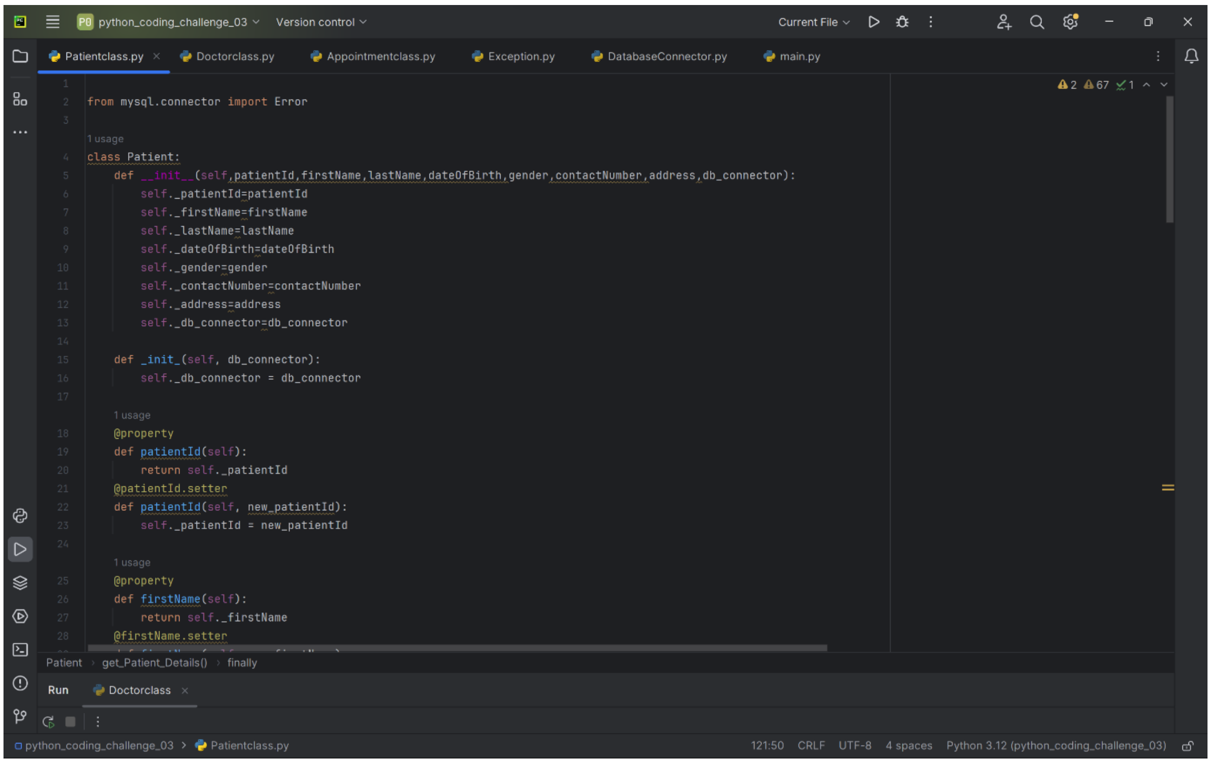
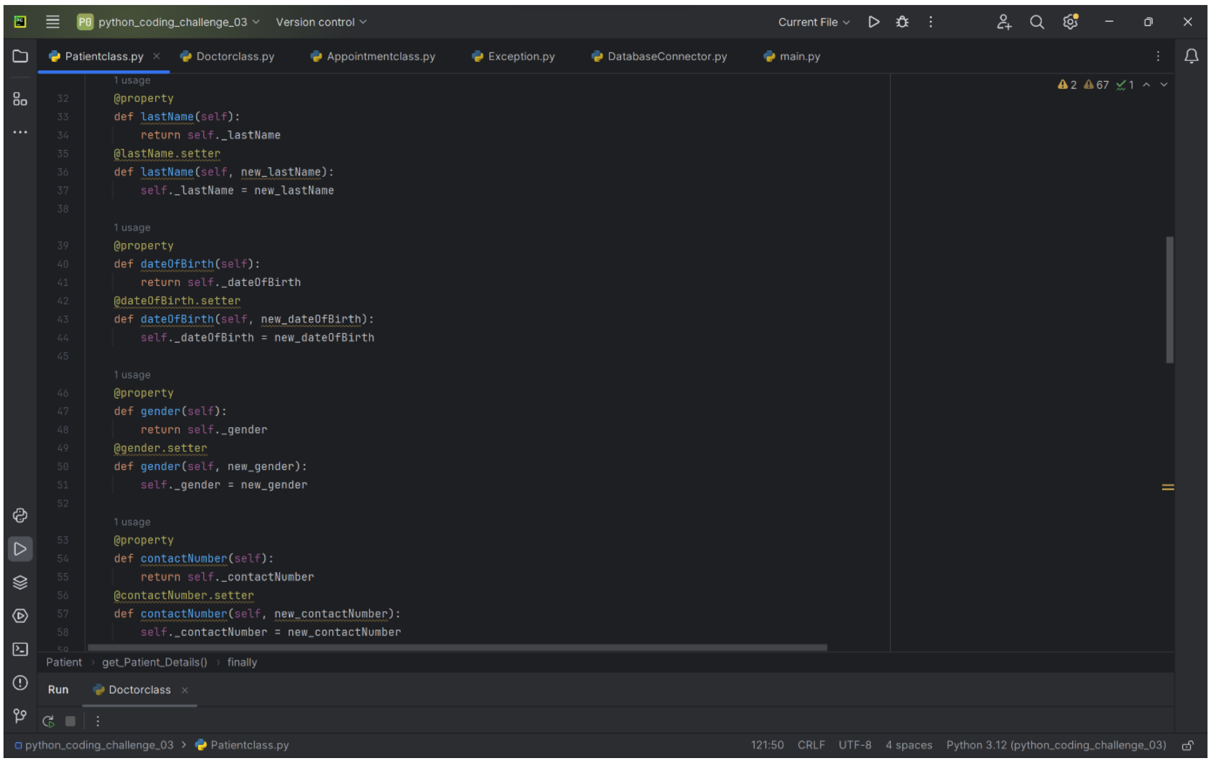
# Coding Challenge

## Patient class

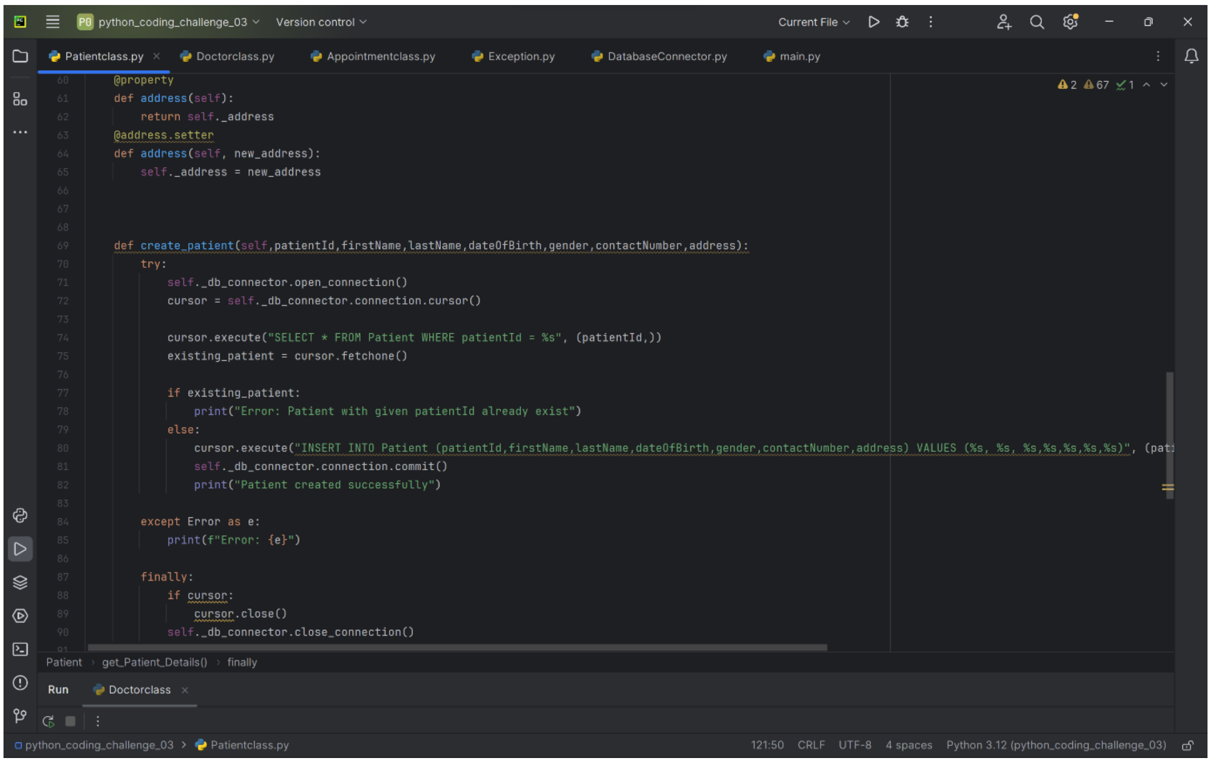


```python

from mysql.connector import Error

1 usage
class Patient:
    def __init__(self,patientId,firstName,lastName,dateOfBirth,gender,contactNumber,address,db_connector):
        self._patientId=patientId
        self._firstName=firstName
        self._lastName=lastName
        self._dateOfBirth=dateOfBirth
        self._gender=gender
        self._contactNumber=contactNumber
        self._address=address
        self._db_connector=db_connector

    def _init_(self, db_connector):
        self._db_connector = db_connector

    1 usage
    @property
    def patientId(self):
        return self._patientId
    @patientId.setter
    def patientId(self, new_patientId):
        self._patientId = new_patientId

    1 usage
    @property
    def firstName(self):
        return self._firstName
    @firstName.setter
```

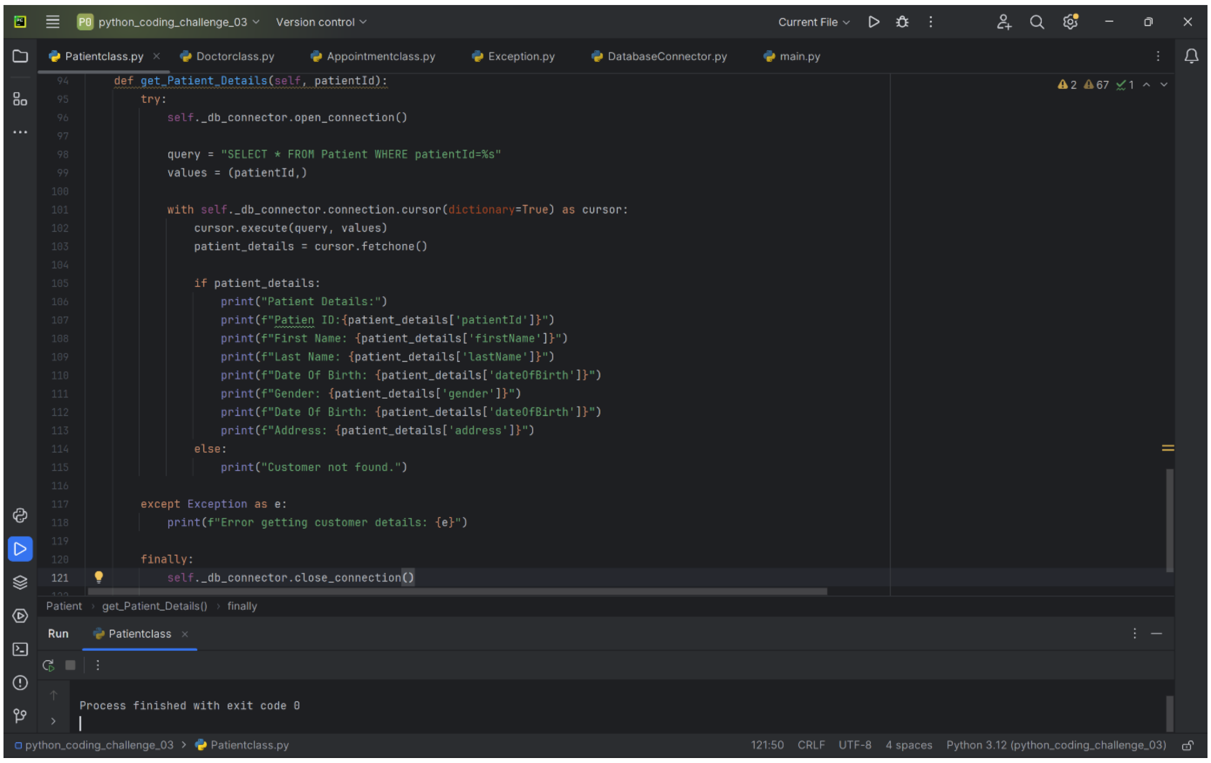
python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py
Patient › get_Patient_Details() › finally
Run
Doctorclass
python_coding_challenge_03 > Patientclass.py
121:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)


python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py
1 usage
Patient › get_Patient_Details() › finally
Run
Doctorclass
python_coding_challenge_03 > Patientclass.py
121:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

```python
    @property
    def lastName(self):
        return self._lastName
    @lastName.setter
    def lastName(self, new_lastName):
        self._lastName = new_lastName

    @property
    def dateOfBirth(self):
        return self._dateOfBirth
    @dateOfBirth.setter
    def dateOfBirth(self, new_dateOfBirth):
        self._dateOfBirth = new_dateOfBirth

    @property
    def gender(self):
        return self._gender
    @gender.setter
    def gender(self, new_gender):
        self._gender = new_gender

    @property
    def contactNumber(self):
        return self._contactNumber
    @contactNumber.setter
    def contactNumber(self, new_contactNumber):
        self._contactNumber = new_contactNumber
```


python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py
Patient › get_Patient_Details() › finally
Run
Doctorclass
python_coding_challenge_03 > Patientclass.py
121:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

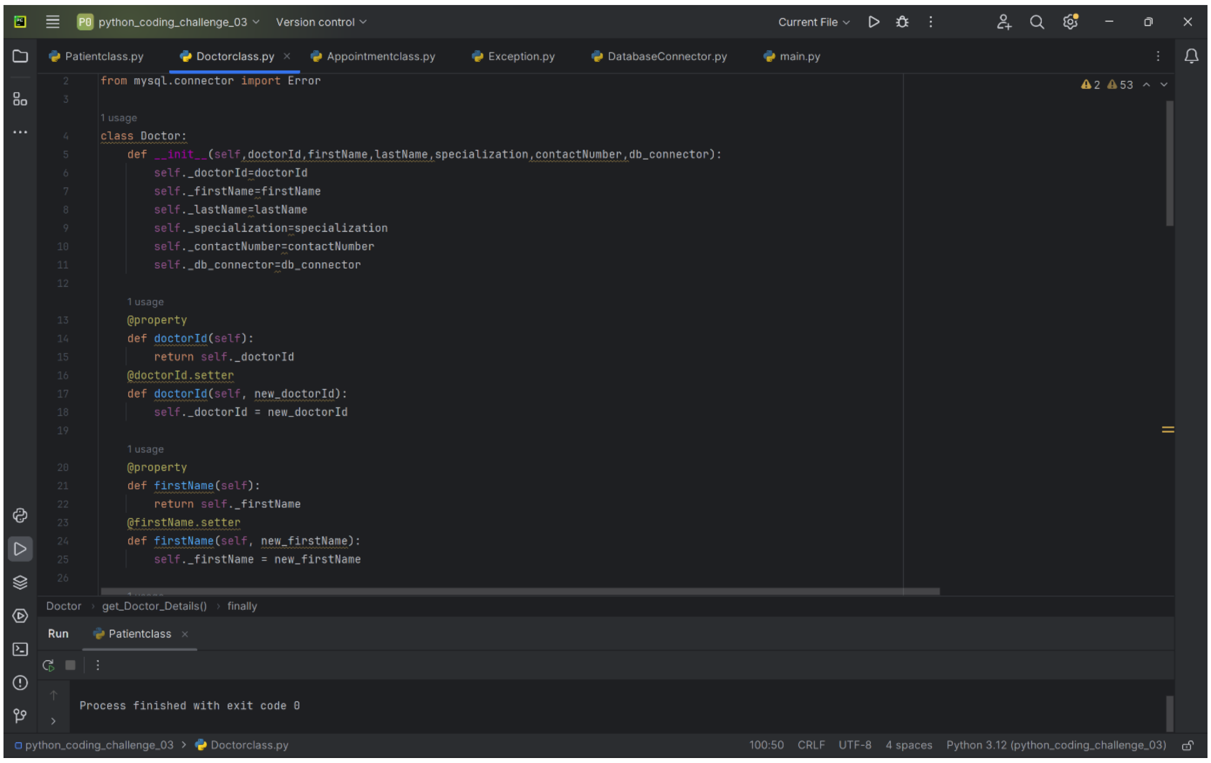
```python
    @property
    def address(self):
        return self._address
    @address.setter
    def address(self, new_address):
        self._address = new_address



    def create_patient(self,patientId,firstName,lastName,dateOfBirth,gender,contactNumber,address):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Patient WHERE patientId = %s", (patientId,))
            existing_patient = cursor.fetchone()

            if existing_patient:
                print("Error: Patient with given patientId already exist")
            else:
                cursor.execute("INSERT INTO Patient (patientId,firstName,lastName,dateOfBirth,gender,contactNumber,address) VALUES (%s, %s, %s,%s,%s,%s,%s)", (pati
                self._db_connector.connection.commit()
                print("Patient created successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()
```



```
    def get_Patient_Details(self, patientId):
        try:
            self._db_connector.open_connection()

            query = "SELECT * FROM Patient WHERE patientId=%s"
            values = (patientId,)

            with self._db_connector.connection.cursor(dictionary=True) as cursor:
                cursor.execute(query, values)
                patient_details = cursor.fetchone()

                if patient_details:
                    print("Patient Details:")
                    print(f"Patien ID:{patient_details['patientId']}")
                    print(f"First Name: {patient_details['firstName']}")
                    print(f"Last Name: {patient_details['lastName']}")
                    print(f"Date Of Birth: {patient_details['dateOfBirth']}")
                    print(f"Gender: {patient_details['gender']}")
                    print(f"Date Of Birth: {patient_details['dateOfBirth']}")
                    print(f"Address: {patient_details['address']}")
                else:
                    print("Customer not found.")

        except Exception as e:
            print(f"Error getting customer details: {e}")

        finally:
            self._db_connector.close_connection()
```

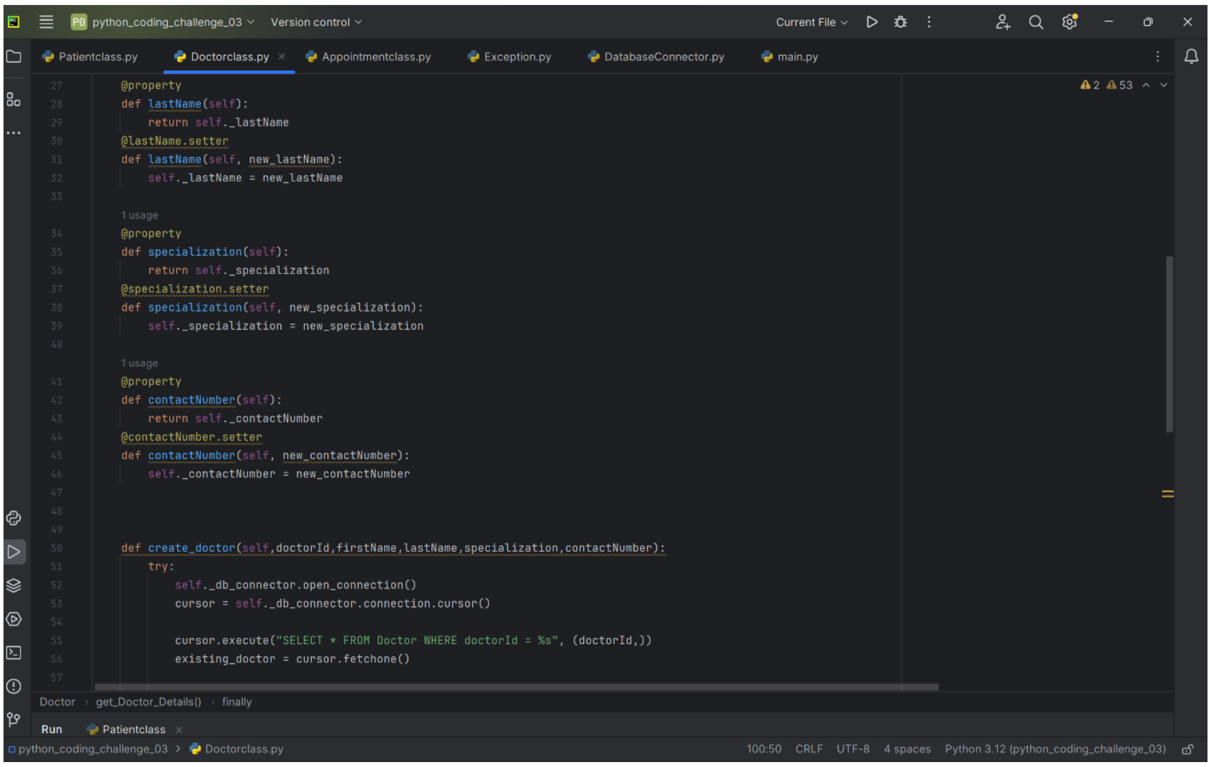
```
Process finished with exit code 0
```

## Doctor class

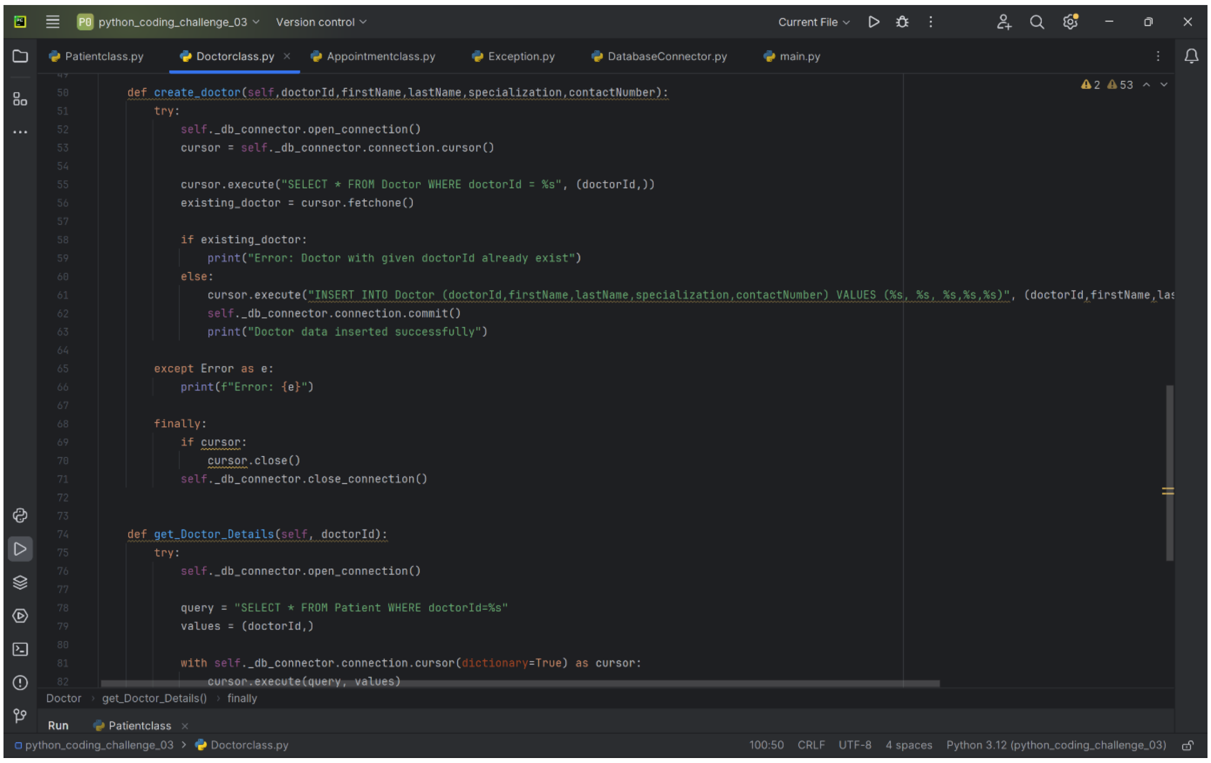


```
from mysql.connector import Error

1 usage
class Doctor:
    def __init__(self,doctorId,firstName,lastName,specialization,contactNumber,db_connector):
        self._doctorId=doctorId
        self._firstName=firstName
        self._lastName=lastName
        self._specialization=specialization
        self._contactNumber=contactNumber
        self._db_connector=db_connector

    1 usage
    @property
    def doctorId(self):
        return self._doctorId
    @doctorId.setter
    def doctorId(self, new_doctorId):
        self._doctorId = new_doctorId

    1 usage
    @property
    def firstName(self):
        return self._firstName
    @firstName.setter
    def firstName(self, new_firstName):
        self._firstName = new_firstName

```

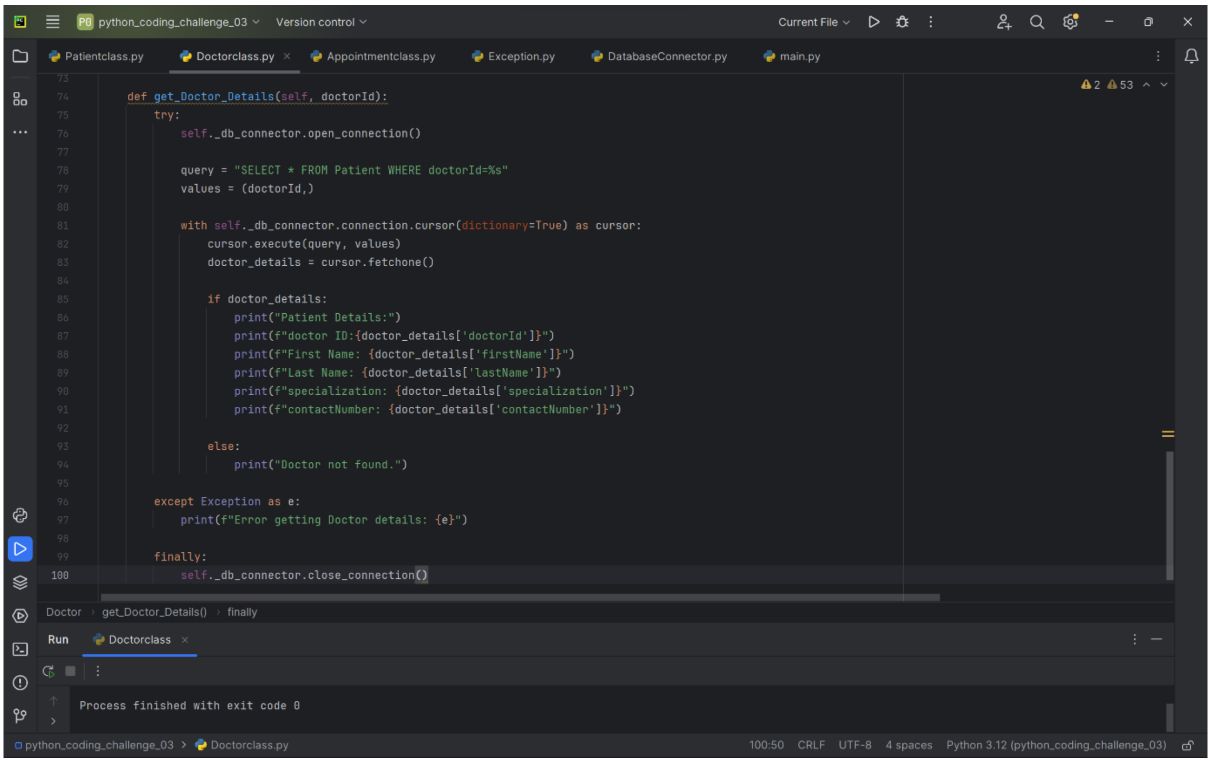
```
Process finished with exit code 0
```



```
    @property
    def lastName(self):
        return self._lastName
    @lastName.setter
    def lastName(self, new_lastName):
        self._lastName = new_lastName

    1 usage
    @property
    def specialization(self):
        return self._specialization
    @specialization.setter
    def specialization(self, new_specialization):
        self._specialization = new_specialization

    1 usage
    @property
    def contactNumber(self):
        return self._contactNumber
    @contactNumber.setter
    def contactNumber(self, new_contactNumber):
        self._contactNumber = new_contactNumber



    def create_doctor(self,doctorId,firstName,lastName,specialization,contactNumber):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Doctor WHERE doctorId = %s", (doctorId,))
            existing_doctor = cursor.fetchone()

```


python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py
Doctor › get_Doctor_Details() › finally
Run
Patientclass
python_coding_challenge_03 > Doctorclass.py
100:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

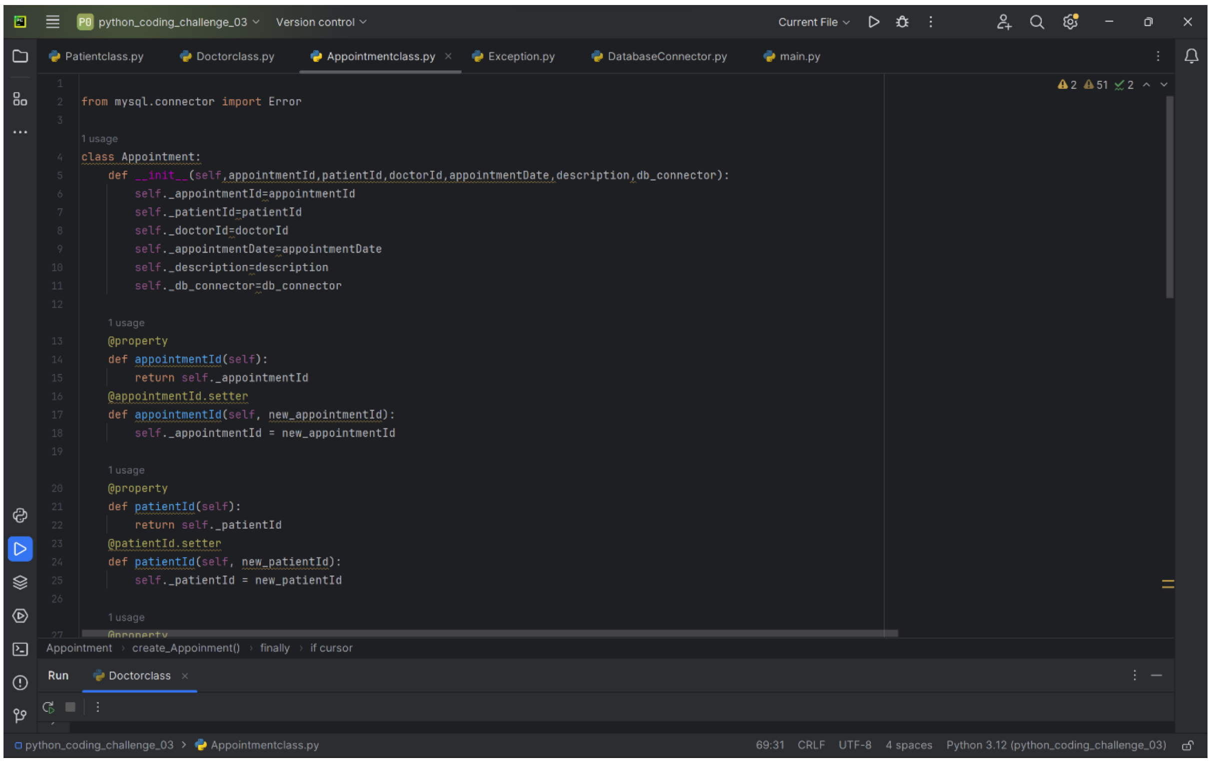
```python
    def create_doctor(self,doctorId,firstName,lastName,specialization,contactNumber):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Doctor WHERE doctorId = %s", (doctorId,))
            existing_doctor = cursor.fetchone()

            if existing_doctor:
                print("Error: Doctor with given doctorId already exist")
            else:
                cursor.execute("INSERT INTO Doctor (doctorId,firstName,lastName,specialization,contactNumber) VALUES (%s, %s, %s,%s,%s)", (doctorId,firstName,las
                self._db_connector.connection.commit()
                print("Doctor data inserted successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()


    def get_Doctor_Details(self, doctorId):
        try:
            self._db_connector.open_connection()

            query = "SELECT * FROM Patient WHERE doctorId=%s"
            values = (doctorId,)

            with self._db_connector.connection.cursor(dictionary=True) as cursor:
                cursor.execute(query, values)
```


python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py
Doctor › get_Doctor_Details() › finally
Run
Doctorclass
python_coding_challenge_03 > Doctorclass.py
100:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

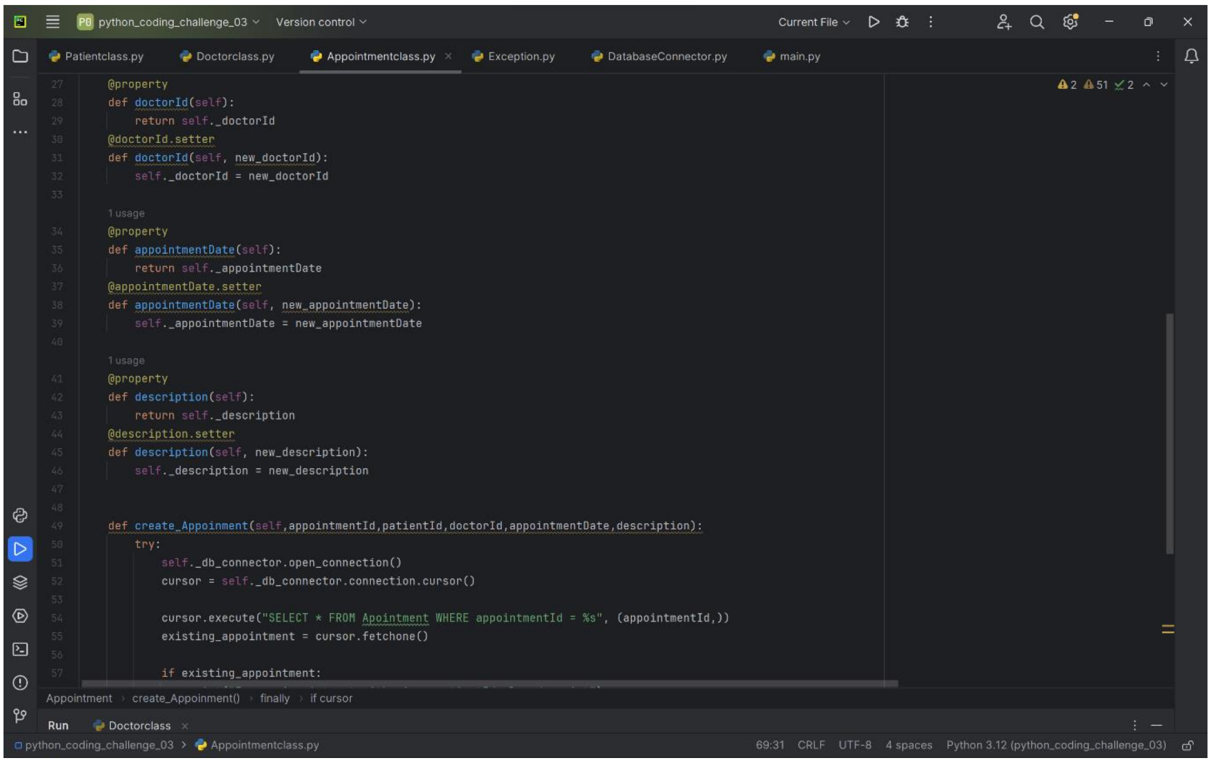
```python
    def get_Doctor_Details(self, doctorId):
        try:
            self._db_connector.open_connection()

            query = "SELECT * FROM Patient WHERE doctorId=%s"
            values = (doctorId,)

            with self._db_connector.connection.cursor(dictionary=True) as cursor:
                cursor.execute(query, values)
                doctor_details = cursor.fetchone()

                if doctor_details:
                    print("Patient Details:")
                    print(f"doctor ID:{doctor_details['doctorId']}")
                    print(f"First Name: {doctor_details['firstName']}")
                    print(f"Last Name: {doctor_details['lastName']}")
                    print(f"specialization: {doctor_details['specialization']}")
                    print(f"contactNumber: {doctor_details['contactNumber']}")

                else:
                    print("Doctor not found.")

        except Exception as e:
            print(f"Error getting Doctor details: {e}")

        finally:
            self._db_connector.close_connection()
```

```
Process finished with exit code 0
```

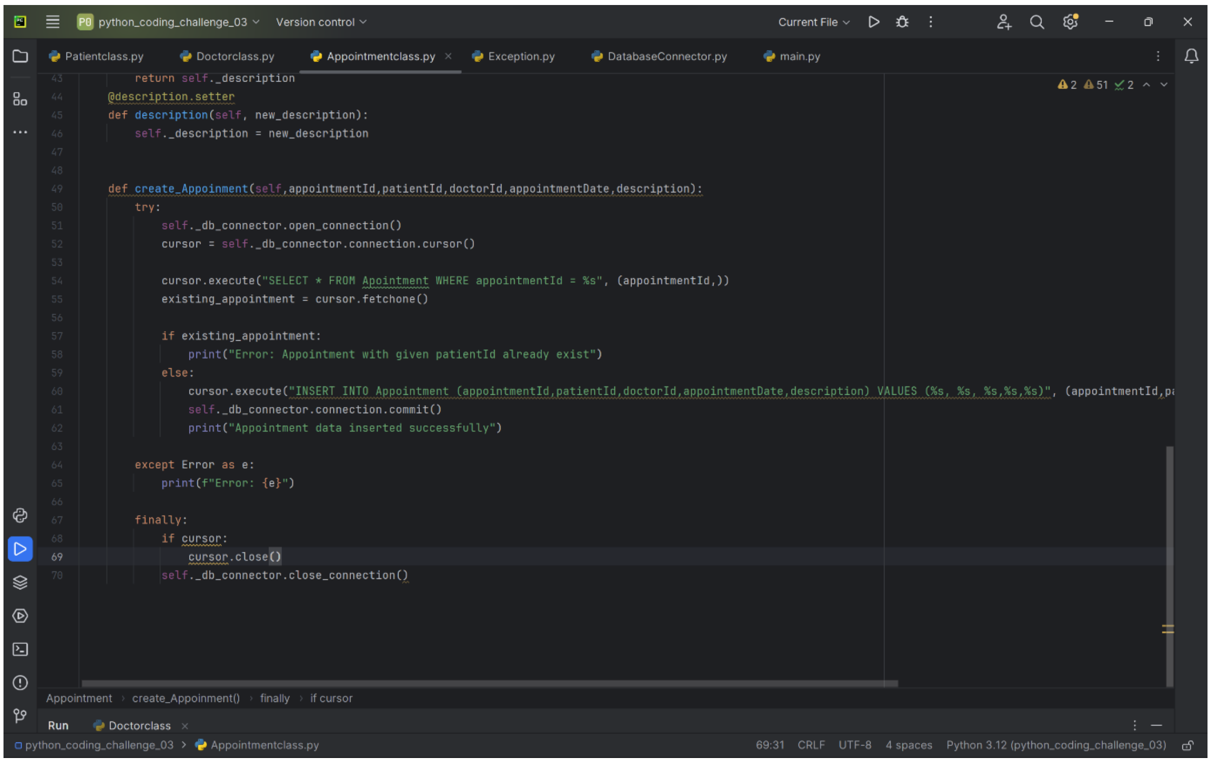
# Appointment class



```python

from mysql.connector import Error

class Appointment:
    def __init__(self,appointmentId,patientId,doctorId,appointmentDate,description,db_connector):
        self._appointmentId=appointmentId
        self._patientId=patientId
        self._doctorId=doctorId
        self._appointmentDate=appointmentDate
        self._description=description
        self._db_connector=db_connector

    @property
    def appointmentId(self):
        return self._appointmentId
    @appointmentId.setter
    def appointmentId(self, new_appointmentId):
        self._appointmentId = new_appointmentId

    @property
    def patientId(self):
        return self._patientId
    @patientId.setter
    def patientId(self, new_patientId):
        self._patientId = new_patientId

```

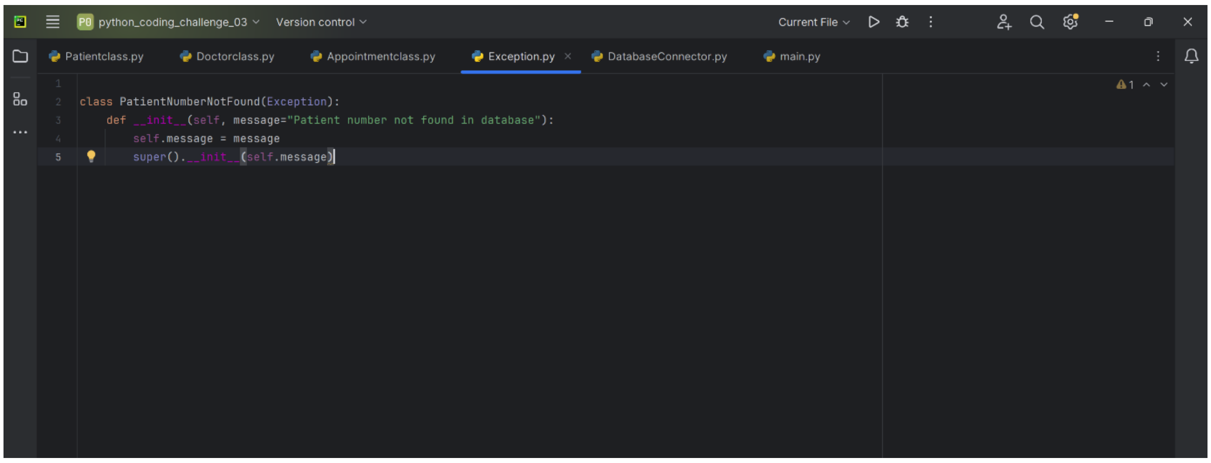


```python
    @property
    def doctorId(self):
        return self._doctorId
    @doctorId.setter
    def doctorId(self, new_doctorId):
        self._doctorId = new_doctorId

    @property
    def appointmentDate(self):
        return self._appointmentDate
    @appointmentDate.setter
    def appointmentDate(self, new_appointmentDate):
        self._appointmentDate = new_appointmentDate

    @property
    def description(self):
        return self._description
    @description.setter
    def description(self, new_description):
        self._description = new_description


    def create_Appoinment(self,appointmentId,patientId,doctorId,appointmentDate,description):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Apointment WHERE appointmentId = %s", (appointmentId,))
            existing_appointment = cursor.fetchone()

            if existing_appointment:
```



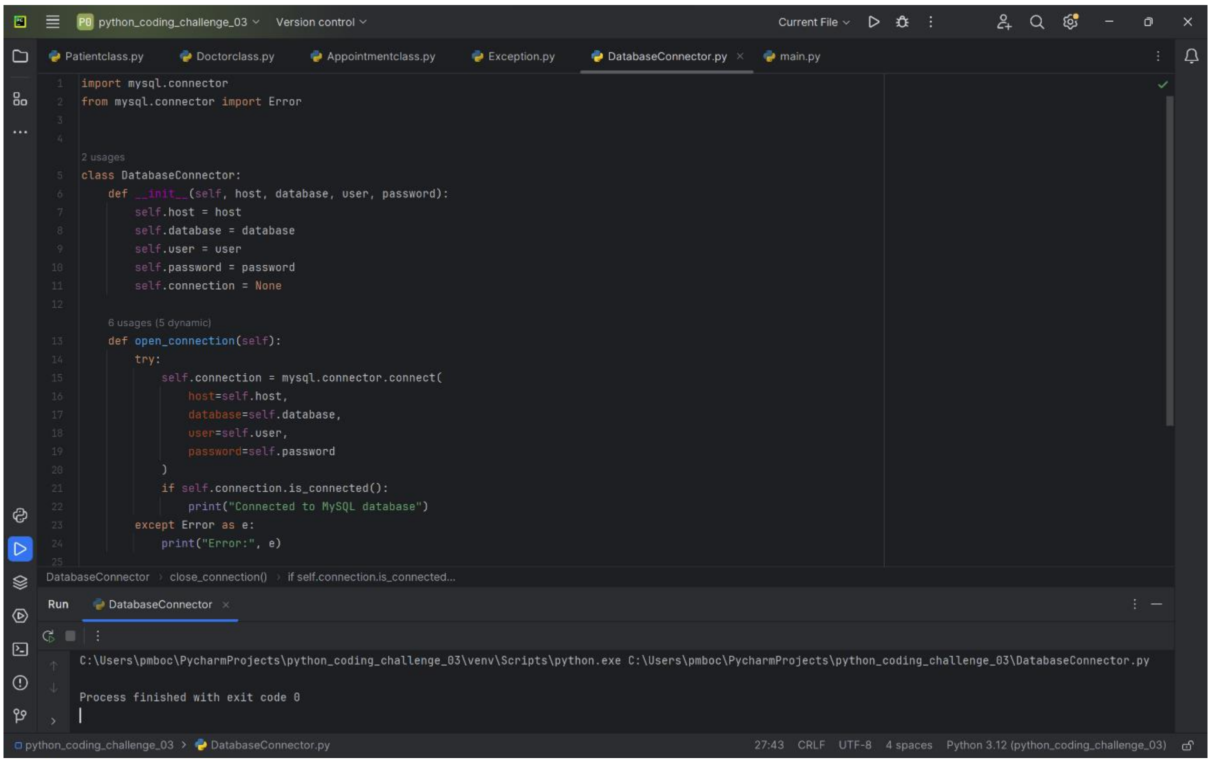
```python
        return self._description
    @description.setter
    def description(self, new_description):
        self._description = new_description


    def create_Appoinment(self,appointmentId,patientId,doctorId,appointmentDate,description):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Apointment WHERE appointmentId = %s", (appointmentId,))
            existing_appointment = cursor.fetchone()

            if existing_appointment:
                print("Error: Appointment with given patientId already exist")
            else:
                cursor.execute("INSERT INTO Appointment (appointmentId,patientId,doctorId,appointmentDate,description) VALUES (%s, %s, %s,%s,%s)", (appointmentId,pa
                self._db_connector.connection.commit()
                print("Appointment data inserted successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()
```

## Exception



```python
class PatientNumberNotFound(Exception):
    def __init__(self, message="Patient number not found in database"):
        self.message = message
        super().__init__(self.message)
```

# Database Connection

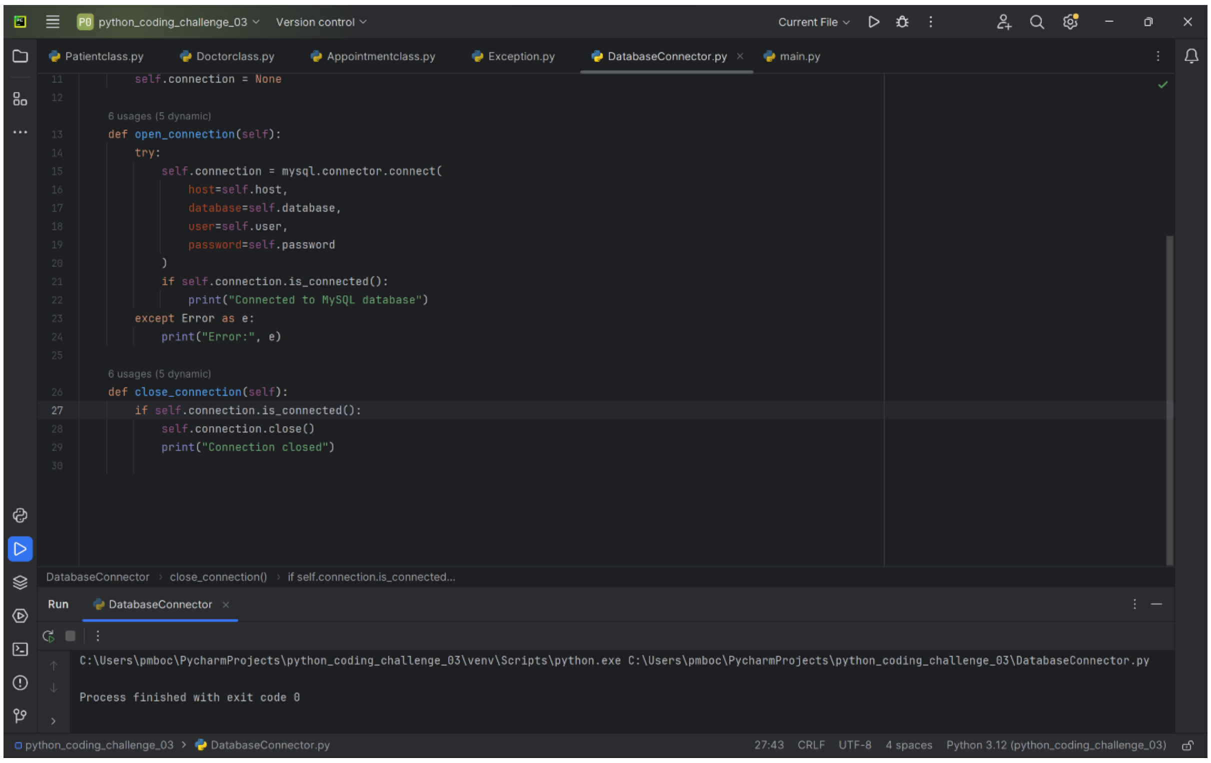


```python
import mysql.connector
from mysql.connector import Error


2 usages
class DatabaseConnector:
    def __init__(self, host, database, user, password):
        self.host = host
        self.database = database
        self.user = user
        self.password = password
        self.connection = None

    6 usages (5 dynamic)
    def open_connection(self):
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                database=self.database,
                user=self.user,
                password=self.password
            )
            if self.connection.is_connected():
                print("Connected to MySQL database")
        except Error as e:
            print("Error:", e)
```

DatabaseConnector › close_connection() › if self.connection.is_connected...
Run
DatabaseConnector

```
C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\venv\Scripts\python.exe C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\DatabaseConnector.py

Process finished with exit code 0
```

python_coding_challenge_03 > DatabaseConnector.py
27:43 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)



python_coding_challenge_03
Version control
Current File
Patientclass.py
Doctorclass.py
Appointmentclass.py
Exception.py
DatabaseConnector.py
main.py

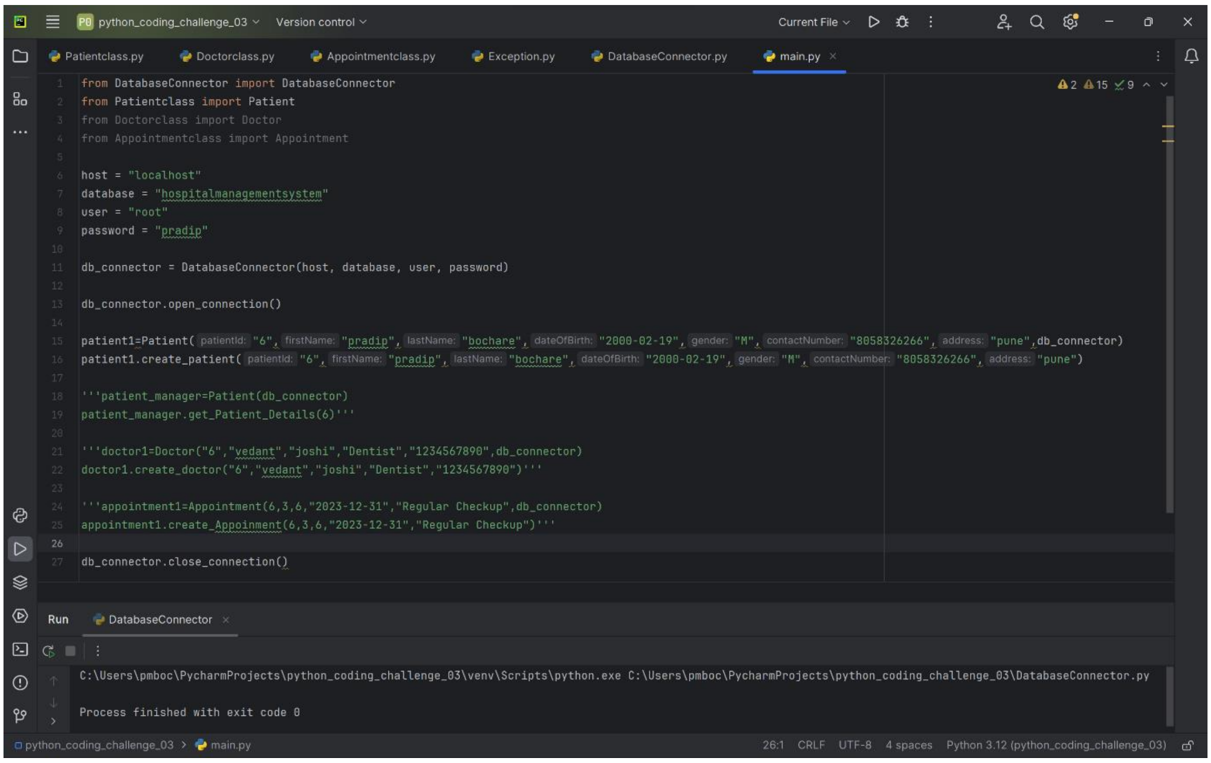
```python
        self.connection = None

    6 usages (5 dynamic)
    def open_connection(self):
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                database=self.database,
                user=self.user,
                password=self.password
            )
            if self.connection.is_connected():
                print("Connected to MySQL database")
        except Error as e:
            print("Error:", e)

    6 usages (5 dynamic)
    def close_connection(self):
        if self.connection.is_connected():
            self.connection.close()
            print("Connection closed")
```

DatabaseConnector › close_connection() › if self.connection.is_connected...
Run
DatabaseConnector

```
C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\venv\Scripts\python.exe C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\DatabaseConnector.py

Process finished with exit code 0
```

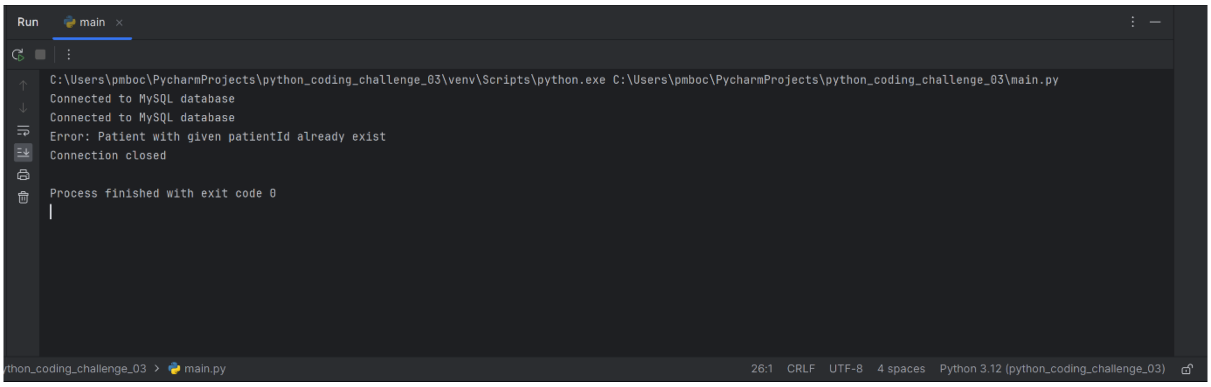
python_coding_challenge_03 > DatabaseConnector.py
27:43 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

# Main File

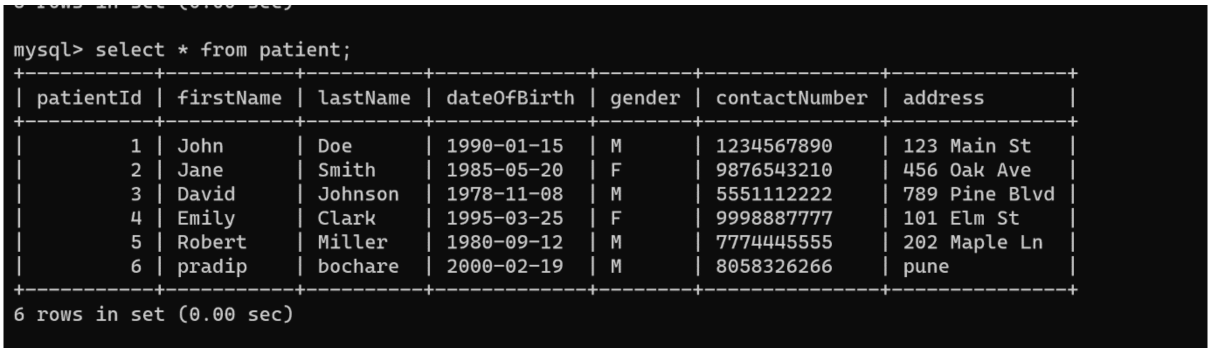


```python
from DatabaseConnector import DatabaseConnector
from Patientclass import Patient
from Doctorclass import Doctor
from Appointmentclass import Appointment

host = "localhost"
database = "hospitalmanagementsystem"
user = "root"
password = "pradip"

db_connector = DatabaseConnector(host, database, user, password)

db_connector.open_connection()

patient1=Patient( patientId: "6", firstName: "pradip", lastName: "bochare", dateOfBirth: "2000-02-19", gender: "M", contactNumber: "8058326266", address: "pune",db_connector)
patient1.create_patient( patientId: "6", firstName: "pradip", lastName: "bochare", dateOfBirth: "2000-02-19", gender: "M", contactNumber: "8058326266", address: "pune")

'''patient_manager=Patient(db_connector)
patient_manager.get_Patient_Details(6)'''

'''doctor1=Doctor("6","vedant","joshi","Dentist","1234567890",db_connector)
doctor1.create_doctor("6","vedant","joshi","Dentist","1234567890")'''

'''appointment1=Appointment(6,3,6,"2023-12-31","Regular Checkup",db_connector)
appointment1.create_Appoinment(6,3,6,"2023-12-31","Regular Checkup")'''

db_connector.close_connection()
```

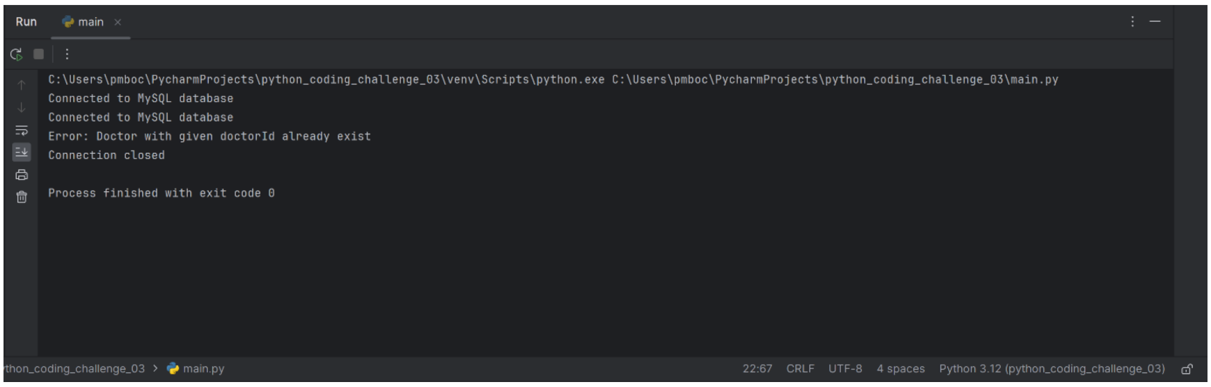
```
C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\venv\Scripts\python.exe C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\DatabaseConnector.py

Process finished with exit code 0
```



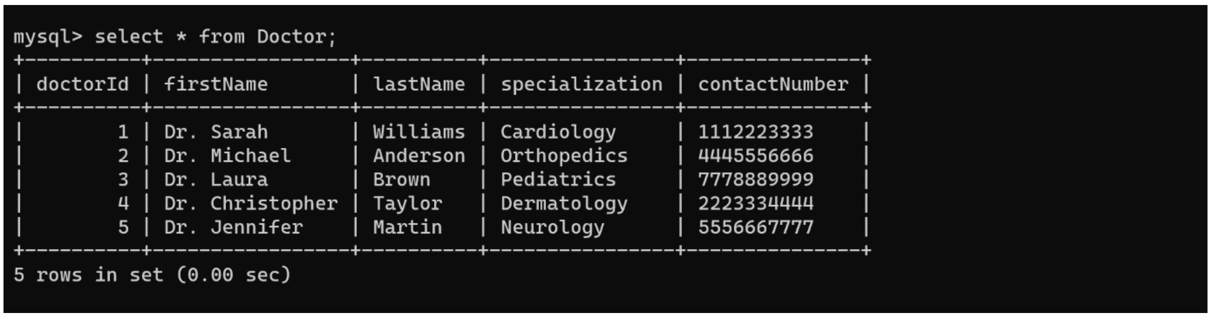
```
C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\venv\Scripts\python.exe C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\main.py
Connected to MySQL database
Connected to MySQL database
Error: Patient with given patientId already exist
Connection closed

Process finished with exit code 0
```



```
mysql> select * from patient;
+-----------+-----------+----------+-------------+--------+---------------+----------------+
| patientId | firstName | lastName | dateOfBirth | gender | contactNumber | address        |
+-----------+-----------+----------+-------------+--------+---------------+----------------+
|         1 | John      | Doe      | 1990-01-15  | M      | 1234567890    | 123 Main St    |
|         2 | Jane      | Smith    | 1985-05-20  | F      | 9876543210    | 456 Oak Ave    |
|         3 | David     | Johnson  | 1978-11-08  | M      | 5551112222    | 789 Pine Blvd  |
|         4 | Emily     | Clark    | 1995-03-25  | F      | 9998887777    | 101 Elm St     |
|         5 | Robert    | Miller   | 1980-09-12  | M      | 7774445555    | 202 Maple Ln   |
|         6 | pradip    | bochare  | 2000-02-19  | M      | 8058326266    | pune           |
+-----------+-----------+----------+-------------+--------+---------------+----------------+
6 rows in set (0.00 sec)
```


Run
main
thon_coding_challenge_03 > main.py
22:67 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

```
C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\venv\Scripts\python.exe C:\Users\pmboc\PycharmProjects\python_coding_challenge_03\main.py
Connected to MySQL database
Connected to MySQL database
Error: Doctor with given doctorId already exist
Connection closed

Process finished with exit code 0
```



```
mysql> select * from Doctor;
+----------+-----------------+----------+----------------+---------------+
| doctorId | firstName       | lastName | specialization | contactNumber |
+----------+-----------------+----------+----------------+---------------+
|        1 | Dr. Sarah       | Williams | Cardiology     | 1112223333    |
|        2 | Dr. Michael     | Anderson | Orthopedics    | 4445556666    |
|        3 | Dr. Laura       | Brown    | Pediatrics     | 7778889999    |
|        4 | Dr. Christopher | Taylor   | Dermatology    | 2223334444    |
|        5 | Dr. Jennifer    | Martin   | Neurology      | 5556667777    |
+----------+-----------------+----------+----------------+---------------+
5 rows in set (0.00 sec)
```

Patient Class

```
from mysql.connector import Error

class Patient:
    def
__init__(self,patientId,firstName,lastName,dateOfBirth,gender,contactNumber
,address,db_connector):
        self._patientId=patientId
        self._firstName=firstName
        self._lastName=lastName
        self._dateOfBirth=dateOfBirth
        self._gender=gender
        self._contactNumber=contactNumber
        self._address=address
        self._db_connector=db_connector

    def __init__(self, db_connector):
        self._db_connector = db_connector

    @property
    def patientId(self):
        return self._patientId
    @patientId.setter
    def patientId(self, new_patientId):
        self._patientId = new_patientId

    @property
    def firstName(self):
        return self._firstName
    @firstName.setter
    def firstName(self, new_firstName):
        self._firstName = new_firstName

    @property
    def lastName(self):
        return self._lastName
    @lastName.setter
    def lastName(self, new_lastName):
        self._lastName = new_lastName

    @property
    def dateOfBirth(self):
        return self._dateOfBirth
    @dateOfBirth.setter
    def dateOfBirth(self, new_dateOfBirth):
        self._dateOfBirth = new_dateOfBirth

    @property
    def gender(self):
        return self._gender
    @gender.setter
    def gender(self, new_gender):
        self._gender = new_gender

    @property
    def contactNumber(self):
        return self._contactNumber
    @contactNumber.setter
    def contactNumber(self, new_contactNumber):
        self._contactNumber = new_contactNumber

    @property
    def address(self):
        return self._address
    @address.setter
    def address(self, new_address):
        self._address = new_address



    def
create_patient(self,patientId,firstName,lastName,dateOfBirth,gender,contact
Number,address):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Patient WHERE patientId = %s",
(patientId,))
            existing_patient = cursor.fetchone()

            if existing_patient:
                print("Error: Patient with given patientId already exist")
            else:
                cursor.execute("INSERT INTO Patient
(patientId,firstName,lastName,dateOfBirth,gender,contactNumber,address)
VALUES (%s, %s, %s,%s,%s,%s,%s)",
(patientId,firstName,lastName,dateOfBirth,gender,contactNumber,address))
                self._db_connector.connection.commit()
                print("Patient created successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()



    def get_Patient_Details(self, patientId):
        try:
            self._db_connector.open_connection()

            query = "SELECT * FROM Patient WHERE patientId=%s"
            values = (patientId,)

            with self._db_connector.connection.cursor(dictionary=True) as
cursor:
                cursor.execute(query, values)
                patient_details = cursor.fetchone()

                if patient_details:
                    print("Patient Details:")
                    print(f"Patien ID:{patient_details['patientId']}")
                    print(f"First Name: {patient_details['firstName']}")
                    print(f"Last Name: {patient_details['lastName']}")
                    print(f"Date Of Birth:
{patient_details['dateOfBirth']}")
                    print(f"Gender: {patient_details['gender']}")
                    print(f"Date Of Birth:
{patient_details['dateOfBirth']}")
                    print(f"Address: {patient_details['address']}")
                else:
                    print("Customer not found.")

        except Exception as e:
            print(f"Error getting customer details: {e}")

        finally:


            self._db_connector.close_connection()
```

Doctor class

```
from mysql.connector import Error

class Doctor:
    def
__init__(self,doctorId,firstName,lastName,specialization,contactNumber,db_c
onnector):
        self._doctorId=doctorId
        self._firstName=firstName
        self._lastName=lastName
        self._specialization=specialization
        self._contactNumber=contactNumber
        self._db_connector=db_connector

    @property
    def doctorId(self):
        return self._doctorId
    @doctorId.setter
    def doctorId(self, new_doctorId):
        self._doctorId = new_doctorId

    @property
    def firstName(self):
        return self._firstName
    @firstName.setter
    def firstName(self, new_firstName):
        self._firstName = new_firstName

    @property
    def lastName(self):
        return self._lastName
    @lastName.setter
    def lastName(self, new_lastName):
        self._lastName = new_lastName

    @property
    def specialization(self):
        return self._specialization
    @specialization.setter
    def specialization(self, new_specialization):
        self._specialization = new_specialization

    @property
    def contactNumber(self):
        return self._contactNumber
    @contactNumber.setter
    def contactNumber(self, new_contactNumber):
        self._contactNumber = new_contactNumber



    def
create_doctor(self,doctorId,firstName,lastName,specialization,contactNumber
):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()

            cursor.execute("SELECT * FROM Doctor WHERE doctorId = %s",
(doctorId,))
            existing_doctor = cursor.fetchone()

            if existing_doctor:
                print("Error: Doctor with given doctorId already exist")
            else:
                cursor.execute("INSERT INTO Doctor
(doctorId,firstName,lastName,specialization,contactNumber) VALUES (%s, %s,
%s,%s,%s)", (doctorId,firstName,lastName,specialization,contactNumber))
                self._db_connector.connection.commit()
                print("Doctor data inserted successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()


    def get_Doctor_Details(self, doctorId):
        try:
            self._db_connector.open_connection()

            query = "SELECT * FROM Patient WHERE doctorId=%s"
            values = (doctorId,)

            with self._db_connector.connection.cursor(dictionary=True) as
cursor:
                cursor.execute(query, values)
                doctor_details = cursor.fetchone()

                if doctor_details:
                    print("Patient Details:")
                    print(f"doctor ID:{doctor_details['doctorId']}")
                    print(f"First Name: {doctor_details['firstName']}")
                    print(f"Last Name: {doctor_details['lastName']}")
                    print(f"specialization:
{doctor_details['specialization']}")
                    print(f"contactNumber:
{doctor_details['contactNumber']}")

                else:
                    print("Doctor not found.")

        except Exception as e:
            print(f"Error getting Doctor details: {e}")

        finally:
            self._db_connector.close_connection()
```

Appointment class

```
from mysql.connector import Error

class Appointment:
    def
__init__(self,appointmentId,patientId,doctorId,appointmentDate,description,
db_connector):
        self._appointmentId=appointmentId
        self._patientId=patientId
        self._doctorId=doctorId
        self._appointmentDate=appointmentDate
        self._description=description
        self._db_connector=db_connector

    @property
    def appointmentId(self):
        return self._appointmentId
    @appointmentId.setter
    def appointmentId(self, new_appointmentId):
        self._appointmentId = new_appointmentId

    @property
    def patientId(self):
        return self._patientId
    @patientId.setter
    def patientId(self, new_patientId):
        self._patientId = new_patientId

    @property
    def doctorId(self):
        return self._doctorId
    @doctorId.setter
    def doctorId(self, new_doctorId):
        self._doctorId = new_doctorId

    @property
    def appointmentDate(self):
        return self._appointmentDate
    @appointmentDate.setter
    def appointmentDate(self, new_appointmentDate):
        self._appointmentDate = new_appointmentDate

    @property
    def description(self):
        return self._description
    @description.setter
    def description(self, new_description):
        self._description = new_description


    def
create_Appoinment(self,appointmentId,patientId,doctorId,appointmentDate,des
cription):
        try:
            self._db_connector.open_connection()
            cursor = self._db_connector.connection.cursor()
            cursor.execute("SELECT * FROM Apointment WHERE appointmentId =
%s", (appointmentId,))
            existing_appointment = cursor.fetchone()

            if existing_appointment:
                print("Error: Appointment with given patientId already
exist")
            else:
                cursor.execute("INSERT INTO Appointment
(appointmentId,patientId,doctorId,appointmentDate,description) VALUES (%s,
%s, %s,%s,%s)",
(appointmentId,patientId,doctorId,appointmentDate,description))
                self._db_connector.connection.commit()
                print("Appointment data inserted successfully")

        except Error as e:
            print(f"Error: {e}")

        finally:
            if cursor:
                cursor.close()
            self._db_connector.close_connection()
```

Database Connector

```
import mysql.connector
from mysql.connector import Error


class DatabaseConnector:
    def __init__(self, host, database, user, password):
        self.host = host
        self.database = database
        self.user = user
        self.password = password
        self.connection = None

    def open_connection(self):
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                database=self.database,
                user=self.user,
                password=self.password
            )
            if self.connection.is_connected():
                print("Connected to MySQL database")
        except Error as e:
            print("Error:", e)

    def close_connection(self):
        if self.connection.is_connected():
            self.connection.close()
            print("Connection closed")
```

Main file

```python
from DatabaseConnector import DatabaseConnector
from Patientclass import Patient
from Doctorclass import Doctor
from Appointmentclass import Appointment

host = "localhost"
database = "hospitalmanagementsystem"
user = "root"
password = "pradip"

db_connector = DatabaseConnector(host, database, user, password)

db_connector.open_connection()

'''patient1=Patient("6","pradip","bochare","2000-02-19","M","8058326266","pune",db_connector)
patient1.create_patient("6","pradip","bochare","2000-02-19","M","8058326266","pune")'''

'''patient_manager=Patient(db_connector)
patient_manager.get_Patient_Details(6)'''

'''doctor1=Doctor("6","vedant","joshi","Dentist","1234567890",db_connector)
doctor1.create_doctor("6","vedant","joshi","Dentist","1234567890")'''

appointment1=Appointment(6,3,6,"2023-12-31","Regular Checkup",db_connector)
appointment1.create_Appoinment(6,3,6,"2023-12-31","Regular Checkup")

db_connector.close_connection()
```