What is RDBMS? Explain its features.

Explanation:

- RDBMS (Relational Database Management System) is a database management system based on the relational model. It stores data in tables, which consist of rows and columns. **Features:**
    - **Data Integrity:** Ensures accuracy and consistency of data.
    - **Data Security:** Allows permissions and access controls.
    - **Normalization:** Reduces data redundancy and dependency.
    - **Scalability:** Can handle large amounts of data efficiently.
    - **Support for ACID Properties:** Ensures reliable transactions.

Subquestions and Solutions:

- What is the relational model?
    - The relational model organizes data into tables (relations) of rows and columns.
- Why is data integrity important in RDBMS?
    - It ensures the accuracy and consistency of data, preventing corruption.
- How does normalization help in RDBMS?
    - It organizes data to reduce redundancy and improve data integrity.

What is SQL?

Explanation:

- SQL (Structured Query Language) is a standardized language used to manage and manipulate relational databases.

Subquestions and Solutions:

- What are the basic operations you can perform with SQL?
    - SQL can be used to perform CRUD operations: Create, Read, Update, and Delete.
- Why is SQL considered a powerful tool for database management?
    - It allows for complex queries, data manipulation, and integration with other systems.

What are the Advantages of SQL?

Explanation:

- **Portability:** Works on various systems from PCs to mainframes.
- **Interactive Language:** Easy to learn and use for queries.
- **Standardized Language:** Follows ANSI and ISO standards.
- **Integration:** Can be embedded within other languages.
- **Efficient Data Management:** Provides powerful tools for data manipulation.

Subquestions and Solutions:

- How does SQL ensure efficient data management?
  - By providing commands for complex queries and transactions.
- Why is the portability of SQL advantageous?
  - It allows databases to be moved and accessed across different platforms.

Differentiate between SQL and NoSQL databases.

Explanation:

- **SQL Databases:** Relational, use structured query language, have a fixed schema, and are suitable for complex queries.
- **NoSQL Databases:** Non-relational, can store structured, semi-structured, or unstructured data, have dynamic schemas, and are suitable for big data and real-time web applications.

Subquestions and Solutions:

- What are some examples of SQL and NoSQL databases?
  - SQL: MySQL, PostgreSQL, Oracle. NoSQL: MongoDB, Cassandra, Redis.
- When would you choose NoSQL over SQL?
  - When dealing with large-scale data or requiring high flexibility in schema.

Explain DDL, DML, DQL, DCL.

Explanation:

- **DDL (Data Definition Language):** Commands that define the database structure. E.g., CREATE, ALTER, DROP.
- **DML (Data Manipulation Language):** Commands that manipulate data. E.g., INSERT, UPDATE, DELETE.
- **DQL (Data Query Language):** Commands that query the database. E.g., SELECT.
- **DCL (Data Control Language):** Commands that control access to data. E.g., GRANT, REVOKE.

Subquestions and Solutions:

- What command would you use to add a new table in SQL?
- How do you update data in a table?

What are wildcards used in databases for Pattern Matching?

Explanation:

- Wildcards are special characters used in SQL with the LIKE operator to search for a specified pattern in a column.
    - **%:** Represents zero or more characters.
    - **_ :** Represents a single character.

Subquestions and Solutions:

- How would you find all records where a column starts with 'A'?
    - `SELECT * FROM table WHERE column LIKE 'A%';`
- How do you use a wildcard to match a single character?
    - `SELECT * FROM table WHERE column LIKE '_a';`

What is the difference between the "DELETE", "TRUNCATE" and "DROP" commands?

Explanation:

- **DELETE:** Removes rows from a table based on a condition. Can be rolled back.
- **TRUNCATE:** Removes all rows from a table, cannot be rolled back in most databases.
- **DROP:** Deletes the table from the database.

Subquestions and Solutions:

- How do you delete specific rows from a table?
    - Using `DELETE FROM table WHERE condition;`.
- What is the main difference between DELETE and TRUNCATE?
    - DELETE can be rolled back, TRUNCATE cannot.

Explain the difference between UNION and UNION ALL.

Explanation:

- **UNION:** Combines the result sets of two queries and removes duplicates.
- **UNION ALL:** Combines the result sets of two queries and includes duplicates.

Subquestions and Solutions:

- When would you use UNION instead of UNION ALL?
    - When you need to remove duplicate rows from the result set.
- How does UNION ALL affect performance compared to UNION?
    - UNION ALL is generally faster because it does not remove duplicates.

How do you implement one-to-one, one-to-many, and many-to-many relationships while designing tables?

Explanation:

- **One-to-One:** Use a unique foreign key in one table.
- **One-to-Many:** Use a foreign key in the many table.
- **Many-to-Many:** Use a junction table with foreign keys referencing the primary keys of both tables.

Subquestions and Solutions:

- How do you design a one-to-one relationship in SQL?
  - Using a unique constraint on the foreign key.
- What table structure is required for a many-to-many relationship?
  - A junction table with foreign keys referencing both related tables.

What is the difference between the "WHERE" clause and the "HAVING" clause?

Explanation:

- **WHERE Clause:** Filters rows before aggregation.
- **HAVING Clause:** Filters groups after aggregation.

Subquestions and Solutions:

- When would you use a HAVING clause?
  - When you need to filter aggregated results.
- Can you use HAVING without GROUP BY?
  - No, HAVING is used with GROUP BY to filter aggregated data.

What are constraints? Explain different types of constraints.

Explanation:

- Constraints are rules applied to table columns to enforce data integrity.
  - **NOT NULL:** Ensures a column cannot have NULL value.
  - **UNIQUE:** Ensures all values in a column are unique.
  - **PRIMARY KEY:** A combination of NOT NULL and UNIQUE. Uniquely identifies each row.
  - **FOREIGN KEY:** Ensures referential integrity between tables.
  - **CHECK:** Ensures all values in a column satisfy a specific condition.
  - **DEFAULT:** Sets a default value for a column if no value is specified.

Subquestions and Solutions:

- How do you enforce a column to have unique values?
  - Using the UNIQUE constraint.
- What constraint ensures referential integrity?
  - The FOREIGN KEY constraint.

Explain all Keys.

Explanation:

- **Primary Key:** Uniquely identifies each record in a table.
- **Foreign Key:** A field in one table that uniquely identifies a row of another table.
- **Candidate Key:** A field, or combination of fields, that can uniquely identify a record.
- **Super Key:** A set of one or more keys that can uniquely identify a record.
- **Composite Key:** A primary key composed of multiple columns.
- **Alternate Key:** A candidate key that is not the primary key.

Subquestions and Solutions:

- What is the difference between a primary key and a unique key?
  - A primary key cannot be NULL, while a unique key can be NULL.
- How do you define a composite key?
  - By combining multiple columns into a primary key.

What is Join? Explain All.

Explanation:

- Joins are used to combine rows from two or more tables, based on a related column.
  - **INNER JOIN:** Returns rows with matching values in both tables.
  - **LEFT JOIN:** Returns all rows from the left table, and the matched rows from the right table.
  - **RIGHT JOIN:** Returns all rows from the right table, and the matched rows from the left table.
  - **FULL OUTER JOIN:** Returns all rows when there is a match in one of the tables.
  - **CROSS JOIN:** Returns the Cartesian product of both tables.
  - **SELF JOIN:** Joins a table to itself.

Subquestions and Solutions:

- What join would you use to get all records from both tables, including unmatched rows?
  - FULL OUTER JOIN.
- How does an INNER JOIN work?
  - It returns only the rows with matching values in both tables.

What is an Entity-Relationship diagram?

Explanation:

- An ER diagram is a visual representation of the database structure. It shows entities, their attributes, and the relationships between them.

What is the difference between a “Local Temporary Table” and “Global Temporary Table”?

Explanation:

- **Local Temporary Table:** Exists only for the duration of the session that created it. Prefix with a single `#`.
- **Global Temporary Table:** Exists for all sessions, but is deleted when the session that created it ends. Prefix with `##`.

Subquestions and Solutions:

- When would you use a local temporary table?
  - When you need a temporary table that is session-specific.
- How do you create a global temporary table?
  - Use the prefix `##` before the table name.

What is Aggregate Functions?

Explanation:

- Aggregate functions perform a calculation on a set of values and return a single value.
  - **SUM:** Adds up all the values.
  - **AVG:** Calculates the average of the values.
  - **COUNT:** Counts the number of rows.
  - **MAX:** Returns the maximum value.
  - **MIN:** Returns the minimum value.

Subquestions and Solutions:

- How do you find the total of a numeric column?
  - Using the `SUM()` function.
- What function would you use to find the number of records?
  - `COUNT()`.

Describe the differences between the CHAR and VARCHAR data types.

Explanation:

- **CHAR:** Fixed-length character data type. Padded with spaces if not fully utilized.
- **VARCHAR:** Variable-length character data type. Uses only as much space as needed.

Subquestions and Solutions:

- When would you choose VARCHAR over CHAR?
  - When you expect the data to have variable lengths and want to save space.
- How does CHAR handle storage for shorter strings?
  - It pads the remaining space with blanks.

What is the difference between “Clustered Index” and “Non-Clustered Index”?

Explanation:

- **Clustered Index:** Determines the physical order of data in the table. Each table can have only one.
- **Non-Clustered Index:** Does not alter the physical order of data. A table can have multiple non-clustered indexes.

Subquestions and Solutions:

- What is the key benefit of using a clustered index?
  - Faster retrieval of data based on the index.
- How do non-clustered indexes affect query performance?
  - They allow for faster searching and sorting but do not affect the physical order of the table.

What is Identity?

Explanation:

- Identity is an auto-incrementing feature for a column, often used to create unique identifiers.

Subquestions and Solutions:

- How do you define an identity column in SQL?
  - By using the `IDENTITY` property during table creation.

What is the difference between “Stored Procedure” and “Function”?

Explanation:

- **Stored Procedure:** A set of SQL statements that can perform actions like modifying data. Can return multiple values.
- **Function:** A set of SQL statements that perform a calculation and return a single value. Cannot modify data.

Subquestions and Solutions:

- Can a function be used in a SELECT statement?
  - Yes, it can be used to return computed values.
- What is a key difference between procedures and functions?
  - Procedures can perform actions, while functions are used for computations.

What is database normalization? What are the different types of normalization?

Explanation:

- **Normalization:** The process of organizing data to reduce redundancy and improve data integrity.
  - **1NF (First Normal Form):** Ensures each column contains atomic values.
  - **2NF (Second Normal Form):** Meets 1NF criteria and all non-key attributes are fully functional dependent on the primary key.
  - **3NF (Third Normal Form):** Meets 2NF criteria and all attributes are functionally dependent on the primary key, with no transitive dependency.
  - **BCNF (Boyce-Codd Normal Form):** A stricter version of 3NF.

Subquestions and Solutions:

- Why is normalization important?
  - It helps in minimizing data redundancy and improving data integrity.
- What is the difference between 2NF and 3NF?
  - 2NF eliminates partial dependencies, while 3NF eliminates transitive dependencies.

What is a view in SQL? How to create one?

Explanation:

- A view is a virtual table based on the result set of an SQL query. **How to Create:**
- `CREATE VIEW view_name AS SELECT columns FROM table WHERE condition;`

Subquestions and Solutions:

- When would you use a view?
  - To simplify complex queries and enhance security by limiting data access.
- Can a view be updated?
  - Yes, if it is based on a single table without aggregation.

What is a Trigger? How to create one?

Explanation:

- A trigger is a stored procedure that automatically executes in response to specific events on a table, such as INSERT, UPDATE, or DELETE. **How to Create:**
- `CREATE TRIGGER trigger_name BEFORE/AFTER INSERT/UPDATE/DELETE ON table FOR EACH ROW BEGIN ... END;`

Subquestions and Solutions:

- What is the purpose of a trigger?
  - To enforce business rules, audit changes, or automate complex tasks.
- How do you create a trigger that runs before an insert operation?
  - Use `BEFORE INSERT` in the trigger definition.

What is a transaction? What are ACID properties?

Explanation:

- A transaction is a sequence of operations performed as a single logical unit of work. **ACID Properties:**
  - **Atomicity:** Ensures that all operations within the transaction are completed successfully or none are.
  - **Consistency:** Ensures that the database moves from one valid state to another.
  - **Isolation:** Ensures that transactions do not affect each other.
  - **Durability:** Ensures that once a transaction is committed, it remains so even in the event of a system failure.

Subquestions and Solutions:

- What is the importance of ACID properties in a transaction?
  - They ensure the reliability and integrity of transactions in a database.
- How does isolation affect concurrent transactions?
  - It prevents transactions from interfering with each other, ensuring data consistency.

What is a Database Lock?

Explanation:

- A database lock is a mechanism to prevent multiple transactions from accessing the same data simultaneously, ensuring data integrity.

Subquestions and Solutions:

- How does a lock affect database performance?
  - While it ensures data consistency, excessive locking can lead to deadlocks and performance degradation.
- What are different types of locks?
  - **Shared Lock:** Allows multiple transactions to read but not modify.
  - **Exclusive Lock:** Allows only one transaction to read and modify.