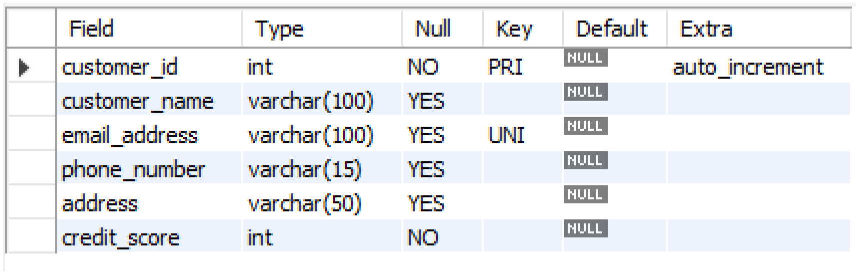
CODING ASSESSMENT - PYTHON - Hanif Mohammed

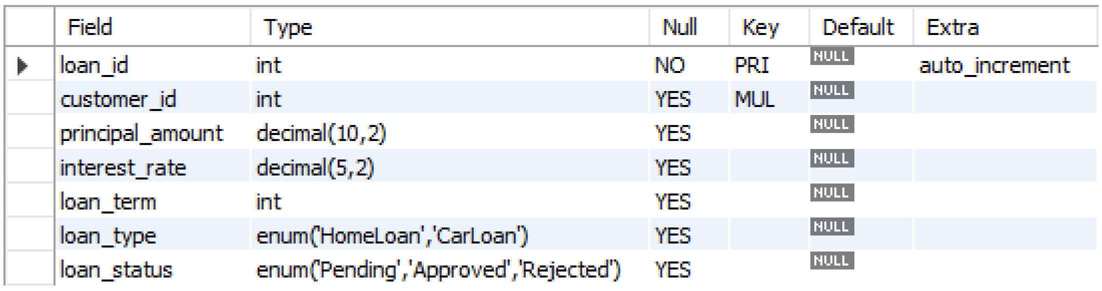
MYSQL Part for Schema Creation

```
create database loan_management;
use loan_management;

create table customer (
    customer_id int auto_increment,
    customer_name varchar(100),
    email_address varchar(100) unique,
    phone_number varchar(15),
    address varchar(50),
    credit_score int not null,
    primary key(customer_id)
);


desc customer;
```



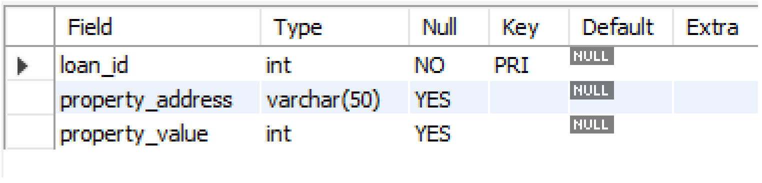
Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
customer_name	varchar(100)	YES		NULL	
email_address	varchar(100)	YES	UNI	NULL	
phone_number	varchar(15)	YES		NULL	
address	varchar(50)	YES		NULL	
credit_score	int	NO		NULL	

```
create table loan (
    loan_id int auto_increment,
    customer_id int,
    principal_amount decimal(10, 2),
    interest_rate decimal(5, 2),
    loan_term int,
    loan_type enum('HomeLoan', 'CarLoan'),
    loan_status enum('Pending', 'Approved', 'Rejected'),
    primary key(loan_id),
    foreign key (customer_id) references customer(customer_id)
);

desc loan;
```



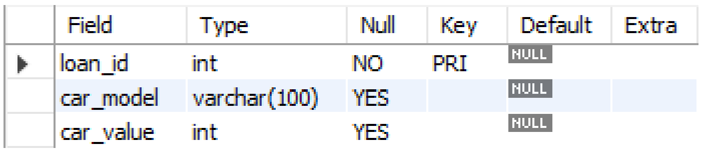
Field	Type	Null	Key	Default	Extra
loan_id	int	NO	PRI	NULL	auto_increment
customer_id	int	YES	MUL	NULL	
principal_amount	decimal(10,2)	YES		NULL	
interest_rate	decimal(5,2)	YES		NULL	
loan_term	int	YES		NULL	
loan_type	enum('HomeLoan','CarLoan')	YES		NULL	
loan_status	enum('Pending','Approved','Rejected')	YES		NULL	

```
create table home_loan (
   loan_id int,
   property_address varchar(50),
   property_value int,
   primary key (loan_id),
   foreign key (loan_id) references loan(loan_id)
);

desc home_loan;
```



Field	Type	Null	Key	Default	Extra
loan_id	int	NO	PRI	NULL	
property_address	varchar(50)	YES		NULL	
property_value	int	YES		NULL	

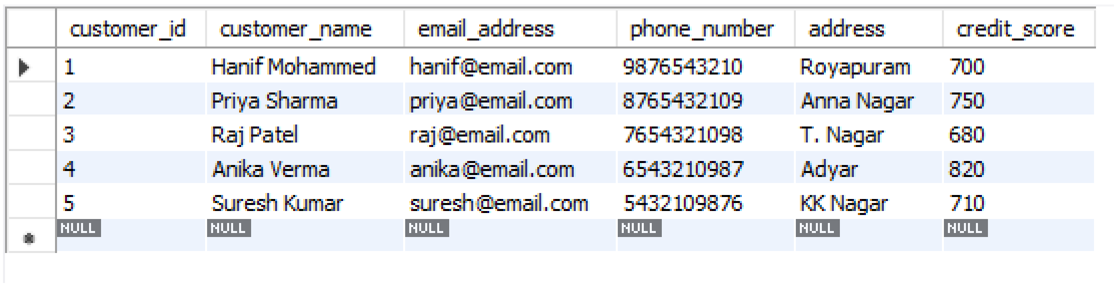
```
create table car_loan (
   loan_id int,
   car_model varchar(100),
   car_value int,
   primary key(loan_id),
   foreign key (loan_id) references loan(loan_id)
);

desc car_loan;
```



Field	Type	Null	Key	Default	Extra
loan_id	int	NO	PRI	NULL	
car_model	varchar(100)	YES		NULL	
car_value	int	YES		NULL	

-Inserting dummy record manually

```
insert into customer (customer_name, email_address, phone_number, address,
credit_score)
values ('Hanif Mohammed', 'hanif@email.com', '9876543210', 'Royapuram', 700),
       ('Priya Sharma', 'priya@email.com', '8765432109', 'Anna Nagar', 750),
       ('Raj Patel', 'raj@email.com', '7654321098', 'T. Nagar', 680),
       ('Anika Verma', 'anika@email.com', '6543210987', 'Adyar', 820),
       ('Suresh Kumar', 'suresh@email.com', '5432109876', 'KK Nagar', 710);

select * from customer;
```



customer_id	customer_name	email_address	phone_number	address	credit_score
1	Hanif Mohammed	hanif@email.com	9876543210	Royapuram	700
2	Priya Sharma	priya@email.com	8765432109	Anna Nagar	750
3	Raj Patel	raj@email.com	7654321098	T. Nagar	680
4	Anika Verma	anika@email.com	6543210987	Adyar	820
5	Suresh Kumar	suresh@email.com	5432109876	KK Nagar	710
NULL	NULL	NULL	NULL	NULL	NULL

PYTHON Part for Class Files

Dao Files

Directory - loan_management/dao/iloan_repository.py

```
from abc import ABC, abstractmethod

class ILoanRepository(ABC):

   @abstractmethod
   def apply_loan(self, loan):
      pass

   @abstractmethod
   def calculate_interest(self, loan_id):
      pass

   @abstractmethod
   def calculate_interest_with_params(self, principal_amount, interest_rate,
loan_term):
      pass

   @abstractmethod
   def loan_status(self, loan_id):
      pass

   @abstractmethod
   def calculate_emi(self, loan_id):
      pass

   @abstractmethod
   def calculate_emi_with_params(self, principal_amount, interest_rate, loan_term):
      pass

   @abstractmethod
   def loan_repayment(self, loan_id, amount):
      pass

   @abstractmethod
   def get_all_loan(self):
      pass

   @abstractmethod
   def get_loan_by_id(self, loan_id):
      pass
```

Directory - loan_management/dao/iloan_repository_impl.py

```
import pymysql
from dao.iloan_repository import ILoanRepository
from util.db_conn_util import DBConnUtil
from entity.home_loan import HomeLoan
from entity.car_loan import CarLoan
from exception.invalid_loan_exception import InvalidLoanException

class LoanRepositoryImpl(ILoanRepository):

    def __init__(self):
        self.conn = DBConnUtil.get_connection('db.properties')

    def apply_loan(self, loan):
        try:
            cursor = self.conn.cursor()
            confirm = input("Do you want to proceed with applying the loan? (Yes/No): 
").strip().lower()
            if confirm != 'yes':
                print("Loan application cancelled.")
                return

            insert_loan_sql = """
                INSERT INTO loan (customer_id, principal_amount, interest_rate, 
loan_term, loan_type, loan_status)
                VALUES (%s, %s, %s, %s, %s, %s)
            """
            loan_data = (
                loan.customer.customer_id,
                loan.principal_amount,
                loan.interest_rate,
                loan.loan_term,
                loan.loan_type,
                loan.loan_status
            )
            cursor.execute(insert_loan_sql, loan_data)
            self.conn.commit()

            loan_id = cursor.lastrowid
            loan.loan_id = loan_id

            if isinstance(loan, HomeLoan):
                cursor.execute("""
                    INSERT INTO home_loan (loan_id, property_address, property_value)
                    VALUES (%s, %s, %s)
                """, (loan_id, loan.property_address, loan.property_value))
            elif isinstance(loan, CarLoan):
                cursor.execute("""
                    INSERT INTO car_loan (loan_id, car_model, car_value)
                    VALUES (%s, %s, %s)
                """, (loan_id, loan.car_model, loan.car_value))

            self.conn.commit()
            print(f"Loan applied successfully. Loan ID: {loan_id}")

        except Exception as e:
            print("Error in apply_loan:", e)
            self.conn.rollback()
```

```
    def calculate_interest(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT principal_amount, interest_rate, loan_term FROM
loan WHERE loan_id = %s", (loan_id,))
            result = cursor.fetchone()

            if result:
                principal, rate, term = result
                interest = (principal * rate * term) / 12
                print(f"Calculated Interest for Loan ID {loan_id}: ₹{interest:.2f}")
                return interest
            else:
                raise InvalidLoanException(f"Loan with ID {loan_id} not found.")

        except InvalidLoanException as e:
            print("Invalid", e)
        except Exception as e:
            print("Error calculating interest:", e)

    def calculate_interest_with_params(self, principal_amount, interest_rate,
loan_term):
        try:
            interest = (principal_amount * interest_rate * loan_term) / 12
            print(f"Calculated Interest (manual): ₹{interest:.2f}")
            return interest
        except Exception as e:
            print("Error calculating interest with parameters:", e)

    def loan_status(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("""
                SELECT c.credit_score FROM loan l
                JOIN customer c ON l.customer_id = c.customer_id
                WHERE l.loan_id = %s
            """, (loan_id,))
            result = cursor.fetchone()

            if result:
                credit_score = result[0]
                new_status = 'Approved' if credit_score > 650 else 'Rejected'
            cursor.execute("UPDATE loan SET loan_status = %s WHERE loan_id =
%s", (new_status, loan_id))
            self.conn.commit()

            print(f"Loan ID {loan_id} has been {new_status.lower()} based on credit
score {credit_score}.")
         else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not found.")

      except InvalidLoanException as e:
         print("Invalid", e)
      except Exception as e:
         print("Error updating loan status:", e)

   def calculate_emi(self, loan_id):
      try:
         cursor = self.conn.cursor()
         cursor.execute("SELECT principal_amount, interest_rate, loan_term FROM
loan WHERE loan_id = %s", (loan_id,))
         result = cursor.fetchone()

         if result:
            P, annual_rate, N = result
            R = annual_rate / 12 / 100
            EMI = (P * R * (1 + R)**N) / ((1 + R)**N - 1)
            print(f"Calculated EMI for Loan ID {loan_id}: ₹{EMI:.2f}")
            return EMI
         else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not found.")

      except InvalidLoanException as e:
         print("Invalid", e)
      except Exception as e:
         print("Error calculating EMI:", e)

   def calculate_emi_with_params(self, principal_amount, interest_rate, loan_term):
      try:
         R = interest_rate / 12 / 100
         N = loan_term
         EMI = (principal_amount * R * (1 + R)**N) / ((1 + R)**N - 1)
         print(f"Calculated EMI (manual): ₹{EMI:.2f}")
         return EMI
      except Exception as e:
         print("Error calculating EMI with parameters:", e)
```

```
    def loan_repayment(self, loan_id, amount):
        try:
            emi = self.calculate_emi(loan_id)
            if emi is None:
                return

            emi = float(emi)

            if amount < emi:
                print("Payment amount is less than a single EMI. Payment rejected.")
                return

            num_emis_paid = int(amount // emi)
            remaining_amount = amount % emi
            print(f" You can pay {num_emis_paid} EMI(s) with ₹{amount}. Remaining:
₹{remaining_amount:.2f}")

        except Exception as e:
            print("Error processing loan repayment:", e)

    def get_all_loan(self):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT * FROM loan")
            results = cursor.fetchall()

            if results:
                for row in results:
                    print(row)
            else:
                print("No loans found.")

        except Exception as e:
            print("Error fetching all loans:", e)

    def get_loan_by_id(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT * FROM loan WHERE loan_id = %s", (loan_id,))
            result = cursor.fetchone()

            if result:
                print("Loan Details:", result)
                return result
            else:
                raise InvalidLoanException(f"Loan with ID {loan_id} not found.")

        except InvalidLoanException as e:
            print("Invalid", e)
        except Exception as e:
            print("Error fetching loan by ID:", e)
```

Entity Files

Directory - loan_management/ entity/customer.py

```
class Customer:
    def __init__(self, customer_id=None, name='', email_address='',
phone_number='', address='', credit_score=0):
        self.customer_id = customer_id
        self.name = name
        self.email_address = email_address
        self.phone_number = phone_number
        self.address = address
        self.credit_score = credit_score

    def __str__(self):
        return f"Customer[ID={self.customer_id}, Name={self.name},
Email={self.email_address}, " \
               f"Phone={self.phone_number}, Address={self.address},
CreditScore={self.credit_score}]"
```

Directory - loan_management/ entity/loan.py

```
from entity.customer import Customer

class Loan:
    def __init__(self, loan_id=None, customer=None, principal_amount=0.0,
                 interest_rate=0.0, loan_term=0, loan_type='', loan_status='Pending'):
        self.loan_id = loan_id
        self.customer = customer
        self.principal_amount = principal_amount
        self.interest_rate = interest_rate
        self.loan_term = loan_term
        self.loan_type = loan_type
        self.loan_status = loan_status

    def __str__(self):
        return f"Loan[ID={self.loan_id}, CustomerID={self.customer.customer_id if 
self.customer else 'N/A'}, " \
               f"Principal={self.principal_amount}, InterestRate={self.interest_rate}, 
Term={self.loan_term}, " \
               f"Type={self.loan_type}, Status={self.loan_status}]"
```

Directory - loan_management/ entity/car_loan.py

```
from entity.loan import Loan

class CarLoan(Loan):
    def __init__(self, loan_id=None, customer=None, principal_amount=0.0, 
interest_rate=0.0,
                 loan_term=0, loan_status='Pending', car_model='', car_value=0):
        super().__init__(loan_id, customer, principal_amount, interest_rate, loan_term, 
'CarLoan', loan_status)
        self.car_model = car_model
        self.car_value = car_value

    def __str__(self):
        return super().__str__() + f", CarModel={self.car_model}, 
CarValue={self.car_value}"
```

Directory - loan_management/ entity/home_loan.py

```
from entity.loan import Loan

class HomeLoan(Loan):
    def __init__(self, loan_id=None, customer=None, principal_amount=0.0, 
interest_rate=0.0,
                 loan_term=0, loan_status='Pending', property_address='', 
property_value=0):
        super().__init__(loan_id, customer, principal_amount, interest_rate, loan_term, 
'HomeLoan', loan_status)
        self.property_address = property_address
        self.property_value = property_value

    def __str__(self):
        return super().__str__() + f", PropertyAddress={self.property_address},
PropertyValue={self.property_value}"
```

Exception File

Directory - loan_management/ exception/__init__.py

# Empty File

Directory - loan_management/ exception/invalid_loan_exception.py

```
class InvalidLoanException(Exception):
    def __init__(self, message="Invalid Loan! Please check the loan ID or data."):
        super().__init__(message)
```

Util File

Directory - loan_management/ util/db_conn_util.py

```
import pymysql
import os
from util.db_property_util import DBPropertyUtil

class DBConnUtil:
    @staticmethod
    def get_connection(prop_file_name):
        try:
            # Get absolute path to the project root
            base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
            prop_path = os.path.join(base_dir, prop_file_name)

            props = DBPropertyUtil.get_property_value(prop_path)
            connection = pymysql.connect(
                host=props['host'],
                user=props['user'],
                password=props['password'],
                database=props['database']
            )
            return connection
        except Exception as e:
            print("Error while connecting to DB:", e)
            return None
```

Directory - loan_management/ util/db_property_util.py

```
import configparser
import os

class DBPropertyUtil:
    @staticmethod
    def get_property_value(file_name):
        print(f" Trying to load: {file_name}")
        if not os.path.exists(file_name):
            print(" File not found at given path.")
        else:
            with open(file_name, 'r') as f:
                print(" Contents of db.properties:")
                print(f.read())

        config = configparser.ConfigParser()
        config.read(file_name)

        return {
            'host': config.get('mysql', 'host'),
            'user': config.get('mysql', 'user'),
            'password': config.get('mysql', 'password'),
            'database': config.get('mysql', 'database')
        }
```

Directory - loan_management/ util/test_db_connection.py

```
from util.db_conn_util import DBConnUtil
import os
print(" CWD:", os.getcwd())

conn = DBConnUtil.get_connection('./db.properties')
if conn:
    print(" Connection successful!")
else:
    print(" Connection failed.")
```

Directory - loan_management/ util/db.properties

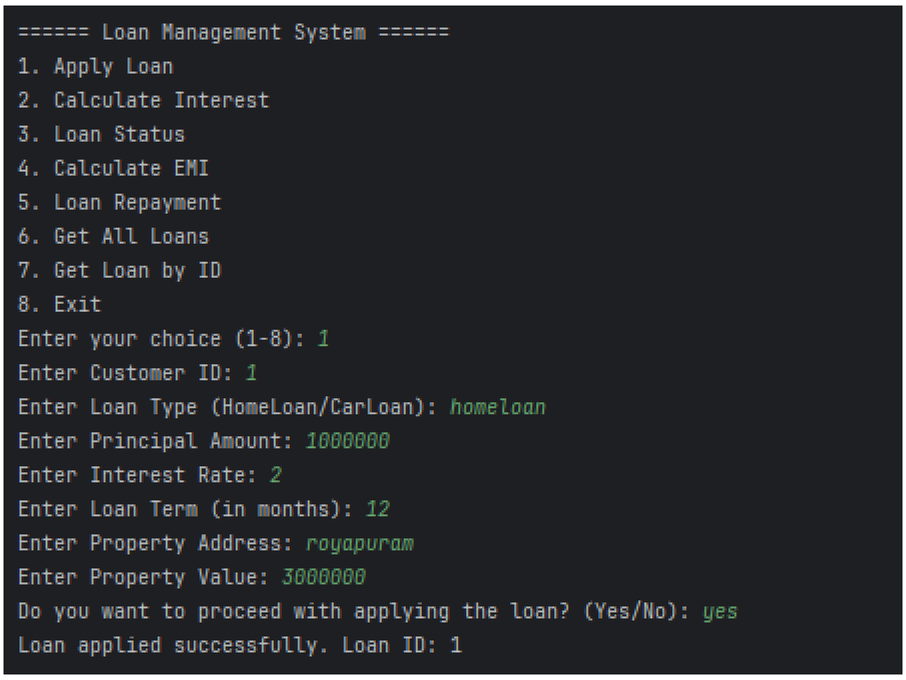
```
[mysql]
host = localhost
user = root
password = root
database = loan_management
```

Main File

Directory - loan_management/ main/main_module.py

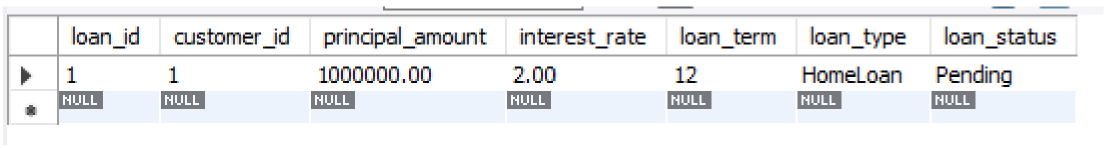
```
from dao.loan_repository_impl import LoanRepositoryImpl
from entity.customer import Customer
from entity.home_loan import HomeLoan
from entity.car_loan import CarLoan

def main():
    service = LoanRepositoryImpl()

    while True:
        print("\n====== Loan Management System ======")
        print("1. Apply Loan")
        print("2. Calculate Interest")
        print("3. Loan Status")
        print("4. Calculate EMI")
        print("5. Loan Repayment")
        print("6. Get All Loans")
        print("7. Get Loan by ID")
        print("8. Exit")
        choice = input("Enter your choice (1-8): ")

        if choice == '1':
            customer = Customer(customer_id=int(input("Enter Customer ID: ")))
            loan_type = input("Enter Loan Type (HomeLoan/CarLoan): ").strip()

            principal = float(input("Enter Principal Amount: "))
            rate = float(input("Enter Interest Rate: "))
            term = int(input("Enter Loan Term (in months): "))
    if loan_type.lower() == 'homeloan':
        prop_addr = input("Enter Property Address: ")
        prop_val = int(input("Enter Property Value: "))
        loan = HomeLoan(customer=customer, principal_amount=principal,
                        interest_rate=rate, loan_term=term, loan_status='Pending',
                        property_address=prop_addr, property_value=prop_val)
    elif loan_type.lower() == 'carloan':
        car_model = input("Enter Car Model: ")
        car_val = int(input("Enter Car Value: "))
        loan = CarLoan(customer=customer, principal_amount=principal,
                       interest_rate=rate, loan_term=term, loan_status='Pending',
                       car_model=car_model, car_value=car_val)
    else:
        print(" Invalid Loan Type!")
        continue

    service.apply_loan(loan)

elif choice == '2':
    sub_choice = input("1. By Loan ID\n2. By Manual Entry\nChoose: ")
    if sub_choice == '1':
        loan_id = int(input("Enter Loan ID: "))
        service.calculate_interest(loan_id)
    elif sub_choice == '2':
        principal = float(input("Enter Principal Amount: "))
        rate = float(input("Enter Interest Rate: "))
        term = int(input("Enter Loan Term (in months): "))
        service.calculate_interest_with_params(principal, rate, term)

elif choice == '3':
    loan_id = int(input("Enter Loan ID to check and update status: "))
    service.loan_status(loan_id)

elif choice == '4':
    sub_choice = input("1. By Loan ID\n2. By Manual Entry\nChoose: ")
    if sub_choice == '1':
        loan_id = int(input("Enter Loan ID: "))
        service.calculate_emi(loan_id)
    elif sub_choice == '2':
        principal = float(input("Enter Principal Amount: "))
        rate = float(input("Enter Interest Rate: "))
        term = int(input("Enter Loan Term (in months): "))
        service.calculate_emi_with_params(principal, rate, term)
        elif choice == '5':
            loan_id = int(input("Enter Loan ID: "))
            amount = float(input("Enter repayment amount: "))
            service.loan_repayment(loan_id, amount)

        elif choice == '6':
            service.get_all_loan()

        elif choice == '7':
            loan_id = int(input("Enter Loan ID: "))
            service.get_loan_by_id(loan_id)

        elif choice == '8':
            print("Exiting Loan Management System. Goodbye!")
            break
        else:
            print(" Invalid choice. Please try again.")

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

Output while choosing 1:



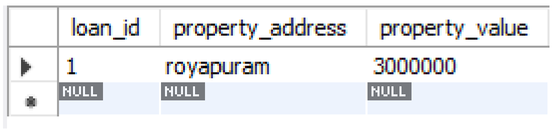
```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 1
Enter Customer ID: 1
Enter Loan Type (HomeLoan/CarLoan): homeloan
Enter Principal Amount: 1000000
Enter Interest Rate: 2
Enter Loan Term (in months): 12
Enter Property Address: royapuram
Enter Property Value: 3000000
Do you want to proceed with applying the loan? (Yes/No): yes
Loan applied successfully. Loan ID: 1
```

select * from loan;



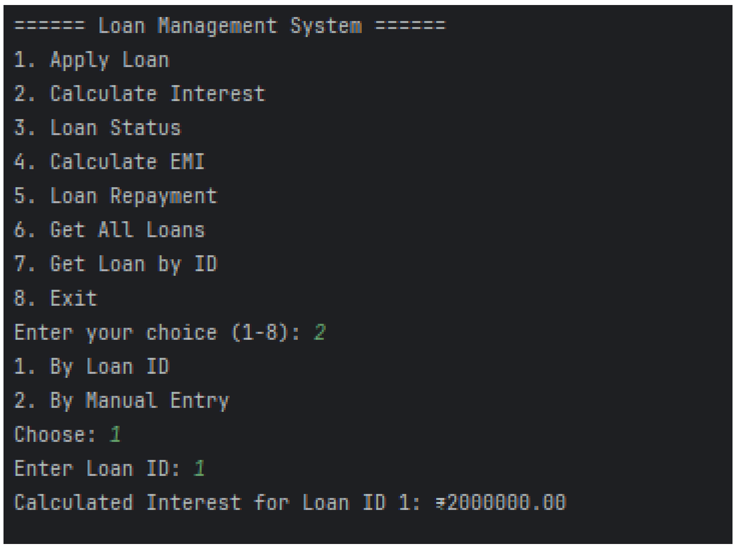
	loan_id	customer_id	principal_amount	interest_rate	loan_term	loan_type	loan_status
▶	1	1	1000000.00	2.00	12	HomeLoan	Pending
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

select * from home_loan;



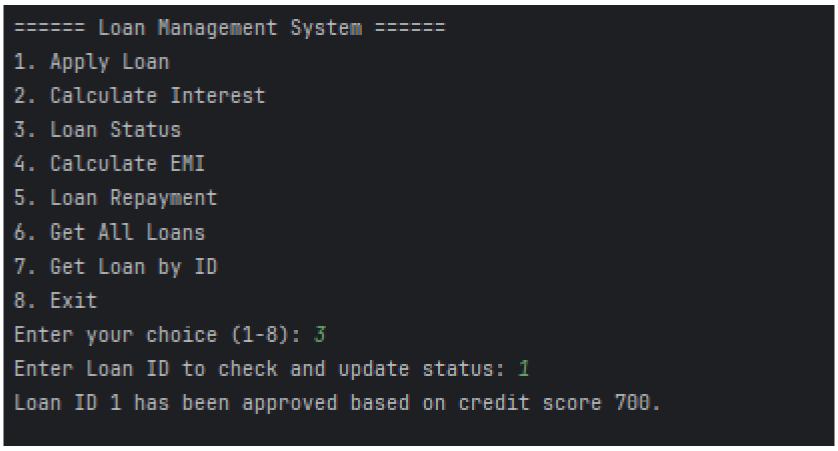
	loan_id	property_address	property_value
▶	1	royapuram	3000000
*	NULL	NULL	NULL

Output while choosing 2:



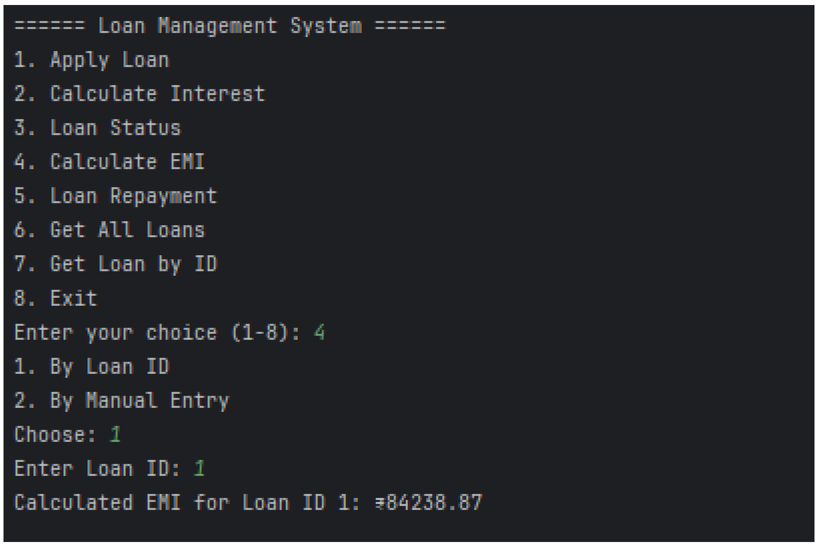
```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 2
1. By Loan ID
2. By Manual Entry
Choose: 1
Enter Loan ID: 1
Calculated Interest for Loan ID 1: ₹2000000.00
```

Output while choosing 3:



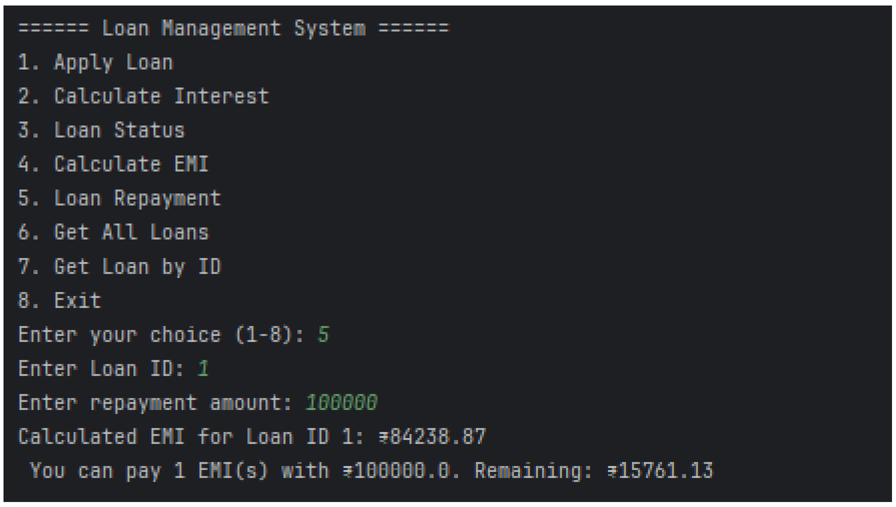
```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 3
Enter Loan ID to check and update status: 1
Loan ID 1 has been approved based on credit score 700.
```

Output while choosing 4:



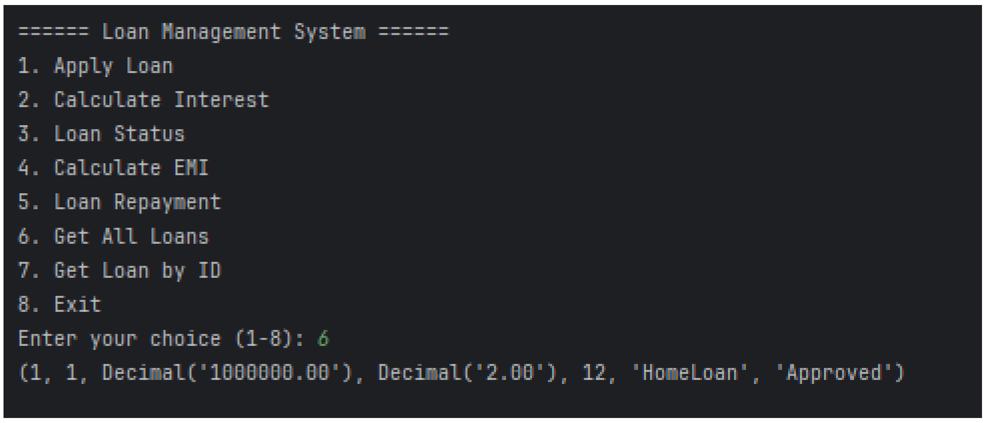
```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 4
1. By Loan ID
2. By Manual Entry
Choose: 1
Enter Loan ID: 1
Calculated EMI for Loan ID 1: ₹84238.87
```

Output while choosing 5:



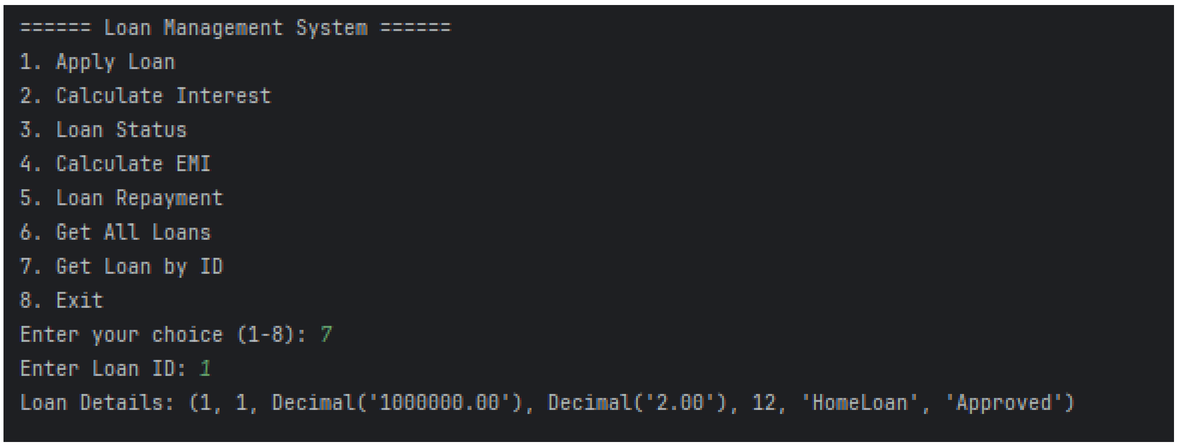
```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 5
Enter Loan ID: 1
Enter repayment amount: 100000
Calculated EMI for Loan ID 1: ₹84238.87
 You can pay 1 EMI(s) with ₹100000.0. Remaining: ₹15761.13
```

Output while choosing 6:



```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 6
(1, 1, Decimal('1000000.00'), Decimal('2.00'), 12, 'HomeLoan', 'Approved')
```

Output while choosing 7:



```
====== Loan Management System ======
1. Apply Loan
2. Calculate Interest
3. Loan Status
4. Calculate EMI
5. Loan Repayment
6. Get All Loans
7. Get Loan by ID
8. Exit
Enter your choice (1-8): 7
Enter Loan ID: 1
Loan Details: (1, 1, Decimal('1000000.00'), Decimal('2.00'), 12, 'HomeLoan', 'Approved')
```