SQL INTERVIEW MASTER SHEET

1. What is DBMS?

- A Database Management System (DBMS) is a program that controls creation, maintenance and use of a database. DBMS can be termed as File Manager that manages data in a database rather than saving it in file systems.

2. What is RDBMS?

- **RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.**
- **Example: SQL Server.**

3. What is SQL?

- **SQL stands for Structured Query Language , and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.**
- **Standard SQL Commands are Select.**

4. What is a Database?

- **Database is nothing but an organized form of data for easy access, storing, retrieval and managing of data. This is also known as structured form of data which can be accessed in many ways.**
- **Example: School Management Database, Bank Management Database.**

5. What are tables and Fields?

- **A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has specified number of column called fields but can have any number of rows which is called record.**
- **Example:.**
  Table: Employee.
  Field: Emp ID, Emp Name, Date of Birth. Data: 201456, David, 11/15/1960.

6. What is a primary key?

- **A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.**

7. What are the different types of SQL commands?

- **Answer: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language), and TCL (Transaction Control Language).**

8. What is a foreign key?

- **A foreign key is a field (or a collection of fields) in one table that uniquely identifies a row of another table, thereby creating a relationship between the two tables. This key ensures referential integrity by enforcing that the value in the foreign key column(s) matches a primary key value in the referenced table. Foreign keys help maintain data consistency by preventing actions that would destroy links between tables. They are crucial in relational database design for defining relationships and enforcing integrity constraints. Foreign keys can also prevent invalid data from being inserted into the foreign key column.**

9. What is a JOIN?

- **A JOIN clause is used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables in a single query, providing a way to merge information stored in different tables. JOINs are fundamental in SQL for building complex queries that involve multiple tables. Types of JOINs include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN, each serving different purposes based on the required results. JOINs facilitate comprehensive data analysis by integrating related data from separate sources.**

10. What are the types of JOINs in SQL?

- **Answer: The main types of JOINs in SQL are INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN. INNER JOIN returns only the rows with matching values in both tables. LEFT JOIN (or LEFT OUTER JOIN) returns all rows from the left table and the matched rows from the right table; unmatched rows in the right table return NULL. RIGHT JOIN (or RIGHT OUTER JOIN) returns all rows from the right table and the matched rows from the left table; unmatched rows in the left table return NULL. FULL JOIN (or FULL OUTER JOIN) returns rows when there is a match in one of the tables and returns NULL for non-matching rows in both tables.**

11. What is a database index?

- **Answer: A database index is a data structure that improves the speed of data retrieval operations on a table at the cost of additional storage space and maintenance overhead. Indexes are used to quickly locate data without having to search every row in a table each time a database table is accessed. They are especially useful for large tables, significantly enhancing query performance. However, indexes need to be updated whenever data is modified, which can introduce overhead. Common types of indexes include single-column indexes, composite indexes, unique indexes, and full-text indexes.**

12. What is normalization?

- **Answer: Normalization is the process of organizing data in a database to minimize redundancy and improve data integrity. It involves dividing large tables into smaller, related tables and defining relationships between them. The main objectives are to eliminate duplicate data, ensure logical data storage, and maintain data consistency. Normalization typically follows a series of steps called normal forms, from 1NF (First Normal Form) to 5NF (Fifth Normal Form), each with specific rules. This process helps in maintaining a clean and efficient database structure, facilitating easier maintenance and scalability.**

13. What is denormalization?

- **Answer: Denormalization is the process of combining normalized tables to improve read performance at the cost of additional data redundancy and potential write anomalies. It involves merging tables and duplicating data to reduce the number of joins required for data retrieval, which can speed up query performance in read-heavy applications. While denormalization can improve performance for specific queries, it can also lead to data anomalies, increased storage requirements, and more complex data maintenance. It is typically used in scenarios where the read performance gains outweigh the costs of redundancy.**

14. What is a query?

- **Answer: A query is a request for data or information from a database table or combination of tables. It is written in SQL and allows users to specify criteria to retrieve specific data subsets. Queries can perform various operations, such as filtering, sorting, joining tables, and aggregating data. They are fundamental to interacting with a database, enabling users to extract meaningful insights from stored data. Queries can be simple, retrieving data from a single table, or complex, involving multiple tables and various SQL functions.**

15. Explain the difference between WHERE and HAVING clauses.

- **Answer: The WHERE clause is used to filter records before any groupings are made, directly affecting the rows considered by the query. It is typically used with SELECT, UPDATE, DELETE statements to filter rows based on specified conditions. The HAVING clause, on the other hand, is used to filter values after they have been grouped by the GROUP BY clause. HAVING applies conditions to groups of rows created by the GROUP BY clause and is often used with aggregate functions (e.g., COUNT, SUM). WHERE cannot be used with aggregates, making HAVING necessary for such operations.**

16. What is a subquery?

- **Answer: A subquery is a query nested inside another query. It can be used in various places within an SQL statement, including the WHERE, FROM, and SELECT clauses. Subqueries allow for more complex queries by enabling operations that depend on the results of another query. They can return a single value, a set of values, or an entire result set and are used for tasks like filtering, comparisons, and**

inserting data based on conditions. Subqueries can be classified as single-row, multiple-row, or correlated subqueries, each serving different purposes in SQL operations.

17. What is a view in SQL?

- **Answer: A view is a virtual table based on the result set of an SQL statement. It contains rows and columns just like a real table, but the data is derived from one or more underlying tables. Views are used to simplify complex queries, enhance security by restricting data access, and present data in a specific format. They do not store data themselves but dynamically fetch it from the underlying tables whenever accessed. Views can be used for abstraction, presenting a simplified interface to complex data relationships, and can be updated if the underlying tables allow it.**

18. Explain the difference between UNION and UNION ALL.

- **Answer: UNION and UNION ALL are used to combine the result sets of two or more SELECT statements. UNION removes duplicate records from the combined result set, ensuring that each row is unique. This operation involves additional processing overhead due to the elimination of duplicates. UNION ALL, however, includes all records from the combined result sets, including duplicates, which makes it faster since no duplicate removal is performed. The choice between the two depends on whether duplicate records are acceptable in the final result set or not.**

19. What is a stored procedure?

- **Answer: A stored procedure is a set of SQL statements that can be stored and executed on the database server. It allows for the encapsulation of complex business logic within the database, improving code reuse and maintainability. Stored procedures can accept input parameters, return output parameters, and include programming constructs like loops and conditionals. They enhance performance by reducing the amount of data transferred between the database and application and can be used to enforce data integrity and security rules. Stored procedures are invoked using the CALL or EXECUTE commands.**

20. What is a trigger in SQL?

- **Answer: A trigger is a database object that is automatically executed or fired when certain events occur in a table, such as INSERT, UPDATE, or DELETE operations. Triggers are used to enforce business rules, maintain audit trails, and automatically update or validate data. They can be defined to execute before or after the triggering event, ensuring that specific actions are taken as part of the transaction. Triggers help maintain data integrity and consistency by automating responses to data changes. However, they should be used judiciously to avoid performance overhead and complex dependencies.**

21. Explain the ACID properties.

- **Answer: The ACID properties (Atomicity, Consistency, Isolation, Durability) are a set of principles that ensure reliable transaction processing in a database. Atomicity ensures that all parts of a transaction are completed successfully; otherwise, the transaction is rolled back. Consistency guarantees that a transaction brings the database from one valid state to another, maintaining data integrity. Isolation ensures that transactions are executed independently without interference, providing a consistent view of the data. Durability ensures that once a transaction is committed, the changes are permanent, even in the event of a system failure.**

22. What is an aggregate function?

- **Answer: Aggregate functions perform a calculation on a set of values and return a single value. Common aggregate functions include COUNT (counts the number of rows), SUM (calculates the total sum of a numeric column), AVG (calculates the average value), MIN (finds the minimum value), and MAX (finds the maximum value). These functions are often used with the GROUP BY clause to group rows that share a property and perform calculations on each group. Aggregate functions are essential for summarizing and analyzing data within a database.**

23. What are window functions in SQL?

- **Answer: Window functions perform calculations across a set of table rows related to the current row, providing analytical capabilities within SQL queries. Unlike aggregate functions, window functions do not collapse rows into a single result; instead, they maintain the original row structure while adding computed values. Examples include ROW_NUMBER (assigns a unique number to each row), RANK (assigns a rank to each row within a partition), and LAG/LEAD (accesses data from previous or subsequent rows). Window functions are useful for complex calculations like running totals, moving averages, and ranking.**

24. Explain the difference between DELETE and TRUNCATE.

- **Answer: DELETE and TRUNCATE are both used to remove records from a table, but they operate differently. DELETE removes rows one by one and can include a WHERE clause to specify which rows to delete; it is a DML operation and logs individual row deletions, making it slower. TRUNCATE removes all rows in a table without logging individual deletions, which makes it faster; it is a DDL operation and resets identity columns. DELETE allows for rollback if within a transaction, whereas TRUNCATE cannot be rolled back once executed if not within a transaction.**

25. What is a Common Table Expression (CTE)?

- **Answer: A Common Table Expression (CTE) is a temporary result set defined within the execution scope of a single SELECT, INSERT, UPDATE, or DELETE statement. CTEs are created using the WITH clause and can be referenced within the main query to simplify complex joins and subqueries. They improve readability and maintainability by breaking down complicated queries into modular parts. CTEs can be recursive, allowing for hierarchical data retrieval. They are especially useful for querying data that requires repeated references or recursive processing.**

26. Explain the term 'deadlock' in SQL.

- **Answer: A deadlock is a situation where two or more transactions block each other by holding locks on resources and requesting locks on the resources held by each other. This results in a cycle of dependencies with no transaction able to proceed, effectively halting progress. Deadlocks are detrimental to database performance and require intervention to resolve. Database management systems detect deadlocks and typically resolve them by aborting one of the transactions, allowing the others to continue. Proper transaction management and lock handling techniques can help minimize the occurrence of deadlocks.**

27. What is a schema in SQL?

- **Answer: A schema is a logical container for database objects such as tables, views, indexes, and stored procedures. It helps organize and separate database objects for better manageability and security. Schemas provide a namespace for database objects, allowing for the same object name to be used in different schemas within the same database. They facilitate user access control by enabling permissions to be granted at the schema level. Schemas also help in structuring databases, especially in large systems with multiple users or applications, by categorizing objects based on functionality or ownership.**

28. How would you handle database migration for a client?

- **Answer: Handling database migration for a client involves several steps. First, plan the migration by assessing the current database structure and requirements. Create a complete backup of the existing database to prevent data loss. Migrate the schema and data to the new database, ensuring compatibility and data integrity. Perform thorough testing to verify that the migration was successful and that the new database functions as expected. Finally, switch over to the new database, monitor the transition, and provide support for any issues that arise during the migration process.**

29. How do you ensure data security in a client's database?

- **Answer: Ensuring data security in a client's database involves implementing several measures. Use encryption for data at rest and in transit to protect sensitive information. Implement access controls and user authentication mechanisms to restrict unauthorized access. Regularly update and patch the database software to address security vulnerabilities. Perform regular security audits and vulnerability assessments to identify and mitigate potential threats. Additionally, enforce strong password policies and educate users on security best practices to enhance overall database security.**

30. What steps would you take to optimize a client's slow query?

- **Answer: To optimize a client's slow query, start by analyzing the query to understand its structure and logic. Use EXPLAIN or similar tools to examine the query execution plan and identify bottlenecks. Check for appropriate indexing on columns used in**

31. You need to retrieve a hierarchical data structure from a table (such as an organizational chart) where each row references its parent row. Will you use a recursive CTE or a series of subqueries? Explain your choice.

- **Answer: A recursive CTE would be more appropriate for retrieving hierarchical data structures, such as an organizational chart. Recursive CTEs are specifically designed to handle such scenarios efficiently by repeatedly referencing themselves, making the code cleaner and easier to understand. Subqueries can become complex and inefficient for hierarchical queries.**

32. You need to filter a large dataset based on a complex condition that is used multiple times within the query. Would you use a subquery or a CTE to make the query more readable and maintainable? Justify your answer.

- **Answer: Using a CTE would be preferable in this situation. CTEs enhance readability and maintainability by allowing the complex condition to be defined once and referenced multiple times within the main query. This makes the query structure clearer and easier to manage compared to embedding the same subquery multiple times.**

33. You need to update a table based on a complex condition derived from another table. Would you use a subquery within an UPDATE statement or a JOIN operation? Explain your choice.

- **Answer: Using a JOIN operation within an UPDATE statement is generally more efficient and readable for updating a table based on a condition derived from another table. JOINs can directly match rows from both tables and apply updates in a more straightforward manner compared to subqueries, which may be less performant and harder to maintain.**

34. You are asked to find the top N sales records for each region in your sales database. Would you use a window function or a GROUP BY clause with subqueries? Justify your decision.

- **Answer: A window function, specifically the ROW_NUMBER() function, would be the best choice for finding the top N sales records for each region. Window functions provide a way to assign a unique number to rows within a partition (in this case, each region) without collapsing the data into groups, making it easier to filter the top N records. GROUP BY with subqueries would be more complex and less efficient for this purpose.**

35. You need to enforce a rule that certain records in a table should always satisfy a complex business condition whenever they are inserted or updated. Would you use a trigger or a CHECK constraint? Explain your reasoning.

- **Answer: Using a trigger would be more appropriate for enforcing complex business rules that cannot be easily expressed with a CHECK constraint. Triggers provide greater flexibility and can include complex logic and calculations to enforce rules during insertions and updates. CHECK constraints are more limited and best suited for simpler conditions.**

36. You need to enforce a rule that certain records in a table should always satisfy a complex business condition whenever they are inserted or updated. Would you use a trigger or a CHECK constraint? Explain your reasoning.

- **Answer: Using a trigger would be more appropriate for enforcing complex business rules that cannot be easily expressed with a CHECK constraint. Triggers provide greater flexibility and can include complex logic and calculations to enforce rules during insertions and updates. CHECK constraints are more limited and best suited for simpler conditions.**

37. You are tasked with creating a report that must include both aggregated and detailed data from a single table. Would you use a GROUP BY clause with subqueries or window functions? Provide your reasoning.

- **Answer: Window functions would be the better choice for creating a report that includes both aggregated and detailed data. Window functions allow you to perform aggregations while retaining the detailed rows, which makes it easier to include both levels of information in the same query. GROUP BY with subqueries would separate the aggregated and detailed data, making it harder to combine them in a single report.**

38. You need to compare the performance of two different SQL queries to determine which one is more efficient. Would you use the execution plan or database profiling tools? Explain your choice.

- **Answer: Using the execution plan is essential for comparing the performance of two different SQL queries. The execution plan provides detailed insights into how the database engine executes each query, including information on indexes used, join methods, and estimated costs. Database profiling tools can also be useful but may not provide the same level of detailed analysis as the execution plan.**

39. You have a table that experiences high read and write operations, and you need to improve query performance. Would you use indexing or database partitioning? Justify your answer.

- **Answer: Both indexing and database partitioning can improve query performance, but the choice depends on the specific scenario. Indexing is generally the first step to improve read performance by speeding up data retrieval. However, if the table is very large and experiences high write operations, database partitioning might be more effective. Partitioning divides the table into smaller, more manageable pieces, reducing the amount of data processed during each read and write operation and improving overall performance.**

40. How will you connect SQL data to Power BI?

Answer: To connect SQL data to Power BI, you need to follow these steps:

1. **Open Power BI Desktop: Launch Power BI Desktop on your computer.**
2. **Get Data: Click on the 'Home' tab, then select 'Get Data' and choose 'SQL Server' from the list of available data sources.**
3. **Enter SQL Server Details: In the dialog box that appears, enter the server name and database name of your SQL Server. You may also specify a custom SQL query if needed.**
4. **Authentication: Choose the appropriate authentication method (Windows or SQL Server authentication) and provide the necessary credentials.**
5. **Select Data: Once connected, Power BI will display a navigator window where you can select the tables or views you want to import. You can also transform the data using Power Query Editor before loading it into Power BI.**
6. **Load Data: Click 'Load' to import the selected data into Power BI. After the data is loaded, you can start creating visualizations and reports using the imported SQL data.**

41. What are the differences between DROP, DELETE, and TRUNCATE in SQL?

Feature	DROP	DELETE	TRUNCATE
Purpose	Removes the entire table, including its structure and data.	Deletes specific rows based on a condition.	Removes all rows from a table, but keeps the structure intact.
Usage	DROP TABLE table_name;	DELETE FROM table_name WHERE condition;	TRUNCATE TABLE table_name;
Rollback	Cannot be rolled back once executed.	Can be rolled back if within a transaction.	Cannot be rolled back if not within a transaction.
Performance	Fast, as it removes the table structure and data in one step.	Slower for large datasets, as it deletes rows one by one and logs each deletion.	Fast, as it removes all rows without logging individual deletions.
Indexes & Constraints	All indexes, constraints, and triggers are removed along with the table.	Retains the table structure, indexes, constraints, and triggers.	Retains the table structure, indexes, constraints, and triggers.
Identity Reset	Not applicable.	Does not reset identity column values.	Resets identity column values (if any).

Q.42) Could you explain the differences between the rank, dense rank, and row number functions in SQL? When and how would you use each of them?

Answer:

The rank, dense rank, and row number functions are used to assign a unique sequential number to each row in a result set, but they differ in how they handle ties (rows with the same values). Here's a breakdown of each function:

1. **Rank Function: The rank function assigns a unique rank to each distinct row in the result set, leaving gaps in the ranking sequence when there are ties. If multiple rows share the same value, they are assigned the same rank, and the next rank is incremented accordingly. For example, if two rows are tied for the second position, the next rank assigned would be 4.**
2. **Dense Rank Function: The dense rank function also assigns a unique rank to each distinct row but does not leave gaps in the ranking sequence, even when there are ties. It assigns consecutive ranks to tied**

rows without skipping any ranks in between. For example, if two rows are tied for the second position, the next rank assigned would be 3.

3. Row Number Function: The row number function assigns a unique sequential number to each row in the result set, without regard for ties. It simply numbers each row in the order they appear in the result set, starting from 1.

Usage:

- Rank Function: Use the rank function when you want to assign ranks to rows, but you're okay with having gaps in the ranking sequence.
- Dense Rank Function: Use the dense rank function when you want to assign ranks to rows and ensure there are no gaps in the ranking sequence, even if there are ties.
- Row Number Function: Use the row number function when you simply want to number each row in the result set sequentially.

Tables:

Consider two tables: Employees and Salaries.

- Employees Table:
  - EmployeeID (INT): Primary key for identifying employees.
  - Name (VARCHAR): Name of the employee.
  - Department (VARCHAR): Department in which the employee works.

EmployeeID	Name	Department
1	John	HR
2	Emma	Finance
3	Michael	HR
4	Sophia	IT

- Salaries Table:
  - EmployeeID (INT): Foreign key referencing EmployeeID in the Employees table.
  - Salary (DECIMAL): Salary of the employee.

EmployeeID	Salary
1	50000.00
2	60000.00
3	55000.00
4	58000.00

Example:

You have a scenario where you want to rank employees based on their salaries to determine their positions within the company. In this case, you can use the rank, dense rank, or row number functions to assign a unique sequence number to each employee based on their salary, depending on your specific requirements for handling ties and gaps in the ranking sequence.

output

Name	Department	Salary	Rank	DenseRank	RowNum
Emma	Finance	60000.00	1	1	1
Sophia	IT	58000.00	2	2	2
Michael	HR	55000.00	3	3	3
John	HR	50000.00	4	4	4