SQL INTERVIEW Q&A

Q1. Define Database ?

Ans. A Database is defined as a structured form of data storage in a computer or a collection of data in an organized manner and can be accessed in various ways. It is also the collection of schemas, tables, queries, views, etc. Databases help us with easily storing, accessing, and manipulating data held on a computer. The Database Management System allows a user to interact with the database.

Q2. What is DBMS?

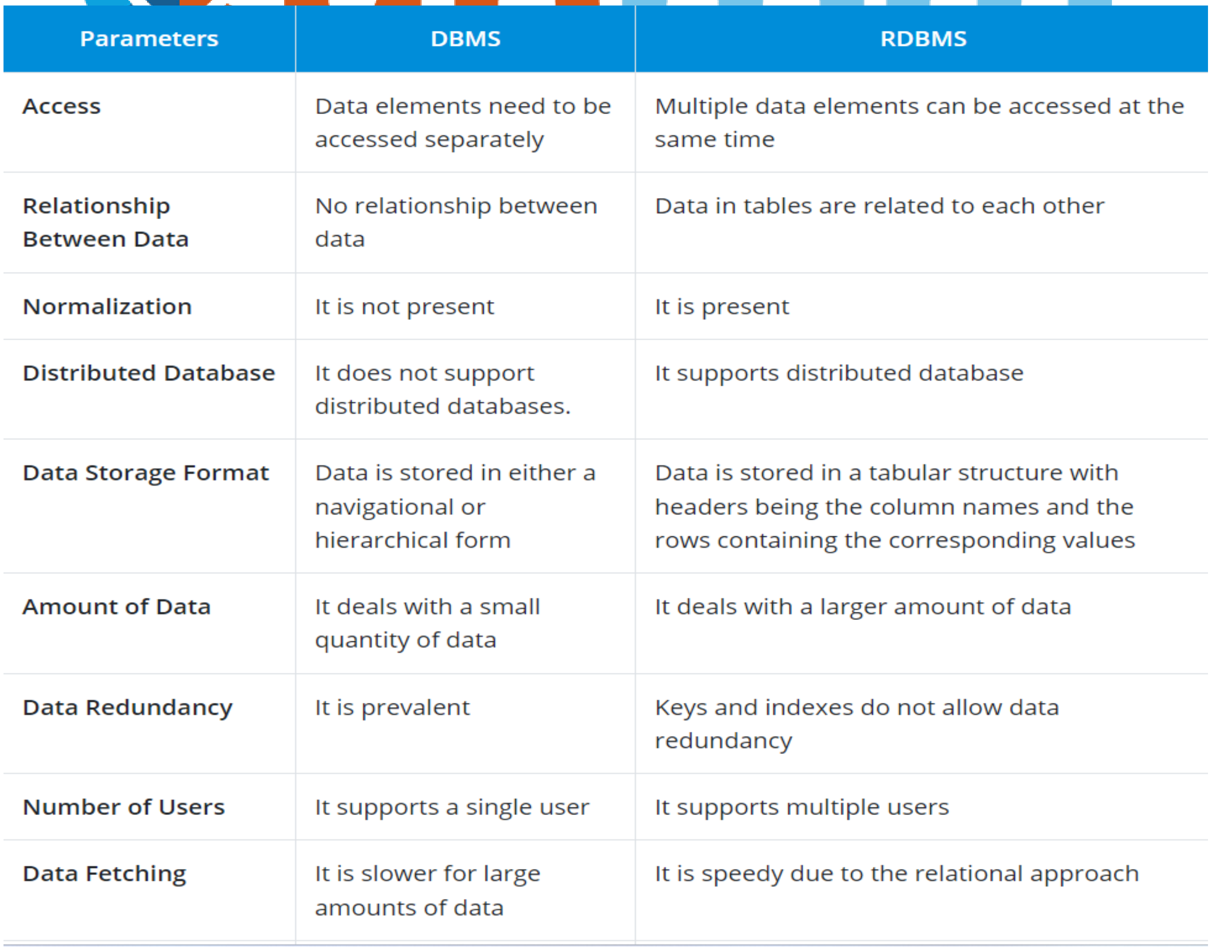
Ans. A Database Management System (DBMS) is a software application that interacts with the user, applications, and the database itself to capture and analyze data. A database is a structured collection of data.

Q3. What is RDMS?

Ans. An RDBMS stores data in the form of a collection of tables. The relations are defined between the common fields of these tables. MS SQL Server, MySQL, IBM DB2, Oracle, and Amazon Redshift are all based on RDBMS.

Q4. Difference Between DBMS and RDBMS?

Ans. Acc. To E.F CODD 12 rules(i.e 13 rules. First rule is known as zero rule) which Management system follows more than 7 rules is a RDBMS.



Parameters	DBMS	RDBMS
Access	Data elements need to be accessed separately	Multiple data elements can be accessed at the same time
Relationship Between Data	No relationship between data	Data in tables are related to each other
Normalization	It is not present	It is present
Distributed Database	It does not support distributed databases.	It supports distributed database
Data Storage Format	Data is stored in either a navigational or hierarchical form	Data is stored in a tabular structure with headers being the column names and the rows containing the corresponding values
Amount of Data	It deals with a small quantity of data	It deals with a larger amount of data
Data Redundancy	It is prevalent	Keys and indexes do not allow data redundancy
Number of Users	It supports a single user	It supports multiple users
Data Fetching	It is slower for large amounts of data	It is speedy due to the relational approach

Data Security	Low-security levels when it comes to data manipulation	Multiple levels of data security exist
Software and Hardware Requirements	Low	High
Examples	XML, Window Registry, etc.	MySQL, SQL Server, Oracle, Microsoft Access, PostgreSQL, etc.

Q4. What is SQL?

Ans. SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

Q5. What is difference between SQL and MYSQL?

Ans. 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.

SQL	MySQL
It is a structured query language used in a database	It is a database management system
It is used for query and operating database systems,	It allows data handling, storing, and modification in an organized manner
It is always the same	It keeps updating
It supports only a single storage engine	It supports multiple storage engines
The server is independent	During backup sessions, the server blocks the database

Q6. What are the subset of SQL?

Ans. SQL queries are divided into four main categories:

Data Definition Language (DDL)
DDL queries are made up of SQL commands that can be used to define the structure of the database and modify it.

CREATE: Creates databases, tables, schema, etc.

DROP: Drops tables and other database objects

ALTER: Alters the definition of database objects

TRUNCATE: Removes tables, views, procedures, and other database objects

Rename: Rename the column and table name.

Data Manipulation Language (DML)
These SQL queries are used to manipulate data in a database.

SELECT : Selects data from one table and inserts it into another .(DQL-data query language).

INSERT: Inserts data or records into a table

UPDATE: Updates the value of any record in the database

DELETE: Deletes records from a table

Data Control Language (DCL)
These SQL queries manage the access rights and permission control of the database.

GRANT: Grants access rights to database objects

REVOKE: Withdraws permission from database objects

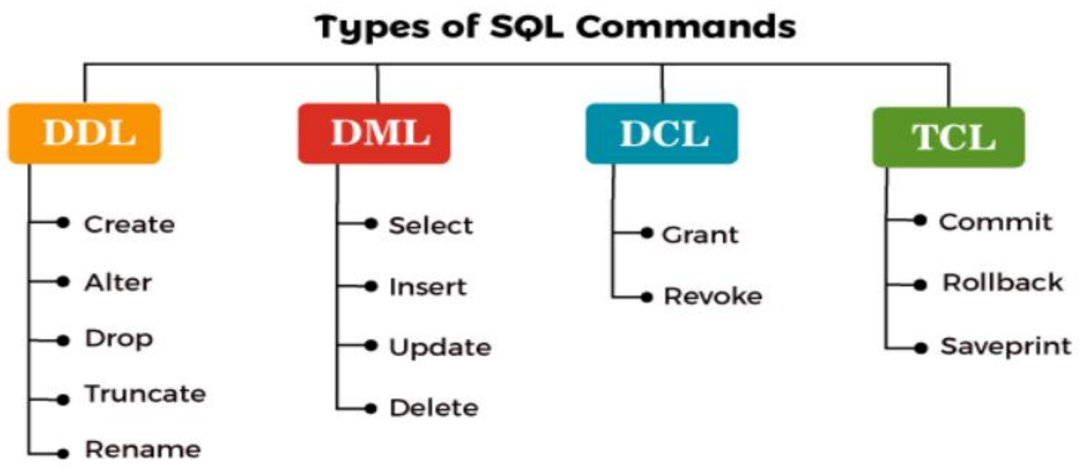
Transaction Control Language (TCL)
TCL is a set of commands that essentially manages the transactions in a database and the changes made by the DML statements. TCL allows statements to be grouped together into logical transactions.

COMMIT: Commits an irreversible transaction, i.e., the previous image of the database before the transaction cannot be retrieved.

ROLLBACK: Reverts the steps in a transaction in case of an error

SAVEPOINT: Sets a savepoint in the transaction to which rollback can be executed

SET TRANSACTION: Sets the characteristics of the transaction



Q7. What are Tables and Fields?

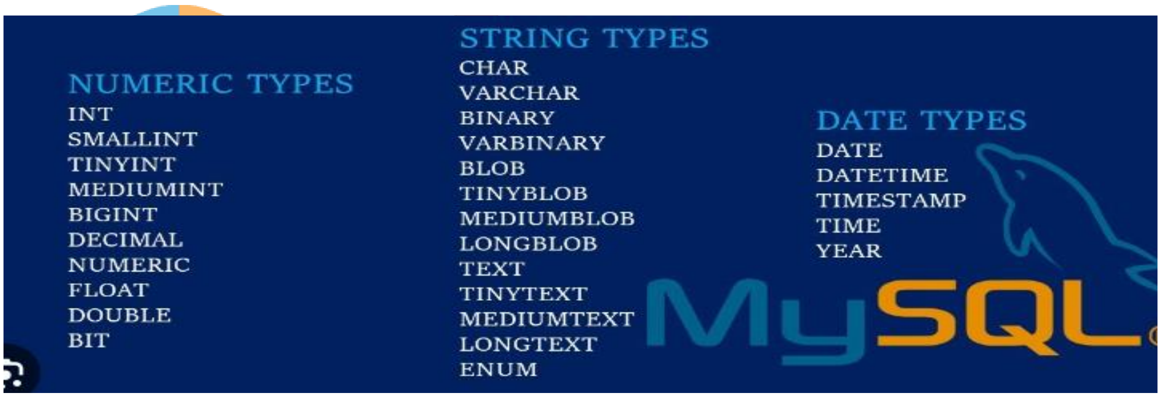
Ans. A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

Q8. What are datatypes in mysql?

Ans. A Data Type specifies a particular type of data, like integer, floating points, Boolean, etc. It also identifies the possible values for that type, the operations that can be performed on that type, and the way the values of that type are stored. In MySQL, each database table has many columns and contains specific data types for each column.

We can determine the data type in MySQL with the following characteristics:

- The type of values (fixed or variable) it represents.
- The storage space it takes is based on whether the values are a fixed-length or variable length.
- How MySQL performs a comparison of values of a particular data type.



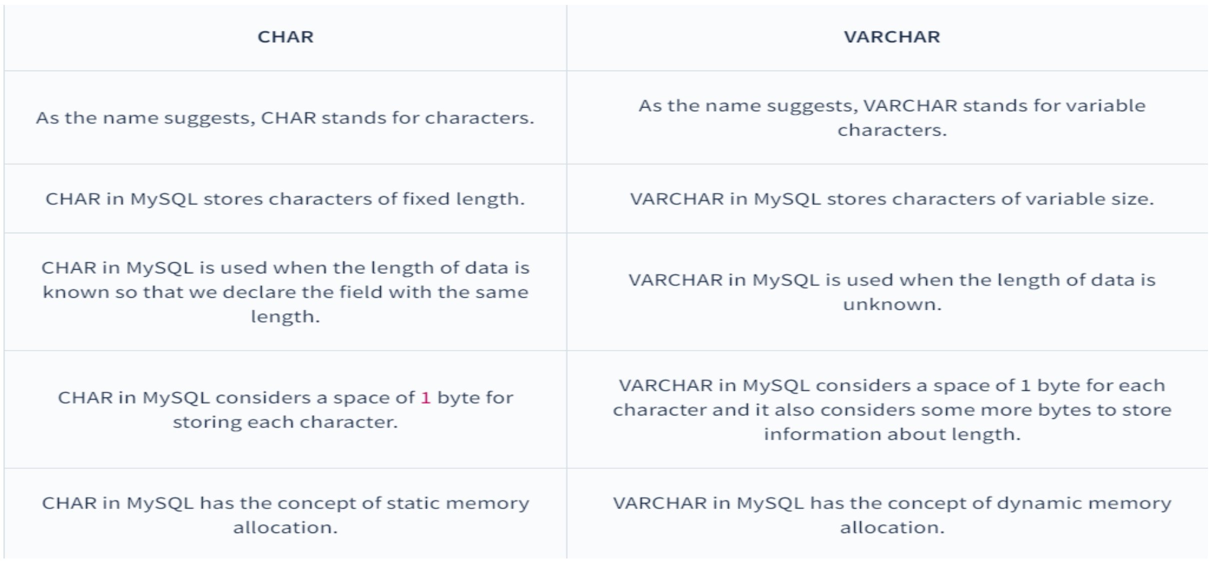
NUMERIC TYPES
INT
SMALLINT
TINYINT
MEDIUMINT
BIGINT
DECIMAL
NUMERIC
FLOAT
DOUBLE
BIT

STRING TYPES
CHAR
VARCHAR
BINARY
VARBINARY
BLOB
TINYBLOB
MEDIUMBLOB
LONGBLOB
TEXT
TINYTEXT
MEDIUMTEXT
LONGTEXT
ENUM

DATE TYPES
DATE
DATETIME
TIMESTAMP
TIME
YEAR

Q9.Difference between chat and varchar?

Ans.



CHAR	VARCHAR
As the name suggests, CHAR stands for characters.	As the name suggests, VARCHAR stands for variable characters.
CHAR in MySQL stores characters of fixed length.	VARCHAR in MySQL stores characters of variable size.
CHAR in MySQL is used when the length of data is known so that we declare the field with the same length.	VARCHAR in MySQL is used when the length of data is unknown.
CHAR in MySQL considers a space of 1 byte for storing each character.	VARCHAR in MySQL considers a space of 1 byte for each character and it also considers some more bytes to store information about length.
CHAR in MySQL has the concept of static memory allocation.	VARCHAR in MySQL has the concept of dynamic memory allocation.

Q10. Difference between datetime and timestamp ?

Ans. Range:

DATETIME: The DATETIME data type can represent dates and times in a range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. It has a wide range of values, covering a vast span of time.

TIMESTAMP: The TIMESTAMP data type has a more limited range compared to DATETIME. It can represent dates and times from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. This limitation is due to the use of a 32-bit integer internally to store Unix timestamps.

Time Zone Handling:

DATETIME: DATETIME values are not associated with any time zone. They are stored as provided and do not change when the time zone setting of the MySQL server changes. This means that DATETIME values are typically considered to be in the server's time zone.

TIMESTAMP: TIMESTAMP values are sensitive to the time zone setting of the MySQL server. When you insert or retrieve a TIMESTAMP value, it is automatically converted to and from the server's time zone. This makes TIMESTAMP suitable for recording events with consistent time references across time zones.

Q11. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- NOT NULL - Restricts NULL value from being inserted into a column.
- CHECK - Verifies that all values in a field satisfy a condition.
- DEFAULT - Automatically assigns a default value if no value has been specified for the field.
- UNIQUE - Ensures unique values to be inserted into the field.
- INDEX - Indexes a field providing faster retrieval of records.
- PRIMARY KEY - Uniquely identifies each record in a table.
- FOREIGN KEY - Ensures referential integrity for a record in another table.

Q.12 What is Primary key ?

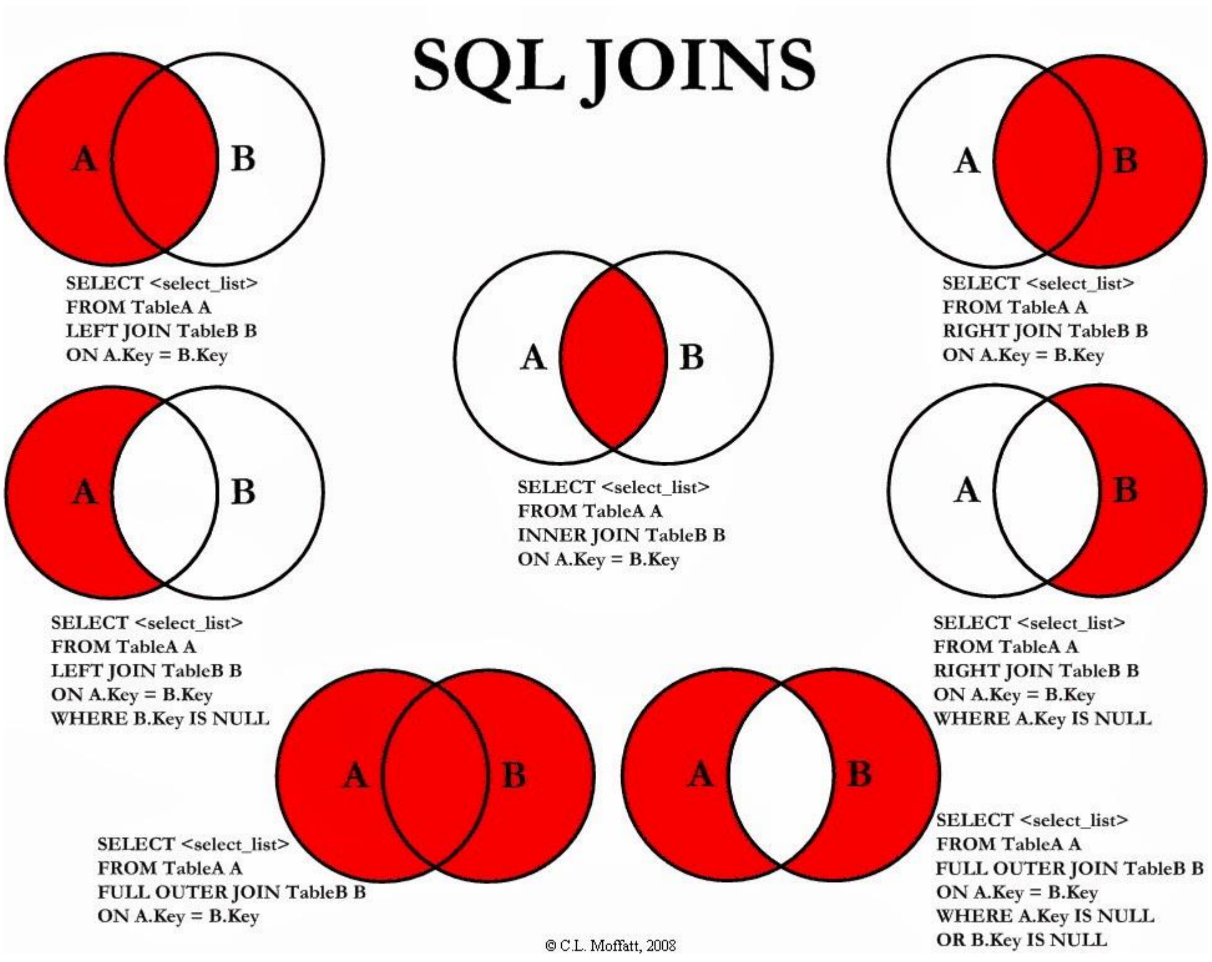
Ans. A PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

Q.13 What is foreign Key?

Ans. A foreign key (FK) is a column or combination of columns that is used to establish and enforce a link between the data in two tables to control the data that can be stored in the foreign key table. In a foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table. This column becomes a foreign key in the second table.

Q.14 What is a Join? List its different types.

Ans. The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.



Cross join : In SQL, CROSS JOINs are used to combine each row of one table with each row of another table, and return the Cartesian product of the sets of rows from the tables that are joined.

Inner join: Inner joins combine records from two tables whenever there are matching values in a field common to both tables. You can use INNER JOIN with the Departments and Employees tables to select all the employees in each department.

Left join: Inner join (matched value of both table) +remaining value of left table.

Right join: Inner join (matched value of both table) +remaining value of right table.

Full join: Inner join (matched value of both table) +remaining value of left table + remaining value of right table.

Natural join: When we combine rows of two or more tables based on a common column between them, this operation is called joining. A natural join is a type of join operation that creates an implicit join by combining tables based on columns with the same name and data type. It is similar to the INNER or LEFT JOIN, but we cannot use the ON or USING clause with natural join as we used in them.

Ques.15 What is self join ?

Ans. A SELF JOIN is a join that is used to join a table with itself. In the previous sections, we have learned about the joining of the table with the other tables using different JOINS, such as INNER, LEFT, RIGHT, and CROSS JOIN. However, there is a need to combine data with other data in the same table itself. In that case, we use Self Join.

We can perform Self Join using table aliases. The table aliases allow us not to use the same table name twice with a single statement. If we use the same table name more than one time in a single query without table aliases, it will throw an error.

The table aliases enable us to use the temporary name of the table that we are going to use in the query. Let us understand the table aliases with the following explanation.

Ques.16 what is data integrity?

Ans. Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle and is a critical aspect of the design, implementation, and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

Ques.17 What is a Subquery? What are its types?

Ans.17 A subquery is a query within another query, also known as a nested query or inner query. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

There are two types of subquery - Correlated and Non-Correlated.

1. A correlated subquery cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
2. A non-correlated subquery can be considered as an independent query and the output of the subquery is substituted in the main query.

Ques. 18 What are some common clauses used with SELECT query in SQL?

Ans.18 SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

Some common SQL clauses used in conjuction with a SELECT query are as follows:

- WHERE clause in SQL is used to filter records that are necessary, based on specific conditions.
- ORDER BY clause in SQL is used to sort the records based on some field(s) in ascending (ASC) or descending order (DESC).
- GROUP BY clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- HAVING clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

Ques.19 What is difference between Having and where Clause?

Ans. 19

Where Clause in SQL	**Having Clause in SQL**
Filter table based data catering to specific condition	Group based data under set condition
Applicable without GROUP BY clause	Does not function without GROUP BY clause
Row functions	Column functions
Select, update and delete statements	Only select statement
Applied before GROUP BY clause	Used after GROUP BY clause
Used with single row operations such as Upper, Lower and so on	Applicable with multiple row functions such as Sum, count and so on

Ques.20 Explain the order of execution of SQL ?

Ans.20 Order of Execution of SQL:

- FROM: Specifies the tables, joins, and subqueries to retrieve data.
- WHERE: Filters the data based on conditions.
- GROUP BY: Groups rows sharing a property so that aggregate functions can be applied.
- HAVING: Filters groups based on aggregate properties.
- SELECT: Selects the columns to be returned.
- ORDER BY: Sorts the result set.
- LIMIT/OFFSET: Specifies the number of rows to return.

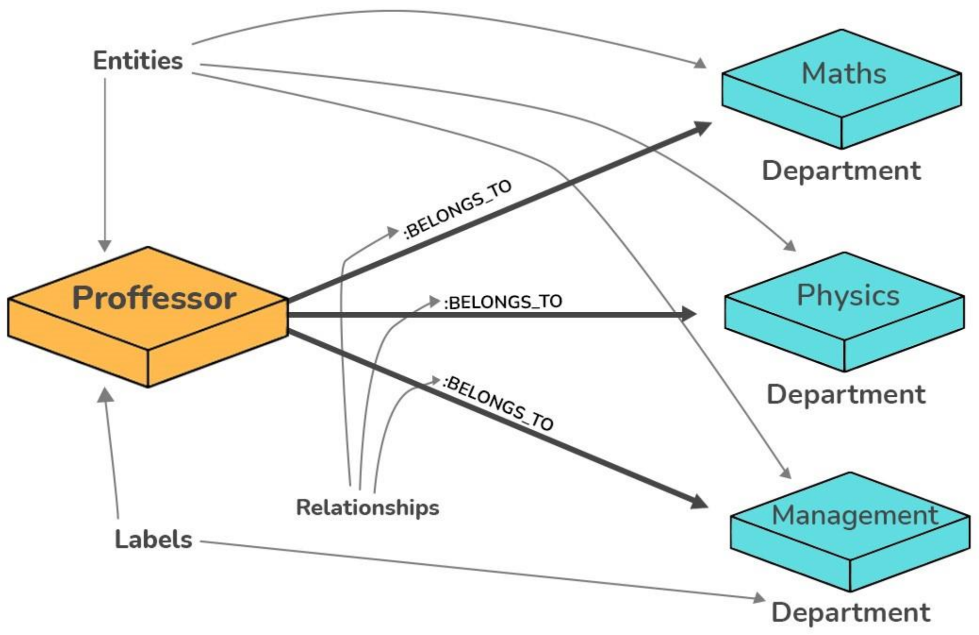
Ques.21 What is index ?

Ans.21 A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.**it reduces the I/O cost.**

Ques.22 What are Entities and Relationships?

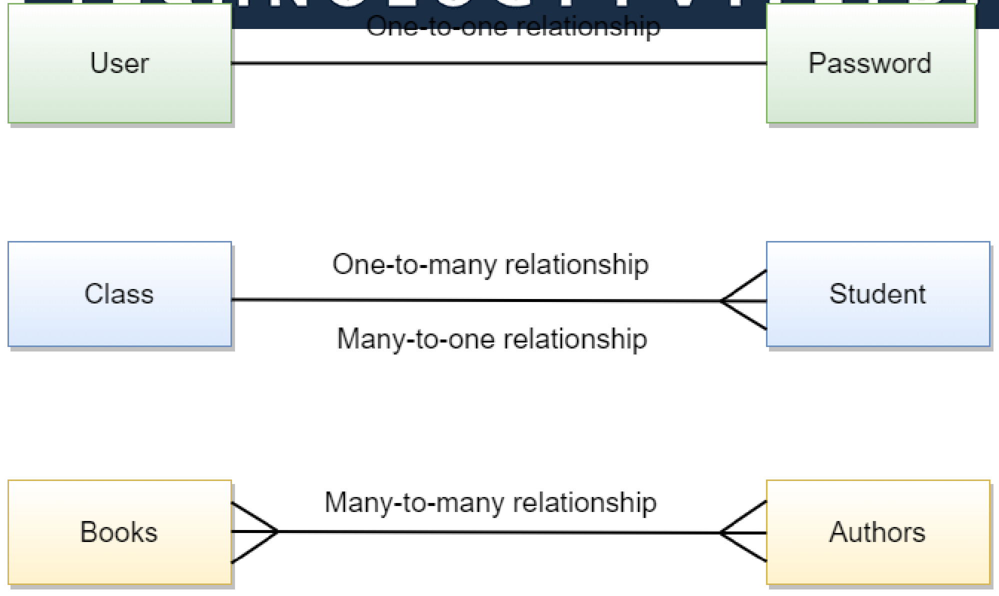
Ans.22 Entity: An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database



There are 3 main types of relationship in a database:

1. one-to-one
2. one-to-many
3. many-to-many.



Ques.23 What are operators in mysql and types of operator?

Ans. The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query. SQL operators are used for filtering the table's data by a specific condition in the SQL statement. In SQL, an operator can either be a unary or binary operator. The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.

SQL operators are categorized in the following categories:

a) SQL Arithmetic Operators:

The Arithmetic Operators perform the mathematical operation on the numerical data of the SQL tables.
— SQL Addition Operator (+)
— SQL Subtraction Operator (-)
— SQL Multiplication Operator (+)
— SQL Division Operator (-)
— SQL Modulus Operator (+)

b) SQL Comparison Operators:

The Comparison Operators in SQL compare two different data of SQL table and check whether they are the same, greater, and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries. — SQL Equal Operator (=)
— SQL Not Equal Operator (!=)
— SQL Greater Than Operator (>)
— SQL Greater Than Equals to Operator (>=)
— SQL Less Than Operator (< =)

c) SQL Logical Operators:

The Logical Operators in SQL perform the Boolean operations, which give two results True and False. These operators provide True value if both operands match the logical condition.

SQL AND operator: The AND operator in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True. It is also known as the conjunctive operator and is used with the WHERE clause.
SELECT column1, …., columnN FROM table_Name WHERE condition1 AND conditi on2 AND condition3 AND ……. AND conditionN;

SQL OR operator: The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True.
SELECT column1, …., columnN FROM table_Name WHERE condition1 OR conditio n2 OR condition3 OR ……. OR conditionN;

SQL BETWEEN operator: The BETWEEN operator in SQL shows the record within the range mentioned in the SQL query. This operator operates on the numbers, characters, and date/time operands. If there is no value in the given range, then this operator shows NULL value.

SQL IN operator: The IN operator in SQL allows database users to specify two or more values in a WHERE clause.

SQL NOT operator: The NOT operator in SQL shows the record from the table if the condition evaluates to false. It is always used with the WHERE clause.

SQL ANY operator: The ANY operator in SQL shows the records when any of the values returned by the sub-query meet the condition. The ANY logical operator must match at least one record in the inner query.

SQL LIKE operator: The LIKE operator in SQL shows those records from the table which match with the given pattern specified in the sub-query.

SQL Wildcard Characters:

The percentage (%) sign is a wildcard which is used with this logical operator “Like”. This operator is used in the WHERE clause with the following three statements:
1. SELECT statement
2. UPDATE statement
3. DELETE statement

Ques.24 What is difference between Delete and Truncate?

Ans.24

DELETE		TRUNCATE
The DELETE command is used to delete particular records from a table.	Definition	The TRUNCATE command is used to delete the complete data from the table.
It is a DML command.	Language Type	It is a DDL command
The DELETE command acquires the lock on every deleting record; thus, it requires more locks and resources.	Locks and Resources	The TRUNCATE command requires fewer locks and resources before deleting the data page because it acquires the lock on the data page
It works with the WHERE clause.	WHERE Clause	It does not work with the WHERE clause.
DELETE operation operates on data records and executes deletion one-by-one on records in the order of the queries processed	Working	TRUNCATE operates on data pages and executes deletion of the whole table data at a time.

Its speed is slow as it makes operations in rows and records it in transaction logs	Speed	Its speed is fast as it only records data pages in transaction logs.
It records all the deleted data rows in the transaction log.	Transaction Log	It records only the deleted data pages in the transaction log.
You can restore the data using the COMMIT or ROLLBACK command.	Rollback	You cannot restore the deleted data after executing this command.
The DELETE statement deletes the records and does not interfere with the table's identity.	Table Identity	The TRUNCATE statement does not delete the table structure but resets the identity of the table
It works with an indexed view.	Indexed View	It does not work with indexed views.
It activates the triggers on the table and causes them to fire	Triggers	It does not activate the triggers applied on the table.

Ques.25 what is CTE ?

Ans. In MySQL, every statement or query produces a temporary result or relation. A common table expression or CTE is used to name those temporary results set that exist within the execution scope of that particular statement, such as CREATE, INSERT, SELECT, UPDATE, DELETE, etc.

Some of the key point related to CTE are:

- It is defined by using the WITH clause.
- The WITH clause allows us to specify more than one CTEs in a single query.
- A CTE can reference other CTEs that are part of the same WITH clause, but those CTEs should be defined earlier.
- The execution scope of CTE exists within the particular statement in which it is used.

Ques. 26 What is View ? Uses of view?

Ans.26 MySQL Views are defined as the virtual tables that can be generated by the query output. These MySQL Views are considered objects, so they can be easily queried by leveraging the SELECT statement. MySQL Views do not store the physical data on the database. When you run a SELECT statement on a database view, it will carry out the query and populate the data from the underlying tables utilized to create MySQL Views.

CREATE VIEW `view_name` AS SELECT statement;

- Increased Reusability: By leveraging MySQL Views, you can greatly simplify complex queries, turning them into a single line of code. This code can then be seamlessly integrated with your application thus getting rid of the cumbersome effort required in writing the same formula in your query over and over again. This allows the code to be more readable and reusable.
- Simplification of Complex Queries: By using MySQL Views, users can simplify the most sophisticated MySQL queries. If you are using a complex query for a specific purpose, you can create MySQL Views based on it to use a simple SELECT statement as opposed to typing out the intricate query yet again.
- Enabling Backward Compatibility: You can also use MySQL Views to enable backward compatibility within legacy systems. For instance, say you want to dissect a large table into several smaller ones without impacting the current applications that refer to the table. In this scenario, you can create MySQL Views with the same name as the real table so that the current applications can refer to the View as if it were a table.
- Robust Data Security: You can leverage MySQL Views to depict only authorized information to the users and conceal pivotal data like banking and personal information. You can limit which information can be accessed by your consumers by authoring only the necessary data for them.

Ques.27 what is the difference between view and table ?

Ans.27

SN	Table	View
1.	A table is used to organize data in the form of rows and columns and displayed them in a structured format. It makes the stored information more understandable to the human.	Views are treated as a virtual/logical table used to view or manipulate parts of the table. It is a database object that contains rows and columns the same as real tables.
2.	Table is a physical entity that means data is actually stored in the table.	The view is a virtual entity, which means data is not actually stored in the table.
3.	It is used to store the data.	It is used to extract data from the table.

4.	It generates a fast result.	The view generates a slow result because it renders the information from the table every time we query it.
5.	It is an independent data object.	It depends on the table. Therefore we cannot create a view without using tables.
6.	Table allows us to perform DML operations.	The view will enable us to perform DML operations.
7.	It is not an easy task to replace the table directly because of its physical storage.	It is an easy task to replace the view and recreate it whenever needs.
8.	It occupies space on the systems.	It does not occupy space on the systems.

Ques.28 what is stored procedure ?

Ans.28 A procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database. It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements. We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc. It was first introduced in MySQL version 5. Presently, it can be supported by almost all relational database systems.

- Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.
- Stored procedure reduces the traffic between application and database server. Because the application has to send only the stored procedure's name and parameters instead of sending multiple SQL statements.
- Stored procedures are reusable and transparent to any applications.
- A procedure is always secure. The database administrator can grant permissions to applications that access stored procedures in the database without giving any permissions on the database tables.

Ques29. What are parameters in stored procedure ? and types ?

Ans29.

DELIMITER $$

```
CREATE PROCEDURE procedure_name [[IN | OUT | INOUT] parameter_name datatype [, parameter d
atatype]) ]

BEGIN

   Declaration_section

   Executable_section

END $$

DELIMITER ;
```

IN parameter

It is the default mode. It takes a parameter as input, such as an attribute. When we define it, the calling program has to pass an argument to the stored procedure. This parameter's value is always protected.

OUT parameters

It is used to pass a parameter as output. Its value can be changed inside the stored procedure, and the changed (new) value is passed back to the calling program. It is noted that a procedure cannot access the OUT parameter's initial value when it starts.

INOUT parameters

It is a combination of IN and OUT parameters. It means the calling program can pass the argument, and the procedure can modify the INOUT parameter, and then passes the new value back to the calling program.

We can use the CALL statement to call a stored procedure. This statement returns the values to its caller through its parameters (IN, OUT, or INOUT). The following syntax is used to call the stored procedure in MySQL:

```
CALL procedure_name ( parameter(s)) ;
```

Ques.30 What are the difference between function and stored procedure ?

Ans. 30

Functions	Procedures
A function has a return type and returns a value.	A procedure does not have a return type. But it returns values using the OUT parameters.
You cannot use a function with Data Manipulation queries. Only Select queries are allowed in functions.	You can use DML queries such as insert, update, select etc… with procedures.
A function does not allow output parameters	A procedure allows both input and output parameters.
You cannot manage transactions inside a function.	You can manage transactions inside a procedure.
You cannot call stored procedures from a function	You can call a function from a stored procedure.
You can call a function using a select statement.	You cannot call a procedure using select statements.

Ques. 31 What is an Alias in SQL?

Ans.31 An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

Ques.32 What is trigger ? and types of triggers in mySql?

Ans.32 A trigger in MySQL is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on any DML statement such as INSERT, UPDATE, or DELETE.

A trigger is called a special procedure because it cannot be called directly like a stored procedure. The main difference between the trigger and procedure is that a trigger is called automatically when a data modification event is made against a table. In contrast, a stored procedure must be called explicitly.

- triggers are of two types according to the SQL standard: row-level triggers and statement-level triggers.
    - Row-Level Trigger: It is a trigger, which is activated for each row by a triggering statement such as insert, update, or delete. For example, if a table has inserted,

updated, or deleted multiple rows, the row trigger is fired automatically for each row affected by the insert, update, or delete statement.

- Statement-Level Trigger: It is a trigger, which is fired once for each event that occurs on a table regardless of how many rows are inserted, updated, or deleted.

SYNTAX:

```
Delimiter $$

CREATE TRIGGER trigger_name

   (AFTER | BEFORE) (INSERT | UPDATE | DELETE)

       ON table_name FOR EACH ROW

       BEGIN

      --variable declarations

      --trigger code

      END;

Delimiter $$
```

WHY TRIGGER?

- Triggers help us to enforce business rules.
- Triggers help us to validate data even before they are inserted or updated.
- Triggers help us to keep a log of records like maintaining audit trails in tables.
- SQL triggers provide an alternative way to check the integrity of data.
- Triggers provide an alternative way to run the scheduled task.
- Triggers increases the performance of SQL queries because it does not need to compile each time the query is executed.
- Triggers reduce the client-side code that saves time and effort.
- Triggers help us to scale our application across different platforms.
- Triggers are easy to maintain.

Types of triggers:

We can define the maximum six types of actions or events in the form of triggers:

- Before Insert: It is activated before the insertion of data into the table.
- After Insert: It is activated after the insertion of data into the table.
- Before Update: It is activated before the update of data in the table.
- After Update: It is activated after the update of the data in the table.
- Before Delete: It is activated before the data is removed from the table.
- After Delete: It is activated after the deletion of data from the table.

When we use a statement that does not use INSERT, UPDATE or DELETE query to change the data in a table, the triggers associated with the trigger will not be invoked.

Ques.33 what are Window functions ? and types of window function ?

Ans.33 A window function in MySQL used to do a calculation across a set of rows that are related to the current row. The current row is that row for which function evaluation occurs. Window functions perform a calculation similar to a calculation done by using the aggregate functions. But, unlike aggregate functions that perform operations on an entire table, window functions do not produce a result to be grouped into one row. It means window functions perform operations on a set of rows and produces an aggregated value for each row. Therefore each row maintains the unique identities.

Syntax:

```
window_function_name(expression)
OVER (
  [partition_defintion]
  [order_definition]
  [frame_definition]  )
```

In the syntax, it can be seen that we have first specified the name of the window functions, which is followed by an expression. Then, we specify the OVER clause that contains three expressions that are partition_definition, order_definition, and frame_definition.

It makes sure that an OVER clause always has an opening and closing parentheses, even it does not have any expression.

Let us see the syntax of each expression used in the OVER clause:

Partition Clause

This clause is used to divide or breaks the rows into partitions, and the partition boundary separates these partitions. The window function operates on each partition, and when it crosses the partition boundary, it will be initialized again. The syntax of this clause is given below:

PARTITION BY <expression>[{,<expression>...}]

In the partition clause, we can define one or more expressions that are separated by commas.

ORDER BY Clause

This clause is used to specify the order of the rows within a partition. The following are the syntax of ORDER BY clause:

ORDER BY <expression> [ASC|DESC], [{,<expression>...}]

We can also use it to order the rows within a partition on multiple keys where each key specified by an expression. This clause can also define one or more expressions that are separated by commas. Although the ORDER BY clause can work with all window functions, it is recommended to use it with order-sensitive window function.

Frame Clause

A frame is the subset of the current partition in window functions. So we use frame clause to define a subset of the current partition. The syntax of creating a subset of the current partition using frame clause is as follows:

frame_unit {<frame_start>|<frame_between>}

We can use the current row to define a Frame that allows moving within a partition with respect to the position of the current row.

In the syntax, the frame_unit that can be ROWS or RANGE is responsible for defining the type of relationship between the frame row and the current row. If the frame_unit is ROWS, then the offset of the frame rows and the current row is row number. While if the frame_unit is RANGE, then the offset is row values.

The frame_start and frame_between expressions are used to specify the frame boundary. The frame_start expression has three things:

UNBOUNDED PRECEDING: Here, the frame starts from the first row of a current partition.

N PRECEDING: Here, N is a literal number or an expression that evaluates in numbers. It is the number of rows before the first current row.

CURRENT ROW: It specifies the row of the recent calculation.

Ques.34 what are types of window function ?

Ans.34 Types of Window Function

We can categorize the window functions mainly in three types that are given below:

Aggregate Functions

It is a function that operates on multiple rows and produces the result in a single row. Some of the important aggregate functions are:

COUNT, SUM, AVG, MIN, MAX, and many more.

Ranking Functions

It is a function that allows us to rank each row of a partition in a given table. Some of the important ranking functions are:

RANK, DENSE_RANK, PERCENT_RANK, ROW_NUMBER, CUME_DIST, etc.

Analytical Functions

It is a function, which is locally represented by a power series. Some of the important analytical functions are:

NTILE, LEAD, LAG, NTH, FIRST_VALUE, LAST_VALUE, etc.

Name	**Description**
CUME_DIST	Calculates the cumulative distribution of a value in a set of values.
DENSE_RANK	Assigns a rank to every row within its partition based on the ORDER BY clause. It assigns the same rank to the rows with equal values. If two or more rows have the same rank, then there will be no gaps in the sequence of ranked values.
FIRST_VALUE	Returns the value of the specified expression with respect to the first row in the window frame.

Name	Description
LAG	Returns the value of the Nth row before the current row in a partition. It returns NULL if no preceding row exists.
LAST_VALUE	Returns the value of the specified expression with respect to the last row in the window frame.
LEAD	Returns the value of the Nth row after the current row in a partition. It returns NULL if no subsequent row exists.
NTH_VALUE	Returns value of argument from Nth row of the window frame
NTILE	Distributes the rows for each window partition into a specified number of ranked groups.
PERCENT_RANK	Calculates the percentile rank of a row in a partition or result set
RANK	Similar to the DENSE_RANK() function except that there are gaps in the sequence of ranked values when two or more rows have the same rank.
ROW_NUMBER	Assigns a sequential integer to every row within its partition

Ques. 35 What is Normalization?

Ans. Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

Ques. 36 What is Denormalization?

Ans. 36 Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

Ques.37 What are the various type of normalization?

Ans.37 Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

First Normal Form:
A relation is in first normal form if every attribute in that relation is a single-valued attribute. If a relation contains a composite or multi-valued attribute, it violates the first normal form. Let's consider the following students table. Each student in the table, has a name, his/her address, and the books they issued from the public library -

Students Table

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form -

Students Table (1st Normal Form)

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

Second Normal Form:

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has no partial dependency, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples -

Example 1 - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new Primary Key attribute to identify the individual records in the Students table. The Foreign Key constraint will be set on the other table to ensure referential integrity.

Students Table (2nd Normal Form)

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.

Student_ID	Student	Address	Salutation
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.
4	Ansh	Windsor Street 777	Mr.

Books Table (2nd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Example 2 - Consider the following dependencies in relation to R(W,X,Y,Z)

WX -> Y [W and X together determine Y]

XY -> Z [X and Y together determine Z]

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn’t determine any non-prime attribute in the relation.

Third Normal Form

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is no transitive dependency between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

Example 1 - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single candidate key Student_ID (primary key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new Foreign Key constraint to identify the salutations for individual records in the Students table. The Primary Key constraint for the same will be set on the Salutations table to identify each record uniquely.

Students Table (3rd Normal Form)

Student_ID	Student	Address	Salutation_ID
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

Books Table (3rd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Salutations Table (3rd Normal Form)

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

Example 2 - Consider the following dependencies in relation to R(P,Q,R,S,T)

P -> QR [P together determine C]

RS -> T [B and C together determine D]

Q -> S

T -> P

For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

Boyce-Codd Normal Form

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand-Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form X –> Y, such that X is always a super key. For example - In the above example, Student_ID serves as the sole unique identifier for the Students Table and Salutation_ID for the Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student_ID.

Ques.38 write retention query in SQL?

Ans. 38 Writing a retention query in SQL involves calculating the percentage of users who return to your application or service after their initial visit. Retention can be measured over various time periods (e.g., daily, weekly, monthly). Here's a general approach to writing a retention query, using a hypothetical scenario where we have a user_activities table with columns user_id, activity_date.

Steps:

Identify Initial Activity: First, find the initial activity date for each user.

Calculate Retention Intervals: Then, for each user's activity, calculate the time interval between that activity and the user's initial activity.

Aggregate and Calculate Retention: Finally, group users based on the interval and calculate retention.

Sample SQL Query:

```
WITH FirstActivity AS (
  SELECT  user_id,  MIN(activity_date) as first_activity_date
  FROM user_activities GROUP BY  user_id),
ActivityIntervals AS (
  SELECT
    fa.user_id,
    ua.activity_date,
    DATEDIFF(day, fa.first_activity_date, ua.activity_date) as days_since_first_activity
  FROM  user_activities ua
  JOIN FirstActivity fa ON ua.user_id = fa.user_id
),
Retention AS (
  SELECT
    days_since_first_activity,
    COUNT(DISTINCT user_id) as retained_users,
    (SELECT COUNT(DISTINCT user_id) FROM FirstActivity) as total_users,
    COUNT(DISTINCT user_id) * 100.0 / (SELECT COUNT(DISTINCT user_id) FROM FirstActivity) as
retention_rate FROM ActivityIntervals
  WHERE days_since_first_activity > 0
  GROUP BY days_since_first_activity
)
SELECT * FROM Retention ORDER BY days_since_first_activity;
```

This query does the following:

FirstActivity: Identifies the first activity date for each user.

ActivityIntervals: Joins the user's activities with their first activity to compute the number of days since the first activity.

Retention: Calculates the retention rate for each interval since the first activity.

The DATEDIFF function is used to calculate the difference in days between activities, which assumes a daily retention calculation. Adjust the part inside DATEDIFF if you want to calculate weekly or monthly retention.

Ques.39 write year on year growth in SQL?

Ans. 39 To calculate the year-on-year (YoY) growth in SQL, you typically need data that includes a date and a metric to measure (e.g., sales, users, revenue). The basic idea is to compare the metric at a certain period with the same period in the previous year and calculate the percentage growth.

Assuming you have a table sales with columns sale_date (date of the sale) and amount (sale amount), here is how you could write a SQL query to calculate the YoY growth for each month:

```
WITH MonthlySales AS (
   SELECT
      DATE_PART('year', sale_date) AS year,
      DATE_PART('month', sale_date) AS month,
      SUM(amount) AS total_sales
   FROM
      sales
   GROUP BY
      1, 2
),

YearlyComparison AS (
   SELECT
      a.year,
        a.month,
        a.total_sales AS sales_current_year,
        b.total_sales AS sales_previous_year,
        (a.total_sales - b.total_sales) * 100.0 / b.total_sales AS yoy_growth_percentage
    FROM
        MonthlySales a
    LEFT JOIN
        MonthlySales b ON a.month = b.month AND a.year = b.year + 1
)
SELECT * FROM YearlyComparison
ORDER BY year, month;
```

MonthlySales CTE: This Common Table Expression computes the total sales for each month and year.

YearlyComparison CTE: This CTE joins the MonthlySales with itself. For each record in MonthlySales, it finds the corresponding record from the previous year (same month, previous year) and calculates the YoY growth.

Final SELECT: The final SELECT statement fetches the results, presenting the year, month, sales for the current year, sales for the previous year, and the YoY growth percentage.

The DATE_PART function is used to extract the year and month components from the sale_date. This function's exact name and syntax can vary between SQL dialects (e.g., YEAR(), MONTH() in some databases, or you might need to use EXTRACT(YEAR FROM sale_date)).

Ques.40 Differentiate between UNION and UNION ALL?

Ans.40

S.No.	UNION	UNION ALL
1.	It has the ability to remove duplicate rows from the table.	It cannot remove duplicate rows from the table.
2.	Here, the performance is slow because it also eliminates the duplicate rows.	Here, the performance is fast because it does not eliminate the duplicate values.
3.	Most of the people used Union operators.	Hardly users use this operator.

4.	Syntax:
	
	SELECT column_list FROM table1
	
	UNION
	
	SELECT column_list FROM table2;
		Syntax:
		
		SELECT column_list FROM table1
		
		UNION ALL
		
		SELECT column_list FROM table2

Ques.41 What is OLTP?

Ans.41 OLTP stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An essential attribute of an OLTP system is its ability to maintain concurrency. To avoid single points of failure, OLTP systems are often decentralized. These systems are usually designed for a large number of users who conduct short transactions. Database queries are usually simple, require sub-second response times, and return relatively few records. Here is an insight into the working of an OLTP system.

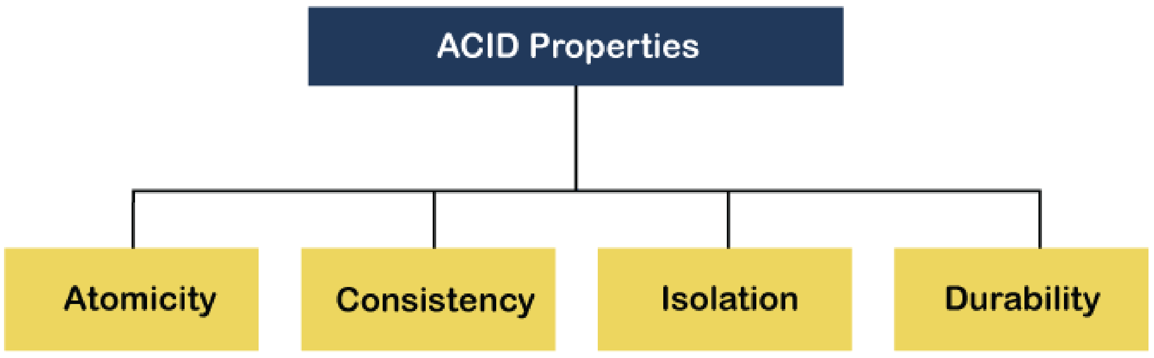
Ques.42 what is difference between OLAP and OLTP?

Ans.42 OLTP stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems often follow a decentralized architecture to avoid single points of failure. These systems are generally designed for a large audience of end-users who conduct short transactions. Queries involved in such databases are generally simple, need fast response times, and return relatively few records. A number of transactions per second acts as an effective measure for such systems.

OLAP stands for Online Analytical Processing, a class of software programs that are characterized by the relatively low frequency of online transactions. Queries are often too complex and involve a bunch of aggregations. For OLAP systems, the effectiveness measure relies highly on response time. Such systems are widely used for data mining or maintaining aggregated, historical data, usually in multi-dimensional schemas.

Ques.43 What are ACID properties?

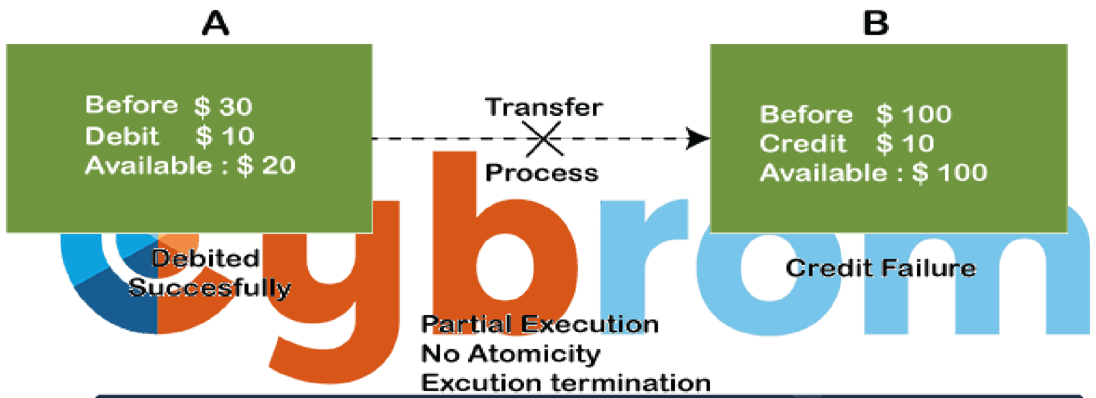
Ans.43



1) Atomicity

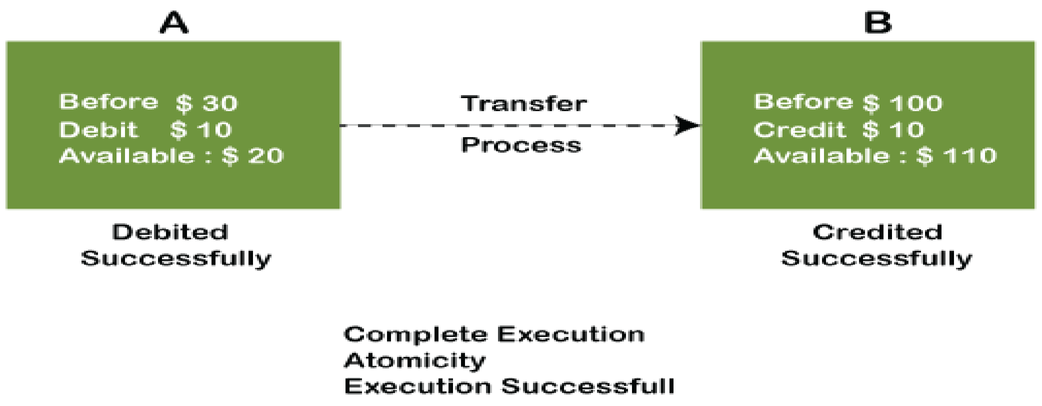
The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Example : If Remo has account A having $30 in his account from which he wishes to send $10 to Sheero's account, which is B. In account B, a sum of $ 100 is already present. When $10 will be transferred to account B, the sum will become $110. Now, there will be two operations that will take place. One is the amount of $10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account. Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails. Thus, in Remo's account A, the value becomes $20, and to that of Sheero's account, it remains $100 as it was previously present.



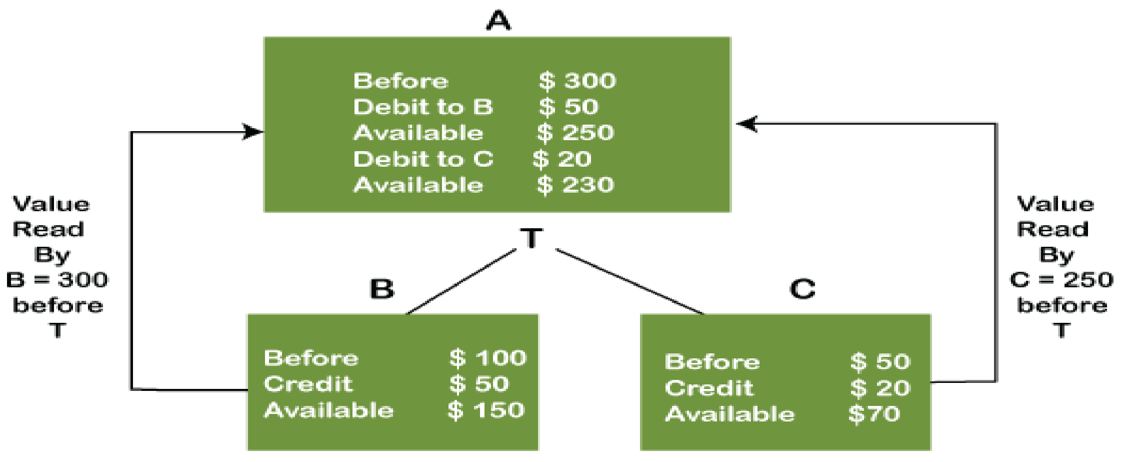
In the above diagram, it can be seen that after crediting $10, the amount is still $100 in account B. So, it is not an atomic transaction.

The below image shows that both debit and credit operations are done successfully. Thus the transaction is atomic.



2) Consistency

The word consistency means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.



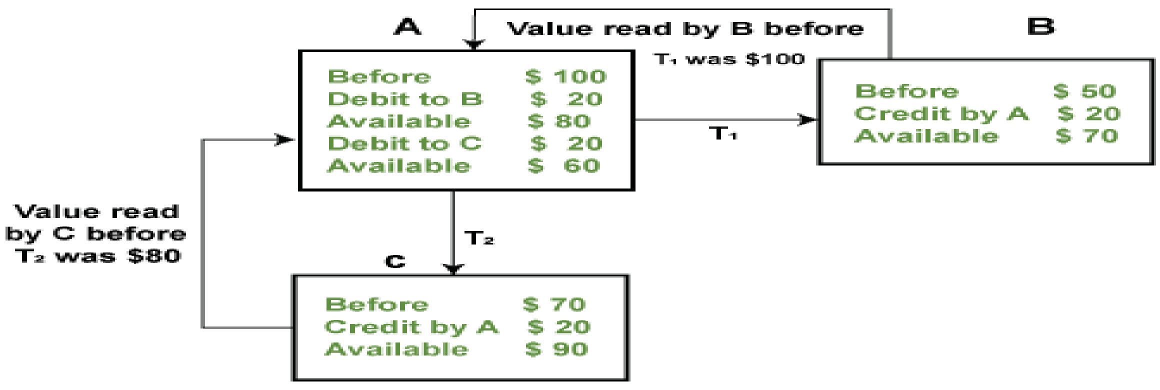
Data Consistent

In the above figure, there are three accounts, A, B, and C, where A is making a transaction T one by one to both B & C. There are two operations that take place, i.e., Debit and Credit. Account A firstly debits $50 to account B, and the amount in account A is read $300 by B before the transaction. After the successful transaction T, the available amount in B becomes $150. Now, A debits $20 to account C, and that time, the value read by C is $250 (that is correct as a debit of $50 has been successfully done to B). The debit and credit operation from account A to C has been done successfully. We can see that the transaction is done successfully, and the value is also read correctly. Thus, the data is consistent. In case the value read by B and C is $300, which means that data is inconsistent because when the debit operation executes, it will not be consistent.

3) Isolation

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Example: If two operations are concurrently running on two different accounts, then the value of both accounts should not get affected. The value should remain persistent. As you can see in the below diagram, account A is making T1 and T2 transactions to account B and C, but both are executing independently without affecting each other. It is known as Isolation.



Isolation - Independent execution of T_1 & T_2 by A

4) Durability

Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

Therefore, the ACID property of DBMS plays a vital role in maintaining the consistency and availability of data in the database.

Ques 44. What are Aggregate and Scalar functions?

Ans.44 An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- AVG() - Calculates the mean of a collection of values.
- COUNT() - Counts the total number of records in a specific table or view.
- MIN() - Calculates the minimum of a collection of values.
- MAX() - Calculates the maximum of a collection of values.
- SUM() - Calculates the sum of a collection of values.

Note: All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- LEN() - Calculates the total length of the given field (column).
- UCASE() - Converts a collection of string values to uppercase characters.
- LCASE() - Converts a collection of string values to lowercase characters.
- MID() - Extracts substrings from a collection of string values in a table.
- CONCAT() - Concatenates two or more strings.
- RAND() - Generates a random collection of numbers of a given length.
- ROUND() - Calculates the round-off integer value for a numeric field (or decimal point value).
- NOW() - Returns the current date & time.
- FORMAT() - Sets the format to display a collection of values

Ques.45 What is Cursor? How to use a Cursor?

Ans.45 A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

Working with SQL Cursor:

- DECLARE a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.
- Open cursor to initialize the result set. The OPEN statement must be called before fetching rows from the result set.
- FETCH statement to retrieve and move to the next row in the result set.
- Call the CLOSE statement to deactivate the cursor.
- Finally use the DEALLOCATE statement to delete the cursor definition and release the associated resources.

Ques.46 What is cardinality?

Ans.46 In data modelling, the cardinality of one data table with respect to another data table is a critical aspect of database design. Relationships between data tables define cardinality when explaining how each table links to another. In the relational model, tables can be related as any of: many-to-many, many-to-one (rev. one-to-many), or one-to-one. This is said to be the cardinality of a given table in relation to another.

In a database, cardinality usually represents the relationship between the data in two different tables by highlighting how many times a specific entity occurs in comparison to another.

High cardinality describes a data set that has a large number of unique values or entities. This represents a significant level of diversity and very little repetition. For example, a data set that lists the name of each unique customer would have high cardinality because the names are likely to vary.

Low cardinality refers to a data set that has a large quantity of the same values or entities. In a low cardinality data set, many of the same entities repeat themselves and there's less variety. For example, a data set that lists the category of each product for a small retail store may have low cardinality because there are only a few categories that are likely to repeat.

Why's cardinality important in databases?

Cardinality is important in databases because it creates links from one table or entity to another in a structured manner. This has a significant impact on the query execution plan, which is a sequence of steps users can take to search for and access data within a database system. Having a well-structured query execution plan can make it easier for users to locate the data they need quickly.

How's cardinality different from modality?

While cardinality and modality are both modeling concepts that professionals use in database design to analyze entities and their relationships with each other, there are some key differences between these two methods. Cardinality measures the maximum number of associations between two different table rows or columns. Alternatively, modality represents whether a relationship between two or more entities exists at all. In other words, modality focuses on the minimum number of associations, whether a relationship is mandatory and if the relationship is null.

Ques 47. What is conditional aggregation in SQL? How can you use it to obtain the results you want from grouped data?

Ans 47. Conditional aggregation involves using a CASE statement with aggregate functions such as SUM, COUNT, and AVG to include or exclude certain rows in an aggregation. You can use it to apply different aggregate functions with specific conditions, offering a high level of control over the final result set.

Ques 48. What are some factors that can affect a database's functionality?

Ans.48 Database design and schema. A database's design determines how data is organized and accessed. A well-structured schema with correct indexing and normalization enhances query performance.

Indexing. Using appropriate indexing on frequently queried columns can significantly improve the performance of your database.

Query optimization. Well-structured queries that use appropriate joins, filters, and grouping can improve performance. Optimizing your SQL queries with the proper syntax and avoiding operations like full table scans can also improve your database's efficiency.

Configuration and tuning. You can configure database settings, buffer sizes, memory allocation, and more to optimize database performance. Performing regular database tuning with actions like defragmenting indexes also helps to maintain database performance.

Additional factors that can impact performance include concurrent control, isolation levels, cashing, hardware, and network speed, among others.

Ques 49. Write a query that delivers the names of all employees who work in the same department as the employee with the highest salary.

Ans.49

```
SELECT e2.employee_name

FROM employees e1

JOIN employees e2 ON e1.department_id = e2.department_id

WHERE e1.salary = (SELECT MAX(salary) FROM employees);
```

Ques.50 Write a query to calculate the 7-day moving average of sales for each product in a given range using SQL window functions.

Ans. 50

```
SELECT product_id, sale_date,
AVG(sales_amount) OVER (PARTITION BY product_id ORDER BY sale_date ROWS BETWEEN 6
PRECEDING AND CURRENT ROW) AS moving_average  FROM sales;
```

Ques.51 Write a query to get the last record from a table?

Ans.51 select * from Student where RowID = select max(RowID) from Student;

https://medium.com/@nawal.ambavkar/top-tricky-sql-interview-problems-medium-to-hard-144bf5dc0886

open this link and solve all these Query question ?(Very Important)

Cybrom

TECHNOLOGY PVT. LTD.