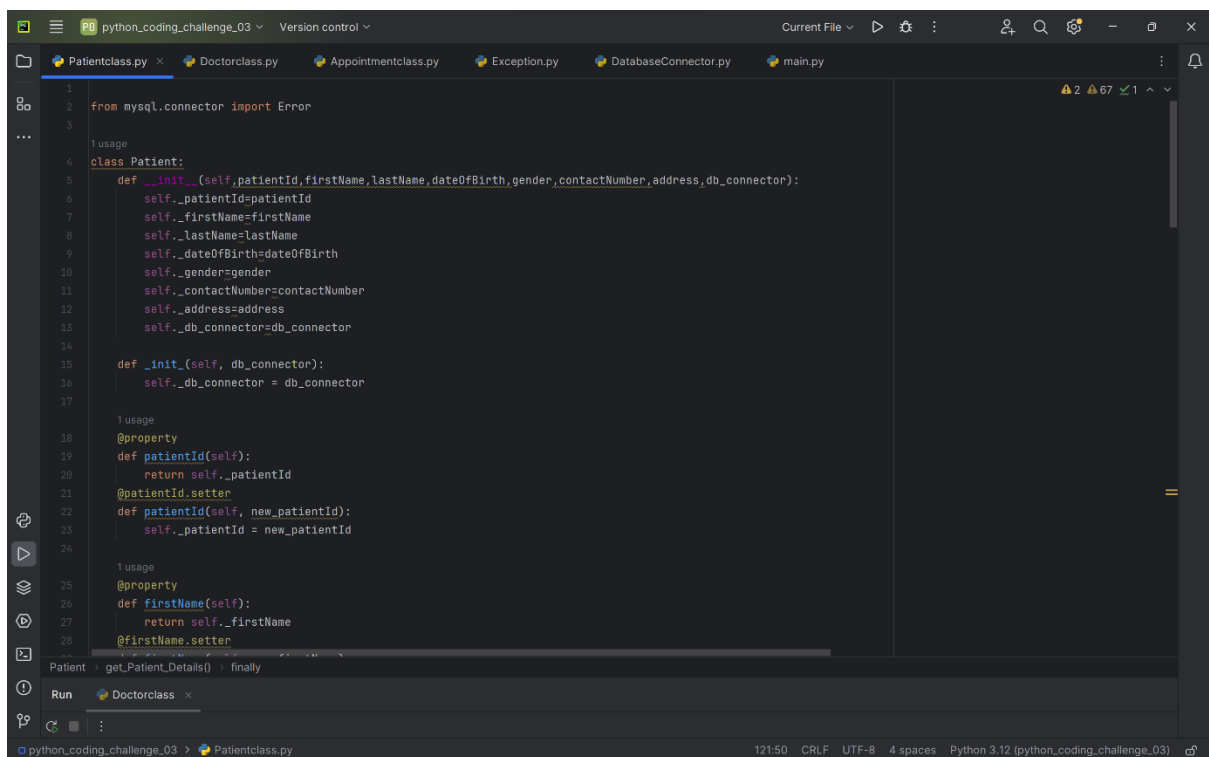


Coding Challenge

Patient class



```
1
2 from mysql.connector import Error
3
4 1 usage
5 class Patient:
6     def __init__(self, patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address, db_connector):
7         self._patientId = patientId
8         self._firstName = firstName
9         self._lastName = lastName
10        self._dateOfBirth = dateOfBirth
11        self._gender = gender
12        self._contactNumber = contactNumber
13        self._address = address
14        self._db_connector = db_connector
15
16    def __init__(self, db_connector):
17        self._db_connector = db_connector
18
19    1 usage
20    @property
21    def patientId(self):
22        return self._patientId
23    @patientId.setter
24    def patientId(self, new_patientId):
25        self._patientId = new_patientId
26
27    1 usage
28    @property
29    def firstName(self):
30        return self._firstName
31    @firstName.setter
32    def firstName(self, new_firstName):
33        self._firstName = new_firstName
34
35    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)

```
1 usage
32 @property
33 def lastName(self):
34     return self._lastName
35 @lastName.setter
36 def lastName(self, new_lastName):
37     self._lastName = new_lastName
38
39 1 usage
40 @property
41 def dateOfBirth(self):
42     return self._dateOfBirth
43 @dateOfBirth.setter
44 def dateOfBirth(self, new_dateOfBirth):
45     self._dateOfBirth = new_dateOfBirth
46
47 1 usage
48 @property
49 def gender(self):
50     return self._gender
51 @gender.setter
52 def gender(self, new_gender):
53     self._gender = new_gender
54
55 1 usage
56 @property
57 def contactNumber(self):
58     return self._contactNumber
59 @contactNumber.setter
60 def contactNumber(self, new_contactNumber):
61     self._contactNumber = new_contactNumber
62
63 Patient > get_Patient_Details() > finally
64 Run Doctorclass x
65 python_coding_challenge_03 > Patientclass.py
```

```
60 @property
61 def address(self):
62     return self._address
63 @address.setter
64 def address(self, new_address):
65     self._address = new_address
66
67
68
69 def create_patient(self, patientId, firstName, lastName, dateOfBirth, gender, contactNumber, address):
70     try:
71         self._db_connector.open_connection()
72         cursor = self._db_connector.connection.cursor()
73
74         cursor.execute("SELECT * FROM Patient WHERE patientId = %s", (patientId,))
75         existing_patient = cursor.fetchone()
76
77         if existing_patient:
78             print("Error: Patient with given patientId already exist")
79         else:
80             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))
81             self._db_connector.connection.commit()
82             print("Patient created successfully")
83
84     except Error as e:
85         print(f"Error: {e}")
86
87     finally:
88         if cursor:
89             cursor.close()
90         self._db_connector.close_connection()
91
92 Patient > get_Patient_Details() > finally
93 Run Doctorclass x
94 python_coding_challenge_03 > Patientclass.py
```

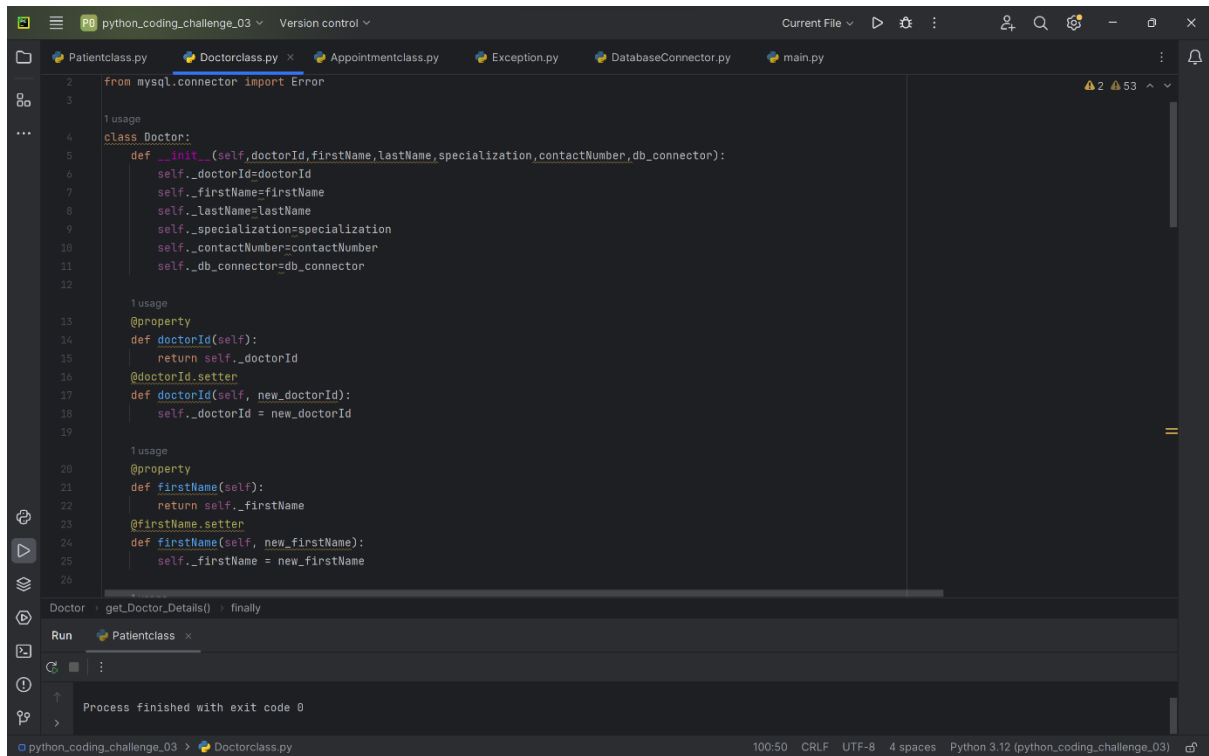
```
94 def get_Patient_Details(self, patientId):
95     try:
96         self._db_connector.open_connection()
97
98         query = "SELECT * FROM Patient WHERE patientId=?"
99         values = (patientId,)
100
101         with self._db_connector.connection.cursor(dictionary=True) as cursor:
102             cursor.execute(query, values)
103             patient_details = cursor.fetchone()
104
105             if patient_details:
106                 print("Patient Details:")
107                 print(f"Patient ID: {patient_details['patientId']}")
108                 print(f"First Name: {patient_details['firstName']}")
109                 print(f>Last Name: {patient_details['lastName']}")
110                 print(f"Date Of Birth: {patient_details['dateOfBirth']}")
111                 print(f"Gender: {patient_details['gender']}")
112                 print(f"Date Of Birth: {patient_details['dateOfBirth']}")
113                 print(f"Address: {patient_details['address']}")
114             else:
115                 print("Customer not found.")
116
117     except Exception as e:
118         print(f"Error getting customer details: {e}")
119
120     finally:
121         self._db_connector.close_connection()
```

Run Patientclass x

Process finished with exit code 0

python_coding_challenge_03 Patientclass.py 121:50 CRLF UTF-8 4 spaces Python 3.12 (python_coding_challenge_03)

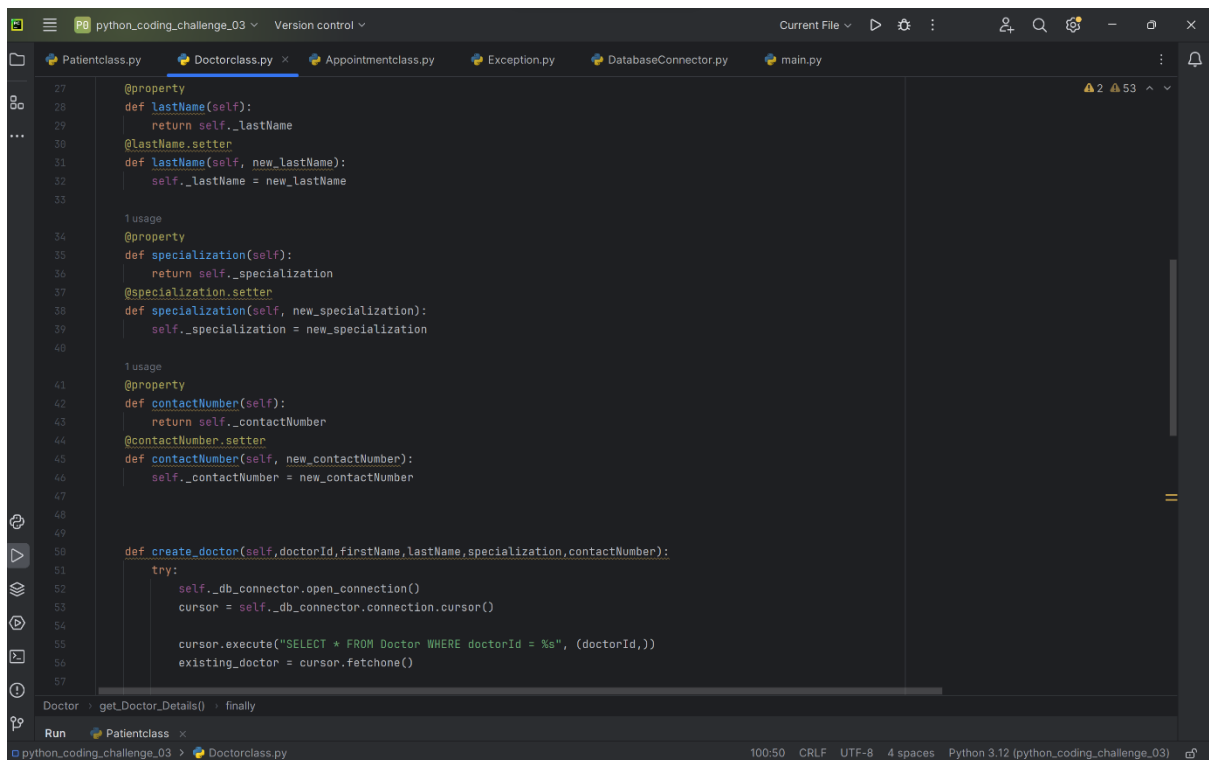
Doctor class



The screenshot shows a code editor with the file `Doctorclass.py` open. The code defines a `Doctor` class with the following attributes and methods:

```
2 from mysql.connector import Error
3
4 class Doctor:
5     def __init__(self, doctorId, firstName, lastName, specialization, contactNumber, db_connector):
6         self._doctorId = doctorId
7         self._firstName = firstName
8         self._lastName = lastName
9         self._specialization = specialization
10        self._contactNumber = contactNumber
11        self._db_connector = db_connector
12
13    @property
14    def doctorId(self):
15        return self._doctorId
16    @doctorId.setter
17    def doctorId(self, new_doctorId):
18        self._doctorId = new_doctorId
19
20    @property
21    def firstName(self):
22        return self._firstName
23    @firstName.setter
24    def firstName(self, new_firstName):
25        self._firstName = new_firstName
26
```

The bottom of the editor shows a terminal window with the message: "Process finished with exit code 0".



The screenshot shows the continuation of the `Doctor` class implementation in `Doctorclass.py`. The code includes properties for `lastName`, `specialization`, and `contactNumber`, along with a `create_doctor` method.

```
27 @property
28 def lastName(self):
29     return self._lastName
30 @lastName.setter
31 def lastName(self, new_lastName):
32     self._lastName = new_lastName
33
34 @property
35 def specialization(self):
36     return self._specialization
37 @specialization.setter
38 def specialization(self, new_specialization):
39     self._specialization = new_specialization
40
41 @property
42 def contactNumber(self):
43     return self._contactNumber
44 @contactNumber.setter
45 def contactNumber(self, new_contactNumber):
46     self._contactNumber = new_contactNumber
47
48
49
50 def create_doctor(self, doctorId, firstName, lastName, specialization, contactNumber):
51     try:
52         self._db_connector.open_connection()
53         cursor = self._db_connector.connection.cursor()
54
55         cursor.execute("SELECT * FROM Doctor WHERE doctorId = %s", (doctorId,))
56         existing_doctor = cursor.fetchone()
57
```

The bottom of the editor shows a terminal window with the message: "Process finished with exit code 0".

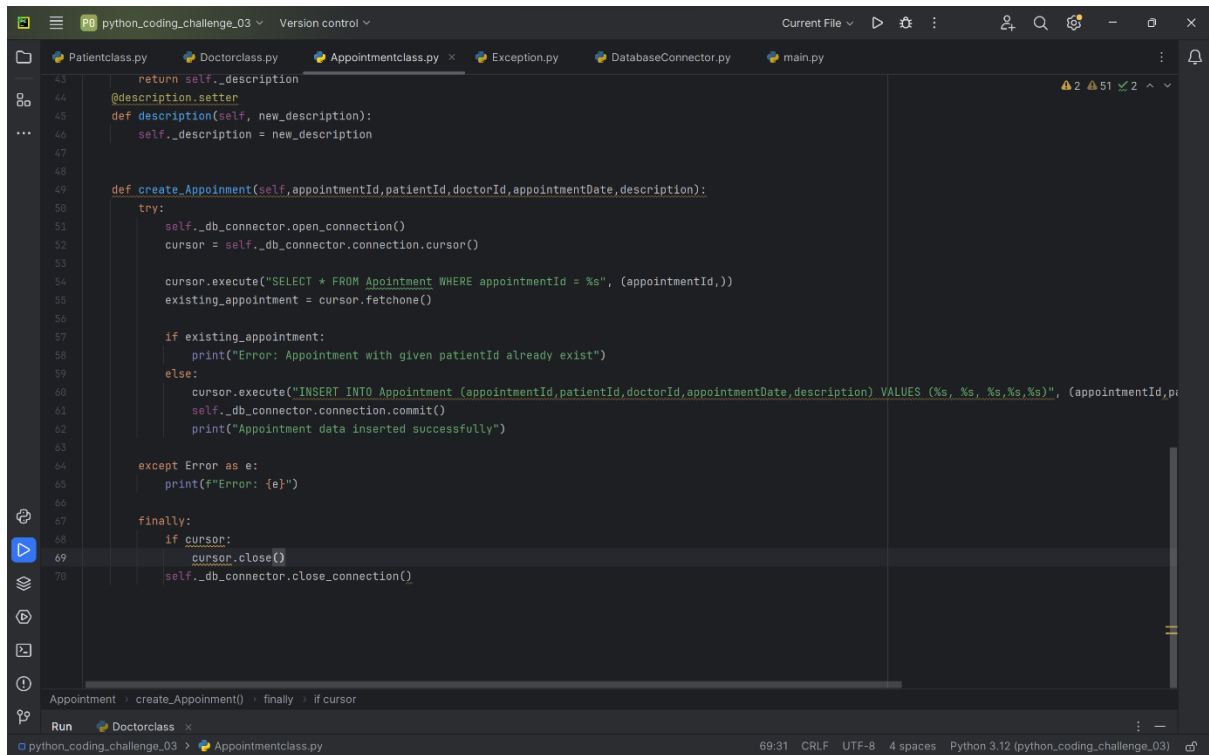
```
python_coding_challenge_03  Version control  Current File  2 53  ~  x
Patientclass.py  Doctorclass.py  Appointmentclass.py  Exception.py  DatabaseConnector.py  main.py
...
50  def create_doctor(self, doctorId, firstName, lastName, specialization, contactNumber):
51      try:
52          self._db_connector.open_connection()
53          cursor = self._db_connector.connection.cursor()
54
55          cursor.execute("SELECT * FROM Doctor WHERE doctorId = %s", (doctorId,))
56          existing_doctor = cursor.fetchone()
57
58          if existing_doctor:
59              print("Error: Doctor with given doctorId already exist")
60          else:
61              cursor.execute("INSERT INTO Doctor (doctorId, firstName, lastName, specialization, contactNumber) VALUES (%s, %s, %s, %s, %s)", (doctorId, firstName, lastName, specialization, contactNumber))
62              self._db_connector.connection.commit()
63              print("Doctor data inserted successfully")
64
65      except Error as e:
66          print(f"Error: {e}")
67
68      finally:
69          if cursor:
70              cursor.close()
71          self._db_connector.close_connection()
72
73
74  def get_Doctor_Details(self, doctorId):
75      try:
76          self._db_connector.open_connection()
77
78          query = "SELECT * FROM Patient WHERE doctorId=%s"
79          values = (doctorId,)
80
81          with self._db_connector.connection.cursor(dictionary=True) as cursor:
82              cursor.execute(query, values)
83
84      finally:
85          self._db_connector.close_connection()
86
87  Doctor > get_Doctor_Details() > finally
Run  Patientclass  x
python_coding_challenge_03 > Doctorclass.py  100:50  CRLF  UTF-8  4 spaces  Python 3.12 (python_coding_challenge_03)
```

```
python_coding_challenge_03  Version control  Current File  2 53  ~  x
Patientclass.py  Doctorclass.py  Appointmentclass.py  Exception.py  DatabaseConnector.py  main.py
...
73  def get_Doctor_Details(self, doctorId):
74      try:
75          self._db_connector.open_connection()
76
77          query = "SELECT * FROM Patient WHERE doctorId=%s"
78          values = (doctorId,)
79
80          with self._db_connector.connection.cursor(dictionary=True) as cursor:
81              cursor.execute(query, values)
82              doctor_details = cursor.fetchone()
83
84          if doctor_details:
85              print("Patient Details:")
86              print(f"doctor ID: {doctor_details['doctorId']}")
87              print(f"First Name: {doctor_details['firstName']}")
88              print(f>Last Name: {doctor_details['lastName']}")
89              print(f"specialization: {doctor_details['specialization']}")
90              print(f"contactNumber: {doctor_details['contactNumber']}")
91
92          else:
93              print("Doctor not found.")
94
95      except Exception as e:
96          print(f"Error getting Doctor details: {e}")
97
98      finally:
99          self._db_connector.close_connection()
100
101  Doctor > get_Doctor_Details() > finally
Run  Doctorclass  x
Process finished with exit code 0
python_coding_challenge_03 > Doctorclass.py  100:50  CRLF  UTF-8  4 spaces  Python 3.12 (python_coding_challenge_03)
```

Appointment class

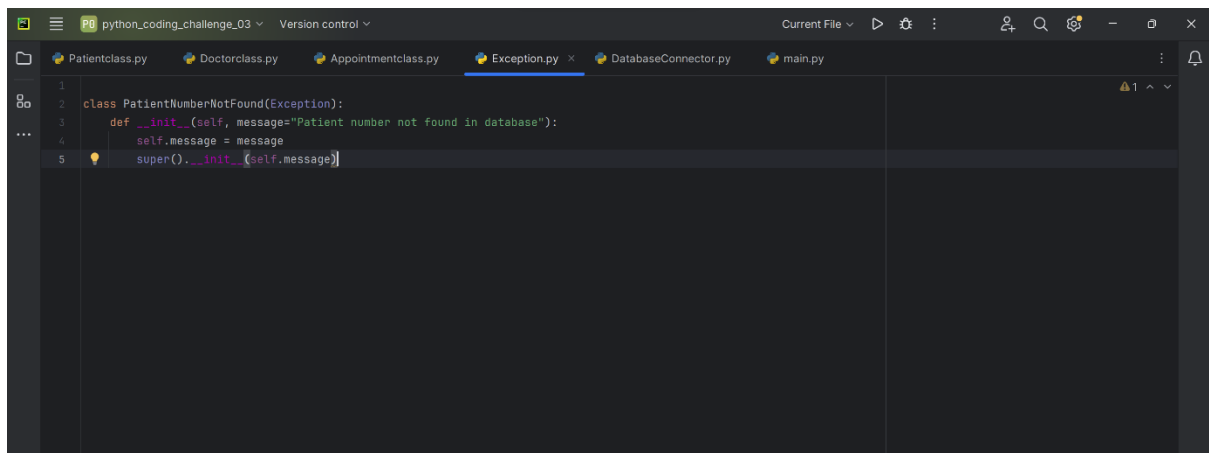
```
1 from mysql.connector import Error
2
3 1 usage
4 class Appointment:
5     def __init__(self, appointmentId, patientId, doctorId, appointmentDate, description, db_connector):
6         self._appointmentId = appointmentId
7         self._patientId = patientId
8         self._doctorId = doctorId
9         self._appointmentDate = appointmentDate
10        self._description = description
11        self._db_connector = db_connector
12
13    1 usage
14    @property
15    def appointmentId(self):
16        return self._appointmentId
17    @appointmentId.setter
18    def appointmentId(self, new_appointmentId):
19        self._appointmentId = new_appointmentId
20
21    1 usage
22    @property
23    def patientId(self):
24        return self._patientId
25    @patientId.setter
26    def patientId(self, new_patientId):
27        self._patientId = new_patientId
28
29    1 usage
30    @property
31    def doctorId(self):
32        return self._doctorId
33    @doctorId.setter
34    def doctorId(self, new_doctorId):
35        self._doctorId = new_doctorId
36
37    1 usage
38    @property
39    def appointmentDate(self):
40        return self._appointmentDate
41    @appointmentDate.setter
42    def appointmentDate(self, new_appointmentDate):
43        self._appointmentDate = new_appointmentDate
44
45    1 usage
46    @property
47    def description(self):
48        return self._description
49    @description.setter
50    def description(self, new_description):
51        self._description = new_description
52
53    def create_Appointment(self, appointmentId, patientId, doctorId, appointmentDate, description):
54        try:
55            self._db_connector.open_connection()
56            cursor = self._db_connector.connection.cursor()
57
58            cursor.execute("SELECT * FROM Appointment WHERE appointmentId = %s", (appointmentId,))
59            existing_appointment = cursor.fetchone()
60
61            if existing_appointment:
```

```
27    @property
28    def doctorId(self):
29        return self._doctorId
30    @doctorId.setter
31    def doctorId(self, new_doctorId):
32        self._doctorId = new_doctorId
33
34    1 usage
35    @property
36    def appointmentDate(self):
37        return self._appointmentDate
38    @appointmentDate.setter
39    def appointmentDate(self, new_appointmentDate):
40        self._appointmentDate = new_appointmentDate
41
42    1 usage
43    @property
44    def description(self):
45        return self._description
46    @description.setter
47    def description(self, new_description):
48        self._description = new_description
49
50    def create_Appointment(self, appointmentId, patientId, doctorId, appointmentDate, description):
51        try:
52            self._db_connector.open_connection()
53            cursor = self._db_connector.connection.cursor()
54
55            cursor.execute("SELECT * FROM Appointment WHERE appointmentId = %s", (appointmentId,))
56            existing_appointment = cursor.fetchone()
57
58            if existing_appointment:
```



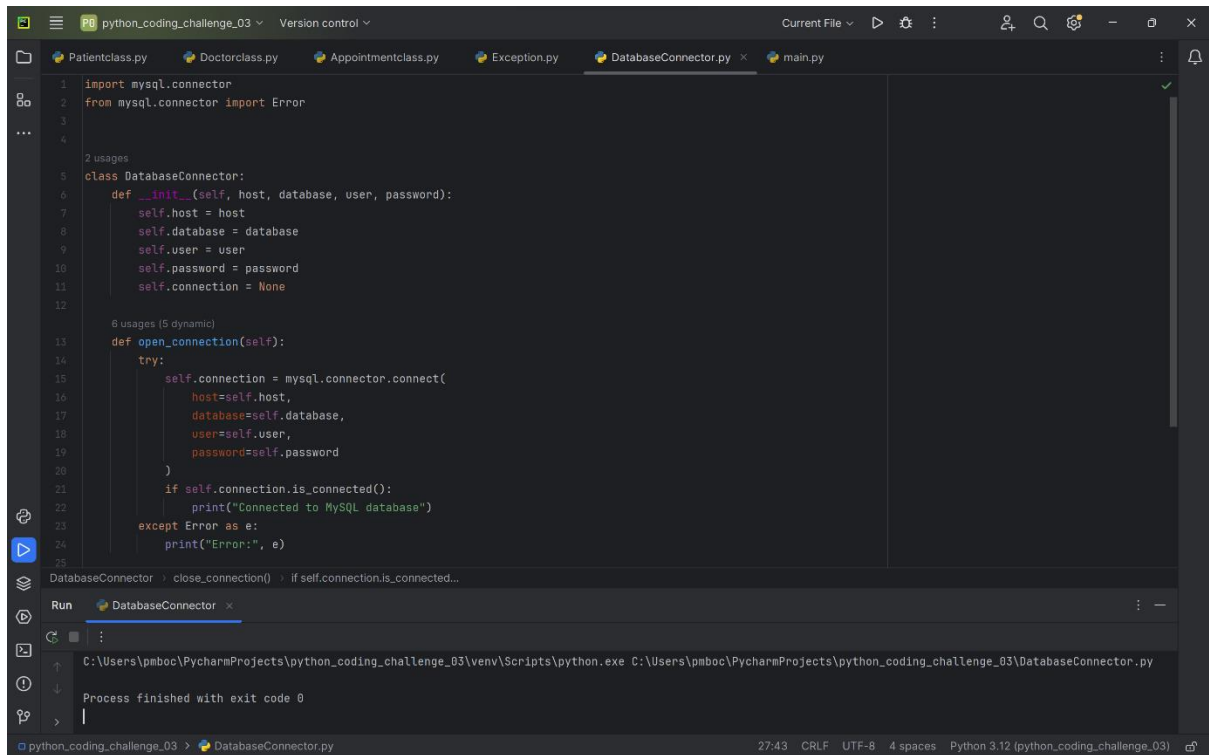
```
43     return self._description
44     @description.setter
45     def description(self, new_description):
46         self._description = new_description
47
48
49     def create_Appointment(self, appointmentId, patientId, doctorId, appointmentDate, description):
50         try:
51             self._db_connector.open_connection()
52             cursor = self._db_connector.connection.cursor()
53
54             cursor.execute("SELECT * FROM Appointment WHERE appointmentId = %s", (appointmentId,))
55             existing_appointment = cursor.fetchone()
56
57             if existing_appointment:
58                 print("Error: Appointment with given patientId already exist")
59             else:
60                 cursor.execute("INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate, description) VALUES (%s, %s, %s, %s, %s)", (appointmentId, patientId, doctorId, appointmentDate, description))
61                 self._db_connector.connection.commit()
62                 print("Appointment data inserted successfully")
63
64         except Error as e:
65             print(f"Error: {e}")
66
67         finally:
68             if cursor:
69                 cursor.close()
70             self._db_connector.close_connection()
```

Exception



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

Database Connection



The screenshot shows the PyCharm IDE with the file `DatabaseConnector.py` open. The code defines a `DatabaseConnector` class with an `__init__` method that initializes attributes and an `open_connection` method that attempts to connect to a MySQL database. The `open_connection` method uses a try-except block to handle connection errors. The `Run` button is visible, and the console shows the process finished with exit code 0.

```
1 import mysql.connector
2 from mysql.connector import Error
3
4
5 2 usages
6 class DatabaseConnector:
7     def __init__(self, host, database, user, password):
8         self.host = host
9         self.database = database
10        self.user = user
11        self.password = password
12        self.connection = None
13
14 6 usages (5 dynamic)
15 def open_connection(self):
16     try:
17         self.connection = mysql.connector.connect(
18             host=self.host,
19             database=self.database,
20             user=self.user,
21             password=self.password
22         )
23         if self.connection.is_connected():
24             print("Connected to MySQL database")
25     except Error as e:
26         print("Error:", e)
```

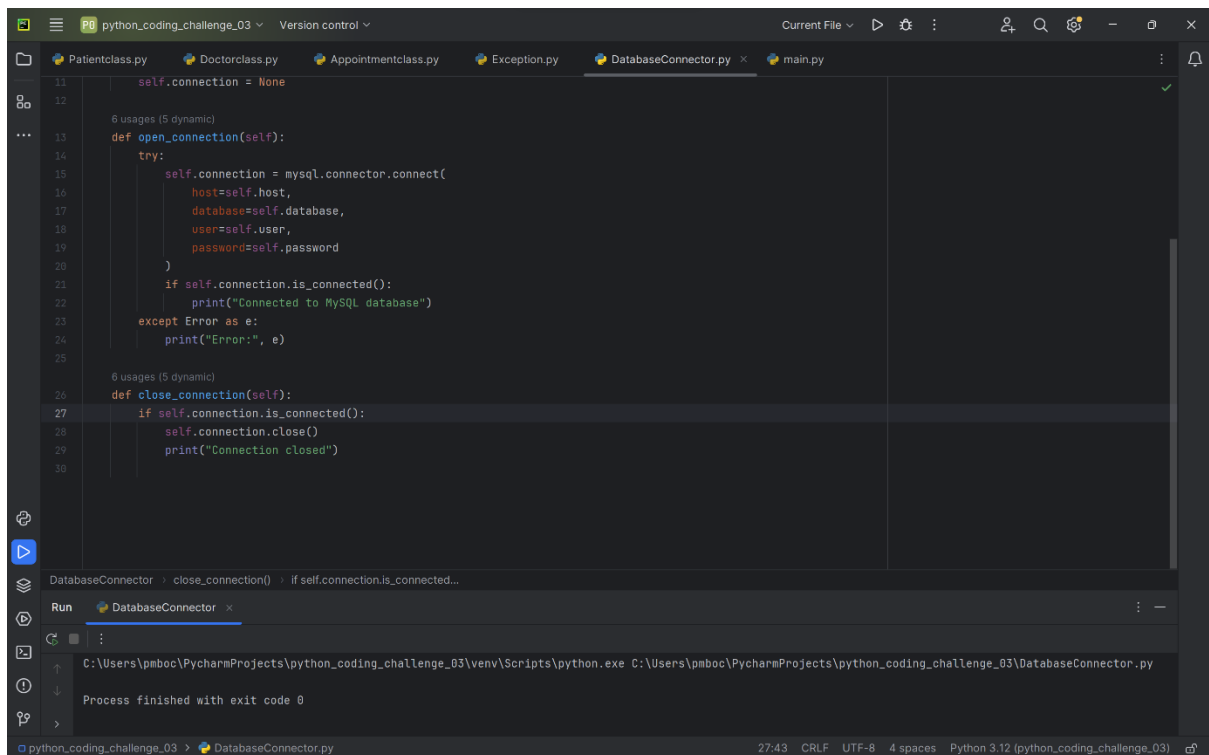
DatabaseConnector → close_connection() → if self.connection.is_connected...

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)



The screenshot shows the PyCharm IDE with the file `DatabaseConnector.py` open. The code now includes a `close_connection` method that checks if the connection is active and closes it. The `Run` button is visible, and the console shows the process finished with exit code 0.

```
11 self.connection = None
12
13 6 usages (5 dynamic)
14 def open_connection(self):
15     try:
16         self.connection = mysql.connector.connect(
17             host=self.host,
18             database=self.database,
19             user=self.user,
20             password=self.password
21         )
22         if self.connection.is_connected():
23             print("Connected to MySQL database")
24     except Error as e:
25         print("Error:", e)
```

6 usages (5 dynamic)

```
26 def close_connection(self):
27     if self.connection.is_connected():
28         self.connection.close()
29         print("Connection closed")
30
```

DatabaseConnector → close_connection() → if self.connection.is_connected...

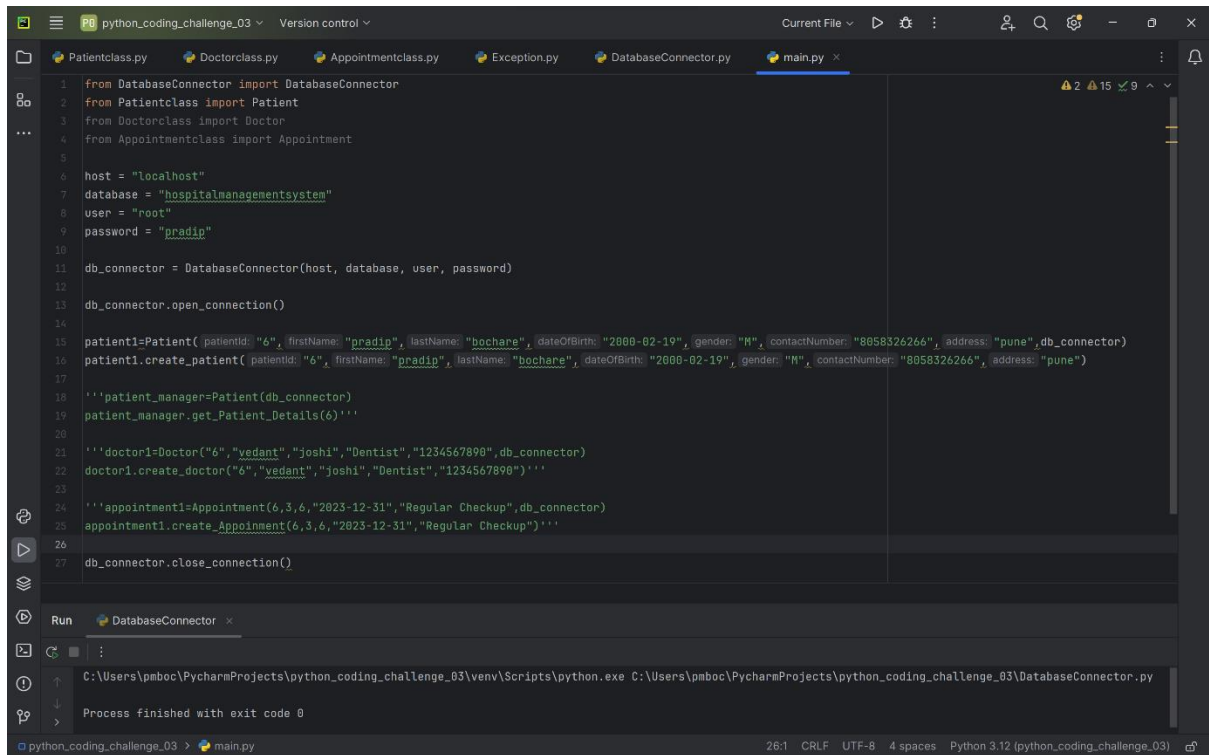
Run DatabaseConnector ×

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

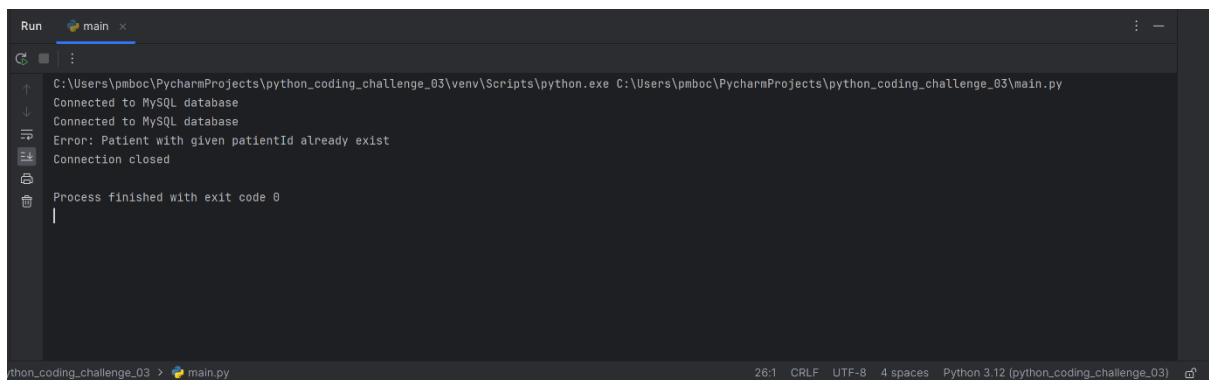


```
1 from DatabaseConnector import DatabaseConnector
2 from Patientclass import Patient
3 from Doctorclass import Doctor
4 from Appointmentclass import Appointment
5
6 host = "localhost"
7 database = "hospitalmanagementsystem"
8 user = "root"
9 password = "pradip"
10
11 db_connector = DatabaseConnector(host, database, user, password)
12
13 db_connector.open_connection()
14
15 patient1=Patient( patientId: "6", firstName: "pradip", lastName: "bochare", dateOfBirth: "2000-02-19", gender: "M", contactNumber: "8058326266", address: "pune",db_connector)
16 patient1.create_patient( patientId: "6", firstName: "pradip", lastName: "bochare", dateOfBirth: "2000-02-19", gender: "M", contactNumber: "8058326266", address: "pune")
17
18 '''patient_manager=Patient(db_connector)
19 patient_manager.get_Patient_Details(6)'''
20
21 '''doctor1=Doctor("6", "vedant", "joshi", "Dentist", "1234567890", db_connector)
22 doctor1.create_doctor("6", "vedant", "joshi", "Dentist", "1234567890")'''
23
24 '''appointment1=Appointment(6,3,6,"2023-12-31","Regular Checkup",db_connector)
25 appointment1.create_Appointment(6,3,6,"2023-12-31","Regular Checkup")'''
26
27 db_connector.close_connection()
```

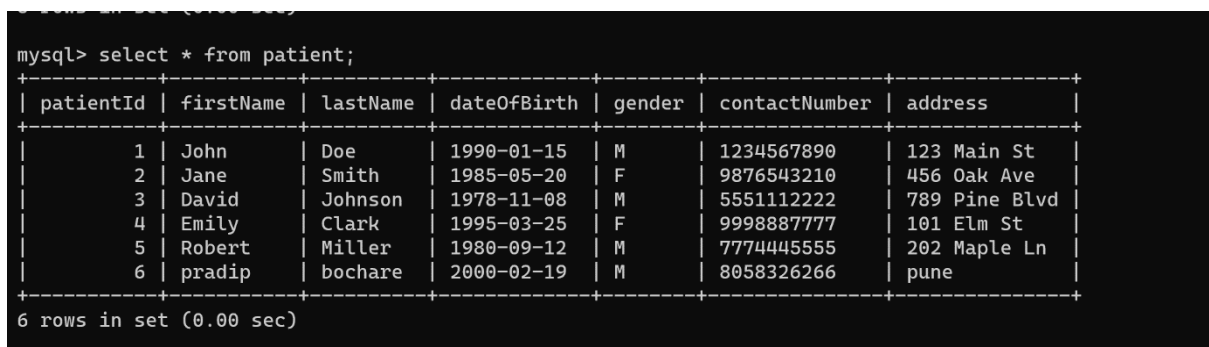
Run DatabaseConnector

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



```
Run main
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 x
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
```

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

```
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
onconnector):
    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 Appointment 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

```
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_Appointment(6,3,6,"2023-12-31","Regular Checkup")

db_connector.close_connection()
```