**1.What is OLTP and OLAP**

Two separate methods for processing data in information systems are called OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing), and they each have a distinctive function.  
  
**Online Transaction Processing :**  
OLTP systems are made for transaction-oriented applications, such as those that require the simultaneous processing of numerous brief online operations (such INSERT, UPDATE, and DELETE).These systems are designed to record and retrieve little amounts of data quickly and efficiently. Normalised schemas are commonly seen in OLTP databases in order to reduce redundancy and guarantee data integrity. Order processing, banking, and airline reservation systems are a few examples of OLTP applications.

**Online Analytical Processing :**   
OLAP systems are intended for analytical and decision-support applications where reporting and analysis require the execution of intricate queries over substantial amounts of data. These systems are designed to allow complicated research, including trend analysis, forecasting, and data mining, by efficiently querying and collecting vast amounts of data. Denormalized or star/snowflake schemas are frequently used by OLAP databases to pre-aggregate data and improve query efficiency. Executive information systems, data warehouses, and business intelligence systems are a few examples of OLAP applications.

**2.Differences between OLTP & OLAP**

| **Aspect** | **OLTP** | **OLAP** |
| --- | --- | --- |
| Purpose | Manage day-to-day transactional data. | Analyze historical data for decision-making. |
| Workload | High volume of short, atomic transactions. | Complex queries over large datasets. |
| Data Structure | Normalized database schemas. | Denormalized or star/snowflake schemas. |
| Response Time | Low, for real-time transaction processing. | May be longer due to complex analytical queries. |
| User Interaction | Operational staff performing transactions. | Analysts, managers, decision-makers performing analysis. |

**3.Database Normal forms(4 normal forms)**

Database normalization is the process of organizing the attributes and tables of a relational database to minimize redundancy and dependency. There are several normal forms, each building upon the previous one, aiming to eliminate data anomalies and ensure data integrity. Here are the first four normal forms:

**First Normal Form (1NF):**

The first normal form requires that each column in a table contains atomic (indivisible) values, and there are no repeating groups or arrays of data. It ensures that each cell contains only a single value, and there are no multi-valued attributes or arrays within a table.

Example: Suppose you have a table of customers where each customer can have multiple phone numbers. To normalize this table to 1NF, you would create a separate table for phone numbers linked to the customer by a foreign key.

**Second Normal Form (2NF):**

The second normal form builds on the first normal form and requires that each non-key attribute is fully functionally dependent on the entire primary key. It eliminates partial dependencies where attributes depend on only a part of the primary key.

Example: In a table where the primary key consists of multiple attributes (composite key), if any non-key attribute depends on only a subset of those attributes, it violates 2NF. To normalize to 2NF, you would move such attributes to a separate table.

**Third Normal Form (3NF):**

The third normal form extends the normalization process by eliminating transitive dependencies. It requires that non-key attributes are not transitively dependent on the primary key via other non-key attributes.

Example: Suppose you have a table of employees with attributes like employee ID, department, and manager. If department depends on employee ID and manager depends on department, there is a transitive dependency. To normalize to 3NF, you would create separate tables for department and manager, each linked to the employee table by foreign keys.

**Boyce-Codd Normal Form (BCNF):**

Boyce-Codd Normal Form is a stricter version of 3NF and applies when there are multiple candidate keys in a table. It requires that every determinant is a candidate key, meaning that every non-trivial functional dependency in the table is a dependency on a superkey.

Example: If a table has multiple candidate keys, and a non-key attribute depends on a subset of those candidate keys, it violates BCNF. To normalize to BCNF, you would need to decompose the table further to ensure that every determinant is a candidate key.

**4.Dimension VS Fact table**

Dimension tables and fact tables are fundamental components of a dimensional model used in data warehousing and business intelligence systems. They serve distinct purposes and contain different types of data:

**Dimension Table:**

* Dimension tables contain descriptive attributes or characteristics by which data is analyzed or categorized.
* They provide the context or perspective for measuring and analyzing facts in a data warehouse.
* Dimension tables are typically relatively small in size compared to fact tables.
* Examples of dimension tables include:
* Customer dimension (containing attributes like customer ID, name, address).
* Product dimension (containing attributes like product ID, name, category).
* Time dimension (containing attributes like date, month, year).
* Dimension tables are usually connected to fact tables through foreign key relationships.

**Fact Table:**

Fact tables contain numerical measures or metrics, as well as keys to dimension tables, enabling analysis at different levels of granularity.

* They store the quantitative data that is being analyzed or measured.
* Fact tables are typically larger in size compared to dimension tables.
* Examples of fact tables include:
* Sales fact table (containing measures like revenue, quantity sold) linked to dimensions such as time, product, and customer.
* Inventory fact table (containing measures like stock levels, reorder quantities) linked to dimensions such as product and location.
* Web traffic fact table (containing measures like page views, unique visitors) linked to dimensions such as time, website, and geography.
* Fact tables often contain foreign keys that establish relationships with one or more dimension tables.

**5.Types of Dimensions**

Certainly! Here's a list of common dimensions used in data warehousing:

* Time Dimension
* Product Dimension
* Customer Dimension
* Geographic Dimension
* Sales Channel Dimension
* Employee Dimension
* Promotion Dimension
* Supplier Dimension
* Market Segment Dimension
* Currency Dimension
* Organization Dimension
* Asset Dimension
* Account Dimension
* Inventory Dimension
* Service Dimension

These dimensions cover various aspects of business operations and facilitate comprehensive analysis and reporting in data warehousing and business intelligence systems.

**6.Snowflake VS Star Schema**

Certainly! Here's a comparison of Snowflake and Star schemas in table format:

| **Aspect** | **Star Schema** | **Snowflake Schema** |
| --- | --- | --- |
| Structure | Denormalized structure with dimension tables linked directly to the fact table | Normalized structure with dimension tables normalized into multiple related tables |
| Dimension Tables | Dimension tables are independent and not normalized | Dimension tables may be normalized into multiple related tables |
| Complexity | Simpler and more straightforward design | More complex due to normalization and multiple table joins |
| Performance | Generally better performance for simple queries | May be slower for complex queries due to additional joins |
| Storage | May require more storage space due to denormalization | Typically requires less storage space due to normalization |
| Flexibility | Less flexible but easier to understand and maintain | More flexible and scalable, suitable for evolving data models |

This table provides a concise comparison of the key differences between Snowflake and Star schemas across various aspects.