• *Theory Questions*

1. SQL Vs NoSQL Databases

SQL databases are a good choice when you have structured data with predefined schemas and the need for complex queries and relationships between tables. They provide strong data integrity through ACID compliance and are suitable for applications that require strict consistency and transactional support. SQL databases are commonly used in traditional enterprise applications, financial systems, and situations where data integrity is critical.

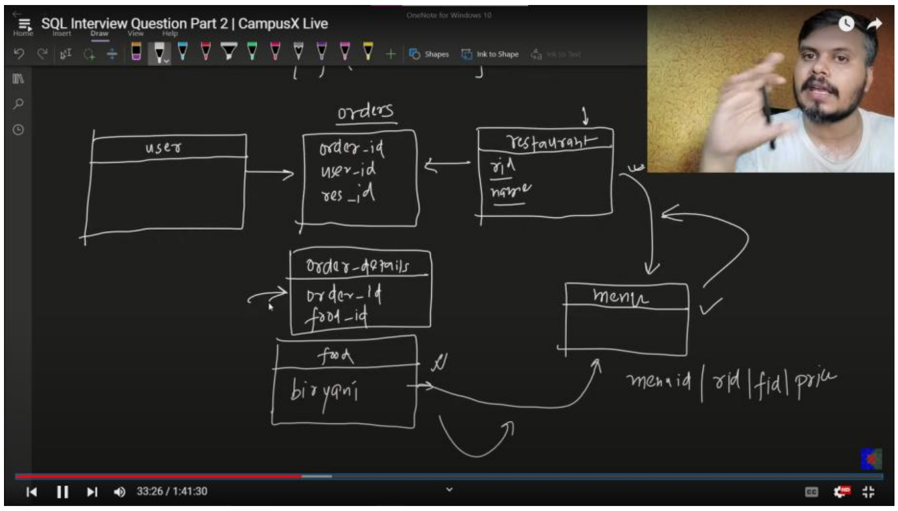
NoSQL databases, on the other hand, excel when dealing with unstructured or semi-structured data, where flexibility and scalability are paramount. They offer dynamic schemas, horizontal scalability, and high throughput for read/write operations. NoSQL databases are popular in modern web and mobile applications, content management systems, real-time analytics, and scenarios where rapid development and scaling are essential. They allow developers to quickly iterate and adapt to changing data requirements without the constraints of a fixed schema.

2. Tables required for many to many relationship

In a many-to-many relationship, you need three tables: two main entity tables and a junction table.
For example, let's consider a scenario where you have "Students" and "Courses". Each student can enroll in multiple courses, and each course can have multiple students.

1. **Students Table:**
   Columns: student_id, name, age, and other student attributes.
2. **Courses Table:**
   Columns: course_id, name, and other course attributes.
3. **Junction Table:**
   This table connects the Students and Courses tables, allowing you to associate students with courses.
   Columns: student_id, course_id.
   Each row represents a student-course pairing.
   The junction table helps establish the many-to-many relationship by storing the student_id and course_id pairs. You can use these tables to query and retrieve information about student enrollments in courses.

3. **Draw schema design for swiggy.**



4. **Types of Joins -> Union Vs Union All**

When it comes to joining tables in SQL, the UNION and UNION ALL operators are used to combine the result sets of two or more SELECT statements. However, there are some key differences between them:

UNION: The UNION operator is used to combine the result sets of multiple SELECT statements into a single result set. It eliminates duplicate rows from the final result by default. The columns in the SELECT statements must have the same data types and be in the same order.

UNION ALL: The UNION ALL operator also combines the result sets of multiple SELECT statements into a single result set. However, unlike UNION, it does not eliminate duplicate rows. It simply appends all rows from each SELECT statement to the final result, including any duplicate rows. The columns in the SELECT statements must have the same data types and be in the same order.

In summary:

UNION eliminates duplicate rows, while UNION ALL includes all rows from each SELECT statement.
UNION requires the columns in the SELECT statements to have the same data types and order, while UNION ALL has the same requirement.
UNION is useful when you want to combine and deduplicate rows from multiple SELECT statements, while UNION ALL is used when you want to combine all rows without removing duplicates.

5. **OLAP Vs OLTP [Normalized Vs Denormalized Data]**

OLAP (Online Analytical Processing) and OLTP (Online Transaction Processing) are two different types of database systems that serve distinct purposes:

OLAP:

OLAP databases are designed for complex analytical queries and reporting.

They are optimized for read-heavy workloads and provide fast access to aggregate and summarized data.OLAP databases typically store historical and large volumes of data.

The data in OLAP databases is structured in a multidimensional model, organized into dimensions (such as time, geography, and product) and measures (such as sales revenue).

OLAP databases often use deformalized data structures, where data is grouped and pre-calculated for efficient querying and analysis.OLAP systems are used for business intelligence, data mining, and decision support applications.

OLTP:

OLTP databases are designed for transactional processing, focusing on fast and efficient data modifications and real-time transaction handling.

They are optimized for write-intensive workloads and ensure data consistency and integrity.

OLTP databases typically store current and operational data, handling day-to-day transactions.

The data in OLTP databases is typically structured in a normalized model, where data is organized into separate tables to minimize redundancy and ensure data consistency.

OLTP systems are used for applications such as e-commerce, banking systems, inventory management, and online reservations.

Normalized vs. Deformalized Data:

In normalized data models, the data is organized into separate tables with relationships defined by primary and foreign keys. This minimizes data redundancy and ensures data consistency but can result in more complex queries and joins for analysis.

In deformalized data models, the data is combined and pre-calculated in a way that simplifies queries and improves performance. This approach allows for faster analysis but can result in redundancy and decreased data consistency.

In summary, OLAP databases are used for complex analysis and reporting, store historical data, and often use deformalized data structures. On the other hand, OLTP databases focus on real-time transaction processing, store current data, and typically use normalized data structures to ensure data integrity.

ETL (Extract, Transform, Load) is a process used to collect data from different sources, modify or format it, and then load it into a target system or database.

Extract: In this step, data is gathered from various sources such as databases, files, or web services.
Transform: The extracted data is then transformed or modified to meet specific requirements. This may involve cleaning the data, removing errors, converting data types, or aggregating and summarizing information.
Load: Finally, the transformed data is loaded into a target system, such as a data warehouse or database, where it can be easily accessed and analyzed.

6. Database keys

In a relational database, keys are used to uniquely identify and establish relationships between entities within the database. There are several types of keys commonly used:

Primary Key: A primary key is a unique identifier for each record in a table. It ensures that each row in the table can be uniquely identified. Usually, a primary key is a single column, such as an auto-incrementing integer or a combination of columns that uniquely identify a record.

Foreign Key: A foreign key is a field in a table that refers to the primary key in another table. It establishes a relationship between the two tables. The foreign key enforces referential integrity, ensuring that the values in the foreign key column correspond to existing values in the referenced primary key column.

Composite Key: A composite key is a primary key that consists of two or more columns. It is used when a single column cannot uniquely identify a record, but the combination of multiple columns can.

Candidate Key: A candidate key is a column or a set of columns that can uniquely identify a record in a table. Multiple candidate keys may exist, but one is chosen as the primary key.

7. Types of normalization

Normalization is a process in database design that helps eliminate data redundancy and ensures data integrity by organizing data into well-structured relations (tables). There are several normal forms in database normalization, each addressing a specific type of data redundancy. The most commonly recognized normal forms are:
First Normal Form (1NF):

Ensures atomicity by eliminating repeating groups and ensuring each column contains only a single value.
Requires that each column in a table holds only atomic (indivisible) values.

Second Normal Form (2NF):
Builds upon 1NF and eliminates partial dependencies.
Requires that all non-key attributes depend on the entire primary key, not just part of it.
Involves splitting the table into multiple tables, each with a unique primary key.

Third Normal Form (3NF):
Builds upon 2NF and eliminates transitive dependencies.
Requires that no non-key attribute depends on another non-key attribute.
Involves further splitting tables to remove indirect relationships.

Boyce-Codd Normal Form (BCNF):
Builds upon 3NF and eliminates non-trivial functional dependencies.
Requires that every determinant (attribute determining another attribute) is a candidate key.

The goal of normalization is to minimize data redundancy, maintain data integrity, and facilitate efficient data storage and retrieval. It is important to carefully analyze the requirements and dependencies of a database to determine the appropriate level of normalization to apply.

8. What are transactions in databases? Explain ACID properties.

Transactions in databases represent a logical unit of work that consists of one or more database operations. These operations can include read, write, or modify actions on the database. Transactions are designed to ensure the integrity, consistency, and reliability of data by following a set of properties known as ACID properties.

ACID is an acronym that stands for:
Atomicity:
- Atomicity ensures that a transaction is treated as a single, indivisible unit of work.
- Either all the operations within the transaction are executed successfully, or none of them are.
- If a transaction fails at any point, all changes made by the transaction are rolled back to the state before the transaction began.

Consistency:

- Consistency ensures that a transaction brings the database from one consistent state to another.
- It enforces integrity constraints and rules defined on the database.
- Any transaction must preserve the overall consistency of the database, even if the transaction itself fails.

Isolation:

- Isolation ensures that each transaction is executed in isolation from other concurrent transactions.
- Each transaction operates as if it is the only transaction running on the database, even if there are multiple concurrent transactions.
- Isolation prevents interference between transactions, maintaining data integrity and preventing data corruption.

Durability:

- Durability ensures that once a transaction is committed and changes are written to the database, they are permanent and survive system failures.
- The changes made by a committed transaction should persist even in the event of power outages, crashes, or other failures.
- Durability is typically achieved through mechanisms like write-ahead logging and transaction logs.

9. **Define**

   a. **Views**

   Views are virtual tables derived from predefined queries. They present data from one or more underlying tables in a customized or simplified manner. Views are used to simplify complex queries, provide data security, and present a customized view of data to different users.

   b. **Stored Procedures**

   Stored Procedures are named sets of SQL statements stored and executed as a single unit. They encapsulate frequently performed operations or complex tasks. Stored procedures enhance security, improve performance, promote code reusability, and maintainability by centralizing and streamlining data operations within the database.