WORKSHEET- VI

***Q.1: ans**. – a,c,d.*

***Q.2: ans**. – a,c,d.*

***Q.3: ans**. – b.*

***Q.4: ans**. – c.*

***Q.5: ans**. – b.*

***Q.6: ans**. – b.*

***Q.7: ans**. – a.*

***Q.8: ans**. – c.*

***Q.9: ans**. – a.*

***Q.10: ans**. a.*

***Q.11: ans**. - Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean 'reversing normalization' or 'not to normalize'. It is an optimization technique that is applied after normalization.*

Basically, The process of taking a normalized schema and making it non-normalized is called denormalization, and designers use it to tune the performance of systems to support time-critical operations.

In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in a database.

For example, in a normalized database, we might have a Courses table and a Teachers table. Each entry in Courses would store the teacherID for a Course but not the teacherName. When we need to retrieve a list of all Courses with the Teacher's name, we would do a join between these two tables.

In some ways, this is great; if a teacher changes his or her name, we only have to update the name in one place.
The drawback is that if tables are large, we may spend an unnecessarily long time doing joins on tables. Denormalization, then, strikes a different compromise. Under denormalization, we decide that we're okay with some redundancy and some extra effort to update the database in order to get the efficiency advantages of fewer joins.

Pros of Denormalization:

1. *Retrieving data is faster since we do fewer joins*

2. *Queries to retrieve can be simpler(and therefore less likely to have bugs), since we need to look at fewer tables. Updates and inserts are more expensive.*
3. *Denormalization can make update and insert code harder to write.*
4. *Data may be inconsistent.*
5. *Data redundancy necessitates more storage.*
6. *In a system that demands scalability, like that of any major tech company, we almost always use elements of both normalized and denormalized databases.*

Q.12: ans*. - A database cursor is an identifier associated with a group of rows. It is, in a sense, a pointer to the current row in a buffer.*

You must use a cursor in the following cases:

- *Statements that return more than one row of data from the database server:*
    - *A SELECT statement requires a select cursor.*
    - *An EXECUTE FUNCTION statement requires a function cursor.*
- *An INSERT statement that sends more than one row of data to the database server requires an insert cursor.*

For more information about how to use cursors, see the IBM Informix Guide to SQL: Tutorial.

- *Receive more than one row*
- *Send more than one row*
- *Name the cursor*
- *Optimize cursor execution*

Q.13: ans*. - Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.*

These SQL commands are mainly categorized into five categories as:

1. *DDL – Data Definition Language*

WORKSHEET- VI

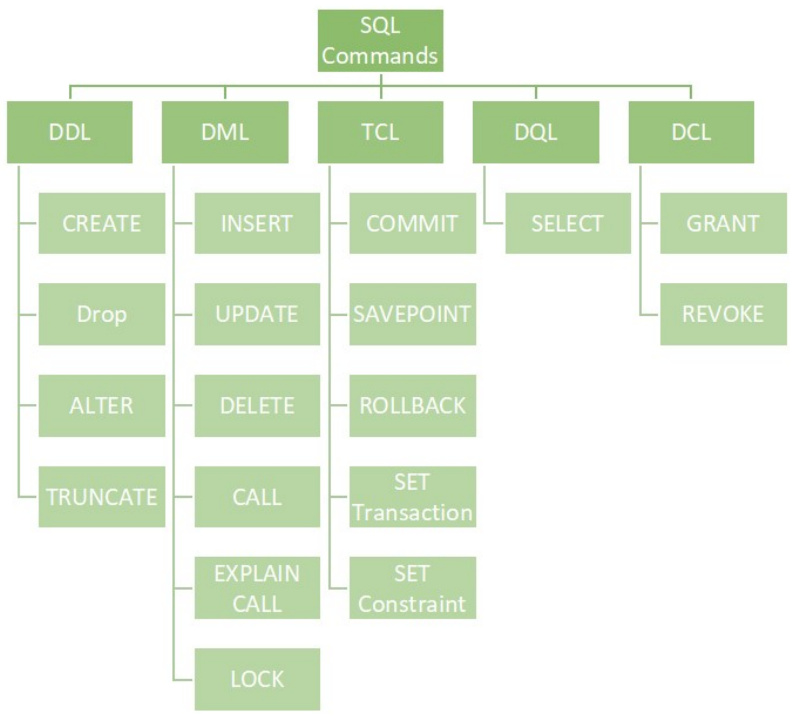
2. DQL – Data Query Language

3. DML – Data Manipulation Language

4. DCL – Data Control Language

5. TCL – Transaction Control Language

Now, we will see all of these in detail.



DDL (Data Definition Language):

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

- *CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).*

- *DROP: This command is used to delete objects from the database.*

WORKSHEET- VI

- *ALTER: This is used to alter the structure of the database.*
- *TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.*
- *COMMENT: This is used to add comments to the data dictionary.*
- *RENAME: This is used to rename an object existing in the database.*

DQL (Data Query Language):

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

- *SELECT: It is used to retrieve data from the database.*

DML(Data Manipulation Language):

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

- *INSERT : It is used to insert data into a table.*
- *UPDATE: It is used to update existing data within a table.*
- *DELETE : It is used to delete records from a database table.*
- *LOCK: Table control concurrency.*
- *CALL: Call a PL/SQL or JAVA subprogram.*
- *EXPLAIN PLAN: It describes the access path to data.*

DCL (Data Control Language):

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

WORKSHEET- VI

List of DCL commands:

- *GRANT: This command gives users access privileges to the database.*
- *REVOKE: This command withdraws the user's access privileges given by using the GRANT command.*

TCL (Transaction Control Language):

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure. You can explore more about transactions here. Hence, the following TCL commands are used to control the execution of a transaction:

- *COMMIT: Commits a Transaction.*
- *ROLLBACK: Rollbacks a transaction in case of any error occurs.*
- *SAVEPOINT: Sets a save point within a transaction.*
- *SET TRANSACTION: Specifies characteristics for the transaction.*

.

***Q.14: ans**. - Many random variables have distributions that are asymptotically Gaussian but may be significantly non-Gaussian for small numbers. For example the Poisson Distribution, which describes (among other things) the number of unlikely events occurring after providing a sufficient opportunity for a few events to occur. It is pretty non-Gaussian unless the mean number of events is very large. The mathematical form of the distribution is still Poisson, but a histogram of the number of events after many trials with a large average number of events eventually looks fairly Gaussian.*

For me, the best examples come from my field of research (astrophysical data analysis). For example, something that comes up all the time is that we detect stars in astronomical images and solve for their celestial coordinates. My current project uses images about 1.5 degrees on a side and typically detects 60 to 80 thousand stars per image, with the number well modeled as a Poisson Distribution, assuming that the image is not of a star cluster surrounded by mostly empty space. That's about 8 or 9 stars per square arcminute. If we cut out "postage stamps"from the image that are half an arcminute per side, then the mean number of detected stars in them is about 2. If we do that for (say) 1000 postage stamps and make a histogram of the number of detected stars in them, it will not look very Gaussian, but as we increase the size of the postage stamps, it becomes asymptotically Gaussian.

What generally never becomes Gaussian, however, is the Uniform Distribution. A histogram of the stars' right ascensions or declinations (the azimuthal and elevation angles used in astronomy) looks a lot like a step function, i.e., flat within the image boundaries. The positions are not uniformly spaced, but they *are distributed in the same way as a uniformly distributed random variable for any size postage stamp, including the entire image.*

Another example is the location of the centers of raindrop ripples on a pond; they are not uniformly spaced in (say) the east-west direction, but they are uniformly distributed.

The simplest example is the distribution of numbers that show up on the top of a fair die after a large number of throws. Each number from 1 to 6 will occur with approximately equal frequency. Increasing the number of throws will not tend to produce a bell-shaped histogram, in fact the fractional occurrence will approach a constant 1/6 over the possible numbers.

***Q.15: ans**. For me, the best examples come from my field of research (astrophysical data analysis). For example, something that comes up all the time is that we detect stars in astronomical images and solve for their celestial coordinates. My current project uses images about 1.5 degrees on a side and typically detects 60 to 80 thousand stars per image, with the number well modeled as a Poisson Distribution, assuming that the image is not of a star cluster surrounded by mostly empty space. That's about 8 or 9 stars per square arcminute. If we cut out "postage stamps"from the image that are half an arcminute per side, then the mean number of detected stars in them is about 2. If we do that for (say) 1000 postage stamps and make a histogram of the number of detected stars in them, it will not look very Gaussian, but as we increase the size of the postage stamps, it becomes asymptotically Gaussian.*

What generally never becomes Gaussian, however, is the Uniform Distribution. A histogram of the stars' right ascensions or declinations (the azimuthal and elevation angles used in astronomy) looks a lot like a step function, i.e., flat within the image boundaries. The positions are not uniformly spaced, but they are distributed in the same way as a uniformly distributed random variable for any size postage stamp, including the entire image.

Many random variables have distributions that are asymptotically Gaussian but may be significantly non-Gaussian for small numbers. For example the Poisson Distribution, which describes (among other things) the number of unlikely events occurring after providing a sufficient opportunity for a few events to occur. It is pretty non-Gaussian unless the mean number of events is very large. The mathematical form of the distribution is still Poisson, but a histogram of the number of events after many trials with a large average number of events eventually looks fairly Gaussian.

For me, the best examples come from my field of research (astrophysical data analysis). For example, something that comes up all the time is that we detect stars in astronomical images and solve for their celestial coordinates. My current project uses images about 1.5 degrees on a side and typically detects 60 to 80 thousand stars per image, with the number well modeled as a Poisson Distribution, assuming that the image is not of a star cluster surrounded by mostly empty space. That's about 8 or 9 stars per square arcminute. If we cut out "postage stamps"from the image that are half an arcminute per side, then the mean number of detected stars in them is about 2. If we do that for (say) 1000 postage stamps and make a histogram of the number of detected stars in them, it will not look very Gaussian, but as we increase the size of the postage stamps, it becomes asymptotically Gaussian.

WORKSHEET- VI

What generally never becomes Gaussian, however, is the Uniform Distribution. A histogram of the stars' right ascensions or declinations (the azimuthal and elevation angles used in astronomy) looks a lot like a step function, i.e., flat within the image boundaries. The positions are not uniformly spaced, but they are distributed in the same way as a uniformly distributed random variable for any size postage stamp, including the entire image.

Another example is the location of the centers of raindrop ripples on a pond; they are not uniformly spaced in (say) the east-west direction, but they are uniformly distributed.

The simplest example is the distribution of numbers that show up on the top of a fair die after a large number of throws. Each number from 1 to 6 will occur with approximately equal frequency. Increasing the number of throws will not tend to produce a bell-shaped histogram, in fact the fractional occurrence will approach a constant 1/6 over the possible numbers.