CMT 428 Module 1:

**Module 1: Introduction to Big Data and Hadoop**

| **Topic** | **Subtopic** | **Key Concepts** | **Practical Example** |
|---|---|---|---|
| **1. Introduction to Big Data and Hadoop** | **- What is Big Data?** | - Defines Big Data and its 3 V's (Volume, Velocity, Variety) + 2 more V's (Veracity, Value). <br> - Explains the challenges and opportunities Big Data presents. | - **Example:** A telecommunications company collects massive amounts of call detail records (CDRs) daily (**Volume**), requiring real-time processing for fraud detection (**Velocity**) and analysis of diverse data types like text messages and location data (**Variety**). Ensuring data accuracy and reliability (**Veracity**) is crucial for extracting meaningful insights to improve customer service and optimize network performance (**Value**). |
| | **- Hadoop Ecosystem** | - Introduces Hadoop as an open-source framework for distributed storage and processing of Big Data. <br>- Overview of core components: <br> * HDFS (Hadoop Distributed File System) <br> * MapReduce (programming model) <br> * YARN (resource management) <br> * Other tools (Hive, Pig, HBase, etc.) | - **Example:** An e-commerce company uses Hadoop to process large volumes of customer data, clickstream logs, and product information. HDFS stores this data, MapReduce performs analysis, and YARN manages resources for efficient processing, enabling personalized product recommendations and targeted marketing campaigns. |
| | **- HDFS in Depth** | - Explains HDFS architecture: <br> * NameNode and DataNodes <br> * Data replication and fault tolerance <br> * Block concept and data distribution | - **Example:** When a user uploads a large file to HDFS, it is divided into blocks and distributed across multiple DataNodes. If one DataNode fails, the file remains accessible due to replication on other nodes, ensuring high availability and durability. |
| | **- MapReduce Paradigm** | - Describes the MapReduce programming model: <br> * Mapper and Reducer functions <br> * Data partitioning and shuffling | - **Example:** To count word frequencies in a massive text corpus, MapReduce assigns each mapper a portion of the text. Mappers count word occurrences in |

| | | | |
|---|---|---|---|
| | | <br> * Distributed processing and aggregation | their portion, and reducers aggregate these counts to provide a final word frequency list, showcasing parallel processing power. |