Q. P code No. 79484

Q.1 a) 1. Atomicity: This property states that a transaction must be treated as an atomic unit, that is, either all its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
2. Consistency: The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
3. Durability:The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
4. Isolation:In a database system where more than one transaction is being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

b) Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.
There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.
Types of Attributes

(a) **Simple attribute** – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
(b) **Composite attribute** – Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
(c) **Derived attribute** – Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
(d) **Single-value attribute** – Single-value attributes contain single value. For example – Social_Security_Number.
(e) **Multi-value attribute** – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

c) An entity set that does not have a primary key is referred to as a weak entity set.
The existence of a weak entity set depends on the existence of a identifying entity set:

- It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set.
- Identifying relationship depicted using a double diamond.

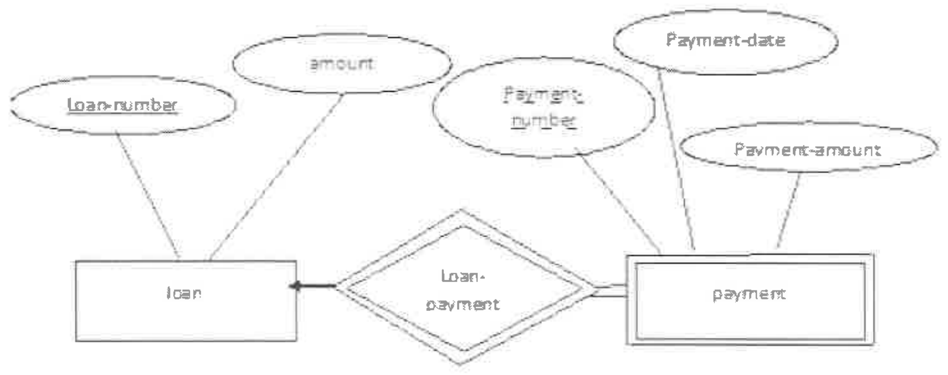
The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

We depict a weak entity set by double rectangles. We underline the discriminator of a weak entity set with a dashed line.

Example:

- payment-number – discriminator of the payment entity set
- Primary key for payment – (loan-number, payment-number).

The primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
If loan-number were explicitly stored, payment could be made a strong entity, but then the relationship between payment and loan would be duplicated by an implicit relationship defined by the attribute loan-number common to payment and loan.
Example of weak entity set:

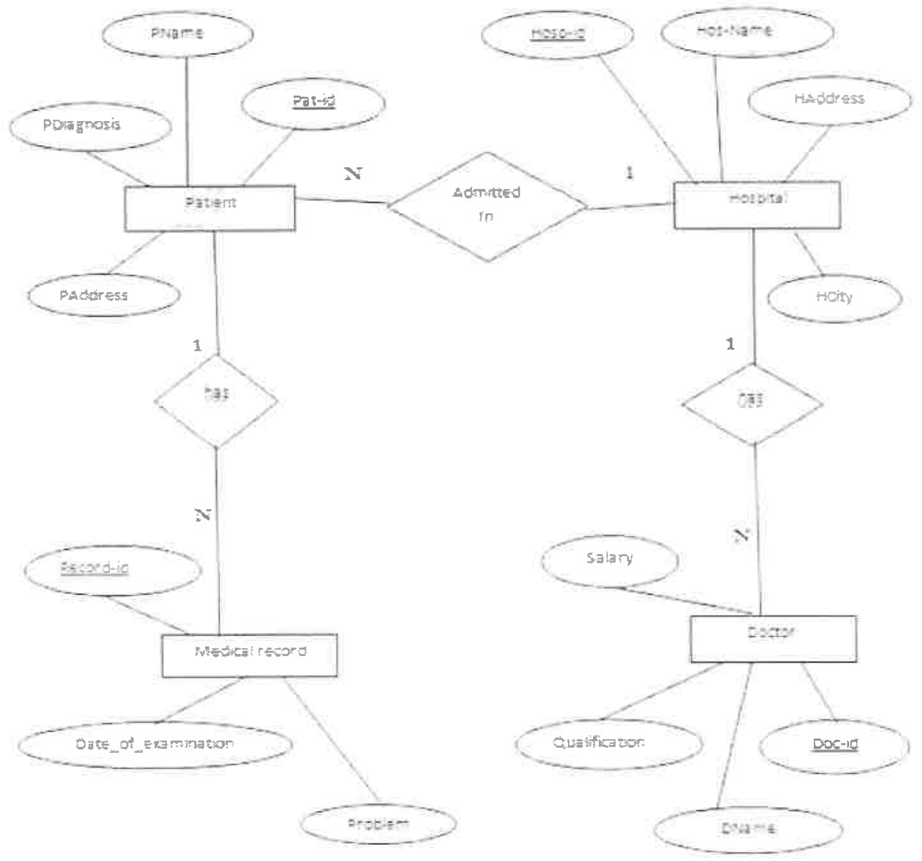


d) Example :
Given Student Report Database, in which student marks assessment is recorded. In such schema, create a trigger so that the total and average of specified marks is automatically inserted whenever a record is insert.

```
create trigger stud_marks
before INSERT
on
Student
for each row
set new.total = new.subj1 + new.subj2 + new.subj3, new.per = new.total * 100/300;
```

e) AVG, COUNT, MIN ,MAX, SUM

Q.5a) E-R Diagram



Converting the E-R Diagram into Tables
Converting entity to table and attribute to columns
Hospital

Hosp-id	Primary Key
HCity	
HAddress	
Hos-Name	
Pat-id	Foreign key references to Pat-id of Patient table
Doc-id	Foreign key references to Doc-id of Doctor table

b)

```
(a)  selectname
frommemberm,book b, borrowed br
where
         m.member_no = br.member_noand br..isbn = b.isbnandb.publisher='McGrawHill';

(b) select distinct m.name
from
member m
where not exists
((
Selectisbn
frombooks
where
publisher='McGrawHill')
except
(selectisbn
from borrowed b
Whereb.member_no = m.member_no));

(c) selectpublisher, name
from
(selectpublisher,name,count(isbn)
from
member m,books b, borrowed br
where
m.member_no = br.member_no andbr.isbn = b.isbn
group bypublisher, name)
as
emppub(publisher, name, count books)
where
count books>5
```

Q.6a)

```
(a)  selectcustomer_name from depositor
     wherecustomer_namenot in
     (selectcustomer_name from borrower);

(b)  select distinctbranch_name
     fromaccountnatural joindepositornatural joincustomer
wherecustomercity= 'Harrison';

(c) delete from account
where balance < (select avg(balance)
from account;
```

```
(d)select sum(amount)
from loan;

(e ) selectbranch-name,avg(balance)
fromaccount
group by
branch-name
having avg(balance) >$1200;
```