Q 1

**A):** SELECT title,released FROM Titles WHERE numberVotes > 10000;

**b):** SELECT title,averageRating FROM Titles WHERE averageRating

>AVG(averageRating);

**c):**

SELECT name\_id, name, birthYear, deathYear, profession1,

profession2, profession3, knownForTitle1, knownFortitle2

FROM Names WHERE deathYear=null;

**d):** SELECT title FROM Titles WHERE title

LIKE ‘%The%’;

**e):** SELECT released,COUNT(\*)

FROM Titles GROUP BY released;

Q 2

*a):*

Titles table

It is not in first normal form because of gener2,3,4

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any **composite** or **multi-valued attribute**. A relation is in first normal form if every attribute in that relation is singled valued attribute.

Names table

It is not in first normal form because of profession1,2,3

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any **composite** or **multi-valued attribute**. A relation is in first normal form if every attribute in that relation is singled valued attribute.

**b):**

* There should not be repeating values present in the table.​ Repeating values consumes a lot of extra memory and makes the search and update slow
* Working with the tables having the columns with multiple values separated by ‘,’ very difficult while searching for a specific value in the database by splitting all the values of a column and again
* It will cause of insertion,deletion,updation anomalies

**c):** Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity**during an update/delete/insert into a table we have different types of constraints primary key,foreign key,check,not null,default …

The advantage of constraint is data integrity.

If we create separate table for general,professions and store the data of three colums in one table and use prof\_id ,gen\_id as foreign key in it’s own table it will provide the following advantages

It will avoid repeating values and data duplicates

It will avoid insertion,deleting,updating anomalies

pk

|  |  |
| --- | --- |
|  |  |

k

|  |  |
| --- | --- |
| Gen\_id | Genre |
|  |  |
|  |  |

The above table store the data of genre1,2,3 columns used gen\_id as foreign key in the title table

FK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Title\_id | Title | Released | Runtime | averageRating | NumberVotes | Gen\_id |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Profession\_id | Profession |
|  |  |

The above table store the data of profession1,2,3 columns used profession\_id as foreign key in the names table

FK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name\_id | Name | birthYear | deathYear | Profession\_id | knownforTitle | knownforTitle |
|  |  |  |  |  |  |  |

**d):** UPDATE titles set averageRating=new\_rating where title\_id=title\_UPDATE;

**e):**

averageRating is 9.9 is the largest value

numberVotes is 9999999999 largest value

birthYear is 9999 largest value

**f):**

the disadvantes of the previous data supplicates and insertion,deletion,updation anomalies

the new design will be like this has the following advantages

It will avoid repeating values and data duplicates

It will avoid insertion,deleting,updating anomalies

Pk

|  |  |
| --- | --- |
| Gen\_id | Genre |
|  |  |
|  |  |

Pk

FK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Title\_id | Title | Released | Runtime | averageRating | NumberVotes | Gen\_id |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Pk

|  |  |
| --- | --- |
| Profession\_id | Profession |
|  |  |

FK

Pk

FK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name\_id | Name | birthYear | deathYear | Profession\_id | Title\_id |
|  |  |  |  |  |  |

**e):**

create table genre(gen\_id number,genr varchar2(50));

alter table genre add constraint genre\_genidpk primary key(gen\_id);

create table professions(profession\_id number,profession varchar2(50));

alter table professions add constraint professions\_profidpk primary key(profession\_id);

create table names(name\_id number,name varchar2(50),birthYear varchar2(4),profession\_id number,knownfortitle varchar2(9),knownfortitle varchar2(9));

alter table names add constraint names\_nameidpk primary key(name\_id);

alter table names add constraint names\_professionidfk foreign key(profession\_id) references professions(profession\_id);

create table titles(title\_id number,title varchar2(50),released varchar2(4),averageRating number,numberVotes number,gen\_id number);

alter table titles add constraint title\_titleidpk primary key(title\_id);

alter table titles add constraint titles\_genidfk foreign key(gen\_id) references genre(gen\_id);

alter table names add constraint names\_titleidfk foreign key(title\_id) references titles(title\_id);

**Q 3**

**a):** distinct keyword is used to take out different values of the column not the duplicate

we represent different names not duplicate that’s why it is being used.

**b):** SELECT DISTINCT name,title from title from titles ,names WHERE title\_id=knownfortitle1 and averageRating=6.5 and numberVotes>25000;

**c):**

DECLARE

Title\_id NUMBER;

TBL\_NME VARCHAR2(30);

V\_STRNG VARCHAR2(4000);

BEGIN

FOR i IN (SELECT TABLE\_NAME FROM USER\_TABLES WHERE TABLE\_NAME = 'titels' or table\_name='names') LOOP

TBL\_NME := i.TABLE\_NAME;

DBMS\_OUTPUT.PUT\_LINE('TABLE EXTRACTED IS ' || TBL\_NME);

FOR j IN(SELECT titel\_id,title FROM i.table\_name) LOOP

V\_EXEC\_OBJ\_STRNG := 'SELECT title\_id FROM ' || i.TABLE\_NAME;

EXECUTE IMMEDIATE V\_STRNG INTO LOG\_ID;

DBMS\_OUTPUT.PUT\_LINE('title\_id IS ' || title\_id || ' FOR TABLE ' || i.TABLE\_NAME);

END LOOP;

END LOOP;

END;

**d):** indexing makes columns faster to query by creating pointers to where data is stored within a database. Imagine you want to find a piece of information that is within a large database. To get this information out of the database the computer will look through every row until it finds it.

Title\_id,title,name\_id,name should be indexed

**e):**

create index index\_x on titles(title\_id,title);

create index index index\_y on names(name\_id,name);

**f):**

Hard Disk Drives Cost Advantage

Proven technology that’s been in use for decades makes hard disk drives cheap—much cheaper, per gigabyte than SSDs. HDD storage can run as low as three cents per gigabyte. You don’t spend a lot but you get lots of space. HDD makers continue to improve storage capacity while keeping costs low, so HDDs remain the choice of anyone looking for a lot of storage without spending a lot of money

In the Opposite Corner: The Solid-state Drive

SSDs have become much more common in recent years. They’re standard issue across Apple’s laptop line, for example the MacBook, MacBook Pro and MacBook Air all come standard with SSDs. So does the Mac Pro.

Inside an SSD Solid state is industry shorthand for an integrated circuit, and that’s the key difference between an SSD and a HDD: there are no moving parts inside an SSD. Rather than using disks, motors and read-write heads, SSDs use flash memory instead—that is, computer chips that retain their information even when the power is turned off

**g):** plan B

**h):** inner join is faster the implicit join because it returns only the matching records

select distinct name,title from titles inner join names on titles.title\_id=names.knownfortitle1;

**Q 4**

**a):**  
released will be changed 20113 in first row

8.6 will be changed to 9.1 in second row

1188129 will be changed to 0 in second row again

**b):**  
to bundle them into single transaction we begin starting poing then quries and end;

**c):** Serial schedule both by definition and execution means that the transactions bestowed upon it will take place serially, that is, one after the other. This leaves no place for inconsistency within the database. But, when a set of transactions are scheduled non-serially, they are interleaved leading to the problem of concurrency within the database. Non-serial schedules do not wait for one transaction to complete for the other one to begin

if you don't use "begin transaction" ... "end transaction" then every statement is treated as a transaction. This behaviour is often described as "auto commit" and you can switch that on as an option in MySQL.

Like above each statement is treated as separate transaction

**d):** A database lock is used to “lock” some data in a database so that only one database user/session may update that particular data. So, database locks exist to prevent two or more database users from updating the same exact piece of data at the same exact time.

**e):** Row-level locking means that only the row that is accessed by an application will be locked. Hence, all other rows that belong to the same page are free and can be used by other applications

Advantages of row-level locking:

Fewer lock conflicts when accessing different rows in many threads.

Fewer changes for rollbacks.

Makes it possible to lock a single row a long time.

Disadvantages of row-level locking:

Takes more memory than page-level or table-level locks.

Is slower than page-level or table-level locks when used on a large part of the table because you must acquire many more locks.

Is definitely much worse than other locks if you often do GROUP BY operations on a large part of the data or if you often must scan the entire table.

**f):**  
The lost update problem occurs when 2 concurrent transactions try to read and update the same data. Let’s understand this with the help of an example.

Suppose we have a table named “Product” that stores id, name, and ItemsinStock for a product.

It is used as part of an online system that displays the number of items in stock for a particular product and so needs to be updated each time a sale of that product is made.

The table looks like this:

Title\_id

averagerating

tt1345836

0.6

Now consider a scenario where a user arrives and initiates the process of entring a title\_id. This will initiate a transaction. Let’s call this transaction, transaction 1.

At the same time another user logs into the system and initiates a transaction, let’s call this transaction2

Take a look at the following figure.

