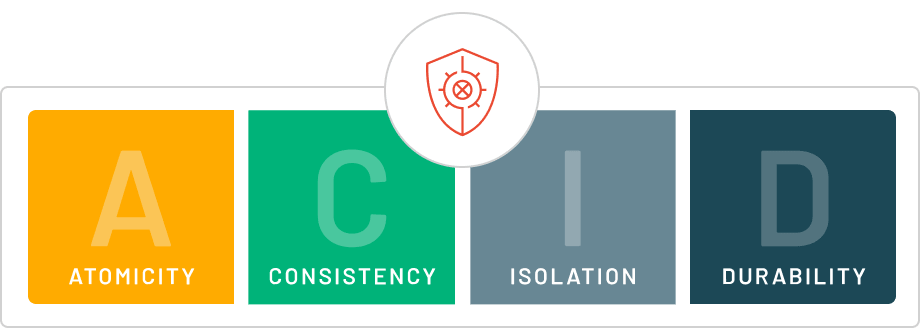
**UNIT – III**

**Introduction to SQL:** Introduction to Database Testing, DML statements - Select, insert, update, delete, truncate, DDL Statements - Create, Alter, Drop, DCL & TCL statements-Grant, Revoke & commit, Rollback, Save point. How to configure the JDBC driver. JDBC API.

**What is a transaction?**

In the context of databases and data storage systems, a **transaction** is any operation that is treated as a single unit of work, which either completes fully or does not complete at all, and leaves the storage system in a consistent state. The classic example of a transaction is what occurs when you withdraw money from your bank account. Either the money has left your bank account, or it has not — there cannot be an in-between state.

**A.C.I.D. properties: Atomicity, Consistency, Isolation, and Durability**

ACID is an acronym that refers to the set of 4 key properties that define a transaction: **Atomicity, Consistency, Isolation,** and **Durability.** If a database operation has these ACID properties, it can be called an ACID transaction, and data storage systems that apply these operations are called transactional systems. ACID transactions guarantee that each read, write, or modification of a table has the following properties:

* **Atomicity** - each statement in a transaction (to read, write, update or delete data) is treated as a single unit. Either the entire statement is executed, or none of it is executed. This property prevents data loss and corruption from occurring if, for example, if your streaming data source fails mid-stream.
* **Consistency** - ensures that transactions only make changes to tables in predefined, predictable ways. Transactional consistency ensures that corruption or errors in your data do not create unintended consequences for the integrity of your table.
* **Isolation** - when multiple users are reading and writing from the same table all at once, isolation of their transactions ensures that the concurrent transactions don't interfere with or affect one another. Each request can occur as though they were occurring one by one, even though they're actually occurring simultaneously.
* **Durability** - ensures that changes to your data made by successfully executed transactions will be saved, even in the event of system failure.

**Why are ACID transactions a good thing to have?**

ACID transactions ensure the highest possible data reliability and integrity. They ensure that your data never falls into an inconsistent state because of an operation that only partially completes. For example, without ACID transactions, if you were writing some data to a database table, but the power went out unexpectedly, it's possible that only some of your data would have been saved, while some of it would not. Now your database is in an inconsistent state that is very difficult and time-consuming to recover from.

**Delta Lake: Reliable, consistent data with the guarantees of ACID transactions**

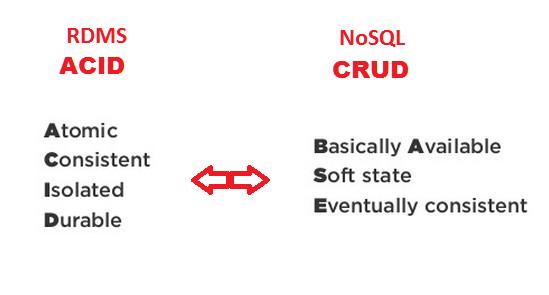
ACID transactions have long been one of the most enviable properties of data warehouses, but [Delta Lake](https://www.databricks.com/product/delta-lake-on-databricks) has now brought them to [data lakes](https://www.databricks.com/discover/data-lakes/introduction). They allow users to see consistent views of their data even while new data is being written to the table in real-time, because each write is an isolated transaction that is recorded in an ordered transaction log. [Delta Lake employs the highest level of isolation possible (serializable isolation), ensuring that reads and writes to a single table are consistent and reliable.] By implementing ACID transactions, Delta Lake effectively solves for several of the previously listed criticisms of Lambda architecture: its complexity, incorrect views of data, and the rework and reprocessing needed after Lambda pipelines inevitably break. Users can perform multiple concurrent transactions on their data, and in the event of an error with a data source or a stream, Delta Lake cancels execution of the transaction to ensure that the data is kept clean and intact. The beauty of ACID transactions is that users can trust the data that is stored in Delta Lake. A data analyst making use of Delta Lake tables to perform ETL on his or her data to ready it for dashboarding can count on the fact that the KPIs he or she is seeing represent the actual state of the data. A machine learning engineer using Delta Lake tables to perform feature engineering can be 100% confident that all of his or her transformations and aggregations either executed exactly as intended, or didn't execute at all (in which case, he or she would be notified). The value of knowing that the mental model you have of your data is actually reflective of its true underlying state cannot be overstated.

In computer science, the word ACID has a special significance. The is one of important nature which differentiates the traditional RDBMS and NoSQL databases.

When I was going through some NOSQL database architectures like HBase, Cassandra etc..I was thinking like why Pega doesn't support NoSQL database as a primary storage DB to improve the storage capacity. Then this ACID concept cleared all my all my questions.

**What is ACID nature?**

ACID defines some set of properties which comes into picture when we deal with database transactions. ACID stands for (Atomicity, Consistency, Isolation, Durability).



Transaction can be anything like transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another. They comes under a single transaction.

It can guarantee that the database transactions are processed reliably.  
Lets discuss with one example.

Lets say, I want to transfer some amount from my account A to other account B.

**1.If Atomicity doesn't exist:**

When the transaction is going on, suddenly my network is down. Now the database state should be left unchanged. This is possible only if the DB supports Atomicity nature.

**2.If Consistency doesn't exist:**

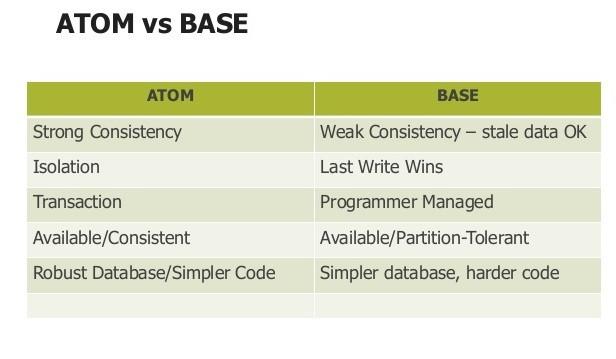
The consistency property ensures that the correctness of the transaction.That means any data written to the database must be valid with the set of all rules defined.

**3.If Isolation doesn't exist:**

The isolation property ensures concurrency control of multiple transactions.

**4.If Durability doesn't exist:**

The isolation property ensures that once the transaction is committed ( or completed), results need to be stored permanently in the database even if there is a system crash or power failure immediately.



ACID and BASE are two databases transaction models.

**Conclusion:**

Though No-SQL databases may dispense with various portions of ACID as discussed above, they can still provide certain other benefits--partition tolerance, performance..etc

**What is Relational Model?**

**Relational Model (RM)** represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

**Some popular Relational Database management systems are:**

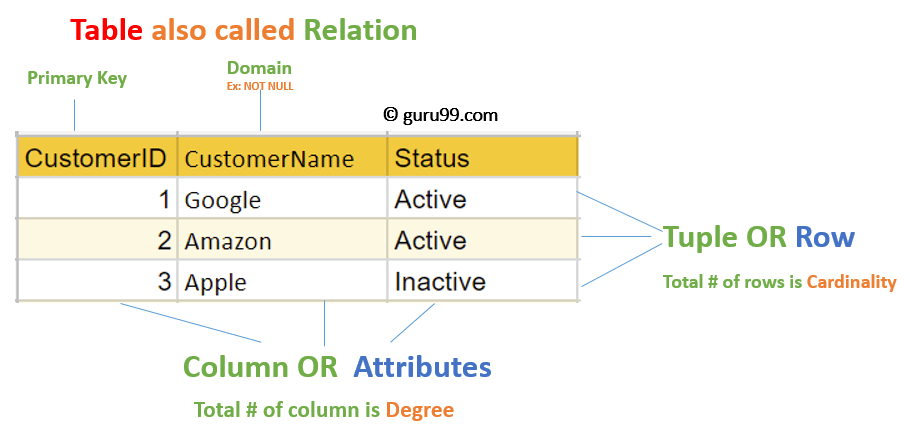
* DB2 and Informix Dynamic Server – IBM
* Oracle and RDB – Oracle
* SQL Server and Access – Microsoft

In this tutorial, you will learn

* [Relational Model Concepts in DBMS](https://www.guru99.com/relational-data-model-dbms.html#2)
* [Relational Integrity Constraints](https://www.guru99.com/relational-data-model-dbms.html#3)
* [Operations in Relational Model](https://www.guru99.com/relational-data-model-dbms.html#4)
* [Best Practices for creating a Relational Model](https://www.guru99.com/relational-data-model-dbms.html#5)
* [Advantages of Relational Database Model](https://www.guru99.com/relational-data-model-dbms.html#6)
* [Disadvantages of Relational Model](https://www.guru99.com/relational-data-model-dbms.html#7)

**Relational Model Concepts in DBMS**

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student\_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:**Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** – Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain



**Relational Integrity Constraints**

Relational Integrity constraints in DBMS are referred to conditions which must be present for a valid relation. These Relational constraints in DBMS are derived from the rules in the mini-world that the database represents.

There are many types of Integrity Constraints in DBMS. Constraints on the Relational database management system is mostly divided into three main categories are:

1. Domain Constraints
2. Key Constraints
3. Referential Integrity Constraints

**Domain Constraints**

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

**Example:**

Create DOMAIN CustomerName

CHECK (value not NULL)

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

**Key Constraints**

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

**Example:**

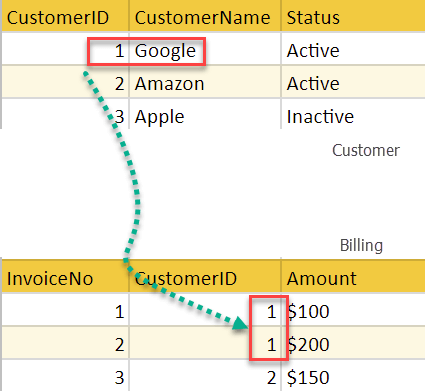
In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =” Google”.

| **CustomerID** | **CustomerName** | **Status** |
| --- | --- | --- |
| 1 | Google | Active |
| 2 | Amazon | Active |
| 3 | Apple | Inactive |
|  |  |  |

**Referential Integrity Constraints**

Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

**Example:**



In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300

**Operations in Relational Model**

Four basic update operations performed on relational database model are

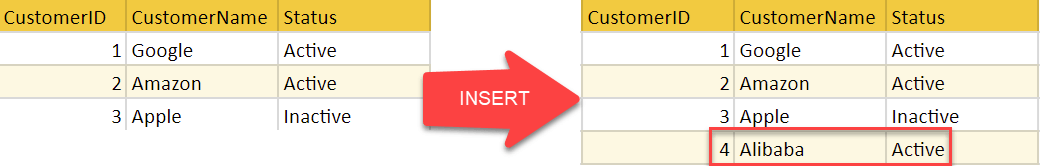
Insert, update, delete and select.

* Insert is used to insert data into the relation
* Delete is used to delete tuples from the table.
* Modify allows you to change the values of some attributes in existing tuples.
* Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

**Insert Operation**

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.



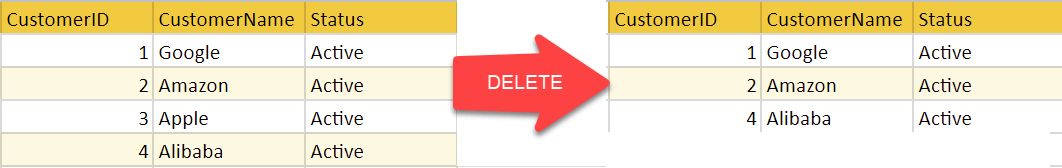
**Update Operation**

You can see that in the below-given relation table CustomerName= ‘Apple’ is updated from Inactive to Active.



**Delete Operation**

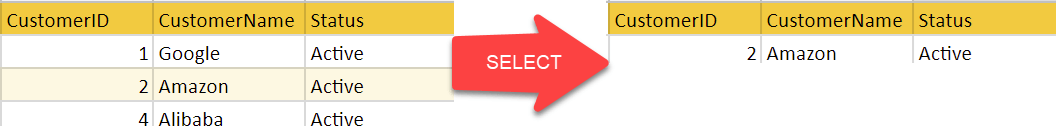
To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.



In the above-given example, CustomerName= “Apple” is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same [database](https://www.guru99.com/introduction-to-database-sql.html).

**Select Operation**



In the above-given example, CustomerName=”Amazon” is selected

**Best Practices for creating a Relational Model**

* Data need to be represented as a collection of relations
* Each relation should be depicted clearly in the table
* Rows should contain data about instances of an entity
* Columns must contain data about attributes of the entity
* Cells of the table should hold a single value
* Each column should be given a unique name
* No two rows can be identical
* The values of an attribute should be from the same domain

**Advantages of Relational Database Model**

* **Simplicity**: A Relational data model in DBMS is simpler than the hierarchical and network model.
* **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
* **Easy to use**: The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand
* **Query capability**: It makes possible for a high-level query language like [SQL](https://www.guru99.com/sql.html) to avoid complex database navigation.
* **Data independence**: The Structure of Relational database can be changed without having to change any application.
* **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

**Disadvantages of Relational Model**

* Few relational databases have limits on field lengths which can’t be exceeded.
* Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
* Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

**Summary**

* The Relational database modelling represents the database as a collection of relations (tables)
* Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation instance, are some important components of Relational Model
* Relational Integrity constraints are referred to conditions which must be present for a valid Relation approach in DBMS
* Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type
* Insert, Select, Modify and Delete are the operations performed in Relational Model constraints
* The relational database is only concerned with data and not with a structure which can improve the performance of the model
* Advantages of Relational model in DBMS are simplicity, structural independence, ease of use, query capability, data independence, scalability, etc.
* Few relational databases have limits on field lengths which can’t be exceeded.

SQL (Structured Query Language) is a programming language mainly used to manage and manipulate relational databases. So when you want to perform some operation on the data available on the database, you have to write a query for that, and that SQL query should follow the predefined SQL syntax.

If the user is not writing the SQL syntax correctly, then the Query will not be executed, and the SQL server will give you an error message. So to get the requested records retrieved from the target database, you have to write the query in the correct SQL syntax.

* In a SQL statement, the first word will always be a Keyword.
* **SQL keywords** are the words that have a specific meaning in the SQL language, such as **SELECT, FROM, WHERE**, etc. These keywords can be written in both uppercase and lowercase letters, but it is common practice to write them in **uppercase** to improve the readability of the SQL query. This is because uppercase letters stand out more than lowercase letters and make it easier to identify the keywords in the query.
* SQL syntax must end with a Semicolon (;).
* **SQL statements or syntax are dependent on text lines**. This means an SQL statement’s structure and format are determined by how it is written on the text lines. A single SQL statement can be written on one or multiple text lines, affecting how it is executed. For example, a single SQL statement written on one line will be executed as a single statement. In contrast, a single SQL statement written on multiple lines will be executed as separate statements.
* SQL is a powerful and versatile language that can perform a wide range of actions on a database. With SQL statements, you can **create, modify, and delete databases, tables**, and other objects and **insert, update**, and **retrieve data from the database**. SQL is also used to set permissions and manage users and user groups.
* SQL syntax is based on mathematical concepts called relational algebra and tuple relational calculus. These concepts define the relationships between data in a database and the operations that can be performed on that data. This means that the SQL language is designed to work with data in a structured, relational way, allowing users to manipulate and query the data logically and mathematically.
* **SQL is not case-sensitive, meaning** the SQL keywords and commands do not have to be entered in a specific case (uppercase or lowercase) to be recognized and executed by the database management system.  
    
  This feature of SQL can make it more user-friendly and convenient for developers and QA’s, as they do not have to worry about the case of their SQL keywords and commands. It also allows for more flexibility when writing SQL statements, as developers can use the case they prefer or that is most legible.
* When creating table and column names, be sure to use lowercase letters. If you need to use multiple words, you can separate them with an underscore (\_) or camelCase.
* If you are working with any String, you need to use single quotes (‘ ‘);
* And during working with numeric values, they should not be enclosed in quotes.
* While working with dates, the date format should be in “YYYY-MM-DD” format, and date values should be enclosed by single quotes (‘ ‘).

Remember that the SQL syntax may vary slightly depending on your specific database management system.

**SQL Syntax Commands**

Below we have listed down all the SQL syntax of various SQL commands. But we will discuss all of those SQL commands in detail and learn how to use those efficiently.

**Select Statement**

* **SELECT** "column\_name" **FROM** "table\_name";

**Distinct**

* **SELECT** **DISTINCT** "column\_name"
* **FROM** "table\_name";

**Where**;

* **SELECT** "column\_name"
* **FROM** "table\_name"
* **WHERE** "condition"

**And/Or**

* **SELECT** "column\_name"
* **FROM** "table\_name"
* **WHERE** "simple condition"
* {[AND|OR] "simple condition"};

**In**

* **SELECT** "column\_name"
* **FROM** "table\_name"
* **WHERE** "column\_name" IN ('value1', 'value2', ...);

**Between**

* **SELECT** "column\_name"
* **FROM** "table\_name"
* **WHERE** "column\_name" BETWEEN 'value1' AND 'value2';

**Like**

* **SELECT** "column\_name"
* **FROM** "table\_name"
* **WHERE** "column\_name" LIKE {PATTERN};

**Order By**

* **SELECT** "column\_name"
* **FROM** "table\_name"
* [**WHERE** "condition"]
* **ORDER** **BY** "column\_name" [**ASC**, **DESC**];

**Count**

* **SELECT** **COUNT**("column\_name")
* **FROM** "table\_name";

**Group By**

* **SELECT** "column\_name1", **SUM**("column\_name2")
* **FROM** "table\_name"
* **GROUP** **BY** "column\_name1";

**Having**

* **SELECT** "column\_name1", [**Function**("column\_name2")]
* **FROM** "table\_name"
* [**GROUP** **BY** "column\_name1"]
* **HAVING** (arithematic **function** condition);

**Create Table Statement**

* **CREATE** **TABLE** "table\_name"
* ("column 1" "data type for column 1" [**column** 1 **constraint**(s)],
* "column 2" "data type for column 2" [**column** 2 **constraint**(s)],
* ...
* [**table** **constraint**(s)]);

**Drop Table Statement**

* **DROP** **TABLE** "table\_name";

**Truncate Table Statement**

* **TRUNCATE** **TABLE** "table\_name";

**Insert Into Statement**

* **INSERT** **INTO** "table\_name" ("column1", "column2", ...)
* **VALUES** ("value1", "value2", ...);

**Insert Into Select Statement**

* **INSERT** **INTO** "table1" ("column1", "column2", ...)
* **SELECT** "column3", "column4", ...
* **FROM** "table2";

**Update Statement**

* **UPDATE** "table\_name"
* **SET** "column\_1" = [new value]
* **WHERE** "condition";

**Delete From Statement**

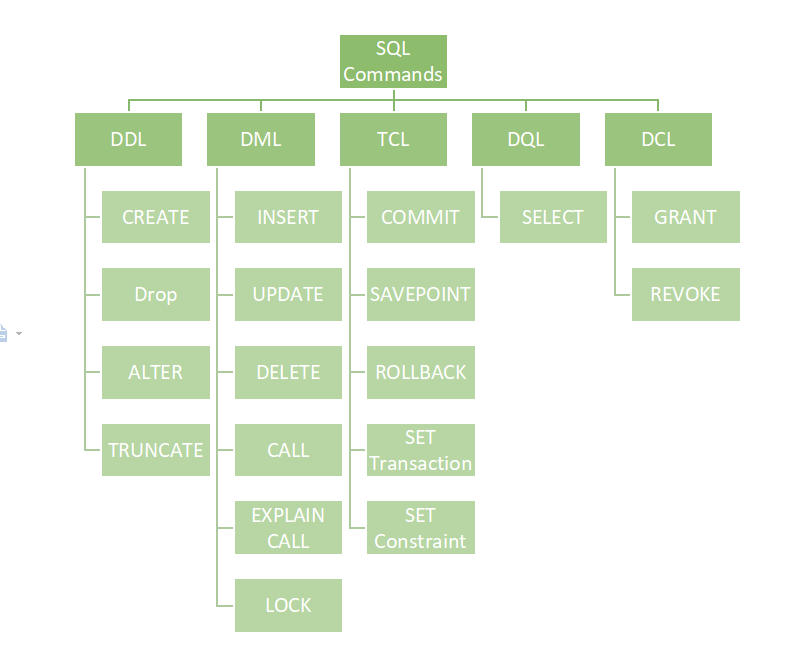
* **DELETE** **FROM** "table\_name"
* **WHERE** "condition";

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](https://www.geeksforgeeks.org/structured-query-language/) uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These [SQL](https://www.geeksforgeeks.org/sql-concepts-and-queries/)commands are mainly categorized into five categories as:

1. DDL – Data Definition Language
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](https://www.geeksforgeeks.org/features-of-structured-query-language-sql/) 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**](https://www.geeksforgeeks.org/sql-create/): This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
* [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/): This command is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)**:**This is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)**:**This is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/): This is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-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**](https://www.geeksforgeeks.org/sql-select-clause/)**:**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**](https://www.geeksforgeeks.org/sql-insert-statement/) : It is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/)**:** It is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) : It is used to delete records from a database table.
* [**LOCK:**](https://www.geeksforgeeks.org/sql-lock-table/) 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.

List of  DCL commands:

* [**GRANT:**](https://www.geeksforgeeks.org/mysql-grant-revoke-privileges/)This commandgives users access privileges to the database.
* [**REVOKE:**](https://www.geeksforgeeks.org/difference-between-grant-and-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***](https://www.geeksforgeeks.org/sql-transactions/). Hence, the following TCL commands are used to control the execution of a transaction:

* [**COMMIT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Commits a Transaction.
* [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)**:**Rollbacks a transaction in case of any error occurs.
* [**SAVEPOINT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Sets a save point within a transaction.
* [**SET TRANSACTION:**](https://www.geeksforgeeks.org/sql-transactions/)Specifies characteristics for the transaction.

**What is Database Normalization?**

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the [relational model](https://www.guru99.com/relational-data-model-dbms.html) Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

**Database Normal Forms**

Here is a list of Normal Forms in SQL:

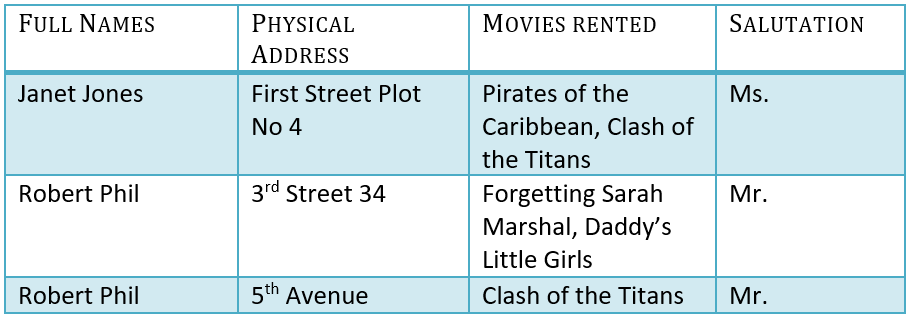
* 1NF (First Normal Form)
* 2NF (Second Normal Form)
* 3NF (Third Normal Form)
* BCNF (Boyce-Codd Normal Form)
* 4NF (Fourth Normal Form)
* 5NF (Fifth Normal Form)
* 6NF (Sixth Normal Form)

The Theory of Data Normalization in MySQL server is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form**. The evolution of Normalization in SQL theories is illustrated below-

Database Normal FormsDatabase Normal Forms

**Database Normalization With Examples**

Database **Normalization Example** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below. Let’s understand Normalization database with normalization example with solution:



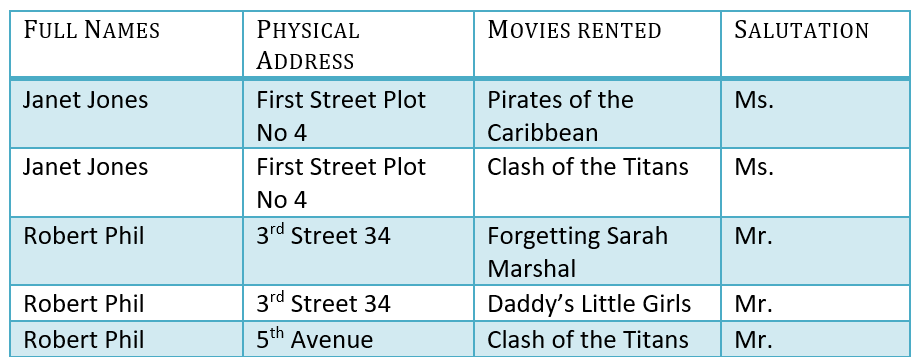
Here you see **Movies Rented column has multiple values.** Now let’s move into 1st Normal Forms:

**1NF (First Normal Form) Rules**

* Each table cell should contain a single value.
* Each record needs to be unique.

The above table in 1NF-

**1NF Example**



Example of 1NF in DBMS

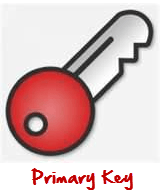
Before we proceed let’s understand a few things —

**What is a KEY in SQL?**

A **KEY in SQL** is a value used to identify records in a table uniquely. An SQL KEY is a single column or combination of multiple columns used to uniquely identify rows or tuples in the table. SQL Key is used to identify duplicate information, and it also helps establish a relationship between multiple tables in the database.

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

**What is a Primary Key?**



Primary Key in DBMS

A primary is a single column value used to identify a database record uniquely.

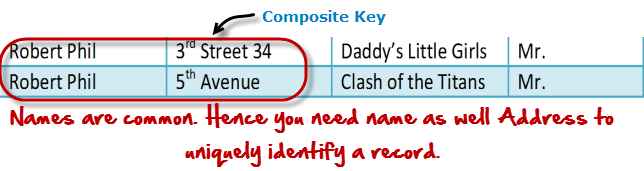
It has following attributes

* A [primary key](https://www.guru99.com/dbms-keys.html) cannot be NULL
* A primary key value must be unique
* The primary key values should rarely be changed
* The primary key must be given a value when a new record is inserted.

**What is Composite Key?**

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.



Composite key in Database

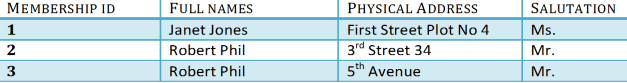
Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

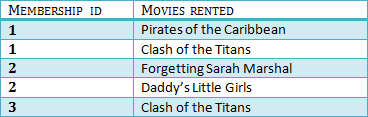
Let’s move into second normal form 2NF

**2NF (Second Normal Form) Rules**

* Rule 1- Be in 1NF
* Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key relation

It is clear that we can’t move forward to make our simple database in 2nd Normalization form unless we partition the table above.



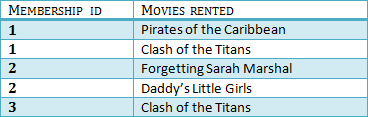


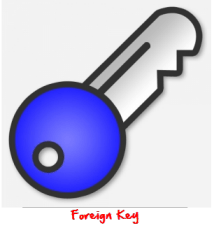
We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

**Database – Foreign Key**

In Table 2, Membership\_ID is the Foreign Key

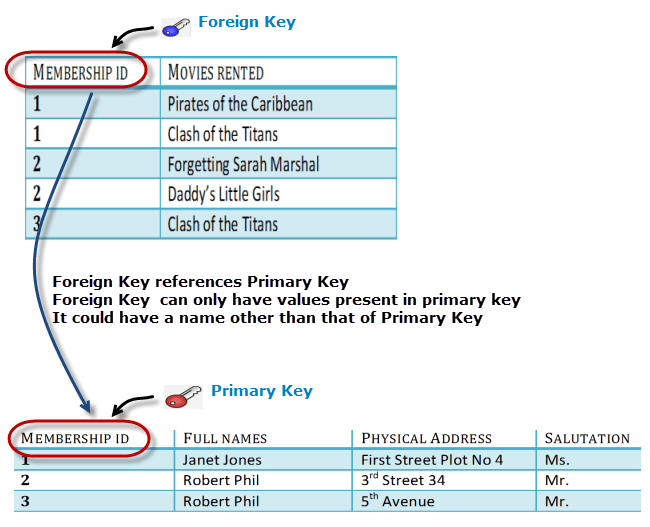




Foreign Key in DBMS

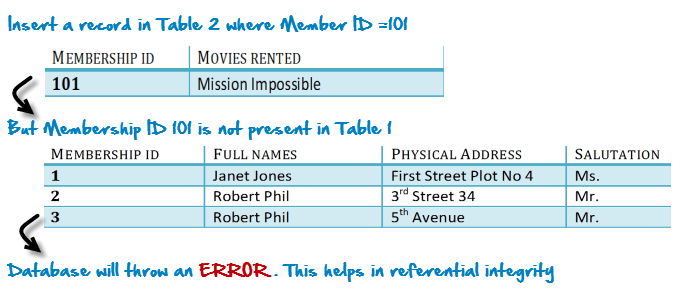
Foreign Key references the primary key of another Table! It helps connect your Tables

* A foreign key can have a different name from its primary key
* It ensures rows in one table have corresponding rows in another
* Unlike the Primary key, they do not have to be unique. Most often they aren’t
* Foreign keys can be null even though primary keys can not



**Why do you need a foreign key?**

Suppose, a novice inserts a record in Table B such as



You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.

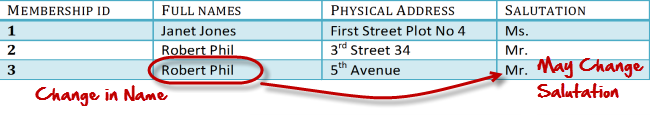
The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

**What are transitive functional dependencies?**

A transitive [functional dependency](https://www.guru99.com/dbms-functional-dependency.html) is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table 1. Changing the non-key column Full Name may change Salutation.



Let’s move into 3NF

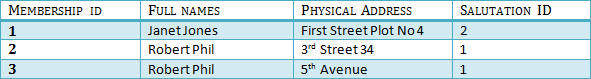
**3NF (Third Normal Form) Rules**

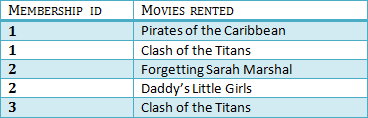
* Rule 1- Be in 2NF
* Rule 2- Has no transitive functional dependencies

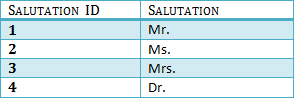
To move our 2NF table into 3NF, we again need to again divide our table.

**3NF Example**

Below is a 3NF example in SQL database:







We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF

In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Now our little example is at a level that cannot further be decomposed to attain higher normal form types of normalization in DBMS. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of normalisation in DBMS in brief in the following.

**BCNF (Boyce-Codd Normal Form)**

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate**Key.

Sometimes is BCNF is also referred as **3.5 Normal Form.**

**4NF (Fourth Normal Form) Rules**

If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4th Normal Form.

**5NF (Fifth Normal Form) Rules**

A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

**6NF (Sixth Normal Form) Proposed**

6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future…

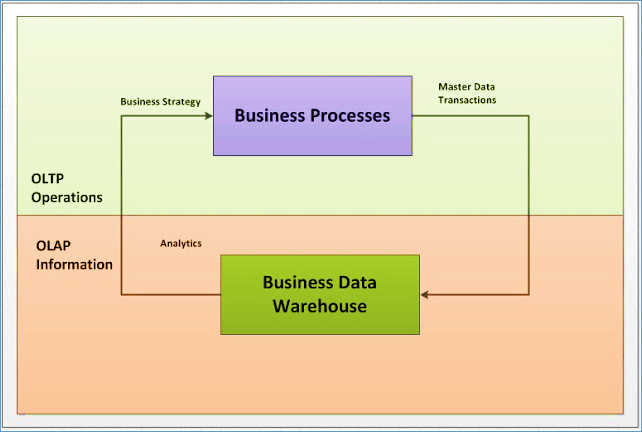
That’s all to SQL Normalization!!!

**Summary**

* Database designing is critical to the successful implementation of a database management system that meets the data requirements of an enterprise system.
* Normalization in DBMS is a process which helps produce database systems that are cost-effective and have better security models.
* Functional dependencies are a very important component of the normalize data process
* Most database systems are normalized database up to the third normal forms in DBMS.
* A primary key uniquely identifies are record in a Table and cannot be null
* A foreign key helps connect table and references a primary key

**Key Difference between OLTP and OLAP**

* Online Analytical Processing (OLAP) is a category of software tools that analyze data stored in a database, whereas Online transaction processing (OLTP) supports transaction-oriented applications in a 3-tier architecture.
* OLAP creates a single platform for all types of business analysis needs which includes planning, budgeting, forecasting, and analysis, while OLTP is useful for administering day-to-day transactions of an organization.
* OLAP is characterized by a large volume of data, while OLTP is characterized by large numbers of short online transactions.
* In OLAP, a data warehouse is created uniquely so that it can integrate different data sources for building a consolidated database, whereas OLTP uses traditional DBMS.

OLTP vs OLAP

**What is OLAP?**

Online Analytical Processing, a category of software tools which provide analysis of data for business decisions. [OLAP systems](https://www.guru99.com/online-analytical-processing.html) allow users to analyze database information from multiple database systems at one time.

**The primary objective is data analysis and not data processing**.

**What is OLTP?**

Online transaction processing shortly known as [OLTP](https://www.guru99.com/what-is-oltp.html) supports transaction-oriented applications in a 3-tier architecture. OLTP administers day to day transaction of an organization.

**The primary objective is data processing and not data analysis**

**Example of OLAP**

Any Datawarehouse system is an OLAP system. Uses of OLAP are as follows

* A company might compare their mobile phone sales in September with sales in October, then compare those results with another location which may be stored in a sperate database.
* Amazon analyzes purchases by its customers to come up with a personalized homepage with products which likely interest to their customer.

**Example of OLTP system**

An example of OLTP system is ATM center. Assume that a couple has a joint account with a bank. One day both simultaneously reach different ATM centers at precisely the same time and want to withdraw total amount present in their bank account.

However, the person that completes authentication process first will be able to get money. In this case, OLTP system makes sure that withdrawn amount will be never more than the amount present in the bank. The key to note here is that OLTP systems are optimized for**transactional superiority instead data analysis.**  
Other examples of OLTP applications are:

* Online banking
* Online airline ticket booking
* Sending a text message
* Order entry
* Add a book to shopping cart

**Difference between OLTP and OLAP**

Below is the difference between OLAP and OLTP in Data Warehouse:

| **Parameters** | **OLTP** | **OLAP** |
| --- | --- | --- |
| **Process** | It is an online transactional system. It manages database modification. | OLAP is an online analysis and data retrieving process. |
| **Characteristic** | It is characterized by large numbers of short online transactions. | It is characterized by a large volume of data. |
| **Functionality** | OLTP is an online database modifying system. | OLAP is an online database query management system. |
| **Method** | OLTP uses traditional DBMS. | OLAP uses the data warehouse. |
| **Query** | Insert, Update, and Delete information from the database. | Mostly select operations |
| **Table** | Tables in OLTP database are normalized. | Tables in OLAP database are not normalized. |
| **Source** | OLTP and its transactions are the sources of data. | Different OLTP databases become the source of data for OLAP. |
| **Data Integrity** | OLTP database must maintain data integrity constraint. | OLAP database does not get frequently modified. Hence, data integrity is not an issue. |
| **Response time** | It’s response time is in millisecond. | Response time in seconds to minutes. |
| **Data quality** | The data in the OLTP database is always detailed and organized. | The data in OLAP process might not be organized. |
| **Usefulness** | It helps to control and run fundamental business tasks. | It helps with planning, problem-solving, and decision support. |
| **Operation** | Allow read/write operations. | Only read and rarely write. |
| **Audience** | It is a market orientated process. | It is a customer orientated process. |
| **Query Type** | Queries in this process are standardized and simple. | Complex queries involving aggregations. |
| **Back-up** | Complete backup of the data combined with incremental backups. | OLAP only need a backup from time to time. Backup is not important compared to OLTP |
| **Design** | DB design is application oriented. Example: Database design changes with industry like Retail, Airline, Banking, etc. | DB design is subject oriented. Example: Database design changes with subjects like sales, marketing, purchasing, etc. |
| **User type** | It is used by Data critical users like clerk, DBA & Data Base professionals. | Used by Data knowledge users like workers, managers, and CEO. |
| **Purpose** | Designed for real time business operations. | Designed for analysis of business measures by category and attributes. |
| **Performance metric** | Transaction throughput is the performance metric | Query throughput is the performance metric. |
| **Number of users** | This kind of Database users allows thousands of users. | This kind of [Database](https://www.guru99.com/introduction-to-database-sql.html) allows only hundreds of users. |
| **Productivity** | It helps to Increase user’s self-service and productivity | Help to Increase productivity of the business analysts. |
| **Challenge** | Data Warehouses historically have been a development project which may prove costly to build. | An OLAP cube is not an open SQL server data warehouse. Therefore, technical knowledge and experience is essential to manage the OLAP server. |
| **Process** | It provides fast result for daily used data. | It ensures that response to the query is quicker consistently. |
| **Characteristic** | It is easy to create and maintain. | It lets the user create a view with the help of a spreadsheet. |
| **Style** | OLTP is designed to have fast response time, low data redundancy and is normalized. | A data warehouse is created uniquely so that it can integrate different data sources for building a consolidated database |

**Benefits of using OLAP services**

* [OLAP](https://www.guru99.com/online-analytical-processing.html) creates a single platform for all types of business analytical needs which includes planning, budgeting, forecasting, and analysis.
* The main benefit of OLAP is the consistency of information and calculations.
* Easily apply security restrictions on users and objects to comply with regulations and protect sensitive data.

**Benefits of OLTP method**

* It administers daily transactions of an organization.
* OLTP widens the customer base of an organization by simplifying individual processes.

**Drawbacks of OLAP service**

* Implementation and maintenance are dependent on IT professional because the traditional OLAP tools require a complicated modeling procedure.
* OLAP tools need cooperation between people of various departments to be effective which might always be not possible.

**Drawbacks of OLTP method**

* If [OLTP](https://www.guru99.com/what-is-oltp.html) system faces hardware failures, then online transactions get severely affected.
* OLTP systems allow multiple users to access and change the same data at the same time which many times created unprecedented situation.

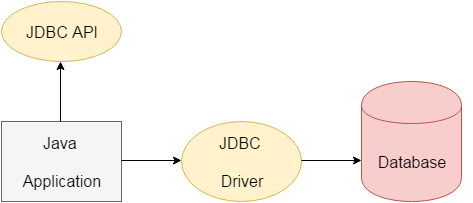
# Java JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

* JDBC-ODBC Bridge Driver,
* Native Driver,
* Network Protocol Driver, and
* Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular *classes* of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.