FAQ

1)WHAT IS A SERVER?

* A server is a software or hardware device that accepts and responds to requests made over a network.
* It Contain a database application to provide database functionalities to client application

2)WHAT IS SQL ?

* SQL IS STANDS FOR STRUCTURED QUERY LANGUAGE
* SQL computer language that we use to interact with a relational database. SQL is a tool for organizing, managing, and retrieving data from a database IN MINIMAL TIME (OR)LESS PERIOD OF TIME

3) ADVANTAGES OF SQL ?

* Faster Query Processing
* TIME CONSUMING IS LESS
* Portable:dependent of any platform
* Security: SQL databases have built-in security features that help protect data
* Data Integrity: SQL databases enforce data integrity by enforcing constraints such as unique keys, primary keys, and foreign keys, which help prevent data
* Backup and Recovery: SQL databases have built-in backup and recovery tools that help recover data in case of system failures, crashes, or other disasters.duplication and maintain data
* Data Consistency: SQL databases ensure consistency of data across multiple tables through the use of transactions, which ensure that changes made to one table are reflected in all related tables.accuracy.

**4)WHAT IS DATA ?**

DATA ARE THE RAW FACTS WHICH DESCRIBE THE ATTRIBUTES OF ENTITY

* ATTRIBUTES MEANS PROPERTIES
* ENTITY MEANS OBJECTS

5)WHAT IS DATABASE ?

* IT IS A PLACE WHICH DATAS ARE STORED IN SYSTEMATIC AND ORGANIZED WAY.

CRUD OPERATIONS

C- CREATE OR INSERT

R- READ OR RETRIEVE

U- UPDATE OR MODIFY

D- DELETE OR DROP

6)WHAT IS DBMS ?

* DBMS IS STAND FOR DATABASE MANAGEMENT SYSTEM
* DBMS IS A SOFTWARE WHICH IS USED TO MAINTAIN AND MANAGE THE DATABASE.
* It is in file format and it was a text file
* FEATURES
  + SECURITY
  + AUTHORIZATION

7)DIFFERENT BETWEEN THE DATABASE AND DBMS?

DATABASE

* IT IS A PLACE OR MEDIUM IN WHICH WE STORE THE DATA IN SYSTEMATIC AND ORGANIZED WAY
* ITS CONTAINS DATA AND STRUCTURE OF TABLE

DBMS

* IT IS SOFTWARE. IT IS USED TO MAINTAIN OR MANAGE THE DATABASE
* IT IS CONTAINS QUERY PROCESSING SOFTWARE AND STORAGE MANAGEMENT SOFTWARE

8)WHAT IS RDBMS?

* RDBMS IS STAND FOR RELATIONAL DATABASE MANAGEMENT SYSTEM
* IN RDBMS DATAS ARE STORED IN A TABULAR FORMAT OR IN THE FORM OF d deTABLE.
* Relation and Normalization is possible in RDBMS
* It has Security(more than DBMS) and Authorization
* SQL IS USED TO INTERACT OR COMMUNICATE WITH RDBMS.

9)DIFFERENT BETWEEN THE DBMS AND RDBMS?

DBMS

* STORES THE DATA IN FORM OF FILE
* PROVIDES LESS SECURITY
* IN DBMS NORMALIZATION IS NOT POSSIBLE

RDBMS

* STORES THE DATA IN THE FORM OF TABLE
* HIGH SECURITY
* NORMALIZATION IS POSSIBLE
* SQL IS USED TO INTERACT AND COMMUNICATE THE RDBMS

11)SQL SERVER?

* SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is
* primarily designed and developed to compete with MySQL and Oracle database.

10)TABLE

* A Table is an object which stores data in Row & Column format. Below Diagram, shows Rows and Column respectively
* ROWS ALSO CALLED AS RECORD, TUPLES
* COLUMNS ALSO CALLED ATTRIBUTES

11)CELLS

* INTERSECTION OF ROWS AND COLUMNS IS CALLED CELL
* SMALLEST UNIT OF TABLE IS CALLED CELL

12)Data types:

* Data types are used to describe the type/kind of data that will be stored in a particular memory location.
* For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column

14)CLASSIFICATION OF DATA TYPES ?

Data types mainly classified into three categories for every database.

• String Data types

• Numeric Data types

• Date and time Data types

**String Data types**

* char(n)

It is a fixed width character string data type. Its size can be up to 8000 characters.

* varchar(n)

It is a variable width character string data type. Its size can be up to 8000 characters.

* varchar(max)

It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.

* text

It is a variable width character string data type. Its size can be up to 2GB of text data.

* nvarchar

It is a variable width Unicode string data type. Its size can be up to 4000 characters.

* image

It is also a variable width Binary string data type. Its size can be up to 2GB.

**Numeric Data types**

* bit

It is an integer that can be 0, 1 or null.

* tinyint

It allows whole numbers from 0 to 255.

* Smallint

It allows whole numbers between -32,768 and 32,767.

* Int

It allows whole numbers between -2,147,483,648 and 2,147,483,647.

* bigint

It allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807.

* Numeric(Precision , Scale)

Examples:

* Number(4,0) - It will accept 0 - 9999
* Number(4,2) - It will accept 0.99 - 9999.99 (4543.83)
* Number(3,5) - It will accept 0.99999 - 999.99999 (675.43781)

**Date and time Data types**

* Datetime(dd-mm-yyyy:HH:MM:SS)

It is used to specify date and time combination. It supports range from January 1, 1753, to December 31, 9999 with an accuracy of 3.33 milliseconds.

* datetime2

It is used to specify date and time combination. It supports range from January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds

* Date(dd-mm-yyyy)

It is used to store date only. It supports range from January 1, 0001 to December 31, 9999

* time

It stores time only to an accuracy of 100 nanoseconds

14)CONSTRAINTS ?

* SQL constraints are rules that you can imply on the data in a table.
* It allows you to restrict only specific data that meets the regulations to go to a table.
* To put it simply, only if the data meets the constraint’s rules, the insert operation will be successful, or else it will be aborted.

**How to Create Constraints in SQL?**

* you can create constraints in SQL using the CREATE TABLE command while creating a new table or ALTER TABLE command while altering an existing table.

**syntax for create:**

CREATE TABLE table\_name(

column\_name1 data\_type(size) constraint\_name,

column\_name2 data\_type(size) constraint\_name,

….

);

**syntax for alter:**

ALTER TABLE table\_name ALTER COLUMN column\_name data\_type(size) constraint\_name

15)WHAT ARE THE DIFFERENT TYPES OF CONSTRAINTS?

Five types of constraints

1. unique
2. Not noll
3. check
4. Primary key
5. Foreign key.

16)Unique

* It is used to avoid duplicate records or repeated entries:

17)Not null constraints

* When not null assign on particular column which is mandatory to be filled

18)DEFAULT

* The DEFAULT constraint in SQL is used to assign a default value to the left empty fields.
* This will prevent the constraint error, as the field will be given the default value and will not remain NULL.

18)Check

* It is an extra validation to validate the data with some conditions.
* If condition satisfied it accept the data or it will reject

19)Primary Key

The primary key is used uniquely identify the records from the table

* It is always combination of unique and not null
* It will never accept duplicates.
* It will never accept null values.
* In one table,only one primary key should be there.

**IDENTITY :**

* Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
* By Default starting value for IDENTITY is 1, and it will increment by 1
* IDENTITY(10,5) -start at value 10 and increment by 5

20)Foreign key.

Foreign key is used to establish relation from one table to another table

* It can accept duplicate records.
* It can accept null value.
* In one table multiple foreign key Can be possible

Condition :

1. One table should contains Primary key (PK)and other table contains foreign key(FK)
2. A common column in both table
3. Common column data type should be same in both table

21)SQL Commands

* SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
* SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users

22)what are the types of statement?

There are Five types of statement

1. Data definition language. (DDL)
2. Data Manipulation language. (DME)
3. Data control Language (DCL)
4. Transaction control language (TCL)
5. Data Query Language (DQL)

##### **Data Definition Language (DDL)**

* DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
* All the command of DDL are auto-committed that means it permanently save all the changes in the database

List of DDL commands:

* CREATE: This command is used to create the database or its objects (like table, views, store procedure).
* DROP: This command is used to delete objects from the database.
* ALTER: This is used to alter the structure of the database.
* TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

### Data Manipulation Language (DML)

* The SQL commands that deal with the manipulation of data present in the database
* It is responsible for all form of changes in the database.
* The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback

List of DML commands:

* INSERT: It is used to insert data into a table.
* UPDATE: It is used to update existing data within a table.
* DELETE: It is used to delete records from a database table.

**DCL (Data Control Language)**

* DCL mainly deal with the rights, permissions, and other controls of the database system

List of DCL commands:

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

**TCL (Transaction Control Language)**

* TCL commands are used to control the execution of a transaction:
* TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
  + BEGIN: Opens a Transaction.
  + COMMIT: Commit command is used to save all the transactions to the database
  + ROLLBACK:Rollback command is used to undo transactions that have not already been saved to the database
  + NSAVEPOINT: : It is used to roll the transaction back to a certain point without rolling back the entire transaction.

**Data Query Language**

* DQL is used to fetch the data from the database.

It uses only one command:

* SELECT
* The SELECT statement is used to select data from a database.
* The data returned is stored in a result table, called the result-set

**SELECT ALL COLUMNS :**

* SELECT \* FROM table\_name;

**SELECT SPECIFIC COLUMN :**

* SELECT column1, column2, ...FROM table\_name;

**SELECT DISTINCT :**

* **The SELECT DISTINCT statement is used to return only distinct (different) values**

**SYNTAX :**

* **SELECT DISTINCT column1, column2, ...FROM table\_name**

**SELECT TOP :**

* **The SELECT TOP clause is used to specify the number of records to return**

**SYNTAX :**

* **SELECT TOP number|percent column\_name(s) FROM table\_name WHERE condition;**

**SELECT WHERE CLAUSE :**

* **The WHERE clause is used to filter records.**
* **It is used to extract only those records that fulfill a specified condition**

**SYNTAX :**

* **SELECT column1, column2, ...FROM table\_name WHERE condition;**

**SELECT 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**

**JOINS**

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

**Types of the JOINs in SQL:**

**(INNER) JOIN:**

* **Returns records that have matching values in both tables**

**LEFT (OUTER) JOIN:**

* **Returns all records from the left table, and the matched records from the right table**

**RIGHT (OUTER) JOIN:**

* **Returns all records from the right table, and the matched records from the left table**

**FULL (OUTER) JOIN:**

* **Returns all records when there is a match in either left or right table**

**CROSS JOIN :**

* **The SQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN.**
* **This kind of result is called a Cartesian Product.**

**SYNTAX :**

**SELECT column\_name(s) FROM table1**

**JOIN TYPE table2**

**ON**

**table1.column\_name =table2.column\_name;**

**SQL 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;**

**EXAMPLE :**

* **SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City FROM Customers A, Customers B**

**WHERE A.CustomerID <> B.CustomerID**

**AND A.City = B.City**

**ORDER BY A.City;**

**SQL Arithmetic Operators**

**+ Add**

**- Subtract**

**\* Multiply**

**/ Divide**

**% Modulo**

**SQL Comparison Operators**

**= Equal to**

**> Greater than**

**< Less than**

**>= Greater than or equal to**

**<= Less than or equal to**

**<> Not equal to**

**SQL Logical Operators**

* **ALL**

**TRUE if all of the subquery values meet the condition**

* **AND**

**TRUE if all the conditions separated by AND is TRUE**

* **ANY**

**TRUE if any of the subquery values meet the condition**

* **BETWEEN**

**TRUE if the operand is within the range of comparisons**

* **EXISTS**

**TRUE if the subquery returns one or more records**

* **IN**

**TRUE if the operand is equal to one of a list of expressions**

* **LIKE**

**TRUE if the operand matches a pattern**

* **NOT**

**Displays a record if the condition(s) is NOT TRUE**

* **OR**

**TRUE if any of the conditions separated by OR is TRUE**

* **SOME**

**TRUE if any of the subquery values meet the condition**

**SQL Aliases**

* **SQL aliases are used to give a table, or a column in a table, a temporary name.**
* **An alias only exists for the duration of that query.**
* **An alias is created with the AS keyword.**

**Alias Column Syntax**

* **SELECT column\_name AS alias\_name FROM table\_name;**

**Alias Table Syntax**

* **SELECT column\_name(s) FROM table\_name AS alias\_name;**

**SQL aggregation function**

* **SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.**
* **It is also used to summarize the data**

**COUNT() :**

* The COUNT() function is used to count the number of rows or non-null values in a column

**Syntax:**

**Select count(\*) /select (columnname) from table**

**Select (distinct columnname)from table**

**SUM()**

* **The SUM() function returns the total sum of a numeric column.**

**AVG()**

* **The AVG() function returns the average value of a numeric column**

**MIN()**

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

**MAX()**

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

**GROUP BY CLAUSE**

* **The GROUP BY clause in the SELECT statement is used to group rows based on one or more columns.**
* **It is typically used with aggregate functions to perform calculations on each group**

**HAVING CLAUSE**

* **The HAVING clause in the SELECT statement is used to filter grouped rows based on specific conditions.**
* **It operates on the results of the GROUP BY clause.**

**COMMON SYNTAX**

**SELECT column\_name(s)**

**FROM table\_name**

**WHERE condition**

**GROUP BY column\_name(s)**

**HAVING condition**

**ORDER BY column\_name(s)**

**SET OPERATORS**

* **SET operators are special types of operators which are used to combine the result of two queries.**
  + **UNION**
  + **UNION ALL**
  + **INTERSECT**
  + **EXCEPT**

**CONDITIONS :**

**The number and order of columns must be the same.**

**Data types must be compatible.**

**SYNTAX :**

**SELECT COLUMNS FROM TABLE 1**

**UNION/UNION ALL/INTERSECT/EXCEPT**

**SELECT COLUMNS FROM TABLE2**

**UNION**

* UNION will be used to combine the result of two select statements.
* Duplicate rows will be eliminated from the results obtained after performing the UNION operation.

**UNION ALL**

* **It combines all the records from both the queries.**
* **Duplicate rows will be not be eliminated from the results**

**INTERSECT**

* **it only returns the records which are common from both SELECT statements**

**EXCEPT**

* **It displays the rows which are present in the first query but absent in the second query with no duplicates**

**In SQL, a view is a virtual table based on the result-set of an SQL statement.**

**VIEWS**

* **In SQL, a view is a virtual table based on the result-set of an SQL statement**
* A view contains rows and columns, just like a real table.
* The fields in a view are fields from one or more real tables in the database.

CREATE VIEW Syntax

* CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition

DROP VIEW Syntax

* DROP VIEW view\_name

STORED PROCEDURE

* A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
* So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
* You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

CREATE PROCEDURE SYNTAX

CREATE PROCEDURE procedure\_name

(parameter1 data\_type, parameter2 data\_type, …)

AS

BEGIN

— SQL statements to be executed

END

TO MODIFY

ALTER PROCEDURE procedure\_name

(parameter1 data\_type, parameter2 data\_type, …)

AS

BEGIN

— SQL statements to be executed

END

TO EXECUTE

EXEC procedure\_name parameter1\_value, parameter2\_value

EXAMPLE

CREATE PROCEDURE GetCustomersByCountry

@Country VARCHAR(50)

AS

BEGIN

SELECT CustomerName, ContactName

FROM Customers

WHERE Country = @Country;

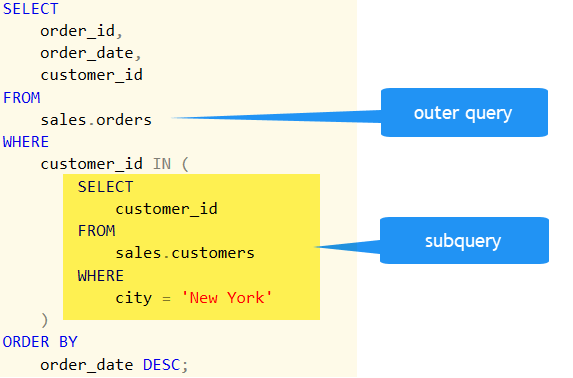
END;

Advantages

1. It is faster.
2. It is pre-compiled.
3. It reduces network traffic
4. It is reusable.
5. Its security is high .

SUB QUERY IN SQL

* In SQL a Subquery (or) Nested query can be simply defined as a query within another query
* The outer query is called as main query and inner query is called as subquery.
* Subquery must be enclosed in parentheses.
* ORDER BY command cannot be used in a Subquery. GROUP BY command can be used to perform the same function as ORDER BY command.
* Use single-row operators with single row Subqueries. Use multiple-row operators with multiple-row Subqueries.

TYPES

1. Non correlated
   1. Single row subquery
   2. Multi row subquery
2. Correlated

Non correlated

* inner query does not depend on the outer query. They both can run separately.

Correlated

* inner query is dependent on the outer query

CONDITIONAL STATEMENTS

* IF -ELSE
* CASE

IF- ELSE CONDITION

* it evaluates the expression, and if the condition is true, then it executes the statement mentioned in the IF block , otherwise statements within the ELSE clause are executed.

SYNTAX

IF (Expression )

BEGIN

-- If the condition is TRUE then execute the following statement

True Statements;

END

ELSE

BEGIN

-- If the condition is False then execute the following statement

False Statements

END

CASE CONDITION

* The CASE statement is used to perform conditional logic within a SQL statement.
* The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
* It allows you to evaluate multiple conditions and return different values based on the result.

SYNTAX

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

WHEN conditionN THEN resultN

ELSE result

END;

Example

SELECT OrderID, Quantity,

CASE

WHEN Quantity > 30 THEN 'The quantity is greater than 30'

WHEN Quantity = 30 THEN 'The quantity is 30'

ELSE 'The quantity is under 30'

END AS QuantityText

FROM OrderDetails;

NORMALIZATION

* Normalization is the process to eliminate data redundancy and enhance data integrity in the table.
* Normalization also helps to organize the data in the database.
* It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables
* It involves splitting tables into smaller, more manageable entities

LEVEL OR TYPES OF NORMALIZATION

a. The different normal forms are:

i. First Normal Form (INF):

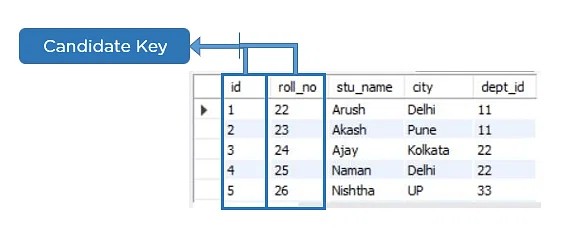
* A table is referred to as being in its First Normal Form if atomicity of the table is 1.
* Here, atomicity states that a single cell cannot hold multiple values. It must hold only a single-valued attribute.

ii. Second Normal Form (2NF):

* The table should be in 1NF
* The table should not possess partial dependency.
* The non-key attributes doesn't partially depends on subset of candidate key
* Ensures that each non-key column depends on the entire primary key.

Candidate Key

* A candidate key is a set of one or more columns that can identify a record uniquely in a table,



iii. Third Normal Form (3NF):

* The Table should be in 2NF
* The table should not possess Transitive dependency
* Ensures that each non-key column depends only on the primary key and not on other non-key columns.
* Therefore, a transitive dependency is a functional dependency in which A → C (A determines C) indirectly, because of A → B and B → C (where it is not the case that B → A).
* The third Normal Form ensures the reduction of data duplication. It is also used to achieve data integrity.

iv.Boyce CoddNormal Form (BCNF)

v. Fourth Normal Form (4NF):

* Eliminates multi-valued dependencies.

vi. Fifth Normal Form (5NF):

* Eliminates join dependencies

BUILD-IN SQL FUNCTIONS

SQL | String functions

* String functions are used to perform an operation on input string and return an output string.

ASCII():

* This function is used to find the ASCII value of a character.

Syntax:

SELECT ascii('t');

Output: 116

LEN():

* This function is used to find the length of a word.

Syntax:

SELECT char\_length('Hello!');

Output: 6