PROJECT TITLE:

Indexing, Modifying, and Searching on all data structures

Name: Sadneya Sadanand Samant
Submitted To : Agratas EduTech
Date Of Submission: 22/01/25

ABSTRACT

This report demonstrates a series of key operations—indexing, modifying, searching, and sorting—on a manually created dataset consisting of diverse data types. These operations are foundational in data science and are commonly used to preprocess, analyze, and efficiently retrieve meaningful insights from data. The dataset includes a range of data types such as integers, strings, dates, lists, and boolean values, each representing different kinds of information that are often encountered in real-world applications. By performing these operations, we can transform raw data into a structured format, making it suitable for deeper analysis and interpretation. This report explores how these operations facilitate tasks such as cleaning, organizing, and querying data, which are essential for deriving valuable insights in data-driven decision-making.

Table of Contents

INTRODUCTION

This report focuses on the essential operations of indexing, modifying, searching, and sorting on a dataset. These operations demonstrate the application of foundational data structures in Python to process and analyze data efficiently.

1.1. Objectives

1. To showcase the practical application of data structures in handling diverse datasets.
2. To perform essential operations on structured data for real-world applicability.
3. To highlight efficient data retrieval, manipulation, and organization methods.

1.2 Data Structures Overview

We have created a sample dataset consisting of records with the following fields:

1. **id:** Unique identifier for each record (integer).
2. **name:** Name of the person (string).
3. **age:** Age of the person (integer).
4. **join_date:** Date when the person joined (date).
5. **hobbies:** List of hobbies (list of strings).
6. **is_active:** Whether the person is currently active (boolean).

METHODOLOGY

The following steps were implemented:

1. **Dataset Creation**: A sample dataset was manually crafted using dictionaries and lists.
2. **Operations**: Key operations (indexing, modifying, searching, and sorting) were applied to transform and analyze the data.
3. **Coding Practices**: Python features like lambda functions, list comprehensions, and dictionary indexing were used for concise implementation.

IMPLEMENTATION

---

1. **Create dataset:**

```
import datetime

# Sample dataset with diverse data types
dataset = [
    {'id': 1, 'name': 'Alice', 'age': 25, 'join_date': datetime.date(2020, 5, 15), 'hobbies':
['reading', 'hiking'], 'is_active': True},
    {'id': 2, 'name': 'Bob', 'age': 30, 'join_date': datetime.date(2021, 8, 10), 'hobbies':
['cycling', 'swimming'], 'is_active': False},
    {'id': 3, 'name': 'Charlie', 'age': 28, 'join_date': datetime.date(2022, 3, 22), 'hobbies':
['photography', 'traveling'], 'is_active': True},
    {'id': 4, 'name': 'David', 'age': 35, 'join_date': datetime.date(2019, 11, 5), 'hobbies':
['gaming', 'music'], 'is_active': False},
    {'id': 5, 'name': 'Eve', 'age': 22, 'join_date': datetime.date(2023, 7, 19), 'hobbies':
['cooking', 'dancing'], 'is_active': True},
    {'id': 6, 'name': 'Fiona', 'age': 29, 'join_date': datetime.date(2022, 6, 10), 'hobbies':
['gardening', 'painting'], 'is_active': True}
]
```

2. **Indexing Operation**

Indexing helps to retrieve specific records efficiently using unique keys.

```
# Create an index based on 'id'
indexed_data = {entry['id']: entry for entry in dataset}

# Access a record by its 'id'
record = indexed_data[3]
print("Record with ID 3:", record)
```

```
Record with ID 3: {'id': 3, 'name': 'Charlie', 'age': 28, 'join_date': datetime.date(2022, 3, 22), 'hobbies': ['photogr
aphy', 'traveling'], 'is_active': True}
```

3. **Modifying Operation**

Modification updates specific fields of a record.

Example: Update Age

```
# Update Bob's age (ID 2)
indexed_data[2]['age'] = 31
print("Updated Bob's record:", indexed_data[2])
```

```
Updated Bob's record: {'id': 2, 'name': 'Bob', 'age': 31, 'join_date': datetime.date(2021, 8, 10), 'hobbies': ['cycli
ng', 'swimming'], 'is_active': False}
```

4. **Searching Operation**

Search operations help retrieve records matching specific criteria.

Example: Find Active Users

```
# Search for active users
active_users = [entry for entry in dataset if entry['is_active']]
print("Active Users:", active_users)
```

```
Active Users: [{'id': 1, 'name': 'Alice', 'age': 25, 'join_date': datetime.date(2020, 5, 15), 'hobbies': ['reading',
'hiking'], 'is_active': True}, {'id': 3, 'name': 'Charlie', 'age': 28, 'join_date': datetime.date(2022, 3, 22), 'hobb
ies': ['photography', 'traveling'], 'is_active': True}, {'id': 5, 'name': 'Eve', 'age': 22, 'join_date': datetime.dat
e(2023, 7, 19), 'hobbies': ['cooking', 'dancing'], 'is_active': True}, {'id': 6, 'name': 'Fiona', 'age': 29, 'join_da
te': datetime.date(2022, 6, 10), 'hobbies': ['gardening', 'painting'], 'is_active': True}]
```

5. **Sorting Operation**
   Sorting arranges records in a specified order based on a chosen field.

   Example: Sort by Age

```
# Sort dataset by 'age'
sorted_by_age = sorted(dataset, key=lambda x: x['age'])
print("Sorted by Age:", sorted_by_age)
```

```
Sorted by Age: [{'id': 5, 'name': 'Eve', 'age': 22, 'join_date': datetime.date(2023, 7, 19), 'hobbies': ['cooking', '
dancing'], 'is_active': True}, {'id': 1, 'name': 'Alice', 'age': 25, 'join_date': datetime.date(2020, 5, 15), 'hobbie
s': ['reading', 'hiking'], 'is_active': True}, {'id': 3, 'name': 'Charlie', 'age': 28, 'join_date': datetime.date(202
2, 3, 22), 'hobbies': ['photography', 'traveling'], 'is_active': True}, {'id': 6, 'name': 'Fiona', 'age': 29, 'join_d
ate': datetime.date(2022, 6, 10), 'hobbies': ['gardening', 'painting'], 'is_active': True}, {'id': 2, 'name': 'Bob',
'age': 30, 'join_date': datetime.date(2021, 8, 10), 'hobbies': ['cycling', 'swimming'], 'is_active': False}, {'id': 4
, 'name': 'David', 'age': 35, 'join_date': datetime.date(2019, 11, 5), 'hobbies': ['gaming', 'music'], 'is_active': F
alse}]
```

CONCLUSION

The project underscores the significance of indexing, modifying, searching, and sorting in data preprocessing and analysis. Python's robust data structures and features make these operations scalable and adaptable for real-world applications.

REFERENCES

1. Python Official Documentation: https://docs.python.org
2. Agratas EduTech Curriculum
3. Project-Based Learning Resources