1. What's the difference between batch,script,transaction,BackUp

1. Batch

A **batch** refers to a group of commands or operations that are executed together as a unit, typically in a predefined sequence. In database systems or programming environments, batch processing allows multiple tasks to run automatically without manual intervention.

Use case: Running a series of SQL statements to update records in bulk.

2. Script

A **script** is a file containing a series of instructions written in a programming or scripting language (e.g., SQL, Python, Bash). Scripts are used to automate repetitive tasks and ensure consistent execution.

Use case: An SQL script may include commands to create tables, insert data, and update records in a database.

3. Transaction

A **transaction** is a logical unit of work that consists of one or more operations which must either all succeed or all fail as a single unit. Transactions ensure data integrity and consistency, particularly in database systems.

Key properties (ACID):

- **Atomicity:** All operations succeed or none do
- **Consistency:** Data remains in a valid state
- **Isolation:** Transactions do not interfere with each other
- **Durability:** Once committed, changes are permanent

Use case: Transferring funds between two bank accounts, where both debit and credit actions must be completed together.

4. Backup

A **backup** is a copy of data stored separately from the original system to protect against data loss or corruption. Backups are critical for disaster recovery and maintaining business continuity.

Use case: Creating a scheduled daily backup of a production database to restore in case of hardware failure or accidental deletion.

2. What meant by logging transaction and why this happens

Logging a transaction refers to the process of recording details of database operations in a separate, persistent file known as the transaction log. This log captures information about each change made to the data, including insertions, deletions, and updates, along with timestamps and other metadata.

Purpose of Logging Transactions

1. **Data Recovery**
   In the event of a system failure, power outage, or crash, the transaction log can be used to **recover the database** to a consistent state by replaying committed transactions or rolling back incomplete ones.
2. **Rollback Support**
   Logging allows the system to **undo** a transaction if an error occurs before it is completed, ensuring data integrity.
3. **Auditing and Tracking**
   Logs provide a record of all changes, which can be useful for **auditing purposes** and for tracking user actions and system behavior over time.
4. **Concurrency Control**
   In systems that allow multiple users to access the database at once, logging supports **isolation** between concurrent transactions, helping prevent conflicts or inconsistencies.
5. **Replication and Mirroring**
   Transaction logs can be used to replicate changes to other servers in **real-time**, ensuring data consistency across distributed systems.

3. What’s the difference between soft delete and hard delete

In database management, deleting data can be handled in two primary ways: soft delete and hard delete. The choice between them depends on the system's requirements for data retention, auditing, and recovery**.**

Soft Delete

A **soft delete** means that a record is **not physically removed** from the database. Instead, it is marked as deleted using a specific flag or status column, such as `is_deleted` or `deleted_at`.

Characteristics:

- Data remains in the table and can be recovered easily.
- A flag (e.g., `is_deleted = 1`) indicates the record is considered deleted.
- Typically used in systems that require audit trails or historical data.
- Requires additional filtering in queries to exclude "deleted" records.

Use Case:
Applications where data needs to be restored later, such as user accounts or order histories.

Hard Delete

A **hard delete** refers to **permanently removing** a record from the database. Once deleted, the data cannot be recovered unless a backup is available.

Characteristics:

- The record is completely erased from the table.
- No trace of the deleted data remains in the database.
- Typically used when the data is no longer needed or must be removed for compliance reasons.

Use Case:
Removing expired temporary data, logs, or sensitive records that must be erased for privacy regulations.

Bonus

4. What is trigger and why use it

A **trigger** in a database is a special type of stored procedure that is **automatically executed** in response to certain events on a specific table or view. These events can be `INSERT`, `UPDATE`, or `DELETE` operations.

Purpose of Using Triggers:

1. **Enforce Business Rules:**
   Automatically enforce complex rules or restrictions without requiring application-side logic.
2. **Data Validation and Consistency:**
   Ensure data integrity across related tables.
3. **Auditing and Logging:**
   Track changes to sensitive data by automatically logging when and how changes occur.
4. **Automatic Actions:**
   Perform additional updates, notifications, or cleanup tasks in response to changes in the database.

Example Use Case:

- Automatically insert a log entry into an audit table whenever a record is deleted from the `Users` table.

Triggers help **automate logic inside the database layer**, ensuring that critical rules are enforced even if application logic is bypassed.

5. What is permissions and how w grant and revoke it

Permissions in a database system define what actions a user or role is allowed to perform. These include actions like reading data, writing data, modifying structures, or executing stored procedures.

Common Permission Types:

- `SELECT` – Read data from a table
- `INSERT` – Add data to a table
- `UPDATE` – Modify existing data
- `DELETE` – Remove data from a table
- `EXECUTE` – Run stored procedures or functions

How to Grant Permissions:

To allow a user to perform a specific action, use the `GRANT` command.