrowed,
        EXTRACT(YEAR FROM borrows.issue_date)         AS year
FROM
        borrows
GROUP BY
        EXTRACT(YEAR FROM borrows.issue_date)
ORDER BY
        books_borrowed DESC
FETCH FIRST 1 ROWS ONLY;
```

9)
```
SELECT DISTINCT
        person.name
FROM
        person,
        popular_books,
        borrows,
        member
WHERE
        borrows.card_id = member.card_id
        AND popular_books.book_id = borrows.book_id
        AND person.person_id = member.person_id
        AND popular_books.times_checked_out = (
        SELECT
        MAX(popular_books.times_checked_out)
        FROM
        popular_books
        GROUP BY
        popular_books.times_checked_out
        fetch first row only
        );
```

10)

```
SELECT
   BOOK.BOOK_ID, BOOK.TITLE
FROM
   BORROWS,
   BOOK
WHERE
   BORROWS.BOOK_ID = BOOK.BOOK_ID
      AND ISSUE_DATE >= (SELECT
         EMP_START_DATE
      FROM
         EMPLOYEE
      ORDER BY EMP_START_DATE DESC
      LIMIT 1)
GROUP BY BOOK_ID;
```

11)

```
select person.name, issue_date, emp_start_date
from person
inner join employee on person.person_id = employee.person_id
inner join member on person.person_id = member.person_id
where member.is_silver = 0
and member.issue_date between employee.emp_start_date and
add_months(employee.emp_start_date, 1);
```

12)

```
SELECT
        SUM(nvl(borrows.payment_owed, 0))           AS total_fees,
        EXTRACT(MONTH FROM borrows.return_date)     AS month
FROM
```

```
	borrows
WHERE
	borrows.return_date IS NOT NULL
	AND borrows.return_date >= add_months(sysdate, - 12)
	AND ( ( borrows.return_date <= ( sysdate - EXTRACT(DAY FROM sysdate) )
	AND borrows.return_date > add_months(sysdate - EXTRACT(DAY FROM sysdate), - 1)
)
	OR ( borrows.return_date <= add_months(sysdate - EXTRACT(DAY FROM sysdate), -
1)
	AND borrows.return_date > add_months(sysdate - EXTRACT(DAY FROM sysdate), - 2)
)
	OR ( borrows.return_date <= add_months(sysdate - EXTRACT(DAY FROM sysdate), -
2)
	AND borrows.return_date > add_months(sysdate - EXTRACT(DAY FROM sysdate), - 3)
) )
GROUP BY
	EXTRACT(MONTH FROM borrows.return_date)
ORDER BY
	month DESC;
```

13)

```
SELECT NAME
FROM PERSON
  JOIN MEMBER M on PERSON.PERSON_ID = M.PERSON_ID
WHERE IS_SILVER = 1 AND ISSUE_DATE < add_months(sysdate, 60);
```

14.

```
select potential_gold_members.name, amt_borrowed
from potential_gold_members, (
 select max(count(borrows.issue_date) ) as amt_borrowed
 from borrows
 where issue_date >= add_months(sysdate, -12)
 group by borrows.card_id), member
where potential_gold_members.card_id = member.card_id;
```