DEFINE OLTP AND OLAP?

**OLTP:** Online Transaction Processing, or OLTP, is a kind of data processing that includes carrying out several simultaneous operations, such as text messaging, order entry, online banking, and shopping. Traditionally, they have been referred to as economic or financial transactions; the information is safeguarded and preserved so that an organization can access it at any time for reporting or accounting needs.

In the past, OLTP was limited to real-world interactions in which something was exchanged–money, products, information, request for services, and so on. But the definition of transaction in this context has expanded over the years, especially since the advent of the internet, to encompass any kind of digital interaction or engagement with a business that can be triggered from anywhere in the world and via any web-connected sensor. It also includes any kind of interaction or action such as downloading pdfs on a web page, viewing a specific video, or automatic maintenance triggers or comments on social channels that maybe critical for a business to record to serve their customers better.

**OLAP:** Online analytical processing, or OLAP, is a computing technique for handling complicated analytical applications. Large volumes of data from a data mart, data warehouse, or other data storage unit are processed using this business intelligence tool. Cubes are used by OLAP to show data in several categories. An OLAP cube, in contrast to a typical graph with only two axes, includes three dimensions to display a third category. For financial planning, budgeting, and forecasting, businesses employ this data representation.   
Three data categories are usually present in an OLAP cube: products, time, and location. Other data types, such as customers or demographics, might be included in cubes. To determine current trends and examine how the many components connect to one another, the cube arranges data along its three axes.

# DIFFERENCE BETWEEN OLAP AND OLTP:

This table provides a simplified overview of the main differences between OLAP and OLTP.

Top of Form

| **Aspect** | **OLAP** | **OLTP** |
| --- | --- | --- |
| **Purpose** | Analytical processing for decision-making | Transaction processing for day-to-day operations |
| **Usage** | Queries and analysis of historical data | Recording, updating, and retrieving real-time data |
| **Data Type** | Aggregated, summarized data | Detailed, atomic data |
| **Database Structure** | Denormalized, star or snowflake schema | Normalized schema for efficiency in transactions |
| **Response Time** | Longer, optimized for complex queries | Shorter, optimized for quick transactions |
| **Users** | Typically, business analysts and decision-makers | End-users, employees, customers |
| **Examples** | Business Intelligence systems, Data Warehouses | Online banking systems, e-commerce platforms |

# TYPES OF NORMAL FORMS:

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

Every column in a table must hold atomic values, meaning each value should be indivisible.

Eliminates repeating groups within a table.

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

Meets all the criteria of 1NF.

No partial dependencies exist, meaning every non-prime attribute is fully functionally dependent on the primary key.

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

Meets all the criteria of 2NF.Eliminates transitive dependencies, ensuring that no non-prime attribute is dependent on another non-prime attribute.

These normal forms help in organizing and structuring relational databases efficiently, reducing redundancy and improving data integrity.

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

BCNF is a stricter form of 3NF that ensures that each determinant in a table is a candidate key. In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.

**Fourth Normal Form (4NF):**

4NF is a further refinement of BCNF that ensures that a table does not contain any multi-valued dependencies.

**Fifth Normal Form (5NF):** 5NF is the highest level of normalization and involves decomposing a table into smaller tables to remove data redundancy and improve data integrity.

# DIMENSION Vs FACT TABLE:

| **Aspect** | **Dimension Table** | **Fact Table** |
| --- | --- | --- |
| **Purpose** | Contains descriptive attributes about data | Contains quantitative measurements or metrics |
| **Contents** | Non-numerical textual or categorical data | Numerical data, often with foreign keys to dimensions |
| **Size** | Typically smaller in size compared to fact tables | Tends to be larger due to storing detailed data |
| **Relationships** | Often connected to fact tables via foreign keys | Usually contains foreign keys to dimension tables |
| **Example** | Customer, Product, Time dimension tables | Sales, Orders, Inventory fact tables |

# TYPES OF DIMENSION TABLES:

In data warehousing, dimension tables are used to provide descriptive information about the data stored in fact tables. There are several types of dimension tables commonly used in data warehousing:

Slowly Changing Dimensions (SCDs): These dimensions capture changes in data over time. There are different types of SCDs:

Type 1 SCD: Overwrites existing data with new data, losing historical information.

Type 2 SCD: Maintains historical data by creating new records for changes and preserving old records.

Type 3 SCD: Keeps both old and new values in separate columns, allowing limited historical tracking.

Conformed Dimensions: These dimensions are shared and consistent across multiple data marts or data warehouse systems. They ensure consistency and allow for integration of data across different parts of the organization.

Junk Dimensions: These dimensions are used to store low-cardinality flags or indicators that do not fit into other dimensions. They help in simplifying the design by reducing the number of dimension tables.

Degenerate Dimensions: These dimensions are derived from the fact table itself. They contain attributes that are part of the fact table's composite primary key but are not part of any dimension table.

Role-Playing Dimensions: These are dimensions that are used multiple times in a single fact table, but with different meanings. For example, a "Date" dimension could be used for "Order Date," "Ship Date," and "Delivery Date."

Derived Dimensions: These dimensions are created by deriving attributes from existing dimensions or other sources. For instance, a "Age Group" dimension can be derived from a "Date of Birth" attribute in a "Customer" dimension.

Snowflake Dimensions: These dimensions are normalized, meaning that hierarchies are stored in separate tables instead of being denormalized into a single table. This can help in saving storage space but may complicate queries due to additional joins.

# SNOWFLAKE Vs STAR SCHEMA:

**Snowflake Schema:**

Normalization: Snowflake schema is normalized, meaning dimension tables are broken down into multiple related tables.

Complexity: It tends to have more complex relationships due to normalization.

Storage Efficiency: It can save storage space as redundant data is reduced.

Query Performance: Query performance might be impacted due to additional joins required to retrieve data from multiple tables.

**Star Schema:**

Denormalization: Star schema is denormalized, with dimension tables directly connected to the fact table.

Simplicity: It offers simpler relationships between tables compared to snowflake schema.

Storage Requirements: It might require more storage space due to redundant data in dimension tables.

Query Performance: Generally, offers better query performance as it involves fewer joins, making queries simpler and faster.

In essence, Snowflake schema prioritizes normalization and storage efficiency, while Star schema prioritizes simplicity and query performance. The choice between the two depends on the specific requirements of the data warehousing environment.