**What is Data Partitioning?**

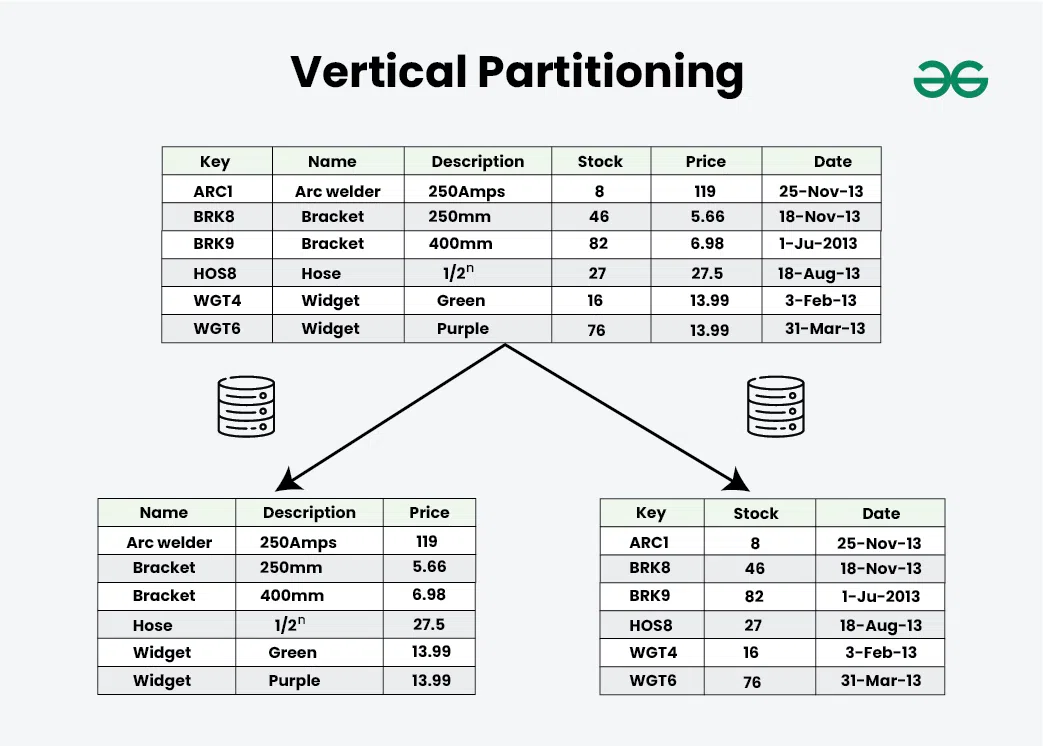
* The process of splitting a dataset into more manageable, smaller pieces in order to improve efficiency, [scalability](https://www.geeksforgeeks.org/what-is-scalability/), and performance is known as data partitioning.
* It can be accomplished by either vertical partitioning, which separates data into columns, or horizontal partitioning, which divides data into rows according to particular criteria.
* This method is especially helpful in databases, big data processing frameworks, and machine learning applications since it enables quicker query execution, simpler management of large datasets, and better resource use.

**1. Vertical Partitioning:**

* Vertical partitioning divides a database table into smaller tables containing specific groups of columns.
* It is used when different sets of columns are frequently accessed together or for managing large, wide tables with numerous columns.
* Each smaller table contains a subset of the columns, but they share a common primary key for reassembly if needed.

**Steps**:

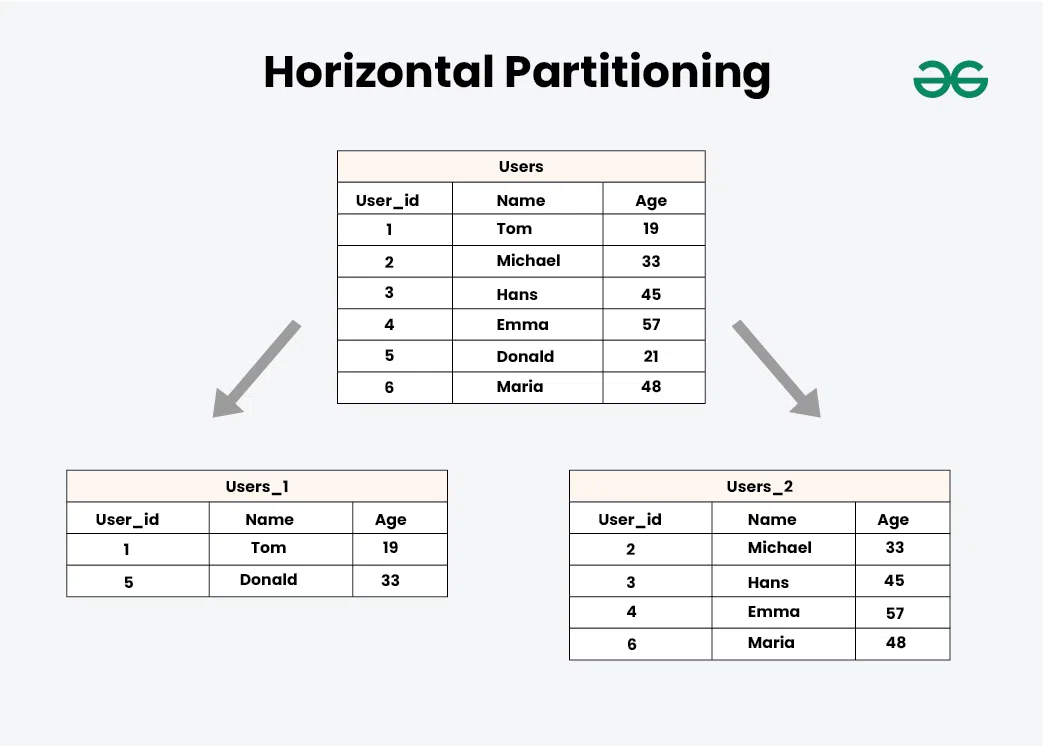
1. Identify columns that are frequently accessed together (e.g., for specific queries or reports).
2. Group these columns into separate tables based on their usage.
3. Retain a common primary key in all partitions to maintain the relationship.

**2.**[Horizontal Partitioning/Sharding](https://www.geeksforgeeks.org/database-sharding-a-system-design-concept/):

* This technique divides a dataset into subsets of rows, with each partition typically distributed across multiple servers or storage devices.
* It is commonly used in distributed systems to improve parallelism and enable load balancing.

**Steps**:

1. Determine the partitioning key (e.g., region, time, or category).
2. Create separate tables or partitions for each subset of rows.
3. Ensure that queries are routed to the correct partition based on the partitioning key.

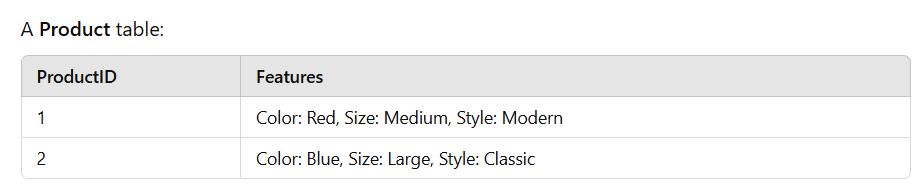


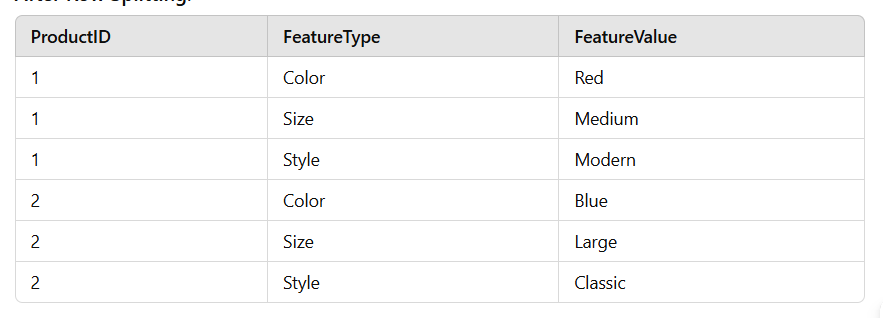
## **Row Splitting:**

* Row splitting involves dividing a single row with multiple attributes into multiple rows.
* This is useful when handling complex or hierarchical data where storing all attributes in one row is inefficient.

**Steps**:

1. Identify rows with excessive attributes or hierarchical data.
2. Split the data into multiple rows with a logical connection to the original row.
3. Use an identifier to relate split rows back to the original entity.

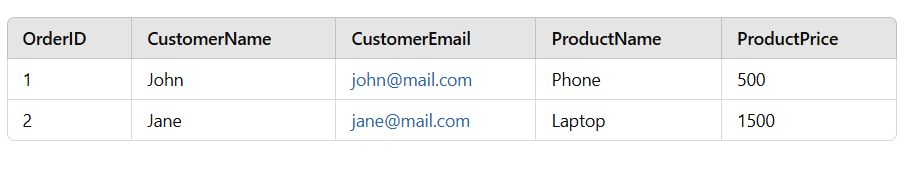




**Normalization:** Normalization organizes data into smaller, related tables to reduce redundancy and dependency. It follows a series of steps (normal forms) to achieve an efficient database design.

**Steps**:

1. Identify data redundancy and repeated values.
2. Create separate tables for each logical grouping of related data.
3. Use foreign keys to maintain relationships between tables.



**After Normalization:**

