Srinivasa Ramanujan Institute of Technology
(AUTONOMOUS)
Rotarypuram Village, B K Samudram Mandal, Ananthapuramu - 515 701

Continuous Alternate Assessment –I

DataBase Management Systems

(23CSE302)

submitted by

T.Reshma Taj

234G1A05D1

II B-Tech I Semester

Regulation: **SRIT-R23**

Department of Computer Science & Engineering

Details of CAA-I:

Date of Issue	:	12/9/24				
Last Date to Submit	:	20/9/24				
Date of Submission	:	18/9/24				
Question Numbers	:	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
Marks Obtained	:	2	2	2	2	2
Total Marks	:	10		**Maximum Marks: 10**		
Signature of the Faculty	:					

(2024-2025)

Unit - I

Explain how the Centralized and Client - Server architecture for database systems differ. What are the advantages and disadvantages of each architecture in terms of Scalability and Performance?

Centralized Architecture for Database Systems:

1. Structure: In a centralized architecture, the entire database system resides on a single server or location. All clients or users access this single system over a network.

2. Performance: Since all database operations occur on a server, Performance can be good as long as the server has sufficient resources to handle the load.

Bottlenecks can arise if too many users access the server simultaneously or if the system is processing large amounts of data, leading to slower response times.

3. Scalability:

→ Centralized systems have limited scalability. As the number of users or data volume increases, the servers Performance can degrade significantly because of limited Processing Power, memory, and disk space.

→ Scaling the system generally requires upgrading

the hardware of the single server, which can be expensive and has physical limits.

Advantages:

→ Easier to manage and maintain since all data is located in one place.

→ Security can be managed more effectively with a single point of control.

→ Simplified backup and recovery process.

Disadvantages:

→ Single point of failure: If the central server goes down, the entire system becomes inaccessible.

→ limited scalability: Hard to accomodate large increases in users or data

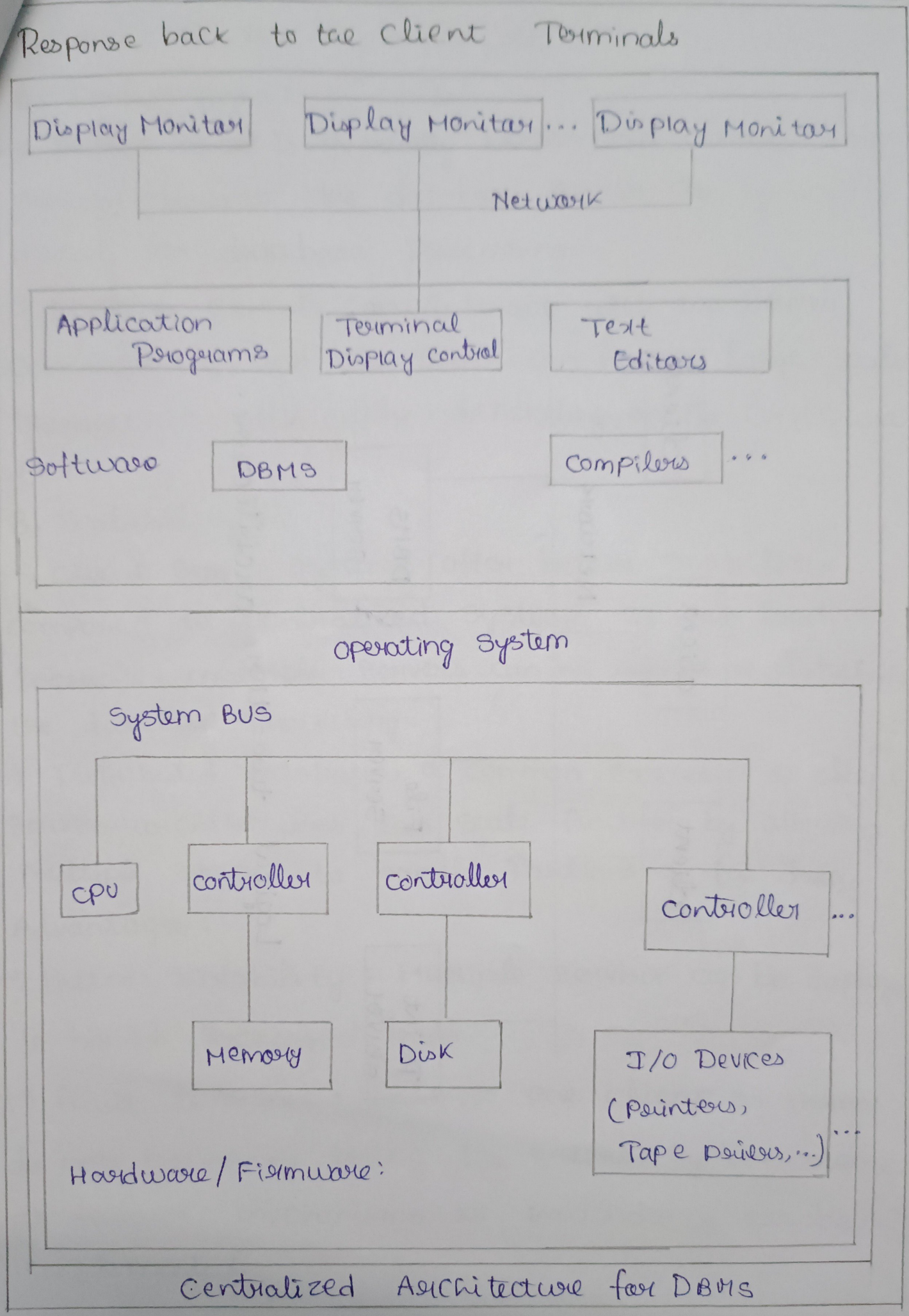
→ High dependency on the central server, making it a bottleneck for performance.

Ex: Mainframe - based database systems used in earlier computing eras.

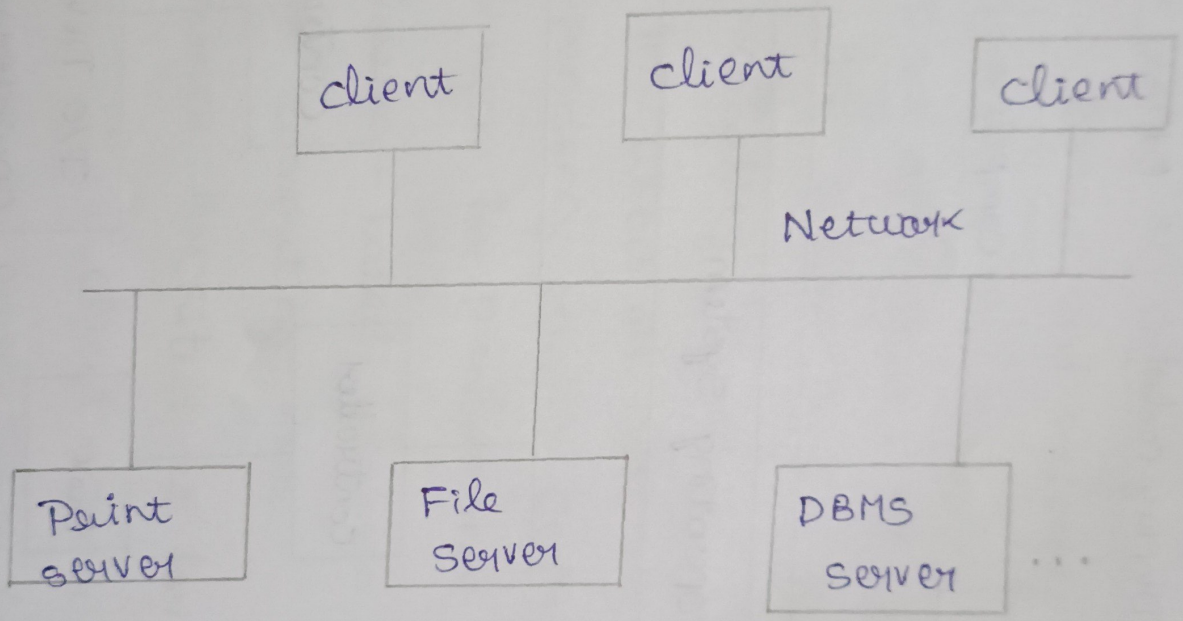
Client - Server Architecture for Database Systems:

1. Structure: In a client - server architecture, the data base resides on a server (or Multiple servers) while clients (end users) run applications that request services from the data base.

→ The server handles requests, process data,



Centralized Architecture for DBMS



Logical two-tier architecture.

returns the results to the clients.

2. Performance:

→ Performance is generally better in a client-server system because the database server can be optimized for database processing.

→ Multiple clients can interact with the server concurrently, and the server can handle large-scale transactions efficiently, depending on its configuration.

3. Scalability:

→ client server systems offer better scability compared to centralized systems If the load increase, multiple servers can be added to distribute the database processing.

→ Distributed databases, a common extension of client server architectures, can scale further by allowing multiple servers to handle Portions of the data.

Advantages:

→ Better Scalability: Multiple servers can be deployed to handle increased leads.

→ fault tolerance: failures one client or server do not necessarily bring the entire system down, as failover mechanisms or redundancy can be implemented.

→ flexibility: Different clients can interacts with the database, allowing a diverse range of users and applications.

Disadvantages:

→ more complex to mange and maintain, especially as the system grows.

→ Security risks: Communication between clients and servers over a network can introduce vulnerabilities

→ More complex backup and recovery process because of the distributed nature of the data.

Ex: Modern database systems like MySQL, Postgre SQL, and oracle operate on client-server Models.

(2) What are the different types of attributes in an ER model? Provide examples of simple, composite, derived, and multivalued attributes.

In a Database Management system (DBMS), an attribute is a property or characteristic of an entity that is used to describe an entity. essentially, it is a column in a table that holds data values. An entity many contain any no. of attributes. one of attribute is considered as the primary key, in an entity-Relation model.

attributes are represented in an ellipse (or) elliptical shape.

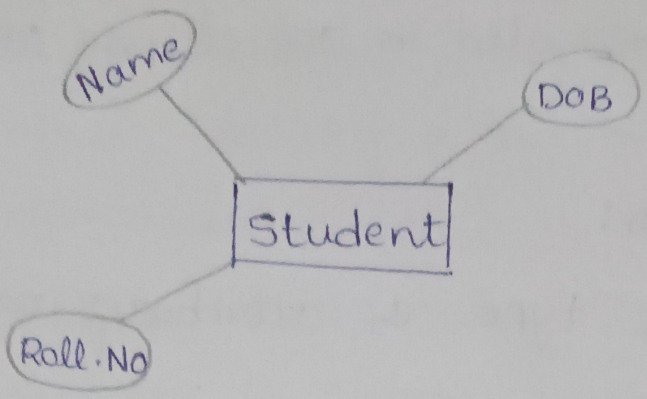
Types of Attributes:

There are different types of attributes as follows:

→ Simple Attribute
→ Composite Attribute
→ Single - Valued Attribute
→ Multi - Valued Attribute
→ Derived Attribute
→ Complex Attribute
→ Stored Attribute
→ Key Attribute
→ Null Attribute
→ Descriptive Attribute

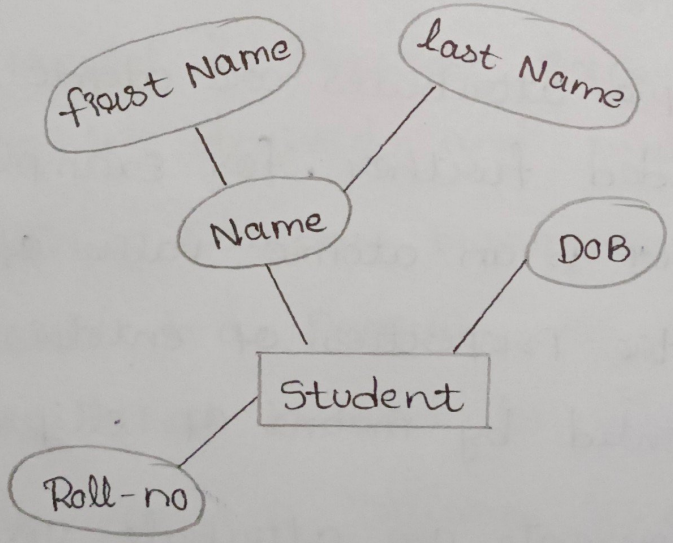
Simple Attribute: Simple attributes are atomic values which cannot be divided further. for example, a students Phone number is an atomic value of 10 digits. Attributes are the properties of entities. Attributes are represented by means of ellipses.

Every ellipse represents one attribute and is directly connected to its entity (rectangle).



Composite attribute: Composite attributes are made by more than one simple attribute. For example, a Students complete name may have first_name & last_name.

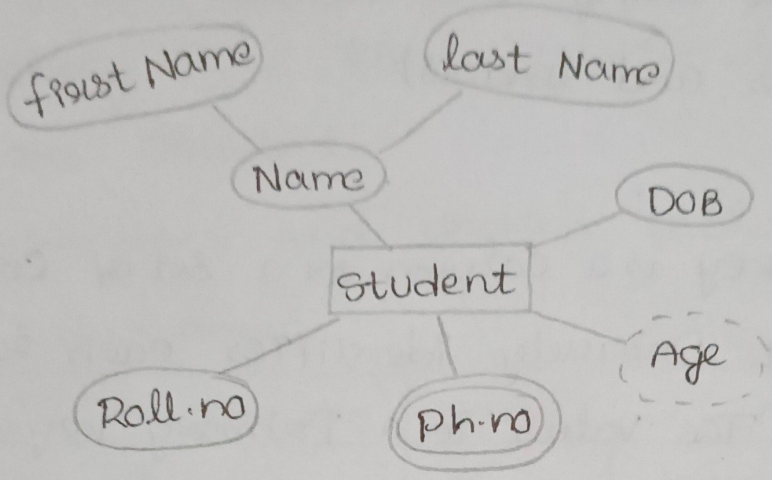
If the attributes are composite, they are further derived in a tree like structure. Every node is then connected to its attribute. That is, Composite attributes are represented by ellipses that are connected with an ellipse



Derived Attribute: Derived attributes are the attributes that do not exist in the Physical database but - their values are derived from other attributes present in the database. for ex avg-salary

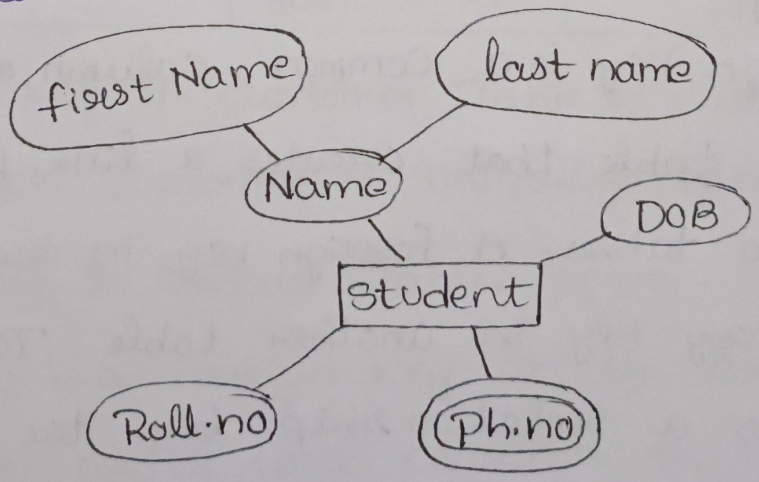
in a department should not be saved directly in the database, instead it can be derived for another example, avg can be derived from date of Birth

Derived attributes are depicted by dashed ellipse



Multi-value attribute: Multi-Value attributes may contain more than one values. for example, a Person can have more than one phone number, email-address, etc.

multi-valued attributes are represented by double ellipse.



Unit - II

③ Explain the concept of a Primary key and a foreign key. How do these constraints ensure data Integrity in relational databases? Illustrate with an example using two related tables (e.g, Customers and orders).

Primary key:

A Primary key is a column or a set of Colums in a table that Uniquely identifies each record in that table. The Value in a Primary key must be:

- Unique: No two records be can have the same value for the Primary key.
- Non-null: Every record must have a value for the Primary key (it cannot be empty)

Foreign key:

A foreign key is a common, column or set of colums in a table that creates a link between the data in two tables. A foreign key in one table Points to the Primary key in another table. The foreign key enforces a relationship b/w the two tables by ensuring that a value in the foreign key must already exist as a value in the Primary ~~Set~~ key of the referenced table.

Ensuring Data - Integrity :

* Primary key ensures that each record in a table is unique, preventing duplicate entries and enforcing a non-null constraints.
* Foreign key ensures that referential integrity by making sure that relationships between tables are consistent.

Ex: Customers and orders tables

Table : Customers

Customer ID (Primary key)	Name	Email
1	Arun	Arun@gmail.com
2	Arjun	Arjun@gmail.com

Table : orders

Order ID (Primary key)	Order Date	Customer ID (Foreign key)
101	2024-9-13	1
102	2024-9-13	2

* Primary key in Customer Table : Customer ID ensures each Customer is uniquely identified.
* Foreign key in orders Table:

Customer ID is a foreign key in the orders table, linking each order to a consumer in the Customers table. This ensures that every order must be placed by a valid Customer.

④ Explain the Concepts of domain, attribute, tuple and relation in the context of a relational database. Illustrate these Concepts with a simple example from a real-world database.

Domain :⇒ A domain is a set of possible values that an attribute can take. A common method top Specifying a domain is to specify a datatype from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in Interpreting its values.

Ex: In a "Students" table, if the age attribute only accepts values between 1 and 120, then the domain of Age is Integers in the range [1, 120].

Attribute: A relation Schema "R" denoted by $R(A_1, A_2, \ldots A_n)$ is made up of a relation name R and a list of attributes $A_1, A_2, \ldots A_n$. Each attribute A_i is the name of a role played by some Domain D in in the relation Schema R. D is Called Domain of A_i and is denoted by dom(A).

Ex:- In a "Students" table, Name, Age and Student ID are attributes each Column represents a Specific Characteristic of a Student.

Tuple: A tuple is a single row in a relational table. It represents a single record or instance of the entity being modeled. each tuple consists of values for the attributes.

Ex:- In the "Students" table, a tuple could be a specific student record like: ('Rani', 18, '5152')
Here 'Rani' is the value for the Name attribute, 18 for Age, and '5152' for Student ID.

Relation: A (named) set of tuples all of the same form (i.e, having the same set of attributes). The term table is a loose synonym.

Ex: The "Students" table is a relation that contains multiple tuples.

Example from a Real-World Data base:

Student table:

Student ID	Name	Age	Major
5152	Rani	18	Physics
5245	Rahul	20	Chemistry
5352	Rahim	21	Computer Science

→ The domain of Age is Integers (eg. [1, 120]), the domain of Name is Strings, and domain of major is Strings.
→ The table has 4 attributes: Student ID, Name, Age, and Major.

→ The tuple ('5152', 'Rani', 18, 'physics') is one record in the table.

→ Relation: The "Students" table as a whole is relation that consists of the set of all tuples and their corresponding attributes.

UNIT - III

⑤ Explain the use of Date and Time functions in SQL. Write an SQL query to retrieve all records from a Bookings table where the booking data is within the last 30 days.

Date and Time functions in SQL:

Date and Time functions in SQL are used to manipulate and extract information from date & time data types. common operations include retrieving the current date, calculating differences b/w dates, formatting dates and extracting specific parts of dates (like the year or month).

Some commonly used Date & Time functions are:

- Now(): Returns the current Date and time
- CURDATE(): Returns the current date (without time)
- DATE(): Extract the date part from a DATETIME or TIMESTAMP value.

- DATE DIFF(): Returns the difference b/w two dates
- DATE ADD() (or) ADDDATE(): Add a specific interval (eg: days, months) to a date.
- DATE - SUB(): Subtract a specific interval from date
- EXTRACT(): Extract a specific part of the date, like YEAR, MONTH or DAY.

SQL query to retrieve records from the last 30 days:

To retrieve all records from a Bookings table where the booking date is within the last 30 days, you can use the Now(), along with DATE-SUB() (or) (CURDATE() for the Current DATE).

Heres an example query:

```
SELECT *
from Bookings
WHERE Booking Date >= DATE-SUB (CURDATE(), INTERVAL
                                30 DAY);
```

If the Booking Date includes time:

```
SELECT*
from Bookings
WHERE Booking Date >= DATE-SUB( Now(), INTERVAL
                                30 DAY);
```