Characteristics of a data warehouse for an interview:

**Subject Oriented**: Data is organized around specific subjects or business areas, such as sales, marketing, or finance. For example, in a retail company's data warehouse, you might have subjects like sales, inventory, and customer data.

**Integrated**: Data from different sources is integrated to ensure consistency and accuracy. For instance, in a healthcare data warehouse, patient information from various departments like cardiology, radiology, and pathology are integrated to provide a comprehensive view of the patient's health history.

**Time-Variant**: Data in a warehouse includes historical information, allowing analysis of trends and patterns over time. For instance, in a financial institution's data warehouse, you could analyze stock market trends over the past decade.

**Non-Volatile**: Once data is stored in the warehouse, it is not altered or updated. Instead, new data is appended to the existing data. For example, in an e-commerce data warehouse, past sales transactions remain unchanged even if product prices are updated.

Types of data warehouse:

Data warehouses can generally be classified into several basic types based on their architecture, purpose, and usage. Here are some common types:

**Enterprise Data Warehouse (EDW)**:

Centralized repository for all structured data within an organization.

Integrates data from multiple sources across various departments.

Supports decision-making processes across the entire enterprise.

Designed for long-term storage and complex analytics.

**Operational Data Store (ODS)**:

Focuses on real-time or near real-time data integration.

Stores current or short-term operational data.

Provides a consistent and integrated view of operational data.

Often used for operational reporting and data synchronization.

**Data Mart**:

Subset of a data warehouse focused on specific business lines, departments, or functions.

Contains summarized or aggregated data tailored to the needs of a particular user group.

Provides faster query performance for targeted analysis.

Can be either independent or integrated with an enterprise data warehouse.

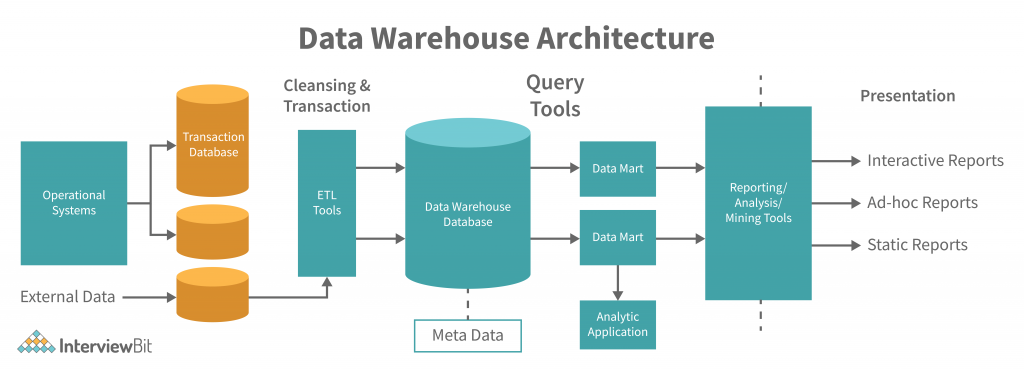
**Virtual Data Warehouse**:

Does not store data physically; instead, it provides a logical view of data spread across different sources.

Integrates data from disparate sources in real-time or near real-time.

Offers a unified interface for querying and analysis without the need for data movement or replication.

Useful for organizations with distributed data sources or where data movement is impractical or undesirable.



**Data Mart Definition:** A data mart is a subset of a data warehouse that is focused on specific business functions or departments within an organization. It contains a curated set of data relevant to a particular group's or department's needs, making it easier for users to access and analyze data related to their specific area of interest or responsibility.

**Advantages of Data Mart:**

**Focused Analysis**: Data marts provide a focused and tailored dataset for analysis, catering to the specific needs of a department or business function.

**Improved Performance**: Since data marts contain a subset of data, querying and reporting performance can be optimized, leading to faster response times.

**Simplified Access**: Users can access relevant data more easily, without having to navigate through the entire enterprise data warehouse, enhancing usability and efficiency.

**Customization**: Data marts can be customized to meet the unique requirements of different departments or business units, allowing for greater flexibility and adaptability.

**Cost-Effectiveness**: Implementing and maintaining data marts can be more cost-effective than managing a comprehensive enterprise data warehouse, especially for smaller departments or projects.

**Disadvantages of Data Warehouse:**

**Complexity**: Building and managing a data warehouse requires significant time, resources, and expertise. The complexity of integrating data from multiple sources and ensuring data quality can pose challenges.

**Cost**: Data warehouse implementation and maintenance costs can be substantial, including expenses for hardware, software, staffing, and ongoing support and maintenance.

**Data Latency**: In some cases, there may be delays in updating data in the data warehouse, leading to potential data latency issues for real-time or near-real-time analytics.

**Scalability Challenges**: Scaling a data warehouse to accommodate growing data volumes and user demands can be challenging, requiring careful planning and investment in infrastructure.

**Data Governance Concerns**: Ensuring data governance, security, and compliance within a data warehouse environment can be complex, particularly when dealing with sensitive or regulated data.

**Single Point of Failure**: A centralized data warehouse can become a single point of failure if it experiences downtime or data corruption, potentially disrupting business operations.

Types of data marts:

**Dependent Data Mart**:

Derived from a centralized data warehouse.

Contains a subset of data relevant to a specific business function or department.

Example: A sales data mart created from an enterprise data warehouse, focusing exclusively on sales-related data for the sales department's analysis.

**Independent Data Mart**:

Created independently from the data warehouse.

Often developed to meet the unique needs of a specific business unit or department.

Example: A marketing data mart developed separately from the main data warehouse to cater to the marketing team's analysis of campaign effectiveness and customer behavior.

**Distributed Data Mart**:

Geographically distributed across different locations.

Data is stored and managed locally but can be integrated or federated for global analysis.

Example: A multinational corporation with regional sales data marts in different countries, which are consolidated for global sales performance analysis.

**Virtual Data Mart**:

Doesn't physically store data but provides a logical view of integrated data from multiple sources.

Queries are executed across source systems in real-time.

Example: A virtual marketing data mart that aggregates data from various online advertising platforms like Google Ads, Facebook Ads, and Twitter Ads for unified campaign performance analysis.

**Benefits of ODS:**

**ODS Definition:** An Operational Data Store (ODS) is a centralized database that integrates data from multiple operational systems in real-time or near-real-time. It serves as a repository for current and detailed transactional data, making it readily accessible for operational reporting and decision-making.

**Real-Time Insights**: Provides immediate access to current transactional data, enabling real-time insights and decision-making.

**Operational Efficiency**: Supports operational reporting and monitoring, facilitating efficient management of day-to-day operations.

**Data Integration**: Integrates data from disparate operational systems, providing a unified view of organizational activities and processes.

**Improved Data Quality**: Ensures data accuracy, consistency, and completeness through data validation and cleansing processes.

**Flexibility**: Adaptable to changing business requirements and data sources, accommodating evolving operational needs.

**Support for Analytics**: Serves as a foundation for downstream analytical processes, feeding data into data warehouses or analytical platforms for further analysis and reporting.

Ewd approaches:

**Top-Down Approach**:

Involves designing and building the entire data warehouse as a single, comprehensive solution.

Emphasizes a centralized and holistic view of the organization's data.

Example: A large retail company implements a top-down approach to create an enterprise data warehouse that integrates data from all departments, including sales, inventory, and finance, to support company-wide analytics and reporting needs.

**Bottom-Up Approach**:

Begins with building departmental or functional data marts independently.

Data marts are later integrated to form a centralized enterprise data warehouse.

Provides quick wins and delivers value to specific business units before expanding to the entire organization.

Example: A healthcare organization starts by implementing separate data marts for different departments, such as cardiology, radiology, and pharmacy. These data marts are then integrated to form an enterprise data warehouse that supports comprehensive analytics across all areas of patient care.

**Hybrid Approach**:

Combines elements of both top-down and bottom-up approaches.

Begins with building a core data warehouse foundation using a top-down approach.

Iteratively adds departmental data marts using a bottom-up approach to address specific business needs.

Example: A financial services company adopts a hybrid approach by initially building a central data warehouse to store core financial data. Later, it implements additional data marts for specific functions like risk management, investment analysis, and customer relationship management, integrating them with the core data warehouse as needed.

Normalization is a database design technique used to organize data into well-structured relations (tables) to minimize redundancy and dependency. It involves breaking down a large table into smaller, related tables and establishing relationships between them. Here, I'll explain the first three normal forms (1NF, 2NF, and 3NF) with examples, along with tables that you can copy.

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

In the first normal form, each column in a table contains atomic (indivisible) values, and there are no repeating groups or arrays.

Example:

Consider a table storing information about students and the courses they are enrolled in:

| **Student ID** | **Student Name** | **Courses** |
| --- | --- | --- |
| 101 | John Doe | Math, Physics |
| 102 | Jane Smith | Chemistry |
| 103 | Alice Johnson | Physics, Biology |

To convert this into 1NF, we need to split the repeating values in the **Courses** column:

| **Student ID** | **Student Name** | **Course** |
| --- | --- | --- |
| 101 | John Doe | Math |
| 101 | John Doe | Physics |
| 102 | Jane Smith | Chemistry |
| 103 | Alice Johnson | Physics |
| 103 | Alice Johnson | Biology |

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

In the second normal form, a table must be in 1NF, and each non-key attribute must be fully functionally dependent on the entire primary key.

Example:

Let's extend our previous example with grades:

| **Student ID** | **Student Name** | **Course** | **Grade** |
| --- | --- | --- | --- |
| 101 | John Doe | Math | A |
| 101 | John Doe | Physics | B |
| 102 | Jane Smith | Chemistry | B+ |
| 103 | Alice Johnson | Physics | A- |
| 103 | Alice Johnson | Biology | A |

Here, the primary key is **(Student ID, Course)**. **Grade** is dependent only on the **Course**. So, we remove it to a new table:

**Student\_Courses:**

| **Student ID** | **Course** |
| --- | --- |
| 101 | Math |
| 101 | Physics |
| 102 | Chemistry |
| 103 | Physics |
| 103 | Biology |

**Course\_Grades:**

| **Student ID** | **Course** | **Grade** |
| --- | --- | --- |
| 101 | Math | A |
| 101 | Physics | B |
| 102 | Chemistry | B+ |
| 103 | Physics | A- |
| 103 | Biology | A |

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

In the third normal form, a table must be in 2NF, and no transitive dependency should exist.

Example:

Let's say we have a table containing information about students and their departments:

| **Student ID** | **Student Name** | **Department** | **Department Location** |
| --- | --- | --- | --- |
| 101 | John Doe | Computer Science | Building A |
| 102 | Jane Smith | Chemistry | Building B |
| 103 | Alice Johnson | Physics | Building A |

Here, **Department Location** depends only on **Department**, not on **Student ID**. So, we remove it to a new table:

**Student\_Departments:**

| **Student ID** | **Department** |
| --- | --- |
| 101 | Computer Science |
| 102 | Chemistry |
| 103 | Physics |

**Departments:**

| **Department** | **Department Location** |
| --- | --- |
| Computer Science | Building A |
| Chemistry | Building B |
| Physics | Building A |

These are simplified examples of normalization steps. In practice, database normalization can involve more complex scenarios and additional normalization forms like BCNF (Boyce-Codd Normal Form) and 4NF.

Sure, here are the tables you provided in the format you requested:

**Employee Table:**

| **Employee\_ID** | **Employee\_Name** | **Department** | **Manager** |
| --- | --- | --- | --- |
| 101 | John Smith | Sales | Mary Brown |
| 102 | Alice Johnson | Marketing | Mark Davis |

**Employee\_ID and Employee\_Name Table:**

| **Employee\_ID** | **Employee\_Name** |
| --- | --- |
| 101 | John Smith |
| 102 | Alice Johnson |

**Department Table:**

| **Department\_ID** | **Department** |
| --- | --- |
| 1 | Sales |
| 2 | Marketing |

**Employee\_ID, Department\_ID, and Manager Table:**

| **Employee\_ID** | **Department\_ID** | **Manager** |
| --- | --- | --- |
| 101 | 1 | Mary Brown |
| 102 | 2 | Mark Davis |

**Types of Facts:**

**Transactional Facts:**

Represent detailed, atomic data at the lowest level of granularity.

Typically captured from operational systems and used for detailed analysis.

Example: Sales transactions, customer orders, or stock trades.

**Additive Facts:**

Numeric facts that can be aggregated across all dimensions in a data model.

Aggregation functions like sum or average are applicable.

Example: Sales revenue, quantity sold, or total cost.

**Semi-Additive Facts:**

Numeric facts that can be aggregated across some dimensions but not all.

Aggregation functions are applicable only to certain dimensions.

Example: Inventory levels, account balances, or product prices over time.

**Non-Additive Facts:**

Numeric facts that cannot be aggregated at all.

Aggregating these facts would result in meaningless or misleading results.

Example: Unit price, discount percentage, or average interest rate.

<https://helicaltech.com/types-of-facts-in-data-warehouse>

[What are the types of facts in a data warehouse? (educative.io)](https://www.educative.io/answers/what-are-the-types-of-facts-in-a-data-warehouse)

**Transaction Fact Table**:

The **transaction fact table** captures individual business events or transactions.

Each row in this table corresponds to a specific transaction (e.g., a sale, an order, or a customer interaction).

It contains **additive facts**, such as **sales revenue**, **quantity sold**, or **profit**.

[Example: An e-commerce company’s **sales transaction data**](https://www.javatpoint.com/types-of-facts-table)[1](https://www.javatpoint.com/types-of-facts-table)[2](https://medium.com/@datawithzon/fact-dimension-tables-what-are-they-and-how-can-we-use-them-667b79426138).

**Key Points:**

Represents events at the primary point (e.g., sales, orders).

Often structured in a one-dimensional framework.

Contains raw, detailed data.

**Example: Sales Transactional Fact Table**

| **Transaction\_ID** | **Date** | **Product\_ID** | **Customer\_ID** | **Quantity** | **Sales\_Amount** |
| --- | --- | --- | --- | --- | --- |
| 1 | 2024-03-01 | 101 | 201 | 2 | 200.00 |
| 2 | 2024-03-01 | 102 | 202 | 1 | 100.00 |
| 3 | 2024-03-02 | 101 | 203 | 3 | 300.00 |

**Snapshot Fact Table**:

The **snapshot fact table** describes the state of things at a specific point in time.

It contains **semi-additive** and **non-additive facts**.

[Example: **Daily balance** (expressed by customer dimension but not time dimension) or performance summaries of salespeople during specific periods](https://www.javatpoint.com/types-of-facts-table)[1](https://www.javatpoint.com/types-of-facts-table).

**Key Points:**

Captures data at regular intervals (e.g., end of the day, week, or month).

Useful for tracking changes over time.

**Example: Inventory Snapshot Fact Table**

| **Snapshot\_Date** | **Product\_ID** | **Location\_ID** | **On\_Hand\_Quantity** | **Reserved\_Quantity** | **Available\_Quantity** |
| --- | --- | --- | --- | --- | --- |
| 2024-03-01 | 101 | 501 | 100 | 20 | 80 |
| 2024-03-01 | 102 | 502 | 150 | 30 | 120 |
| 2024-03-02 | 101 | 501 | 95 | 25 | 70 |

**Accumulated Fact Table**:

The **accumulated fact table** shows the activity of a process with a clear beginning and end.

Example: Tracking the progress of an order as it moves through processing steps.

Rows in this table are updated as steps toward completing the process are achieved.

**Key Points:**

* Reflects progress over time.
* Useful for process-oriented analysis.
* Tracks the lifecycle of each order from the time it's received to when it's delivered.
* Captures key events or milestones associated with each order, such as shipping and delivery dates.
* Provides a comprehensive view of order processing activities, allowing for analysis of order fulfillment efficiency and cycle times.
* Enables tracking of historical order statuses and performance metrics over time.

**Example: Order Processing Accumulated Fact Table**

| **Order\_ID** | **Order\_Date** | **Customer\_ID** | **Product\_ID** | **Order\_Status** | **Order\_Amount** | **Shipped\_Date** | **Delivered\_Date** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2024-03-01 | 101 | 201 | Received | 200.00 | NULL | NULL |
| 2 | 2024-03-02 | 102 | 202 | In Progress | 150.00 | NULL | NULL |
| 3 | 2024-03-03 | 103 | 203 | Delivered | 300.00 | 2024-03-04 | 2024-03-05 |

**Fact-less Fact Tables**:

These tables contain no measures (facts).

Used to capture the action of a business process without quantitative data.

Example: A fact-less fact table capturing business actions without specific measurements.

**Types of Dimensions:**

**Conformed Dimension:**

A dimension that is shared and consistent across multiple data marts or data warehouse components.

Provides a consistent view of the same dimension across different parts of the organization.

Example: Date dimension, which contains attributes like year, month, and day, used consistently across various analytical applications.

**Example: Date Dimension Table (Conformed Dimension)**

| Date\_ID | Date | Day | Month | Year |
| --- | --- | --- | --- | --- |
| 1 | 2024-03-01 | Monday | March | 2024 |
| 2 | 2024-03-02 | Tuesday | March | 2024 |
| ... | ... | ... | ... | ... |

**Degenerate Dimension:**

A dimension that is derived from the fact table and does not have a corresponding dimension table.

Represents attributes that are not associated with any specific dimension but are useful for analysis.

Example: Order number or transaction ID, which are unique identifiers for each transaction stored directly in the fact table.

**Example: Order Number (Degenerate Dimension)**

| Order\_Number |
| --- |
| ORD12345 |
| ORD12346 |
| ORD12347 |

**Junk Dimension:**

A dimension that contains flags or indicators that do not fit into existing dimensions.

Consolidates low-cardinality attributes into a single dimension to reduce the number of dimension tables.

Example: A junk dimension named "Payment Method" containing flags for cash, credit card, or check payment options.

**Example: Promotion Dimension Table (Junk Dimension)**

| Promotion\_ID | Promotion\_Type |
| --- | --- |
| 1 | Discount |
| 2 | Coupon |
| 3 | Free Shipping |

**Role-Playing Dimension:**

A dimension that is used multiple times in a fact table with different meanings.

Represents the same concept but is interpreted differently depending on the context.

Example: Date dimension used to analyze both order date and ship date in a sales transaction.

**Example: Date Dimension (Role-Playing Dimension)**

| **Order\_Date\_ID** | **Ship\_Date\_ID** | **Delivery\_Date\_ID** |
| --- | --- | --- |
| 1 | 2 | 3 |
| 2 | 3 | 4 |
| ... | ... | ... |

**star schema:**

Fact Table: Sales

The fact table contains quantitative measures (facts) of business processes, surrounded by foreign keys to related dimension tables.

| Transaction\_ID | Date\_ID | Product\_ID | Customer\_ID | Sales\_Amount |
| --- | --- | --- | --- | --- |
| 1 | 101 | 201 | 301 | 100.00 |
| 2 | 102 | 202 | 302 | 150.00 |
| 3 | 103 | 203 | 303 | 200.00 |

**Transaction\_ID:** Unique identifier for each transaction.

**Date\_ID:** Foreign key referencing the Date dimension.

**Product\_ID:** Foreign key referencing the Product dimension.

**Customer\_ID:** Foreign key referencing the Customer dimension.

**Sales\_Amount:** The measure representing the sales amount.

Dimension Tables:

**Date Dimension:**

| Date\_ID | Date | Day | Month | Year |
| --- | --- | --- | --- | --- |
| 101 | 2024-03-01 | Monday | March | 2024 |
| 102 | 2024-03-02 | Tuesday | March | 2024 |
| ... | ... | ... | ... | ... |

**Product Dimension:**

| Product\_ID | Product\_Name | Category | Brand |
| --- | --- | --- | --- |
| 201 | Product A | Electronics | Brand X |
| 202 | Product B | Clothing | Brand Y |
| ... | ... | ... | ... |

**Customer Dimension:**

| Customer\_ID | Customer\_Name | City | Country |
| --- | --- | --- | --- |
| 301 | John Smith | New York | USA |
| 302 | Alice Johnson | London | UK |
| ... | ... | ... | ... |

Explanation:

The **Sales** fact table contains quantitative measures such as sales amount and foreign keys referencing related dimensions.

The **Date**, **Product**, and **Customer** dimension tables provide descriptive attributes for the data in the fact table, such as date, product details, and customer information.

The fact table is connected to dimension tables through foreign key relationships.

The star schema is denormalized, optimized for query performance, and provides a simple and intuitive structure for analytical queries and reporting.

Queries involving aggregations, filtering, and grouping can be efficiently performed using the star schema design.

**Snowflake schema:**

The snowflake schema is an extension of the star schema where dimension tables are normalized into multiple related tables, forming a shape similar to a snowflake. This normalization reduces data redundancy and improves data integrity, but it may lead to more complex queries. Here's an example of a snowflake schema with tables:

Example: Sales Data Warehouse Snowflake Schema

**Fact Table: Sales**

| **Transaction\_ID** | **Date\_ID** | **Product\_ID** | **Customer\_ID** | **Sales\_Amount** |
| --- | --- | --- | --- | --- |
| 1 | 101 | 201 | 301 | 100.00 |
| 2 | 102 | 202 | 302 | 150.00 |
| 3 | 103 | 203 | 303 | 200.00 |

**Dimension Tables:**

**Date Dimension:**

| **Date\_ID** | **Date** | **Day** | **Month** | **Year** |
| --- | --- | --- | --- | --- |
| 101 | 2024-03-01 | Monday | March | 2024 |
| 102 | 2024-03-02 | Tuesday | March | 2024 |
| ... | ... | ... | ... | ... |

**Product Dimension:**

| **Product\_ID** | **Product\_Name** | **Category\_ID** |
| --- | --- | --- |
| 201 | Product A | 501 |
| 202 | Product B | 502 |
| ... | ... | ... |

**Product Category Dimension:**

| **Category\_ID** | **Category\_Name** |
| --- | --- |
| 501 | Electronics |
| 502 | Clothing |
| ... | ... |

**Customer Dimension:**

| **Customer\_ID** | **Customer\_Name** | **City\_ID** |
| --- | --- | --- |
| 301 | John Smith | 701 |
| 302 | Alice Johnson | 702 |
| ... | ... | ... |

**Customer City Dimension:**

| **City\_ID** | **City** | **Country\_ID** |
| --- | --- | --- |
| 701 | New York | 901 |
| 702 | London | 902 |
| ... | ... | ... |

**Customer Country Dimension:**

| **Country\_ID** | **Country** |
| --- | --- |
| 901 | USA |
| 902 | UK |
| ... | ... |

Explanation:

The **Sales** fact table remains the same as in the star schema.

The **Date**, **Product**, and **Customer** dimensions are normalized into multiple related tables.

The **Product** dimension is split into two tables: **Product** and **Product Category** to eliminate redundancy and maintain data integrity.

Similarly, the **Customer** dimension is split into three tables: **Customer**, **Customer City**, and **Customer Country**.

Each dimension table is connected to its related tables through foreign key relationships.

The snowflake schema reduces data redundancy and maintains data integrity by normalizing dimension tables into multiple related tables.

Queries may involve more joins compared to the star schema, potentially impacting query performance, but it offers improved data consistency and integrity.

**Natural Key:**

A natural key is a unique identifier for a record that is inherent to the data itself. It is typically a piece of data that already exists in the real world and uniquely identifies each record without the need for additional processing or manipulation. Natural keys are often derived from business data and have a meaningful context within the domain of the data.

**Example:**

Consider a table storing information about students in a school. The Student\_ID could serve as a natural key because it uniquely identifies each student, and it's a piece of data already existing in the real world.

| Student\_ID | Student\_Name | Age | Gender | Grade |
| --- | --- | --- | --- | --- |
| 123456 | John Smith | 15 | Male | 10 |
| 789012 | Alice Johnson | 16 | Female | 11 |

In this example, the Student\_ID is a natural key because it is derived from the real-world data (e.g., student registration number) and uniquely identifies each student.

**Surrogate Key:**

A surrogate key is an artificially generated unique identifier for a record in a table. Unlike natural keys, surrogate keys have no inherent meaning or significance in the real world and are typically used solely for database management purposes. Surrogate keys are often integers that are automatically generated by the database management system.

**Example:**

Continuing with the student example, suppose we add a Student table with a surrogate key:

| Surrogate\_Key | Student\_ID | Student\_Name | Age | Gender | Grade |
| --- | --- | --- | --- | --- | --- |
| 1 | 123456 | John Smith | 15 | Male | 10 |
| 2 | 789012 | Alice Johnson | 16 | Female | 11 |

**Natural Key:**

Derived from real-world data.

Typically has meaning within the domain of the data.

May require additional validation to ensure uniqueness.

**Surrogate Key:**

Artificially generated by the database management system.

Has no inherent meaning in the real world.

Guarantees uniqueness within the database.

**Types of Fact tables:**

Fact tables in data warehouses typically store numerical data (facts) and provide the foundation for analytics and reporting. There are several types of fact tables, each serving different purposes based on the nature of the data they contain. Here are some common types of fact tables with examples:

**1. Transactional Fact Table:**

Transactional fact tables capture individual business transactions at a detailed level.

Example: Sales Transactional Fact Table

| **Transaction\_ID** | **Date** | **Product\_ID** | **Customer\_ID** | **Quantity** | **Sales\_Amount** |
| --- | --- | --- | --- | --- | --- |
| 1 | 2024-03-01 | 101 | 201 | 2 | 200.00 |
| 2 | 2024-03-01 | 102 | 202 | 1 | 100.00 |
| 3 | 2024-03-02 | 101 | 203 | 3 | 300.00 |

**2. Snapshot Fact Table:**

Snapshot fact tables capture metrics at a specific point in time, often used for periodic reporting or analysis.

Example: Inventory Snapshot Fact Table

| **Snapshot\_Date** | **Product\_ID** | **Location\_ID** | **On\_Hand\_Quantity** | **Reserved\_Quantity** | **Available\_Quantity** |
| --- | --- | --- | --- | --- | --- |
| 2024-03-01 | 101 | 501 | 100 | 20 | 80 |
| 2024-03-01 | 102 | 502 | 150 | 30 | 120 |
| 2024-03-02 | 101 | 501 | 95 | 25 | 70 |

**3. Accumulating Snapshot Fact Table:**

Accumulating snapshot fact tables track the progress of a process over time, capturing different stages or milestones.

Example: Order Processing Accumulating Snapshot Fact Table

| **Order\_ID** | **Order\_Date** | **Product\_ID** | **Customer\_ID** | **Order\_Status** | **Order\_Amount** | **Shipped\_Date** | **Delivered\_Date** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2024-03-01 | 101 | 201 | Received | 200.00 | NULL | NULL |
| 2 | 2024-03-02 | 102 | 202 | In Progress | 150.00 | NULL | NULL |
| 3 | 2024-03-03 | 101 | 203 | Delivered | 300.00 | 2024-03-04 | 2024-03-05 |

**4. Factless Fact Table:**

Factless fact tables contain only foreign keys and no measures, used to represent events or occurrences without numerical data.

Example: Student Attendance Factless Fact Table

| **Date** | **Student\_ID** | **Course\_ID** | **Attendance\_Status** |
| --- | --- | --- | --- |
| 2024-03-01 | 101 | 201 | Present |
| 2024-03-01 | 102 | 201 | Absent |
| 2024-03-02 | 101 | 201 | Present |

**5. Periodic Snapshot Fact Table:**

Periodic snapshot fact tables capture metrics at regular intervals, typically used for trend analysis.

Example: Monthly Sales Snapshot Fact Table

| **Month** | **Product\_ID** | **Sales\_Amount** |
| --- | --- | --- |
| 2024-01 | 101 | 1000.00 |
| 2024-01 | 102 | 1500.00 |
| 2024-02 | 101 | 1200.00 |

**Types of Slowly Changing Dimensions:**

**Type 1: Overwrite (No History Maintenance):**

Overwrites existing attribute values with new values without preserving historical data.

Simplest and least resource-intensive approach.

Suitable for dimensions where historical data is not required or changes are insignificant.

**Type 1 Example:**

Consider a Product dimension where the price of a product changes over time. In Type 1 SCD, the old price is overwritten with the new price without preserving historical data.

| Product\_ID | Product\_Name | Price |
| --- | --- | --- |
| 101 | Laptop | $1000 |
| 102 | Smartphone | $800 |
| 103 | T-shirt | $20 |

After Price Change:

| Product\_ID | Product\_Name | Price |
| --- | --- | --- |
| 101 | Laptop | $1200 |
| 102 | Smartphone | $800 |
| 103 | T-shirt | $20 |

**Type 2: Add New Row (Preserve History):**

Adds a new row with the updated attribute values while preserving the old row.

Maintains a complete history of changes, allowing for accurate historical reporting.

Requires additional storage space but provides comprehensive historical data.

**Type 2 Example:**

Using the same Product dimension, in Type 2 SCD, a new row is added with the updated price while preserving the old row.

| Product\_ID | Product\_Name | Price | Start\_Date | End\_Date |
| --- | --- | --- | --- | --- |
| 101 | Laptop | $1000 | 2023-01-01 | 2024-03-05 |
| 102 | Smartphone | $800 | 2023-01-01 | 2024-03-05 |
| 103 | T-shirt | $20 | 2023-01-01 | 2024-03-05 |

After Price Change:

| Product\_ID | Product\_Name | Price | Start\_Date | End\_Date |
| --- | --- | --- | --- | --- |
| 101 | Laptop | $1000 | 2023-01-01 | 2024-03-05 |
| 102 | Smartphone | $800 | 2023-01-01 | 2024-03-05 |
| 103 | T-shirt | $20 | 2023-01-01 | 2024-03-05 |
| 101 | Laptop | $1200 | 2024-03-06 |  |

**Type 3: Add New Columns (Limited History Maintenance):**

Adds new columns to the dimension table to store a limited history of changes.

Retains previous and current attribute values, allowing for some historical analysis.

Suitable for dimensions where only recent changes need to be tracked.

**Type 3 Example:**

Extending the Product dimension with Type 3 SCD, additional columns are added to store limited historical data.

| Product\_ID | Product\_Name | Price | Previous\_Price | Previous\_Price\_Date |
| --- | --- | --- | --- | --- |
| 101 | Laptop | $1000 | $1200 | 2024-03-05 |
| 102 | Smartphone | $800 | - | - |
| 103 | T-shirt | $20 | - | - |

After Price Change:

| Product\_ID | Product\_Name | Price | Previous\_Price | Previous\_Price\_Date |
| --- | --- | --- | --- | --- |
| 101 | Laptop | $1200 | $1000 | 2023-01-01 |
| 102 | Smartphone | $800 | - | - |
| 103 | T-shirt | $20 | - | - |

| **Aspect** | **Star Schema** | **Snowflake Schema** |
| --- | --- | --- |
| Structure | Denormalized structure. | Normalized structure with multiple related tables. |
| Dimension Tables | Dimension tables are directly connected to the fact table. | Dimension tables are normalized, leading to additional levels of related tables. |
| Complexity | Simple structure, easier to understand and query. | More complex structure with additional joins, potentially impacting query performance. |
| Query Performance | Generally offers better query performance due to fewer joins. | May have slower query performance due to additional joins. |
| Maintenance | Easier to maintain and manage due to simpler structure. | More complex to maintain and manage due to additional normalization. |
| Storage Space | Requires more storage space as data is denormalized. | More efficient use of storage space due to normalization. |
| Data Integrity | May have redundancy but simplifies data integrity management. | Reduces redundancy and improves data integrity through normalization. |
| Usage | Suitable for simpler data models and reporting requirements. | Suitable for complex data models and when data integrity is critical. |

| **Aspect** | **Operational Data Source** | **Data Warehouse** |
| --- | --- | --- |
| Purpose | Primarily used for day-to-day operational transactions. | Used for analytical processing, reporting, and decision-making. |
| Type of Data | Contains current and real-time transactional data. | Stores historical, integrated, and structured data from multiple sources. |
| Schema Design | Typically designed for transactional processing, often normalized. | Designed for analytical processing, often denormalized for query performance. |
| Data Granularity | Granular, focusing on individual transactions and real-time updates. | Aggregated, focusing on summarized data for analysis. |
| Query and Reporting | Optimized for transactional queries and operational reporting. | Optimized for complex analytical queries and business intelligence reporting. |
| Time Horizon | Short-term focus, capturing current operational data. | Long-term focus, capturing historical data for trend analysis. |
| Performance and Scalability | Designed for high-speed transaction processing. | Designed for high-performance analytics and scalability. |
| Data Freshness | Provides real-time or near real-time data updates. | Provides periodic or scheduled data updates. |
| Data Usage | Used by operational staff for daily tasks and processes. | Used by business analysts, managers, and executives for strategic decision-making. |
| Backup and Recovery | Focuses on backup and recovery strategies to ensure operational continuity. | Focuses on backup and recovery strategies to protect historical data integrity. |
| Data Governance and Compliance | Often focuses on transactional data integrity and compliance. | Focuses on maintaining data consistency, quality, and compliance standards. |

| **Aspect** | **Data Mart** | **Data Warehouse** |
| --- | --- | --- |
| Purpose | Subset of a data warehouse, focusing on specific business areas or user groups. | Central repository for storing integrated data from multiple sources for enterprise-wide analytics. |
| Scope | Typically smaller in scope, focused on a specific department, function, or business unit. | Comprehensive and enterprise-wide, serving the entire organization. |
| Data Size | Smaller datasets compared to data warehouses, containing aggregated or summarized data. | Larger datasets containing detailed, integrated data from various sources. |
| Schema Design | May use a denormalized schema design optimized for specific analytical needs. | Typically uses a denormalized schema design optimized for complex analytical queries. |
| Data Granularity | Can be granular or aggregated, tailored to the specific needs of the target audience. | Supports various levels of granularity, from raw data to summarized aggregates. |
| Query and Reporting | Optimized for specific departmental or functional reporting and analysis needs. | Supports enterprise-wide reporting and analysis across multiple business areas. |
| Implementation Time | Faster implementation due to smaller scope and focused requirements. | Longer implementation time due to the complexity of integrating data from multiple sources. |
| Flexibility | Offers greater flexibility to adapt to the changing needs of specific user groups or business units. | Provides a comprehensive view of the organization's data, supporting various analytical needs. |
| Cost | Generally lower cost due to smaller scale and focused scope. | Higher cost due to the scale and complexity of integrating and managing enterprise-wide data. |
| Data Governance and Compliance | May have decentralized governance and compliance measures specific to the department or function. | Requires centralized data governance and compliance measures to ensure consistency and adherence to regulations across the organization. |
| Business Users | Primarily used by specific departments or functional teams for localized analysis and decision-making. | Used by business analysts, managers, and executives across the organization for strategic decision-making and planning. |