Lab Work 1

The objective of this practical work is to **design, build, and manage a large-scale relational database** using an open dataset from **Kaggle**. You will import data, establish relationships between tables, and execute advanced SQL queries.

1. Dataset Selection

- Choose a **large and structured dataset** from Kaggle.com that can be organized into multiple related tables.
- Examples of suitable datasets:
  - E-commerce transactions
  - Movie ratings and reviews
  - Financial transactions
  - Healthcare records
  - Social media interactions

2. Database Creation & Data Import

- Use **PostgreSQL, MySQL, or SQLite** to create your database.
- Write **SQL scripts** to define tables with appropriate data types, keys, and constraints.
- Import data from **CSV files** into the corresponding tables.

3. Data Analysis

Execute SQL queries to analyze the data, including:

- Aggregations: `SUM`, `AVG`, `COUNT`, `MAX`, `MIN`.
- Implement **indexing** on large tables to improve query performance.

4. Web Interface Development

- Design a **web-based interface** using **HTML, CSS, and JavaScript** to interact with the database.
- Implement basic **CRUD operations** (Create, Read, Update, Delete) to allow users to manage records.

Lab Work 2

Intelligent Query Processing

The goal of this practical work is to implement intelligent query processing techniques to enhance user interactions with databases. You will explore:

1. **Levenshtein Distance** for **auto-correction** of misspelled queries.
2. **Autocomplete using Trees** to suggest relevant queries based on user input.
3. **BK-Tree (Burkhard-Keller Tree)** for efficient **fuzzy searching** in large datasets.

Instructions :

1. Create SQL Database with big datasets
2. Create a **web-based interface** using **HTML, CSS, and JavaScript** to interact with the SQL query .
3. Use the Levenshtein algorithm to detect and correct
4. Implement Autocomplete using a Trie (Prefix Tree). Example: If the user types "SEL", the system suggests "SELECT", "SELF", etc.
5. Implement a **BK-Tree** to efficiently handle **approximate matching** in large datasets.

   This structure is useful for **quickly finding the closest matches** to a given input.

   Example: If searching for "Biksra", the system finds similar names like "Biskra",.

Lab Work 3

Database Indexing and TF-IDF for Efficient Search

The goal of this practical work is to explore **database indexing** techniques to optimize query performance and implement **TF-IDF (Term Frequency - Inverse Document Frequency)** for text search relevance. You will:

- **Create and use indexes** to speed up SQL queries.
- **Implement TF-IDF** to rank search results based on relevance.
- **Compare performance** between indexed and non-indexed queries.

1. Create a Database and Load Dataset **(e**.g., articles, product reviews, or customer transactions).
2. Create Indexes for Faster Queries
3. Calculate Term Frequency (TF) : Compute the frequency of a word in a document
4. Calculate Inverse Document Frequency (IDF) : Compute the importance of a word across all documents
5. Implement a Query using TF-IDF Ranking
6. Compare indexed vs. non-indexed queries and measure execution time.
7. Display ranked search results in Interface

Lab Work 4

Recommendation System & Product Comparison

The aim of this practical work is to build a recommendation system using TF-IDF (Term Frequency - Inverse Document Frequency) to compare product descriptions and suggest similar items. You will:

1. Extract textual features from product descriptions.
2. Compute TF-IDF scores to measure word importance.
3. Use cosine similarity to compare and recommend similar products.
4. Evaluate the effectiveness of TF-IDF for recommendations.

Lab Work 5

Big Data Processing with Hadoop

The objective of this practical work is to introduce students to Hadoop, a powerful framework for distributed storage and processing of large datasets. Students will set up a Hadoop environment, process data using HDFS (Hadoop Distributed File System), and perform MapReduce operations to analyze a dataset.

1. Download and install **Hadoop** (Single-node), Configure core-site.xml, hdfs-site.xml, and mapred-site.xml.
2. Download a dataset (e.g., a Kaggle dataset like movie reviews, stock market data, or web logs).
3. Word Count Example in Java : Implement a MapReduce job that counts word occurrences in a dataset.
4. Download and process a large dataset (e.g., customer reviews, social media posts).
    a. Use HDFS to store the dataset.
    b. Implement a MapReduce job to analyze trends (e.g., most common words, user activity).

Lab Work 6

NoSQL Database Management with MongoDB

The objective of this practical work is to introduce students to **MongoDB**, a NoSQL database used for handling large amounts of unstructured and semi-structured data. Students will learn how to:

- and Install and configure MongoDB
- Create and manage collections and documents
- Perform CRUD (Create, Read, Update, Delete) operations
- Execute complex queries using MongoDB's aggregation framework

1. Install MongoDB on your system (MongoDB Download)
2. Create a Database
3. Manage Collections & Documents :
    i. Insert Data into a Collection
    ii. Retrieve Data
    iii. Delete Documents
    iv. Update Documents
4. Integrating MongoDB with a Web Application

Lab Work 7
Big Data Storage and Processing with Cassandra

The goal of this practical work is to introduce students to **Cassandra**, a distributed, scalable, and NoSQL database designed for handling large amounts of data across multiple nodes with high availability. Students will learn how to :

- Set up an Cassandra environment
- Create and manage tables
- Perform CRUD (Create, Read, Update, Delete) operations
- Execute advanced queries using CQL (Cassandra Query Language) and Java API

Instructions

1. Download and **Install Cassandra (Standalone)**
2. Create a Table in Cassandra Shell
    a. Insert Data into the Table
    b. Retrieve Data from the Table
    c. Update Data
    d. Delete Data

3. Set Up a Java Project with Cassandra