SmartShelf

Library Management System

Computer Science Project 2024-25

INDIAN SCHOOL AL MAABELA

LEAD KINDLY LIGHT

INDIAN SCHOOL AL MAABELA

(ISO 9001:2015 CERTIFIED INSTITUTION)

COMPUTER SCIENCE PROJECT

2024-2025

SMARTSHELF

LIBRARY MANAGEMENT SYSTEM

Submitted by	:	Rida Riyaz
Class	:	12-A
Registration No	:	

INDIAN SCHOOL AL MAABELA

CERTIFICATE

*It is hereby certified that **Miss Rida Riyaz** of class **XII A** has carried out the necessary project work in **Computer Science (083)** as per the syllabus prescribed by the **C**entral **B**oard of **S**econdary **E**ducation, New Delhi, for the year 2024-2025.*

Internal Examiner ***External Examiner***

Date ***Examiner No:***

School Seal ***Principal***

ACKNOWLEDGEMENT

On the successful completion of my ***Computer Science*** *project, first of all I thank the almighty* ***GOD*** *who gave me the strength and courage to accomplish this task.*

I extend my sincere thanks to this esteemed institution, ***Indian school Al Maabela****, for nurturing me all these years and molding me into a good academic and culture citizen.*

I place on record my heartfelt gratitude to our beloved Principal ***Mr. P Parveen Kumar*** *and Vice Principal* ***Mrs. Savita Saluja*** *for creating a conducive learning environment and being an inspiration, which motivated me to fulfill my aspirations and achieve my goals.*

I am deeply indebted to my Computer Science project guide ***Mrs. Deepa C M*** *under whose vital suggestions and patient guidance I could complete this project most successfully.*

On a moral personal note, my deepest appreciation to my loving ***parents*** *and friends, who put their trust in me and provided me with unceasing encouragement and support.*

November 2024

INDEX

HISTORY OF PYTHON

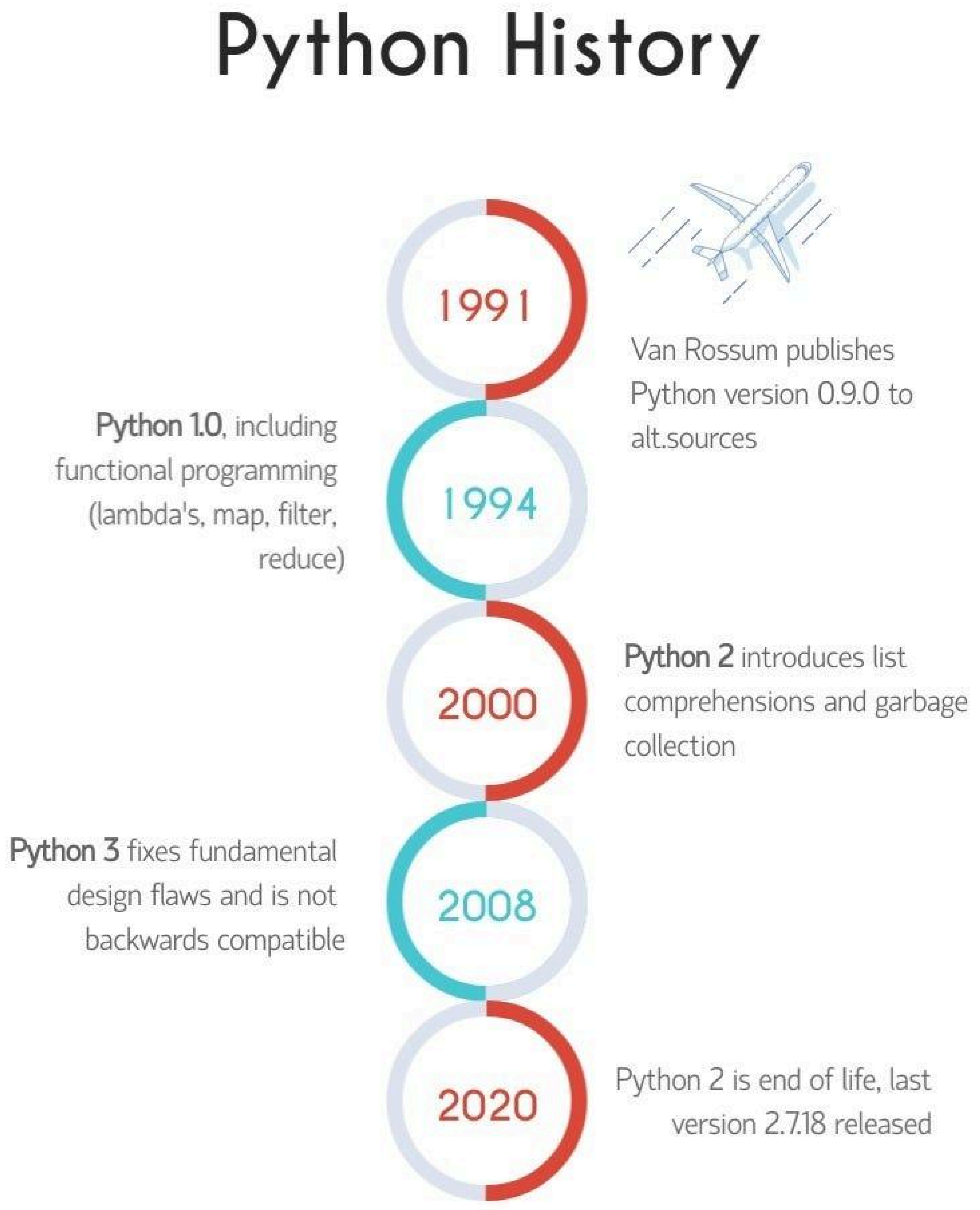
Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility.
Python was created by *Guido van Rossum*, a Dutch programmer in the late 1980's. He is the Python programming language creator, for which he was the "benevolent dictator for life" (BDFL) until he stepped down from the position on 12 July 2018. This was created as a successor to the ABC programming language, which was designed for teaching and prototyping but it had limitations. So he created this with more focus on readability and simplicity and released it on February 20, 1991. The name Python comes from an old BBC television comedy sketch series called *Monty Python's Flying Circus* to reflect its fun and approachable nature.

When it was released, it used fewer codes to express the concepts, compared with Java, C++ &amp; C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions
The first official version of Python (Python 1.0) was released in January 1994, introducing key features like modules, exception handling, core data types (list, str, dict, etc). The language gained recognition as a powerful scripting tool, used in both academia and industry.

The second version of Python(Python 2.0) was released on October 16, 2000, and introduced many new features like List comprehensions, Garbage collection and full Unicode support, becoming widely adopted for nearly two decades. However, Python 2 had limitations, especially in handling Unicode and other modern programming needs which led to the development of Python 3.
Python 3 (Python 3.0)was first released in 2008 introducing significant improvements but was not backward-compatible with Python 2. It was aimed to address the fundamental flaws in Python 2.x while cleaning up the

language to make it more consistent and efficient. Python 2.7, released in 2010, was the last release in the Python 2.x series, and official support for Python 2 ended on January 1, 2020. Transitioning from Python 2 to Python 3 took several years due to compatibility issues, but Python 3 has since then become the standard.

Python versions 3.6-3.9 brought significant performance improvements, type annotations, easier string formatting and other enhancements. Python 3.10 was released in 2021, introduced pattern matching and Python continues to evolve with the addition of new features and optimizations in subsequent versions.


Python History
1991
Van Rossum publishes Python version 0.9.0 to alt.sources
1994
Python 1.0, including functional programming (lambda's, map, filter, reduce)
2000
Python 2 introduces list comprehensions and garbage collection
2008
Python 3 fixes fundamental design flaws and is not backwards compatible
2020
Python 2 is end of life, last version 2.7.18 released

Over the years, Python has gained immense popularity due to its simplicity, versatility, and wide usage across various fields, including web development, data science, automation, and artificial intelligence. Its vibrant open-source community has contributed to its continued evolution. Python’s growth from a simple scripting language to one of the most widely used languages in the world is a testament to its adaptability, ease of use, and vibrant community. It continues to evolve and impact numerous industries.

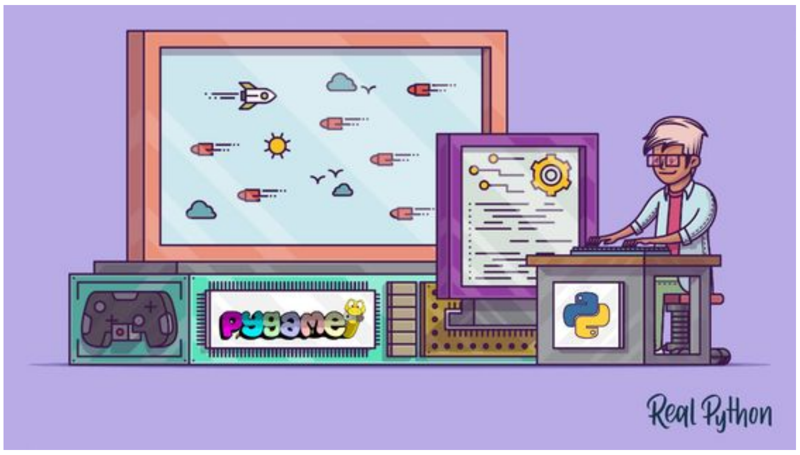
APPLICATIONS OF PYTHON

Python is an open-source and free programming language which is versatile and used in a wide range of applications across different domains.
Here are some key areas where python is used:
Web development: Python has powerful frameworks like django,Flas and pyramid that simplifies the process of building robust web applications.also python offers a vast number of libraries which help in making good visualisation,convenience in development,enhanced security and fast development process.

Machine Learning and Artificial Intelligence: ML and AI are one of the most developing fields of technology right now. Python with its inbuilt libraries and tools helps in the development of AI and ML algorithms.it offers simple,concise and readable code.these help the developers to write complex algorithms and provide a versatile flow.some of the inbuilt libraries for AI and ML algorithms are Numpy for complex data analysis, Keras for Machine learning ,SciPy for technical computing ,Seaborn for data visualisation.

Data Science: Data science involves a range of activities including data collection, sorting, analysis, and visualization. Python excels in handling statistical and intricate mathematical computations, thanks to its robust libraries. These libraries offer significant convenience for data science practitioners. Notable examples include TensorFlow, Pandas, and Scikit-learn, which form a comprehensive ecosystem for refining data models, preprocessing data, and conducting sophisticated analyses. These tools streamline the data science workflow, making, complex tasks are more manageable and efficient.

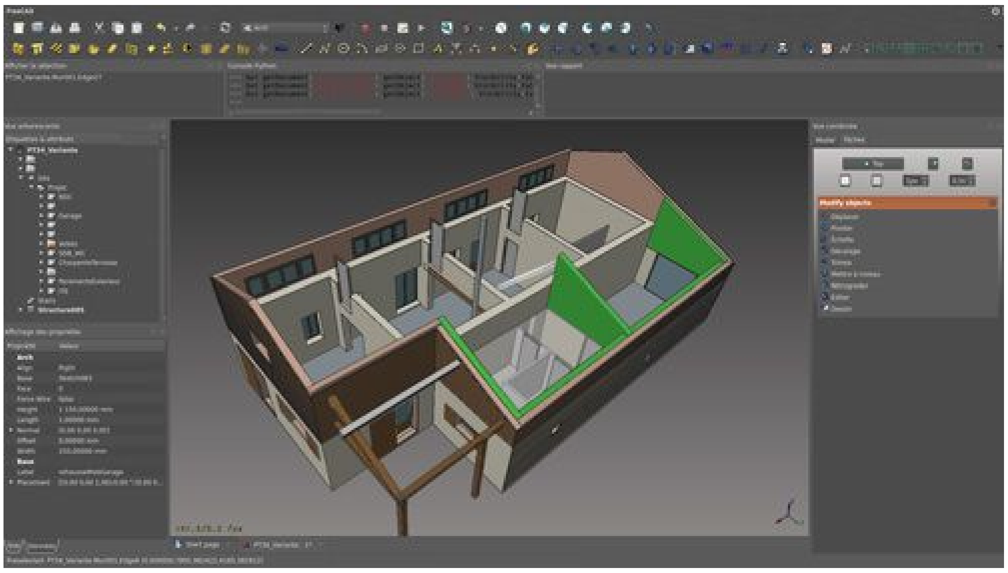
Game Development: As the gaming industry continues to expand rapidly, Python has emerged as a standout choice for game development. Renowned games such as Pirates of the Caribbean, Bridge Commander, and Battlefield 2 leverage Python for various functionalities and enhancements. With popular 2D and 3D game development libraries like Pygame, Panda3D, and Cocos2D, Python simplifies and streamlines the game creation process, making it more accessible and efficient.



Audio and Visual Applications: Python excels in audio and video applications, offering a wealth of tools and libraries to achieve impressive results. Renowned platforms like Netflix, Spotify, and YouTube utilize Python for various aspects of their operations. Libraries such as Dejavu, Pyo, Mingus, SciPy, and OpenCV are instrumental in handling audio processing, image manipulation, and multimedia tasks, making Python a powerful choice for developing and managing audio and video applications.

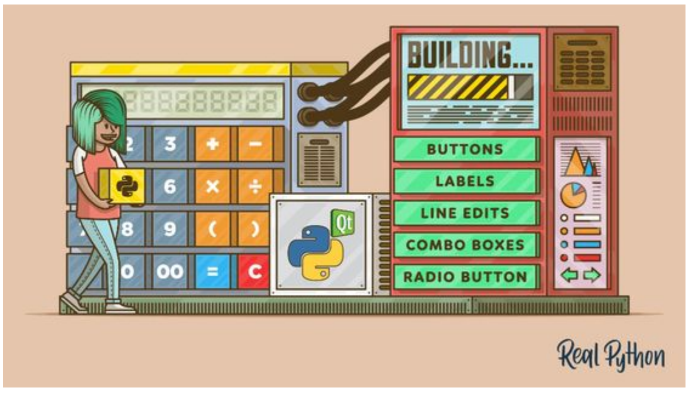
Software Development: Python is an excellent choice for software development, with major applications like Google, Netflix, and Reddit using it. The language offers several remarkable features, including platform independence, which ensures it works seamlessly across various operating systems. Python's extensive libraries and frameworks simplify the development process, while its clean syntax enhances code reusability and readability. Additionally, Python's high compatibility with other technologies and systems makes it a versatile tool. It is also well-equipped to handle rapidly evolving fields like machine learning and artificial intelligence, thanks to its robust ecosystem. These attributes collectively make Python a popular and effective option for software development.

CAD Applications: CAD, or computer-aided design, involves the digital creation of 3D and 2D models. This technology has replaced traditional manual drafting and is widely used by architects, product designers, and construction managers to create highly precise designs. Python supports a range of powerful CAD applications, including Blender, FreeCAD, and OpenCascade. These tools offer advanced features such as technical drawing, dynamic system development, and capabilities for recording, file export, and import, making Python a valuable asset in the realm of digital design.



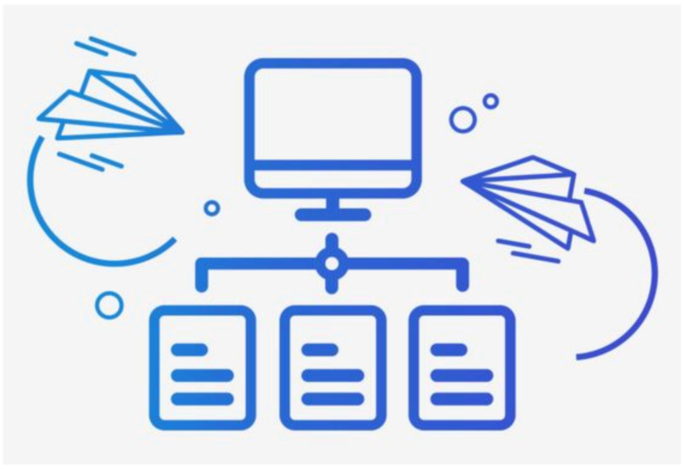
Business Applications: Python offers robust security and scalability features that make it an excellent choice for developing high-performance business applications. Its extensive range of libraries and tools enhances its capabilities, including options like Odoo, which provides automated solutions for various business processes, streamlining operations and improving efficiency. Another notable tool is Tryton, an easy-to-use open-source business software that integrates essential features such as financial accounting, sales, CRM, purchasing, and shipping. These features collectively make Python particularly well-suited for creating effective and scalable business applications.

Desktop GUI: Python is an interactive programming language that simplifies the creation of graphical user interfaces (GUIs) with ease and efficiency. It boasts a wide array of built-in tools and libraries for this purpose, including PyQt, Kivy, and wxWidgets. These libraries enable developers to build fully functional and secure desktop GUIs effectively, offering a range of features that streamline the development process and enhance the overall user experience.



Web scraping application: Web scraping is an automated process used to extra information from website in an easier and faster way.the information is used by researchers in an easier and faster way.The information is used by researcher,organisation, and analysts for a wide variety of tasks.Python has a wide range of feature that make it suitable for web scraping some of them are:

- A concise syntax enhances readability and saves you time.
- A wide range of libraries and tools like pandas,matplotlib, and Selenium makes the web scraping process easy and effect
- Easy to use and understand



Conclusion: Python is a concise yet exceptionally powerful language, which is why its applications are rapidly gaining traction. It stands at the forefront of groundbreaking technologies such as AI, automation, and machine learning. Additionally, Python plays a crucial role in key areas like data analysis and data visualisation, making it a central tool in these dynamic fields.

DATABASE-BASIC CONCEPTS

What is a DBMS?

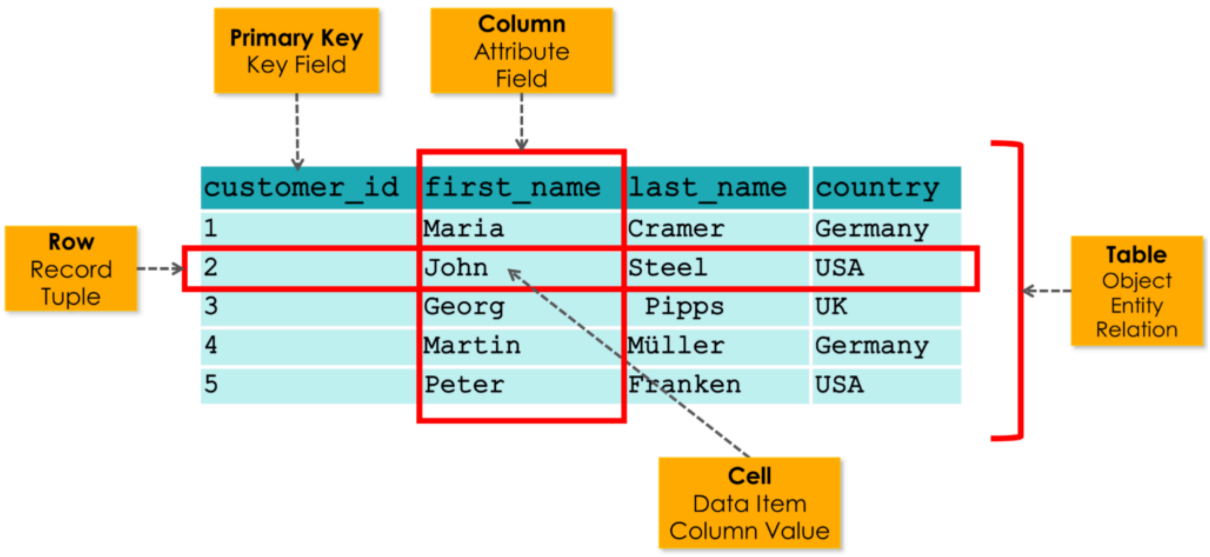
A Database Management System (DBMS) is software that helps users create, manage, and manipulate databases in a structured and organised way. It ensures data is stored efficiently, retrieved quickly, and managed securely. Acting as an interface between users and the database, a DBMS allows operations such as adding, updating, deleting data, and querying (asking questions about the data). It ensures data integrity by enforcing rules and constraints, and enhances data security by controlling access to sensitive information.

Additionally, a DBMS provides features like multi-user access, which allows multiple people to work with the database simultaneously without conflicts. It reduces data redundancy by organising data centrally and efficiently, and supports complex queries through SQL. A DBMS also offers backup and recovery mechanisms to ensure data is safe in case of system failure, and provides concurrency control to maintain data consistency when multiple transactions happen at the same time. Overall, a DBMS simplifies the management of large amounts of data, ensuring accuracy, reliability, and ease of use.

Key Terminologies in DBMS:

- **Relation (Table):** In a relational database, data is organised into tables. Each table is referred to as a *relation*. A table consists of rows and columns, where each row represents a record, and each column represents a specific type of data.
- **Row (Tuple):** A *row* in a table is also known as a *tuple*. It represents a single, complete set of related data. For example, in a table of students, each row might contain the information of one student, such as their name, age, and grade.
- **Column (Attribute):** A *column* in a table is also known as an *attribute*. Each column contains a specific type of data that describes a property of the records in the table. For instance, in a table of students, columns might include "Name", "Age", and "Grade".
- **Attribute (Field):** Each *attribute* (column) in a relation (table) is called a field. Fields contain information about a specific property of the data. For example, a field could be "Name" in a student table, and it will store the names of all students.
- **Domain:** The *domain* of an attribute is the set or pool of all possible values that the column can have. For example, if the column is "Age", the domain would be all valid ages, such as the range of numbers from 1 to 100.
- **Degree:** The *degree* of a relation is the number of columns (attributes) it has. For example, if a table has four columns (Name, Age, Grade, and Address), its degree is 4.
- **Cardinality:** The *cardinality* of a relation is the number of rows (tuples) in a table. For example, if a table has 100 rows (representing 100 students), its cardinality is 100.

- **Record:** A *record* is a complete set of data about a single item in the database, such as a row in a table. For instance, all the information about one student (their name, age, and grade) forms a record.
- **Table:** A *table* is a named collection of records. It stores all the occurrences of a particular type of logical record. For example, a "Students" table might store records about each student.
- **Database Schema:** A *database schema* is the overall design or structure of a database. It defines how the database is organized, including the names of tables, the columns in each table, the type of data each column can hold, and any constraints (rules) applied to the data, like ensuring a column can't be left empty.
- **Metadata:** The *metadata* is data about the structure of the database. It includes information about the database schema, the constraints applied to the data, and other details about how the data is organised and managed. This information is stored by the DBMS in a special area called a *catalogue* or *data dictionary*.



customer_id	first_name	last_name	country
1	Maria	Cramer	Germany
2	John	Steel	USA
3	Georg	Pipps	UK
4	Martin	Müller	Germany
5	Peter	Franken	USA

Concept of Keys:

In relational databases, keys play an important role in uniquely identifying records and establishing relationships between tables. They help enforce *referential integrity*, ensuring that relationships between tables are valid.

1. **Primary Key:** A *primary key* is a column (or a set of columns) that uniquely identifies each record in a table. No two records in a table can have the same primary key value, and it cannot be left empty. For example, in a "Students" table, the "Student ID" could be the primary key because it uniquely identifies each student.
2. **Candidate Key:** A *candidate key* is a column or a combination of columns that could potentially be used as a primary key. A table can have multiple candidate keys, but only one of them will be chosen as the primary key.

3. **Alternate Key:** An *alternate key* is a candidate key that was not selected to be the primary key. It can still uniquely identify records, but it's not used as the primary key.
4. **Foreign Key:** A *foreign key* is a column in one table that links to the primary key of another table. It establishes a relationship between the two tables. For example, if you have a "Courses" table and a "Students" table, the "Student ID" in the "Courses" table could be a foreign key linking back to the primary key in the "Students" table.

Why Use a DBMS?

Using a DBMS offers several advantages:

- **Reduces Data Redundancy:** A DBMS ensures that the same piece of data is not stored in multiple places unnecessarily, which reduces data duplication and inconsistencies.
- **Efficient Data Access:** It provides efficient ways to retrieve and manipulate large volumes of data, improving performance.
- **Data Integrity and Security:** A DBMS helps maintain the accuracy and consistency of data. It also enforces security measures, ensuring only authorised users can access or modify the data.
- **Data Independence:** It separates the data from the applications that use it, meaning that changes to the database structure do not affect the applications.

SQL

Types of SQL Commands:

1) **Data Manipulation Language (DML)** Data Manipulation Language (DML) refers to SQL commands that allow the modification of data stored in database tables. These commands are essential for working with the actual data, whether it involves adding new records, modifying existing ones, or removing data from the database.In short, SQL commands used to interact with the actual data inside the database
   Key DML Commands:
   - INSERT INTO: Adds new rows (records) to a table.
   - UPDATE: Modifies existing data in a table.
   - DELETE: Removes rows from a table.

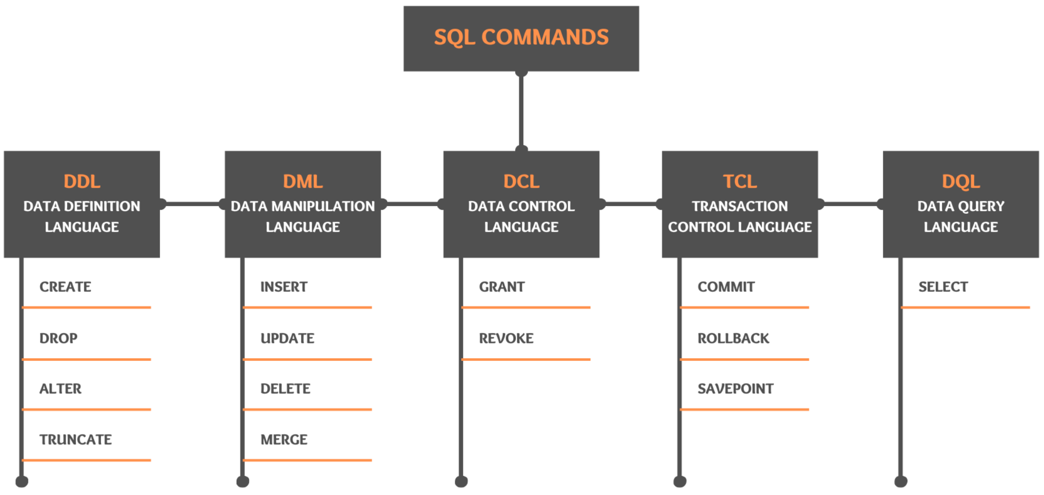
2) **Data Definition Language (DDL)** Data Definition Language (DDL) is used to define and modify the **structure** of the database. DDL commands work with **the design** of

the database rather than the data itself. For example, you can use DDL commands to create a new table or change the design of an existing one.

Key DDL Commands:

- CREATE: Used to create new tables, databases, indexes, or other database objects.
- ALTER: Allows modifications to existing database objects (e.g., adding a new column to a table).
- DROP: Deletes tables, databases, or other objects completely.
- TRUNCATE: Removes all data from a table but keeps its structure intact.

3) **Transaction Control Language (TCL)** Transaction Control Language (TCL) is all about **managing transactions** in a database. A **transaction** is a group of tasks or operations that are performed together as a single unit. Think of it like a package of actions where everything either completes successfully, or nothing is applied (in case of any errors).
-*Commit*– Commits a Transaction.
-*Rollback*– Rollbacks a transaction in case of any errors.
-*Savepoint*–Sets a savepoint within a transaction.
-*Set transition*–specify characteristics for the transaction.



DATABASE CONNECTIVITY

Database connectivity is essential for applications that is needed to interact with databases to store,retrieve, and manipulate data. Python offers several libraries that make it easy to connect to and work with various types of databases.heres a detailed look at how you can achieve database connectivity in python:

MySQL:

It's a popular relational database management system.The **mysql-connector-python** library allows python to connect to MySQL databases.

Steps to connect to a database

1. Install the Required Library:
    - Use ‘pip’ to install the necessary library for your database.

```
pip install mysql-connector-python
```

2.Establish a connection:

- Use the library’s connection method to connect to your database by providing the necessary credentials (host, username, password, database name).

```
import mysql.connector
conn = mysql.connector.connect(
   host="localhost",       # Host
   user="yourusername",      # Username
   password="yourpassword", # Password
   database="yourdatabase"  # Database name
```

3.Execute Queries:

- Use the cursor object to execute SQL queries or commands specific to your database.

```
cursor = conn.cursor()
cursor.execute("SELECT * FROM  Books")
conn.commit() # To commit the transaction
```

4.Fetching Results:

- Fetch and process the results of your queries as needed by using methods like *fetchone()* or *fetchall()* to retrieve the data.

```
cursor.execute("SELECT * FROM Books")
   books = cursor.fetchall()
```

5.Close the Connection:

- Always close the connection to free up resources.

```
conn.close()
```

Python’s extensive library support and straightforward syntax make it an excellent choice for database connectivity, enabling developers to efficiently manage and interact with various types of databases.

IMPLEMENTATION

The **Library Management System** is a simple project built using **Python** and **MySQL** It automates key functions of managing a library, such as adding books, issuing books, returning books, and deleting books from the system. The database is managed using **SQL**, which stores book records for persistence and provides basic data querying.
This system is needed and mainly required in libraries because:

1. **Efficiency**: Book management becomes faster and easier. Instead of manually writing down which book is available or issued, everything is recorded and updated in the database.
2. **Error Reduction**: Since the system handles book records, there is less chance of human errors, such as issuing the same book twice or losing track of issued books.
3. **Quick Access to Information**: You can search for any book by its name and know immediately if it is available or issued.
4. **Automation**: Functions like issuing and returning books are automated, saving time and effort.

<u>Key components of the system:</u>

The main purpose of this project is to create an efficient way to manage books in a library. This system uses modules to handle:

- Adding new books.
  AddBook() -Allows users to add new books to Books list where it is all stored
  Prompt details like Book name, number, author, genre and status
  Also checks for duplicates.
- Viewing all available books.
  ViewBooks()-Displays all the books currently added in the list
  It returns no books available if there are no books entered
- Searching for books.
  SearchBook()-Searches for books by its name and displays it if any matches are found
  Notifies if there are none
- Choosing similar books.
  ChooseBook(matches)-It lists all matching books and allows the user to select one
  based on the unique book number.
- Issuing books.
  IssueBook()-Prompts for a book name to issue it to a user
  If the selected book is ‘available’ , it then changes its status to ‘issued’
  If the book is already issued, it informs the user.
- Returning the books.
  ReturnBook()-Prompts for a book name to return.
  Lets user choose from matching books if any and changes the status
  Back to ‘available’.

- Deleting books.
  DeleteBook()-Allows the user to delete a book by name.
  User can select which book to remove from matching list if any.
  It removes the books from the Book list.

The library system ensures that:

- Books cannot be issued twice without being returned.
- Books that are returned are marked as "available."
- Book records are stored in a persistent **Mysql.connector database**, so even if the program is closed, the data is not lost.

The system is divided into several functions, each performing a specific task related to managing the library.

PYTHON PROGRAM-

```
#Library management system
Books = []

def AddBook():
    while True:
        bname = input("Enter book name: ")
        try:
            bno = int(input("Enter book number: "))
        except ValueError:
            print("Invalid book number. Please enter a valid integer.")
            continue

        # Check if book number already exists
        exists = False
        for book in Books:
            if book[1] == bno:
                exists = True
                break
        if exists:
            print("Sorry! Book number already exists.")
            continue

        author = input("Enter author's name: ")
        genre = input("Enter genre: ")
        status = input("Enter status (Available(A)/Issued(I)): ").lower()

        # Validate status input
        if status in ['a', 'available']:
            status = "available"
        elif status in ['i', 'issued']:
            status = "issued"
        else:
            print("Invalid choice. Please enter 'A' for Available or 'I' for Issued.")
            continue

        Books.append([bname, bno, author, genre, status])
        print("Book added successfully.")
        break

def ViewBooks():
    if Books:
        print("Books List:")
        for book in Books:
            print(book)
    else:
        print("No books available.")

def SearchBook():
    bname = input("Enter book name: ")
    found = False
    for book in Books:
        if book[0].lower() == bname.lower():
            print(book)
            found = True
    if not found:
        print("No books found with that name.")

def ChooseBook(matches):
    if not matches:
        print("No books found with that name.")
        return None

    print("Matching books:")
    for book in matches:
        print(f"Book Number: {book[1]}, Status: {book[4]}")

    while True:
        try:
            bno = int(input("Enter the book number of the book you want to select: "))
            for book in matches:
                if book[1] == bno:
                    return book
            print("Invalid book number. Please choose a valid book number from the list.")
        except ValueError:
            print("Invalid input. Please enter a valid integer for the book number.")

def IssueBook():
    bname = input("Enter the book name you want to issue: ")
    matches = [book for book in Books if book[0].lower() == bname.lower()]

    if len(matches) == 1:  # Only one match, proceed directly
        book = matches[0]
    elif len(matches) > 1:  # Multiple matches, use ChooseBook
        book = ChooseBook(matches)
    else:  # No matches found
        print("No matching books found.")
        return

    if book:
        if book[4] == "available":
            book[4] = "issued"
            print("Book issued successfully.")
        else:
            print("Sorry, this copy of the book is already issued.")

def ReturnBook():
    bname = input("Enter the book name you want to return: ")
    matches = [book for book in Books if book[0].lower() == bname.lower()]

    if len(matches) == 1:  # Only one match, proceed directly
        book = matches[0]
    elif len(matches) > 1:  # Multiple matches, use ChooseBook
        book = ChooseBook(matches)
    else:  # No matches found
        print("No matching books found.")
        return

    if book:
        if book[4] == "issued":
            book[4] = "available"
            print("Book returned successfully.")
        else:
            print("This book is already available.")

def DeleteBook():
    bname = input("Enter the book name you want to delete: ")
    matches = [book for book in Books if book[0].lower() == bname.lower()]

    if len(matches) == 1:  # Only one match, proceed directly
        book = matches[0]
    elif len(matches) > 1:  # Multiple matches, use ChooseBook
        book = ChooseBook(matches)
    else:  # No matches found
        print("No matching books found.")
        return

    if book:
        Books.remove(book)
        print(f"The book '{book[0]}' with book number {book[1]} has been deleted.")

# Main Menu
print("Menu:\n1. Add book\n2. View books\n3. Search book\n4. Issue book\n5. Return book\n6. Delete book\n7. Exit")

while True:
    try:
        choice = int(input("Enter your option: "))
        if choice == 1:
            AddBook()
        elif choice == 2:
            ViewBooks()
        elif choice == 3:
            SearchBook()
        elif choice == 4:
            IssueBook()
        elif choice == 5:
            ReturnBook()
        elif choice == 6:
            DeleteBook()
        elif choice == 7:
            print("Exiting system.")
            break
        else:
            print("Invalid option. Please enter a number between 1 and 7.")
    except ValueError:
        print("Invalid input. Please enter a number.")
```

SQL PROGRAM-

```
import mysql.connector

# Connect to MySQL database
conn = mysql.connector.connect(
    host='your_host',
    user='your_user',
    password='your_password',
    database='library'  # Make sure the 'library' database exists in your MySQL server
)
if conn.is_connected():
    print("successfully connected to MySQL database")

cursor = conn.cursor()

# Create the Books table if it doesn't already exist
cursor.execute('''
CREATE TABLE IF NOT EXISTS Books (
    book_name VARCHAR(255) NOT NULL,
    book_number INT PRIMARY KEY,
    author VARCHAR(255),
    genre VARCHAR(255),
    status ENUM('available', 'issued') NOT NULL
)
''')
conn.commit()

# Add a book to the database
def AddBook():
    while True:
        bname = input("Enter book name: ")
        try:
            bno = int(input("Enter book number: "))
        except ValueError:
            print("Invalid book number. Please enter a valid integer.")
            continue

        # Check if book number already exists
        cursor.execute("SELECT * FROM Books WHERE book_number = %s", (bno,))
        if cursor.fetchone():
            print("Sorry! Book number already exists.")
            continue
        author = input("Enter author's name: ")
        genre = input("Enter genre: ")
        status = input("Enter status (Available(A)/Issued(I)): ").lower()

        # Validate status input
        if status in ['a', 'available']:
            status = "available"
        elif status in ['i', 'issued']:
            status = "issued"
        else:
            print("Invalid choice. Please enter 'A' for Available or 'I' for Issued.")
            continue

        # Insert book into the database
        cursor.execute("INSERT INTO Books (book_name, book_number, author, genre, status)
VALUES (%s, %s, %s, %s, %s)",
                       (bname, bno, author, genre, status))
        conn.commit()
        print("Book added successfully.")
        break

# View all books in the database
def ViewBooks():
    cursor.execute("SELECT * FROM Books")
    books = cursor.fetchall()
    if books:
        print("Books List:")
        for book in books:
            print(book)
    else:
        print("No books available.")

# Search for a book by name
def SearchBook():
    bname = input("Enter book name to search: ").lower()
    cursor.execute("SELECT * FROM Books WHERE LOWER(book_name) = %s",
(bname,))
    books = cursor.fetchall()
    if books:
        print("Matching books:")
        for book in books:
            print(book)
    else:
        print("No books found with that name.")
```

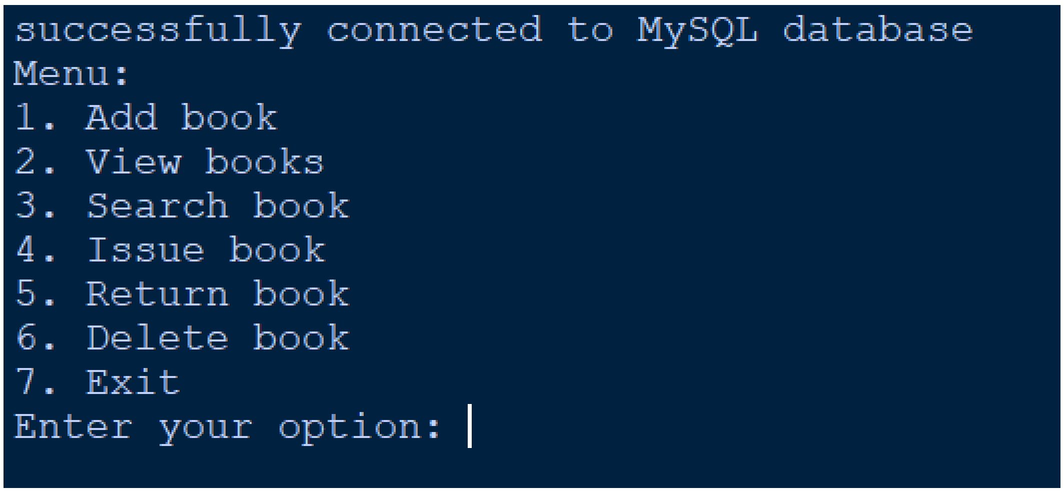
```
# Select a book by its book number from search results
def ChooseBook(matches):
    print("Matching books:")
    for book in matches:
        print(f"Book Number: {book[1]}, Status: {book[4]}")
    while True:
        try:
            bno = int(input("Enter the book number of the book you want to select: "))
            for book in matches:
                if book[1] == bno:
                    return book
            print("Invalid book number. Please choose a valid book number from the list.")
        except ValueError:
            print("Invalid input. Please enter a valid integer for the book number.")

# Issue a book
def IssueBook():
    bname = input("Enter the book name you want to issue: ").lower()
    cursor.execute("SELECT * FROM Books WHERE LOWER(book_name) = %s",
(bname,))
    matches = cursor.fetchall()

    book = matches[0] if len(matches) == 1 else ChooseBook(matches) if matches else None
    if book:
        if book[4] == "available":
            cursor.execute("UPDATE Books SET status = 'issued' WHERE book_number = %s",
(book[1],))
            conn.commit()
            print("Book issued successfully.")
        else:
            print("Sorry, this copy of the book is already issued.")
    else:
        print("No matching books found.")

# Return a book
def ReturnBook():
    bname = input("Enter the book name you want to return: ").lower()
    cursor.execute("SELECT * FROM Books WHERE LOWER(book_name) = %s AND
status = 'issued'", (bname,))
    matches = cursor.fetchall()

    book = matches[0] if len(matches) == 1 else ChooseBook(matches) if matches else None
    if book:
        cursor.execute("UPDATE Books SET status = 'available' WHERE book_number = %s", (book[1],))
        conn.commit()
        print("Book returned successfully.")
    else:
        print("The book cannot be returned or is already available.")

# Delete a book
def DeleteBook():
    bname = input("Enter the book name you want to delete: ").lower()
    cursor.execute("SELECT * FROM Books WHERE LOWER(book_name) = %s", (bname,))
    matches = cursor.fetchall()

    book = matches[0] if len(matches) == 1 else ChooseBook(matches) if matches else None
    if book:
        cursor.execute("DELETE FROM Books WHERE book_number = %s", (book[1],))
        conn.commit()
        print(f"The book '{book[0]}' with book number {book[1]} has been deleted.")
    else:
        print("Book not found.")

# Main Menu
def main():
    print("Menu:\n1. Add book\n2. View books\n3. Search book\n4. Issue book\n5. Return book\n6. Delete book\n7. Exit")
    while True:
        try:
            ch = int(input("Enter your option: "))
            if ch == 1:
                AddBook()
            elif ch == 2:
                ViewBooks()
            elif ch == 3:
                SearchBook()
            elif ch == 4:
                IssueBook()
            elif ch == 5:
                ReturnBook()
            elif ch == 6:
                DeleteBook()
            elif ch == 7:
                print("Exiting system.")
                break
            else:
```

```
            print("Invalid option. Please enter a number between 1 and 7.")
        except ValueError:
            print("Invalid input. Please enter a number.")

# Run the main menu
main()

# Close the connection when done
conn.close()
```

RESULTS

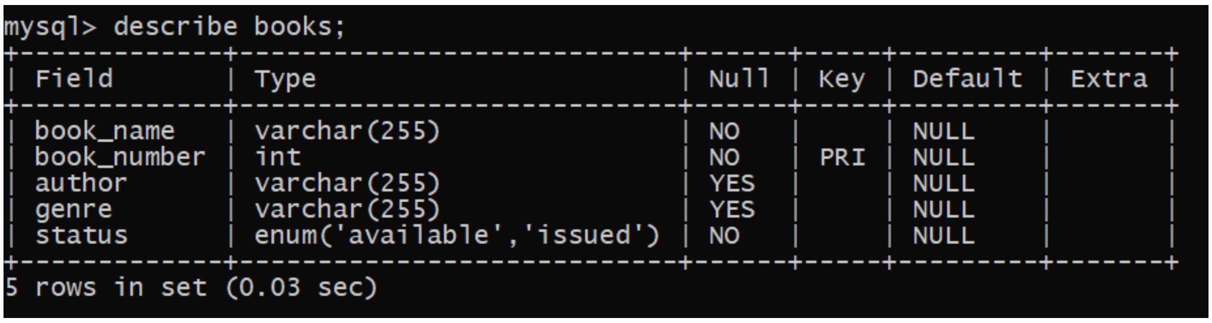
PYTHON

The program offers seven options for users to choose from. These options, displayed in the screenshot below, allow users to execute various functions by simply typing the corresponding number. This setup provides a convenient way to navigate the library management system through SQL.



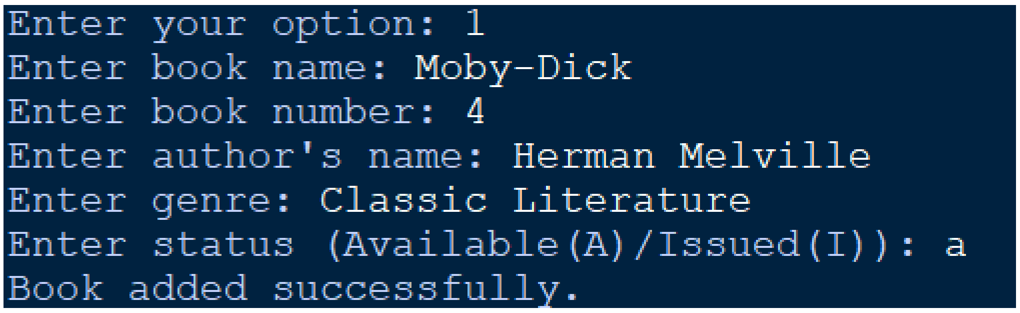
```
successfully connected to MySQL database
Menu:
1. Add book
2. View books
3. Search book
4. Issue book
5. Return book
6. Delete book
7. Exit
Enter your option: 
```

STRUCTURE



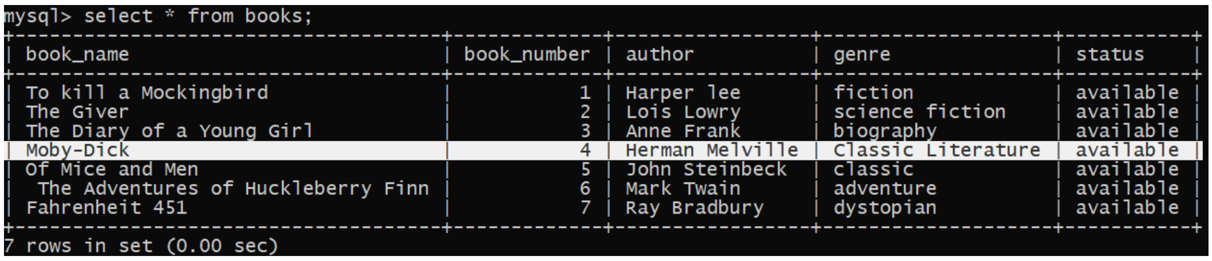
```
mysql> describe books;
+-------------+--------------------------+------+-----+---------+-------+
| Field       | Type                     | Null | Key | Default | Extra |
+-------------+--------------------------+------+-----+---------+-------+
| book_name   | varchar(255)             | NO   |     | NULL    |       |
| book_number | int                      | NO   | PRI | NULL    |       |
| author      | varchar(255)             | YES  |     | NULL    |       |
| genre       | varchar(255)             | YES  |     | NULL    |       |
| status      | enum('available','issued') | NO   |     | NULL    |       |
+-------------+--------------------------+------+-----+---------+-------+
5 rows in set (0.03 sec)
```

1. ADD BOOK



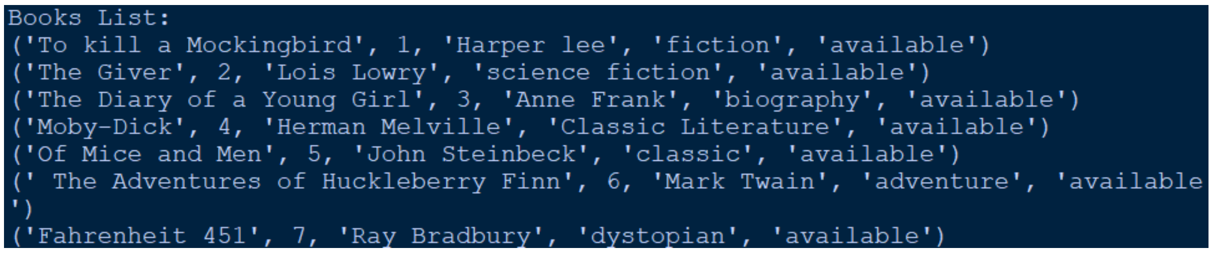
```
Enter your option: 1
Enter book name: Moby-Dick
Enter book number: 4
Enter author's name: Herman Melville
Enter genre: Classic Literature
Enter status (Available(A)/Issued(I)): a
Book added successfully.
```

SQL



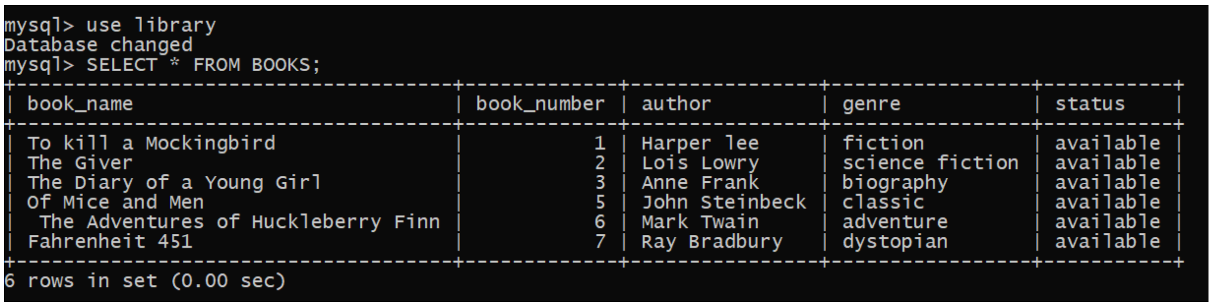
```
mysql> select * from books;
+------------------------------------+-------------+-----------------+--------------------+-----------+
| book_name                          | book_number | author          | genre              | status    |
+------------------------------------+-------------+-----------------+--------------------+-----------+
| To kill a Mockingbird              |           1 | Harper lee      | fiction            | available |
| The Giver                          |           2 | Lois Lowry      | science fiction    | available |
| The Diary of a Young Girl          |           3 | Anne Frank      | biography          | available |
| Moby-Dick                          |           4 | Herman Melville | Classic Literature | available |
| Of Mice and Men                    |           5 | John Steinbeck  | classic            | available |
|  The Adventures of Huckleberry Finn |          6 | Mark Twain      | adventure          | available |
| Fahrenheit 451                     |           7 | Ray Bradbury    | dystopian          | available |
+------------------------------------+-------------+-----------------+--------------------+-----------+
7 rows in set (0.00 sec)
```

2.VIEW BOOK



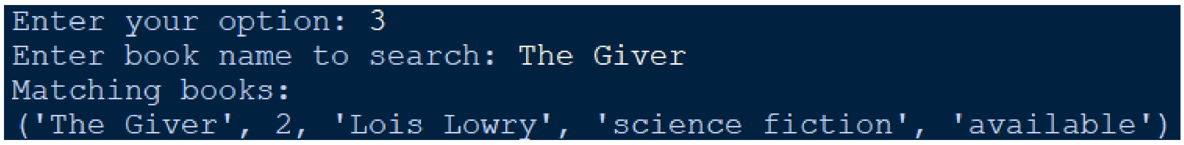
```
Books List:
('To kill a Mockingbird', 1, 'Harper lee', 'fiction', 'available')
('The Giver', 2, 'Lois Lowry', 'science fiction', 'available')
('The Diary of a Young Girl', 3, 'Anne Frank', 'biography', 'available')
('Moby-Dick', 4, 'Herman Melville', 'Classic Literature', 'available')
('Of Mice and Men', 5, 'John Steinbeck', 'classic', 'available')
(' The Adventures of Huckleberry Finn', 6, 'Mark Twain', 'adventure', 'available
')
('Fahrenheit 451', 7, 'Ray Bradbury', 'dystopian', 'available')
```

STORED DATA



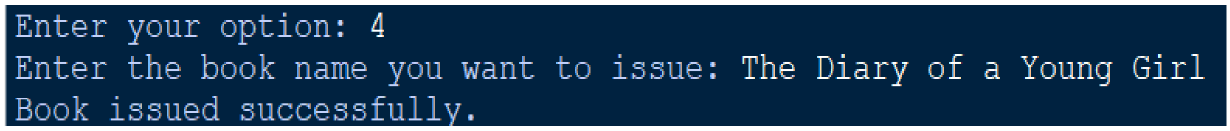
```
mysql> use library
Database changed
mysql> SELECT * FROM BOOKS;
+-------------------------------------+-------------+----------------+-----------------+-----------+
| book_name                           | book_number | author         | genre           | status    |
+-------------------------------------+-------------+----------------+-----------------+-----------+
| To kill a Mockingbird               |           1 | Harper lee     | fiction         | available |
| The Giver                           |           2 | Lois Lowry     | science fiction | available |
| The Diary of a Young Girl           |           3 | Anne Frank     | biography       | available |
| Of Mice and Men                     |           5 | John Steinbeck | classic         | available |
|  The Adventures of Huckleberry Finn |           6 | Mark Twain     | adventure       | available |
| Fahrenheit 451                      |           7 | Ray Bradbury   | dystopian       | available |
+-------------------------------------+-------------+----------------+-----------------+-----------+
6 rows in set (0.00 sec)
```

3.SEARCH BOOK



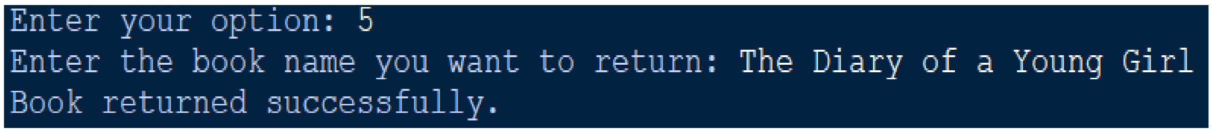
```
Enter your option: 3
Enter book name to search: The Giver
Matching books:
('The Giver', 2, 'Lois Lowry', 'science fiction', 'available')
```

4.ISSUE BOOK



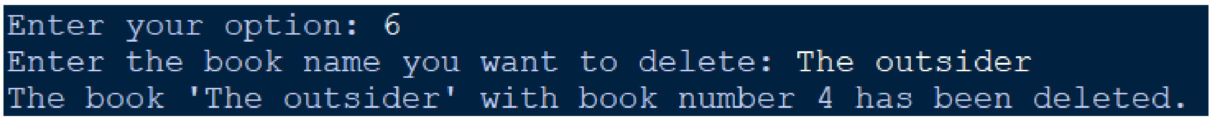
```
Enter your option: 4
Enter the book name you want to issue: The Diary of a Young Girl
Book issued successfully.
```

5.RETURN BOOK



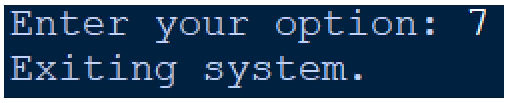
```
Enter your option: 5
Enter the book name you want to return: The Diary of a Young Girl
Book returned successfully.
```

6.DELETE BOOK



```
Enter your option: 6
Enter the book name you want to delete: The outsider
The book 'The outsider' with book number 4 has been deleted.
```

7.EXIT



```
Enter your option: 7
Exiting system.
```

CONCLUSION

The **Library Management System** project demonstrates the efficient use of Python programming to handle typical library operations such as adding, issuing, searching, returning, and deleting books from a collection. By incorporating basic database management concepts, this project allows for easy and effective manipulation of data stored in lists, which mimics a simplified version of real-life database management systems.

One of the key takeaways from this project is the understanding of how data can be organized and manipulated using structured programming. The system effectively showcases the use of Python functions to create modular, reusable code for performing specific tasks such as issuing or returning books. This modular approach not only simplifies the code but also enhances readability and maintainability, making it easier to modify and scale the project in the future.

The project also demonstrates core database management concepts, such as **data integrity** (ensuring that books are either "available" or "issued"), **data retrieval** (searching for books by name), and **data consistency** (making sure that the status of a book is updated when it is issued or returned). These principles are critical in real-world database systems, and this project serves as a foundation for understanding them.

In terms of features:

- The project allows users to add new books with essential details like book name, author, genre, and status (available/issued).
- It supports searching for books by name, making it user-friendly for both librarians and users.
- The system efficiently handles issuing and returning books, updating the book's availability status.
- It also provides functionality for deleting books that are no longer part of the collection.

In conclusion, the **Library Management System** is a valuable project that covers the core aspects of database management in a simple yet effective way. With potential enhancements, it can evolve into a more sophisticated system, offering real-world applications and deeper insights into database connectivity and management.

BIBLIOGRAPHY

[1] Sumita Arora. *Computer Science with Python - Class XII*. Dhanpat Rai & Co.

[2] Python Documentation. Python Software Foundation, 2021.

[3] SQLite Documentation.

[4] W3Schools - SQL Tutorial.

[5] GeeksforGeeks - DBMS Keys.