DBMS NOTES

1.What is a DBMS?

Answer: A DBMS is software that manages databases, providing an interface to store, retrieve, and manipulate data efficiently and securely.

2. What are the different types of DBMS models?

Answer: The main types are:
Relational DBMS (RDBMS)
NoSQL DBMS (Document, Key-Value, Columnar, Graph)

3. Explain ACID properties in DBMS.

Answer: ACID stands for Atomicity, Consistency, Isolation, and Durability, which ensure database transactions are reliable and maintain data integrity.

Summary in Database Context

For the bank transfer example:

- Atomicity: Either both the debit and credit happen, or neither does.
- Consistency: The transfer follows all rules (e.g., no negative balances).
- Isolation: Simultaneous transfers don't interfere with each other.
- Durability: Once a transfer completes, the changes are permanently saved.

This example demonstrates how ACID properties keep database transactions reliable and consistent, especially in critical applications like banking.

Atomicity এর বাংলা অর্থ হলো "অখণ্ডতা" বা "অখণ্ডনীয়তা"।

Consistency এর বাংলা অর্থ হলো "সঙ্গতি" বা "নিয়মিততা"।

Isolation এর বাংলা অর্থ হলো "বিচ্ছিন্নতা" বা "স্বতন্ত্রতা"।

Durability এর বাংলা অর্থ হলো "স্থায়িত্ব" বা "স্থিতিশীলতা"।

4. What is normalization in DBMS?

Answer: Normalization is the process of organizing data to eliminate redundancy and improve data integrity in a relational database.

5. What is the difference between a primary key and a foreign key?

Answer: A primary key uniquely identifies a record in a table, while a foreign key refers to the primary key of another table, establishing a relationship between the two.

6. Explain the difference between DELETE and TRUNCATE in SQL.

Answer: DELETE removes rows from a table one by one with transaction logging, while TRUNCATE removes all rows at once without transaction logging.

7. What is the difference between INNER JOIN and OUTER JOIN?

Answer: INNER JOIN returns only matching rows from both tables, while OUTER JOIN (LEFT, RIGHT, FULL) returns matching rows as well as non-matching rows from one or both tables.

8. What is a stored procedure?
Answer: A stored procedure is a precompiled set of SQL statements that can be stored and executed on the database server to perform specific tasks.

9. Explain the difference between a database and a DBMS.

Answer: A database is a collection of related data, while a DBMS is software used to manage, store, and retrieve data efficiently from the database.

10. What is the purpose of an index in a database?
Answer: An index improves the speed of data retrieval operations by creating a data structure that allows the database to locate and access data quickly.

11. What is a deadlock, and how can it be avoided?
Answer: A deadlock occurs when two or more transactions are unable to proceed because each is waiting for the other to release a lock. Deadlocks can be avoided by using proper transaction management and lock acquisition strategies.

12. What is a trigger in DBMS?
Answer: A trigger is a set of actions that automatically execute in response to specific database events, such as insert, update, or delete operations on a table.

13. Explain the concept of data integrity in a DBMS.
Answer: Data integrity ensures that data is accurate, consistent, and valid throughout the database, preventing unauthorized or erroneous changes.

14. What is a self-join in SQL?
Answer: A self-join is a SQL query where a table is joined with itself to combine related rows based on a common column.

15. What is the difference between a heap table and a clustered table?
Answer: A heap table does not have a clustered index, while a clustered table stores data physically sorted based on the indexed column.

16. What are the advantages and disadvantages of denormalization?
Answer: Denormalization can improve read performance but may lead to data redundancy and increased complexity in maintaining data consistency.

17. Explain the concept of a transaction in DBMS.
Answer: A transaction is a sequence of one or more database operations that are treated as a single unit, ensuring data consistency and allowing rollback in case of failures.

18. What is the purpose of the COMMIT statement in SQL?
Answer: The COMMIT statement is used to permanently save all changes made during a transaction.

19. What is the difference between a clustered and non-clustered index?
Answer: A clustered index determines the physical order of data rows in a table, whereas a non-clustered index creates a separate structure for quick data access.

20. What are the advantages of using a NoSQL database over an RDBMS?
Answer: NoSQL databases offer flexibility, scalability, and better performance for unstructured or semi-structured data compared to traditional RDBMS.

21. Explain the CAP theorem in the context of distributed databases.
Answer: The CAP theorem states that it is impossible for a distributed database to simultaneously achieve Consistency, Availability, and Partition tolerance. It can satisfy at most two out of the three.

22. What are triggers in the context of database events?
Answer: Triggers are special stored procedures that are automatically executed in response to specific data manipulation events, like INSERT, UPDATE, or DELETE operations.

23. Explain the difference between optimistic and pessimistic locking.
Answer: Optimistic locking assumes no conflicts between transactions until the end, checking for conflicts only during the commit. Pessimistic locking acquires locks immediately, assuming conflicts might occur.

24. What is a materialized view in DBMS?
Answer: A materialized view is a database object that stores the result of a query, allowing faster data retrieval since the results are precomputed and stored.

25. What is a B-tree index?
Answer: A B-tree index is a self-balancing tree data structure used to speed up data retrieval by maintaining a sorted hierarchy of values.

26. Explain the concept of sharding in database systems.
Answer: Sharding involves splitting a large database into smaller, more manageable pieces (shards) to distribute the data across multiple servers or nodes for improved scalability and performance.

27. What is an Entity-Relationship Diagram (ERD)?
Answer: An ERD is a visual representation of the relationships among entities in a database, showing how different tables are connected.

28. How can you optimize the performance of a SQL query?
Answer: Query performance can be improved by adding indexes, optimizing table structures, and rewriting complex queries.

29. What is a transaction log in a database, and why is it essential?
Answer: The transaction log records all changes made to a database and is crucial for recovery in case of system failures or crashes.

30. Explain the concept of referential integrity in a relational database.
Answer: Referential integrity ensures that relationships between tables are maintained by enforcing valid foreign key constraints, preventing the creation of orphaned records . Orphaned typically happens when a foreign key (a field that references a key in another table) points to a record that has been deleted or doesn't exist.

31. Explain the purpose of the SQL JOIN clause in DBMS.
Answer: The **JOIN clause in SQL** is used to **combine rows from two or more tables** based on a related column between them. It allows data from different tables to be retrieved together to create more comprehensive result sets.

32.What are the advantages and disadvantages of using stored procedures in a DBMS?
Answer: Advantages of stored procedures include improved performance (as they are precompiled), reduced network traffic (since the code resides on the server), and enhanced security (as users only need execute permissions). However, disadvantages include increased complexity and maintenance overhead, limited portability across different database systems, and a potential lack of direct debugging capabilities.

33. What are the different types of locks in a DBMS, and how do they affect data concurrency?
Answer: The main types of locks are:

- **Shared Lock (Read Lock):** Allows multiple transactions to read data simultaneously but not modify it. Shared locks promote data read consistency, allowing multiple transactions to read the same data simultaneously.
- **Exclusive Lock (Write Lock):** Only one transaction can acquire this lock, preventing other transactions from reading or writing the locked data. Exclusive locks ensure data integrity by preventing multiple transactions from modifying the same data simultaneously.
- **Update Lock:** A hybrid lock that allows multiple transactions to read the data but prevents concurrent updates.
- **Intent Lock:** Indicates that a transaction intends to acquire a higher-level lock to protect a range of data.

Locks play a vital role in managing data concurrency. **Proper lock management and lock escalation techniques are essential for efficient data concurrency control.**

Q1. Difference between groupby and orderby?
Difference between GROUP BY and ORDER BY:

GROUP BY and ORDER BY are two distinct clauses used in SQL for different purposes:

GROUP BY:

- GROUP BY is used to group rows based on specific columns.
- It is typically used with aggregate functions like SUM, COUNT, AVG, etc., to perform operations on groups of rows rather than individual rows.
- When you use GROUP BY, the result set is divided into groups, and the aggregate functions are applied to each group, returning one row per group.
- The columns listed in the GROUP BY clause determine the grouping criteria, and all rows with the same values in those columns are grouped together.

Example:

```
SELECT department, COUNT(*) AS num_employees
FROM employees
GROUP BY department;
```

ORDER BY:

- ORDER BY is used to sort the result set based on one or more columns.
- It does not group rows but rather rearranges the rows in the output based on the specified sort criteria.
- By default, ORDER BY sorts the result in ascending order. You can use the DESC keyword to sort in descending order.

Example:

```
SELECT name, age
FROM students
ORDER BY age DESC;
```

In summary, GROUP BY is used for aggregation and dividing rows into groups, while ORDER BY is used for sorting the result set based on specified columns.

35 : What are Views in DBMS?

A view is a virtual table that is derived from one or more base tables or other views. It does not store any data itself but represents a tailored, pre-defined query that simplifies data retrieval. Views act as a layer of abstraction over the underlying tables, providing a more user-friendly and secure way to interact with the data.

Benefits of using views:

- **Data Abstraction**: Views allow users to work with a simplified representation of the data, hiding unnecessary details and complexity.
- **Security:** Views can be used to restrict access to certain columns or rows, providing a level of security by only showing specific data to specific users.
- **Simplified Querying:** Complex queries can be encapsulated within views, making it easier for users to retrieve the desired information without writing complex SQL statements.
- **Data Independence:** If the underlying schema changes, views can remain the same, and applications using the views will not be affected.

36 : How Database is accessed through Application Programs?
Ans - Apps (written in C++/JAVA) interact with DB using Interface/API. API is provided to send DML/DDL statements to DB and retrieve the results.

37. What is the role of Order by and Group by in SQL?
Ans - Order by clause is used to sort the result **set of a query based** on one or more columns in ascending or descending order. **Group by clause** is used to collect data from multiple records and group the result based on one or more columns. It is used with aggregation functions like count(), avg().

38.What is Alias in MySQL?
Ans - In MySQL, an alias is a temporary name assigned to a column, table, or expression used in a SQL query. It works as a Nickname for expressing the tables or column names.

39 - What are Views in MYSQL?
Ans - A view is a **database object** that has **no values**. It contains **rows and columns similar to a real table but does not contain any data of its own.**

40 - What is a Functional Dependency (FD)?
A **Functional Dependency (FD)** is a relationship between two sets of attributes in a database. It means that if you know the value of one attribute (or a combination of attributes), you can determine the value of another attribute. In simpler terms, if attribute **A** determines attribute **B**, then **B** is functionally dependent on **A**.

41 - What are Rules of FD(Armstrong's Axioms)
Ans - 1. Reflexive, 2. Augmentation, 3. Transitivity

42 - What are the different types of anomalies introduced by data redundancy?
Ans - There are **3 types of anomalies** introduced by data redundancy - 1. Insertion Anomaly, 2. Deletion Anomaly, 3. Updation Anomaly.

43 - How to implement Atomicity and Durability in Transactions?

Ans - **Method 1 - Shadow Copy Scheme** - Based on making copies of DB(aka shadow copies).
Method 2 - Log Based Recovery Methods - Maintaining Logs of each transaction in some stable storage for easy recovery in case of failures.

44 - Are NoSQL Databases always relational? Do they never follow ACID Properties?
Ans - False. **NoSql databases can store relational Data**. MongoDB supports ACID Transactions, unlike other NoSQL Databases.

45 - What are the different types of NoSQL Data Models?
Ans - 1. Key-Value Stores, 2. Column-Oriented, 3. Graph-based stores, 4. Document-based stores.

46. Mention the issues with traditional file-based systems that make DBMS a better choice?

- **Slow Data Access:** Lack of indexing forces full-page scans, making data retrieval slow.
- **Data Redundancy & Inconsistency:** Duplicate data across files leads to inconsistencies when updated.
- **Disorganized Data:** Traditional systems lack structure, making data access more difficult.
- **No Concurrency Control:** One operation can lock the entire file, unlike DBMS where multiple operations can run concurrently.
- **Missing Features:** No integrity checks, data isolation, atomicity, or security features present in traditional systems.

47.Explain a few advantages of a DBMS.

- **Data Sharing:** Multiple users can access the same database simultaneously.
- **Integrity Constraints:** Ensure refined and accurate data storage.
- **Data Redundancy Control:** Combines data into a single database to minimize duplication.
- **Data Independence:** Structure changes do not affect running applications.
- **Backup and Recovery:** Automatically backs up data and restores it when needed.
- **Data Security:** Ensures secure data storage and transfer through authentication and encryption.

48.Explain different languages present in DBMS.

DDL (Data Definition Language):

- Defines and modifies database structure.
- Commands: CREATE, ALTER, DROP, TRUNCATE, RENAME.

DML (Data Manipulation Language):

- Manipulates data within the database.
- Commands: SELECT, UPDATE, INSERT, DELETE.

DCL (Data Control Language):

- Manages user permissions and access controls.
- Commands: GRANT, REVOKE.

TCL (Transaction Control Language):

- Manages database transactions.
- Commands: COMMIT, ROLLBACK, SAVEPOINT.

49.Are NULL values in a database the same as that of blank space or zero?

Answer : No, a NULL value is distinct from zero and blank space in that it denotes a value that is assigned, unknown, unavailable, or not applicable, as opposed to blank space, which denotes a character, and zero, which denotes a number.
For instance, a NULL value in "number of courses" taken by a student indicates that the value is unknown, but a value of 0 indicates that the student has not taken any courses.

50. What are super, primary, candidate, and foreign keys?

- **Super Key:** A set of attributes in a table where all other attributes depend on it. The values in a super key are unique for every row.
- **Candidate Key:** The smallest possible super key. It has no unnecessary attributes, and no part of it can form another super key.
- **Primary Key:** One of the candidate keys chosen as the main unique identifier for the table. A table can only have one primary key.
- **Foreign Key:** A field (or set of fields) in one table that links to the primary key in another table, creating a relationship between them.

51. What is the difference between primary key and unique constraints?

Although the primary key cannot have a NULL value, the unique constraints can. A table has just one main key, but it might have numerous unique constraints.

52. Explain RDBMS with examples.

Ans : RDBMS stands for Relational Database Management System, and it was first introduced in the 1970s to make it easier to access and store data than DBMS. In contrast to DBMS, which stores data as files, RDBMS stores data as tables. When opposed to DBMS, storing data as rows and columns makes it easier to locate specific values in the database and makes it more efficient. MySQL, Oracle DB, and other prominent RDBMS systems are examples.

53.What is a checkpoint in DBMS?
The Checkpoint is a technique that removes all previous logs from the system and stores them permanently on the storage drive.

54-What do you mean by Data Model?
A **data model** consists of a set of tools for **describing data, semantics, and constraints**. They also assist in the description of the relationship between data entities and their attributes.

55-When does checkpoint occur in DBMS?
A **checkpoint is a snapshot of the database management system's current state. after system crash**, checkpoints are utilised to **recover the database**. The log-based recovery solution employs checkpoints. When we need to restart the system because of a system crash, we use checkpoints. As a result, we won't have to execute the transactions from the beginning.

56-What is the difference between an entity and an attribute?
In a database, an entity is a real-world thing. Employee, designation, department, and so on are examples of different entities in an employee database.
A trait that describes an entity is called an attribute. For example, the entity "employee" can have properties such as name, ID, and age.

57- What are the various kinds of interactions catered by DBMS?
DBMS can handle a variety of interactions, including: **Data definition, Update, Retrieval, Administration .**

58- What are the different levels of abstraction in the DBMS?

- **Physical Level :** The physical level of abstraction specifies how data is stored and is the lowest degree of abstraction.
- **Logical Layer :** After the Physical level, there is the Logical level of abstraction. This layer decides what data is saved in the database and how the data pieces relate to one another**.**
- **View Level:** The greatest level of abstraction, the View Level describes only a portion of the entire database.

59- What is an entity-relationship model?

It's a diagrammatic approach to database architecture in which real-world things are represented as entities and relationships between them are mentioned. This method allows the DBA staff to quickly grasp the schema.

60 .What do you understand by the terms Entity, Entity Type, and Entity Set in DBMS?

- **Entity:** An entity is a real-world object with attributes, which are nothing more than the object's qualities. An employee, for example, is a type of entity. This entity can have attributes like empid, empname, and so on.
- **Entity Type:** An entity type is a collection of entities with similar attributes. An entity type, in general, refers to one or more related tables in a database. As a result, entity type can be thought of as a trait that uniquely identifies an entity. Employees can have attributes such as empid, empname, department, and so on
- **Entity Set:** In a database, an entity set is a collection of all the entities of a specific entity type.An entity set can include, for example, a group of employees, a group of companies, and a group of persons**.**

61.What is Relationship? An association between two or more entities is characterised as a relationship. In a database management system, there are three types of relationships:

- **One-to-One:** In this case, one record of any object can be linked to another object's record.
- **One-to-Many (many-to-one):** In this case, one record of any object can be linked to many records of other objects, and vice versa.
- **Many-to-many:** In this case, multiple records of one item can be linked to n records of another object.

62. What is Data Abstraction in DBMS?
In a database management system, data abstraction is the process of hiding unimportant facts from users. Because database systems are made up of complicated data structures, user interaction with the database is made possible .

63- What is the difference between having and where clause?
In a select statement, HAVING is used to establish a condition for a group or an aggregate function. Before grouping, the WHERE clause picks.

After grouping, the HAVING clause picks rows. The WHERE clause, cannot contain aggregate functions.

64- What is a transaction? What are ACID properties?
A **database transaction** is a collection of database operations that must be handled as a whole, meaning that all or none of the **actions must be executed**. **A bank transaction from one account to another is a good illustration.** Either both debit and credit operations must be completed, or none of them must be completed. The ACID qualities (Atomicity, Consistency, Isolation, and Durability) ensure that database transactions are processed reliably.

65- What is Denormalization?
Denormalization is a database optimization method in which duplicated data is added to one or more tables.

66- What is the use of the DROP command and what are the differences between DROP, TRUNCATE and DELETE commands?

- **Drop :** Permanently deletes the entire table or database, Cannot be undone.
- **TRUNCATE :** Removes all rows from a table but keeps the table structure intact.Cannot be undone and is faster than DELETE.
- **DELETE :** Removes specific rows from a table based on a condition.Can be undone if used with a transaction (ROLLBACK).

67- What is the concept of sub-query in terms of SQL?
A sub-query is a query that is contained within another query. It is also known as an inner query because it is found within the outer query.

68-What is E-R model in the DBMS?
In relational databases, the E-R model is known as an Entity-Relationship model, and it is built on the concept of Entities and the relationships that exist between them.

Q 5. What is the difference between a primary key and a unique key?

Ans: A primary key is used to uniquely identify a row in a table and must have a unique value. On the other hand, a unique key ensures that a column or combination of columns has a unique value but does not necessarily identify the row.

Q 7. What are the different types of normalization?

Ans: The different types of normalization are:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF) or Project-Join Normal Form (PJNF)

Q 9. What is the difference between DELETE and TRUNCATE in SQL?

Ans: The **DELETE statement** is used to remove specific rows from a table based on a condition. It can be rolled back and generates individual delete operations for each row.

TRUNCATE, on the other hand, is used to remove all rows from a table. It cannot be rolled back, and it is faster than DELETE as it deallocates the data pages instead of logging individual row deletions.

Q 10. What is the difference between UNION and UNION ALL?

Ans: UNION and UNION ALL are used to combine the result sets of two or more SELECT statements.

UNION removes duplicate rows from the combined result set.

whereas **UNION ALL** includes all rows, including duplicates.

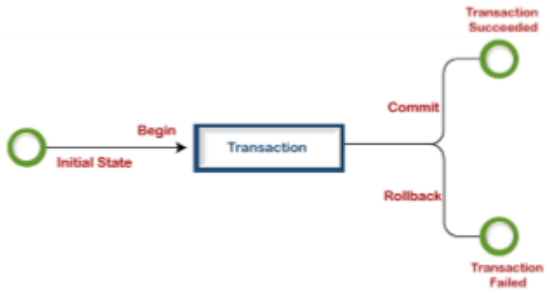
Q 11. What is the difference between the HAVING clause and the WHERE clause?

Ans: **The WHERE clause** is used to filter rows based on a condition before the data is grouped or aggregated. It operates on individual rows.

The HAVING clause, on the other hand, is used to filter grouped rows based on a condition after the data is grouped or aggregated using the GROUP BY clause.

Q 12. What is a transaction in SQL?

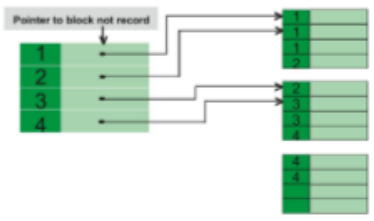
Ans: A transaction is a sequence of SQL statements that are executed as a single logical unit of work. It ensures data consistency and integrity by either committing all changes or rolling them back if an error occurs.



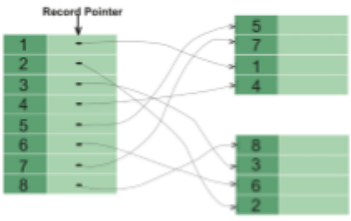
Q 13. What is the difference between a clustered and a non-clustered index?

Ans: A **clustered index** determines the physical order of data in a table. It changes the way the data is stored on disk and can be created on only one column. A table can have only one clustered index.

A **non-clustered index** does not affect the physical order of data in a table. It is stored separately and contains a pointer to the actual data. A table can have multiple non-clustered indexes.



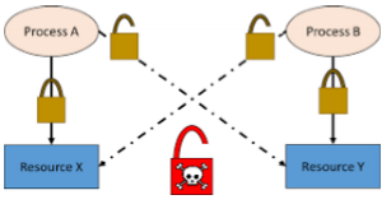
Clustered Index



Non-Clustered Index

Q 15. What is a deadlock?

Ans: A deadlock occurs when two or more transactions are waiting for each other to release resources, resulting in a circular dependency. As a result, none of the transactions can proceed, and the system may become unresponsive.



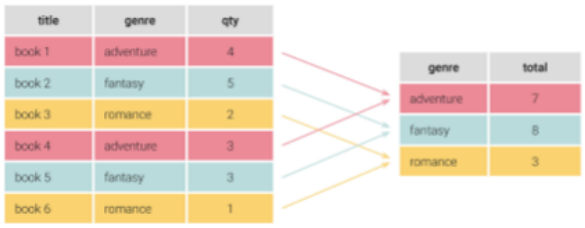
Q 16. What is the difference between a database and a schema?

Ans: A database is a container that holds multiple objects, such as tables, views, indexes, and procedures. It represents a logical grouping of related data.

A schema, on the other hand, is a container within a database that holds objects and defines their ownership. It provides a way to organize and manage database objects.

Q 18. What is the purpose of the GROUP BY clause?

Ans: The GROUP BY clause is used to group rows based on one or more columns in a table. It is typically used in conjunction with aggregate functions, such as SUM, AVG, COUNT, etc., to perform calculations on grouped data.

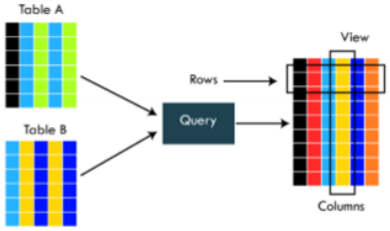


title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1

genre	total
adventure	7
fantasy	8
romance	3

Q 22. What is a view?

Ans: A view is a virtual table based on the result of an SQL statement. It allows users to retrieve and manipulate data as if



Q 23. What is the difference between a cross join and an inner join?

Ans: A cross join (Cartesian product) returns the combination of all rows from two or more tables.

An inner join returns only the matching rows based on a join condition.

Q 24. What is the purpose of the COMMIT statement?

Ans: The COMMIT statement is used to save changes made in a transaction permanently. It ends the transaction and makes the changes visible to other users.

Q 25. What is the purpose of the ROLLBACK statement?

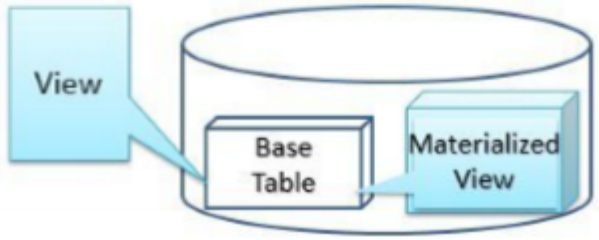
Ans: The ROLLBACK statement is used to undo changes made in a transaction. It reverts the database to its previous state before the transaction started.

Q 26. What is the purpose of the NULL value in SQL?

Ans: NULL represents the absence of a value or unknown value. It is different from zero or an empty string and requires special handling in SQL queries.

Q 27. What is the difference between a view and a materialized view?

Ans: A materialized view is a physical copy of the view's result set stored in the database, which is updated periodically. It improves query performance at the cost of data freshness.

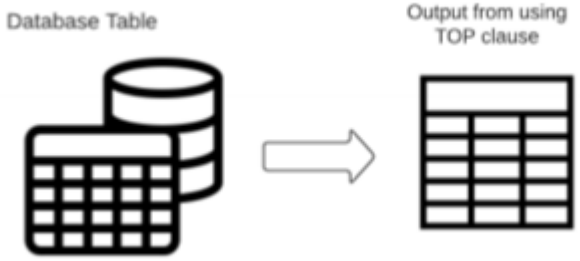


Q 31. What is the difference between the IN and EXISTS operators?

Ans: The IN operator checks for a value within a set of values or the result of a subquery. The EXISTS operator checks for the existence of rows returned by a subquery.

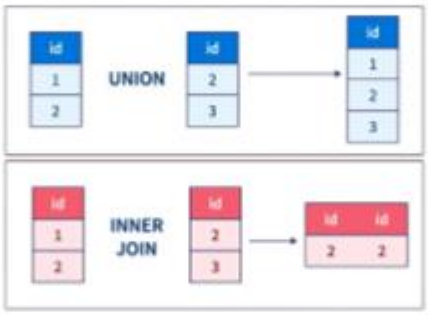
Q 34. What is the purpose of the TOP or LIMIT clause?

Ans: The TOP (in SQL Server) or LIMIT (in MySQL) clause is used to limit the number of rows returned by a query. It is often used with an ORDER BY clause.



Q 35. What is the difference between the UNION and JOIN operators?

Ans: UNION combines the result sets of two or more SELECT statements vertically, while JOIN combines columns from two or more tables horizontally based on a join condition.

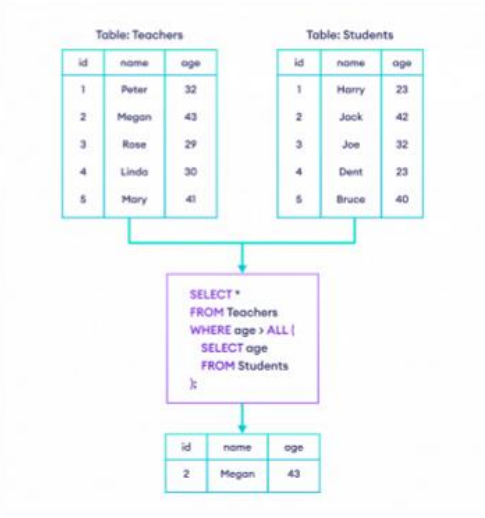


Q 36. What is a data warehouse?

Ans: A data warehouse is a large, centralized repository that stores and manages data from various sources. It is designed for efficient reporting, analysis, and business intelligence purposes.

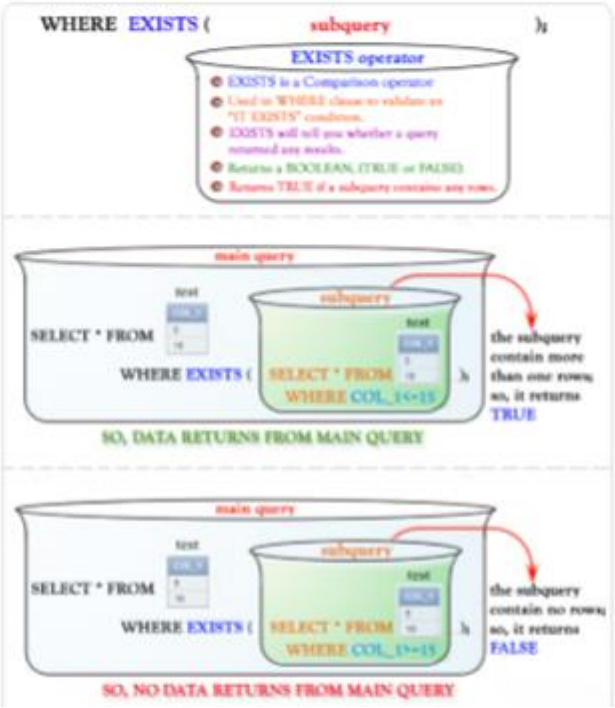
Q 45. What is the purpose of the ALL keyword in SQL?

Ans: The CASCADE DELETE constraint is used to automatically delete related rows in child tables when a row in the parent table is deleted.

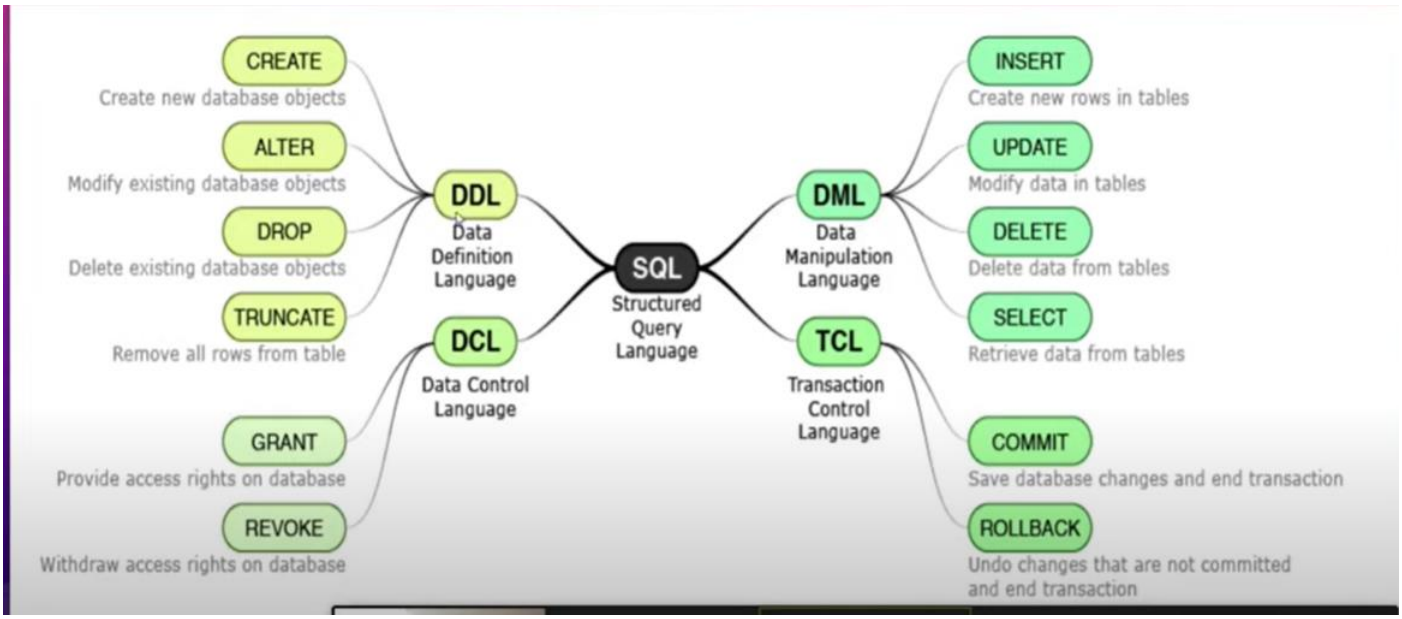


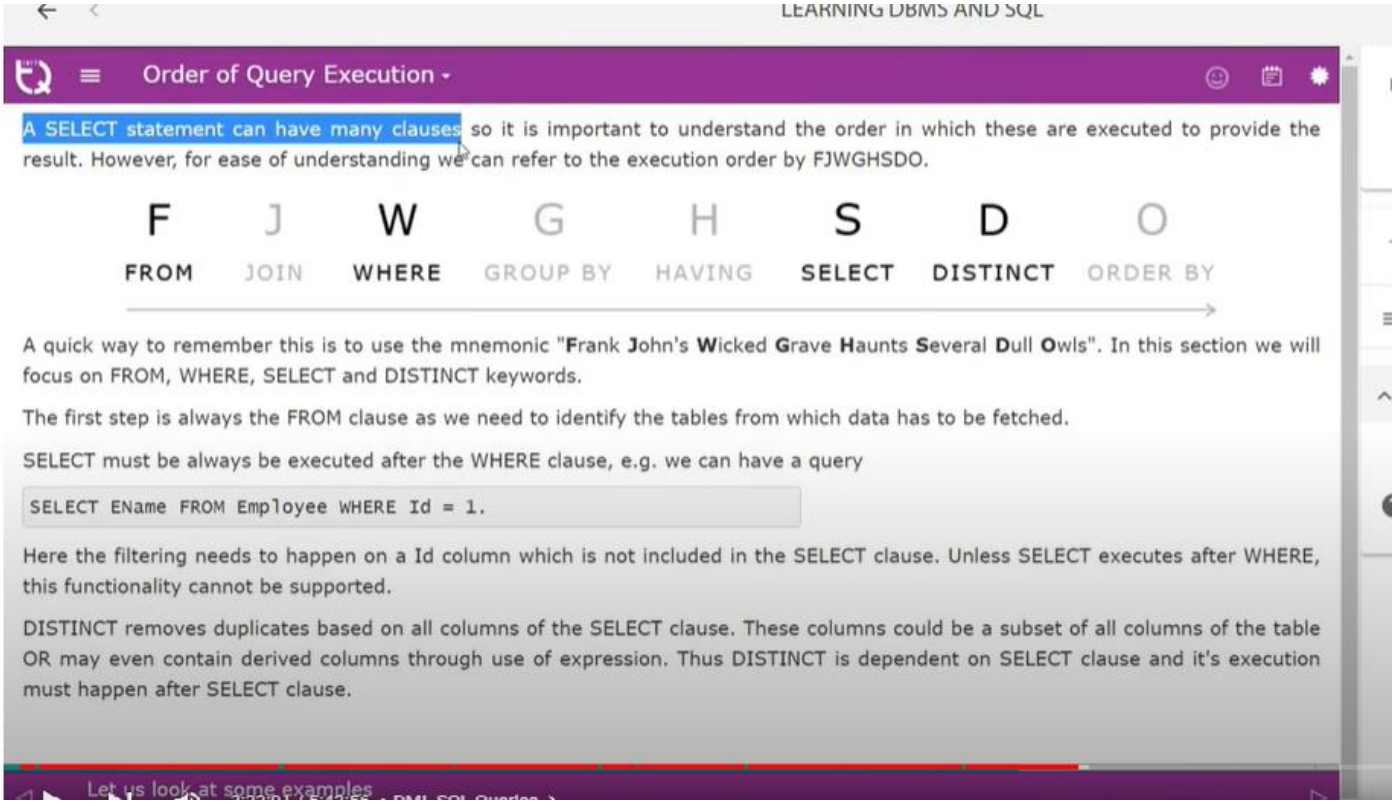
Q 46. What is the difference between the EXISTS and NOT EXISTS operators?

Ans: The EXISTS operator returns true if a subquery returns any rows, while the NOT EXISTS operator returns true if a subquery returns no rows.



SQL NOTES


SQL
Structured Query Language
DDL
Data Definition Language
DML
Data Manipulation Language
DCL
Data Control Language
TCL
Transaction Control Language
CREATE
Create new database objects
ALTER
Modify existing database objects
DROP
Delete existing database objects
TRUNCATE
Remove all rows from table
GRANT
Provide access rights on database
REVOKE
Withdraw access rights on database
INSERT
Create new rows in tables
UPDATE
Modify data in tables
DELETE
Delete data from tables
SELECT
Retrieve data from tables
COMMIT
Save database changes and end transaction
ROLLBACK
Undo changes that are not committed and end transaction


LEARNING DBMS AND SQL
Order of Query Execution
A SELECT statement can have many clauses so it is important to understand the order in which these are executed to provide the result. However, for ease of understanding we can refer to the execution order by FJWGHSDO.
F J W G H S D O
FROM JOIN WHERE GROUP BY HAVING SELECT DISTINCT ORDER BY
A quick way to remember this is to use the mnemonic "Frank John's Wicked Grave Haunts Several Dull Owls". In this section we will focus on FROM, WHERE, SELECT and DISTINCT keywords.
The first step is always the FROM clause as we need to identify the tables from which data has to be fetched.
SELECT must be always be executed after the WHERE clause, e.g. we can have a query
SELECT EName FROM Employee WHERE Id = 1.
Here the filtering needs to happen on a Id column which is not included in the SELECT clause. Unless SELECT executes after WHERE, this functionality cannot be supported.
DISTINCT removes duplicates based on all columns of the SELECT clause. These columns could be a subset of all columns of the table OR may even contain derived columns through use of expression. Thus DISTINCT is dependent on SELECT clause and it's execution must happen after SELECT clause.
Let us look at some examples

SQL Basics Cheat Sheet

LearnSQL.com

SQL, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

COUNTRY

id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY

id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

QUERYING SINGLE TABLE

Fetch all columns from the `country` table:

```
SELECT *
FROM country;
```

Fetch `id` and `name` columns from the `city` table:

```
SELECT id, name
FROM city;
```

Fetch city names sorted by the `rating` column in the default ASCending order:

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

Fetch city names sorted by the `rating` column in the DESCending order:

```
SELECT name
FROM city
ORDER BY rating DESC;
```

ALIASES

COLUMNS

```
SELECT name AS city_name
FROM city;
```

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

FILTERING THE OUTPUT

COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
  AND name != 'Madrid';
```

TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%'
  OR name LIKE '%s';
```

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

Fetch names of cities that don't miss a rating value:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

Fetch names of cities that are in countries with IDs 1, 4, 7, or 8:

```
SELECT name
FROM city
WHERE country_id IN (1, 4, 7, 8);
```

QUERYING MULTIPLE TABLES

INNER JOIN

`JOIN` (or explicitly **`INNER JOIN`**) returns rows that have matching values in both tables.

```
SELECT city.name, country.name
FROM city
[INNER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	3	Iceland

LEFT JOIN

`LEFT JOIN` returns all rows from the left table with corresponding rows from the right table. If there's no matching row, **`NULL`**s are returned as values from the second table.

```
SELECT city.name, country.name
FROM city
LEFT JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL

RIGHT JOIN

`RIGHT JOIN` returns all rows from the right table with corresponding rows from the left table. If there's no matching row, **`NULL`**s are returned as values from the left table.

```
SELECT city.name, country.name
FROM city
RIGHT JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
NULL	NULL	NULL	3	Iceland

FULL JOIN

`FULL JOIN` (or explicitly **`FULL OUTER JOIN`**) returns all rows from both tables – if there's no matching row in the second table, **`NULL`**s are returned.

```
SELECT city.name, country.name
FROM city
FULL [OUTER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL
NULL	NULL	NULL	3	Iceland

CROSS JOIN

`CROSS JOIN` returns all possible combinations of rows from both tables. There are two syntaxes available.

```
SELECT city.name, country.name
FROM city
CROSS JOIN country;

SELECT city.name, country.name
FROM city, country;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
1	Paris	1	2	Germany
2	Berlin	2	1	France
2	Berlin	2	2	Germany

NATURAL JOIN

`NATURAL JOIN` will join tables by all columns with the same name.

```
SELECT city.name, country.name
FROM city
NATURAL JOIN country;
```

CITY			COUNTRY	
country_id	id	name	name	id
6	6	San Marino	San Marino	6
7	7	Vatican City	Vatican City	7
5	9	Greece	Greece	9
10	11	Monaco	Monaco	10

`NATURAL JOIN` used these columns to match rows: **`city.id`**, **`city.name`**, **`country.id`**, **`country.name`**.
`NATURAL JOIN` is very rarely used in practice.

Try out the interactive **SQL Basics** course at **LearnSQL.com**, and check out our other SQL courses.

LearnSQL.com is owned by Vertabelo SA
vertabelo.com | CC BY-NC-ND Vertabelo SA

SQL Basics Cheat Sheet

LearnSQL.com

AGGREGATION AND GROUPING

`GROUP BY` **groups** together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY

id	name	country_id
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	Munich	2
3	Warsaw	4
105	Cracow	4

→

CITY	
country_id	count
1	3
2	3
4	2

AGGREGATE FUNCTIONS

- `avg(expr)` – average value for rows within the group
- `count(expr)` – count of values for rows within the group
- `max(expr)` – maximum value within the group
- `min(expr)` – minimum value within the group
- `sum(expr)` – sum of values within the group

EXAMPLE QUERIES

Find out the number of cities:

```
SELECT COUNT(*)
FROM city;
```

Find out the number of cities with non-null ratings:

```
SELECT COUNT(rating)
FROM city;
```

Find out the number of distinctive country values:

```
SELECT COUNT(DISTINCT country_id)
FROM city;
```

Find out the smallest and the greatest country populations:

```
SELECT MIN(population), MAX(population)
FROM country;
```

Find out the total population of cities in respective countries:

```
SELECT country_id, SUM(population)
FROM city
GROUP BY country_id;
```

Find out the average rating for cities in respective countries if the average is above 3.0:

```
SELECT country_id, AVG(rating)
FROM city
GROUP BY country_id
HAVING AVG(rating) > 3.0;
```

SUBQUERIES

A subquery is a query that is nested inside another query, or inside another subquery. There are different types of subqueries.

SINGLE VALUE

The simplest subquery returns exactly one column and exactly one row. It can be used with comparison operators `=`, `<`, `<=`, `>`, or `>=`.
This query finds cities with the same rating as Paris:

```
SELECT name
FROM city
WHERE rating = (
  SELECT rating
  FROM city
  WHERE name = 'Paris'
);
```

MULTIPLE VALUES

A subquery can also return multiple columns or multiple rows. Such subqueries can be used with operators `IN`, `EXISTS`, `ALL`, or `ANY`.
This query finds cities in countries that have a population above 20M:

```
SELECT name
FROM city
WHERE country_id IN (
  SELECT country_id
  FROM country
  WHERE population > 20000000
);
```

CORRELATED

A correlated subquery refers to the tables introduced in the outer query. A correlated subquery depends on the outer query. It cannot be run independently from the outer query.
This query finds cities with a population greater than the average population in the country:

```
SELECT *
FROM city main_city
WHERE population > (
  SELECT AVG(population)
  FROM city average_city
  WHERE average_city.country_id = main_city.country_id
);
```

This query finds countries that have at least one city:

```
SELECT name
FROM country
WHERE EXISTS (
  SELECT *
  FROM city
  WHERE country_id = country.id
);
```

SET OPERATIONS

Set operations are used to combine the results of two or more queries into a single result. The combined queries must return the same number of columns and compatible data types. The names of the corresponding columns can be different.

CYCLING

id	name	country
1	YK	DE
2	ZG	DE
3	WT	PL
...	...	...

SKATING

id	name	country
1	YK	DE
2	DF	DE
3	AK	PL
...	...	...

UNION

`UNION` combines the results of two result sets and removes duplicates. **`UNION ALL`** doesn't remove duplicate rows.

This query displays German cyclists together with German skaters:

```
SELECT name
FROM cycling
WHERE country = 'DE'
UNION / UNION ALL
SELECT name
FROM skating
WHERE country = 'DE';
```

INTERSECT

`INTERSECT` returns only rows that appear in both result sets.

This query displays German cyclists who are also German skaters at the same time:

```
SELECT name
FROM cycling
WHERE country = 'DE'
INTERSECT
SELECT name
FROM skating
WHERE country = 'DE';
```

EXCEPT

`EXCEPT` returns only the rows that appear in the first result set but do not appear in the second result set.

This query displays German cyclists unless they are also German skaters at the same time:

```
SELECT name
FROM cycling
WHERE country = 'DE'
EXCEPT / MINUS
SELECT name
FROM skating
WHERE country = 'DE';
```

Try out the interactive **SQL Basics** course at **LearnSQL.com**, and check out our other SQL courses.

LearnSQL.com is owned by Vertabelo SA
vertabelo.com | CC BY-NC-ND Vertabelo SA

BOSSCODER ACADEMY

MOST ASKED SQL INTERVIEW QUESTIONS

at MAANG Companies

Disclaimer

Everyone learns uniquely.

Master SQL concepts by through these important interview questions.

Take the help of this doc, and crack your SQL questions like a pro.

Medium-Level SQL Questions

Q1.

Explain the difference between INNER JOIN and LEFT JOIN.

Ans.

An **‘INNER JOIN’** returns only the rows where there is a match in both tables. A **‘LEFT JOIN’** returns all rows from the left table and matched rows from the right table, and **‘NULL’** for non-matching rows in the right table.

Q2.

What is the purpose of the GROUP BY clause?

Ans.

The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like "find the number of customers in each country."

Q3.

How can you delete duplicate rows in a SQL table?

Ans.

```sql
DELETE FROM your_table
WHERE id NOT IN (
  SELECT MIN(id)
  FROM your_table
  GROUP BY column_with_duplicates
);
```

Q4.

What are subqueries and how are they used?

Ans.

Subqueries are nested queries used within a main query to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Q5.

Explain the concept of a ‘VIEW’ in SQL.

Ans.

A ‘VIEW’ is a virtual table based on the result set of an SQL query. It can simplify complex queries, improve security by restricting data access, and present a consistent, logical view of the data.

Q6.

What is a ‘UNION’ operator?

Ans.

The ‘UNION’ operator is used to combine the result sets of two or more ‘SELECT’ statements. Each ‘SELECT’ statement must have the same number of columns in the result sets with similar data types.

Q7.

How do you use the ‘CASE’ statement in SQL?

Ans.

```sql
SELECT name,
  CASE
    WHEN score >= 90 THEN 'A'
    WHEN score >= 80 THEN 'B'
    ELSE 'C'
  END as grade
FROM students;
```

Q8.

What is an ‘INDEX’ and why is it important?

Ans.

An index is a database object that improves the speed of data retrieval operations on a table at the cost of additional writes and storage space.

Q9.

Describe the use of ‘TRIGGERS’ in SQL.

Ans.

Triggers are database objects that automatically execute a specified SQL procedure when a certain event occurs on a table or view, such as an ‘INSERT’, ‘UPDATE’, or ‘DELETE’.

Q10.

Explain the concept of ‘NORMALIZATION’.

Ans.

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.

Advanced SQL Questions

Q11.

What is a ‘CTE’ (Common Table Expression)?

Ans.

```sql
WITH Sales_CTE (SalesPerson, TotalSales) AS (
  SELECT SalesPerson, SUM(SalesAmount)
  FROM Sales
  GROUP BY SalesPerson
)
SELECT * FROM Sales_CTE;
```

Q12.

Explain the differences between ‘DELETE’, ‘TRUNCATE’, and ‘DROP’ commands.

Ans.

‘DELETE’: Removes rows one at a time and logs individual row deletions. It can include a WHERE clause.

‘TRUNCATE’: Removes all rows from a table without logging individual row deletions. Cannot include a ‘WHERE’ clause.

‘DROP’: Removes a table from the database entirely, including its structure.

Q13.

How do you optimize a slow-running query?

Ans.

Techniques include indexing, query rewriting, reducing the number of joins, avoiding SELECT *, and analyzing the execution plan.

Q14.

What are ‘window functions’ in SQL?

Ans.

```sql
SELECT name, department,
       salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank
FROM employees;
```

Q15.

Discuss the concept of ‘ACID’ properties in transactions.

Ans.

Atomicity: Ensures that all operations within a transaction are completed; if not, the transaction is aborted.

Consistency: Ensures that a transaction brings the database from one valid state to another.

Isolation: Ensures that transactions are securely and independently processed at the same time without interference.

Durability: Ensures that once a transaction is committed, it will remain so, even in the event of a power loss, crash, or error.

Q16.

How do you handle transactions in SQL?

Ans.

```sql
BEGIN TRANSACTION;
-- SQL statements
COMMIT;
```

Q17.

What is the difference between ‘HAVING’ and ‘WHERE’ clauses?

Ans.

‘WHERE’ is used to filter records before any groupings are made, while ‘HAVING’ is used to filter values after grouping.

Q18.

Explain the concept of ‘sharding’ in databases.

Ans.

Sharding is a type of database partitioning that splits large databases into smaller, more manageable pieces, called shards, which can be spread across multiple servers.

Q19.

What are ‘stored procedures’ and how are they used?

Ans.

Stored procedures are precompiled collections of SQL statements stored under a name and processed as a unit. They can accept parameters and are used to perform tasks like data validation or access control.

Q20.

Explain the concept of ‘Data Warehousing’.

Ans.

Data warehousing involves the collection, storage, and management of large volumes of data from multiple sources to provide meaningful business insights. It typically involves ETL processes and uses schemas like star or snowflake schema.

Scenario-Based SQL Questions

Scenario 1:

Employee Management System

Tables:

- Employees

```sql
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  DepartmentID INT,
  Salary DECIMAL(10, 2),
  HireDate DATE
);
```

- Departments

```sql
CREATE TABLE Departments (
  DepartmentID INT PRIMARY KEY,
  DepartmentName VARCHAR(50)
);
```

Questions:

Q21.

Find the second highest salary in the Employees table.

Ans.

```sql
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

Q22.

List all employees who have been hired in the last year.

Ans.

```sql
SELECT * FROM Employees
WHERE HireDate >= DATEADD(year, -1, GETDATE());
```

Q23.

Find the average salary for each department.

Ans.

```sql
SELECT d.DepartmentName, AVG(e.Salary) AS AverageSalary
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID
GROUP BY d.DepartmentName;
```

Q24.

List all departments along with the count of employees in each department.

Ans.

```sql
SELECT d.DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount
FROM Departments d
LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
GROUP BY d.DepartmentName;
```

Scenario 2:

Online Retail Database

Tables:

- Orders

```sql
CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  OrderDate DATE,
  TotalAmount DECIMAL(10, 2)
);
```

- OrderDetails

```sql
CREATE TABLE OrderDetails (
  OrderDetailID INT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT,
  Price DECIMAL(10, 2)
);
```

- **Products**

```sql
CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  CategoryID INT,
  Price DECIMAL(10, 2)
);
```

Questions:

Q25.

Find the top 3 most sold products in the last month.

Ans.

```sql
SELECT TOP 3 p.ProductName, SUM(od.Quantity) AS TotalSold
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Orders o ON od.OrderID = o.OrderID
WHERE o.OrderDate >= DATEADD(month, -1, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalSold DESC;
```

Q26.

Calculate the total sales for each product category.

Ans.

```sql
SELECT c.CategoryName, SUM(od.Quantity * od.Price) AS TotalSales
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName;
```

Q27.

Find all customers who placed orders in the last 7 days.

Ans.

```sql
SELECT DISTINCT CustomerID
FROM Orders
WHERE OrderDate >= DATEADD(day, -7, GETDATE());
```

Q28.

Calculate the average order value for each customer.

Ans.

```sql
SELECT CustomerID, AVG(TotalAmount) AS AverageOrderValue
FROM Orders
GROUP BY CustomerID;
```

Q29.

Identify the product that generated the highest revenue.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity * od.Price) AS TotalRevenue
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalRevenue DESC
LIMIT 1;
```

Q30.

List customers who have placed more than 5 orders.

Ans.

```sql
SELECT CustomerID, COUNT(OrderID) AS OrderCount
FROM Orders
GROUP BY CustomerID
HAVING COUNT(OrderID) > 5;
```

Q31.

Find the month-over-month growth rate of total sales.

Ans.

```sql
SELECT
    DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m'))) / LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m')) * 100 AS GrowthRate
FROM Orders
GROUP BY DATE_FORMAT(OrderDate, '%Y-%m');
```

Q32.

Identify the employees who processed the highest number of orders.

Ans.

```sql
SELECT e.EmployeeID, COUNT(o.OrderID) AS OrderCount
FROM Orders o
JOIN Employees e ON o.EmployeeID = e.EmployeeID
GROUP BY e.EmployeeID
ORDER BY OrderCount DESC
LIMIT 1;
```

Q33.

Find the products that have not been sold in the last 6 months.

Ans.

```sql
SELECT p.ProductName
FROM Products p
LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID
LEFT JOIN Orders o ON od.OrderID = o.OrderID AND o.OrderDate >= DATEADD
(month, -6, GETDATE())
WHERE o.OrderID IS NULL;
```

Q34.

List customers and the total amount they have spent.

Ans.

```sql
SELECT CustomerID, SUM(TotalAmount) AS TotalSpent
FROM Orders
GROUP BY CustomerID;
```

Q35.

Calculate the average number of items per order.

Ans.

```sql
SELECT AVG(ItemCount) AS AvgItemsPerOrder
FROM (
    SELECT OrderID, SUM(Quantity) AS ItemCount
    FROM OrderDetails
    GROUP BY OrderID
) AS OrderItemCounts;
```

Q36.

Find overlapping date ranges for product availability.

Ans.

sql Copy code

```sql
CREATE TABLE ProductAvailability (
    ProductID INT,
    StartDate DATE,
    EndDate DATE
);

SELECT a.ProductID, a.StartDate, a.EndDate, b.StartDate, b.EndDate
FROM ProductAvailability a, ProductAvailability b
WHERE a.ProductID = b.ProductID
AND a.StartDate < b.EndDate
AND b.StartDate < a.EndDate
AND a.StartDate <> b.StartDate;
```

Q37.

Retrieve the most recent order for each customer.

Ans.

```sql
SELECT CustomerID, MAX(OrderDate) AS MostRecentOrderDate
FROM Orders
GROUP BY CustomerID;
```

Q38.

List products that have been ordered more than 100 times.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity) AS TotalOrdered
FROM Products p
JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName
HAVING SUM(od.Quantity) > 100;
```

Q39.

Create a query to find the longest consecutive sequence of orders for each customer.

Ans.

sql

Copy code

```sql
WITH ConsecutiveOrders AS (
    SELECT
        CustomerID,
        OrderDate,
        ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS RowNum
    FROM Orders
)
SELECT CustomerID, COUNT(*) AS LongestStreak
FROM ConsecutiveOrders
GROUP BY CustomerID, DATE_SUB(OrderDate, INTERVAL RowNum DAY)
ORDER BY LongestStreak DESC;
```

Q40.

Fetch the last N records in a table, ordered by a specific column.

Ans.

```sql
SELECT *
FROM Orders
ORDER BY OrderDate DESC
LIMIT N;
```

Q41.

Audit data changes in a SQL database.

Ans.

sql Copy code

```sql
CREATE TABLE AuditLog (
    AuditID INT PRIMARY KEY,
    TableName VARCHAR(50),
    Action VARCHAR(10),
    OldValue VARCHAR(255),
    NewValue VARCHAR(255),
    ChangeDate DATETIME,
    ChangedBy VARCHAR(50)
);

CREATE TRIGGER trgAudit
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
    INSERT INTO AuditLog (TableName, Action, OldValue, NewValue, ChangeDate,
                          ChangedBy)
    VALUES ('Orders', 'UPDATE', OLD.TotalAmount, NEW.TotalAmount, NOW(), USER());
END;
```

Q42.

Design a query to retrieve hierarchical data in a parent-child relationship.

Ans.

sql Copy code

```sql
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    ManagerID INT
);

WITH EmployeeHierarchy AS (
    SELECT EmployeeID, Name, ManagerID, 1 AS Level
    FROM Employees
    WHERE ManagerID IS NULL
    UNION ALL
    SELECT e.EmployeeID, e.Name, e.ManagerID, eh.Level + 1
    FROM Employees e
    JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID
)
SELECT * FROM EmployeeHierarchy
ORDER BY Level, ManagerID;
```

Q43.

Implement soft deletes in SQL.

Ans.

```sql
ALTER TABLE Orders ADD COLUMN IsDeleted BOOLEAN DEFAULT FALSE;

-- Mark as deleted
UPDATE Orders SET IsDeleted = TRUE WHERE OrderID = 1;

-- Retrieve non-deleted orders
SELECT * FROM Orders WHERE IsDeleted = FALSE;
```

Q44.

Write a query to calculate the year-over-year growth of sales.

Ans.

```sql
SELECT
    YEAR(OrderDate) AS Year,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)))
    / LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)) * 100 AS YoYGrowth
FROM Orders
GROUP BY YEAR(OrderDate);
```

Q45.

Handle data migration from one database to another.

Ans.

```sql
-- Using MySQL Workbench or similar tools for data migration.
-- Example:
mysqldump -u source_user -p source_db > source_db.sql
mysql -u target_user -p target_db < source_db.sql
```

Q46.

How to implement full-text search in SQL?

Ans.

```sql
CREATE FULLTEXT INDEX idx_productname ON Products(ProductName);

SELECT * FROM Products
WHERE MATCH(ProductName) AGAINST('search term');
```

Q47.

Write a query to find the median value in a numeric column.

Ans.

```sql
SELECT AVG(value) AS MedianValue
FROM (
    SELECT value
    FROM table
    ORDER BY value
    LIMIT 2 - (SELECT COUNT(*) FROM table) % 2
    OFFSET (SELECT (COUNT(*) - 1) / 2 FROM table)
) AS MedianValues;
```

Q48.

How to perform a bulk insert in SQL?

Ans.

```sql
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES
(1, 1, '2023-01-01', 100.00),
(2, 2, '2023-01-02', 150.00),
(3, 1, '2023-01-03', 200.00);
```

Q49.

Calculate the running total of sales for each day.

Ans.

```sql
SELECT OrderDate,
       TotalAmount,
       SUM(TotalAmount) OVER (ORDER BY OrderDate) AS RunningTotal
FROM Orders;
```

Q48.

Identify customers who have not placed any orders in the last 6 months.

Ans.

```sql
SELECT CustomerID
FROM Customers
WHERE CustomerID NOT IN (
    SELECT DISTINCT CustomerID
    FROM Orders
    WHERE OrderDate >= DATEADD(month, -6, GETDATE())
);
```

BOSSCODER ACADEMY

WHY BOSSCODER?

1000+ Alumni placed at Top Product-based companies.

More than **136% hike** for every **2 out of 3** working professional.

Average package of **24LPA.**

The syllabus is most **up-to-date** and the list of problems provided covers **all important topics.**

Lavanya
Meta

Course is very well **structured** and **streamlined** to crack any **MAANG** company

Rahul .
Google

EXPLORE MORE