COMPUTER PROGRAMMING 2

M001

SEMI FINAL OUTPUT PORTFOLIO

SUBMITTED BY:
CARL VICTOR A. BUENAFE

SUBMITTED TO:
DEAN RODRIGO BELLEZA JR.

SEMI-FINAL COURSE OUTCOMES:

CREATE A PYTHON PROGRAM CAPABLE OF EXECUTING CREATE, RETRIEVE, UPDATE, AND DELETE (CRUD) OPERATIONS WITHIN A DATABASE USING STRUCTURED QUERY LANGUAGE (SQL).

PERFORMANCE INDICATOR

CONSTRUCT AN OBJECT-ORIENTED PYTHON PROGRAM THAT EXECUTES CRUD (CREATE, RETRIEVE, UPDATE, DELETE) OPERATIONS WITHIN A MYSQL DATABASE UTILIZING STRUCTURED QUERY LANGUAGE (SQL).
THE PERFORMANCE INDICATOR REQUIRES CREATING AN OBJECT-ORIENTED PYTHON PROGRAM THAT INTERACTS WITH A MYSQL DATABASE TO EXECUTE CRUD OPERATIONS USING SQL. THIS INVOLVES ESTABLISHING A CONNECTION TO THE DATABASE, IMPLEMENTING METHODS FOR EACH CRUD OPERATION (CREATE, RETRIEVE, UPDATE, DELETE) USING SQL STATEMENTS, HANDLING ERRORS, AND CONDUCTING THOROUGH TESTING TO ENSURE FUNCTIONALITY AND RELIABILITY.

SYSTEM DESCRIPTION

THE SYSTEM DESCRIPTION CALLS FOR THE DEVELOPMENT OF A PYTHON PROGRAM CAPABLE OF PERFORMING CRUD OPERATIONS (CREATE, RETRIEVE, UPDATE, DELETE) ON A DATABASE USING SQL. THIS INVOLVES CONNECTING TO THE DATABASE, DEFINING FUNCTIONS FOR EACH CRUD OPERATION, EXECUTING SQL STATEMENTS WITHIN THESE FUNCTIONS, HANDLING ERRORS, AND TESTING THE PROGRAM TO ENSURE ITS FUNCTIONALITY.

LIST OF TABLES

Body_Style_ID	Body_Style_Name
1	Sedan
2	Coupe
3	Hatchback
NULL	NULL

```
CREATE TABLE BODY_STYLES (
 BODY_STYLE_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 BODY_STYLE_NAME VARCHAR(255) NOT NULL UNIQUE
);

INSERT INTO BODY_STYLES (BODY_STYLE_NAME) VALUES
('SEDAN'),
('COUPE'),
('HATCHBACK');

SELECT * FROM BODY_STYLES;
```

Engine_Type_ID	Engine_Description
1	1.5L Turbocharged I4
3	1.8L I4
2	2.0L Turbocharged I4
NULL	NULL

```
CREATE TABLE ENGINE_TYPES (
 ENGINE_TYPE_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 ENGINE_DESCRIPTION VARCHAR(255) NOT NULL UNIQUE
);

INSERT INTO ENGINE_TYPES (ENGINE_DESCRIPTION) VALUES
('1.5L TURBOCHARGED I4'),
('2.0L TURBOCHARGED I4'),
('1.8L I4');

SELECT * FROM ENGINE_TYPES;
```

LIST OF TABLES

	Model_Year	Horsepower
▶	2024	220

```
CREATE TABLE CIVIC_MODELS (
 MODEL_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 MODEL_YEAR INT NOT NULL,
 BODY_STYLE_ID INT NOT NULL,
 ENGINE_TYPE_ID INT NOT NULL,
 HORSEPOWER INT,
 TRANSMISSION VARCHAR(255),
 FOREIGN KEY (BODY_STYLE_ID) REFERENCES BODY_STYLES(BODY_STYLE_ID),
 FOREIGN KEY (ENGINE_TYPE_ID) REFERENCES ENGINE_TYPES(ENGINE_TYPE_ID)
);

INSERT INTO CIVIC_MODELS (MODEL_YEAR, BODY_STYLE_ID, ENGINE_TYPE_ID,
HORSEPOWER, TRANSMISSION) VALUES
(2023, 1, 1, 180, 'CVT'),
(2023, 1, 2, 200, 'MANUAL'),
(2020, 3, 1, 174, 'CVT'),
(2018, 3, 1, 170, 'CVT'),
(2015, 1, 3, 140, 'CVT'),
(2024, 2, 2, 220, 'AUTOMATIC');

SELECT MODEL_YEAR, HORSEPOWER FROM CIVIC_MODELS WHERE HORSEPOWER > 200;
```

	Model_ID	Model_Year	Body_Style_ID	Engine_Type_ID	Horsepower	Transmission
	1	2023	1	1	180	CVT
	2	2023	1	2	200	Manual
▶*	NULL	NULL	NULL	NULL	NULL	NULL

```
SELECT * FROM CIVIC_MODELS WHERE MODEL_YEAR = 2023;
```

LIST OF TABLES

	Body_Style_Name	Model_Year	Engine_Description	Horsepower	Transmission
▶	Coupe	2024	2.0L Turbocharged I4	220	Automatic
	Sedan	2023	1.5L Turbocharged I4	180	CVT
	Sedan	2023	2.0L Turbocharged I4	200	Manual
	Hatchback	2020	1.5L Turbocharged I4	174	CVT
	Hatchback	2018	1.5L Turbocharged I4	170	CVT
	Sedan	2015	1.8L I4	140	CVT

```
SELECT BS.BODY_STYLE_NAME, CM.MODEL_YEAR, ET.ENGINE_DESCRIPTION,
CM.HORSEPOWER, CM.TRANSMISSION
FROM CIVIC_MODELS CM
INNER JOIN BODY_STYLES BS ON CM.BODY_STYLE_ID = BS.BODY_STYLE_ID
INNER JOIN ENGINE_TYPES ET ON CM.ENGINE_TYPE_ID = ET.ENGINE_TYPE_ID
ORDER BY CM.MODEL_YEAR DESC;
```

SQL

```
CREATE DATABASE HONDA_CIVIC_HISTORY;

DROP DATABASE HONDA_CIVIC_HISTORY; -- ONLY IF MISTAKE IS MADE

USE HONDA_CIVIC_HISTORY;

CREATE TABLE BODY_STYLES (
  BODY_STYLE_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  BODY_STYLE_NAME VARCHAR(255) NOT NULL UNIQUE
);

CREATE TABLE ENGINE_TYPES (
  ENGINE_TYPE_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  ENGINE_DESCRIPTION VARCHAR(255) NOT NULL UNIQUE
);

CREATE TABLE CIVIC_MODELS (
  MODEL_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  MODEL_YEAR INT NOT NULL,
  BODY_STYLE_ID INT NOT NULL,
  ENGINE_TYPE_ID INT NOT NULL,
  HORSEPOWER INT,
  TRANSMISSION VARCHAR(255),
  FOREIGN KEY (BODY_STYLE_ID) REFERENCES BODY_STYLES(BODY_STYLE_ID),
  FOREIGN KEY (ENGINE_TYPE_ID) REFERENCES ENGINE_TYPES(ENGINE_TYPE_ID)
);

-- INSERT BODY STYLES
INSERT INTO BODY_STYLES (BODY_STYLE_NAME) VALUES
('SEDAN'),
('COUPE'),
('HATCHBACK');

-- INSERT ENGINE TYPES
INSERT INTO ENGINE_TYPES (ENGINE_DESCRIPTION) VALUES
('1.5L TURBOCHARGED I4'),
('2.0L TURBOCHARGED I4'),
('1.8L I4');

-- INSERT CIVIC MODELS
INSERT INTO CIVIC_MODELS (MODEL_YEAR, BODY_STYLE_ID, ENGINE_TYPE_ID, HORSEPOWER,
TRANSMISSION) VALUES
(2023, 1, 1, 180, 'CVT'),
(2023, 1, 2, 200, 'MANUAL'),
(2020, 3, 1, 174, 'CVT'),
(2018, 3, 1, 170, 'CVT'),
(2015, 1, 3, 140, 'CVT'),
(2024, 2, 2, 220, 'AUTOMATIC');

-- DELETE A RECORD
DELETE FROM CIVIC_MODELS WHERE MODEL_ID = 3;

-- UPDATE ENGINE TYPE FOR A MODEL
UPDATE CIVIC_MODELS SET ENGINE_TYPE_ID = 2 WHERE MODEL_ID = 4;

-- RETRIEVE DATA


SELECT * FROM BODY_STYLES;


SELECT * FROM ENGINE_TYPES;


SELECT MODEL_YEAR, HORSEPOWER FROM CIVIC_MODELS WHERE HORSEPOWER > 200;


SELECT * FROM CIVIC_MODELS WHERE MODEL_YEAR = 2023;


SELECT BS.BODY_STYLE_NAME, CM.MODEL_YEAR, ET.ENGINE_DESCRIPTION, CM.HORSEPOWER,
CM.TRANSMISSION
FROM CIVIC_MODELS CM
INNER JOIN BODY_STYLES BS ON CM.BODY_STYLE_ID = BS.BODY_STYLE_ID
INNER JOIN ENGINE_TYPES ET ON CM.ENGINE_TYPE_ID = ET.ENGINE_TYPE_ID
ORDER BY CM.MODEL_YEAR DESC;
```

COMPLETE PYTHON SOURCE CODE

```
IMPORT MYSQL.CONNECTOR
FROM MYSQL.CONNECTOR IMPORT ERRORCODE
IMPORT OS

# DATABASE CONNECTION DETAILS
DB_NAME = "HONDA_CIVIC_HISTORY"
DB_USER = "ROOT"
DB_PASSWORD = "ALIAHCAKES"
DB_HOST = "LOCALHOST"
DB_PORT = 3306  # REPLACE WITH THE CORRECT PORT NUMBER (INTEGER)

# CONNECTION
DEF CONNECT():
    TRY:
        CON = MYSQL.CONNECTOR.CONNECT(
            USER=DB_USER,
            PASSWORD=DB_PASSWORD,
            HOST=DB_HOST,
            PORT=DB_PORT,
            DATABASE=DB_NAME
        )
        PRINT('CONNECTION SUCCESSFUL')
        RETURN CON
    EXCEPT MYSQL.CONNECTOR.ERROR AS ERR:
        IF ERR.ERRNO == ERRORCODE.ER_ACCESS_DENIED_ERROR:
            PRINT("SOMETHING IS WRONG WITH YOUR USERNAME OR PASSWORD")
        ELIF ERR.ERRNO == ERRORCODE.ER_BAD_DB_ERROR:
            PRINT("DATABASE DOES NOT EXIST")
                        ELIF    ERR.ERRNO    ==    ERRORCODE.ER_HOST_NOT_PRIVILEGED    OR    ERR.ERRNO    ==
ERRORCODE.ER_HOST_IS_BLOCKED:
            PRINT(F"ERROR CONNECTING TO THE DATABASE: {ERR}")
        ELSE:
            PRINT(F"ERROR CONNECTING TO THE DATABASE: {ERR}")
        RETURN NONE

# CREATE (INSERT) RECORD
DEF ADD_CIVIC_MODEL():
    CON = CONNECT()
    TRY:
        WHILE TRUE:
            MODEL_YEAR = INT(INPUT('ENTER MODEL YEAR: '))
            BODY_STYLE_ID = INT(INPUT('ENTER BODY STYLE ID (FROM AVAILABLE OPTIONS): '))
            ENGINE_TYPE_ID = INT(INPUT('ENTER ENGINE TYPE ID (FROM AVAILABLE OPTIONS): '))
            HORSEPOWER = INT(INPUT('ENTER HORSEPOWER: '))
            TRANSMISSION = INPUT('ENTER TRANSMISSION (E.G., CVT, MANUAL): ')

            # EXECUTE SQL
            CURSOR = CON.CURSOR()
                    SQL  =  "INSERT  INTO  CIVIC_MODELS  (MODEL_YEAR,  BODY_STYLE_ID,  ENGINE_TYPE_ID,
HORSEPOWER, TRANSMISSION) VALUES (%S, %S, %S, %S, %S)"
            VALUES = (MODEL_YEAR, BODY_STYLE_ID, ENGINE_TYPE_ID, HORSEPOWER, TRANSMISSION)
            CURSOR.EXECUTE(SQL, VALUES)
            CON.COMMIT()

            PRINT('NEW CIVIC MODEL RECORD ADDED SUCCESSFULLY')
            X = INPUT("DO YOU WANT TO ADD ANOTHER RECORD [Y/N]? ")
            IF X == 'Y':
                OS.SYSTEM('CLS')
                CONTINUE
            ELSE:
                BREAK

    EXCEPT MYSQL.CONNECTOR.ERROR AS ERR:
        PRINT(ERR)

    ELSE:
        CURSOR.CLOSE()
        CON.CLOSE()

# RETRIEVE (SELECT) RECORDS
DEF RETRIEVE_CIVIC_MODELS(CRITERIA=NONE):
    CON = CONNECT()
    TRY:
        CURSOR = CON.CURSOR()

        IF NOT CRITERIA:
            # RETRIEVE ALL RECORDS
              SQL = "SELECT CM.*, BS.BODY_STYLE_NAME, ET.ENGINE_DESCRIPTION FROM CIVIC_MODELS CM
INNER JOIN BODY_STYLES BS ON CM.BODY_STYLE_ID = BS.BODY_STYLE_ID INNER JOIN ENGINE_TYPES ET
ON CM.ENGINE_TYPE_ID = ET.ENGINE_TYPE_ID ORDER BY CM.MODEL_YEAR DESC"
```

COMPLETE PYTHON SOURCE CODE

```
 ELSE:
 # RETRIEVE BASED ON CRITERIA (E.G., MODEL YEAR, HORSEPOWER)
    SQL = F"SELECT CM.*, BS.BODY_STYLE_NAME, ET.ENGINE_DESCRIPTION FROM CIVIC_MODELS CM INNER JOIN BODY_STYLES BS ON CM.BODY_STYLE_ID = BS.BODY_STYLE_ID INNER JOIN ENGINE_TYPES ET ON CM.ENGINE_TYPE_ID = ET.ENGINE_TYPE_ID WHERE {CRITERIA} ORDER BY CM.MODEL_YEAR DESC"

 CURSOR.EXECUTE(SQL)
 ROWS = CURSOR.FETCHALL()

 IF ROWS:
 PRINT("HONDA CIVIC MODELS:")
 FOR ROW IN ROWS:
 PRINT(F" MODEL YEAR: {ROW[1]}")
 PRINT(F" BODY STYLE: {ROW[6]}")
 PRINT(F" ENGINE TYPE: {ROW[3]}")
 PRINT(F" HORSEPOWER: {ROW[4]}")
 PRINT(F" TRANSMISSION: {ROW[5]}")
 PRINT()
 ELSE:
 PRINT("NO RECORDS FOUND BASED ON THE PROVIDED CRITERIA.")

 EXCEPT MYSQL.CONNECTOR.ERROR AS ERR:
 PRINT(ERR)

 ELSE:
 CURSOR.CLOSE()
 CON.CLOSE()

# UPDATE RECORD
DEF UPDATE_CIVIC_MODEL():
 CON = CONNECT()
 TRY:
 WHILE TRUE:
 MODEL_ID = INT(INPUT('ENTER MODEL ID OF THE RECORD TO UPDATE: '))
 UPDATE_CHOICE = INPUT('UPDATE (1) BODY STYLE, (2) ENGINE TYPE, (3) HORSEPOWER, (4) TRANSMISSION: ')

 CURSOR = CON.CURSOR()
 IF UPDATE_CHOICE == '1':
 NEW_BODY_STYLE_ID = INT(INPUT('ENTER NEW BODY STYLE ID: '))
 SQL = "UPDATE CIVIC_MODELS SET BODY_STYLE_ID = %S WHERE MODEL_ID = %S"
```

COMPLETE PYTHON SOURCE CODE

```
 VALUES = (NEW_BODY_STYLE_ID, MODEL_ID)
        ELIF UPDATE_CHOICE == '2':
            NEW_ENGINE_TYPE_ID = INT(INPUT('ENTER NEW ENGINE TYPE ID: '))
            SQL = "UPDATE CIVIC_MODELS SET ENGINE_TYPE_ID = %S WHERE MODEL_ID = %S"
            VALUES = (NEW_ENGINE_TYPE_ID, MODEL_ID)
        ELIF UPDATE_CHOICE == '3':
            NEW_HORSEPOWER = INT(INPUT('ENTER NEW HORSEPOWER: '))
            SQL = "UPDATE CIVIC_MODELS SET HORSEPOWER = %S WHERE MODEL_ID = %S"
            VALUES = (NEW_HORSEPOWER, MODEL_ID)
        ELIF UPDATE_CHOICE == '4':
            NEW_TRANSMISSION = INPUT('ENTER NEW TRANSMISSION: ')
            SQL = "UPDATE CIVIC_MODELS SET TRANSMISSION = %S WHERE MODEL_ID = %S"
            VALUES = (NEW_TRANSMISSION, MODEL_ID)
        ELSE:
            PRINT("INVALID UPDATE CHOICE. PLEASE TRY AGAIN.")
            CONTINUE

        CURSOR.EXECUTE(SQL, VALUES)
        CON.COMMIT()

        PRINT('CIVIC MODEL RECORD UPDATED SUCCESSFULLY')
        X = INPUT("DO YOU WANT TO UPDATE ANOTHER RECORD [Y/N]? ")
        IF X == 'Y':
            OS.SYSTEM('CLS')
            CONTINUE
        ELSE:
            BREAK

    EXCEPT MYSQL.CONNECTOR.ERROR AS ERR:
        PRINT(ERR)

    ELSE:
        CURSOR.CLOSE()
        CON.CLOSE()

# DELETE RECORD
DEF DELETE_CIVIC_MODEL():
    CON = CONNECT()
    TRY:
        WHILE TRUE:
            MODEL_ID = INT(INPUT('ENTER MODEL ID OF THE RECORD TO DELETE: '))
```

COMPLETE PYTHON SOURCE CODE

```
CURSOR = CON.CURSOR()
 SQL = F"DELETE FROM CIVIC_MODELS WHERE MODEL_ID = %S"
 VALUE = (MODEL_ID,)
 CURSOR.EXECUTE(SQL, VALUE)
 CON.COMMIT()

 PRINT('CIVIC MODEL RECORD DELETED SUCCESSFULLY')
 X = INPUT("DO YOU WANT TO DELETE ANOTHER RECORD [Y/N]? ")
 IF X == 'Y':
 OS.SYSTEM('CLS')
 CONTINUE
 ELSE:
 BREAK

 EXCEPT MYSQL.CONNECTOR.ERROR AS ERR:
 PRINT(ERR)

 ELSE:
 CURSOR.CLOSE()
 CON.CLOSE()

DEF MAIN():
 WHILE TRUE:
 OS.SYSTEM('CLS') # CLEAR THE CONSOLE
 PRINT("\NHONDA CIVIC HISTORY MANAGEMENT SYSTEM")
 PRINT("1. ADD NEW CIVIC MODEL")
 PRINT("2. VIEW CIVIC MODELS")
 PRINT("3. UPDATE CIVIC MODEL")
 PRINT("4. DELETE CIVIC MODEL")
 PRINT("5. EXIT")

 CHOICE = INPUT("ENTER YOUR CHOICE: ")

 IF CHOICE == "1":
 OS.SYSTEM('CLS')
 CON = CONNECT()
 ADD_CIVIC_MODEL()
 ELIF CHOICE == "2":
 OS.SYSTEM('CLS')
 CON = CONNECT()
 CRITERIA = INPUT("ENTER SEARCH CRITERIA (E.G., MODEL YEAR > 2020): ")
```

COMPLETE PYTHON SOURCE CODE

```
RETRIEVE_CIVIC_MODELS(CRITERIA)
 ELIF CHOICE == "3":
 OS.SYSTEM('CLS')
 CON = CONNECT()
 UPDATE_CIVIC_MODEL()
 ELIF CHOICE == "4":
 OS.SYSTEM('CLS')
 CON = CONNECT()
 DELETE_CIVIC_MODEL()
 ELIF CHOICE == "5":
 PRINT("EXITING PROGRAM...")
 BREAK
 ELSE:
 PRINT("INVALID CHOICE. PLEASE TRY AGAIN.")

 X = INPUT("RETURN TO MAIN MENU [Y/N]? ")
 IF X != 'Y':
 BREAK

IF __NAME__ == "__MAIN__":
 MAIN()
```

SCREENSHOTS OF OUTPUT

```
Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice: 1
Connection successful
Enter Model Year: 2023
Enter Body Style ID (from available options): 1
Enter Engine Type ID (from available options): 2
Enter Horsepower: 100
Enter Transmission (e.g., CVT, Manual): Manual
Civic model record added successfully

Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice:
```

```
Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice: 2
Enter search criteria (e.g., Model Year > 2020): 2024
Connection successful
Honda Civic Models:
  Model Year: 2024
  Body Style: Coupe
  Engine Type: 2
  Horsepower: 220
  Transmission: Automatic

  Model Year: 2023
  Body Style: Sedan
  Engine Type: 1
  Horsepower: 180
  Transmission: CVT

  Model Year: 2023
  Body Style: Sedan
  Engine Type: 2
  Horsepower: 200
  Transmission: Manual

  Model Year: 2023
  Body Style: Sedan
  Engine Type: 2
  Horsepower: 100
  Transmission: Manual

  Model Year: 2020
  Body Style: Hatchback
  Engine Type: 1
  Horsepower: 174
  Transmission: CVT
```

SCREENSHOTS OF OUTPUT

```
  Model Year: 2018
  Body Style: Hatchback
  Engine Type: 1
  Horsepower: 170
  Transmission: CVT

  Model Year: 2015
  Body Style: Sedan
  Engine Type: 3
  Horsepower: 140
  Transmission: CVT


Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice:
```

```
Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice: 3
Connection successful
Enter Model ID of the record to update: 1
Update (1) Body Style, (2) Engine Type, (3) Horsepower, (4) Transmission: 3
Enter new Horsepower: 200
Civic model record updated successfully

Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice:
```

SCREENSHOTS OF OUTPUT

```
Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice: 4
Connection successful
Enter Model ID of the record to delete: 2
Civic model record deleted successfully

Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice:
```

```
Honda Civic History Management System
1. Add New Civic Model
2. View Civic Models
3. Update Civic Model
4. Delete Civic Model
5. Exit
Enter your choice: 5
Exiting program...
>>>
```

LEARNING REFLECTION

IN COMPUTER PROGRAMMING 2, I EXPLORED THE LEARNING OBJECTIVES RELATED TO SQL DURING THE SEMI-FINAL PHASE OF THE COURSE. THIS REQUIRED BECOMING FAMILIAR WITH MYSQL WORKBENCH AND MYSQL SERVER INSTALLATION. HOWEVER, I FOUND THE INSTALLATION PROCESS TO BE VERY FRUSTRATING BECAUSE IT WAS DIFFICULT TO UNDERSTAND THE DETAILED INSTRUCTIONS.

HOWEVER, THE MAIN PURPOSE OF SQL WAS TO MAKE IT EASIER TO CREATE DATABASES AND USE STRUCTURED QUERY LANGUAGE (SQL) TO PERFORM CRUD (CREATE, RETRIEVE, UPDATE, DELETE) ACTIVITIES WITHIN THESE DATABASES. THIS REQUIRED NOT JUST INSTALLING MYSQL AND ALL OF ITS PREREQUISITES, BUT ALSO KNOWING HOW TO WORK WITH DATABASES BY USING SQL QUERIES.

INSTALLING THE MYSQL PYTHON CONNECTOR, WHICH ENABLES COMMUNICATION BETWEEN PYTHON CODE AND MYSQL DATABASES, WAS ANOTHER STEP IN THE LEARNING PROCESS. WITH THE HELP OF THIS INTEGRATION, I WAS ABLE TO EASILY LINK PYTHON AND MYSQL TOGETHER IN A SINGLE PYTHON CODE, FACILITATING JOINT USE OF THE TWO PLATFORMS.

DESPITE THE INITIAL DIFFICULTIES ENCOUNTERED DURING THE INSTALLATION PROCEDURE, THE KNOWLEDGE GAINED FROM THE EXPERIENCE WAS INVALUABLE IN UNDERSTANDING HOW SQL IS INTEGRATED INTO PYTHON PROGRAMMING. IN ORDER TO INCREASE PRODUCTIVITY AND EFFICIENCY IN MY PROGRAMMING PURSUITS, MY FUTURE GOALS INCLUDE REFINING MY UNDERSTANDING OF DATABASE MANAGEMENT, EXPLORING AND MASTERING SQL CONCEPTS, AND STREAMLINING THE INSTALLATION PROCESS.