Source code

Source code

```
import os
import sys
import mysql.connector
from mysql.connector import Error
from datetime import datetime, timedelta
# -------------------------
# Database connection setup
# -------------------------
DB_CONFIG = {
    "host": "localhost",
    "user": "root",
    "password": "dbms",
    "database": "atm"
}

try:
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()
    print("✅ Database connected successfully!")
except Error as e:
    print("❌ Database connection error:", e)
    sys.exit(1)


# -------------------------
# Utility functions
# -------------------------
def clear_screen():
    os.system('cls')
def center_text(text, width=70):
    return str(text).center(width)
def pause(msg="Press Enter to continue..."):
    input(msg)
# -------------------------
# Admin functions
# -------------------------
def admin_login():
    clear_screen()
    print(center_text("--- ADMIN LOGIN ---"))
    admin_id = input("Enter Admin ID: ").strip()
    admin_password = input("Enter Password: ").strip()
    try:
        cursor.execute("SELECT * FROM admins WHERE admin_id=%s AND password=%s", (admin_id,
admin_password))
        result = cursor.fetchone()
    except Error as e:
        print("DB error:", e)
        pause()
```

```
        return

    if result:
        print("✅ Admin login successful.")
        pause()
        admin_menu()
    else:
        print("❌ Invalid Admin credentials.")
        pause()

def admin_menu():
    while True:
        clear_screen()
        print("\n" + "=" * 70)
        print(center_text("--- ADMIN MENU ---"))
        print("=" * 70)
        print(center_text("1. Create Account 🆕"))
        print(center_text("2. View All Users 📋"))
        print(center_text("3. Update User ✏️"))
        print(center_text("4. Delete User 🗑️"))
        print(center_text("5. View Transactions 🧾"))
        print(center_text("6. Back 🔙"))
        print("=" * 70)

        choice = input(center_text("Enter your choice: ")).strip()

        if choice == '1':
            create_account_admin()
        elif choice == '2':
            view_all_users()
        elif choice == '3':
            update_user()
        elif choice == '4':
            delete_user()
        elif choice == '5':
            admin_view_transactions()
        elif choice == '6':
            break
        else:
            print("❌ Invalid choice.")
            pause()
def create_account_admin():
    clear_screen()
    print(center_text("--- CREATE NEW ACCOUNT (Admin) ---"))
    while True:
        acc_no = input("Enter new Account Number: ").strip()
        if acc_no == "":
            print("Account number cannot be blank.")
            continue
        # check if already exists
        cursor.execute("SELECT acc_no FROM users WHERE acc_no=%s", (acc_no,))
        if cursor.fetchone():
            print("Account number already exists. Choose another.")
            continue
```

```
        break
    name = input("Enter Full Name: ").strip()
    while True:
        pin = input("Enter 4-digit PIN: ").strip()
        if not (pin.isdigit() and len(pin) == 4):
            print("PIN must be exactly 4 digits.")
            continue
        break
    phone_no = input("Enter phone number (optional): ").strip()
    address = input("Enter address (optional): ").strip()
    try:
        cursor.execute(
            "INSERT INTO users (acc_no, name, pin, balance, phone_no, address) VALUES (%s, %s, %s, %s, %s, %s)",
            (acc_no, name, pin, 0.00, phone_no if phone_no else None, address if address else None)
        )
        conn.commit()
        print("✅ Account created successfully with Rs.0 balance.")
    except Error as e:
        conn.rollback()
        print("❌ Failed to create account:", e)
    pause()
def view_all_users():
    clear_screen()
    try:
        cursor.execute("SELECT acc_no, name, phone_no, address, balance FROM users")
        users = cursor.fetchall()
    except Error as e:
        print("DB error:", e)
        pause()
        return
    # Column widths
    w_acc = 14
    w_name = 25
    w_phone = 15
    w_addr = 28
    w_bal = 12

    total = w_acc + w_name + w_phone + w_addr + w_bal + 13
    print("\n" + "-" * total)
    print(center_text("--- USER LIST ---", total))
    print("-" * total)

    header = "| {0:<14} | {1:<25} | {2:<15} | {3:<28} | {4:<12} |".format(
        "Account No", "Name", "Phone", "Address", "Balance (Rs)")
    print(header)
    print("-" * total)

    for u in users:
        acc = str(u[0])[:w_acc - 1]
        name = str(u[1])[:w_name - 1]
        phone = (str(u[2]) if u[2] is not None else "")[:w_phone - 1]
        addr = (str(u[3]) if u[3] is not None else "")[:w_addr - 1]
        bal = f"{u[4]:.2f}"
        row = "| {0:<14} | {1:<25} | {2:<15} | {3:<28} | {4:<12} |".format(acc, name, phone, addr, bal)
```

```
        print(row)

    print("-" * total)
    pause()


def update_user():
    clear_screen()
    print(center_text("--- UPDATE USER DETAILS ---"))
    acc_no = input("Enter Account Number to update: ").strip()
    try:
        cursor.execute("SELECT acc_no, name, phone_no, address FROM users WHERE acc_no=%s", (acc_no,))
        user = cursor.fetchone()
    except Error as e:
        print("DB error:", e)
        pause()
        return

    if not user:
        print("No user found with that account number.")
        pause()
        return

    print(f"Current Name : {user[1]}")
    print(f"Current Phone: {user[2] if user[2] else ''}")
    print(f"Current Addr : {user[3] if user[3] else ''}\n")

    new_name = input("Enter new name (leave blank to keep current): ").strip()
    new_phone = input("Enter new phone (leave blank to keep current): ").strip()
    new_addr = input("Enter new address (leave blank to keep current): ").strip()

    # Use current values if blank
    if new_name == "":
        new_name = user[1]
    if new_phone == "":
        new_phone = user[2]
    if new_addr == "":
        new_addr = user[3]

    try:
        cursor.execute(
            "UPDATE users SET name=%s, phone_no=%s, address=%s WHERE acc_no=%s",
            (new_name, new_phone, new_addr, acc_no)
        )
        conn.commit()
        print("✅ User updated successfully.")
    except Error as e:
        conn.rollback()
        print("❌ Update failed:", e)
    pause()


def delete_user():
    clear_screen()
```

```
    print(center_text("--- DELETE USER ---"))
    acc_no = input("Enter Account Number to delete: ").strip()
    if acc_no == "":
        print("Account number cannot be blank.")
        pause()
        return
    confirmation = input(f"Type 'YES' to permanently delete account {acc_no}: ").strip()
    if confirmation != 'YES':
        print("Deletion cancelled.")
        pause()
        return
    try:
        cursor.execute("DELETE FROM users WHERE acc_no=%s", (acc_no,))
        conn.commit()
        if cursor.rowcount == 0:
            print("No user deleted — account not found.")
        else:
            print("🗑️ User deleted successfully.")
    except Error as e:
        conn.rollback()
        print("❌ Delete failed:", e)
    pause()


def admin_view_transactions():
    clear_screen()
    print(center_text("--- ALL TRANSACTIONS (Latest 100) ---"))
    try:
        cursor.execute("SELECT txn_id, acc_no, txn_type, amount, txn_date FROM transactions ORDER BY txn_date
DESC LIMIT 100")
        txns = cursor.fetchall()
    except Error as e:
        print("DB error:", e)
        pause()
        return

    print(f"{'Txn_ID':<8} {'Account':<14} {'Date':<20} {'Type':<10} {'Amount (Rs)':<12}")
    print("-" * 68)
    for t in txns:
        txn_id = str(t[0])
        acc = str(t[1])
        typ = str(t[2])
        amt = f"{t[3]:.2f}"
        dt = t[4].strftime("%Y-%m-%d %H:%M:%S") if isinstance(t[4], datetime) else str(t[4])
        print(f"{txn_id:<8} {acc:<14} {dt:<20} {typ:<10} {amt:<12}")
    print("-" * 68)
    pause()


# -------------------------
# User functions
# -------------------------
def user_login():
    clear_screen()
```

```
    print(center_text("--- USER LOGIN ---"))
    acc_no = input("Enter Account Number: ").strip()
    pin = input("Enter PIN: ").strip()
    try:
        cursor.execute("SELECT acc_no, name FROM users WHERE acc_no=%s AND pin=%s", (acc_no, pin))
        result = cursor.fetchone()
    except Error as e:
        print("DB error:", e)
        pause()
        return

    if result:
        print(f"✅ Welcome, {result[1]}!")
        pause()
        atm_menu(acc_no)
    else:
        print("❌ Invalid account or PIN.")
        pause()


def atm_menu(acc_no):
    while True:
        clear_screen()
        print("\n" + "=" * 70)
        print(center_text("--- ATM MENU ---"))
        print("=" * 70)
        print(center_text("1. Check Balance 💰"))
        print(center_text("2. Deposit 💵"))
        print(center_text("3. Withdraw 💸"))
        print(center_text("4. Change PIN 🔑"))
        print(center_text("5. View Transactions 🧾"))
        print(center_text("6. Fixed Deposit (FD) 🏦"))
        print(center_text("7. Logout 🔙"))
        print("=" * 70)

        choice = input(center_text("Enter your choice: ")).strip()

        if choice == '1':
            check_balance(acc_no)
        elif choice == '2':
            deposit_amount(acc_no)
        elif choice == '3':
            withdraw_amount(acc_no)
        elif choice == '4':
            change_pin(acc_no)
        elif choice == '5':
            view_transactions(acc_no)
        elif choice == '6':
            fd_menu(acc_no)
        elif choice == '7':
            print(center_text("👋 Logged out successfully."))
            pause()
            break
        else:
```

```
            print(center_text("❌ Invalid choice."))
            pause()


def check_balance(acc_no):
    try:
        cursor.execute("SELECT balance FROM users WHERE acc_no=%s", (acc_no,))
        res = cursor.fetchone()
        if not res:
            print("Account not found.")
            pause()
            return
        balance = res[0]
        print(center_text(f"Your current balance is: Rs.{balance:.2f}"))
    except Error as e:
        print("DB error:", e)
    pause()


def deposit_amount(acc_no):
    try:
        amount_str = input("Enter amount to deposit: ").strip()
        amount = float(amount_str)
        if amount <= 0:
            print("Amount must be positive.")
            pause()
            return
    except ValueError:
        print("Invalid amount.")
        pause()
        return

    try:
        cursor.execute("UPDATE users SET balance = balance + %s WHERE acc_no=%s", (amount, acc_no))
        cursor.execute("INSERT INTO transactions (acc_no, txn_type, amount) VALUES (%s, 'Deposit', %s)", (acc_no,
amount))
        conn.commit()
        print(f"✅ Rs.{amount:.2f} deposited successfully.")
    except Error as e:
        conn.rollback()
        print("❌ Transaction failed:", e)
    pause()


def withdraw_amount(acc_no):
    try:
        amount_str = input("Enter amount to withdraw: ").strip()
        amount = float(amount_str)
        if amount <= 0:
            print("Amount must be positive.")
            pause()
            return
    except ValueError:
        print("Invalid amount.")
```

```
        pause()
        return

    try:
        cursor.execute("SELECT balance FROM users WHERE acc_no=%s", (acc_no,))
        row = cursor.fetchone()
        if not row:
            print("Account not found.")
            pause()
            return
        current_balance = float(row[0])
    except Error as e:
        print("DB error:", e)
        pause()
        return

    if amount <= current_balance:
        try:
            cursor.execute("UPDATE users SET balance = balance - %s WHERE acc_no=%s", (amount, acc_no))
            cursor.execute("INSERT INTO transactions (acc_no, txn_type, amount) VALUES (%s, 'Withdraw', %s)",
(acc_no, amount))
            conn.commit()
            print(f"✅ Rs.{amount:.2f} withdrawn successfully.")
        except Error as e:
            conn.rollback()
            print("❌ Withdrawal failed:", e)
    else:
        print("⚠️ Insufficient balance.")
    pause()


def change_pin(acc_no):
    while True:
        new_pin = input("Enter new 4-digit PIN: ").strip()
        if not (new_pin.isdigit() and len(new_pin) == 4):
            print("PIN must be exactly 4 digits.")
            continue
        confirm = input("Confirm new PIN: ").strip()
        if new_pin != confirm:
            print("PINs do not match.")
            continue
        break
    try:
        cursor.execute("UPDATE users SET pin=%s WHERE acc_no=%s", (new_pin, acc_no))
        conn.commit()
        print("✅ PIN changed successfully.")
    except Error as e:
        conn.rollback()
        print("❌ Failed to change PIN:", e)
    pause()


def view_transactions(acc_no):
    clear_screen()
```

```
    print(center_text("--- Recent Transactions ---"))
    try:
        cursor.execute(
            "SELECT txn_id, txn_date, txn_type, amount FROM transactions WHERE acc_no=%s ORDER BY txn_date
DESC LIMIT 20",
            (acc_no,)
        )
        txns = cursor.fetchall()
    except Error as e:
        print("DB error:", e)
        pause()
        return

    if not txns:
        print("No transactions found.")
        pause()
        return

    # Pretty print
    print(f"{'Txn_ID':<8} {'Date':<20} {'Type':<12} {'Amount (Rs)':<12}")
    print("-" * 58)
    for t in txns:
        txn_id = str(t[0])
        dt = t[1].strftime("%Y-%m-%d %H:%M:%S") if isinstance(t[1], datetime) else str(t[1])
        typ = str(t[2])
        amt = f"{t[3]:.2f}"
        print(f"{txn_id:<8} {dt:<20} {typ:<12} {amt:<12}")
    print("-" * 58)
    pause()

# -------------------------
# Fixed Deposit (FD) functions
# -------------------------
FD_RATE = 6.5  # percent per annum (predefined)
FD_MIN_AMOUNT = 1000.0
FD_OPTIONS_MONTHS = { '1': 6, '2': 12, '3': 24 }  # choices

def fd_menu(acc_no):
    while True:
        clear_screen()
        print(center_text("--- FIXED DEPOSIT (FD) ---"))
        print(center_text("1. Create FD"))
        print(center_text("2. View My FDs"))
        print(center_text("3. Back"))
        choice = input(center_text("Enter your choice: ")).strip()
        if choice == '1':
            create_fd(acc_no)
        elif choice == '2':
            view_fd(acc_no)
        elif choice == '3':
            break
        else:
            print("Invalid choice.")
            pause()
```

```
def create_fd(acc_no):
    clear_screen()
    print(center_text("--- CREATE FIXED DEPOSIT ---"))
    try:
        amount_str = input("Enter FD amount (minimum Rs.1000): ").strip()
        amount = float(amount_str)
    except ValueError:
        print("Invalid amount.")
        pause()
        return

    if amount < FD_MIN_AMOUNT:
        print(f"Minimum FD amount is Rs.{FD_MIN_AMOUNT:.2f}")
        pause()
        return

    print("Choose FD tenure (predefined):")
    print("1) 6 months")
    print("2) 12 months")
    print("3) 24 months")
    choice = input("Enter option (1/2/3): ").strip()
    if choice not in FD_OPTIONS_MONTHS:
        print("Invalid option.")
        pause()
        return

    months = FD_OPTIONS_MONTHS[choice]
    years = months / 12.0
    rate = FD_RATE
    # simple interest
    interest = (amount * rate * years) / 100.0
    maturity_amount = amount + interest

    start_dt = datetime.now()
    mature_dt = start_dt + timedelta(days=30 * months)  # approximate, fine for demo

    try:
        cursor.execute(
            "INSERT INTO fd_accounts (account_no, amount, rate, start_date, mature_date, status) VALUES (%s, %s, %s,
%s, %s, %s)",
            (acc_no, round(amount, 2), rate, start_dt, mature_dt, 'OPEN')
        )

        cursor.execute(
            "INSERT INTO transactions (acc_no, amount, txn_type, txn_date) VALUES (%s,%s,'FD',NOW())",
            (acc_no, amount)
        )



        conn.commit()
```

```
        # reduce user available balance by FD amount
        cursor.execute("UPDATE users SET balance = balance - %s WHERE acc_no=%s", (amount, acc_no))
        conn.commit()
        print("✅ FD created successfully.")
        print(f"Principal: Rs.{amount:.2f}")
        print(f"Tenure : {months} months")
        print(f"Rate   : {rate:.2f}% p.a. (simple interest)")
        print(f"Maturity Amount (approx): Rs.{maturity_amount:.2f}")
        print(f"Matures on: {mature_dt.strftime('%Y-%m-%d %H:%M:%S')}")
    except Error as e:
        conn.rollback()
        print("❌ Failed to create FD:", e)
    pause()
def view_fd(acc_no):
    clear_screen()
    print(center_text("--- MY FIXED DEPOSITS ---"))
    try:
        cursor.execute(
            "SELECT fd_id, amount, rate, start_date, mature_date, status FROM fd_accounts WHERE account_no=%s
ORDER BY start_date DESC",
            (acc_no,)
        )
        fds = cursor.fetchall()
    except Error as e:
        print("DB error:", e)
        pause()
        return

    if not fds:
        print("No FDs found.")
        pause()
        return

    print(f"{'FD_ID':<6} {'Amount (Rs)':<14} {'Rate%':<7} {'Start Date':<20} {'Mature Date':<20} {'Status':<8}")
    print("-" * 80)
    for f in fds:
        fd_id = str(f[0])
        amt = f"{f[1]:.2f}"
        rate = f"{f[2]:.2f}"
        sdt = f[3].strftime("%Y-%m-%d") if isinstance(f[3], datetime) else str(f[3])
        mdt = f[4].strftime("%Y-%m-%d") if isinstance(f[4], datetime) else str(f[4])
        status = f[5]
        print(f"{fd_id:<6} {amt:<14} {rate:<7} {sdt:<20} {mdt:<20} {status:<8}")
    print("-" * 80)
    pause()
# -------------------------
# Main menu (note: no create account here)
# -------------------------
def main_menu():
    while True:
        clear_screen()
        print("\n" + "=" * 70)
        print(center_text("🏦 WELCOME TO ATM SYSTEM 🏦"))
        print("=" * 70)
```

```
        print(center_text("1. Admin Login 👨‍💼"))
        print(center_text("2. User Login 👤"))
        print(center_text("3. Exit 🚪"))
        print("=" * 70)

        choice = input(center_text("Enter your choice: ")).strip()

        if choice == '1':
            admin_login()
        elif choice == '2':
            user_login()
        elif choice == '3':
            print(center_text("👋 Thank you for using our ATM system!"))
            # close DB
            try:
                cursor.close()
                conn.close()
            except:
                pass
            break
        else:
            print("❌ Invalid choice. Try again.")
            pause()
# -------------------------
# Run program
# -------------------------
if __name__ == "__main__":
    try:
        main_menu()
    except KeyboardInterrupt:
        print("\nExiting...")
    finally:
        try:
            cursor.close()
            conn.close()
        except:
            pass
```

SQL Table

SQL TABLES

```
mysql> use ATM;
Database changed
mysql> CREATE TABLE users (
    ->     acc_no VARCHAR(20) PRIMARY KEY,
    ->     name VARCHAR(50) NOT NULL,
    ->     pin VARCHAR(10) NOT NULL,
    ->     balance DECIMAL(10,2) DEFAULT 0.00,
    ->     phone_no VARCHAR(15),
    ->     address VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO users (acc_no, name, pin, balance, phone_no, address) VALUES
    -> ('AC1001', 'Priyanshu Supyal', '1234', 5000.00, NULL, NULL),
    -> ('AC1002', 'Manish Mishra', '5678', 3000.00, NULL, NULL),
    -> ('AC1003', 'Prathmesh Mehta', '4321', 7000.00, NULL, NULL);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from users;
+--------+-----------------+------+---------+----------+---------+
| acc_no | name            | pin  | balance | phone_no | address |
+--------+-----------------+------+---------+----------+---------+
| AC1001 | Priyanshu Supyal | 1234 | 5000.00 | NULL     | NULL    |
| AC1002 | Manish Mishra    | 5678 | 3000.00 | NULL     | NULL    |
| AC1003 | Prathmesh Mehta  | 4321 | 7000.00 | NULL     | NULL    |
+--------+-----------------+------+---------+----------+---------+
3 rows in set (0.01 sec)
```

```
mysql> CREATE TABLE admins (
    ->     admin_id VARCHAR(20) PRIMARY KEY,
    ->     password VARCHAR(20) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO admins (admin_id, password) VALUES
    -> ('admin1', 'admin123'),
    -> ('manager', 'atm2025');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from admins;
+----------+----------+
| admin_id | password |
+----------+----------+
| admin1   | admin123 |
| manager  | atm2025  |
+----------+----------+
2 rows in set (0.00 sec)
```

```
mysql> CREATE TABLE login_logs (
    ->     log_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     acc_no VARCHAR(20),
    ->     login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->     status ENUM('Success','Failure')
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc login_logs;
+------------+---------------------------+------+-----+-------------------+-------------------+
| Field      | Type                      | Null | Key | Default           | Extra             |
+------------+---------------------------+------+-----+-------------------+-------------------+
| log_id     | int                       | NO   | PRI | NULL              | auto_increment    |
| acc_no     | varchar(20)               | YES  |     | NULL              |                   |
| login_time | timestamp                 | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| status     | enum('Success','Failure') | YES  |     | NULL              |                   |
+------------+---------------------------+------+-----+-------------------+-------------------+
4 rows in set (0.01 sec)
```

```
mysql> CREATE TABLE transactions (
    ->     txn_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     acc_no VARCHAR(20),
    ->     txn_type VARCHAR(20) NOT NULL,
    ->     amount DECIMAL(10,2) NOT NULL,
    ->     txn_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->     FOREIGN KEY (acc_no) REFERENCES users(acc_no)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> DESC transactions;
+----------+---------------+------+-----+-------------------+-------------------+
| Field    | Type          | Null | Key | Default           | Extra             |
+----------+---------------+------+-----+-------------------+-------------------+
| txn_id   | int           | NO   | PRI | NULL              | auto_increment    |
| acc_no   | varchar(20)   | YES  | MUL | NULL              |                   |
| txn_type | varchar(20)   | NO   |     | NULL              |                   |
| amount   | decimal(10,2) | NO   |     | NULL              |                   |
| txn_date | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+----------+---------------+------+-----+-------------------+-------------------+
5 rows in set (0.04 sec)
```

```
mysql> CREATE TABLE IF NOT EXISTS fd_accounts (
    ->     fd_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     account_no VARCHAR(20),
    ->     amount DECIMAL(10,2),
    ->     rate FLOAT,
    ->     start_date DATETIME,
    ->     mature_date DATETIME,
    ->     status VARCHAR(10)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc fd_accounts;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| fd_id       | int           | NO   | PRI | NULL    | auto_increment |
| account_no  | varchar(20)   | YES  |     | NULL    |                |
| amount      | decimal(10,2) | YES  |     | NULL    |                |
| rate        | float         | YES  |     | NULL    |                |
| start_date  | datetime      | YES  |     | NULL    |                |
| mature_date | datetime      | YES  |     | NULL    |                |
| status      | varchar(10)   | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

```
mysql> SHOW TABLES;
+---------------+
| Tables_in_atm |
+---------------+
| admins        |
| fd_accounts   |
| login_logs    |
| transactions  |
| users         |
+---------------+
5 rows in set (0.02 sec)
```

---

function and module

Functions and Module

1. import os

Purpose

The os module allows Python to interact with the operating system.

Explanation

- os.system() executes system-level commands
- 'cls' → clears the screen in Windows

---

2. import sys

Purpose

The sys module provides access to Python runtime system features.

Explanation

- sys.exit(1) terminates the program immediately
- 1 indicates abnormal termination (error occurred)
- Used when database connection fails

---

3. import mysql.connector

Purpose

This module is used to connect Python with a MySQL database.

What it enables

- Establish database connection
- Execute SQL queries
- Fetch and manipulate record

---

4. from mysql.connector import Error

Purpose

To handle database-related exceptions.

Explanation

- Catches MySQL-specific errors
- Prevents program crash
- Displays meaningful error messages

---

5. from datetime import datetime, timedelta

Purpose

Used to handle date and time operations.

Practical ATM Uses

- Transaction timestamps
- Card expiry validation
- Daily withdrawal limits
- Session timeout handling

---

6. DB_CONFIG Dictionary

DB_CONFIG = { "host": "localhost", "user": "root", "password": "dbms", "database": "atm" }

Purpose

Stores database credentials in a single reusable object.

7. Utility Functions

clear_screen()

```
def clear_screen():
    os.system('cls' ')
```

✔ Clears terminal screen
✔ Improves user experience

---

center_text()

def center_text(text, width=70): return str(text).center(width)

✔ Aligns output neatly
✔ Enhances UI readability

---

pause()

def pause(msg="Press Enter to continue..."): input(msg)
✔ Stops execution until user input
✔ Used between menu transitions

8. cursor.execute()

What it does:

Executes a single SQL query (INSERT, SELECT, UPDATE, DELETE).

Syntax:

cursor.execute(query, values)

Example:

cursor.execute("SELECT * FROM customers WHERE id = %s", (101,))

---

9. cursor.fetchall()

What it does:

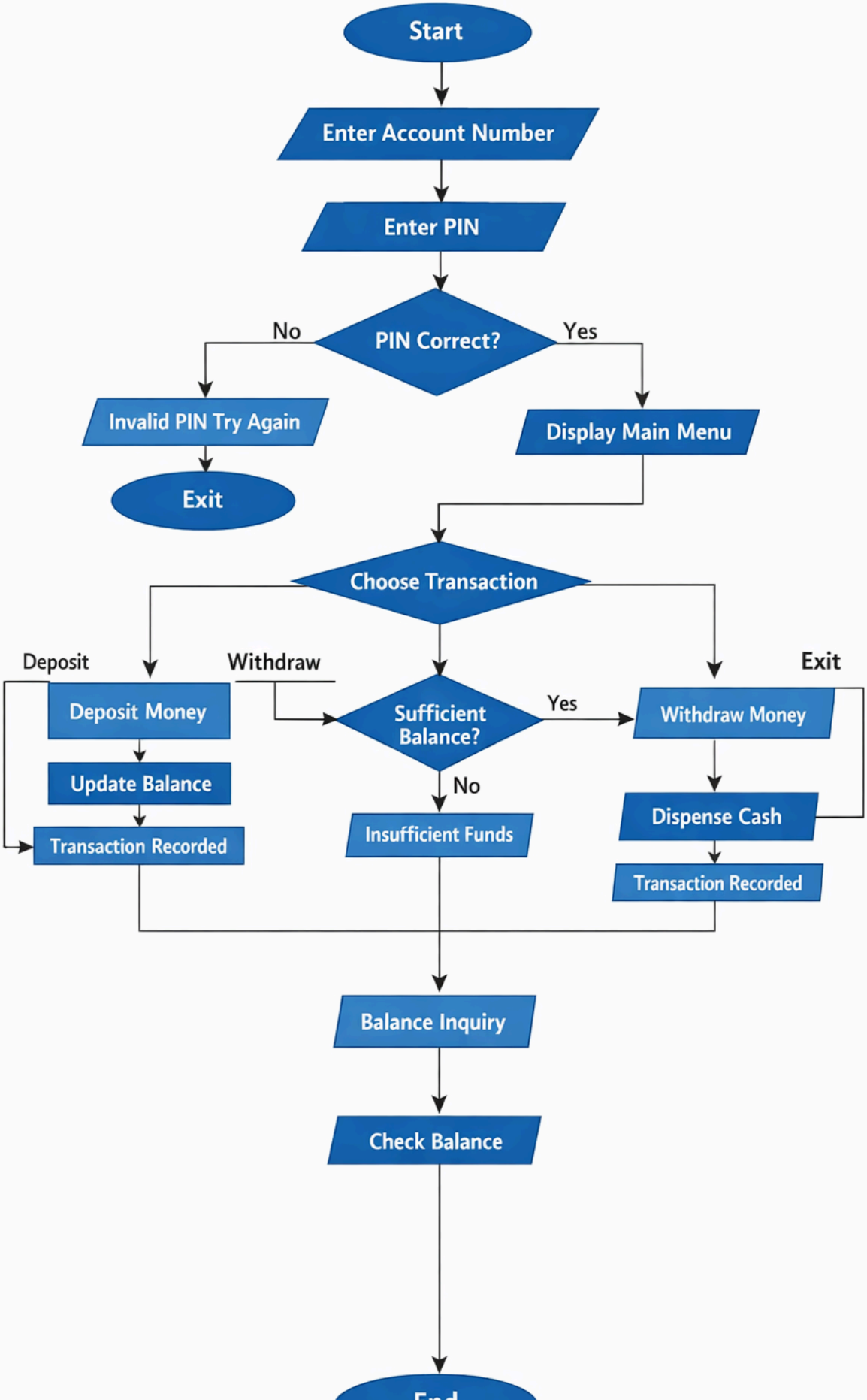
Fetches all rows returned by the last execute() query.

Returns:

A list of tuples.

FLOWCHART

ATM Management System Flowchart



output

OUTPUT

```
======================================================================
                    🏦 WELCOME TO ATM SYSTEM 🏦
======================================================================
                        1. Admin Login 👨‍💼
                        2. User Login 👤
                           3. Exit 🚪
======================================================================
                        Enter your choice:
```

```
                          --- ADMIN LOGIN ---
Enter Admin ID: admin1
Enter Password: admin123
✅ Admin login successful.
Press Enter to continue...
```

```
======================================================================
                          --- ADMIN MENU ---
======================================================================
                        1. Create Account 🆕
                        2. View All Users 📋
                        3. Update User ✏️
                        4. Delete User 🗑️
                        5. View Transactions 🧾
                             6. Back 🔙
======================================================================
                        Enter your choice:
```

```
                  --- CREATE NEW ACCOUNT (Admin) ---
Enter new Account Number: AC1004
Enter Full Name: DIVYANSHU
Enter 4-digit PIN: 9999
Enter phone number (optional): 9897198791
Enter address (optional): BIHAR
✅ Account created successfully with Rs.0 balance.
Press Enter to continue...
```

```
--- USER LIST ---
| Account No | Name | Phone | Address | Balance (Rs) |
| AC1001 | Priyanshu Supyal | | | 5000.00 |
| AC1002 | Manish Mishra | | | 3000.00 |
| AC1003 | Prathmesh Mehta | | | 7000.00 |
| AC1004 | DIVYANSHU | 9897198791 | BIHAR | 0.00 |
Press Enter to continue...
```

```
--- UPDATE USER DETAILS ---
Enter Account Number to update: AC1004
Current Name : DIVYANSHU
Current Phone: 9897198791
Current Addr : BIHAR

Enter new name (leave blank to keep current):
Enter new phone (leave blank to keep current):
Enter new address (leave blank to keep current): HALDWANI
✅ User updated successfully.
Press Enter to continue...
```

```
--- DELETE USER ---
Enter Account Number to delete: AC1004
Type 'YES' to permanently delete account AC1004: YES
🗑️User deleted successfully.
Press Enter to continue...
```

```
--- ALL TRANSACTIONS (Latest 100) ---
Txn_ID   Account        Date                 Type       Amount (Rs)
----------------------------------------------------------------------
6        ac1002         2025-12-29 18:10:30  Deposit    99999.00
5        AC1001         2025-12-29 18:08:08  FD         1000.00
4        AC1001         2025-12-29 18:05:22  Withdraw   99999.00
3        AC1001         2025-12-29 18:04:57  Deposit    100000.00
2        AC1001         2025-12-29 18:04:51  Deposit    1.00
----------------------------------------------------------------------
Press Enter to continue...
```

```
                        --- USER LOGIN ---
Enter Account Number: AC1001
Enter PIN: 1234
✅ Welcome, Priyanshu Supyal!
Press Enter to continue...
```

```
======================================================================
                        --- ATM MENU ---
======================================================================
                      1. Check Balance 💰
                         2. Deposit 💵
                        3. Withdraw 💸
                       4. Change PIN 🔑
                    5. View Transactions 📃
                   6. Fixed Deposit (FD) 🏦
                         7. Logout 🔙
======================================================================
                      Enter your choice: 1
                Your current balance is: Rs.5000.00
Press Enter to continue...
```

```
                      Enter your choice: 2
Enter amount to deposit: 9999999999
❌ Transaction failed: 1264 (22003): Out of range value for column 'balance' at row 1
Press Enter to continue...
```

```
                      Enter your choice: 3
Enter amount to withdraw: 99999
✅ Rs.99999.00 withdrawn successfully.
Press Enter to continue...
```

```
                      Enter your choice: 4
Enter new 4-digit PIN: 9999
Confirm new PIN: 9999
✅ PIN changed successfully.
Press Enter to continue...
```

```
--- FIXED DEPOSIT (FD) ---
1. Create FD
2. View My FDs
3. Back
Enter your choice:
```

```
--- Recent Transactions ---
Txn_ID   Date                 Type         Amount (Rs)
------------------------------------------------------------
4        2025-12-29 18:05:22  Withdraw     99999.00
3        2025-12-29 18:04:57  Deposit      100000.00
2        2025-12-29 18:04:51  Deposit      1.00
------------------------------------------------------------
Press Enter to continue...
```

```
--- CREATE FIXED DEPOSIT ---
Enter FD amount (minimum Rs.1000): 1000
Choose FD tenure (predefined):
1) 6 months
2) 12 months
3) 24 months
Enter option (1/2/3): 2
✅ FD created successfully.
Principal: Rs.1000.00
Tenure : 12 months
Rate   : 6.50% p.a. (simple interest)
Maturity Amount (approx): Rs.1065.00
Matures on: 2026-12-24 18:08:08
Press Enter to continue...
```

```
--- MY FIXED DEPOSITS ---
FD_ID  Amount (Rs)    Rate%   Start Date          Mature Date         Status
------------------------------------------------------------------------------
1      1000.00        6.50    2025-12-29          2026-12-24          OPEN
------------------------------------------------------------------------------
Press Enter to continue...
```

```
================================================================
                🏦 WELCOME TO ATM SYSTEM 🏦
================================================================
                    1. Admin Login 👨‍💼
                     2. User Login 👤
                        3. Exit 🚪
================================================================
                    Enter your choice: 3
                👋 Thank you for using our ATM system!
```