INSURANCE MANAGEMENT SYSTEM

ENTITY

User:

```
from util.DBConnUtil import DBConnection


class User(DBConnection):
    def __init__(self):
        super().__init__()
        self.userId = 0
        self.userName = ''
        self.password = ''
        self.role = ''

    # SETTERS
    def set_userId(self, value):
        self.userId = value

    def set_userName(self, value):
        self.userName = value

    def set_password(self, value):
        self.password = value

    def set_role(self, value):
        self.role = value

    # GETTERS
    def get_userId(self):
        return self.userId

    def get_userName(self):
        return self.userName

    def get_password(self):
        return self.password

    def get_role(self):
        return self.role

    def __str__(self):
        return f'User ID: {self.userId} User Name: {self.userName} Role: {self.role}'
```

Client:

```
from entity.Policy import Policy


class Client(Policy):
    def __init__(self):
        super().__init__()
        self.clientId = 0
        self.clientName = ''
        self.contactInfo = 0
        self.policyId = ''

    #Setters
    def set_clientId(self, value):
        self.clientId = value

    def set_clientName(self, value):
        self.clientName = value

    def set_contactInfo(self, value):
        self.contactInfo = value

    def set_policyId(self, value):
        self.policyId = value

    #Getters
    def get_clientId(self):
        return self.clientId

    def get_clientName(self):
        return self.clientName

    def get_contactInfo(self):
        return self.contactInfo

    def get_policyId(self):
        return self.policyId

    def __str__(self):
        return f'Client ID: {self.clientId} Client Name: {self.clientName} Contact
Info: {self.contactInfo}\n' \
               f' Policy ID: {self.policyId}'
```

Claim:

```
from entity.Client import Client


class Claim(Client):
    def __init__(self):
        super().__init__()
        self.claimId = 0
        self.claimNumber = 0
        self.dateFiled = ''
        self.claimAmount = 0.0
        self.status = ''
        self.policyId = ''
        self.clientId = 0

    #Setters
    def set_claimId(self, value):
        self.claimId = value

    def set_claimNumber(self, value):
        self.claimNumber = value

    def set_dateFiled(self, value):
        self.dateFiled = value

    def set_claimAmount(self, value):
        self.claimAmount = value

    def set_status(self, value):
        self.status = value

    def set_policyId(self, value):
        self.policyId = value

    def set_clientId(self, value):
        self.clientId = value

    #Getters
    def get_claimId(self):
        return self.claimId

    def get_claimNumber(self):
        return self.claimNumber

    def get_dateFiled(self):
        return self.dateFiled

    def get_claimAmount(self):
        return self.claimAmount

    def get_status(self):
        return self.status

    def get_policyId(self):
        return self.status

    def get_clientId(self):
        return self.clientId


    def __str__(self):
        return f'Claim ID: {self.claimId} Claim Number: {self.claimNumber}\n' \
               f'Date Filed: {self.dateFiled} Claim Amount: {self.claimAmount} Status:
{self.status}\n' \
               f'Policy ID: {self.policyId} Client ID: {self.clientId}'
```

Policy:

```
from util.DBConnUtil import DBConnection

class Policy(DBConnection):
    def __init__(self):
        super().__init__()
        self.policyId = 0
        self.policyName = ' '

    #Setters

    def set_policyId(self,value):
        self.policyId = value

    def set_policyName(self):
        self.policyName = value

    #Getters

    def get_policyId(self):
        return self.policyId

    def get_policyName(self):
        return self.policyName

    def __str__(self):
        return f'Policy ID : {self.policyId} Policy Name : {self.policyName}'
```

Payment:

```
from entity.Client import Client

class Payment(Client):
    def __init__(self):
        super().__init__()
        self.paymentId = 0
        self.paymentDate = ' '
        self.paymentAmount = 0.0
        self.clientId = 0

    #Setters
    def set_paymentId(self, value):
        self.paymentId = value

    def set_paymentDate(self, value):
        self.paymentDate = value

    def set_paymentAmount(self, value):
        self.paymentAmount = value

    def set_clientId(self, value):
        self.clientId = value


    #Getters
    def get_paymentId(self):
        return self.paymentId

    def get_paymentDate(self):
        return self.paymentDate
```

```
    def get_paymentAmount(self):
        return self.paymentAmount

    def get_clientId(self):
        return self.clientId

    def __str__(self):
        return f'Payment ID: {self.paymentId} Payment Date: {self.paymentDate} Payment
Amount: {self.paymentAmount}\n' \
               f' Client ID: {self.clientId}'
```

DAO

UserDAO:

```
from entity.User import User

class UserDAO(User):
    def __init__(self):
        super().__init__()

    def perform_user_actions(self):
        while True:
            print("(User 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_user_table()
            elif ch == 2:
                print(self.add_user())
            elif ch == 3:
                print(self.update_user())
            elif ch == 4:
                print(self.delete_user())
            elif ch == 5:
                self.select_user()
            elif ch == 0:
                break
            else:
                print("Enter Valid Choice")
    def create_user_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS User(
            userId INT PRIMARY KEY,
            userName VARCHAR(50),
            password VARCHAR(50) NOT NULL,
            role VARCHAR(50)
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("User Table Created Successfully")
        except Exception as e:
            print(e)

    def add_user(self):
        try:
            self.open()
            self.userId = int(input("Enter user ID: "))
            self.userName = input("Enter UserName: ")
            self.password = input("Enter Password: ")
            self.role = input("Enter role: ")
            data = [(self.userId,self.userName,self.password,self.role)]
```

```
        insert_str = '''INSERT INTO User(userId,userName,password,role)
                        VALUES(%s,%s,%s,%s)'''
        self.stmt.executemany(insert_str,data)
        self.conn.commit()
        self.close()
        return True
    except Exception as e:
        print(e)

def update_user(self):
    try:
        self.open()
        user_id = int(input("Enter userID to be updated: "))
        self.userName = input("Enter userName: ")
        self.password = input("Enter password: ")
        self.role = input("Enter new role: ")
        data = [(self.userName,self.password,self.role,user_id)]
        update_str = '''UPDATE User SET userName = %s, password = %s, role = %s
                        WHERE userId = %s'''
        self.stmt.executemany(update_str,data)
        self.conn.commit()
        self.close()
        return True
    except Exception as e:
        print(e)

def delete_user(self):
    try:
        self.open()
        user_id = int(input("Enter user ID to be deleted: "))
        delete_str = f'''DELETE FROM User
                        WHERE userId = {user_id}'''
        self.stmt.execute(delete_str)
        self.conn.commit()
        self.close()
        return True
    except Exception as e:
        print(e)

def select_user(self):
    try:
        select_str = '''SELECT * FROM User'''
        self.open()
        self.stmt.execute(select_str)
        records = self.stmt.fetchall()
        self.close()
        print("Records in Table")
        for i in records:
            print(i)
    except Exception as e:
        print(e)
```

ClientDAO:

```
from entity.Client import Client


class ClientDAO(Client):
    def __init__(self):
        super().__init__()

    def perform_client_actions(self):
        while True:
            print("(Client) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_client_table()
            elif ch == 2:
                print(self.add_client())
            elif ch == 3:
                print(self.update_client())
            elif ch == 4:
                print(self.delete_client())
            elif ch == 5:
                self.select_client()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_client_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Client (
            clientId INT PRIMARY KEY,
            clientName VARCHAR(50),
            contactInfo INT,
            policyId INT,
            FOREIGN KEY(policyId) REFERENCES Policy(policyId) ON DELETE CASCADE ON
UPDATE CASCADE)'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Client Table Created successfully.')
        except Exception as e:
            print(e)

    def add_client(self):
        try:
            self.open()
            self.clientId = int(input('Enter Client ID: '))
            self.clientName = input('Enter Client Name: ')
            self.contactInfo = int(input('Enter Contact Info: '))
            self.policyId = int(input('Enter Policy ID: '))
            data = [(self.clientId, self.clientName, self.contactInfo, self.policyId)]
            insert_str = '''INSERT INTO Client(clientId, clientName, contactInfo,
policyId)
                            VALUES(%s, %s, %s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def update_client(self):
        try:
            self.open()
            client_id = int(input('Input Client ID to be Updated: '))
            self.clientName = input('Enter Client Name: ')
```

```
            self.contactInfo = int(input('Enter Contact Info: '))
            self.policyId = int(input('Enter Policy ID: '))
            data = [(self.clientName, self.contactInfo, self.policyId, client_id)]
            update_str = '''UPDATE Client SET clientName=%s, contactInfo=%s,
policyId=%s
                            WHERE clientId = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def delete_client(self):
        try:
            self.open()
            client_id = int(input('Input Client ID to be Deleted: '))
            delete_str = f'''DELETE FROM Client WHERE clientId = {client_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_client(self):
        try:
            select_str = '''SELECT * FROM Client'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Client Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

ClaimDAO:

```
from entity.Claim import Claim


class ClaimDAO(Claim):
    def __init__(self):
        super().__init__()

    def perform_claim_actions(self):
        while True:
            print("(Claim) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_claim_table()
            elif ch == 2:
                print(self.add_claim())
            elif ch == 3:
                print(self.update_claim())
            elif ch == 4:
                print(self.delete_claim())
            elif ch == 5:
                self.select_claim()
            elif ch == 0:
                break
            else:
                print("Invalid choice")
```

```
    def create_claim_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Claim (
            claimId INT PRIMARY KEY,
            claimNumber INT,
            dateFiled DATE,
            claimAmount FLOAT,
            status VARCHAR(50),
            policyId INT,
            clientId INT,
            FOREIGN KEY(policyId) REFERENCES Policy(policyId) ON DELETE CASCADE ON
UPDATE CASCADE,
            FOREIGN KEY(clientId) REFERENCES Client(clientId) ON DELETE CASCADE ON
UPDATE CASCADE)'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Claim Table Created successfully.')
        except Exception as e:
            print(e)

    def add_claim(self):
        try:
            self.open()
            self.claimId = int(input('Enter Claim ID: '))
            self.claimNumber = int(input('Enter Claim Number: '))
            self.dateFiled = input('Enter Date Filed: ')
            self.claimAmount = float(input('Enter Claim Amount: '))
            self.status = input('Enter Status: ')
            self.policyId = int(input('Enter Policy ID: '))
            self.clientId = int(input('Enter Client ID: '))
            data = [(self.claimId, self.claimNumber, self.dateFiled, self.claimAmount,
self.status, self.policyId, self.clientId)]
            insert_str = '''INSERT INTO Claim(claimId, claimNumber, dateFiled,
claimAmount, status, policyId, clientId)
                            VALUES(%s, %s, %s, %s, %s, %s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def update_claim(self):
        try:
            self.open()
            claim_id = int(input('Enter Claim ID to be Updated: '))
            self.claimNumber = int(input('Enter Claim Number: '))
            self.dateFiled = input('Enter Date Filed: ')
            self.claimAmount = float(input('Enter Claim Amount: '))
            self.status = input('Enter Status: ')
            self.policyId = int(input('Enter Policy ID: '))
            self.clientId = int(input('Enter Client ID: '))
            data = [(self.claimNumber, self.dateFiled, self.claimAmount, self.status,
self.policyId, self.clientId, claim_id)]
            update_str = '''UPDATE Claim SET claimNumber=%s, dateFiled=%s,
claimAmount=%s, status=%s, policyId=%s, clientId=%s
                            WHERE claimId = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e
```

```
    def delete_claim(self):
        try:
            self.open()
            claim_id = int(input('Input Claim ID to be Deleted: '))
            delete_str = f'''DELETE FROM Claim WHERE claimId = {claim_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_claim(self):
        try:
            select_str = '''SELECT * FROM Claim'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Claim Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

PolicyDAO:

```
from entity.Policy import Policy


class PolicyDAO(Policy):
    def __init__(self):
        super().__init__()

    def perform_policy_actions(self):
        while True:
            print("(Policy) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_policy_table()
            elif ch == 2:
                print(self.add_policy())
            elif ch == 3:
                print(self.update_policy())
            elif ch == 4:
                print(self.delete_policy())
            elif ch == 5:
                self.select_policy()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_policy_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Policy (
            policyId INT PRIMARY KEY,
            policyName VARCHAR(50))'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Policy Table Created successfully.')
        except Exception as e:
            print(e)
```

```python
    def add_policy(self):
        try:
            self.open()
            self.policyId = int(input('Enter Policy ID: '))
            self.policyName = input('Enter Policy Name: ')
            data = [(self.policyId, self.policyName)]
            insert_str = '''INSERT INTO Policy(policyId, policyName)
                            VALUES(%s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def update_policy(self):
        try:
            self.open()
            policy_id = int(input('Input Policy ID to be Updated: '))
            self.policyName = input('Enter Policy Name: ')
            data = [(self.policyName, policy_id)]
            update_str = '''UPDATE Policy SET policyName=%s
                            WHERE policyId = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            print("Updated Successfully")
            return True
        except Exception as e:
            return e

    def delete_policy(self):
        try:
            self.open()
            policy_id = int(input('Input Policy ID to be Deleted: '))
            delete_str = f'''DELETE FROM Policy WHERE policyId = {policy_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            print("Deleted Successfully")
            return True
        except Exception as e:
            return e

    def select_policy(self):
        try:
            select_str = '''SELECT * FROM Policy'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Policy Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(e)

    def getPolicy(self, policyId):
        pass

    def getAllPolicies(self):
        pass
```

PaymentDAO:

```
from entity.Payment import Payment


class PaymentDAO(Payment):
    def __init__(self):
        super().__init__()

    def perform_payment_actions(self):
        while True:
            print("(Payment) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                self.create_payment_table()
            elif ch == 2:
                print(self.add_payment())
            elif ch == 3:
                print(self.update_payment())
            elif ch == 4:
                print(self.delete_payment())
            elif ch == 5:
                self.select_payment()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    def create_payment_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Payment (
            paymentId INT PRIMARY KEY,
            paymentDate DATE,
            paymentAmount FLOAT,
            clientId INT,
            FOREIGN KEY(clientId) REFERENCES Client(clientId) ON DELETE CASCADE ON
UPDATE CASCADE)'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print('Payment Table Created successfully.')
        except Exception as e:
            print(e)

    def add_payment(self):
        try:
            self.open()
            self.paymentId = int(input('Enter Payment ID: '))
            self.paymentDate = input('Enter Payment Date: ')
            self.paymentAmount = float(input('Enter Payment Amount: '))
            self.clientId = int(input('Enter Client ID: '))
            data = [(self.paymentId, self.paymentDate, self.paymentAmount,
self.clientId)]
            insert_str = '''INSERT INTO Payment(paymentId, paymentDate, paymentAmount,
clientId)
                            VALUES(%s, %s, %s, %s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def update_payment(self):
        try:
            self.open()
            payment_id = int(input('Input Payment ID to be Updated: '))
```

```
            self.paymentDate = input('Enter Payment Date: ')
            self.paymentAmount = float(input('Enter Payment Amount: '))
            self.clientId = int(input('Enter Client ID: '))
            data = [(self.paymentDate, self.paymentAmount, self.clientId, payment_id)]
            update_str = '''UPDATE Payment SET paymentDate=%s, paymentAmount=%s,
clientId=%s
                            WHERE paymentId = %s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def delete_payment(self):
        try:
            self.open()
            payment_id = int(input('Input Payment ID to be Deleted: '))
            delete_str = f'''DELETE FROM Payment WHERE paymentId = {payment_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_payment(self):
        try:
            select_str = '''SELECT * FROM Payment'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print('Records In Payment Table:')
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

InsuranceServiceImpl:

```
from dao.PolicyDAO import PolicyDAO
from exception.PolicyNotFoundException import PolicyNotFoundException

class InsuranceServiceImpl(PolicyDAO):
    def __init__(self):
        super().__init__()

    #To get policy details of a particular policy
    def getPolicy(self,policyId):
        try:
            self.open()
            self.stmt.execute(f'''SELECT COUNT(*) FROM Policy WHERE policyId =
{policyId}''')
            count = self.stmt.fetchone()[0]
            if count == 0:
                return PolicyNotFoundException(policyId)
            else:
                self.stmt.execute(f'''SELECT * FROM Policy WHERE policyId =
{policyId}''')
                records = self.stmt.fetchall()
                self.close()
                return records
        except PolicyNotFoundException as e:
            return e
```

```
        except Exception as e:
            return e

    #To get all Policies

    def getAllPolicies(self):
        try:
            self.open()
            self.stmt.execute(f'''SELECT * FROM Policy''')
            records = self.stmt.fetchall()
            self.close()
            return records
        except Exception as e:
            return e

    def update_policy(self):
        p = PolicyDAO()
        p.update_policy()
        pass

    def delete_policy(self):
        p = PolicyDAO()
        p.delete_policy()
        pass
```

EXCEPTION

PolicyNotFoundException:

```
class PolicyNotFoundException(Exception):
    def __init__(self,policyId):
        super().__init__(f"Policy ID : {policyId} not found in the system..")
```

UTIL

DBpropertyUtil:

```
class PropertyUtil:
    connection_properties = None

    @staticmethod
    def getConnectionString():
        if PropertyUtil.connection_properties is None:
            host = 'localhost'
            database = 'insurance_managementdb'
            user = 'root'
            password = 'Mahitharsha@1'
            PropertyUtil.connection_properties = {'host': host, 'database': database,
'user': user, 'password': password}
        return PropertyUtil.connection_properties
```

DBConnUtil:

```
#import sys
import mysql.connector as sql
from util.DBPropertyUtil import PropertyUtil
```

```
class DBConnection:
    def open(self):
        try:
            # print('--Database Is Connected:--')
            connection_properties = PropertyUtil.getConnectionString()
            self.conn = sql.connect(**connection_properties)
            self.stmt = self.conn.cursor()
        except Exception as e:
            print(str(e) + ' --Database Is Not Connected:--')
            sys.exit(1)

    def close(self):
        self.conn.close()
        # print('--Connection Is Closed:--')
```

MAIN

Main (Executable file)

```
from dao.InsuranceServiceImpl import InsuranceServiceImpl
from dao.UserDAO import UserDAO
from dao.PolicyDAO import PolicyDAO
from dao.ClientDAO import ClientDAO
from dao.ClaimDAO import ClaimDAO
from dao.PaymentDAO import PaymentDAO
from exception.PolicyNotFoundException import PolicyNotFoundException
from util.DBConnUtil import DBConnection


def main():

    dbconnection = DBConnection()

    try:
        dbconnection.open()
        print("--Database Is Connected:--")
    except Exception as e:
        print(e)

    try:
        print("=" * 30)
        print("Insurance Management System")
        print("=" * 30)
        print("Welcome to Insurance Management System!")

        insurance_management_system = InsuranceServiceImpl()

        while True:
            print("1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                u = UserDAO()
                u.perform_user_actions()
            elif ch == 2:
                p = PolicyDAO()
                p.perform_policy_actions()
            elif ch == 3:
                c = ClientDAO()
                c.perform_client_actions()
            elif ch == 4:
                cl = ClaimDAO()
                cl.perform_claim_actions()
            elif ch == 5:
```

```
                pp = PaymentDAO()
                pp.perform_payment_actions()

            elif ch == 0:
                break
            else:
                print("Invalid choice")

        while True:
            print("=" * 10)
            print("---MENU---")
            print("=" * 10)

print("1.getPolicy\n2.getAllPolicies\n3.updatePolicy\n4.deletePolicy\n0.EXIT")
            ch = int(input("Enter choice: "))
            if ch == 1:
                print(insurance_management_system.getPolicy(int(input('Enter Policy ID
to get details: '))))
            elif ch == 2:
                print(f'List of all Policies:
{insurance_management_system.getAllPolicies()}')
            elif ch == 3:
                insurance_management_system.update_policy()
            elif ch == 4:
                insurance_management_system.delete_policy()
            elif ch == 0:
                break
            else:
                print("Invalid choice")

    except PolicyNotFoundException as e:
        print(e)

    except Exception as e:
        print(e)

    finally:
        dbconnection.close()
        print("Thankyou for visiting Insurance Management System!")
        print("--Connection Is Closed:--")


if __name__ == "__main__":
    main()
```

OUTPUTS

#Running the main file

```
Run: main
C:\Users\shyam\PycharmProjects\Insurance_Management_System\venv\Scripts\python.exe C:\Users\shyam\PycharmProjects\Insurance_Management_System\main\main.py
--Database Is Connected:--
==========
Insurance Management System
==========
Welcome to Insurance Management System!
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice:
```

#Selecting the options

```
Run: main
==============================
Insurance Management System
==============================
Welcome to Insurance Management System!
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 1
(User 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 0
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 2
(Policy) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 0
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 3
(Client) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 0
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 4
(Claim) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 0
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 5
(Payment) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter choice: 0
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice:
```

➔ By selecting respective options from above we can perform CREATE,INSERT,UPDATE,DELETE,SELECT operations that automatically reflects in our database

#Directly exit from the loop to enter into menu

```
C:\Users\shyam\PycharmProjects\Insurance_Management_System\venv\Scripts\python.exe C:\Users\shyam\PycharmProjects\Insurance_Management_System\main\main.py
--Database Is Connected:--
==============================
Insurance Management System
==============================
Welcome to Insurance Management System!
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 0
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice:
```

#To get Policy details based on policy ID

```
==============================
Insurance Management System
==============================
Welcome to Insurance Management System!
1.User 2.Policy 3.Client 4.Claim 5.Payment 0.EXIT
Enter choice: 0
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 1
Enter Policy ID to get details: 1
[(1, 'POLICY1')]
```

#To get all the available policy details

```
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 2
List of all Policies: [(1, 'POLICY1'), (2, 'POLICY2'), (3, 'POLICYYY3')]
```

#To update policy details

```
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 3
Input Policy ID to be Updated: 1
Enter Policy Name: Policy111
Updated Successfully
```

```
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 1
Enter Policy ID to get details: 1
[(1, 'Policy111')]
```

#To delete policy details

```
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 4
Input Policy ID to be Deleted: 3
Deleted Successfully
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 2
List of all Policies: [(1, 'Policy111'), (2, 'POLICY2')]
```

#Finally

```
==========
---MENU---
==========
1.getPolicy
2.getAllPolicies
3.updatePolicy
4.deletePolicy
0.EXIT
Enter choice: 0
Thankyou for visiting Insurance Management System!
--Connection Is Closed:--

Process finished with exit code 0
```