**Prompt : - Provide a detailed documentation on normalization consisting all normal forms, importance, practical example and summary.**

**Normalization in Database Management Systems (DBMS)**

**1. Introduction**

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller, related tables and defining relationships among them.

Normalization in Power BI is the process of structuring a data model to reduce data redundancy and improve data integrity. It's not a direct feature or a single button you press in Power BI Desktop. Instead, it's a **data modeling principle** you apply by designing your tables and relationships correctly. The primary goal of normalization is to ensure that data is stored logically, preventing inconsistencies and making it easier to analyze.

The main goal is to:

* Eliminate duplicate data (redundancy).
* Ensure data dependencies are logical.
* Improve efficiency in storage and query performance.

### **Why is Normalization Important?**

Normalization is crucial for several reasons:

* **Reduces Redundancy:** It eliminates duplicate data, which saves storage space and reduces the risk of having conflicting information. For example, instead of repeating a customer's address for every order they place, you store the address once in a **Customers** table and link to it from an **Orders** table.
* **Improves Data Integrity:** By storing each piece of information in only one place, you ensure that updates and changes are consistent across your entire data model.
* **Enhances Performance:** Normalized models are generally more efficient for analysis and reporting because they minimize the amount of data that needs to be processed. Queries on smaller, more focused tables are faster.
* **Simplifies Data Management:** It makes your data model easier to understand, maintain, and expand.

**2. Key Concepts**

* **Redundancy**: Unnecessary repetition of data (e.g., storing the same customer address in multiple rows).
* **Anomalies**: Issues that occur due to poor database design:
  + **Update anomaly**: Updating data in one place but not in others.
  + **Insert anomaly**: Inability to insert data without unnecessary information.
  + **Delete anomaly**: Deleting data unintentionally removes important details.

Normalization addresses these anomalies.

**3. Normal Forms (NF)**

Normalization is done in stages called **Normal Forms (NF)**. Each form builds upon the previous one.

### **3.1 first normal form (1nf)**

**This is the most basic level of normalization. A table is in 1NF if:**

* Each column contains a single, atomic value. There are no repeating groups of data or multiple values in a single cell.
* Each row is unique. You can identify each row with a unique key.

**Example:**

The "Products" column contains multiple values. To bring this into 1NF, you'd split the data into separate rows, with each product getting its own row.

#### **3.2 Second Normal Form (2NF)**

A table is in 2NF if it's already in 1NF and all non-key attributes are fully dependent on the entire primary key. This form primarily applies to tables with composite primary keys (keys made up of multiple columns).

**Example:** Imagine a table with **Order ID** and **Product ID** as a composite primary key. If a column like **Product Name** only depends on **Product ID** and not the **Order ID**, you would move it to a separate **Products** table.

#### **3.3 Third Normal Form (3NF)**

A table is in 3NF if it's in 2NF and has no transitive dependencies. A transitive dependency exists when a non-key attribute is dependent on another non-key attribute.

**Example:** In a table with **Order ID**, **Customer ID**, and **Customer Name**, the **Customer Name** is dependent on the **Customer ID**, which in turn is dependent on the **Order ID**. To normalize, you'd create a separate **Customers** table with **Customer ID** and **Customer Name**, and then link back to it from your **Orders** table using the **Customer ID**.

### **3.4 Boyce-Codd Normal Form (BCNF)**

Also known as **3.5NF**, BCNF is a stricter version of 3NF. A table is in **BCNF** if:

* It is in **3NF**.
* For every functional dependency A -> B, **A must be a superkey**.

BCNF addresses a specific anomaly not covered by 3NF where a table has multiple overlapping candidate keys.

**Example:** Consider a table with Student, Subject, and Professor, with the following functional dependencies:

* Student, Subject -> Professor (A student in a given subject has one professor.)
* Professor -> Subject (Each professor teaches only one subject.)

Here, (Student, Subject) is a candidate key. The table is in 3NF because there are no transitive dependencies. However, it is not in BCNF because the functional dependency Professor -> Subject has a determinant (Professor) that is not a superkey.

### **3.5 Fourth Normal Form (4NF)**

A table is in **4NF** if it meets the following criteria:

* It is in **BCNF**.
* It has no **multi-valued dependencies** (MVDs).

An MVD exists when there are two or more independent multi-valued attributes related to a single primary key.

**Example:** A table showing students, the courses they are taking, and their hobbies. A student can have multiple courses, and multiple hobbies, but the courses and hobbies are unrelated to each other.

This table is not in 4NF because (Course, Hobby) is a multi-valued dependency on Student. To normalize it, you would separate the independent attributes into their own tables.

* A table for Student Courses (Student, Course)
* A table for Student Hobbies (Student, Hobby)

### **3.6 Fifth Normal Form (5NF)**

A table is in **5NF** if it meets the following criteria:

* It is in **4NF**.
* It cannot be decomposed into smaller tables without a loss of data. This is also called being free of **join dependencies**.

5NF deals with very specific, rare scenarios where data redundancy can't be removed by decomposing the table based on functional or multi-valued dependencies alone. It ensures that the table cannot be losslessly broken down further.

### **4. Dimensional Modeling (The Power BI Approach)**

While the principles of normalization are foundational, Power BI and other business intelligence tools often leverage a **dimensional modeling** approach, most famously the **star schema**.

A **star schema** is a denormalized structure that prioritizes query performance and ease of use for reporting. It consists of:

* **Fact Tables:** These contain quantitative, transactional data (e.g., sales, orders, events). They have foreign keys that link to the dimension tables.
* **Dimension Tables:** These contain descriptive, categorical data that provides context for the facts (e.g., customers, products, dates). They are the "look-up" tables.

The dimensional model is a practical compromise: it's not fully normalized to 3NF but is structured to be highly efficient for analytical queries. Power BI's engine, VertiPaq, is highly optimized for this kind of structure, making it the **best practice** for data modeling in Power BI.

## ****5. Advantages of Normalization****

* Reduces redundancy.
* Ensures data consistency and integrity.
* Makes maintenance and updates easier.
* Logical data grouping improves query clarity.
* Saves storage space by eliminating duplication

## ****6. Disadvantages of Normalization****

* Performance overhead due to multiple table joins.
* Complex queries in highly normalized databases.
* In data warehouses, normalization may slow reporting → denormalization often preferred.

## ****7. Practical Example****

### **Unnormalized Form (UNF):**

| OrderID | CustomerName | CustomerAddress | Product | Quantity |

* Issues: Repetition of customer details if multiple orders exist.

### **After Normalization (up to 3NF):**

**Customer Table:**  
| CustomerID | CustomerName | CustomerAddress |

**Order Table:**  
| OrderID | CustomerID |

**OrderDetails Table:**  
| OrderID | Product | Quantity |

This eliminates redundancy and anomalies.

## ****8. Conclusion****

Normalization is an essential process in database design to ensure efficient data storage, reduced redundancy, and better integrity. While up to 3NF/BCNF is sufficient for most practical systems, the choice depends on a trade-off between performance and data consistency.