Session 5 & 6:

- **<u>Constructs in SQL</u>**

  The term "construct" is not a standard keyword or concept. There are several SQL constructs or elements in SQL that is being defined for performing various operation. The basic constructs of programming are **query, table, data types, operators, control structures, loops, functions** etc.

  There are few key elements or constructs in SQL –

  1. **SQL Queries:** Queries are SQL's workhorses. They retrieve data from a database based on specified criteria. It typically consists of SELECT, FROM, WHERE, and other clauses that define the conditions for data retrieval.
  2. **Statements**: SQL statements are versatile and essential. They control various aspects of database interactions, including transactions, program flow, connections, sessions, and diagnostics.
  3. **SQL Table:** SQL construct can refer to a database table, which is a structured set of data organized in rows and columns.
  4. **SQL View:** A SQL view is a virtual table based on the result of a SELECT query. It does not store the data itself but provides a way to represent a set of data stored in one or more tables.
  5. **SQL Procedure:** SQL procedures are a set of SQL statements that can be stored and executed on the database server. They are precompiled and stored for later use.
  6. **Clauses:** Clauses are integral components of queries and statements. They serve as building blocks, allowing you to refine and customize your SQL operations.
  7. **Expressions:** Expressions drive SQL statements. Expressions comprise symbols and operators. Expressions allow you to perform calculations, comparisons, and data transformations.
  8. **Predicates:** Predicates are essential for specifying conditions within SQL operations. Whether its filtering records with a WHERE clause or defining join conditions, predicates determine which data meets specific criteria

- <u>**Data collection**</u>

  Data collection refers to retrieving, storing, and managing data within a relational database. There are key SQL commands and queries can be used for data collection:

  1. **SELECT Statement:**

     The **SELECT** statement is used to retrieve data from one or more tables in a database.

     Syntax: **SELECT column1, column2,... FROM table_name WHERE condition;**

     Example: SELECT first_name, last_name FROM employees WHERE depart_id = 101;

  2. **INSERT Statement:** The **INSERT** statement is used to add new records to a table.

     Syntax: INSERT INTO table_name (column1, column2, ..) VALUES (value1, value2, ...);

     Example:INSERT INTO employees (first_name, last_name, department_id) VALUES ('John', 'Doe', 101);

  3. **UPDATE Statement:** The **UPDATE** statement is used to modify existing records in a table.

     Syntax: UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;

     Example: UPDATE employees SET department_id = 102 WHERE last_name = 'Doe';

  4. **DELETE Statement:** The **DELETE** statement is used to remove records from a table.

     Syntax: **DELETE FROM table_name WHERE condition;**

     Example: DELETE FROM employees WHERE last_name = 'Doe';

  5. **Aggregate Functions:** SQL provides aggregate functions like **COUNT**, **SUM**, **AVG**, **MIN**, and **MAX** for performing calculations on a set of values.

     Example: SELECT AVG(salary) FROM employees WHERE department_id = 101;

  6. **GROUP BY Clause:** The **GROUP BY** clause is used to group rows that have the same values in specified columns into summary rows.

     Example: SELECT department_id, AVG(salary) FROM employees GROUP BY department_id;

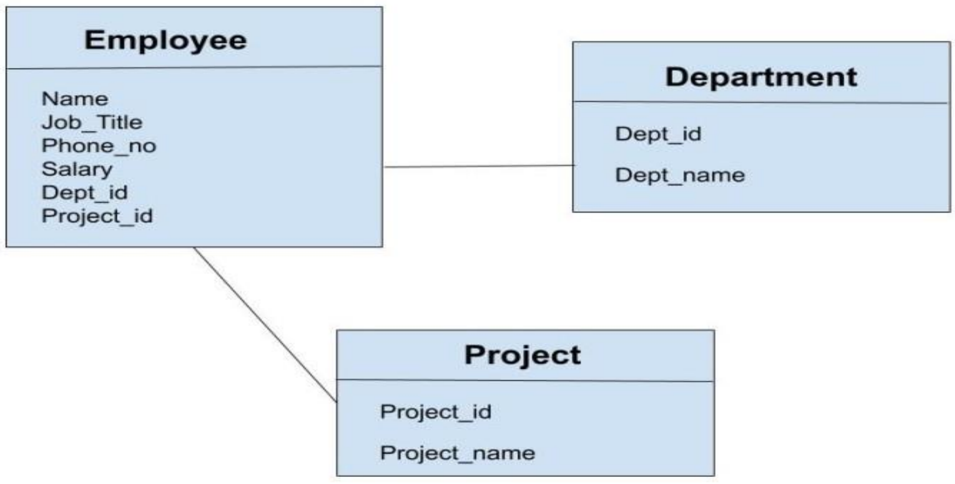
  7. **Joins:** Joins are used to combine rows from two or more tables based on a related column between them.

- **Designing Database Schema**

  A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities.

  A database schema refers to the logical and visual configuration of the entire relational database. The database objects are often grouped and displayed as tables, functions, and relations. A schema describes the organization and storage of data in a database and defines the relationship between various tables.

  A database schema is considered the "blueprint" of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data.



Schema

Types of database schemas

There are three type of database schema:

- **Conceptual schemas**: It gives view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements.
- **Logical database schemas**: It clearly define schema objects with information, such as table names, field names, entity relationships, and integrity constraints—i.e. any rules that govern the database.

- **Physical database schemas** provide the technical information that the logical database schema. it also includes the syntax that will be used to create these data structures within disk storage.

<u>Benefits of database schemas</u>

Database objects and schemas are critical to ensure efficiency in day-to-day company operations. If relational models are poorly organized and poorly documented, it will be harder to maintain, posing problems for both its users and the company.

<u>There are key benefits of database schemas</u>

- **Access and security:** Database schema design helps organize data into separate entities, making it easier to share a single schema within another database. Administrators can also control access through database permissions, adding another layer of security for more proprietary data.

  For example, a single schema may contain personally identifiable information (PII), which you would want to encrypt for privacy and security purposes.
- **Organization and communication:** Documentation of database schemas allow for more organization and better communication among internal stakeholders. Since it provides a common source of truth, it enables users to understand the logical constraints and methods of aggregation across tables.
- **Integrity:** This organization and communication also helps to ensure data validity. For example, it can help administrators manage normalization processes to avoid data duplication. It can also assist in monitoring compliance of the constraints in the schema's database design, enabling adherence to ACID properties (atomicity, consistency, isolation, durability).

Database Schema Designs:

There are many ways to structure a database, and we should use the best-suited schema design for creating our database because ineffective schema designs are difficult to manage & consume extra memory and resources. Schema design mostly depends on the application’s requirements. There are following key schema model as :

A. Flat Model

B. Hierarchical Model

C. Network Model

D. Relational Model

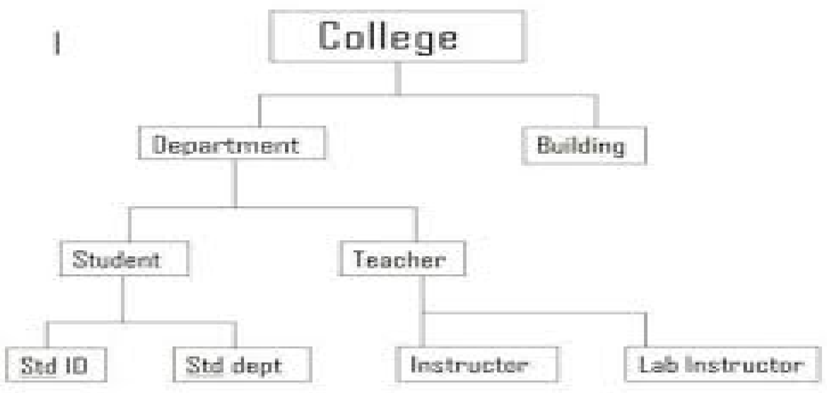
1.Flat Model

A flat model schema is a 2-D array in which every column contains the same type of data/information and the elements with rows are related to each other.

Sr.No	Name	Address
1	A	X
2	B	Y
3	C	Z

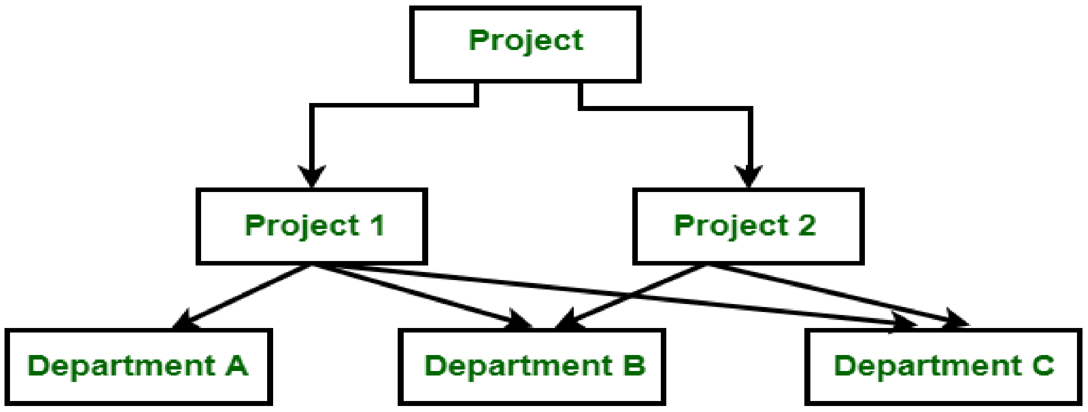
2.Hierarchical Model

Data is arranged using parent-child relationships and a tree-like structure in the Hierarchical Database Model. Because each record consists of several children and one parent, it can be used to illustrate **one-to-many relationships** in diagrams.



3.Network Model

The network model and the hierarchical model are quite similar with an important difference that is related to data relationships. The network model allows **many-to-many relationships.**



4.Relational Model

The relational model is mainly used for relational databases, where the data is stored as relations of the table. This relational model schema is better for object-oriented programming.

Physical Schema Vs Logical Schema

Physical Schema	**Logical Schema**
Physical schema describes the way of storage of data in the disk.	Logical schema provides the conceptual view that defines the relationship between the data entities.
Having Low level of abstraction.	Having a high level of abstraction.
The design of database is independent to any database management system.	The design of a database must work with a specific database management system or hardware platform.
Changes in Physical schema effects the logical schema	Any changes made in logical schema have minimal effect in the physical schema
Physical schema does not include attributes.	Logical schema includes attributes.
Physical schema contains the attributes and their data types.	Logical schema does not contain any attributes or data types.
Examples: Data definition language (DDL), storage structures, indexes.	**Examples:** Entity Relationship diagram, Unified Modelling Language, class diagram.

Advantages of Database Schema

- **Providing Consistency of data:** Database schema ensures the data consistency and prevents the duplicates.

- **Maintaining Scalability:** Well-designed database schema helps in maintaining addition of new tables in database along with that it helps in handling large amounts of data in growing tables.
- **Performance Improvement:** Database schema helps in faster data retrieval which is able to reduce operation time on the database tables.
- **Easy Maintenance:** Database schema helps in maintaining the entire database without affecting the rest of the database
- **Security of Data:** Database schema helps in storing the sensitive data and allows only authorized access to the database.

Topic: Normal Forms and ER Diagram

In database management systems (DBMS), normal forms are set of guidelines that help to ensure that the design of a database is efficient, organized, and free from data anomalies. There are set of guidelines, known as normal forms.

Normalization: Normalization is a process of organizing the data in database to avoid data redundancy, improve data consistency, insertion anomaly, update anomaly & deletion anomaly.

Why Do We Need Normalization?

Normalization is used to reduce data redundancy. It provides a method to remove the following anomalies from the database and bring it to a more consistent state:

A database anomaly is a flaw in the database that occurs because of poor planning and redundancy.

1. **Insertion anomalies**: This occurs when we are not able to insert data into a database because some attributes may be missing at the time of insertion.
2. **Updation anomalies:** This occurs when the same data items are repeated with the same values and are not linked to each other.
3. **Deletion anomalies:** This occurs when deleting one part of the data deletes the other necessary information from the database.

Advantages of Normal Form

- **Reduced data redundancy:** Normalization helps to eliminate duplicate data in tables, reducing the amount of storage space needed and improving database efficiency.

- **Improved data consistency:** Normalization ensures that data is stored in a consistent and organized manner, reducing the risk of data inconsistencies and errors.
- **Simplified database design:** Normalization provides guidelines for organizing tables and data relationships, making it easier to design and maintain a database.
- **Improved query performance:** Normalized tables are typically easier to search and retrieve data from, resulting in faster query performance.
- **Easier database maintenance:** Normalization reduces the complexity of a database by breaking it down into smaller, more manageable tables, making it easier to add, modify, and delete data.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

Types of Normal Forms:

1. **First Normal Form**
2. **Second Normal Form**
3. **Third Normal Form**
4. **BCNF**
5. **Fourth Normal Form**
6. **Fifth Normal Form**

1. **First Normal Form (1NF):** A relation will be 1NF if it contains an atomic value. i.e an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

   First normal form disallows the **multi-valued attribute**, composite attribute, and their combinations. Employee **Phone Number** is a multivalued characteristic in this case. This relationship is, therefore, not 1NF.

Employee Code	Employee Name	Employee Phone Number
101	Jolly	98765623,998234123
102	Rehan	76213908

After performing 1 NF now below relation is in 1 NF

Employee Code	Employee Name	Employee Phone Number
101	Jolly	998234123
101	Jolly	98765623
101	Jolly	89023467
102	Rehan	76213908
103	Snehi	98132452

2. Second Normal Form

A relation to be in the Second Normal Form-

- It should be in the First Normal form.
- And, it should not have Partial Dependency.

What is Partial Dependency?

When a table has a primary key that is made up of two or more columns, then all the columns (not included in the primary key) in that table should depend on the entire primary key and not on a part of it.

Student table:

student_id	student_name	branch
1	Akon	CSE
2	Bkon	Mechanical

Subject Table:

subject_id	subject_name
1	C Language
2	DSA
3	Operating System

table **Score**

student_id	subject_id	marks	teacher_name
1	1	70	Miss. C
1	2	82	Mr. D
2	1	65	Mr. Op

the primary key is **student_id + subject_id**, because both this information are required to select any row of data.

But in the **Score** table, we have a column **teacher_name**, which depends on the subject information or just the **subject_id**, so we *should not keep* that information in the **Score** table.

The column **teacher_name** should be in the **Subjects** table. And then the entire system will be Normalized as per the Second Normal Form.

Updated Subject table:

subject_id	**subject_name**	**teacher_name**
1	C Language	Miss. C
2	DSA	Mr. D
3	Operating System	Mr. Op

Updated Score table:

student_id	**subject_id**	**marks**
1	1	70
1	2	82
2	1	65

3. Third Normal Form

- A table is said to be in the Third Normal Form when,
- It satisfies the First Normal Form and the Second Normal form And, it doesn't have Transitive Dependency.

  Transitive Dependency?

  In a table when a column that acts as the primary key and other columns depends on this column. But what if a column that is not the primary key depends on another column that is also not a primary key or part of it?

The Result table is,

student_id	**subject_id**	**marks**	**exam_type**	**total_marks**
1	1	70	Theory	100
1	2	82	Theory	100
2	1	42	Practical	50

In the table above, the column exam_type depends on both student_id and subject_id, because,

- a student can be in the CSE branch or the Mechanical branch,
- and based on that they may have different exam types for different subjects.
- The CSE students may have both Practical and Theory for Compiler Design,
- whereas Mechanical branch students may only have Theory exams for Compiler Design.
- But the column total_marks just depend on the exam_type column.

4. **BCNF**

Boyce and Codd Normal Form is a higher version of the Third Normal Form.

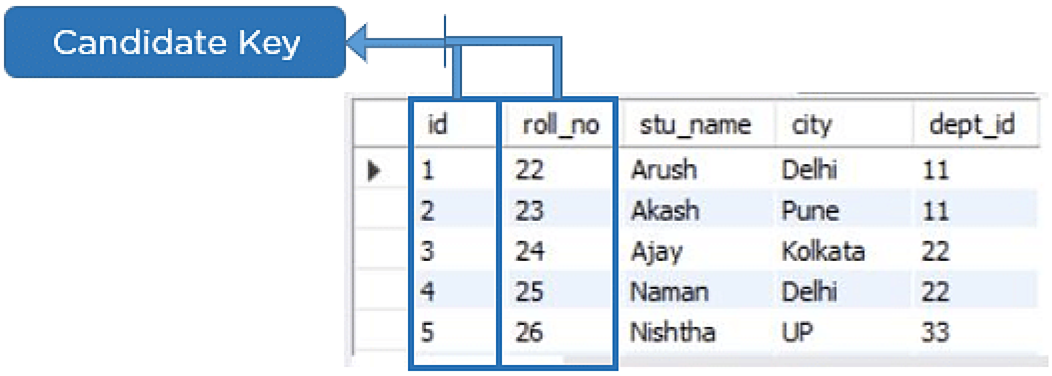
This form deals with a certain type of anomaly that is not handled by 3NF.

A 3NF table that does not have multiple overlapping candidate keys is said to be in BCNF. **For a table to be in BCNF:**

- R must be in the 3rd Normal Form
- and, for each functional dependency (X → Y), X should be a Super Key.

Candidate Key

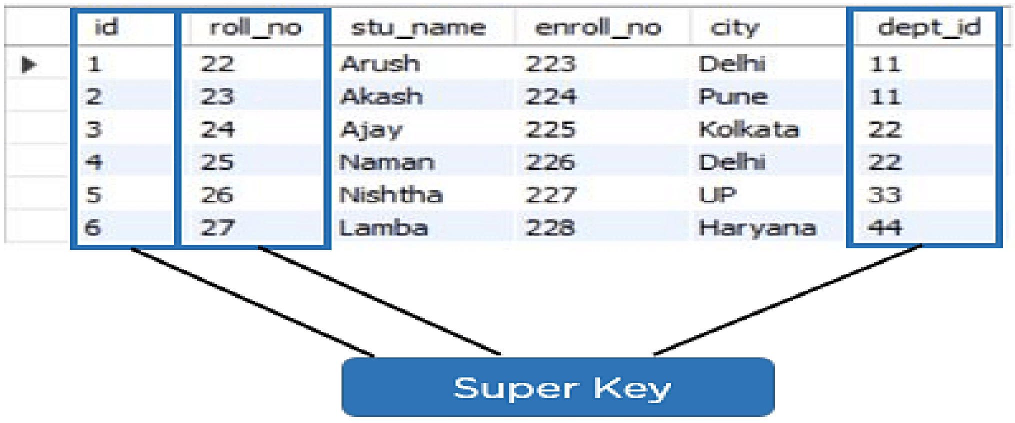
A candidate key is a set of one or more columns that can identify a record uniquely in a table, and YOU can use each candidate key as a Primary Key.



id	roll_no	stu_name	city	dept_id
1	22	Arush	Delhi	11
2	23	Akash	Pune	11
3	24	Ajay	Kolkata	22
4	25	Naman	Delhi	22
5	26	Nishtha	UP	33

Super Key

Super key is a set of over one key that can identify a record uniquely in a table, and the Primary Key is a subset of Super Key.



id	roll_no	stu_name	enroll_no	city	dept_id
1	22	Arush	223	Delhi	11
2	23	Akash	224	Pune	11
3	24	Ajay	225	Kolkata	22
4	25	Naman	226	Delhi	22
5	26	Nishtha	227	UP	33
6	27	Lamba	228	Haryana	44

5. Fourth Normal Form

A table is said to be in the Fourth Normal Form when,

- It is in the Boyce-Codd Normal Form.
- And, it doesn't have Multi-Valued Dependency.

Topic: **Relational DB modelling**

The relational model represents how data is stored in Relational Databases. A relational database consists of a collection of tables, each of which is assigned a unique name.

important Terminologies

- **Attribute:** Attributes are the properties that define an entity. e.g.; **ROLL_NO**, **NAME, ADDRESS**
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.
- **Tuple:** Each row in the relation is known as a tuple.

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is an insertion, deletion, or update in the database.
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The **STUDENT** relation defined above has degree 5.
- **Cardinality:** The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
- **Column:** The column represents the set of values for a particular attribute. The column **ROLL_NO** is extracted from the relation STUDENT.
- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Relation Key:** These are basically the keys that are used to identify the rows uniquely or also help in identifying tables.

Constraints in Relational Model

While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints. These constraints are checked before performing any operation (insertion, deletion, and updation) in the database. If there is a violation of any of the constraints, the operation will fail.

A.Domain Constraints

These are attribute-level constraints. An attribute can only take values that lie inside the domain range. e.g.; If a constraint AGE>0 is applied to STUDENT relation, inserting a negative value of AGE will result in failure.

B.Key Integrity

Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called keys. e.g.; ROLL_NO in STUDENT is key. No two students can have the same roll number. So a key has two properties:

- It should be unique for all tuples.
- It can’t have NULL values.

C. Relation Key: These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types.

- Primary Key
- Candidate Key
- Super Key
- Foreign Key
- Alternate Key
- Composite Key

- **Stored Procedures**

  A stored procedure is a group of one or more pre-compiled SQL statements into a logical unit. A stored procedure is used to retrieve data, modify data, and delete data in database table.

  the features of stored procedure in SQL Server:

  - **Reduced Traffic:** A stored procedure reduces network traffic between the application and the database server, resulting in increased performance. It is because instead of sending several SQL statements, the application only needs to send the name of the stored procedure and its parameters.

- **Stronger Security:** The procedure is always secure because it manages which processes and activities we can perform. It removes the need for permissions to be granted at the database object level and simplifies the security layers.
- **Reusable:** Stored procedures are reusable. It reduces code inconsistency, prevents unnecessary rewrites of the same code, and makes the code transparent to all applications or users.
- **Easy Maintenance:** The procedures are easier to maintain without restarting or deploying the application.
- **Improved Performance:** Stored Procedure increases the application performance. Once we create the stored procedures and compile them the first time, it creates an execution plan reused for subsequent executions. The procedure is usually processed quicker because the query processor does not have to create a new plan.

Types of Stored Procedures

- **User-defined Stored Procedures**

  Database developers or database administrators build user-defined stored procedures. These procedures provide one or more SQL statements for selecting, updating, or removing data from database tables. A stored procedure specified by the user accepts input parameters and returns output parameters. DDL and DML commands are used together in a user-defined procedure.

  This procedure into two types:

  - **T-SQL Stored Procedures:** Transact-SQL procedures are one of the most popular types of SQL Server procedures. It takes parameters and returns them. These procedures handle INSERT, UPDATE, and DELETE statements with or without parameters and output row data.
  - **CLR Stored Procedures:** The SQL Server procedures are a group of SQL commands, and the CLR indicates the common language runtime. CLR stored procedures are made up of the CLR and a stored procedure, which is written in a CLR-based language like VB.NET or C#. CLR procedures are .Net objects that run in the SQL Server database's memory.

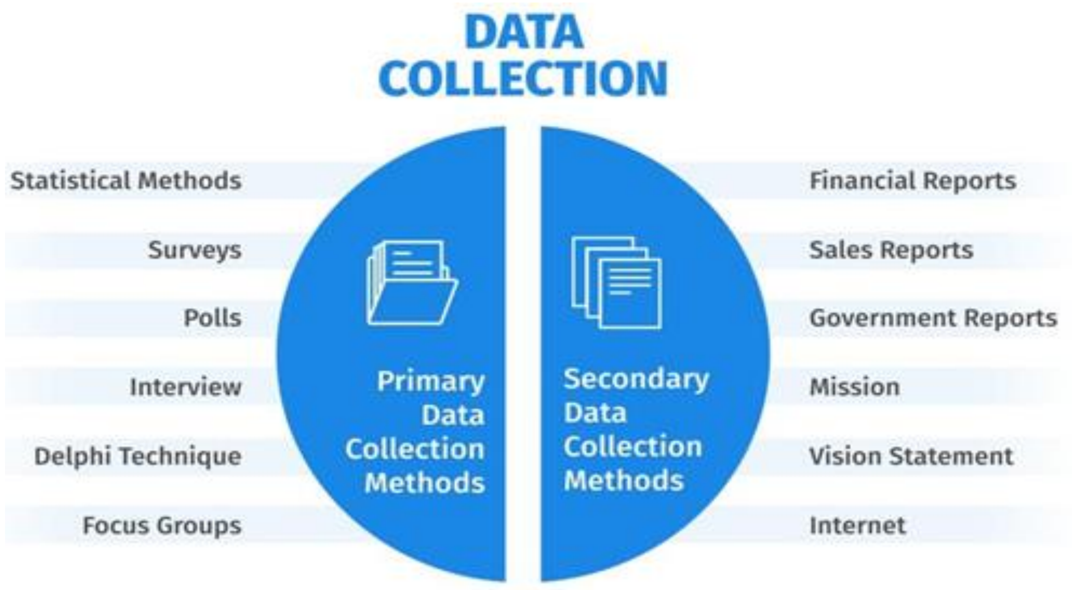
System Stored Procedures

The server's administrative tasks depend primarily on system stored procedures. When SQL Server is installed, it creates system procedures. The system stored procedures prevent the administrator from querying or modifying the system and database catalogue tables directly. Developers often ignore system stored procedures. the stored procedure parameters:

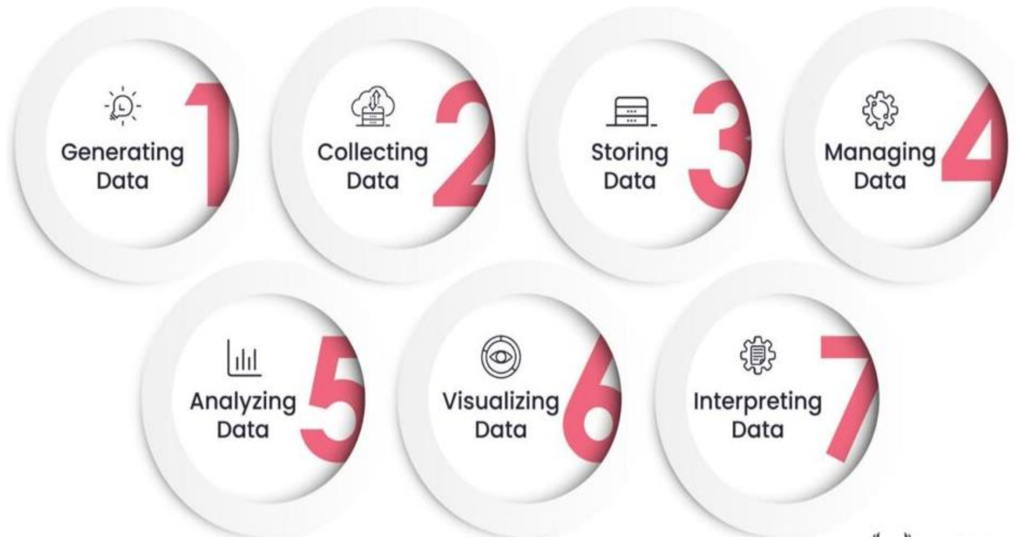
- **Schema_name:** It is the name of your database or schema. By default, a procedure is associated with the current database, but we can also create it into another database by specifying the DB name.
- **Procedure_Name:** It represents the name of your stored procedure that should be meaningful names so that you can identify them quickly. It cannot be the system reserved keywords.
- **Parameter_Name:** It represents the number of parameters. It may be zero or more based upon the user requirements. We must ensure that we used the appropriate data type. **For example**, @Name VARCHAR (50).

Topic: Gathering Data in Systematic fashion

Data collection is a process of gathering information from all the relevant sources. Data is a collection of facts, figures, objects, symbols, and events gathered from different sources. Organizations collect data with various data collection methods to make better decisions. Without data, it would be difficult for organizations to make appropriate decisions, so data is collected from different audiences at various points in time.



Data can be gathered in a systematic fashion as following cycle:


Generating Data
1
Collecting Data
2
Storing Data
3
Managing Data
4
Analyzing Data
5
Visualizing Data
6
Interpreting Data
7