1. What is Database?

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. It is also the collection of schemas, tables, queries, views, etc.

2. What is DBMS?

DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

3. What is RDBMS? How is it different from DBMS?

RDBMS is a database system software that manages and maintains data in the tabular format. The main difference between DBMS and RDBMS is that RDBMS stores data as tables and DBMS stores data as a file.

DBMS	RDBMS
DBMS stores data as file.	RDBMS stores data in a tabular form.
Normalization is not present in DBMS.	**Normalization is** present in RDBMS.
DBMS does **not apply any security** with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Automicity, Consistency, Isolation and Durability) property.
DBMS uses file system to store data, so there will be **no relation between the tables**.	In RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
DBMS **does not support distributed database**. A distributed database is a database that can be stored at different locations.	RDBMS **support distributed database**.
DBMS cannot store large quantities of data.	RDBMS allows users to store a large set of data.
Data fetching is slower for the large amount of data.	Data fetching is fast because of relational approach

Database Management System can only support a single user.	RDBMS allows access to multiple users to the databases.
Examples: XML, File System, Registry etc.	Examples: MySQL, SQL Server, Oracle, Microsoft Access etc.

4. What is SQL?

SQL is a language used to interact with the database, i.e to create a database, to create a table in the database, to retrieve data or update a table in the database, etc. SQL is an ANSI(American National Standards Institute) standard. Using SQL, we can do many things. For example – we can execute queries, we can insert records into a table, we can update records, we can create a database, we can create a table, we can delete a table, etc.

SQL stands for **Structured Query Language** whose main purpose is to interact with the relational databases in the form of inserting and updating/modifying the data in the database.

5. What is the difference between SQL and MySQL?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

MYSQL	**SQL**
MySQL used for data handling, storing, deleting, and updating the data in tabular form.	It is used to query and update the database
MySql is RDBMS that used 'SQL' language to query the database.	SQL is a query language that manage RDBMS.
Allows data handling, modifying, storing, deleting in a tabular format.	Purpose is to query and operate database system.
MySQL is less secure than SQL, as it allows third-party processors to manipulate data files during execution.	SQL server is much more secure than the MySQL server. In SQL, external processes (like third-party apps) cannot access or manipulate the data directly.
It is available only in the English language	It is available in many different language

6. Advantages of DBMS

- Data is stored in a structured way and hence redundancy is controlled.
- Provides backup and recovery of the data when required.
- It provides multiple user interfaces.

7. What do you understand by Data Redundancy?

Duplication of data in the database is known as data redundancy

8. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

Example**:**

Table: Employee.
Field: Emp ID, Emp Name, Date of Birth.
Data: 201456, David, 11/15/1960

9. List the different types of relationships in SQL.

One-to-one: One table has a relationship with another table having the similar kind of column. Each primary key relates to only one or no record in the related table.

One-to-many: One table has a relationship with another table that has primary and foreign key relations. The primary key table contains only one record that relates to none, one or many records in the related table.

Many-to-many: Each record in both the tables can relate to many numbers of records in another table.

10. Explain Normalization and De-Normalization.

Normalization is the process of removing redundant data from the database by splitting the table in a well-defined manner in order to maintain data integrity. This process saves much of the storage space.

De-normalization is the process of adding up redundant data on the table in order to speed up the complex queries and thus achieve better performance.

11. What are the different types of Normalization?

- **First Normal Form (1NF):** A relation is said to be in 1NF only when all the entities of the table contain unique or atomic values.
- **Second Normal Form (2NF):** A relation is said to be in 2NF only if it is in 1NF and all the non-key attribute of the table is fully dependent on the primary key.
- **Third Normal Form (3NF):** A relation is said to be in 3NF only if it is in 2NF and every non-key attribute of the table is not transitively dependent on the primary key

12. What is BCNF?

BCNF is the Boyce Code Normal form. It is the higher version of 3Nf which does not have any multiple overlapping candidate keys.

13. How many SQL statements are used? Define them.

SQL statements are basically divided into three categories, DDL, DML, and DCL.

1) **Data Definition Language (DDL)** commands are used to define the structure that holds the data. These commands are auto-committed i.e. changes done by the DDL commands on the database are saved permanently.

   Five types of DDL commands in SQL are:

a) CREATE

CREATE statements is used to define the database structure schema

Syntax: CREATE DATABASE db_name;
Example: CREATE DATABASE Company;

Syntax: CREATE TABLE table_name(
column1 data_type(size),
column2 data_type(size),
column3 data_type(size),
column4 data_type(size),
…..
);
Example: CREATE TABLE Employee(Emp_Name VARCHAR2(20),
DOB DATE, Mobile INT(10), Email VARCHAR2(20));

b) DROP

Drops commands remove tables and databases from RDBMS.
It is used to drop the whole table. With the help of the “DROP” command we can drop (delete) the whole structure in one go.

Syntax: DROP TABLE table_name;
Example: DROP TABLE Employee;

Example: DROP DATABASE db_name;
DROP DATABASE Company;

c) ALTER

ALTER Command: In an existing table, this command is used to add, delete/drop, or edit columns. It can also be used to create and remove constraints from a table that already exists.

Syntax: ALTER TABLE table_name ADD column_name COLUMN-definition;
Example: ALTER TABLE Employee ADD Address VARCHAR2(20);

d) TRUNCATE

Truncate is used to remove all the records from a table. It deletes all the records from an existing table **but not the table itself.**

Syntax: TRUNCATE TABLE table_name;
Example: TRUNCATE TABLE Employee;

The above command will delete the data from the ‘Employee’ table but not the table.

2) **Data Manipulation Language (DML)** commands are used to manipulate the data of the database. These commands are not auto-committed and can be rolled back.

a) INSERT

It is used to insert data into a table’s row.

Syntax: INSERT INTO TABLE_NAME (col1, col2, col3,…. col N)
VALUES (value1, value2, value3, …. valueN);

Or
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3, …. valueN);

Example: INSERT INTO Employee(Emp_Name, DOB, Mobile, Email) VALUES('Joe', '1995-02-16', 7812865845, 'joe@gmail.com');

b) UPDATE

This command is used to update or modify the value of a column in the table.

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

Example: UPDATE Employee SET Mobile=7840846168 WHERE Emp_Name='Amit';

c) DELETE

This command is used to remove one or more rows from a table. It always used with WHERE clause.

Syntax: DELETE FROM table_name [WHERE condition];

Example: DELETE FROM Employee WHERE Emp_Name='Joe';

3) **Data Control Language (DCL)** is a query language that allows users to retrieve and edit data held in databases. The types of Data Controlling Language commands include Grant and Revoke

a) **Grant Command:** It is used to provide privileges/permissions to modify and retrieve database objects like tables, views, sequences, indexes, and synonyms.

Syntax: GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

Example: GRANT INSERT, SELECT on accounts to Alex

Using this command, Alex has been granted permissions on accounts database objects like he can query or insert into accounts

b) Revoke

It is useful to back permissions from the user.

Syntax: REVOKE privilege_nameON object_nameFROM {user_name |PUBLIC |role_name}

Example: REVOKE SELECT, UPDATE ON student FROM BCA, MCA;

Using this command, the permissions of BCA,MCA like update on student database objects has been removed.

4) **Transaction Control Language(TCL)** commands are used to manage transactions in the database.

a) **COMMIT** command

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent

Please note, that once we have done a COMMIT, we cannot undo it unless it is rolled back.

Syntax: COMMIT;

Example: UPDATE Employee SET DOB='1995-02-17' WHERE Emp_Name='Joe';
COMMIT;

b) **ROLLBACK command**

This command is used to restore the database to its original state since the last command that was committed.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

Syntax: ROLLBACK;

Example: UPDATE Employee SET DOB='1995-02-17' WHERE Emp_Name='Joe';
ROLLBACK;

c) **SAVEPOINT command**

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Syntax: SAVEPOINT SavepointName;

Example: SAVEPOINT S1; //savepoint created
DELETE FROM Employee WHERE Emp_Name = 'Joe'; //deleted
SAVEPOINT S2; //Savepoint created.

5) What is DQL?

Data Query Language (DQL) is used to fetch the data from the database. It uses only one command:

a) SELECT:

This command helps you to select the attribute based on the condition described by the WHERE clause.

Syntax:
SELECT Column_Name
FROM TABLES
WHERE conditions;

Example:
SELECT FirstName

```
FROM Student
WHERE RollNo > 15;
```

14. What is a Primary Key?

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key.

Syntax**:**

```
CREATE TABLE Persons (
 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,
 PRIMARY KEY (ID)
);
```

15. What is a UNIQUE constraint?

A UNIQUE constraint ensures that all values in a column are different.
A PRIMARY KEY constraint automatically has a UNIQUE constraint.

Syntax**:**

```
CREATE TABLE Persons (
 ID int NOT NULL UNIQUE,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int
);
```

16. 8. What is a foreign key?

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Syntax:

```
CREATE TABLE Student (
id int NOT NULL Auto_Increment, name Varchar(50) NOT NULL, age int NOT
NULL,
City Varchar(10) NOT NULL, PRIMARY KEY(id),
FOREIGN KEY(City) REFERENCES city(cid)
);
```

In Table1 Table2_name Table2_column

If table already create

```
ALTER TABLE table_name ADD FOREIGN KEY(city)
REFERENCES city(cid);
```

17. JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

There are four different types of JOINs in SQL:

1. Inner Join

The INNER JOIN keyword selects records that have matching values in both tables.

Note: We can also write JOIN in place of INNER JOIN. It gives the same output.

Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name
;
```

Foreign Key Primary Key

Example:

i) `Select * from personal INNER JOIN city ON personal.city=city.cid;`

ii) Select * from personal p INNER JOIN city c ON p.city = c.cid;

Give Condition

```
Select p.id, p.name, p.percentage, p.age, p.gender, c.cityname
FROM personal p
INNER JOIN city c ON p.city=c.cid
WHERE c.cityname = “Agra”
ORDER BY p.name;
```

2. LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

```
Select * personal p LEFT JOIN city c ON p.city = c.cid;
```

Give Condition

```
Select p.id, p.name, p.percentage, p.age, p.gender,
c.cityname
 FROM personal p LEFT JOIN city c ON p.city=c.cid
WHERE gender=”M” ORDER BY p.name;
```

3. RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

```
Select p.id, p.name, p.percentage, p.age, p.gender, c.cityname
```

```
FROM personal p RIGHT JOIN city c ON p.city = c.cid;
```

4. FULL JOIN

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.
NOTE: FULL OUTER JOIN and FULL JOIN are the same.

Syntax**:**

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

5. SELF JOIN

A self join is a regular join, but the table is joined with itself.

Syntax:

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

6. CROSS JOIN

The CROSS JOIN keyword returns all records from both tables (table1 and table2).
Cross join can be defined as a cartesian product of the two tables included in the join. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

Syntax:

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

Example:

```
Select * FROM personal CROSS JOIN city;
```
or

```
Select p.id, p.name, AS name, c.cityname AS city FROM
personal p CROSS JOIN city c;
```

JOIN MULTIPLE TABLE

Syntax:

```
SELECT Columns_Name FROM table1
INNER JOIN table2 ON table1.Column_Name =
table2.Column_Name
INNER JOIN table3 ON table1.Column_Name =
table3.Column_Name;
```

Example:

```
Select * FROM personal p INNER JOIN
city c ON p.city = c.cid INNER JOIN
courses cr ON p.courses = cr.course_id;
```

18. Group By Clause

GROUP BY Clause is used to collect data from multiple records and group the result by one or more column.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Example:

```
Select city, COUNT(city)
FROM personal
GROUP BY city;
```

Group By clause with JOIN

```
Select columns FROM table1 INNER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition
GROUP BY column_name(s);
```

19. Having Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
Having Clause is always used with GROUP BY clause.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Example:

```
Select c.cityname, count(p.city) AS Total FROM personal p
INNER JOIN city c ON p.city = c.cid
GROUP BY city
HAVING COUNT(p.city)>3;
```

20. What is a Cursor?

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML(Data Manipulation Language) operations on Table by User. Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors.

a) **ImplicitCursors:**
Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

b) **Explicit Cursors :**
Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.
Syntax : (Declare Explicit Cursor Object.)

```
DECLARE cursor_name CURSOR FOR SELECT * FROM
table_name
```

Example:

```
DECLARE s1 CURSOR FOR SELECT * FROM studDetails
```

21. Sub-Query

Sub-query is basically the query which is included inside some other query and can also be called as an inner query which is found inside the outer query.

Syntax:

```
SELECT column FROM table1
WHERE
Column = (SELECT column FROM table2 WHERE condition)
```

(we can also used INSERT, UPDATE, DELETE in place of SELECT command)

Example:

```
SELECT name FROM personal WHERE
courses = (SELECT course_id FROM
courses WHERE course_name = “MCA”);
```

Correlated Subquery

A Subquery is also known as a nested query i.e. a query written inside some query. When a Subquery is executed for each of the rows of the outer query then it is termed as a Correlated Subquery.
Example:

```
SELECT * from EMP WHERE ‘RIYA’ IN (SELECT Name from
DEPT WHERE EMP.EMPID=DEPT.EMPID);
```

22. Differences between DROP, TRUNCATE and DELETE commands?

The DELETE command deletes one or more existing records from the table in the database.	The DROP Command drops the complete table from the database.	The TRUNCATE Command deletes all the rows from the existing table, leaving the row with the column names.
DELETE FROM table_name WHERE condition;	DROP TABLE table_name;	TRUNCATE TABLE table_name;

23. Difference between UNION and UNION ALL?

UNION and UNION ALL are used to join the data from 2 or more tables but UNION removes duplicate rows and picks the rows which are distinct after combining the data from the tables whereas UNION ALL does not remove the duplicate rows, it just picks all the data from the tables.

Union | Union All

Syntax:

```
SELECT column_name(s) FROM table1
UNION / UNION ALL
SELECT column_name(s) FROM table2;
```

Example:

```
SELECT * from Student
        UNION / UNION ALL
SELECT * from Lecturer;
```

(when column name of both columns are same then we use star(*) otherwise not)

24. ACID Property

Atomicity:

Atomicity means that an entire transaction either takes place all at once or it doesn't occur at all. It means that there's no midway.

(the entire transaction take place at once or doesn't happen at all)

Consistency:

Consistency means that we have to maintain the integrity constraints so that any given database stays consistent both before and after a transaction

(the database must be consistent before and after the transaction)

Isolation:

Isolation is a property that guarantees the individuality of each transaction, and prevents them from being affected from other

transactions. It ensures that transactions are securely and independently processed at the same time without interference, but it does not ensure the order of transactions.
(multiple transactions occur independently without interference)

Durability:

The durability property states that once the execution of a transaction is completed, the modifications and updates on the database gets written on and stored in the disk. These persist even after the occurrence of a system failure. Such updates become permanent and get stored in non-volatile memory. Thus, the effects of this transaction are never lost.
(the changes of a successful transaction occurs even if the system failure occurs)

Uses of ACID Properties

It ensures consistency in a way that every transaction acts as a group of operations acting as single units, produces consistent results, operates in an isolated manner from all the other operations, and makes durably stored updates. These ensure the integrity of data in any given database.

25. Explain Entity, Entity Type, and Entity Set in DBMS?

- **Entity** is an object, place or thing which has its independent existence in the real world and about which data can be stored in a database. For Example, any person, book, etc.
- **Entity Type** is a collection of entities that have the same attributes. For Example, the STUDENT table contains rows in which each row is an entity holding the attributes like name, age, and id of the students, hence STUDENT is an Entity Type which holds the entities having the same attributes.
- **Entity Set** is a collection of entities of the same type. For Example, A collection of the employees of a firm.

26. E-R model

E-R model is known as an **Entity-Relationship model**. It is based on the concept of the Entities and the relationship that exists among these entities.

27. Stored Procedure

Stored Procedure is a group of SQL statements in the form of a function that has some unique name and is stored in relational database management systems(RDBMS) and can be accessed whenever required.

Syntax:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Example: (Execute Procedure)

```
EXEC procedure_name;
```

28. Trigger

A trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

29. INDEX

It is used to fast search the data from the table

Syntax:

```
CREATE INDEX index_name ON table_name(column1,
column2, column3, …);
```

Example:

```
CREATE INDEX studob ON Student(dob);
```

30. Data Integrity

Data Integrity defines the accuracy and consistency of data stored in a database.

31. Distinct

The SELECT DISTINCT statement is used to return only distinct (different) values.
Syntax:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Example:

```
SELECT DISTINCT city FROM student;
```

32. WHERE Clause

The WHERE clause is used to filter records.
Note: The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.
Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

33. Order By clause

The ORDER BY keyword is used to sort the result-set in ascending or descending order.
The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example:

```
SELECT * FROM student ORDER BY name;

SELECT * FROM student WHERE city=”Agra”
        ORDER BY name DESC;

SELECT * FROM student ORDER BY name, city;
```

34. Min | Max

The MIN() function returns the smallest value of the selected column.
Syntax:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

The MAX() function returns the largest value of the selected column.

Syntax:

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

35. LIKE operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Example:

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

36. BINARY KEYWORD

'Binary Keyword' is used to check case sensitive. It is used with 'like' operator.

Example:

```
SELECT * FROM student WHERE BINARY name like ' r% ' ;
```

37. WILDCARD

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the LIKE operator.

Example:

```
SELECT * FROM Customers
WHERE City LIKE ' %es% ' ;
```

38. IS NULL

It is used to check NULL is present or not

Example:

```
SELECT * FROM student WHERE birth_date IS NULL;

SELECT * FROM student WHERE name IS NOT NULL;
```

39. Limit & Offset

LIMIT:- LIMIT show data in a limited number;

Syntax:

```
SELECT column_name FROM table_name WHERE condition
LIMIT number;
```

Example:

```
SELECT * FROM student LIMIT 5;
```

OFFSET:- The OFF SET value allows us to specify which row to start from retrieving data

Syntax:

```
SELECT column_name FROM table_name WHERE condition
LIMIT OFFSET, number;
```

Example:

```
SELECT * FROM student LIMIT 3, 5 ;
```

Offset Limit number

40. IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Example:

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

41. Aliases

ALIASES is a keyword in SQL which is used as a temporary name for a table or column, while writing a query.
ALIASES makes the table or column name more readable.
An alias is created with the AS keyword.

Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

Example:

```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

42. Case Clause

It is used with multiple condition like switch case statement

Syntax:

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

Example:

```
SELECT id, name, percentage,
CASE
  WHEN percentage >=80 AND percentage <100 THEN “Merit”
  WHEN percentage >=60 AND percentage <80 THEN “1st Div”
  WHEN percentage >=45 AND percentage <60 THEN “2nd Div”
  ELSE “Not Correct percentage”
END AS Grade
From  Student;
```

43. STRING Function

- UPPER / UCASE

```
SELECT id, UPPER(name) AS Name, percentage FROM
Student;
```

- LOWER / LCASE

```
SELECT id, LOWER(name) AS Name, percentage FROM Student;
```

- CHARACTER_LENGTH / CHAR_LENGTH / LENGTH
  ```
  SELECT id, name, LENGTH (name) AS character FROM Student;
  ```
- CONCAT
  ```
  SELECT id, CONCAT (name, “ _”, percentage) AS Name FROM Student;
  ```
  Concat multiple String
  ```
  SELECT CONCAT(“Yahoo”, “Baba”, “YouTube”,”Hi”) AS Name;
  ```
- TRIM
  It is used to remove space either left or right or both
  - LTRIM
    ```
    SELECT LTRIM (“        Amit      “)AS Name;
    ```
  - RTRIM
    ```
    SELECT RTRIM(“                    Amit          “) AS Name;
    ```
  - TRIM
    ```
    SELECT TRIM (“       Amit                “) AS Name;
    ```
- SUBSTRING / SUBSTR / MID
  ```
  SELECT SUBSTRING(“Welcome God”,3,6) AS Name;
  ```
  Start End
- REVERSE
  ```
  SELECT REVERSE (“Yahoo Baba”) AS Name;
  ```
- REPLACE
  ```
  SELECT REPLACE(“Yahoo Baba”, “Baba”,”Wow”) AS Name;
  ```
  Replace New Replace
- STRCMP
  - If both string are equal, return 0.
    ```
    SELECT STRCMP(“Amit”,”Amit”) AS Name;
    ```
  - If left side string > right side string, return 1.
    ```
    SELECT STRCMP(“Amit kumar”,”Amit”) AS Name;
    ```
  - If left side string < right side string, return -1.
    ```
    SELECT STRCMP(“Amit”,”Amit Kumar”) AS Name;
    ```
- HEX
  It returns the hexadecimal form of string. It is used for Password
  ```
  SELECT HEX(“Yahoo Baba”) AS Password;
  ```

44. Time Function

- TIME
  It returns only time

SELECT TIME(“2020-06-14 13:15:30”);

- CURRENT_TIMESTAMP
  It return date and time
  SELECT CURRENT_TIMESTAMP();
- TIMESTAMP
  It merge the date & time
  SELECT TIMESTAMP (“2019-06-15”,13:15:20”) AS Time;

45. Date Function

- TO_DAYS
  It also return the difference of two days but one day is predefined selected from 0 years to given date.
  Syntax:
  SELECT TO_DAYS(“2019-08-16”) AS Date;
- DATE_FORMAT
  SELECT DATE_FORMAT(“2019-06-15”,”%y) AS Date;
- CURRENT_DATE
  It return the current date of the server
  SELECT CURRENT_DATE();
- SYSDATE / NOW
  It return the date & time of the server
  SELECT SYSDATE();