# Final Assignment: Database Management Tools in Python

**Total Marks: 50 points**
**Submission Deadline:** last day of exam period
**Format:** Individual

---

## Overview

You must complete both **Part I (Core Programming and Algorithms)** and **Part II (Data Ingestion and Analytics)**.

---

## 1. Part I: Data Structures & Algorithms (15 points)

### A. Node-Based Implementations

1. **Linked Nodes:**
   - Implement a **Doubly Linked Node** class.
   - Create a **Stack** class using these nodes.
   - Use two stacks to implement a **Queue** with `enqueue`, `dequeue`, and `peek` methods.
2. **Hash Table with Binary Search Trees (BST):**
   - Implement a Hash Table where each bucket uses a BST to handle collisions.
   - Demonstrate insert, search, and delete operations.
3. **Graph Implementation:**
   - Implement a Graph using a Node class (adjacency list representation).
   - Include traversal algorithms:
     - Depth-First Search (DFS)
     - Breadth-First Search (BFS)

---

## 2. Part II: Databases, Ingestion & Analytics (25 points)

This part of the assignment is designed as a **complete, end-to-end workflow** that mirrors how data is handled in real organizations. Each stage in the pipeline influences the next, and your decisions early in the process will shape what is possible later. Your report should therefore not treat the sections as isolated tasks, but as *interdependent components of a coherent system.* You are expected to explain these dependencies clearly throughout your submission.

### 2.1. Data & Database Engineering

You will begin this task assuming that your role is that of a data engineer. You will therefore design the data model and ingestion approach such that you would be able to store data for analytics tasks. **No codes are required for this section**.

#### 2.1.1 Choosing a Dataset

The first step in this assignment is to select a dataset that is rich enough to support end-to-end work across database selection, ingestion, analytics, visualization, and (optional) machine learning. Your choice of dataset will strongly influence all later decisions in your workflow, including data modeling, storage technology, ingestion pipeline, and analytical methods.

When choosing a dataset, consider the following criteria:

- **Real-World Source:** Prefer datasets that originate from a realistic process (e.g., sensors, user activity, transactions, health measurements, geographic or social systems).
- **Structure:** Identify whether the data is structured, semi-structured, or unstructured, as this determines your database choice.
- **Volume & Velocity:** Ensure that the dataset contains enough records to demonstrate ingestion, querying, and analysis. For time-series or IoT-like tasks, datasets with timestamps are ideal.
- **Relationships:** Choose a dataset that contains interesting relationships (e.g., between users and items, devices and timestamps, nodes and edges), especially if you plan to justify graph or document databases.
- **Analytical Value:** Select data that allows meaningful EDA, trend identification, visualizations, and optional machine learning.

You may use any open dataset, including those listed below.

**Recommended Free Dataset Sources**

- **Kaggle Open Datasets**
  Contains thousands of datasets across all domains (CSV, JSON, images, etc.). Some example dataset from here :
  - Bike store rental
  - IoT-SmartHome
- **Google Dataset Search**
  A search engine for public datasets from research institutions, governments, and open data portals.
- **data.gov (US Government)**
  Large structured datasets (transportation, health, energy, finance), excellent for SQL, time-series, or NoSQL wide-column storage.
- **Open Data Portal Europe**
  EU open datasets across scientific, environmental, social, and demographic domains.
- **Awesome Public Datasets GitHub**
  Curated list of domain-specific datasets (healthcare, sports, education, social networks, etc.).

Choose a dataset appropriate for the level of complexity you wish to demonstrate. Your later discussion should highlight *how the nature of your chosen data directly influenced* your decisions regarding data modeling, ingestion pipeline design, analytics, and visualization.

### 2.1.2 Understanding the Data: Real-World Origins and Characteristics

Begin by describing the **real-world system** that could potentially generates such data. This step is foundational, because:

- The **origin** of the data determines its expected structure and reliability.
- The **frequency** of generation (velocity) dictates ingestion strategies.
- The **volume** influences scalability requirements.
- The **type and richness of relationships** between data elements influences the choice of database technology.

For example, describing the data as "sensor readings produced every second by IoT devices" implies:

- high ingestion rate,
- time-series patterns,
- suitability for wide-column or time-series databases,
- preference for sequential write-optimized storage.

If instead the data came from social platforms, you would likely move towards graph-first storage.

### 2.1.3 Database Selection

The structure you identified in the previous step dictates your database choice. This task is not arbitrary:

- If your data is **strictly structured** with well-defined rows and columns, a relational database may be ideal.
- If your data is **hierarchical or nested** (e.g. JSON-like objects), a document database aligns better with the inherent shape of the data.
- If your data forms **networks of relationships**, a graph database is naturally suited.
- If your data is **high-velocity and time-indexed**, wide-column stores or time-series DBs become a logical choice.

Here, you must explain how the data's real-world nature (Step 1) logically leads to your selected storage model. Your reasoning needs to close the loop between the *source*, the *structure*, and the *storage*. Your reasoning should also show clear explanations on:

- The **shape of your schema or documents**.
- The **choice of row key or identifier** (for wide-column or NoSQL stores).
- The **relationships modeled** in your graph database.
- The **normalization decisions** in SQL.

### 2.1.4 Designing the Ingestion Pipeline: Connecting the Real World to Your Database

Once the database type has been selected, you need to design a pipeline (ETL or ELT) to move the raw data into storage. This stage is *dependent on your earlier decisions*:

- The **volume and velocity** determine whether ingestion is batch-based or streaming.

- The **structure** identified earlier determines necessary transformations before storage.
- The **database technology** influences how loading is performed (SQL inserts, document writes, graph node creation, etc.).

For example, if you selected a document database, you will design a pipeline that preserves nested structures. If you chose a relational system, your pipeline must normalize data or map it into a schema. Explain how each stage of your pipeline (Extract, Load, Transform) is shaped by the realities of the data source and by the constraints of your chosen database.

## 2.2. Analytics & Visualization

Once your data model and ingestion approach are designed, you now "switch roles" from engineer to analyst. However, the transition is not cleanly separated: earlier engineering decisions constrain what analysis is possible here. **This section requires code**. The use libraries to interact with database you chosed earlier should be shown clearly here. Further, use of analytics libraries such as Numpy, Scipy and matplotlib must be shown in this section.

### 2.2.1 Writing Insertion, Retrieval, and Analytical Queries

In this section, you create example queries *as they would be executed on your chosen database*. You should present the codes, that allow you to INSERT and RETRIEVE data from the database. The code should reflect : choice of python modlue used to interect with the choice of database and few example queries. Your query design should illustrate that the earlier database decisions now enable efficient retrieval for analysis.

### 2.2.2 1. Data Loading & Cleaning in Python

You can use any preferred way to read your data in python (depending on what is the file type you got). Include codes needed for reading data, preforming merging of data from different files. If you think any cleaning or preprocessing is required, provide code with discussions on your choice. As you load the dataset into Python for analytical work, reflect on and write a short discussion on:

- How the **storage model** you choose will simplify or complicate obtaining useful data.
- Whether your earlier transformations in the pipeline will help reduce Python-side preprocessing.

### 2.2.3 Exploratory Data Analysis (EDA)

The EDA you perform here is influenced by:

- The structure you identified earlier (time-series data leads to trend analysis, graph data to connectivity analysis).
- The transformations applied in the ingestion pipeline.
- The choice of database, which may have encouraged certain data shapes or denormalization patterns.

Prepare a discussion on what methods you use to extract what information from data. Explain how your engineering decisions prepared the data for insights. Include code to compute the analytics. Include appropriate visualizations in your EDA. Discuss these connections and patterns seen in visualizations explicitly. Include a short reflection discussion on how your EDA and visualizations are influenced by:

- underlying assumptions about time, categories, or relationships.
- Structure of the data.

## 2.3. Machine Learning Component (Bonus)

The machine learning step is the culmination of the entire workflow. Every previous engineering and analytics decision influences what model is appropriate and how well it performs.

### 2.3.1 Problem Definition

Your ML problem must be aligned with:

- the structure of the data,
- the relationships modeled in the database,
- the transformations applied earlier,
- the types of insights revealed in EDA.

### 2.3.2 Feature engineering

Feature engineering is deeply dependent on structure of data.

- A time-series structure (identified earlier) leads to lag features or rolling windows.
- Nested JSON (from document stores) may require flattening for ML input.

- Graph data may produce node embeddings or centrality metrics.
- Normalized vs. denormalized SQL schemas affect join operations.

Although these might not be aapplicable to your specific dataset, include aa discussion on how the engineering and storage choices influence the features you were able to derive. If the dataset of your choice allows you to extract features, discuss the choice of feature and include code.

#### 2.3.3 Model Choice & Justification

Your choice of regression or classification must be justified using:

- available features (shaped by earlier cleaning and storage),
- the natural prediction opportunities in your dataset,
- database schema decisions that influenced feature accessibility.

#### 2.3.4 Model Evaluation

Finally, evaluate your model and discuss how its performance is linked back to:

- data quality from the ingestion pipeline,
- the richness (or limitations) of your chosen storage model,
- the feature engineering decisions,
- the characteristics of the real-world source.

---

## 3. Submission Requirements

- The final submission must be in a single **Jupyter Notebook** file (`.ipynb`).
- The notebook should include both:
  - **Code cells** for implementation, data loading, analysis, and visualization.
  - **Markdown cells** for explanations, reasoning, and interpretation of results.
- Each section of the assignment (Data Structures, Database, Analytics, and Machine Learning) should be clearly separated with appropriate Markdown headings.
- All visual outputs (e.g., plots, graphs, tables) should be displayed directly within the notebook.

- The notebook must be self-contained and executable from start to finish without requiring external manual steps.

---

## 4. Grading Breakdown (50 points)

| **Component** | **Points** | **Description** |
|---|---|---|
| Linked Nodes, Stack, Queue | 5 | Functional and documented code |
| Hash Table + BST | 5 | Correct implementation and usage |
| Graph + Traversal Algorithms | 5 | Efficient and correct traversal |
| Database & Queries | 15 | Schema, data, and analytical queries |
| Visualization | 20 | Quality of analysis, plots, and discussion |
| ML Model (Bonus) | 5 | |
| **Total** | **55 pts** | |

---

## Notes

- Submit via GitHub.
- Late submissions are not accepted.
- You may discuss concepts with peers, but code and analysis must be original.