**AWS**

**1.OLTP-Online Transaction Processing:**

OLTP systems handle the **real-time transactions**, ensuring smooth processing of frequent data changes.

This includes processing orders, updating inventory, and managing customer accounts.

They prioritize speed and accuracy for a high volume of ongoing transactions.

OLTP systems typically use **normalized databases,** where data is stored efficiently to avoid redundancy and maintain data integrity.

**OLAP-Online Analytical Processing**

OLAP focuses on in-depth **analysis of large datasets** to uncover trends, patterns, and valuable insights.

It's used for tasks like generating reports, data mining, and business intelligence.

OLAP works with **historical data,** often retrieved from transactional databases (OLTP) and data warehouses.

The data is pre-processed and organized for efficient analysis, allowing users to see information from various angles.

**2.Difference between OLTP and OLAP:**

**OLTP:**

* **OLTP** is like the **cash register.**
* It handles your individual order, updates the inventory, and keeps track of your payment in real-time.
* It's fast and focused on processing many transactions smoothly.

**OLAP:**

* **OLAP** is like the **accounting software.**
* It analyzes past sales data to see what dishes are popular, when people order the most, and how much profit they make.
* It doesn't care about your specific order but looks at the bigger picture over time.

**3.DATA BASE Normal forms:**

Database normalization is the process of organizing your relational database to minimize data redundancy and improve data integrity.

It involves structuring your tables and columns to follow a set of rules called normal forms.

There are several normal forms, with increasing levels of strictness:

1. **First Normal Form (1NF):**
   * The most basic level.
   * Eliminates repeating groups of data.
2. **Second Normal Form (2NF):**
   * Meets all requirements of 1NF.
   * Eliminates partial dependencies on the primary key.
3. **Third Normal Form (3NF):**
   * Meets all requirements of 2NF.
   * Eliminates transitive dependencies on the primary key.

There are additional normal forms (Boyce-Codd Normal Form, etc.) but 3NF is generally considered sufficient for most database applications.

Here are the benefits of using normal forms:

* **Reduced data redundancy:** Less wasted space and less chance of inconsistencies.
* **Improved data integrity:** Easier to maintain accurate data.
* **Enhanced data flexibility:** Easier to add new data or modify existing data structures.
* **Faster query performance:** Less complex joins required to retrieve data.

**4.what is Dimension and Fact table:**

**Dimension Table:**

* Think of it as a reference book**.**
* It stores descriptive attributes that provide context for the data in the fact table.
* For example, a dimension table for customers might contain columns like customer ID, name, address, demographics.

**Fact Table:**

* Imagine it as a transaction register**.**
* It stores quantitative data (facts) about a business process, often with foreign keys connecting back to dimension tables.
* A sales fact table might have columns for product ID (linked to a product dimension table), customer ID (linked to a customer dimension table), date, sales amount, and quantity sold.

Here's a key difference:

* Dimension tables typically have fewer rows with more descriptive attributes.
* Fact tables have many rows with numerical data points.

**Types of Dimensions:**

* **Conformed Dimensions:** These are standardized dimensions that can be reused across multiple fact tables in a data warehouse.
* **Slowly Changing Dimensions (SCDs):** Over time, dimension table attributes may change (e.g., customer address).
* SCD strategies define how to handle these changes.
* There are different approaches, such as keeping a history of changes or using a separate table for current and past values.
* **Degenerate Dimensions:** In some cases, a dimension table might only have one attribute.
* This can happen when the dimension is included for filtering purposes but doesn't provide much descriptive detail itself.

**5.Snowflake schema vs Star schema:**

Both snowflake schema and star schema are data warehouse schema designs used for organizing and analyzing large datasets for business intelligence.

**Snowflake Schema:**

* More complex design with a snowflake-like structure.
* Dimension tables are normalized, meaning they strictly avoid data redundancy to improve data integrity. This can involve breaking down dimension tables into further sub-tables.
* More complex to design and implement compared to star schema.
* May require slower query performance due to the need for more joins between tables during data analysis.
* Offers better **data integrity** and reduces data redundancy compared to star schema.

**Star Schema:**

* Simpler design with a star-like structure.
* Dimension tables are denormalized, meaning they might contain some redundant data to improve query performance. This redundancy can save time by avoiding complex joins during data analysis.
* Easier to design and implement compared to snowflake schema.
* Generally, offers faster query performance due to the simpler structure and fewer joins required.
* May require more storage space due to data redundancy in dimension tables.