COREBANK MANAGEMENT SYSTEM

AISSCE 2025-26

PROJECT REPORT

INFORMATION PRACTICES (065)

A PROJECT REPORT
ON
COREBANK
MANAGEMENT SYSTEM

SUBMITTED BY:
NAME: RAGHAV AGARWAL
CLASS: XII A
ROLL NO:

UNDER THE GUIDANCE OF
MS. KIRTI AGARWAL
PGT (INFORMATION PRACTICES)
MAHESHWARI PUBLIC SCHOOL, AJMER

CERTIFICATE

This is to certify that **RAGHAV AGARWAL** of class **XII A**, has prepared the report on the project entitled **COREBANK Management System**.

The report is the result of his effort and endeavors. The report is found worthy of acceptance as final project report for the subject of **Informatics Practices(065)** of class XII under the CBSE curriculum 2025-26. He has prepared the report under my guidance.

KIRTI AGARWAL
PGT (INFORMATION PRACTICES)
MAHESHWARI PUBLIC SCHOOL, AJMER

DECLERATION

I hereby declare that the project work entitled COREBANK Management System submitted to Department of Computer Science, Maheshwari Public School, Ajmer is prepared by me. All the coding and Project report is result of my personal effort.

Name: RAGHAV AGARWAL
CLASS: XII A
Roll no:

ACKNOWLEDGEMENT

I would like to express a deep sense of thanks & gratitude to my project guide **Ms. Kirti Agarwal**, for guiding me immensely through the course of the project. She always evinced keen interest in my work. Her constructive advice & constant motivation have been responsible for the successful completion of this project.

My sincere thanks go to **Mr. R.K.Srivastava**, Our principal for his co-ordination in extending every possible support and IT infrastructure for the completion of this project.

I also thanks to my parents for their motivation & support. I must thanks to my classmates for their timely help & support for compilation of this project.

Last but not the least, I would like to thank all those who had helped directly or indirectly towards the completion of this project.

Name: RAGHAV AGARWAL

CLASS: XII A

A BRIEF INTRODUCTION OF PYTHON

Python is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI in Netherland.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
  - ABC Language
  - Module-3

CONTENT

GENERAL OVERVIEW OF PROBLEM

Automate manual tasks :
A bank management system can automate manual tasks like account opening, interest calculations, and transaction processing.

Maintain accurate records :
A bank management system can maintain accurate records of customer KYC details, account balances, and transaction histories without human error.

Improve efficiency :
A bank management system can increase the accuracy and efficiency of daily operations such as fund transfers, cheque clearing, and loan processing.

Enhance user experience :
A bank management system can enhance the overall user experience by providing customers with instant access to services like balance checks, statements, and loan comparisons.

Ensure data security :
A bank management system can ensure data security and integrity through encrypted password hashing, automated backups, and detailed audit logs.

Provide financial insights :
A bank management system can provide valuable financial insights through visual analytics, credit score tracking, and performance dashboards for better decision-making.

OBJECTIVE OF PROJECT

The primary objectives of the CoreBank Management System project are:

• To develop a comprehensive banking management solution that automates routine banking operations including customer account management, transaction processing, and loan management.

• To implement advanced financial features such as EMI calculation with proper amortization schedules, interest computation (simple and compound), and dynamic credit scoring.

• To design and build a secure banking system with password hashing, input validation, data encryption, and complete audit trail for all transactions.

• To provide professional financial analytics and reporting capabilities including visual charts, financial dashboards, and loan comparison tools.

• To create a user-friendly interface that serves multiple user roles (customers, managers, admins) with appropriate access controls and permissions.

REQUIREMENT & ANALYSIS

FUNCTIONAL REQUIREMENTS:

- **Customer Management –** Add, view, search customers with KYC details
- **Account Management –** Open savings, current, FD accounts with interest
- **Transaction Processing –** Deposit, withdraw, transfer with validation
- **Loan Management –** 5 loan types with EMI, amortization schedules
- **Card Management –** Issue debit/credit/premium cards with CVV
- **Cheque Processing –** Issue, clear, and track cheques
- **Financial Reports –** Charts, dashboards, loan comparison tools

NON-FUNCTIONAL REQUIREMENTS:

- **Security –** SHA-256 password hashing, input validation, audit trails
- **Performance –** Process 1000+ transactions efficiently
- **Reliability –** Data backup, error handling, transaction atomicity
- **Usability –** User-friendly interface, role-based access control
- **Scalability –** Support multi-user concurrent operations
- **Data Integrity –** CSV persistence, timestamp tracking, validation rules

REQUIREMENT & ANALYSIS

HARDWARE REQUIREMENT:

[] Macbook Pro
[] AMD Radeon R9 M370X
[] 1TB Storage
[] Intel I7 Processor

SOFTWARE REQUIREMENT:

[] MacOS 14+
[] Python 3.X IDLE
[] Visual Studio Code

SOURCE CODE

PROJECT STRUCTURE:

SECTION BREAKDOWN:

- **Section 1:** Configuration Constants (Interest rates, limits, colors)
- **Section 2:** Utility Functions (Validators, calculators, formatters)
- **Section 3:** Data Persistence Layer (CSV load/save/backup operations)
- **Section 4:** Core Banking (Customers, accounts, transactions)
- **Section 5:** Fund Transfer System (Inter-account transfers, limits)
- **Section 6:** Card Management (Issue, activate, block debit/credit cards)
- **Section 7:** Cheque Processing (Issue, clear, track cheques)
- **Section 8:** Reports & Analytics (6 report types, 5 chart types)
- **Section 9:** Advanced Features (Statements, Dashboard, Loan Comparison)

KEY TECHNOLOGIES:

- Python 3.x with OOP principles
- Pandas for data manipulation
- Matplotlib for charts & visualization
- Hashlib for SHA-256 security
- JSON-in-CSV format for data persistence

SOURCE CODE

```
IMPORT OS
IMPORT SYS
IMPORT JSON
IMPORT HASHLIB
IMPORT RE
IMPORT RANDOM
IMPORT STRING
IMPORT PANDAS AS PD
IMPORT NUMPY AS NP
IMPORT MATPLOTLIB.PYPLOT AS PLT
FROM DATETIME IMPORT DATETIME, TIMEDELTA

# ========================================
# SECTION 1: CONFIGURATION & CONSTANTS
# ========================================

# DATABASE FILE
DB_FILE = 'BANK_DATABASE.CSV'
BACKUP_DIR = 'BACKUPS/'

# SYSTEM CONFIGURATION
FINE_PER_DAY = 2.0 # RUPEES PER DAY FOR OVERDUE

# INTEREST RATES (ANNUAL PERCENTAGE)
SAVINGS_INTEREST = 4.0
CURRENT_INTEREST = 0.0
FD_INTEREST = 7.5
RD_INTEREST = 5.5

# LOAN INTEREST RATES
LOAN_RATES = {
 'HOME LOAN': 8.5,
 'PERSONAL LOAN': 12.0,
 'CAR LOAN': 9.0,
 'EDUCATION LOAN': 7.0,
 'BUSINESS LOAN': 10.0
}

# MINIMUM BALANCES
MIN_SAVINGS = 1000
MIN_CURRENT = 5000
```

```
# TRANSACTION LIMITS
DAILY_WITHDRAWAL_LIMIT = 50000
DAILY_TRANSFER_LIMIT = 200000
LARGE_TRANSACTION_THRESHOLD = 100000

# CARD CONFIGURATION
CARD_VALIDITY_YEARS = 5
CREDIT_CARD_LIMIT_DEFAULT = 50000

# CHEQUE CONFIGURATION
CHEQUE_CLEARANCE_DAYS = 3

# EMI CALCULATION
EMI_PRECISION = 2

# TABLE SCHEMAS (FOR INITIALIZATION)
SCHEMAS = {
 'CUSTOMERS': ['CUSTOMERID', 'NAME', 'DOB', 'GENDER', 'PAN', 'AADHAR', 'ADDRESS', 'CITY',
'STATE', 'PIN', 'PHONE', 'EMAIL', 'REGISTRATIONDATE', 'STATUS', 'KYC_STATUS'],
 'ACCOUNTS': ['ACCOUNTNUMBER', 'CUSTOMERID', 'ACCOUNTTYPE', 'BALANCE',
'MINBALANCE', 'INTERESTRATE', 'OPENINGDATE', 'MATURITYDATE', 'STATUS',
'LASTINTERESTCREDITED'],
 'TRANSACTIONS': ['TRANSACTIONID', 'ACCOUNTNUMBER', 'TRANSACTIONTYPE',
'AMOUNT', 'DEBITCREDIT', 'BALANCE_AFTER', 'DATE', 'TIME', 'REMARKS', 'STATUS'],
 'TRANSFERS': ['TRANSFERID', 'FROMACCOUNT', 'TOACCOUNT', 'AMOUNT', 'TRANSFERTYPE',
'CHARGES', 'DATE', 'STATUS', 'REFERENCE'],
 'LOANS': ['LOANID', 'CUSTOMERID', 'LINKEDACCOUNT', 'LOANTYPE', 'PRINCIPALAMOUNT',
'INTERESTRATE', 'TENURE_MONTHS', 'EMI', 'STARTDATE', 'MATURITYDATE',
'OUTSTANDINGAMOUNT', 'STATUS', 'APPROVALDATE'],
 'LOAN_PAYMENTS': ['PAYMENTID', 'LOANID', 'PAYMENTDATE', 'AMOUNTPAID',
'PRINCIPALPART', 'INTERESTPART', 'OUTSTANDINGAFTER', 'PAYMENTMETHOD', 'STATUS'],
 'CARDS': ['CARDNUMBER', 'CUSTOMERID', 'LINKEDACCOUNT', 'CARDTYPE', 'CREDITLIMIT',
'ISSUEDATE', 'EXPIRYDATE', 'CVV', 'PIN_HASH', 'STATUS'],
 'CHEQUES': ['CHEQUENUMBER', 'ACCOUNTNUMBER', 'ISSUEDTO', 'AMOUNT', 'ISSUEDATE',
'CLEARANCEDATE', 'STATUS', 'REMARKS'],
 'USERS': ['USERID', 'USERNAME', 'PASSWORD_HASH', 'ROLE', 'EMPLOYEEID', 'EMAIL',
'STATUS', 'LASTLOGIN'],
 'AUDIT': ['LOGID', 'USERID', 'ACTION', 'DETAILS', 'TIMESTAMP', 'IPADDRESS', 'STATUS']
}
# MESSAGE COLORS FOR BETTER UX
CLASS COLORS:
 GREEN = '\033[92M'
 RED = '\033[91M'
 YELLOW = '\033[93M'
 BLUE = '\033[94M'
 CYAN = '\033[96M'
 END = '\033[0M'
 BOLD = '\033[1M'
```

```
# ========================================
# SECTION 2: UTILITY FUNCTIONS (VALIDATORS, SECURITY, CALCULATORS)
# ========================================

# --- VALIDATORS ---

DEF VALIDATE_PAN(PAN):
 """VALIDATE PAN FORMAT (10 ALPHANUMERIC: ABCDE1234F)"""
 PATTERN = R'^[A-Z]{5}[0-9]{4}[A-Z]{1}$'
 RETURN BOOL(RE.MATCH(PATTERN, PAN))

DEF VALIDATE_AADHAR(AADHAR):
 """VALIDATE AADHAR FORMAT (12 DIGITS)"""
 RETURN AADHAR.ISDIGIT() AND LEN(AADHAR) == 12

DEF VALIDATE_EMAIL(EMAIL):
 """VALIDATE EMAIL FORMAT"""
 PATTERN = R'^[A-ZA-Z0-9._%+-]+@[A-ZA-Z0-9.-]+\.[A-ZA-Z]{2,}$'
 RETURN BOOL(RE.MATCH(PATTERN, EMAIL))

DEF VALIDATE_PHONE(PHONE):
 """VALIDATE PHONE FORMAT (10 DIGITS)"""
 RETURN PHONE.ISDIGIT() AND LEN(PHONE) == 10

DEF VALIDATE_PIN_CODE(PIN):
 """VALIDATE PIN CODE (6 DIGITS)"""
 RETURN PIN.ISDIGIT() AND LEN(PIN) == 6

DEF VALIDATE_AMOUNT(AMOUNT):
 """VALIDATE AMOUNT (POSITIVE NUMBER)"""
 TRY:
 AMT = FLOAT(AMOUNT)
 RETURN AMT > 0
 EXCEPT VALUEERROR:
 RETURN FALSE

DEF VALIDATE_DATE_FORMAT(DATE_STR):
 """VALIDATE DATE FORMAT (YYYY-MM-DD)"""
 TRY:
 DATETIME.STRPTIME(DATE_STR, '%Y-%M-%D')
 RETURN TRUE
 EXCEPT VALUEERROR:
 RETURN FALSE

# --- SECURITY ---
DEF HASH_PASSWORD(PASSWORD):
 """HASH PASSWORD USING SHA256"""
 RETURN HASHLIB.SHA256(PASSWORD.ENCODE()).HEXDIGEST()
```

```
DEF VERIFY_PASSWORD(PASSWORD, HASH_VALUE):
 """VERIFY PASSWORD AGAINST HASH"""
 RETURN HASH_PASSWORD(PASSWORD) == HASH_VALUE

DEF MASK_PAN(PAN):
 """MASK PAN - SHOW ONLY LAST 4 CHARACTERS"""
 IF LEN(PAN) >= 4:
 RETURN "XXXX" + PAN[-4:]
 RETURN PAN

DEF MASK_AADHAR(AADHAR):
 """MASK AADHAR - SHOW ONLY LAST 4 DIGITS"""
 IF LEN(AADHAR) >= 4:
 RETURN "XXXXXXXX" + AADHAR[-4:]
 RETURN AADHAR

DEF MASK_ACCOUNT_NUMBER(ACCOUNT_NUM):
 """MASK ACCOUNT NUMBER - SHOW ONLY LAST 4 DIGITS"""
 IF LEN(ACCOUNT_NUM) >= 4:
 RETURN "XXX" + ACCOUNT_NUM[-4:]
 RETURN ACCOUNT_NUM

DEF GET_TIMESTAMP():
 """GET CURRENT TIMESTAMP"""
 RETURN DATETIME.NOW().STRFTIME("%Y-%M-%D %H:%M:%S")

DEF GET_DATE():
 """GET CURRENT DATE IN YYYY-MM-DD FORMAT"""
 RETURN DATETIME.NOW().STRFTIME("%Y-%M-%D")

# --- CALCULATORS ---

DEF CALCULATE_EMI(PRINCIPAL, ANNUAL_INTEREST_RATE, TENURE_MONTHS):
 """CALCULATE EMI: [P X R X (1+R)^N] / [(1+R)^N - 1]"""
 IF PRINCIPAL <= 0 OR ANNUAL_INTEREST_RATE < 0 OR TENURE_MONTHS <= 0:
 RETURN 0

 MONTHLY_RATE = ANNUAL_INTEREST_RATE / (12 * 100)
 IF MONTHLY_RATE == 0:
 RETURN ROUND(PRINCIPAL / TENURE_MONTHS, 2)

 NUMERATOR = PRINCIPAL * MONTHLY_RATE * ((1 + MONTHLY_RATE) **
TENURE_MONTHS)
 DENOMINATOR = ((1 + MONTHLY_RATE) ** TENURE_MONTHS) - 1

 EMI = NUMERATOR / DENOMINATOR
 RETURN ROUND(EMI, 2)
```

```
DEF CALCULATE_SIMPLE_INTEREST(PRINCIPAL, ANNUAL_RATE, DAYS):
 """CALCULATE SIMPLE INTEREST: (P X R X T) / 100"""
 TIME_IN_YEARS = DAYS / 365
 INTEREST = (PRINCIPAL * ANNUAL_RATE * TIME_IN_YEARS) / 100
 RETURN ROUND(INTEREST, 2)

DEF CALCULATE_AMORTIZATION_SCHEDULE(PRINCIPAL, ANNUAL_RATE,
TENURE_MONTHS):
 """GENERATE COMPLETE AMORTIZATION SCHEDULE"""
 SCHEDULE = []
 MONTHLY_RATE = ANNUAL_RATE / (12 * 100)
 EMI = CALCULATE_EMI(PRINCIPAL, ANNUAL_RATE, TENURE_MONTHS)
 OUTSTANDING = PRINCIPAL

 FOR MONTH IN RANGE(1, TENURE_MONTHS + 1):
 INTEREST_PART = OUTSTANDING * MONTHLY_RATE
 PRINCIPAL_PART = EMI - INTEREST_PART
 OUTSTANDING -= PRINCIPAL_PART

 SCHEDULE.APPEND({
 'MONTH': MONTH,
 'EMI': ROUND(EMI, 2),
 'PRINCIPAL': ROUND(PRINCIPAL_PART, 2),
 'INTEREST': ROUND(INTEREST_PART, 2),
 'OUTSTANDING': ROUND(MAX(0, OUTSTANDING), 2)
 })

 RETURN SCHEDULE

DEF GENERATE_CARD_NUMBER():
 """GENERATE A 16-DIGIT CARD NUMBER"""
 RETURN ''.JOIN([STR(RANDOM.RANDINT(0, 9)) FOR _ IN RANGE(16)])

DEF GENERATE_CVV():
 """GENERATE A 3-DIGIT CVV"""
 RETURN ''.JOIN([STR(RANDOM.RANDINT(0, 9)) FOR _ IN RANGE(3)])

DEF GENERATE_CHEQUE_NUMBER():
 """GENERATE A 6-DIGIT CHEQUE NUMBER"""
 RETURN ''.JOIN([STR(RANDOM.RANDINT(0, 9)) FOR _ IN RANGE(6)])

DEF MASK_CARD_NUMBER(CARD_NUM):
 """MASK CARD NUMBER - SHOW ONLY LAST 4 DIGITS"""
 IF LEN(CARD_NUM) >= 4:
 RETURN "XXXX-XXXX-XXXX-" + CARD_NUM[-4:]
 RETURN CARD_NUM
```

```
    # ACCOUNT AGE BONUS (MAX 100 POINTS)
    AGE_SCORE = MIN(100, ACCOUNT_AGE_DAYS // 30 * 5)

    # TRANSACTION FREQUENCY BONUS (MAX 100 POINTS)
    TXN_SCORE = MIN(100, TOTAL_TRANSACTIONS * 2)

    # BALANCE BONUS (MAX 100 POINTS)
    BALANCE_SCORE = MIN(100, INT(TOTAL_BALANCE / 10000) * 10)

    # PENALTY FOR DEFAULTS
    DEFAULT_PENALTY = DEFAULTED_LOANS * 100

    FINAL_SCORE = BASE_SCORE + AGE_SCORE + TXN_SCORE + BALANCE_SCORE -
DEFAULT_PENALTY
    RETURN MAX(300, MIN(900, FINAL_SCORE))

DEF CALCULATE_CREDIT_SCORE(TOTAL_BALANCE, NUM_LOANS, NUM_TRANSACTIONS):
    """SIMPLIFIED CREDIT SCORE CALCULATION FOR DASHBOARD"""
    BASE_SCORE = 600

    # BALANCE BONUS (MAX 150 POINTS)
    BALANCE_SCORE = MIN(150, INT(TOTAL_BALANCE / 10000) * 5)

    # TRANSACTION ACTIVITY BONUS (MAX 100 POINTS)
    TXN_SCORE = MIN(100, NUM_TRANSACTIONS * 3)

    # LOAN FACTOR (HAVING LOANS CAN BE POSITIVE IF MANAGED WELL)
    LOAN_SCORE = MIN(50, NUM_LOANS * 10)

    FINAL_SCORE = BASE_SCORE + BALANCE_SCORE + TXN_SCORE + LOAN_SCORE
    RETURN MAX(300, MIN(900, FINAL_SCORE))

DEF LOG_AUDIT(DATA, ACTION, DETAILS, STATUS='SUCCESS'):
    """LOG AN ACTION TO AUDIT TRAIL"""
    LOG_ENTRY = {
    'LOGID': F"LOG{LEN(DATA['AUDIT'])+1:06D}",
    'USERID': 'SYSTEM',
    'ACTION': ACTION,
    'DETAILS': DETAILS,
    'TIMESTAMP': GET_TIMESTAMP(),
    'IPADDRESS': '127.0.0.1',
    'STATUS': STATUS
    }
    DATA['AUDIT'] = PD.CONCAT([DATA['AUDIT'], PD.DATAFRAME([LOG_ENTRY])],
IGNORE_INDEX=TRUE)
    RETURN DATA
```

```
# ========================================
# SECTION 3: DATA MANAGEMENT
# ========================================

DEF INITIALIZE_DATA():
 """INITIALIZE DATABASE FILE IF IT DOESN'T EXIST"""
 IF NOT OS.PATH.EXISTS(DB_FILE):
 PRINT(F"{COLORS.YELLOW}INITIALIZING NEW DATABASE...{COLORS.END}")
 # CREATE EMPTY FILE WITH HEADERS
 PD.DATAFRAME(COLUMNS=['TABLE', 'DATA']).TO_CSV(DB_FILE, INDEX=FALSE)

 # CREATE DEFAULT ADMIN USER
 ADMIN_USER = {
 'USERID': 'USR001',
 'USERNAME': 'ADMIN',
 'PASSWORD_HASH': HASH_PASSWORD('ADMIN@123'),
 'ROLE': 'ADMIN',
 'EMPLOYEEID': 'EMP001',
 'EMAIL': 'ADMIN@COREBANK.COM',
 'STATUS': 'ACTIVE',
 'LASTLOGIN': GET_TIMESTAMP()
 }

 # SAVE INITIAL DATA
 DATA = {TABLE: PD.DATAFRAME(COLUMNS=COLS) FOR TABLE, COLS IN SCHEMAS.ITEMS()}
 DATA['USERS'] = PD.DATAFRAME([ADMIN_USER])
 SAVE_DATA(DATA)

DEF LOAD_DATA():
 """LOAD DATA FROM SINGLE CSV INTO DICTIONARY OF DATAFRAMES"""
 TRY:
 IF NOT OS.PATH.EXISTS(DB_FILE):
 INITIALIZE_DATA()

 DF = PD.READ_CSV(DB_FILE)
 DATA = {}

 FOR TABLE, COLUMNS IN SCHEMAS.ITEMS():
 # FILTER ROWS FOR THIS TABLE
 TABLE_ROWS = DF[DF['TABLE'] == TABLE]['DATA']

 IF NOT TABLE_ROWS.EMPTY:
 # PARSE JSON RECORDS (INDEXED DICTIONARY FORMAT)
 JSON_STR = TABLE_ROWS.ILOC[0]
 INDEXED_DICT = JSON.LOADS(JSON_STR)

 # EXTRACT RECORDS - SKIP 'TABLE' AND 'DATA' KEYS AND NULL VALUES
 RECORDS = []
 FOR KEY, VALUE IN INDEXED_DICT.ITEMS():
```

```
 # SKIP NON-DICT VALUES AND SKIP IF VALUE IS NONE OR EMPTY
 IF ISINSTANCE(VALUE, DICT) AND VALUE AND KEY NOT IN ['TABLE', 'DATA']:
 RECORDS.APPEND(VALUE)

 IF RECORDS:
 DATA[TABLE] = PD.DATAFRAME(RECORDS)
 # ENSURE ALL COLUMNS EXIST (HANDLE SCHEMA EVOLUTION)
 FOR COL IN COLUMNS:
 IF COL NOT IN DATA[TABLE].COLUMNS:
 DATA[TABLE][COL] = NONE
 ELSE:
 # CREATE EMPTY DATAFRAME WITH CORRECT SCHEMA
 DATA[TABLE] = PD.DATAFRAME(COLUMNS=COLUMNS)
 ELSE:
 # CREATE EMPTY DATAFRAME WITH CORRECT SCHEMA
 DATA[TABLE] = PD.DATAFRAME(COLUMNS=COLUMNS)

 # FIX DATA INTEGRITY ISSUES
 DATA = FIX_DATA_INTEGRITY(DATA)
 RETURN DATA
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR LOADING DATA: {STR(E)}{COLORS.END}")
 # RETURN EMPTY STRUCTURE ON ERROR TO PREVENT CRASH
 RETURN {TABLE: PD.DATAFRAME(COLUMNS=COLS) FOR TABLE, COLS IN
SCHEMAS.ITEMS()}

DEF FIX_DATA_INTEGRITY(DATA):
 """FIX COMMON DATA INTEGRITY ISSUES IN LOADED DATA"""
 # FIX TRANSACTIONS - ENSURE DEBITCREDIT AND BALANCE COLUMNS ARE POPULATED
 IF NOT DATA['TRANSACTIONS'].EMPTY:
 # FIX MISSING DEBITCREDIT BASED ON TRANSACTIONTYPE
 DEBIT_TYPES = ['WITHDRAWAL', 'TRANSFER DEBIT', 'CHEQUE DEBIT', 'FUND TRANSFER',
'EMI PAYMENT']
 CREDIT_TYPES = ['DEPOSIT', 'TRANSFER CREDIT', 'CHEQUE CREDIT', 'ACCOUNT OPENING',
'INTEREST CREDIT']

 FOR IDX, ROW IN DATA['TRANSACTIONS'].ITERROWS():
 TXN_TYPE = STR(ROW.GET('TRANSACTIONTYPE', ''))

 # FIX DEBITCREDIT IF MISSING
 IF PD.ISNA(ROW.GET('DEBITCREDIT')) OR ROW.GET('DEBITCREDIT') IS NONE:
 IF ANY(DT IN TXN_TYPE FOR DT IN DEBIT_TYPES):
 DATA['TRANSACTIONS'].AT[IDX, 'DEBITCREDIT'] = 'DEBIT'
 ELIF ANY(CT IN TXN_TYPE FOR CT IN CREDIT_TYPES):
 DATA['TRANSACTIONS'].AT[IDX, 'DEBITCREDIT'] = 'CREDIT'
 ELSE:
 DATA['TRANSACTIONS'].AT[IDX, 'DEBITCREDIT'] = 'N/A'
```

```
# FIX BALANCE_AFTER IF MISSING (USE BALANCEAFTER IF AVAILABLE)
IF PD.ISNA(ROW.GET('BALANCE_AFTER')) OR ROW.GET('BALANCE_AFTER') IS NONE:
BALANCE_AFTER = ROW.GET('BALANCEAFTER')
IF PD.NOTNA(BALANCE_AFTER):
DATA['TRANSACTIONS'].AT[IDX, 'BALANCE_AFTER'] = BALANCE_AFTER

# FIX CARDS - ENSURE LINKEDACCOUNT IS POPULATED
IF NOT DATA['CARDS'].EMPTY AND 'LINKEDACCOUNT' IN DATA['CARDS'].COLUMNS:
FOR IDX, ROW IN DATA['CARDS'].ITERROWS():
IF PD.ISNA(ROW.GET('LINKEDACCOUNT')) OR ROW.GET('LINKEDACCOUNT') IS NONE:
# TRY TO GET FROM ACCOUNTNUMBER COLUMN
ACC_NUM = ROW.GET('ACCOUNTNUMBER')
IF PD.NOTNA(ACC_NUM):
DATA['CARDS'].AT[IDX, 'LINKEDACCOUNT'] = ACC_NUM

RETURN DATA

DEF SAVE_DATA(DATA_DICT):
"""SAVE DICTIONARY OF DATAFRAMES TO SINGLE CSV (JSON-IN-CSV FORMAT)"""
TRY:
ALL_ROWS = []
FOR TABLE, DF IN DATA_DICT.ITEMS():
IF NOT DF.EMPTY:
# CONVERT DATAFRAME TO LIST OF RECORDS
RECORDS = DF.TO_DICT(ORIENT='RECORDS')
# CREATE INDEXED DICTIONARY FORMAT
INDEXED_DICT = {STR(I): RECORD FOR I, RECORD IN ENUMERATE(RECORDS)}
# ADD NULL COLUMNS FOR SCHEMA EVOLUTION
INDEXED_DICT['TABLE'] = NONE
INDEXED_DICT['DATA'] = NONE
# CONVERT TO JSON AND SAVE AS ONE ROW PER TABLE
JSON_DATA = JSON.DUMPS(INDEXED_DICT)
ALL_ROWS.APPEND({
'TABLE': TABLE,
'DATA': JSON_DATA
})
ELSE:
# SAVE EMPTY TABLE
ALL_ROWS.APPEND({
'TABLE': TABLE,
'DATA': JSON.DUMPS({})
})

PD.DATAFRAME(ALL_ROWS).TO_CSV(DB_FILE, INDEX=FALSE)
RETURN TRUE
EXCEPT EXCEPTION AS E:
PRINT(F"{COLORS.RED}ERROR SAVING DATA: {STR(E)}{COLORS.END}")
RETURN FALSE
```

```
DEF BACKUP_DATA():
 """CREATE BACKUP OF THE DATABASE FILE"""
 PRINT(F"\N{COLORS.CYAN}--- CREATING DATA BACKUP ---{COLORS.END}")
 TIMESTAMP = DATETIME.NOW().STRFTIME("%Y%M%D_%H%M%S")
 OS.MAKEDIRS(BACKUP_DIR, EXIST_OK=TRUE)
 BACKUP_PATH = OS.PATH.JOIN(BACKUP_DIR, F"BANK_DATABASE_{TIMESTAMP}.CSV")

 TRY:
 IF OS.PATH.EXISTS(DB_FILE):
 WITH OPEN(DB_FILE, 'R') AS SRC:
 WITH OPEN(BACKUP_PATH, 'W') AS DST:
 DST.WRITE(SRC.READ())
 PRINT(F"{COLORS.GREEN}✓ BACKUP CREATED: {BACKUP_PATH}{COLORS.END}")
 RETURN TRUE
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}✗ BACKUP FAILED: {STR(E)}{COLORS.END}")
 RETURN FALSE

# ========================================
# SECTION 4: CORE FEATURES
# ========================================

# --- CUSTOMER MANAGEMENT ---

DEF ADD_CUSTOMER(DATA):
 PRINT(F"\N{COLORS.CYAN}--- ADD NEW CUSTOMER ---{COLORS.END}")
 TRY:
 # GENERATE ID
 EXISTING_IDS = DATA['CUSTOMERS']['CUSTOMERID'].TOLIST() IF NOT
DATA['CUSTOMERS'].EMPTY ELSE []
 NEW_ID_NUM = MAX([INT(CID.REPLACE('CUST', '')) FOR CID IN EXISTING_IDS IF 'CUST' IN
CID], DEFAULT=0) + 1
 CUSTOMER_ID = F"CUST{NEW_ID_NUM:03D}"

 NAME = INPUT("ENTER CUSTOMER NAME: ").STRIP()
 IF LEN(NAME) < 2: RETURN DATA

 DOB = INPUT("ENTER DOB (YYYY-MM-DD): ").STRIP()
 IF NOT VALIDATE_DATE_FORMAT(DOB):
 PRINT(F"{COLORS.RED}INVALID DATE{COLORS.END}")
 RETURN DATA

 GENDER = INPUT("ENTER GENDER: ").STRIP()
 PAN = INPUT("ENTER PAN: ").STRIP().UPPER()
 IF NOT VALIDATE_PAN(PAN):
 PRINT(F"{COLORS.RED}INVALID PAN{COLORS.END}")
 RETURN DATA
```

```
 AADHAR = INPUT("ENTER AADHAR: ").STRIP()
 IF NOT VALIDATE_AADHAR(AADHAR):
 PRINT(F"{COLORS.RED}INVALID AADHAR{COLORS.END}")
 RETURN DATA

 ADDRESS = INPUT("ENTER ADDRESS: ").STRIP()
 CITY = INPUT("ENTER CITY: ").STRIP()
 STATE = INPUT("ENTER STATE: ").STRIP()
 PIN = INPUT("ENTER PIN: ").STRIP()
 PHONE = INPUT("ENTER PHONE: ").STRIP()
 EMAIL = INPUT("ENTER EMAIL: ").STRIP()

 NEW_CUSTOMER = {
 'CUSTOMERID': CUSTOMER_ID, 'NAME': NAME, 'DOB': DOB, 'GENDER': GENDER,
 'PAN': PAN, 'AADHAR': AADHAR, 'ADDRESS': ADDRESS, 'CITY': CITY,
 'STATE': STATE, 'PIN': PIN, 'PHONE': PHONE, 'EMAIL': EMAIL,
 'REGISTRATIONDATE': GET_DATE(), 'STATUS': 'ACTIVE', 'KYC_STATUS': 'PENDING'
 }

 DATA['CUSTOMERS'] = PD.CONCAT([DATA['CUSTOMERS'],
PD.DATAFRAME([NEW_CUSTOMER])], IGNORE_INDEX=TRUE)
 PRINT(F"{COLORS.GREEN}✓ CUSTOMER {CUSTOMER_ID} ADDED!{COLORS.END}")

 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR: {E}{COLORS.END}")
 RETURN DATA

DEF VIEW_CUSTOMERS(DATA):
 PRINT(F"\N{COLORS.CYAN}--- CUSTOMER LIST ---{COLORS.END}")
 IF DATA['CUSTOMERS'].EMPTY:
 PRINT("NO CUSTOMERS FOUND.")
 RETURN
 PRINT(DATA['CUSTOMERS'][['CUSTOMERID', 'NAME', 'PHONE',
'STATUS']].TO_STRING(INDEX=FALSE))

DEF SEARCH_CUSTOMER(DATA):
 PRINT(F"\N{COLORS.CYAN}--- SEARCH CUSTOMER ---{COLORS.END}")
 QUERY = INPUT("ENTER ID OR NAME: ").STRIP()
 RESULTS = DATA['CUSTOMERS'][
 (DATA['CUSTOMERS']['CUSTOMERID'].STR.CONTAINS(QUERY, CASE=FALSE, NA=FALSE)) |
 (DATA['CUSTOMERS']['NAME'].STR.CONTAINS(QUERY, CASE=FALSE, NA=FALSE))
 ]
 IF RESULTS.EMPTY:
 PRINT("NO RESULTS.")
 ELSE:
 PRINT(RESULTS[['CUSTOMERID', 'NAME', 'PHONE', 'EMAIL']].TO_STRING(INDEX=FALSE))
```

```
# --- ACCOUNT MANAGEMENT ---

DEF OPEN_ACCOUNT(DATA):
 PRINT(F"\N{COLORS.CYAN}--- OPEN NEW ACCOUNT ---{COLORS.END}")
 CUSTOMER_ID = INPUT("ENTER CUSTOMER ID: ").STRIP()

 IF DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUSTOMER_ID].EMPTY:
 PRINT(F"{COLORS.RED}CUSTOMER NOT FOUND{COLORS.END}")
 RETURN DATA

 PRINT("1. SAVINGS (4%) 2. CURRENT (0%) 3. FIXED DEPOSIT (7.5%)")
 CHOICE = INPUT("SELECT TYPE: ").STRIP()

 TYPES = {'1': ('SAVINGS', SAVINGS_INTEREST, MIN_SAVINGS),
 '2': ('CURRENT', CURRENT_INTEREST, MIN_CURRENT),
 '3': ('FIXED DEPOSIT', FD_INTEREST, 0)}

 IF CHOICE NOT IN TYPES: RETURN DATA
 ACC_TYPE, RATE, MIN_BAL = TYPES[CHOICE]

 AMOUNT = FLOAT(INPUT("INITIAL DEPOSIT: ").STRIP())
 IF AMOUNT < MIN_BAL:
 PRINT(F"{COLORS.RED}MINIMUM BALANCE IS ₹{MIN_BAL}{COLORS.END}")
 RETURN DATA

 # GENERATE ACCOUNT NUMBER
 EXISTING = DATA['ACCOUNTS']['ACCOUNTNUMBER'].TOLIST() IF NOT
DATA['ACCOUNTS'].EMPTY ELSE []
 NEW_NUM = MAX([INT(A.REPLACE('ACC', '')) FOR A IN EXISTING IF 'ACC' IN A],
DEFAULT=1000) + 1
 ACC_NUM = F"ACC{NEW_NUM}"

 NEW_ACC = {
 'ACCOUNTNUMBER': ACC_NUM, 'CUSTOMERID': CUSTOMER_ID, 'ACCOUNTTYPE':
ACC_TYPE,
 'BALANCE': AMOUNT, 'MINBALANCE': MIN_BAL, 'INTERESTRATE': RATE,
 'OPENINGDATE': GET_DATE(), 'MATURITYDATE': '', 'STATUS': 'ACTIVE',
 'LASTINTERESTCREDITED': GET_DATE()
 }
 DATA['ACCOUNTS'] = PD.CONCAT([DATA['ACCOUNTS'], PD.DATAFRAME([NEW_ACC])],
IGNORE_INDEX=TRUE)
  # LOG TRANSACTION
 TXN = {
 'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
 'ACCOUNTNUMBER': ACC_NUM, 'TRANSACTIONTYPE': 'ACCOUNT OPENING',
 'AMOUNT': AMOUNT, 'DEBITCREDIT': 'CREDIT', 'BALANCE_AFTER': AMOUNT,
 'DATE': GET_DATE(), 'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
 'REMARKS': 'INITIAL DEPOSIT', 'STATUS': 'SUCCESS' }
```

```
 DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'], PD.DATAFRAME([TXN])],
IGNORE_INDEX=TRUE)

 PRINT(F"{COLORS.GREEN}✓ ACCOUNT {ACC_NUM} OPENED!{COLORS.END}")
 RETURN DATA

DEF CHECK_BALANCE(DATA):
 ACC_NUM = INPUT("ENTER ACCOUNT NUMBER: ").STRIP()
 ACC = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]
 IF ACC.EMPTY:
 PRINT("ACCOUNT NOT FOUND.")
 ELSE:
 ROW = ACC.ILOC[0]
 PRINT(F"\NACCOUNT: {ROW['ACCOUNTNUMBER']} ({ROW['ACCOUNTTYPE']})")
 PRINT(F"BALANCE: ₹{ROW['BALANCE']:.2F}")

# --- TRANSACTIONS ---

DEF DEPOSIT_MONEY(DATA):
 PRINT(F"\N{COLORS.CYAN}--- DEPOSIT ---{COLORS.END}")
 ACC_NUM = INPUT("ACCOUNT NUMBER: ").STRIP()
 ACC = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]

 IF ACC.EMPTY:
 PRINT("ACCOUNT NOT FOUND.")
 RETURN DATA

 AMOUNT = FLOAT(INPUT("AMOUNT: ").STRIP())
 IF AMOUNT <= 0: RETURN DATA

 IDX = ACC.INDEX[0]
 NEW_BAL = DATA['ACCOUNTS'].AT[IDX, 'BALANCE'] + AMOUNT
 DATA['ACCOUNTS'].AT[IDX, 'BALANCE'] = NEW_BAL

 TXN = {
 'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
 'ACCOUNTNUMBER': ACC_NUM, 'TRANSACTIONTYPE': 'DEPOSIT',
 'AMOUNT': AMOUNT, 'DEBITCREDIT': 'CREDIT', 'BALANCE_AFTER': NEW_BAL,
 'DATE': GET_DATE(), 'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
 'REMARKS': 'CASH DEPOSIT', 'STATUS': 'SUCCESS'
 }
 DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'], PD.DATAFRAME([TXN])],
IGNORE_INDEX=TRUE)
 PRINT(F"{COLORS.GREEN}✓ DEPOSITED ₹{AMOUNT}. NEW BALANCE: ₹{NEW_BAL:.2F}
{COLORS.END}")
 RETURN DATA
```

```
DEF WITHDRAW_MONEY(DATA):
 PRINT(F"\N{COLORS.CYAN}--- WITHDRAW ---{COLORS.END}")
 ACC_NUM = INPUT("ACCOUNT NUMBER: ").STRIP()
 ACC = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]

 IF ACC.EMPTY:
 PRINT("ACCOUNT NOT FOUND.")
 RETURN DATA

 AMOUNT = FLOAT(INPUT("AMOUNT: ").STRIP())
 IDX = ACC.INDEX[0]
 CURRENT_BAL = DATA['ACCOUNTS'].AT[IDX, 'BALANCE']
 MIN_BAL = DATA['ACCOUNTS'].AT[IDX, 'MINBALANCE']

 IF CURRENT_BAL - AMOUNT < MIN_BAL:
 PRINT(F"{COLORS.RED}INSUFFICIENT BALANCE (MIN: ₹{MIN_BAL}){COLORS.END}")
 RETURN DATA

 NEW_BAL = CURRENT_BAL - AMOUNT
 DATA['ACCOUNTS'].AT[IDX, 'BALANCE'] = NEW_BAL

 TXN = {
 'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
 'ACCOUNTNUMBER': ACC_NUM, 'TRANSACTIONTYPE': 'WITHDRAWAL',
 'AMOUNT': AMOUNT, 'DEBITCREDIT': 'DEBIT', 'BALANCE_AFTER': NEW_BAL,
 'DATE': GET_DATE(), 'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
 'REMARKS': 'CASH WITHDRAWAL', 'STATUS': 'SUCCESS'
 }
 DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'], PD.DATAFRAME([TXN])],
IGNORE_INDEX=TRUE)
 PRINT(F"{COLORS.GREEN}✓ WITHDRAWN ₹{AMOUNT}. NEW BALANCE: ₹{NEW_BAL:.2F}
{COLORS.END}")
 RETURN DATA

# --- LOANS ---

DEF APPLY_LOAN(DATA):
 PRINT(F"\N{COLORS.CYAN}--- APPLY LOAN ---{COLORS.END}")
 CUST_ID = INPUT("CUSTOMER ID: ").STRIP()
 IF DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUST_ID].EMPTY:
 PRINT("CUSTOMER NOT FOUND.")
 RETURN DATA

 PRINT("LOAN TYPES: " + ", ".JOIN(LOAN_RATES.KEYS()))
 L_TYPE = INPUT("TYPE: ").STRIP()
 IF L_TYPE NOT IN LOAN_RATES: RETURN DATA

 AMOUNT = FLOAT(INPUT("AMOUNT: ").STRIP())
 MONTHS = INT(INPUT("TENURE (MONTHS): ").STRIP())
```

```
 RATE = LOAN_RATES[L_TYPE]
  EMI = CALCULATE_EMI(AMOUNT, RATE, MONTHS)

  LOAN_ID = F"LOAN{LEN(DATA['LOANS'])+1:03D}"
  NEW_LOAN = {
  'LOANID': LOAN_ID, 'CUSTOMERID': CUST_ID, 'LINKEDACCOUNT': '',
  'LOANTYPE': L_TYPE, 'PRINCIPALAMOUNT': AMOUNT, 'INTERESTRATE': RATE,
  'TENURE_MONTHS': MONTHS, 'EMI': EMI, 'STARTDATE': GET_DATE(),
  'MATURITYDATE': '', 'OUTSTANDINGAMOUNT': AMOUNT, 'STATUS': 'ACTIVE',
  'APPROVALDATE': GET_DATE()
  }

  DATA['LOANS'] = PD.CONCAT([DATA['LOANS'], PD.DATAFRAME([NEW_LOAN])],
IGNORE_INDEX=TRUE)
  DATA = LOG_AUDIT(DATA, 'LOAN_APPLIED', F'LOAN {LOAN_ID} FOR ₹{AMOUNT}')
  PRINT(F"{COLORS.GREEN}✓ LOAN {LOAN_ID} APPROVED! EMI: ₹{EMI:.2F}{COLORS.END}")
  RETURN DATA

DEF PAY_LOAN_EMI(DATA):
  """PAY EMI FOR AN ACTIVE LOAN"""
  PRINT(F"\N{COLORS.CYAN}--- PAY LOAN EMI ---{COLORS.END}")

  # SHOW ACTIVE LOANS
  ACTIVE_LOANS = DATA['LOANS'][DATA['LOANS']['STATUS'] == 'ACTIVE']
  IF ACTIVE_LOANS.EMPTY:
  PRINT("NO ACTIVE LOANS FOUND.")
  RETURN DATA

  PRINT("\NACTIVE LOANS:")
  PRINT(ACTIVE_LOANS[['LOANID', 'LOANTYPE', 'EMI',
'OUTSTANDINGAMOUNT']].TO_STRING(INDEX=FALSE))

  LOAN_ID = INPUT("\NENTER LOAN ID: ").STRIP()
  LOAN = DATA['LOANS'][DATA['LOANS']['LOANID'] == LOAN_ID]

  IF LOAN.EMPTY OR LOAN.ILOC[0]['STATUS'] != 'ACTIVE':
  PRINT(F"{COLORS.RED}LOAN NOT FOUND OR NOT ACTIVE{COLORS.END}")
  RETURN DATA

  LOAN_ROW = LOAN.ILOC[0]
  EMI = LOAN_ROW['EMI']
  OUTSTANDING = LOAN_ROW['OUTSTANDINGAMOUNT']
  PRINT(F"\NEMI AMOUNT: ₹{EMI:.2F}")
  PRINT(F"OUTSTANDING: ₹{OUTSTANDING:.2F}")

  # CALCULATE INTEREST AND PRINCIPAL PORTIONS
  MONTHLY_RATE = LOAN_ROW['INTERESTRATE'] / (12 * 100)
  INTEREST_PART = OUTSTANDING * MONTHLY_RATE
  PRINCIPAL_PART = MIN(EMI - INTEREST_PART, OUTSTANDING
```

```
NEW_OUTSTANDING = OUTSTANDING - PRINCIPAL_PART

 # RECORD PAYMENT
 PAYMENT = {
 'PAYMENTID': F"PAY{LEN(DATA['LOAN_PAYMENTS'])+1:05D}",
 'LOANID': LOAN_ID,
 'PAYMENTDATE': GET_DATE(),
 'AMOUNTPAID': EMI,
 'PRINCIPALPART': ROUND(PRINCIPAL_PART, 2),
 'INTERESTPART': ROUND(INTEREST_PART, 2),
 'OUTSTANDINGAFTER': ROUND(NEW_OUTSTANDING, 2),
 'PAYMENTMETHOD': 'CASH',
 'STATUS': 'SUCCESS'
 }
 DATA['LOAN_PAYMENTS'] = PD.CONCAT([DATA['LOAN_PAYMENTS'],
PD.DATAFRAME([PAYMENT])], IGNORE_INDEX=TRUE)

 # UPDATE LOAN OUTSTANDING
 IDX = LOAN.INDEX[0]
 DATA['LOANS'].AT[IDX, 'OUTSTANDINGAMOUNT'] = ROUND(NEW_OUTSTANDING, 2)

 # CLOSE LOAN IF FULLY PAID
 IF NEW_OUTSTANDING <= 0:
 DATA['LOANS'].AT[IDX, 'STATUS'] = 'CLOSED'
 PRINT(F"{COLORS.GREEN}✓ LOAN FULLY PAID AND CLOSED!{COLORS.END}")
 ELSE:
 PRINT(F"{COLORS.GREEN}✓ EMI PAID! OUTSTANDING: ₹{NEW_OUTSTANDING:.2F}
{COLORS.END}")

 DATA = LOG_AUDIT(DATA, 'EMI_PAID', F'EMI ₹{EMI} FOR {LOAN_ID}')
 RETURN DATA

DEF VIEW_LOAN_DETAILS(DATA):
 """VIEW DETAILED LOAN INFORMATION WITH AMORTIZATION"""
 PRINT(F"\N{COLORS.CYAN}--- LOAN DETAILS ---{COLORS.END}")
 LOAN_ID = INPUT("ENTER LOAN ID: ").STRIP()

 LOAN = DATA['LOANS'][DATA['LOANS']['LOANID'] == LOAN_ID]
 IF LOAN.EMPTY:
 PRINT("LOAN NOT FOUND.")
 RETURN

 ROW = LOAN.ILOC[0]

 # HANDLE DIFFERENT COLUMN NAMES FOR TENURE
 TENURE = ROW.GET('TENURE_MONTHS') OR ROW.GET('TENURE') OR 'N/A'
 IF PD.ISNA(TENURE):
 TENURE = 'N/A'
```

```
PRINCIPAL = FLOAT(ROW.GET('PRINCIPALAMOUNT', 0) OR 0)
INTEREST_RATE = FLOAT(ROW.GET('INTERESTRATE', 0) OR 0)
EMI = FLOAT(ROW.GET('EMI', 0) OR 0)
OUTSTANDING = FLOAT(ROW.GET('OUTSTANDINGAMOUNT', 0) OR 0)

PRINT(F"\N{'='*50}")
PRINT(F"LOAN ID: {ROW['LOANID']}")
PRINT(F"TYPE: {ROW.GET('LOANTYPE', 'N/A')}")
PRINT(F"PRINCIPAL: ₹{PRINCIPAL:,.2F}")
PRINT(F"INTEREST RATE: {INTEREST_RATE}% P.A.")
PRINT(F"TENURE: {TENURE} MONTHS")
PRINT(F"EMI: ₹{EMI:,.2F}")
PRINT(F"OUTSTANDING: ₹{OUTSTANDING:,.2F}")
PRINT(F"STATUS: {ROW.GET('STATUS', 'N/A')}")

# CALCULATE REPAYMENT PROGRESS
IF PRINCIPAL > 0:
REPAID = PRINCIPAL - OUTSTANDING
PROGRESS = (REPAID / PRINCIPAL) * 100
PRINT(F"REPAID: ₹{REPAID:,.2F} ({PROGRESS:.1F}%)")
PRINT(F"{'='*50}")

# SHOW PAYMENT HISTORY
PAYMENTS = DATA['LOAN_PAYMENTS'][DATA['LOAN_PAYMENTS']['LOANID'] == LOAN_ID]
IF NOT PAYMENTS.EMPTY:
PRINT(F"\N{COLORS.YELLOW}PAYMENT HISTORY:{COLORS.END}")
# HANDLE DIFFERENT COLUMN NAMES
DISPLAY_COLS = []
FOR COL IN ['PAYMENTDATE', 'DATE', 'AMOUNTPAID', 'AMOUNT', 'PRINCIPALPART',
'INTERESTPART']:
IF COL IN PAYMENTS.COLUMNS:
DISPLAY_COLS.APPEND(COL)
IF DISPLAY_COLS:
PAYMENT_DISPLAY = PAYMENTS[DISPLAY_COLS].COPY()
# FORMAT AMOUNT COLUMNS
FOR COL IN PAYMENT_DISPLAY.COLUMNS:
IF 'AMOUNT' IN COL OR 'PART' IN COL:
PAYMENT_DISPLAY[COL] = PAYMENT_DISPLAY[COL].APPLY(
LAMBDA X: F"₹{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A'
)
PRINT(PAYMENT_DISPLAY.TO_STRING(INDEX=FALSE))
ELSE:
PRINT("NO PAYMENT DETAILS AVAILABLE.")
ELSE:
PRINT("\NNO PAYMENTS RECORDED YET.")
```

```
# ========================================
# SECTION 5: FUND TRANSFER SYSTEM
# ========================================

DEF TRANSFER_FUNDS(DATA):
 """TRANSFER FUNDS BETWEEN ACCOUNTS"""
 PRINT(F"\N{COLORS.CYAN}--- FUND TRANSFER ---{COLORS.END}")
 PRINT("1. TRANSFER TO OWN ACCOUNT")
 PRINT("2. TRANSFER TO OTHER CUSTOMER")

 CHOICE = INPUT("SELECT: ").STRIP()

 FROM_ACC = INPUT("FROM ACCOUNT NUMBER: ").STRIP()
 FROM_ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] ==
FROM_ACC]

 IF FROM_ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}SOURCE ACCOUNT NOT FOUND{COLORS.END}")
 RETURN DATA

 TO_ACC = INPUT("TO ACCOUNT NUMBER: ").STRIP()
 TO_ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == TO_ACC]

 IF TO_ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}DESTINATION ACCOUNT NOT FOUND{COLORS.END}")
 RETURN DATA

 IF FROM_ACC == TO_ACC:
 PRINT(F"{COLORS.RED}CANNOT TRANSFER TO SAME ACCOUNT{COLORS.END}")
 RETURN DATA

 AMOUNT = FLOAT(INPUT("AMOUNT: ₹").STRIP())

 FROM_IDX = FROM_ACCOUNT.INDEX[0]
 FROM_BAL = DATA['ACCOUNTS'].AT[FROM_IDX, 'BALANCE']
 MIN_BAL = DATA['ACCOUNTS'].AT[FROM_IDX, 'MINBALANCE']

 IF FROM_BAL - AMOUNT < MIN_BAL:
 PRINT(F"{COLORS.RED}INSUFFICIENT BALANCE. MIN BALANCE: ₹{MIN_BAL}
{COLORS.END}")
 RETURN DATA

 IF AMOUNT > DAILY_TRANSFER_LIMIT:
 PRINT(F"{COLORS.RED}EXCEEDS DAILY TRANSFER LIMIT OF ₹{DAILY_TRANSFER_LIMIT:,}
{COLORS.END}")
 RETURN DATA
```

```
# PROCESS TRANSFER
 TO_IDX = TO_ACCOUNT.INDEX[0]

 NEW_FROM_BAL = FROM_BAL - AMOUNT
 NEW_TO_BAL = DATA['ACCOUNTS'].AT[TO_IDX, 'BALANCE'] + AMOUNT

 DATA['ACCOUNTS'].AT[FROM_IDX, 'BALANCE'] = NEW_FROM_BAL
 DATA['ACCOUNTS'].AT[TO_IDX, 'BALANCE'] = NEW_TO_BAL

 # GENERATE TRANSFER REFERENCE
 TRANSFER_REF = F"TRF{DATETIME.NOW().STRFTIME('%Y%M%D%H%M%S')}"

 # RECORD TRANSFER
 TRANSFER_TYPE = 'INTERNAL' IF CHOICE == '1' ELSE 'INTER-CUSTOMER'
 TRANSFER_RECORD = {
 'TRANSFERID': F"XFER{LEN(DATA['TRANSFERS'])+1:05D}",
 'FROMACCOUNT': FROM_ACC,
 'TOACCOUNT': TO_ACC,
 'AMOUNT': AMOUNT,
 'TRANSFERTYPE': TRANSFER_TYPE,
 'CHARGES': 0,
 'DATE': GET_DATE(),
 'STATUS': 'SUCCESS',
 'REFERENCE': TRANSFER_REF
 }
 DATA['TRANSFERS'] = PD.CONCAT([DATA['TRANSFERS'],
PD.DATAFRAME([TRANSFER_RECORD])], IGNORE_INDEX=TRUE)

 # CREATE DEBIT TRANSACTION FOR SENDER
 TXN_DEBIT = {
 'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
 'ACCOUNTNUMBER': FROM_ACC,
 'TRANSACTIONTYPE': 'FUND TRANSFER',
 'AMOUNT': AMOUNT,
 'DEBITCREDIT': 'DEBIT',
 'BALANCE_AFTER': NEW_FROM_BAL,
 'DATE': GET_DATE(),
 'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
 'REMARKS': F'TRANSFER TO {TO_ACC} REF:{TRANSFER_REF}',
 'STATUS': 'SUCCESS'
 }
 DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'],
PD.DATAFRAME([TXN_DEBIT])], IGNORE_INDEX=TRUE)

 # CREATE CREDIT TRANSACTION FOR RECEIVER
 TXN_CREDIT = {
 'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
 'ACCOUNTNUMBER': TO_ACC,
 'TRANSACTIONTYPE': 'FUND TRANSFER',
```

```
 'AMOUNT': AMOUNT,
  'DEBITCREDIT': 'DEBIT',
  'BALANCE_AFTER': NEW_FROM_BAL,
  'DATE': GET_DATE(),
  'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
  'REMARKS': F'TRANSFER TO {TO_ACC} REF:{TRANSFER_REF}',
  'STATUS': 'SUCCESS'
  }
  DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'],
PD.DATAFRAME([TXN_DEBIT])], IGNORE_INDEX=TRUE)

  # CREATE CREDIT TRANSACTION FOR RECEIVER
  TXN_CREDIT = {
  'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
  'ACCOUNTNUMBER': TO_ACC,
  'TRANSACTIONTYPE': 'FUND TRANSFER',
  'AMOUNT': AMOUNT,
  'DEBITCREDIT': 'CREDIT',
  'BALANCE_AFTER': NEW_TO_BAL,
  'DATE': GET_DATE(),
  'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
  'REMARKS': F'TRANSFER FROM {FROM_ACC} REF:{TRANSFER_REF}',
  'STATUS': 'SUCCESS'
  }
  DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'],
PD.DATAFRAME([TXN_CREDIT])], IGNORE_INDEX=TRUE)

  PRINT(F"{COLORS.GREEN}✓ TRANSFER SUCCESSFUL!{COLORS.END}")
  PRINT(F"REFERENCE: {TRANSFER_REF}")
  PRINT(F"AMOUNT: ₹{AMOUNT:,.2F}")
  PRINT(F"FROM {FROM_ACC}: ₹{NEW_FROM_BAL:,.2F}")
  PRINT(F"TO {TO_ACC}: ₹{NEW_TO_BAL:,.2F}")

  DATA = LOG_AUDIT(DATA, 'FUND_TRANSFER', F'₹{AMOUNT} FROM {FROM_ACC} TO
{TO_ACC}')
  RETURN DATA

# ========================================
# SECTION 6: CARD MANAGEMENT SYSTEM
# ========================================

DEF CARD_MANAGEMENT_MENU(DATA):
  """CARD MANAGEMENT SUB-MENU"""
  WHILE TRUE:
  PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}=== CARD MANAGEMENT ==={COLORS.END}")
  PRINT("1. ISSUE NEW CARD")
  PRINT("2. VIEW MY CARDS")
  PRINT("3. BLOCK/UNBLOCK CARD")
  PRINT("4. VIEW CARD TRANSACTIONS")
```

```
 PRINT("5. CHANGE CARD PIN")
 PRINT("6. BACK TO MAIN MENU")

 CHOICE = INPUT("\NSELECT: ").STRIP()

 IF CHOICE == '1': DATA = ISSUE_NEW_CARD(DATA)
 ELIF CHOICE == '2': VIEW_CARDS(DATA)
 ELIF CHOICE == '3': DATA = TOGGLE_CARD_STATUS(DATA)
 ELIF CHOICE == '4': VIEW_CARD_TRANSACTIONS(DATA)
 ELIF CHOICE == '5': DATA = CHANGE_CARD_PIN(DATA)
 ELIF CHOICE == '6': BREAK
 ELSE: PRINT("INVALID OPTION.")

 RETURN DATA

DEF ISSUE_NEW_CARD(DATA):
 """ISSUE A NEW DEBIT/CREDIT CARD"""
 PRINT(F"\N{COLORS.CYAN}--- ISSUE NEW CARD ---{COLORS.END}")

 CUST_ID = INPUT("CUSTOMER ID: ").STRIP()
 IF DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUST_ID].EMPTY:
 PRINT(F"{COLORS.RED}CUSTOMER NOT FOUND{COLORS.END}")
 RETURN DATA

 # SHOW CUSTOMER'S ACCOUNTS
 ACCOUNTS = DATA['ACCOUNTS'][DATA['ACCOUNTS']['CUSTOMERID'] == CUST_ID]
 IF ACCOUNTS.EMPTY:
 PRINT(F"{COLORS.RED}NO ACCOUNTS FOUND FOR THIS CUSTOMER{COLORS.END}")
 RETURN DATA

 PRINT("\NCUSTOMER ACCOUNTS:")
 PRINT(ACCOUNTS[['ACCOUNTNUMBER', 'ACCOUNTTYPE',
'BALANCE']].TO_STRING(INDEX=FALSE))

 ACC_NUM = INPUT("\NLINK TO ACCOUNT NUMBER: ").STRIP()
 IF DATA['ACCOUNTS'][(DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM) &
 (DATA['ACCOUNTS']['CUSTOMERID'] == CUST_ID)].EMPTY:
 PRINT(F"{COLORS.RED}ACCOUNT NOT FOUND OR DOESN'T BELONG TO
CUSTOMER{COLORS.END}")
 RETURN DATA

 PRINT("\NCARD TYPE:")
 PRINT("1. DEBIT CARD")
 PRINT("2. CREDIT CARD")
 CARD_CHOICE = INPUT("SELECT: ").STRIP()

 CARD_TYPE = 'DEBIT' IF CARD_CHOICE == '1' ELSE 'CREDIT'
 CREDIT_LIMIT = 0
```

```
 IF CARD_TYPE == 'CREDIT':
  CREDIT_LIMIT = FLOAT(INPUT(F"CREDIT LIMIT (DEFAULT ₹{CREDIT_CARD_LIMIT_DEFAULT}):
 ").STRIP() OR CREDIT_CARD_LIMIT_DEFAULT)

  # GENERATE CARD DETAILS
  CARD_NUMBER = GENERATE_CARD_NUMBER()
  CVV = GENERATE_CVV()
  PIN = INPUT("SET 4-DIGIT PIN: ").STRIP()

  IF LEN(PIN) != 4 OR NOT PIN.ISDIGIT():
  PRINT(F"{COLORS.RED}INVALID PIN FORMAT{COLORS.END}")
  RETURN DATA

  ISSUE_DATE = GET_DATE()
  EXPIRY_DATE = (DATETIME.NOW() +
 TIMEDELTA(DAYS=CARD_VALIDITY_YEARS*365)).STRFTIME("%Y-%M-%D")

  NEW_CARD = {
  'CARDNUMBER': CARD_NUMBER,
  'CUSTOMERID': CUST_ID,
  'LINKEDACCOUNT': ACC_NUM,
  'CARDTYPE': CARD_TYPE,
  'CREDITLIMIT': CREDIT_LIMIT,
  'ISSUEDATE': ISSUE_DATE,
  'EXPIRYDATE': EXPIRY_DATE,
  'CVV': CVV,
  'PIN_HASH': HASH_PASSWORD(PIN),
  'STATUS': 'ACTIVE'
  }

  DATA['CARDS'] = PD.CONCAT([DATA['CARDS'], PD.DATAFRAME([NEW_CARD])],
 IGNORE_INDEX=TRUE)

  PRINT(F"\N{COLORS.GREEN}✓ CARD ISSUED SUCCESSFULLY!{COLORS.END}")
  PRINT(F"{'='*40}")
  PRINT(F"CARD NUMBER: {MASK_CARD_NUMBER(CARD_NUMBER)}")
  PRINT(F"CARD TYPE: {CARD_TYPE}")
  PRINT(F"LINKED A/C: {ACC_NUM}")
  PRINT(F"VALID TILL: {EXPIRY_DATE}")
  PRINT(F"CVV: {CVV} (KEEP SECRET!)")
  PRINT(F"{'='*40}")
  PRINT(F"{COLORS.YELLOW}NOTE: FULL CARD NUMBER WILL BE PRINTED ON PHYSICAL
 CARD.{COLORS.END}")

  DATA = LOG_AUDIT(DATA, 'CARD_ISSUED', F'{CARD_TYPE} CARD FOR {CUST_ID}')
  RETURN DATA
```

```
DEF VIEW_CARDS(DATA):
 """VIEW CARDS FOR A CUSTOMER"""
 PRINT(F"\N{COLORS.CYAN}--- VIEW CARDS ---{COLORS.END}")
 CUST_ID = INPUT("CUSTOMER ID: ").STRIP()

 CARDS = DATA['CARDS'][DATA['CARDS']['CUSTOMERID'] == CUST_ID]
 IF CARDS.EMPTY:
 PRINT("NO CARDS FOUND.")
 RETURN

 PRINT(F"\N{'='*70}")
 FOR _, CARD IN CARDS.ITERROWS():
 CARD_NUM = STR(CARD.GET('CARDNUMBER', 'N/A'))
 PRINT(F"CARD: {MASK_CARD_NUMBER(CARD_NUM)}")
 PRINT(F"TYPE: {CARD.GET('CARDTYPE', 'N/A')}")
 # HANDLE BOTH LINKEDACCOUNT AND ACCOUNTNUMBER COLUMN NAMES
 LINKED_ACC = CARD.GET('LINKEDACCOUNT') OR CARD.GET('ACCOUNTNUMBER') OR 'NOT
LINKED'
 IF PD.ISNA(LINKED_ACC) OR LINKED_ACC IS NONE:
 LINKED_ACC = 'NOT LINKED'
 PRINT(F"ACCOUNT: {LINKED_ACC}")
 PRINT(F"EXPIRY: {CARD.GET('EXPIRYDATE', 'N/A')}")
 PRINT(F"STATUS: {CARD.GET('STATUS', 'N/A')}")
 CARD_TYPE = CARD.GET('CARDTYPE', '')
 IF CARD_TYPE == 'CREDIT' OR 'CREDIT' IN STR(CARD_TYPE):
 CREDIT_LIMIT = CARD.GET('CREDITLIMIT', 0)
 IF PD.NOTNA(CREDIT_LIMIT):
 PRINT(F"LIMIT: ₹{FLOAT(CREDIT_LIMIT):,.2F}")
 PRINT(F"{'-'*70}")

DEF TOGGLE_CARD_STATUS(DATA):
 """BLOCK OR UNBLOCK A CARD"""
 PRINT(F"\N{COLORS.CYAN}--- BLOCK/UNBLOCK CARD ---{COLORS.END}")

 CARD_NUM = INPUT("ENTER LAST 4 DIGITS OF CARD: ").STRIP()
 CARDS = DATA['CARDS'][DATA['CARDS']['CARDNUMBER'].STR.ENDSWITH(CARD_NUM)]

 IF CARDS.EMPTY:
 PRINT("CARD NOT FOUND.")
 RETURN DATA

 CARD = CARDS.ILOC[0]
 IDX = CARDS.INDEX[0]
 CURRENT_STATUS = CARD['STATUS']

 PRINT(F"CURRENT STATUS: {CURRENT_STATUS}")

 IF CURRENT_STATUS == 'ACTIVE':
 CONFIRM = INPUT("BLOCK THIS CARD? (YES/NO): ").STRIP().LOWER()
```

```
IF CONFIRM == 'YES':
 DATA['CARDS'].AT[IDX, 'STATUS'] = 'BLOCKED'
 PRINT(F"{COLORS.GREEN}✓ CARD BLOCKED SUCCESSFULLY{COLORS.END}")
 DATA = LOG_AUDIT(DATA, 'CARD_BLOCKED', F'CARD ENDING {CARD_NUM}')
 ELSE:
 CONFIRM = INPUT("UNBLOCK THIS CARD? (YES/NO): ").STRIP().LOWER()
 IF CONFIRM == 'YES':
 DATA['CARDS'].AT[IDX, 'STATUS'] = 'ACTIVE'
 PRINT(F"{COLORS.GREEN}✓ CARD UNBLOCKED SUCCESSFULLY{COLORS.END}")
 DATA = LOG_AUDIT(DATA, 'CARD_UNBLOCKED', F'CARD ENDING {CARD_NUM}')

 RETURN DATA

DEF VIEW_CARD_TRANSACTIONS(DATA):
 """VIEW TRANSACTIONS MADE BY A CARD'S LINKED ACCOUNT"""
 PRINT(F"\N{COLORS.CYAN}--- CARD TRANSACTIONS ---{COLORS.END}")

 CARD_NUM = INPUT("ENTER LAST 4 DIGITS OF CARD: ").STRIP()
 CARDS = DATA['CARDS'][DATA['CARDS']['CARDNUMBER'].STR.ENDSWITH(CARD_NUM)]

 IF CARDS.EMPTY:
 PRINT("CARD NOT FOUND.")
 RETURN

 LINKED_ACC = CARDS.ILOC[0]['LINKEDACCOUNT']
 TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['ACCOUNTNUMBER'] ==
LINKED_ACC].TAIL(10)

 IF TXNS.EMPTY:
 PRINT("NO TRANSACTIONS FOUND.")
 RETURN

 PRINT(F"\NLAST 10 TRANSACTIONS FOR CARD ENDING {CARD_NUM}:")
 PRINT(TXNS[['DATE', 'TRANSACTIONTYPE', 'AMOUNT', 'DEBITCREDIT',
'BALANCE_AFTER']].TO_STRING(INDEX=FALSE))

DEF CHANGE_CARD_PIN(DATA):
 """CHANGE CARD PIN"""
 PRINT(F"\N{COLORS.CYAN}--- CHANGE CARD PIN ---{COLORS.END}")

 CARD_NUM = INPUT("ENTER LAST 4 DIGITS OF CARD: ").STRIP()
 CARDS = DATA['CARDS'][DATA['CARDS']['CARDNUMBER'].STR.ENDSWITH(CARD_NUM)]

 IF CARDS.EMPTY:
 PRINT("CARD NOT FOUND.")
 RETURN DATA

 IDX = CARDS.INDEX[0]
 OLD_PIN = INPUT("CURRENT PIN: ").STRIP()
```

```
IF NOT VERIFY_PASSWORD(OLD_PIN, CARDS.ILOC[0]['PIN_HASH']):
 PRINT(F"{COLORS.RED}INCORRECT PIN{COLORS.END}")
 RETURN DATA

 NEW_PIN = INPUT("NEW 4-DIGIT PIN: ").STRIP()
 CONFIRM_PIN = INPUT("CONFIRM NEW PIN: ").STRIP()

 IF NEW_PIN != CONFIRM_PIN:
 PRINT(F"{COLORS.RED}PINS DON'T MATCH{COLORS.END}")
 RETURN DATA

 IF LEN(NEW_PIN) != 4 OR NOT NEW_PIN.ISDIGIT():
 PRINT(F"{COLORS.RED}INVALID PIN FORMAT{COLORS.END}")
 RETURN DATA

 DATA['CARDS'].AT[IDX, 'PIN_HASH'] = HASH_PASSWORD(NEW_PIN)
 PRINT(F"{COLORS.GREEN}✓ PIN CHANGED SUCCESSFULLY{COLORS.END}")

 DATA = LOG_AUDIT(DATA, 'PIN_CHANGED', F'CARD ENDING {CARD_NUM}')
 RETURN DATA

# ========================================
# SECTION 7: CHEQUE PROCESSING SYSTEM
# ========================================

DEF CHEQUE_MANAGEMENT_MENU(DATA):
 """CHEQUE PROCESSING SUB-MENU"""
 WHILE TRUE:
 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}=== CHEQUE PROCESSING ===
{COLORS.END}")
 PRINT("1. ISSUE NEW CHEQUE")
 PRINT("2. DEPOSIT CHEQUE")
 PRINT("3. VIEW CHEQUE STATUS")
 PRINT("4. CANCEL CHEQUE")
 PRINT("5. VIEW ALL CHEQUES")
 PRINT("6. BACK TO MAIN MENU")

 CHOICE = INPUT("\NSELECT: ").STRIP()

 IF CHOICE == '1': DATA = ISSUE_CHEQUE(DATA)
 ELIF CHOICE == '2': DATA = DEPOSIT_CHEQUE(DATA)
 ELIF CHOICE == '3': CHECK_CHEQUE_STATUS(DATA)
 ELIF CHOICE == '4': DATA = CANCEL_CHEQUE(DATA)
 ELIF CHOICE == '5': VIEW_ALL_CHEQUES(DATA)
 ELIF CHOICE == '6': BREAK
 ELSE: PRINT("INVALID OPTION.")

 RETURN DATA
```

```
DEF ISSUE_CHEQUE(DATA):
 """ISSUE A NEW CHEQUE"""
 PRINT(F"\N{COLORS.CYAN}--- ISSUE CHEQUE ---{COLORS.END}")

 ACC_NUM = INPUT("FROM ACCOUNT NUMBER: ").STRIP()
 ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]

 IF ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}ACCOUNT NOT FOUND{COLORS.END}")
 RETURN DATA

 ISSUED_TO = INPUT("PAYEE NAME: ").STRIP()
 AMOUNT = FLOAT(INPUT("AMOUNT: ₹").STRIP())

 BALANCE = ACCOUNT.ILOC[0]['BALANCE']
 MIN_BAL = ACCOUNT.ILOC[0]['MINBALANCE']

 IF BALANCE - AMOUNT < MIN_BAL:
 PRINT(F"{COLORS.RED}INSUFFICIENT BALANCE FOR CHEQUE AMOUNT{COLORS.END}")
 RETURN DATA

 CHEQUE_NUM = GENERATE_CHEQUE_NUMBER()

 NEW_CHEQUE = {
 'CHEQUENUMBER': CHEQUE_NUM,
 'ACCOUNTNUMBER': ACC_NUM,
 'ISSUEDTO': ISSUED_TO,
 'AMOUNT': AMOUNT,
 'ISSUEDATE': GET_DATE(),
 'CLEARANCEDATE': NONE,
 'STATUS': 'ISSUED',
 'REMARKS': ''
 }

 DATA['CHEQUES'] = PD.CONCAT([DATA['CHEQUES'], PD.DATAFRAME([NEW_CHEQUE])],
IGNORE_INDEX=TRUE)

 PRINT(F"\N{COLORS.GREEN}✓ CHEQUE ISSUED{COLORS.END}")
 PRINT(F"CHEQUE NUMBER: {CHEQUE_NUM}")
 PRINT(F"AMOUNT: ₹{AMOUNT:,.2F}")
 PRINT(F"PAYEE: {ISSUED_TO}")

 DATA = LOG_AUDIT(DATA, 'CHEQUE_ISSUED', F'CHEQUE {CHEQUE_NUM} FOR ₹
{AMOUNT}')
 RETURN DATA

DEF DEPOSIT_CHEQUE(DATA):
 """DEPOSIT/CLEAR A CHEQUE"""
 PRINT(F"\N{COLORS.CYAN}--- DEPOSIT CHEQUE ---{COLORS.END}")
```

```
CHEQUE_NUM = INPUT("CHEQUE NUMBER: ").STRIP()
 CHEQUE = DATA['CHEQUES'][DATA['CHEQUES']['CHEQUENUMBER'] == CHEQUE_NUM]

 IF CHEQUE.EMPTY:
 PRINT(F"{COLORS.RED}CHEQUE NOT FOUND{COLORS.END}")
 RETURN DATA

 CHEQUE_ROW = CHEQUE.ILOC[0]

 IF CHEQUE_ROW['STATUS'] != 'ISSUED':
 PRINT(F"{COLORS.RED}CHEQUE CANNOT BE DEPOSITED. STATUS:
{CHEQUE_ROW['STATUS']}{COLORS.END}")
 RETURN DATA

 TO_ACC = INPUT("DEPOSIT TO ACCOUNT NUMBER: ").STRIP()
 TO_ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == TO_ACC]

 IF TO_ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}DESTINATION ACCOUNT NOT FOUND{COLORS.END}")
 RETURN DATA

 # CHECK IF SOURCE ACCOUNT HAS SUFFICIENT BALANCE
 FROM_ACC = CHEQUE_ROW['ACCOUNTNUMBER']
 FROM_ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] ==
FROM_ACC]
 FROM_IDX = FROM_ACCOUNT.INDEX[0]
 FROM_BAL = DATA['ACCOUNTS'].AT[FROM_IDX, 'BALANCE']
 MIN_BAL = DATA['ACCOUNTS'].AT[FROM_IDX, 'MINBALANCE']
 AMOUNT = CHEQUE_ROW['AMOUNT']

 IF FROM_BAL - AMOUNT < MIN_BAL:
 # BOUNCE THE CHEQUE
 IDX = CHEQUE.INDEX[0]
 DATA['CHEQUES'].AT[IDX, 'STATUS'] = 'BOUNCED'
 DATA['CHEQUES'].AT[IDX, 'REMARKS'] = 'INSUFFICIENT FUNDS'
 PRINT(F"{COLORS.RED}CHEQUE BOUNCED! INSUFFICIENT FUNDS IN ISSUER ACCOUNT.
{COLORS.END}")
 DATA = LOG_AUDIT(DATA, 'CHEQUE_BOUNCED', F'CHEQUE {CHEQUE_NUM} BOUNCED')
 RETURN DATA

 # PROCESS CHEQUE CLEARANCE
 # DEBIT FROM SOURCE
 NEW_FROM_BAL = FROM_BAL - AMOUNT
 DATA['ACCOUNTS'].AT[FROM_IDX, 'BALANCE'] = NEW_FROM_BAL

 # CREDIT TO DESTINATION
 TO_IDX = TO_ACCOUNT.INDEX[0]
 NEW_TO_BAL = DATA['ACCOUNTS'].AT[TO_IDX, 'BALANCE'] + AMOUNT
 DATA['ACCOUNTS'].AT[TO_IDX, 'BALANCE'] = NEW_TO_BAL
```

```
 # UPDATE CHEQUE STATUS
  IDX = CHEQUE.INDEX[0]
  DATA['CHEQUES'].AT[IDX, 'STATUS'] = 'CLEARED'
  DATA['CHEQUES'].AT[IDX, 'CLEARANCEDATE'] = GET_DATE()

  # CREATE TRANSACTIONS
  TXN_DEBIT = {
  'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
  'ACCOUNTNUMBER': FROM_ACC,
  'TRANSACTIONTYPE': 'CHEQUE DEBIT',
  'AMOUNT': AMOUNT,
  'DEBITCREDIT': 'DEBIT',
  'BALANCE_AFTER': NEW_FROM_BAL,
  'DATE': GET_DATE(),
  'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
  'REMARKS': F'CHEQUE {CHEQUE_NUM} CLEARED',
  'STATUS': 'SUCCESS'
  }
  DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'],
PD.DATAFRAME([TXN_DEBIT])], IGNORE_INDEX=TRUE)

  TXN_CREDIT = {
  'TRANSACTIONID': F"TXN{LEN(DATA['TRANSACTIONS'])+1:05D}",
  'ACCOUNTNUMBER': TO_ACC,
  'TRANSACTIONTYPE': 'CHEQUE CREDIT',
  'AMOUNT': AMOUNT,
  'DEBITCREDIT': 'CREDIT',
  'BALANCE_AFTER': NEW_TO_BAL,
  'DATE': GET_DATE(),
  'TIME': DATETIME.NOW().STRFTIME("%H:%M:%S"),
  'REMARKS': F'CHEQUE {CHEQUE_NUM} DEPOSITED',
  'STATUS': 'SUCCESS'
  }
  DATA['TRANSACTIONS'] = PD.CONCAT([DATA['TRANSACTIONS'],
PD.DATAFRAME([TXN_CREDIT])], IGNORE_INDEX=TRUE)

  PRINT(F"\N{COLORS.GREEN}✓ CHEQUE CLEARED SUCCESSFULLY!{COLORS.END}")
  PRINT(F"AMOUNT: ₹{AMOUNT:,.2F}")
  PRINT(F"DEPOSITED TO: {TO_ACC}")

  DATA = LOG_AUDIT(DATA, 'CHEQUE_CLEARED', F'CHEQUE {CHEQUE_NUM} FOR ₹
{AMOUNT}')
  RETURN DATA

DEF CHECK_CHEQUE_STATUS(DATA):
  """CHECK STATUS OF A CHEQUE"""
  PRINT(F"\N{COLORS.CYAN}--- CHEQUE STATUS ---{COLORS.END}")
```

```
CHEQUE_NUM = INPUT("CHEQUE NUMBER: ").STRIP()
 CHEQUE = DATA['CHEQUES'][DATA['CHEQUES']['CHEQUENUMBER'] == CHEQUE_NUM]

 IF CHEQUE.EMPTY:
 PRINT("CHEQUE NOT FOUND.")
 RETURN

 ROW = CHEQUE.ILOC[0]
 PRINT(F"\NCHEQUE NUMBER: {ROW['CHEQUENUMBER']}")
 PRINT(F"FROM ACCOUNT: {ROW['ACCOUNTNUMBER']}")
 PRINT(F"PAYEE: {ROW['ISSUEDTO']}")
 PRINT(F"AMOUNT: ₹{ROW['AMOUNT']:,.2F}")
 PRINT(F"ISSUE DATE: {ROW['ISSUEDATE']}")
 PRINT(F"STATUS: {ROW['STATUS']}")
 IF ROW['CLEARANCEDATE']:
 PRINT(F"CLEARED ON: {ROW['CLEARANCEDATE']}")
 IF ROW['REMARKS']:
 PRINT(F"REMARKS: {ROW['REMARKS']}")

DEF CANCEL_CHEQUE(DATA):
 """CANCEL AN ISSUED CHEQUE"""
 PRINT(F"\N{COLORS.CYAN}--- CANCEL CHEQUE ---{COLORS.END}")

 CHEQUE_NUM = INPUT("CHEQUE NUMBER: ").STRIP()
 CHEQUE = DATA['CHEQUES'][DATA['CHEQUES']['CHEQUENUMBER'] == CHEQUE_NUM]

 IF CHEQUE.EMPTY:
 PRINT("CHEQUE NOT FOUND.")
 RETURN DATA

 IF CHEQUE.ILOC[0]['STATUS'] != 'ISSUED':
 PRINT(F"{COLORS.RED}ONLY ISSUED CHEQUES CAN BE CANCELLED{COLORS.END}")
 RETURN DATA

 CONFIRM = INPUT("CANCEL THIS CHEQUE? (YES/NO): ").STRIP().LOWER()
 IF CONFIRM == 'YES':
 IDX = CHEQUE.INDEX[0]
 DATA['CHEQUES'].AT[IDX, 'STATUS'] = 'CANCELLED'
 DATA['CHEQUES'].AT[IDX, 'REMARKS'] = 'CANCELLED BY ACCOUNT HOLDER'
 PRINT(F"{COLORS.GREEN}✓ CHEQUE CANCELLED{COLORS.END}")
 DATA = LOG_AUDIT(DATA, 'CHEQUE_CANCELLED', F'CHEQUE {CHEQUE_NUM}')

 RETURN DATA

DEF VIEW_ALL_CHEQUES(DATA):
 """VIEW ALL CHEQUES FOR AN ACCOUNT"""
 PRINT(F"\N{COLORS.CYAN}--- ALL CHEQUES ---{COLORS.END}")
 ACC_NUM = INPUT("ACCOUNT NUMBER: ").STRIP()
 CHEQUES = DATA['CHEQUES'][DATA['CHEQUES']['ACCOUNTNUMBER'] == ACC_NUM]
```

```
IF CHEQUES.EMPTY:
 PRINT("NO CHEQUES FOUND.")
 RETURN

 PRINT(CHEQUES[['CHEQUENUMBER', 'ISSUEDTO', 'AMOUNT', 'STATUS',
'ISSUEDATE']].TO_STRING(INDEX=FALSE))

# ========================================
# SECTION 8: REPORTS & ANALYTICS
# ========================================

DEF REPORT_TRANSACTION_HISTORY(DATA):
 PRINT(F"\N{COLORS.CYAN}--- TRANSACTION HISTORY ---{COLORS.END}")
 ACC_NUM = INPUT("ENTER ACCOUNT NUMBER: ").STRIP()

 TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['ACCOUNTNUMBER'] ==
ACC_NUM]
 IF TXNS.EMPTY:
 PRINT("NO TRANSACTIONS FOUND FOR THIS ACCOUNT.")
 RETURN

 # SORT BY DATE DESCENDING
 TXNS = TXNS.SORT_VALUES('DATE', ASCENDING=FALSE)

 PRINT(F"\NHISTORY FOR {ACC_NUM}:")
 PRINT("-" * 85)
 PRINT(F"{'DATE':<12} {'TYPE':<20} {'AMOUNT':>12} {'DR/CR':<8} {'BALANCE':>15}")
 PRINT("-" * 85)

 FOR _, ROW IN TXNS.ITERROWS():
 # HANDLE NONE/NAN VALUES SAFELY
 DATE_VAL = STR(ROW.GET('DATE', 'N/A') OR 'N/A')[:10]
 TXN_TYPE = STR(ROW.GET('TRANSACTIONTYPE', 'N/A') OR 'N/A')[:18]
 AMOUNT = FLOAT(ROW.GET('AMOUNT', 0) OR 0)
 DEBIT_CREDIT = STR(ROW.GET('DEBITCREDIT', 'N/A') OR 'N/A')

 # GET BALANCE - CHECK BOTH COLUMN NAMES
 BALANCE = ROW.GET('BALANCE_AFTER') OR ROW.GET('BALANCEAFTER') OR 0
 BALANCE = FLOAT(BALANCE) IF PD.NOTNA(BALANCE) ELSE 0

 PRINT(F"{DATE_VAL:<12} {TXN_TYPE:<20} ₹{AMOUNT:>10,.2F} {DEBIT_CREDIT:<8} ₹
{BALANCE:>12,.2F}")
 PRINT("-" * 85)

DEF REPORT_BANK_SUMMARY(DATA):
 PRINT(F"\N{COLORS.CYAN}--- BANK FINANCIAL SUMMARY ---{COLORS.END}")

 TOTAL_DEPOSITS = DATA['ACCOUNTS']['BALANCE'].SUM()
```

```
TOTAL_LOANS = DATA['LOANS']['OUTSTANDINGAMOUNT'].SUM() IF NOT
DATA['LOANS'].EMPTY ELSE 0
 TOTAL_CUSTOMERS = LEN(DATA['CUSTOMERS'])
 TOTAL_ACCOUNTS = LEN(DATA['ACCOUNTS'])
 TOTAL_CARDS = LEN(DATA['CARDS'])

 # CALCULATE TOTAL TRANSACTION VOLUME
 CREDIT_TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['DEBITCREDIT'] ==
'CREDIT']['AMOUNT'].SUM() IF NOT DATA['TRANSACTIONS'].EMPTY ELSE 0
 DEBIT_TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['DEBITCREDIT'] == 'DEBIT']
['AMOUNT'].SUM() IF NOT DATA['TRANSACTIONS'].EMPTY ELSE 0

 # LOAN STATISTICS
 ACTIVE_LOANS = LEN(DATA['LOANS'][DATA['LOANS']['STATUS'] == 'ACTIVE']) IF NOT
DATA['LOANS'].EMPTY ELSE 0

 PRINT(F"\N{'='*50}")
 PRINT(F"{COLORS.BOLD}FINANCIAL OVERVIEW{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL DEPOSITS HELD: ₹{TOTAL_DEPOSITS:>15,.2F}")
 PRINT(F"TOTAL LOANS OUTSTANDING: ₹{TOTAL_LOANS:>15,.2F}")
 PRINT(F"NET LIQUIDITY: ₹{(TOTAL_DEPOSITS - TOTAL_LOANS):>15,.2F}")
 PRINT(F"{'='*50}")
 PRINT(F"{COLORS.BOLD}TRANSACTION VOLUME{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL CREDIT VOLUME: ₹{CREDIT_TXNS:>15,.2F}")
 PRINT(F"TOTAL DEBIT VOLUME: ₹{DEBIT_TXNS:>15,.2F}")
 PRINT(F"NET FLOW: ₹{(CREDIT_TXNS - DEBIT_TXNS):>15,.2F}")
 PRINT(F"{'='*50}")
 PRINT(F"{COLORS.BOLD}STATISTICS{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL CUSTOMERS: {TOTAL_CUSTOMERS:>15}")
 PRINT(F"TOTAL ACCOUNTS: {TOTAL_ACCOUNTS:>15}")
 PRINT(F"ACTIVE LOANS: {ACTIVE_LOANS:>15}")
 PRINT(F"CARDS ISSUED: {TOTAL_CARDS:>15}")
 PRINT(F"{'='*50}")

DEF REPORT_CUSTOMER_BALANCES(DATA):
 PRINT(F"\N{COLORS.CYAN}--- CUSTOMER BALANCES REPORT ---{COLORS.END}")

 IF DATA['CUSTOMERS'].EMPTY:
 PRINT("NO CUSTOMERS.")
 RETURN

 PRINT(F"\N{'ID':<10} {'NAME':<25} {'TOTAL BALANCE':<15} {'ACCOUNTS':<8} {'LOANS':<6}")
 PRINT("-" * 70)

 FOR _, CUST IN DATA['CUSTOMERS'].ITERROWS():
 CID = CUST['CUSTOMERID']
```

```
TOTAL_LOANS = DATA['LOANS']['OUTSTANDINGAMOUNT'].SUM() IF NOT
DATA['LOANS'].EMPTY ELSE 0
 TOTAL_CUSTOMERS = LEN(DATA['CUSTOMERS'])
 TOTAL_ACCOUNTS = LEN(DATA['ACCOUNTS'])
 TOTAL_CARDS = LEN(DATA['CARDS'])

 # CALCULATE TOTAL TRANSACTION VOLUME
 CREDIT_TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['DEBITCREDIT'] ==
'CREDIT']['AMOUNT'].SUM() IF NOT DATA['TRANSACTIONS'].EMPTY ELSE 0
 DEBIT_TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['DEBITCREDIT'] == 'DEBIT']
['AMOUNT'].SUM() IF NOT DATA['TRANSACTIONS'].EMPTY ELSE 0

 # LOAN STATISTICS
 ACTIVE_LOANS = LEN(DATA['LOANS'][DATA['LOANS']['STATUS'] == 'ACTIVE']) IF NOT
DATA['LOANS'].EMPTY ELSE 0

 PRINT(F"\N{'='*50}")
 PRINT(F"{COLORS.BOLD}FINANCIAL OVERVIEW{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL DEPOSITS HELD: ₹{TOTAL_DEPOSITS:>15,.2F}")
 PRINT(F"TOTAL LOANS OUTSTANDING: ₹{TOTAL_LOANS:>15,.2F}")
 PRINT(F"NET LIQUIDITY: ₹{(TOTAL_DEPOSITS - TOTAL_LOANS):>15,.2F}")
 PRINT(F"{'='*50}")
 PRINT(F"{COLORS.BOLD}TRANSACTION VOLUME{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL CREDIT VOLUME: ₹{CREDIT_TXNS:>15,.2F}")
 PRINT(F"TOTAL DEBIT VOLUME: ₹{DEBIT_TXNS:>15,.2F}")
 PRINT(F"NET FLOW: ₹{(CREDIT_TXNS - DEBIT_TXNS):>15,.2F}")
 PRINT(F"{'='*50}")
 PRINT(F"{COLORS.BOLD}STATISTICS{COLORS.END}")
 PRINT(F"{'='*50}")
 PRINT(F"TOTAL CUSTOMERS: {TOTAL_CUSTOMERS:>15}")
 PRINT(F"TOTAL ACCOUNTS: {TOTAL_ACCOUNTS:>15}")
 PRINT(F"ACTIVE LOANS: {ACTIVE_LOANS:>15}")
 PRINT(F"CARDS ISSUED: {TOTAL_CARDS:>15}")
 PRINT(F"{'='*50}")

DEF REPORT_CUSTOMER_BALANCES(DATA):
 PRINT(F"\N{COLORS.CYAN}--- CUSTOMER BALANCES REPORT ---{COLORS.END}")

 IF DATA['CUSTOMERS'].EMPTY:
 PRINT("NO CUSTOMERS.")
 RETURN

 PRINT(F"\N{'ID':<10} {'NAME':<25} {'TOTAL BALANCE':<15} {'ACCOUNTS':<8} {'LOANS':<6}")
 PRINT("-" * 70)

 FOR _, CUST IN DATA['CUSTOMERS'].ITERROWS():
 CID = CUST['CUSTOMERID']
```

```
    LOANS = DATA['LOANS'][DATA['LOANS']['CUSTOMERID'] == CID] IF NOT
    DATA['LOANS'].EMPTY ELSE PD.DATAFRAME()
     TOTAL_BAL = ACCOUNTS['BALANCE'].SUM()
     ACC_COUNT = LEN(ACCOUNTS)
     LOAN_COUNT = LEN(LOANS)

     PRINT(F"{CID:<10} {CUST['NAME']:<25} ₹{TOTAL_BAL:<14,.2F} {ACC_COUNT:<8}
    {LOAN_COUNT:<6}")

    DEF REPORT_DAILY_TRANSACTIONS(DATA):
     """SHOW TODAY'S TRANSACTIONS SUMMARY"""
     PRINT(F"\N{COLORS.CYAN}--- DAILY TRANSACTION SUMMARY ---{COLORS.END}")

     TODAY = GET_DATE()
     TODAY_TXNS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['DATE'] == TODAY]

     IF TODAY_TXNS.EMPTY:
     PRINT("NO TRANSACTIONS TODAY.")
     RETURN

     CREDITS = TODAY_TXNS[TODAY_TXNS['DEBITCREDIT'] == 'CREDIT']
     DEBITS = TODAY_TXNS[TODAY_TXNS['DEBITCREDIT'] == 'DEBIT']

     PRINT(F"\NDATE: {TODAY}")
     PRINT(F"{'='*50}")
     PRINT(F"TOTAL TRANSACTIONS: {LEN(TODAY_TXNS)}")
     PRINT(F"{'='*50}")
     PRINT(F"CREDITS: {LEN(CREDITS)} TRANSACTIONS, ₹{CREDITS['AMOUNT'].SUM():,.2F}")
     PRINT(F"DEBITS: {LEN(DEBITS)} TRANSACTIONS, ₹{DEBITS['AMOUNT'].SUM():,.2F}")
     PRINT(F"{'='*50}")
     PRINT(F"\NTRANSACTION DETAILS:")
     PRINT(TODAY_TXNS[['TRANSACTIONID', 'ACCOUNTNUMBER', 'TRANSACTIONTYPE',
    'AMOUNT', 'DEBITCREDIT']].TO_STRING(INDEX=FALSE))

    DEF REPORT_LOAN_PORTFOLIO(DATA):
     """DETAILED LOAN PORTFOLIO ANALYSIS"""
     PRINT(F"\N{COLORS.CYAN}--- LOAN PORTFOLIO ANALYSIS ---{COLORS.END}")

     IF DATA['LOANS'].EMPTY:
     PRINT("NO LOANS IN SYSTEM.")
     RETURN

     PRINT(F"\N{'='*60}")
     PRINT(F"{COLORS.BOLD}LOAN PORTFOLIO SUMMARY{COLORS.END}")
     PRINT(F"{'='*60}")

     # BY STATUS
     STATUS_SUMMARY = DATA['LOANS'].GROUPBY('STATUS').AGG({
     'LOANID': 'COUNT',
```

```
'PRINCIPALAMOUNT': 'SUM',
 'OUTSTANDINGAMOUNT': 'SUM'
 }).RENAME(COLUMNS={'LOANID': 'COUNT'})

 PRINT(F"\NBY STATUS:")
 PRINT(STATUS_SUMMARY.TO_STRING())

 # BY TYPE
 PRINT(F"\N\NBY LOAN TYPE:")
 TYPE_SUMMARY = DATA['LOANS'].GROUPBY('LOANTYPE').AGG({
 'LOANID': 'COUNT',
 'PRINCIPALAMOUNT': 'SUM',
 'OUTSTANDINGAMOUNT': 'SUM',
 'INTERESTRATE': 'MEAN'
 }).RENAME(COLUMNS={'LOANID': 'COUNT', 'INTERESTRATE': 'AVG RATE'})
 PRINT(TYPE_SUMMARY.TO_STRING())

DEF VIEW_AUDIT_TRAIL(DATA):
 """VIEW SYSTEM AUDIT TRAIL"""
 PRINT(F"\N{COLORS.CYAN}--- AUDIT TRAIL ---{COLORS.END}")

 IF DATA['AUDIT'].EMPTY:
 PRINT("NO AUDIT LOGS.")
 RETURN

 PRINT("1. VIEW ALL LOGS")
 PRINT("2. FILTER BY ACTION TYPE")
 PRINT("3. FILTER BY DATE")

 CHOICE = INPUT("\NSELECT: ").STRIP()

 IF CHOICE == '1':
 LOGS = DATA['AUDIT'].TAIL(50)
 ELIF CHOICE == '2':
 ACTION = INPUT("ENTER ACTION TYPE (E.G., DEPOSIT, WITHDRAWAL, TRANSFER):
").STRIP().UPPER()
 LOGS = DATA['AUDIT'][DATA['AUDIT']['ACTION'].STR.CONTAINS(ACTION, CASE=FALSE,
NA=FALSE)]
 ELIF CHOICE == '3':
 DATE = INPUT("ENTER DATE (YYYY-MM-DD): ").STRIP()
 LOGS = DATA['AUDIT'][DATA['AUDIT']['TIMESTAMP'].STR.STARTSWITH(DATE)]
 ELSE:
 RETURN

 IF LOGS.EMPTY:
 PRINT("NO MATCHING LOGS FOUND.")
 RETURN
```

```
'PRINCIPALAMOUNT': 'SUM',
 'OUTSTANDINGAMOUNT': 'SUM'
 }).RENAME(COLUMNS={'LOANID': 'COUNT'})

 PRINT(F"\NBY STATUS:")
 PRINT(STATUS_SUMMARY.TO_STRING())

 # BY TYPE
 PRINT(F"\N\NBY LOAN TYPE:")
 TYPE_SUMMARY = DATA['LOANS'].GROUPBY('LOANTYPE').AGG({
 'LOANID': 'COUNT',
 'PRINCIPALAMOUNT': 'SUM',
 'OUTSTANDINGAMOUNT': 'SUM',
 'INTERESTRATE': 'MEAN'
 }).RENAME(COLUMNS={'LOANID': 'COUNT', 'INTERESTRATE': 'AVG RATE'})
 PRINT(TYPE_SUMMARY.TO_STRING())

DEF VIEW_AUDIT_TRAIL(DATA):
 """VIEW SYSTEM AUDIT TRAIL"""
 PRINT(F"\N{COLORS.CYAN}--- AUDIT TRAIL ---{COLORS.END}")

 IF DATA['AUDIT'].EMPTY:
 PRINT("NO AUDIT LOGS.")
 RETURN

 PRINT("1. VIEW ALL LOGS")
 PRINT("2. FILTER BY ACTION TYPE")
 PRINT("3. FILTER BY DATE")

 CHOICE = INPUT("\NSELECT: ").STRIP()

 IF CHOICE == '1':
 LOGS = DATA['AUDIT'].TAIL(50)
 ELIF CHOICE == '2':
 ACTION = INPUT("ENTER ACTION TYPE (E.G., DEPOSIT, WITHDRAWAL, TRANSFER):
").STRIP().UPPER()
 LOGS = DATA['AUDIT'][DATA['AUDIT']['ACTION'].STR.CONTAINS(ACTION, CASE=FALSE,
NA=FALSE)]
 ELIF CHOICE == '3':
 DATE = INPUT("ENTER DATE (YYYY-MM-DD): ").STRIP()
 LOGS = DATA['AUDIT'][DATA['AUDIT']['TIMESTAMP'].STR.STARTSWITH(DATE)]
 ELSE:
 RETURN

 IF LOGS.EMPTY:
 PRINT("NO MATCHING LOGS FOUND.")
 RETURN
```

```
PRINT(F"\N{'-'*100}")
 PRINT(F"{'TIMESTAMP':<20} {'ACTION':<20} {'DETAILS':<50} {'STATUS':<10}")
 PRINT(F"{'-'*100}")

 FOR _, LOG IN LOGS.ITERROWS():
 DETAILS = STR(LOG['DETAILS'])[:48] IF LOG['DETAILS'] ELSE ''
 PRINT(F"{STR(LOG['TIMESTAMP']):<20} {STR(LOG['ACTION']):<20} {DETAILS:<50}
{STR(LOG['STATUS']):<10}")

DEF VISUALIZE_ACCOUNT_DISTRIBUTION(DATA):
 PRINT(F"\N{COLORS.CYAN}--- GENERATING ACCOUNT DISTRIBUTION CHART ---
{COLORS.END}")
 IF DATA['ACCOUNTS'].EMPTY:
 PRINT("NO ACCOUNTS DATA TO VISUALIZE.")
 RETURN

 TRY:
 # GROUP BY ACCOUNT TYPE
 TYPE_COUNTS = DATA['ACCOUNTS']['ACCOUNTTYPE'].VALUE_COUNTS()

 PLT.FIGURE(FIGSIZE=(10, 6))
 COLORS = ['#FF9999','#66B3FF','#99FF99','#FFCC99', '#FF99CC']
 PLT.PIE(TYPE_COUNTS, LABELS=TYPE_COUNTS.INDEX, AUTOPCT='%1.1F%%',
STARTANGLE=140, COLORS=COLORS[:LEN(TYPE_COUNTS)])
 PLT.TITLE('DISTRIBUTION OF ACCOUNT TYPES', FONTSIZE=14, FONTWEIGHT='BOLD')
 PLT.AXIS('EQUAL')

 PRINT("DISPLAYING CHART WINDOW...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {E}{COLORS.END}")

DEF VISUALIZE_LOAN_STATUS(DATA):
 PRINT(F"\N{COLORS.CYAN}--- GENERATING LOAN STATUS CHART ---{COLORS.END}")
 IF DATA['LOANS'].EMPTY:
 PRINT("NO LOAN DATA TO VISUALIZE.")
 RETURN

 TRY:
 LOAN_SUMMARY = DATA['LOANS'].GROUPBY('LOANTYPE')['PRINCIPALAMOUNT'].SUM()

 PLT.FIGURE(FIGSIZE=(12, 6))
 BARS = LOAN_SUMMARY.PLOT(KIND='BAR', COLOR=['#3498DB', '#E74C3C', '#2ECC71',
'#F39C12', '#9B59B6'])
 PLT.TITLE('TOTAL LOAN AMOUNT BY TYPE', FONTSIZE=14, FONTWEIGHT='BOLD')
 PLT.XLABEL('LOAN TYPE')
 PLT.YLABEL('TOTAL PRINCIPAL AMOUNT (₹)')
 PLT.XTICKS(ROTATION=45)
 PLT.GRID(AXIS='Y', LINESTYLE='--', ALPHA=0.7)
```

```
PRINT(F"\N{'-'*100}")
 PRINT(F"{'TIMESTAMP':<20} {'ACTION':<20} {'DETAILS':<50} {'STATUS':<10}")
 PRINT(F"{'-'*100}")

 FOR _, LOG IN LOGS.ITERROWS():
 DETAILS = STR(LOG['DETAILS'])[:48] IF LOG['DETAILS'] ELSE ''
 PRINT(F"{STR(LOG['TIMESTAMP']):<20} {STR(LOG['ACTION']):<20} {DETAILS:<50}
{STR(LOG['STATUS']):<10}")

DEF VISUALIZE_ACCOUNT_DISTRIBUTION(DATA):
 PRINT(F"\N{COLORS.CYAN}--- GENERATING ACCOUNT DISTRIBUTION CHART ---
{COLORS.END}")
 IF DATA['ACCOUNTS'].EMPTY:
 PRINT("NO ACCOUNTS DATA TO VISUALIZE.")
 RETURN

 TRY:
 # GROUP BY ACCOUNT TYPE
 TYPE_COUNTS = DATA['ACCOUNTS']['ACCOUNTTYPE'].VALUE_COUNTS()

 PLT.FIGURE(FIGSIZE=(10, 6))
 COLORS = ['#FF9999','#66B3FF','#99FF99','#FFCC99', '#FF99CC']
 PLT.PIE(TYPE_COUNTS, LABELS=TYPE_COUNTS.INDEX, AUTOPCT='%1.1F%%',
STARTANGLE=140, COLORS=COLORS[:LEN(TYPE_COUNTS)])
 PLT.TITLE('DISTRIBUTION OF ACCOUNT TYPES', FONTSIZE=14, FONTWEIGHT='BOLD')
 PLT.AXIS('EQUAL')

 PRINT("DISPLAYING CHART WINDOW...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {E}{COLORS.END}")

DEF VISUALIZE_LOAN_STATUS(DATA):
 PRINT(F"\N{COLORS.CYAN}--- GENERATING LOAN STATUS CHART ---{COLORS.END}")
 IF DATA['LOANS'].EMPTY:
 PRINT("NO LOAN DATA TO VISUALIZE.")
 RETURN

 TRY:
 LOAN_SUMMARY = DATA['LOANS'].GROUPBY('LOANTYPE')['PRINCIPALAMOUNT'].SUM()

 PLT.FIGURE(FIGSIZE=(12, 6))
 BARS = LOAN_SUMMARY.PLOT(KIND='BAR', COLOR=['#3498DB', '#E74C3C', '#2ECC71',
'#F39C12', '#9B59B6'])
 PLT.TITLE('TOTAL LOAN AMOUNT BY TYPE', FONTSIZE=14, FONTWEIGHT='BOLD')
 PLT.XLABEL('LOAN TYPE')
 PLT.YLABEL('TOTAL PRINCIPAL AMOUNT (₹)')
 PLT.XTICKS(ROTATION=45)
 PLT.GRID(AXIS='Y', LINESTYLE='--', ALPHA=0.7)
```

```
 # ADD VALUE LABELS ON BARS
  FOR I, V IN ENUMERATE(LOAN_SUMMARY):
  PLT.TEXT(I, V + V*0.01, F'₹{V:,.0F}', HA='CENTER', VA='BOTTOM', FONTSIZE=9)

  PLT.TIGHT_LAYOUT()
  PRINT("DISPLAYING CHART WINDOW...")
  PLT.SHOW()
  EXCEPT EXCEPTION AS E:
  PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {E}{COLORS.END}")

 DEF VISUALIZE_MONTHLY_TRANSACTIONS(DATA):
  """MONTHLY TRANSACTION TREND CHART"""
  PRINT(F"\N{COLORS.CYAN}--- GENERATING MONTHLY TRANSACTION TREND ---
 {COLORS.END}")

  IF DATA['TRANSACTIONS'].EMPTY:
  PRINT("NO TRANSACTION DATA TO VISUALIZE.")
  RETURN

  TRY:
  # CONVERT DATE AND GROUP BY MONTH
  TXNS = DATA['TRANSACTIONS'].COPY()

  # ENSURE AMOUNT IS NUMERIC
  IF 'AMOUNT' IN TXNS.COLUMNS:
  TXNS['AMOUNT'] = PD.TO_NUMERIC(TXNS['AMOUNT'], ERRORS='COERCE')
  ELSE:
  PRINT("NO AMOUNT COLUMN FOUND IN TRANSACTIONS.")
  RETURN

  # PARSE DATES - HANDLE MULTIPLE DATE FORMATS
  IF 'DATE' IN TXNS.COLUMNS:
  TXNS['DATE'] = PD.TO_DATETIME(TXNS['DATE'], ERRORS='COERCE')
  ELSE:
  PRINT("NO DATE COLUMN FOUND IN TRANSACTIONS.")
  RETURN

  # REMOVE ROWS WITH INVALID DATES OR AMOUNTS
  TXNS = TXNS.DROPNA(SUBSET=['DATE', 'AMOUNT'])

  IF TXNS.EMPTY:
  PRINT("NO VALID TRANSACTION DATA TO VISUALIZE.")
  RETURN

  # EXTRACT MONTH AND SUM BY MONTH
  TXNS['MONTH'] = TXNS['DATE'].DT.TO_PERIOD('M')
  MONTHLY = TXNS.GROUPBY('MONTH')['AMOUNT'].SUM().SORT_INDEX()
```

```
 IF MONTHLY.EMPTY OR LEN(MONTHLY) == 0:
  PRINT("NO DATA AVAILABLE FOR MONTHLY TREND CHART.")
  RETURN

  # CONVERT PERIOD INDEX TO STRING FOR PLOTTING
  MONTHLY.INDEX = MONTHLY.INDEX.ASTYPE(STR)

  PLT.FIGURE(FIGSIZE=(12, 6))
  PLT.PLOT(RANGE(LEN(MONTHLY)), MONTHLY.VALUES, MARKER='O', LINEWIDTH=2,
MARKERSIZE=8, COLOR='#2E86AB')
  PLT.TITLE('MONTHLY TRANSACTION VOLUME', FONTSIZE=14, FONTWEIGHT='BOLD')
  PLT.XLABEL('MONTH')
  PLT.YLABEL('TOTAL AMOUNT (₹)')
  PLT.XTICKS(RANGE(LEN(MONTHLY)), MONTHLY.INDEX, ROTATION=45)
  PLT.GRID(TRUE, LINESTYLE='--', ALPHA=0.7)
  PLT.TIGHT_LAYOUT()

  PRINT("DISPLAYING CHART WINDOW...")
  PLT.SHOW()
  EXCEPT EXCEPTION AS E:
  PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {STR(E)}{COLORS.END}")

DEF VISUALIZE_CUSTOMER_GROWTH(DATA):
  """CUSTOMER GROWTH OVER TIME"""
  PRINT(F"\N{COLORS.CYAN}--- GENERATING CUSTOMER GROWTH CHART ---
{COLORS.END}")

  IF DATA['CUSTOMERS'].EMPTY:
  PRINT("NO CUSTOMER DATA TO VISUALIZE.")
  RETURN

  TRY:
  CUSTOMERS = DATA['CUSTOMERS'].COPY()
  CUSTOMERS['MONTH'] =
PD.TO_DATETIME(CUSTOMERS['REGISTRATIONDATE']).DT.TO_PERIOD('M').ASTYPE(STR)
  MONTHLY_NEW = CUSTOMERS.GROUPBY('MONTH').SIZE().CUMSUM()

  PLT.FIGURE(FIGSIZE=(12, 6))
  PLT.PLOT(MONTHLY_NEW.INDEX, MONTHLY_NEW.VALUES, MARKER='O', LINEWIDTH=2,
MARKERSIZE=8, COLOR='#2ECC71')
  PLT.FILL_BETWEEN(MONTHLY_NEW.INDEX, MONTHLY_NEW.VALUES, ALPHA=0.3,
COLOR='#2ECC71')
  PLT.TITLE('CUMULATIVE CUSTOMER GROWTH', FONTSIZE=14, FONTWEIGHT='BOLD')
  PLT.XLABEL('MONTH')
  PLT.YLABEL('TOTAL CUSTOMERS')
  PLT.GRID(TRUE, LINESTYLE='--', ALPHA=0.7)
  PLT.XTICKS(ROTATION=45)
  PLT.TIGHT_LAYOUT()
  PRINT("DISPLAYING CHART WINDOW...")
```

```
 PLT.SHOW()

EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {E}{COLORS.END}")

DEF CUSTOMER_CREDIT_SCORE(DATA):
 """CALCULATE AND DISPLAY CUSTOMER CREDIT SCORE"""
 PRINT(F"\N{COLORS.CYAN}--- CUSTOMER CREDIT SCORE ---{COLORS.END}")

 CUST_ID = INPUT("ENTER CUSTOMER ID: ").STRIP()
 CUSTOMER = DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUST_ID]

 IF CUSTOMER.EMPTY:
 PRINT("CUSTOMER NOT FOUND.")
 RETURN

 # CALCULATE FACTORS
 REG_DATE = DATETIME.STRPTIME(CUSTOMER.ILOC[0]['REGISTRATIONDATE'], '%Y-%M-
%D')
 ACCOUNT_AGE = (DATETIME.NOW() - REG_DATE).DAYS

 ACCOUNTS = DATA['ACCOUNTS'][DATA['ACCOUNTS']['CUSTOMERID'] == CUST_ID]
 TOTAL_BALANCE = ACCOUNTS['BALANCE'].SUM() IF NOT ACCOUNTS.EMPTY ELSE 0

 ACC_NUMS = ACCOUNTS['ACCOUNTNUMBER'].TOLIST()
 TOTAL_TXNS = LEN(DATA['TRANSACTIONS'][DATA['TRANSACTIONS']
['ACCOUNTNUMBER'].ISIN(ACC_NUMS)]) IF NOT DATA['TRANSACTIONS'].EMPTY ELSE 0

 # CHECK FOR DEFAULTED LOANS
 DEFAULTS = 0
 IF NOT DATA['LOANS'].EMPTY:
 CUST_LOANS = DATA['LOANS'][DATA['LOANS']['CUSTOMERID'] == CUST_ID]
 DEFAULTS = LEN(CUST_LOANS[CUST_LOANS['STATUS'] == 'DEFAULTED'])

 SCORE = GET_CREDIT_SCORE(ACCOUNT_AGE, TOTAL_TXNS, TOTAL_BALANCE, DEFAULTS)

 # DETERMINE RATING
 IF SCORE >= 750:
 RATING = "EXCELLENT"
 COLOR = COLORS.GREEN
 ELIF SCORE >= 650:
 RATING = "GOOD"
 COLOR = COLORS.BLUE
 ELIF SCORE >= 550:
 RATING = "FAIR"
 COLOR = COLORS.YELLOW
 ELSE:
 RATING = "POOR"
 COLOR = COLORS.RED
```

```
# ========================================
# SECTION 8B: ADVANCED VISUALIZATIONS
# ========================================

DEF VISUALIZE_BALANCE_DISTRIBUTION(DATA):
 """VISUALIZE BALANCE DISTRIBUTION ACROSS CUSTOMERS"""
 PRINT(F"\N{COLORS.CYAN}--- BALANCE DISTRIBUTION ANALYSIS ---{COLORS.END}")

 IF DATA['ACCOUNTS'].EMPTY:
 PRINT("NO ACCOUNT DATA TO VISUALIZE.")
 RETURN

 TRY:
 # MERGE ACCOUNTS WITH CUSTOMER NAMES
 ACC_CUST = DATA['ACCOUNTS'].MERGE(
 DATA['CUSTOMERS'][['CUSTOMERID', 'NAME']],
 ON='CUSTOMERID',
 HOW='LEFT'
 )

 # GROUP BY CUSTOMER
 CUST_BALANCES = ACC_CUST.GROUPBY('NAME')
['BALANCE'].SUM().SORT_VALUES(ASCENDING=TRUE)

 FIG, AXES = PLT.SUBPLOTS(1, 2, FIGSIZE=(14, 6))

 # HORIZONTAL BAR CHART
 COLORS = PLT.CM.VIRIDIS(RANGE(0, 256, 256//LEN(CUST_BALANCES)))
 AXES[0].BARH(CUST_BALANCES.INDEX, CUST_BALANCES.VALUES, COLOR=COLORS)
 AXES[0].SET_XLABEL('TOTAL BALANCE (₹)')
 AXES[0].SET_TITLE('CUSTOMER BALANCE COMPARISON', FONTWEIGHT='BOLD')
 AXES[0].XAXIS.SET_MAJOR_FORMATTER(PLT.FUNCFORMATTER(LAMBDA X, P: F'₹
{X/1000:.0F}K'))

 # BALANCE RANGES HISTOGRAM
 RANGES = [0, 100000, 300000, 500000, 1000000, FLOAT('INF')]
 LABELS = ['<₹1L', '₹1-3L', '₹3-5L', '₹5-10L', '>₹10L']
 BALANCE_COUNTS = PD.CUT(CUST_BALANCES, BINS=RANGES,
LABELS=LABELS).VALUE_COUNTS()
 AXES[1].BAR(BALANCE_COUNTS.INDEX, BALANCE_COUNTS.VALUES, COLOR='#3498DB')
 AXES[1].SET_XLABEL('BALANCE RANGE')
 AXES[1].SET_YLABEL('NUMBER OF CUSTOMERS')
 AXES[1].SET_TITLE('BALANCE DISTRIBUTION', FONTWEIGHT='BOLD')

 PLT.TIGHT_LAYOUT()
 PRINT("DISPLAYING CHART WINDOW...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {STR(E)}{COLORS.END}")
```

```
DEF VISUALIZE_TRANSACTION_TYPES(DATA):
 """VISUALIZE TRANSACTION TYPES BREAKDOWN"""
 PRINT(F"\N{COLORS.CYAN}--- TRANSACTION TYPES ANALYSIS ---{COLORS.END}")

 IF DATA['TRANSACTIONS'].EMPTY:
 PRINT("NO TRANSACTION DATA TO VISUALIZE.")
 RETURN

 TRY:
 # GROUP BY TRANSACTION TYPE
 TXN_TYPES = DATA['TRANSACTIONS']['TRANSACTIONTYPE'].VALUE_COUNTS()

 FIG, AXES = PLT.SUBPLOTS(1, 2, FIGSIZE=(14, 6))

 # PIE CHART OF TRANSACTION COUNT
 COLORS = PLT.CM.SET3(RANGE(LEN(TXN_TYPES)))
 AXES[0].PIE(TXN_TYPES.VALUES, LABELS=TXN_TYPES.INDEX, AUTOPCT='%1.1F%%',
 COLORS=COLORS, STARTANGLE=90)
 AXES[0].SET_TITLE('TRANSACTION TYPES BY COUNT', FONTWEIGHT='BOLD')

 # BAR CHART OF TOTAL AMOUNTS BY TYPE
 TXN_AMOUNTS = DATA['TRANSACTIONS'].GROUPBY('TRANSACTIONTYPE')
['AMOUNT'].SUM().SORT_VALUES(ASCENDING=FALSE)
 AXES[1].BAR(RANGE(LEN(TXN_AMOUNTS)), TXN_AMOUNTS.VALUES, COLOR='#2ECC71')
 AXES[1].SET_XTICKS(RANGE(LEN(TXN_AMOUNTS)))
 AXES[1].SET_XTICKLABELS(TXN_AMOUNTS.INDEX, ROTATION=45, HA='RIGHT')
 AXES[1].SET_YLABEL('TOTAL AMOUNT (₹)')
 AXES[1].SET_TITLE('TRANSACTION VOLUME BY TYPE', FONTWEIGHT='BOLD')
 AXES[1].YAXIS.SET_MAJOR_FORMATTER(PLT.FUNCFORMATTER(LAMBDA X, P: F'₹
{X/100000:.1F}L'))

 PLT.TIGHT_LAYOUT()
 PRINT("DISPLAYING CHART WINDOW...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {STR(E)}{COLORS.END}")

DEF VISUALIZE_LOAN_EMI_ANALYSIS(DATA):
 """VISUALIZE LOAN EMI AND OUTSTANDING ANALYSIS"""
 PRINT(F"\N{COLORS.CYAN}--- LOAN EMI ANALYSIS ---{COLORS.END}")

 IF DATA['LOANS'].EMPTY:
 PRINT("NO LOAN DATA TO VISUALIZE.")
 RETURN

 TRY:
 LOANS = DATA['LOANS'].COPY()

 FIG, AXES = PLT.SUBPLOTS(2, 2, FIGSIZE=(14, 10))
```

```
# 1. LOAN TYPE DISTRIBUTION (PIE)
 LOAN_COUNTS = LOANS['LOANTYPE'].VALUE_COUNTS()
 AXES[0, 0].PIE(LOAN_COUNTS.VALUES, LABELS=LOAN_COUNTS.INDEX,
AUTOPCT='%1.1F%%',
 COLORS=PLT.CM.PASTEL1(RANGE(LEN(LOAN_COUNTS))))
 AXES[0, 0].SET_TITLE('LOAN TYPE DISTRIBUTION', FONTWEIGHT='BOLD')

 # 2. PRINCIPAL VS OUTSTANDING (GROUPED BAR)
 LOAN_NAMES = LOANS['LOANID'].TOLIST()
 X = RANGE(LEN(LOAN_NAMES))
 WIDTH = 0.35
 PRINCIPAL = LOANS['PRINCIPALAMOUNT'].ASTYPE(FLOAT).TOLIST()
 OUTSTANDING = LOANS['OUTSTANDINGAMOUNT'].ASTYPE(FLOAT).TOLIST()

 AXES[0, 1].BAR([I - WIDTH/2 FOR I IN X], PRINCIPAL, WIDTH, LABEL='PRINCIPAL',
COLOR='#3498DB')
 AXES[0, 1].BAR([I + WIDTH/2 FOR I IN X], OUTSTANDING, WIDTH, LABEL='OUTSTANDING',
COLOR='#E74C3C')
 AXES[0, 1].SET_XTICKS(X)
 AXES[0, 1].SET_XTICKLABELS(LOAN_NAMES, ROTATION=45)
 AXES[0, 1].SET_YLABEL('AMOUNT (₹)')
 AXES[0, 1].SET_TITLE('PRINCIPAL VS OUTSTANDING', FONTWEIGHT='BOLD')
 AXES[0, 1].LEGEND()
 AXES[0, 1].YAXIS.SET_MAJOR_FORMATTER(PLT.FUNCFORMATTER(LAMBDA X, P: F'₹
{X/100000:.1F}L'))

 # 3. EMI BY LOAN
 EMI_VALUES = LOANS['EMI'].ASTYPE(FLOAT).TOLIST()
 AXES[1, 0].BAR(LOAN_NAMES, EMI_VALUES, COLOR='#9B59B6')
 AXES[1, 0].SET_YLABEL('MONTHLY EMI (₹)')
 AXES[1, 0].SET_TITLE('MONTHLY EMI BY LOAN', FONTWEIGHT='BOLD')
 AXES[1, 0].TICK_PARAMS(AXIS='X', ROTATION=45)

 # 4. REPAYMENT PROGRESS
 PROGRESS = [(P - O) / P * 100 IF P > 0 ELSE 0
 FOR P, O IN ZIP(PRINCIPAL, OUTSTANDING)]
 COLORS = ['#27AE60' IF P > 50 ELSE '#F39C12' IF P > 25 ELSE '#E74C3C' FOR P IN
PROGRESS]
 AXES[1, 1].BARH(LOAN_NAMES, PROGRESS, COLOR=COLORS)
 AXES[1, 1].SET_XLABEL('REPAYMENT PROGRESS (%)')
 AXES[1, 1].SET_TITLE('LOAN REPAYMENT PROGRESS', FONTWEIGHT='BOLD')
 AXES[1, 1].SET_XLIM(0, 100)

 PLT.TIGHT_LAYOUT()
 PRINT("DISPLAYING CHART WINDOW...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING CHART: {STR(E)}{COLORS.END}")
```

```
DEF VISUALIZE_DAILY_ACTIVITY(DATA):
 """VISUALIZE DAILY TRANSACTION ACTIVITY HEATMAP STYLE"""
 PRINT(F"\N{COLORS.CYAN}--- DAILY ACTIVITY ANALYSIS ---{COLORS.END}")

 IF DATA['TRANSACTIONS'].EMPTY:
 PRINT("NO TRANSACTION DATA TO VISUALIZE.")
 RETURN

 TRY:
 TXNS = DATA['TRANSACTIONS'].COPY()
 TXNS['DATE'] = PD.TO_DATETIME(TXNS['DATE'], ERRORS='COERCE')
 TXNS = TXNS.DROPNA(SUBSET=['DATE'])

 IF TXNS.EMPTY:
 PRINT("NO VALID DATE DATA TO VISUALIZE.")
 RETURN

 # GROUP BY DATE
 DAILY = TXNS.GROUPBY(TXNS['DATE'].DT.DATE).AGG({
 'AMOUNT': ['SUM', 'COUNT']
 }).RESET_INDEX()
 DAILY.COLUMNS = ['DATE', 'TOTALAMOUNT', 'TXNCOUNT']
 DAILY = DAILY.SORT_VALUES('DATE')

 FIG, AXES = PLT.SUBPLOTS(2, 1, FIGSIZE=(14, 8))

 # TRANSACTION COUNT BY DAY
 AXES[0].FILL_BETWEEN(RANGE(LEN(DAILY)), DAILY['TXNCOUNT'], ALPHA=0.5,
COLOR='#3498DB')
 AXES[0].PLOT(RANGE(LEN(DAILY)), DAILY['TXNCOUNT'], COLOR='#2980B9', LINEWIDTH=2)
 AXES[0].SET_YLABEL('NUMBER OF TRANSACTIONS')
 AXES[0].SET_TITLE('DAILY TRANSACTION COUNT', FONTWEIGHT='BOLD')
 AXES[0].SET_XTICKS(RANGE(0, LEN(DAILY), MAX(1, LEN(DAILY)//10)))
 AXES[0].SET_XTICKLABELS([STR(D)[:10] FOR D IN DAILY['DATE'].ILOC[::MAX(1,
LEN(DAILY)//10)]], ROTATION=45)

 # TRANSACTION VOLUME BY DAY
 AXES[1].BAR(RANGE(LEN(DAILY)), DAILY['TOTALAMOUNT'], COLOR='#27AE60', ALPHA=0.7)
 AXES[1].SET_YLABEL('TOTAL VOLUME (₹)')
 AXES[1].SET_XLABEL('DATE')
 AXES[1].SET_TITLE('DAILY TRANSACTION VOLUME', FONTWEIGHT='BOLD')
 AXES[1].SET_XTICKS(RANGE(0, LEN(DAILY), MAX(1, LEN(DAILY)//10)))
 AXES[1].SET_XTICKLABELS([STR(D)[:10] FOR D IN DAILY['DATE'].ILOC[::MAX(1,
LEN(DAILY)//10)]], ROTATION=45)
 AXES[1].YAXIS.SET_MAJOR_FORMATTER(PLT.FUNCFORMATTER(LAMBDA X, P: F'₹
{X/1000:.0F}K'))
```

```
DEF VISUALIZE_COMPREHENSIVE_DASHBOARD(DATA):
 """GENERATE A COMPREHENSIVE VISUAL DASHBOARD"""
 PRINT(F"\N{COLORS.CYAN}--- COMPREHENSIVE ANALYTICS DASHBOARD ---
{COLORS.END}")

 TRY:
 FIG = PLT.FIGURE(FIGSIZE=(16, 12))

 # 1. ACCOUNT TYPE DISTRIBUTION (TOP LEFT)
 AX1 = FIG.ADD_SUBPLOT(2, 3, 1)
 IF NOT DATA['ACCOUNTS'].EMPTY:
 ACC_TYPES = DATA['ACCOUNTS']['ACCOUNTTYPE'].VALUE_COUNTS()
 AX1.PIE(ACC_TYPES.VALUES, LABELS=ACC_TYPES.INDEX, AUTOPCT='%1.1F%%',
 COLORS=['#3498DB', '#E74C3C', '#2ECC71', '#F39C12'])
 AX1.SET_TITLE('ACCOUNT TYPES', FONTWEIGHT='BOLD')

 # 2. CUSTOMER REGISTRATION TIMELINE (TOP MIDDLE)
 AX2 = FIG.ADD_SUBPLOT(2, 3, 2)
 IF NOT DATA['CUSTOMERS'].EMPTY:
 CUST = DATA['CUSTOMERS'].COPY()
 CUST['REGDATE'] = PD.TO_DATETIME(CUST['REGISTRATIONDATE'], ERRORS='COERCE')
 CUST = CUST.DROPNA(SUBSET=['REGDATE']).SORT_VALUES('REGDATE')
 CUST['CUMCOUNT'] = RANGE(1, LEN(CUST) + 1)
 AX2.PLOT(CUST['REGDATE'], CUST['CUMCOUNT'], MARKER='O', COLOR='#9B59B6')
 AX2.SET_XLABEL('DATE')
 AX2.SET_YLABEL('TOTAL CUSTOMERS')
 AX2.TICK_PARAMS(AXIS='X', ROTATION=45)
 AX2.SET_TITLE('CUSTOMER GROWTH', FONTWEIGHT='BOLD')

 # 3. BALANCE SUMMARY (TOP RIGHT)
 AX3 = FIG.ADD_SUBPLOT(2, 3, 3)
 TOTAL_BALANCE = DATA['ACCOUNTS']['BALANCE'].SUM() IF NOT
DATA['ACCOUNTS'].EMPTY ELSE 0
 TOTAL_LOANS = DATA['LOANS']['OUTSTANDINGAMOUNT'].SUM() IF NOT
DATA['LOANS'].EMPTY ELSE 0
 NET_WORTH = TOTAL_BALANCE - TOTAL_LOANS
 CATEGORIES = ['TOTAL DEPOSITS', 'LOAN OUTSTANDING', 'NET POSITION']
 VALUES = [TOTAL_BALANCE, TOTAL_LOANS, NET_WORTH]
 COLORS = ['#27AE60', '#E74C3C', '#3498DB']
 BARS = AX3.BAR(CATEGORIES, VALUES, COLOR=COLORS)
 AX3.SET_YLABEL('AMOUNT (₹)')
 AX3.YAXIS.SET_MAJOR_FORMATTER(PLT.FUNCFORMATTER(LAMBDA X, P: F'₹
{X/100000:.1F}L'))
 AX3.SET_TITLE('FINANCIAL SUMMARY', FONTWEIGHT='BOLD')

 # 4. TRANSACTION TREND (BOTTOM LEFT)
 AX4 = FIG.ADD_SUBPLOT(2, 3, 4)
 IF NOT DATA['TRANSACTIONS'].EMPTY:
 TXNS = DATA['TRANSACTIONS'].COPY()
```

```
XNS = TXNS.DROPNA(SUBSET=['DATE', 'AMOUNT'])
 IF NOT TXNS.EMPTY:
 TXNS['MONTH'] = TXNS['DATE'].DT.TO_PERIOD('M')
 MONTHLY = TXNS.GROUPBY('MONTH')['AMOUNT'].SUM().SORT_INDEX()
 MONTHLY.INDEX = MONTHLY.INDEX.ASTYPE(STR)
 AX4.BAR(RANGE(LEN(MONTHLY)), MONTHLY.VALUES, COLOR='#1ABC9C')
 AX4.SET_XTICKS(RANGE(LEN(MONTHLY)))
 AX4.SET_XTICKLABELS(MONTHLY.INDEX, ROTATION=45)
 AX4.SET_XLABEL('MONTH')
 AX4.SET_YLABEL('VOLUME (₹)')
 AX4.SET_TITLE('MONTHLY TRANSACTION VOLUME', FONTWEIGHT='BOLD')

 # 5. LOAN STATUS (BOTTOM MIDDLE)
 AX5 = FIG.ADD_SUBPLOT(2, 3, 5)
 IF NOT DATA['LOANS'].EMPTY:
 LOAN_STATUS = DATA['LOANS']['STATUS'].VALUE_COUNTS()
 COLORS_STATUS = {'ACTIVE': '#27AE60', 'CLOSED': '#95A5A6', 'DEFAULTED': '#E74C3C'}
 AX5.PIE(LOAN_STATUS.VALUES, LABELS=LOAN_STATUS.INDEX, AUTOPCT='%1.1F%%',
 COLORS=[COLORS_STATUS.GET(S, '#3498DB') FOR S IN LOAN_STATUS.INDEX])
 AX5.SET_TITLE('LOAN STATUS', FONTWEIGHT='BOLD')

 # 6. TOP CUSTOMERS BY BALANCE (BOTTOM RIGHT)
 AX6 = FIG.ADD_SUBPLOT(2, 3, 6)
 IF NOT DATA['ACCOUNTS'].EMPTY:
 CUST_BAL = DATA['ACCOUNTS'].GROUPBY('CUSTOMERID')['BALANCE'].SUM().NLARGEST(5)
 CUST_NAMES = []
 FOR CID IN CUST_BAL.INDEX:
 CUST = DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CID]
 NAME = CUST.ILOC[0]['NAME'].SPLIT()[0] IF NOT CUST.EMPTY ELSE CID
 CUST_NAMES.APPEND(NAME)
 AX6.BARH(CUST_NAMES, CUST_BAL.VALUES, COLOR='#F39C12')
 AX6.SET_XLABEL('BALANCE (₹)')
 AX6.SET_TITLE('TOP 5 CUSTOMERS', FONTWEIGHT='BOLD')

 PLT.SUPTITLE('COREBANK ANALYTICS DASHBOARD', FONTSIZE=16, FONTWEIGHT='BOLD',
Y=1.02)
 PLT.TIGHT_LAYOUT()
 PRINT("DISPLAYING COMPREHENSIVE DASHBOARD...")
 PLT.SHOW()
 EXCEPT EXCEPTION AS E:
 PRINT(F"{COLORS.RED}ERROR GENERATING DASHBOARD: {STR(E)}{COLORS.END}")

DEF EXPORT_DATABASE_SUMMARY(DATA):
 """EXPORT A CLEAN SUMMARY OF THE ENTIRE DATABASE"""
 PRINT(F"\N{COLORS.CYAN}--- DATABASE SUMMARY EXPORT ---{COLORS.END}")

 PRINT(F"\N{COLORS.BOLD} ╔{'═'*60}╗ ")
 PRINT(F" ║ {'COREBANK DATABASE SUMMARY':^60} ║ ")
 PRINT(F" ╚{'═'*60}╝ {COLORS.END}")
```

```
PRINT(F"\N{COLORS.YELLOW}=== CUSTOMERS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['CUSTOMERS'])}")
 IF NOT DATA['CUSTOMERS'].EMPTY:
 PRINT(DATA['CUSTOMERS'][['CUSTOMERID', 'NAME', 'CITY', 'PHONE',
'STATUS']].TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== ACCOUNTS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['ACCOUNTS'])}")
 IF NOT DATA['ACCOUNTS'].EMPTY:
 ACC_DISPLAY = DATA['ACCOUNTS'][['ACCOUNTNUMBER', 'CUSTOMERID', 'ACCOUNTTYPE',
'BALANCE', 'STATUS']].COPY()
 ACC_DISPLAY['BALANCE'] = ACC_DISPLAY['BALANCE'].APPLY(LAMBDA X: F"₹{X:,.2F}")
 PRINT(ACC_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== TRANSACTIONS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['TRANSACTIONS'])}")
 IF NOT DATA['TRANSACTIONS'].EMPTY:
 TXN_DISPLAY = DATA['TRANSACTIONS'][['TRANSACTIONID', 'ACCOUNTNUMBER',
'TRANSACTIONTYPE', 'AMOUNT', 'DATE']].HEAD(20).COPY()
 TXN_DISPLAY['AMOUNT'] = TXN_DISPLAY['AMOUNT'].APPLY(LAMBDA X: F"₹
{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A')
 PRINT(TXN_DISPLAY.TO_STRING(INDEX=FALSE))
 IF LEN(DATA['TRANSACTIONS']) > 20:
 PRINT(F" ... AND {LEN(DATA['TRANSACTIONS']) - 20} MORE RECORDS")

 PRINT(F"\N{COLORS.YELLOW}=== LOANS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['LOANS'])}")
 IF NOT DATA['LOANS'].EMPTY:
 LOAN_DISPLAY = DATA['LOANS'][['LOANID', 'CUSTOMERID', 'LOANTYPE',
'PRINCIPALAMOUNT', 'EMI', 'OUTSTANDINGAMOUNT', 'STATUS']].COPY()
 FOR COL IN ['PRINCIPALAMOUNT', 'EMI', 'OUTSTANDINGAMOUNT']:
 LOAN_DISPLAY[COL] = LOAN_DISPLAY[COL].APPLY(LAMBDA X: F"₹{FLOAT(X):,.2F}" IF
PD.NOTNA(X) ELSE 'N/A')
 PRINT(LOAN_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== CARDS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['CARDS'])}")
 IF NOT DATA['CARDS'].EMPTY:
 CARDS_DISPLAY = DATA['CARDS'].COPY()
 # SAFELY GET CARD NUMBER COLUMN
 CARD_NUM_COL = 'CARDNUMBER' IF 'CARDNUMBER' IN CARDS_DISPLAY.COLUMNS ELSE
'CARDID'
 IF CARD_NUM_COL IN CARDS_DISPLAY.COLUMNS:
 CARDS_DISPLAY['MASKEDCARD'] = CARDS_DISPLAY[CARD_NUM_COL].APPLY(LAMBDA X:
F"XXXX-{STR(X)[-4:]}" IF PD.NOTNA(X) ELSE 'N/A')
 ELSE:
 CARDS_DISPLAY['MASKEDCARD'] = 'N/A'
 PRINT(CARDS_DISPLAY[['MASKEDCARD', 'CUSTOMERID', 'CARDTYPE',
'STATUS']].TO_STRING(INDEX=FALSE))
```

```
PRINT(F"\N{COLORS.YELLOW}=== CHEQUES TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['CHEQUES'])}")
 IF NOT DATA['CHEQUES'].EMPTY:
 CHEQUE_DISPLAY = DATA['CHEQUES'][['CHEQUENUMBER', 'ACCOUNTNUMBER',
'ISSUEDTO', 'AMOUNT', 'STATUS']].COPY()
 CHEQUE_DISPLAY['AMOUNT'] = CHEQUE_DISPLAY['AMOUNT'].APPLY(LAMBDA X: F"₹
{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A')
 PRINT(CHEQUE_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== TRANSFERS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['TRANSFERS'])}")
 IF NOT DATA['TRANSFERS'].EMPTY:
 TRANSFER_DISPLAY = DATA['TRANSFERS'][['TRANSFERID', 'FROMACCOUNT', 'TOACCOUNT',
'AMOUNT', 'DATE', 'STATUS']].COPY()
 TRANSFER_DISPLAY['AMOUNT'] = TRANSFER_DISPLAY['AMOUNT'].APPLY(LAMBDA X: F"₹
{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A')
 PRINT(TRANSFER_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== AUDIT LOGS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['AUDIT'])}")
 IF NOT DATA['AUDIT'].EMPTY:
 PRINT(DATA['AUDIT'][['LOGID', 'ACTION', 'TIMESTAMP',
'STATUS']].TAIL(10).TO_STRING(INDEX=FALSE))
 PRINT(F" (SHOWING LAST 10 ENTRIES)")

 PRINT(F"\N{COLORS.GREEN}{'═'*60}")
 PRINT(F"DATABASE STATISTICS")
 PRINT(F"{'═'*60}{COLORS.END}")
 PRINT(F" TOTAL CUSTOMERS: {LEN(DATA['CUSTOMERS'])}")
 PRINT(F" TOTAL ACCOUNTS: {LEN(DATA['ACCOUNTS'])}")
 PRINT(F" TOTAL TRANSACTIONS: {LEN(DATA['TRANSACTIONS'])}")
 PRINT(F" TOTAL LOANS: {LEN(DATA['LOANS'])}")
 PRINT(F" TOTAL CARDS: {LEN(DATA['CARDS'])}")
 PRINT(F" TOTAL CHEQUES: {LEN(DATA['CHEQUES'])}")
 PRINT(F" TOTAL TRANSFERS: {LEN(DATA['TRANSFERS'])}")
 PRINT(F" TOTAL AUDIT LOGS: {LEN(DATA['AUDIT'])}")
 TOTAL_RECORDS = SUM(LEN(DATA[T]) FOR T IN DATA)
 PRINT(F"\N {COLORS.BOLD}GRAND TOTAL: {TOTAL_RECORDS} RECORDS{COLORS.END}")

DEF GENERATE_REPORTS(DATA):
 WHILE TRUE:
 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}=== REPORTS & ANALYTICS ===
{COLORS.END}")
 PRINT(F"\N{COLORS.YELLOW}TEXT REPORTS:{COLORS.END}")
 PRINT("1. TRANSACTION HISTORY")
 PRINT("2. BANK FINANCIAL SUMMARY")
 PRINT("3. CUSTOMER BALANCES REPORT")
 PRINT("4. DAILY TRANSACTIONS SUMMARY")
 PRINT("5. LOAN PORTFOLIO ANALYSIS")
```

```
PRINT(F"\N{COLORS.YELLOW}=== CHEQUES TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['CHEQUES'])}")
 IF NOT DATA['CHEQUES'].EMPTY:
 CHEQUE_DISPLAY = DATA['CHEQUES'][['CHEQUENUMBER', 'ACCOUNTNUMBER',
'ISSUEDTO', 'AMOUNT', 'STATUS']].COPY()
 CHEQUE_DISPLAY['AMOUNT'] = CHEQUE_DISPLAY['AMOUNT'].APPLY(LAMBDA X: F"₹
{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A')
 PRINT(CHEQUE_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== TRANSFERS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['TRANSFERS'])}")
 IF NOT DATA['TRANSFERS'].EMPTY:
 TRANSFER_DISPLAY = DATA['TRANSFERS'][['TRANSFERID', 'FROMACCOUNT', 'TOACCOUNT',
'AMOUNT', 'DATE', 'STATUS']].COPY()
 TRANSFER_DISPLAY['AMOUNT'] = TRANSFER_DISPLAY['AMOUNT'].APPLY(LAMBDA X: F"₹
{FLOAT(X):,.2F}" IF PD.NOTNA(X) ELSE 'N/A')
 PRINT(TRANSFER_DISPLAY.TO_STRING(INDEX=FALSE))

 PRINT(F"\N{COLORS.YELLOW}=== AUDIT LOGS TABLE ==={COLORS.END}")
 PRINT(F"TOTAL RECORDS: {LEN(DATA['AUDIT'])}")
 IF NOT DATA['AUDIT'].EMPTY:
 PRINT(DATA['AUDIT'][['LOGID', 'ACTION', 'TIMESTAMP',
'STATUS']].TAIL(10).TO_STRING(INDEX=FALSE))
 PRINT(F" (SHOWING LAST 10 ENTRIES)")

 PRINT(F"\N{COLORS.GREEN}{'═'*60}")
 PRINT(F"DATABASE STATISTICS")
 PRINT(F"{'═'*60}{COLORS.END}")
 PRINT(F" TOTAL CUSTOMERS: {LEN(DATA['CUSTOMERS'])}")
 PRINT(F" TOTAL ACCOUNTS: {LEN(DATA['ACCOUNTS'])}")
 PRINT(F" TOTAL TRANSACTIONS: {LEN(DATA['TRANSACTIONS'])}")
 PRINT(F" TOTAL LOANS: {LEN(DATA['LOANS'])}")
 PRINT(F" TOTAL CARDS: {LEN(DATA['CARDS'])}")
 PRINT(F" TOTAL CHEQUES: {LEN(DATA['CHEQUES'])}")
 PRINT(F" TOTAL TRANSFERS: {LEN(DATA['TRANSFERS'])}")
 PRINT(F" TOTAL AUDIT LOGS: {LEN(DATA['AUDIT'])}")
 TOTAL_RECORDS = SUM(LEN(DATA[T]) FOR T IN DATA)
 PRINT(F"\N {COLORS.BOLD}GRAND TOTAL: {TOTAL_RECORDS} RECORDS{COLORS.END}")

DEF GENERATE_REPORTS(DATA):
 WHILE TRUE:
 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}=== REPORTS & ANALYTICS ===
{COLORS.END}")
 PRINT(F"\N{COLORS.YELLOW}TEXT REPORTS:{COLORS.END}")
 PRINT("1. TRANSACTION HISTORY")
 PRINT("2. BANK FINANCIAL SUMMARY")
 PRINT("3. CUSTOMER BALANCES REPORT")
 PRINT("4. DAILY TRANSACTIONS SUMMARY")
 PRINT("5. LOAN PORTFOLIO ANALYSIS")
```

```
PRINT("5. LOAN PORTFOLIO ANALYSIS")
 PRINT("6. VIEW AUDIT TRAIL")
 PRINT("7. CUSTOMER CREDIT SCORE")
 PRINT("8. DATABASE SUMMARY (ALL TABLES)")
 PRINT(F"\N{COLORS.YELLOW}VISUAL CHARTS (MATPLOTLIB):{COLORS.END}")
 PRINT("9. ACCOUNT DISTRIBUTION (PIE CHART)")
 PRINT("10. LOAN PORTFOLIO (BAR CHART)")
 PRINT("11. MONTHLY TRANSACTION TREND")
 PRINT("12. CUSTOMER GROWTH CHART")
 PRINT("13. BALANCE DISTRIBUTION ANALYSIS")
 PRINT("14. TRANSACTION TYPES BREAKDOWN")
 PRINT("15. LOAN EMI ANALYSIS (4 CHARTS)")
 PRINT("16. DAILY ACTIVITY TIMELINE")
 PRINT("17. COMPREHENSIVE DASHBOARD (6 CHARTS)")
 PRINT(F"\N18. BACK TO MAIN MENU")

 CHOICE = INPUT("\NSELECT REPORT: ").STRIP()

 IF CHOICE == '1': REPORT_TRANSACTION_HISTORY(DATA)
 ELIF CHOICE == '2': REPORT_BANK_SUMMARY(DATA)
 ELIF CHOICE == '3': REPORT_CUSTOMER_BALANCES(DATA)
 ELIF CHOICE == '4': REPORT_DAILY_TRANSACTIONS(DATA)
 ELIF CHOICE == '5': REPORT_LOAN_PORTFOLIO(DATA)
 ELIF CHOICE == '6': VIEW_AUDIT_TRAIL(DATA)
 ELIF CHOICE == '7': CUSTOMER_CREDIT_SCORE(DATA)
 ELIF CHOICE == '8': EXPORT_DATABASE_SUMMARY(DATA)
 ELIF CHOICE == '9': VISUALIZE_ACCOUNT_DISTRIBUTION(DATA)
 ELIF CHOICE == '10': VISUALIZE_LOAN_STATUS(DATA)
 ELIF CHOICE == '11': VISUALIZE_MONTHLY_TRANSACTIONS(DATA)
 ELIF CHOICE == '12': VISUALIZE_CUSTOMER_GROWTH(DATA)
 ELIF CHOICE == '13': VISUALIZE_BALANCE_DISTRIBUTION(DATA)
 ELIF CHOICE == '14': VISUALIZE_TRANSACTION_TYPES(DATA)
 ELIF CHOICE == '15': VISUALIZE_LOAN_EMI_ANALYSIS(DATA)
 ELIF CHOICE == '16': VISUALIZE_DAILY_ACTIVITY(DATA)
 ELIF CHOICE == '17': VISUALIZE_COMPREHENSIVE_DASHBOARD(DATA)
 ELIF CHOICE == '18': BREAK
 ELSE: PRINT("INVALID OPTION.")
```

```
# ========================================
# SECTION 9: MENUS & MAIN LOOP
# ========================================

DEF DASHBOARD(DATA):
 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}
 ╔═══════════════════════════════════════════════════════════════════════════════
═══╗ ")
 PRINT(F" ║  COREBANK MANAGEMENT SYSTEM  ║ ")

PRINT(F" ╚═══════════════════════════════════════════════════════════════════════════════
═══╝ {COLORS.END}")
 PRINT(F"\N{COLORS.CYAN}QUICK STATS:{COLORS.END}")
 PRINT(F" CUSTOMERS: {LEN(DATA['CUSTOMERS']):<8} ACCOUNTS:
{LEN(DATA['ACCOUNTS']):<8} CARDS: {LEN(DATA['CARDS'])}")
 TOTAL_BALANCE = DATA['ACCOUNTS']['BALANCE'].SUM() IF NOT
DATA['ACCOUNTS'].EMPTY ELSE 0
 TOTAL_LOANS = DATA['LOANS']['OUTSTANDINGAMOUNT'].SUM() IF NOT
DATA['LOANS'].EMPTY ELSE 0
 PRINT(F" BALANCE: ₹{TOTAL_BALANCE:,.2F}")
 PRINT(F" LOANS: {LEN(DATA['LOANS']):<8} OUTSTANDING: ₹{TOTAL_LOANS:,.2F}")
 PRINT(F" TIME: {DATETIME.NOW().STRFTIME('%Y-%M-%D %H:%M:%S')}")
 PRINT(F"{COLORS.CYAN}{'—'*44}{COLORS.END}")

DEF LOAN_MANAGEMENT_MENU(DATA):
 """LOAN MANAGEMENT SUB-MENU"""
 WHILE TRUE:
 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}=== LOAN MANAGEMENT ==={COLORS.END}")
 PRINT("1. APPLY FOR LOAN")
 PRINT("2. PAY LOAN EMI")
 PRINT("3. VIEW LOAN DETAILS")
 PRINT("4. VIEW ALL LOANS")
 PRINT("5. BACK TO MAIN MENU")

 CHOICE = INPUT("\NSELECT: ").STRIP()
 IF CHOICE == '1': DATA = APPLY_LOAN(DATA)
 ELIF CHOICE == '2': DATA = PAY_LOAN_EMI(DATA)
 ELIF CHOICE == '3': VIEW_LOAN_DETAILS(DATA)
 ELIF CHOICE == '4':
 IF DATA['LOANS'].EMPTY:
 PRINT("NO LOANS FOUND.")
 ELSE:
 PRINT(DATA['LOANS'][['LOANID', 'CUSTOMERID', 'LOANTYPE', 'PRINCIPALAMOUNT', 'EMI',
'OUTSTANDINGAMOUNT', 'STATUS']].TO_STRING(INDEX=FALSE))
 ELIF CHOICE == '5': BREAK
 ELSE: PRINT("INVALID OPTION.")

 SAVE_DATA(DATA)
 RETURN DATA
```

```
# ========================================
# SECTION 9A: ADVANCED FEATURES
# ========================================

DEF GENERATE_ACCOUNT_STATEMENT(DATA):
 """GENERATE DETAILED ACCOUNT STATEMENT"""
 PRINT(F"\N{COLORS.CYAN}--- ACCOUNT STATEMENT ---{COLORS.END}")

 ACC_NUM = INPUT("ACCOUNT NUMBER: ").STRIP()
 ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]

 IF ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}ACCOUNT NOT FOUND{COLORS.END}")
 RETURN

 ACC = ACCOUNT.ILOC[0]
 CUSTOMER_ID = ACC['CUSTOMERID']
 CUSTOMER = DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUSTOMER_ID]

 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}{'='*60}")
 PRINT(F"{'COREBANK ACCOUNT STATEMENT':^60}")
 PRINT(F"{'='*60}{COLORS.END}\N")

 PRINT(F"ACCOUNT HOLDER: {CUSTOMER.ILOC[0]['NAME']}")
 PRINT(F"ACCOUNT NUMBER: {ACC_NUM}")
 PRINT(F"ACCOUNT TYPE: {ACC['ACCOUNTTYPE']}")
 PRINT(F"STATEMENT DATE: {GET_DATE()}")
 PRINT(F"\N{COLORS.CYAN}ACCOUNT SUMMARY:{COLORS.END}")
 PRINT(F" OPENING BALANCE: ₹{ACC['BALANCE']:,.2F}")
 PRINT(F" CURRENT BALANCE: ₹{ACC['BALANCE']:,.2F}")
 PRINT(F" INTEREST RATE: {ACC['INTERESTRATE']:.2F}%")
 PRINT(F" MINIMUM BALANCE: ₹{ACC['MINBALANCE']:,.2F}")
 PRINT(F" STATUS: {ACC['STATUS']}")

 # GET LAST 10 TRANSACTIONS
 TRANS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']['ACCOUNTNUMBER'] ==
ACC_NUM]
 IF NOT TRANS.EMPTY:
 TRANS_SORTED = TRANS.SORT_VALUES('DATE', ASCENDING=FALSE).HEAD(10)
 PRINT(F"\N{COLORS.CYAN}RECENT TRANSACTIONS (LAST 10):{COLORS.END}")
 PRINT(F"{'-'*60}")
 FOR IDX, T IN TRANS_SORTED.ITERROWS():
 TXN_TYPE = STR(T.GET('TRANSACTIONTYPE', 'N/A') OR 'N/A')
 SYMBOL = "+" IF TXN_TYPE IN ['DEPOSIT', 'CREDIT', 'CHEQUE CREDIT', 'TRANSFER CREDIT']
ELSE "-"
 AMOUNT = FLOAT(T.GET('AMOUNT', 0) OR 0)
 # HANDLE BOTH COLUMN NAMING CONVENTIONS
 BALANCE = T.GET('BALANCE_AFTER') OR T.GET('BALANCEAFTER') OR 0
 BALANCE = FLOAT(BALANCE) IF PD.NOTNA(BALANCE) ELSE 0
```

```
DATE_STR = STR(T.GET('DATE', 'N/A') OR 'N/A')[:10]
 PRINT(F"{DATE_STR:12} {TXN_TYPE[:10]:10} {SYMBOL}₹{ABS(AMOUNT):>10,.2F} BALANCE: ₹
{BALANCE:>12,.2F}")
 PRINT(F"{'-'*60}\N")
 ELSE:
 PRINT(F"\N{COLORS.YELLOW}NO TRANSACTIONS FOUND{COLORS.END}\N")

DEF CALCULATE_ACCOUNT_INTEREST(DATA):
 """CALCULATE INTEREST ACCRUED ON ACCOUNTS"""
 PRINT(F"\N{COLORS.CYAN}--- INTEREST CALCULATOR ---{COLORS.END}")

 ACC_NUM = INPUT("ACCOUNT NUMBER: ").STRIP()
 ACCOUNT = DATA['ACCOUNTS'][DATA['ACCOUNTS']['ACCOUNTNUMBER'] == ACC_NUM]

 IF ACCOUNT.EMPTY:
 PRINT(F"{COLORS.RED}ACCOUNT NOT FOUND{COLORS.END}")
 RETURN

 ACC = ACCOUNT.ILOC[0]
 MONTHS = INT(INPUT("NUMBER OF MONTHS TO CALCULATE: "))

 BALANCE = ACC['BALANCE']
 RATE = ACC['INTERESTRATE']

 # SIMPLE INTEREST
 SIMPLE_INTEREST = (BALANCE * RATE * MONTHS) / (100 * 12)

 # COMPOUND INTEREST (QUARTERLY)
 COMPOUND_INTEREST = BALANCE * ((1 + RATE/(100*4))**(MONTHS/3) - 1)

 FINAL_BALANCE_SIMPLE = BALANCE + SIMPLE_INTEREST
 FINAL_BALANCE_COMPOUND = BALANCE + COMPOUND_INTEREST

 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}INTEREST CALCULATION{COLORS.END}")
 PRINT(F"CURRENT BALANCE: ₹{BALANCE:,.2F}")
 PRINT(F"ANNUAL INTEREST RATE: {RATE:.2F}%")
 PRINT(F"PERIOD: {MONTHS} MONTHS")
 PRINT(F"\N{COLORS.CYAN}RESULTS:{COLORS.END}")
 PRINT(F" SIMPLE INTEREST: ₹{SIMPLE_INTEREST:,.2F}")
 PRINT(F" FINAL BALANCE (SIMPLE): ₹{FINAL_BALANCE_SIMPLE:,.2F}")
 PRINT(F"\N COMPOUND INTEREST: ₹{COMPOUND_INTEREST:,.2F}")
 PRINT(F" FINAL BALANCE (COMPOUND): ₹{FINAL_BALANCE_COMPOUND:,.2F}")

DEF VIEW_CUSTOMER_FINANCIAL_DASHBOARD(DATA):
 """COMPREHENSIVE CUSTOMER FINANCIAL DASHBOARD"""
 PRINT(F"\N{COLORS.CYAN}--- CUSTOMER FINANCIAL DASHBOARD ---{COLORS.END}")

 CUST_ID = INPUT("CUSTOMER ID: ").STRIP()
 CUSTOMER = DATA['CUSTOMERS'][DATA['CUSTOMERS']['CUSTOMERID'] == CUST_ID]
```

```
 IF CUSTOMER.EMPTY:
  PRINT(F"{COLORS.RED}CUSTOMER NOT FOUND{COLORS.END}")
  RETURN

  CUST = CUSTOMER.ILOC[0]

  # GET ALL ACCOUNTS
  ACCOUNTS = DATA['ACCOUNTS'][DATA['ACCOUNTS']['CUSTOMERID'] == CUST_ID]
  TOTAL_BALANCE = ACCOUNTS['BALANCE'].SUM() IF NOT ACCOUNTS.EMPTY ELSE 0

  # GET ALL LOANS
  LOANS = DATA['LOANS'][DATA['LOANS']['CUSTOMERID'] == CUST_ID]
  TOTAL_LOANS = LOANS['OUTSTANDINGAMOUNT'].SUM() IF NOT LOANS.EMPTY ELSE 0

  # GET ALL CARDS
  CARDS = DATA['CARDS'][DATA['CARDS']['CUSTOMERID'] == CUST_ID]

  # GET TRANSACTIONS COUNT
  ACCOUNT_NUMS = ACCOUNTS['ACCOUNTNUMBER'].TOLIST() IF NOT ACCOUNTS.EMPTY
ELSE []
  TRANS = DATA['TRANSACTIONS'][DATA['TRANSACTIONS']
['ACCOUNTNUMBER'].ISIN(ACCOUNT_NUMS)]

  PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}{'='*60}")
  PRINT(F"{'FINANCIAL DASHBOARD':^60}")
  PRINT(F"{'='*60}{COLORS.END}\N")

  PRINT(F"CUSTOMER NAME: {CUST['NAME']}")
  PRINT(F"CUSTOMER ID: {CUST_ID}")
  PRINT(F"EMAIL: {CUST['EMAIL']}")
  PRINT(F"PHONE: {CUST['PHONE']}")

  PRINT(F"\N{COLORS.CYAN}PORTFOLIO OVERVIEW:{COLORS.END}")
  PRINT(F" TOTAL ACCOUNTS: {LEN(ACCOUNTS)}")
  PRINT(F" TOTAL BALANCE: ₹{TOTAL_BALANCE:,.2F}")
  PRINT(F" TOTAL LOANS: {LEN(LOANS)}")
  PRINT(F" TOTAL OUTSTANDING: ₹{TOTAL_LOANS:,.2F}")
  PRINT(F" CARDS ISSUED: {LEN(CARDS)}")
  PRINT(F" TOTAL TRANSACTIONS: {LEN(TRANS)}")

  NET_WORTH = TOTAL_BALANCE - TOTAL_LOANS
  PRINT(F"\N{COLORS.CYAN}NET WORTH:{COLORS.END}")
  PRINT(F" ASSETS (BALANCE): ₹{TOTAL_BALANCE:,.2F}")
  PRINT(F" LIABILITIES (LOANS): ₹{TOTAL_LOANS:,.2F}")
  PRINT(F" NET POSITION: ₹{NET_WORTH:,.2F}")

  IF TOTAL_LOANS > 0:
  LOAN_RATIO = (TOTAL_LOANS / (TOTAL_BALANCE + TOTAL_LOANS)) * 100
  PRINT(F" DEBT-TO-ASSETS RATIO: {LOAN_RATIO:.2F}%")
```

```
# ACCOUNT BREAKDOWN
 IF NOT ACCOUNTS.EMPTY:
 PRINT(F"\N{COLORS.CYAN}ACCOUNT BREAKDOWN:{COLORS.END}")
 FOR IDX, ACC IN ACCOUNTS.ITERROWS():
 STATUS_COLOR = COLORS.GREEN IF ACC['STATUS'] == 'ACTIVE' ELSE COLORS.RED
 PRINT(F" {STATUS_COLOR}{ACC['ACCOUNTNUMBER']:10} {ACC['ACCOUNTTYPE']:10} ₹
{ACC['BALANCE']:>12,.2F} {ACC['STATUS']}{COLORS.END}")

 # CREDIT SCORE
 CREDIT_SCORE = CALCULATE_CREDIT_SCORE(TOTAL_BALANCE, LEN(LOANS), LEN(TRANS))
 PRINT(F"\N{COLORS.CYAN}CREDIT SCORE: {COLORS.BOLD}{CREDIT_SCORE}
{COLORS.END}")
 IF CREDIT_SCORE >= 750:
 RATING = "EXCELLENT"
 ELIF CREDIT_SCORE >= 650:
 RATING = "GOOD"
 ELIF CREDIT_SCORE >= 550:
 RATING = "FAIR"
 ELSE:
 RATING = "POOR"
 PRINT(F"RATING: {RATING}")

DEF COMPARE_LOAN_OFFERS(DATA):
 """COMPARE DIFFERENT LOAN TYPES AND EMI"""
 PRINT(F"\N{COLORS.CYAN}--- LOAN COMPARISON TOOL ---{COLORS.END}")

 AMOUNT = FLOAT(INPUT("LOAN AMOUNT (₹): ").STRIP())
 TENURE = INT(INPUT("TENURE (MONTHS): ").STRIP())

 PRINT(F"\N{COLORS.BOLD}{COLORS.BLUE}LOAN COMPARISON:{COLORS.END}")
 PRINT(F"AMOUNT: ₹{AMOUNT:,.2F}")
 PRINT(F"TENURE: {TENURE} MONTHS ({TENURE/12:.1F} YEARS)\N")

 LOAN_TYPES = {
 'HOME LOAN': 8.5,
 'PERSONAL LOAN': 12.0,
 'CAR LOAN': 9.5,
 'EDUCATION LOAN': 7.5,
 'BUSINESS LOAN': 11.0}

 PRINT(F"{'LOAN TYPE':<20} {'INTEREST':<12} {'MONTHLY EMI':<15} {'TOTAL AMOUNT':<15}
{'TOTAL INTEREST':<15}")
 PRINT(F"{'-'*77}")

 FOR LOAN_TYPE, RATE IN LOAN_TYPES.ITEMS():
 EMI = CALCULATE_EMI(AMOUNT, RATE, TENURE)
 TOTAL = EMI * TENURE
 INTEREST = TOTAL - AMOUNT
 PRINT(F"{LOAN_TYPE:<20} {RATE:>6.2F}% {EMI:>14,.2F} {TOTAL:>14,.2F}{INTEREST:>14,.2F}")
```

```
# ========================================
# SECTION 9B: MENUS & MAIN LOOP
# ========================================

DEF MAIN():
 INITIALIZE_DATA()
 DATA = LOAD_DATA()

 PRINT(F"\N{COLORS.BOLD}{COLORS.GREEN}")

PRINT(" ╔═══════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╗ ")
 PRINT(" ║  WELCOME TO COREBANK SYSTEM V4.0 - ULTIMATE EDITION  ║ ")
 PRINT(" ║  A COMPREHENSIVE BANKING MANAGEMENT SOLUTION  ║ ")

PRINT(" ╚═══════════════════════════════════════════════════════════════════════
═══════════════════════════════════════╝ ")
 PRINT(F"{COLORS.END}")

 WHILE TRUE:
 DASHBOARD(DATA)
 PRINT(F"\N{COLORS.YELLOW}═══════ MAIN MENU ═══════{COLORS.END}")
 PRINT(F"\N{COLORS.CYAN}CUSTOMER & ACCOUNT:{COLORS.END}")
 PRINT(" 1. ADD CUSTOMER")
 PRINT(" 2. VIEW CUSTOMERS")
 PRINT(" 3. OPEN ACCOUNT")
 PRINT(" 4. CHECK BALANCE")
 PRINT(F"\N{COLORS.CYAN}TRANSACTIONS:{COLORS.END}")
 PRINT(" 5. DEPOSIT MONEY")
 PRINT(" 6. WITHDRAW MONEY")
 PRINT(" 7. FUND TRANSFER")
 PRINT(F"\N{COLORS.CYAN}LOAN MANAGEMENT:{COLORS.END}")
 PRINT(" 8. LOAN MENU")
 PRINT(F"\N{COLORS.CYAN}CARD & CHEQUE:{COLORS.END}")
 PRINT(" 9. CARD MANAGEMENT")
 PRINT(" 10. CHEQUE PROCESSING")
 PRINT(F"\N{COLORS.CYAN}ADVANCED FEATURES:{COLORS.END}")
 PRINT(" 11. ACCOUNT STATEMENT")
 PRINT(" 12. INTEREST CALCULATOR")
 PRINT(" 13. FINANCIAL DASHBOARD")
 PRINT(" 14. COMPARE LOAN OFFERS")
 PRINT(F"\N{COLORS.CYAN}REPORTS & UTILITIES:{COLORS.END}")
 PRINT(" 15. REPORTS & ANALYTICS")
 PRINT(" 16. BACKUP DATA")
 PRINT(" 17. SEARCH CUSTOMER")
 PRINT(F"\N 18. EXIT")
```

```
    CHOICE = INPUT(F"\N{COLORS.BOLD}SELECT OPTION: {COLORS.END}").STRIP()

    IF CHOICE == '1': DATA = ADD_CUSTOMER(DATA)
    ELIF CHOICE == '2': VIEW_CUSTOMERS(DATA)
    ELIF CHOICE == '3': DATA = OPEN_ACCOUNT(DATA)
    ELIF CHOICE == '4': CHECK_BALANCE(DATA)
    ELIF CHOICE == '5': DATA = DEPOSIT_MONEY(DATA)
    ELIF CHOICE == '6': DATA = WITHDRAW_MONEY(DATA)
    ELIF CHOICE == '7': DATA = TRANSFER_FUNDS(DATA)
    ELIF CHOICE == '8': DATA = LOAN_MANAGEMENT_MENU(DATA)
    ELIF CHOICE == '9': DATA = CARD_MANAGEMENT_MENU(DATA)
    ELIF CHOICE == '10': DATA = CHEQUE_MANAGEMENT_MENU(DATA)
    ELIF CHOICE == '11': GENERATE_ACCOUNT_STATEMENT(DATA)
    ELIF CHOICE == '12': CALCULATE_ACCOUNT_INTEREST(DATA)
    ELIF CHOICE == '13': VIEW_CUSTOMER_FINANCIAL_DASHBOARD(DATA)
    ELIF CHOICE == '14': COMPARE_LOAN_OFFERS(DATA)
    ELIF CHOICE == '15': GENERATE_REPORTS(DATA)
    ELIF CHOICE == '16': BACKUP_DATA()
    ELIF CHOICE == '17': SEARCH_CUSTOMER(DATA)
    ELIF CHOICE == '18':
    SAVE_DATA(DATA)
    PRINT(F"\N{COLORS.GREEN}THANK YOU FOR USING COREBANK. GOODBYE!
{COLORS.END}")
    BREAK
    ELSE:
    PRINT(F"{COLORS.RED}INVALID OPTION. PLEASE TRY AGAIN.{COLORS.END}")

    SAVE_DATA(DATA)

IF __NAME__ == "__MAIN__":
    MAIN()
```

OUTPUTS

KEY SYSTEM OUTPUTS:

- **Main Menu Interface** - Color-coded terminal UI with 18 menu options
- **Customer Management** - Create customer records with KYC validation
- **Account Operations** - Open savings, current, FD accounts with auto-generated account numbers
- **Transaction History** - View last 10 transactions with opening/closing balances
- **Fund Transfer Reports** - Track inter-account transfers with balance updates
- **Loan Analysis** - EMI calculations, amortization schedules, payment tracking
- **Financial Dashboards** - Pie charts, bar charts, line charts for analytics
- **Audit Trail Logs** - Complete timestamp-based operation logging
- **Credit Score Display** - Dynamic scoring 300-850 based on account activity
- **Account Statements** - Professional formatted statements with transaction details
- **Backup Files** - Timestamped CSV backups with data preservation
- **Data Export** - Financial reports in structured CSV format

OUTPUTS

```
Raghav@Raghavs-MacBook-Pro IP Project % python3 ./bank_
```

```
Welcome to CoreBank System v4.0 - Ultimate Edition
A Comprehensive Banking Management Solution
```

```
COREBANK MANAGEMENT SYSTEM
```

```
Quick Stats:
  Customers: 12       Accounts: 15       Cards: 8
  Balance:   ₹9,099,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 16:47:51
```

```
=== MAIN MENU ===

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 1

--- Add New Customer ---
Enter customer name: Ashsish Chanchalani
Enter DOB (YYYY-MM-DD): 1998-12-08
Enter gender: Male
Enter PAN: ASHIS0150C
Enter Aadhar: 111122223333
Enter address: Ulhasnagar
Enter city: Mumbai
Enter state: Maharashtra
Enter PIN: 401001
Enter phone: 9876543210
Enter email:
✓ Customer CUST013 added!
```

```
COREBANK MANAGEMENT SYSTEM
```

```
Quick Stats:
  Customers: 13       Accounts: 15       Cards: 8
  Balance:   ₹9,099,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 16:49:20
```

```
=== MAIN MENU ===

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 2

--- Customer List ---
CustomerID                Name      Phone Status
   CUST001      Raghav Agarwal 9876543210 Active
   CUST002        Priya Sharma 8765432109 Active
   CUST003          Amit Kumar 7654321098 Active
   CUST004          Neha Patel 6543210987 Active
   CUST005        Vikram Singh 5432109876 Active
   CUST006        Anjali Verma 4321098765 Active
   CUST007         Rohan Desai 3210987654 Active
   CUST008          Isha Gupta 2109876543 Active
   CUST009          Arjun Nair 1098765432 Active
   CUST010          Divya Iyer 9988776655 Active
   CUST011                Goru 8989898989 Active
   CUST012      Manvink Khatri 3293283232 Active
   CUST013 Ashsish Chanchalani 9876543210 Active
```

OUTPUTS

```
COREBANK MANAGEMENT SYSTEM

Quick Stats:
  Customers: 13       Accounts: 15       Cards: 8
  Balance:   ₹9,099,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 16:51:21
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 3

--- Open New Account ---
Enter Customer ID: CUST013
1. Savings (4%)  2. Current (0%)  3. Fixed Deposit (7.5%)
Select type: 3
Initial deposit: 5000000
✓ Account ACC10006 opened!
```

```
COREBANK MANAGEMENT SYSTEM

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,104,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 16:55:55
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 4
Enter Account Number: ACC10006

Account: ACC10006 (Fixed Deposit)
Balance: ₹5000000.00
```

OUTPUTS

```
╔══════════════════════════════════════╗
║      COREBANK MANAGEMENT SYSTEM      ║
╚══════════════════════════════════════╝

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,104,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 16:59:53
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 5

--- Deposit ---
Account Number: ACC10006
Amount: 5000000
✓ Deposited ₹5000000.0. New Balance: ₹10000000.00
```

```
╔══════════════════════════════════════╗
║      COREBANK MANAGEMENT SYSTEM      ║
╚══════════════════════════════════════╝

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,109,262,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 17:03:12
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 6

--- Withdraw ---
Account Number: ACC10006
Amount: 2500000
✓ Withdrawn ₹2500000.0. New Balance: ₹7500000.00
```

OUTPUTS

```
╔══════════════════════════════════════╗
║      COREBANK MANAGEMENT SYSTEM      ║
╚══════════════════════════════════════╝

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,106,762,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 17:04:00
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 7

--- Fund Transfer ---
1. Transfer to own account
2. Transfer to other customer
Select: 1
From Account Number: ACC1001
To Account Number: ACC1002
Amount: ₹200000
✓ Transfer Successful!
Reference: TRF20251214170617
Amount: ₹200,000.00
From ACC1001: ₹500,047,000.00
To ACC1002: ₹230,000.00
```

```
╔══════════════════════════════════════╗
║      COREBANK MANAGEMENT SYSTEM      ║
╚══════════════════════════════════════╝

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,106,762,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 17:06:17
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 7

--- Fund Transfer ---
1. Transfer to own account
2. Transfer to other customer
Select: 2
From Account Number: ACC1001
To Account Number: ACC10006
Amount: ₹300000
Exceeds daily transfer limit of ₹200,000
```

OUTPUTS

```
        COREBANK MANAGEMENT SYSTEM

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,106,762,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 17:07:11
──────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 8

=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 1

--- Apply Loan ---
Customer ID: CUST013
Loan Types: Home Loan, Personal Loan, Car Loan, Education Loan, Busine
Type: Home Loan
Amount: 2000000
Tenure (months): 12
✓ Loan LOAN008 approved! EMI: ₹174439.56
```

```
=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 2

--- Pay Loan EMI ---

Active Loans:
 LoanID       LoanType       EMI  OutstandingAmount
LOAN001  Personal Loan   8886.00           350000.0
LOAN002      Home Loan  25000.00          2300000.0
LOAN003       Car Loan  12500.00           600000.0
LOAN004 Education Loan  14500.00           850000.0
LOAN005  Business Loan  45000.00          2500000.0
LOAN006  Personal Loan   5330.00           100000.0
LOAN007 Education Loan    327.85             8999.0
LOAN008      Home Loan 174439.56          2000000.0

Enter Loan ID: LOAN008

EMI Amount: ₹174439.56
Outstanding: ₹2000000.00
/Users/raghav/Developer/IP Project/./bank_management_system.py:
710: FutureWarning: The behavior of DataFrame concatenation wit
h empty or all-NA entries is deprecated. In a future version, t
his will no longer exclude empty or all-NA columns when determi
ning the result dtypes. To retain the old behavior, exclude the
 relevant entries before the concat operation.
  data['loan_payments'] = pd.concat([data['loan_payments'], pd.
DataFrame([payment])], ignore_index=True)
✓ EMI paid! Outstanding: ₹1839727.11

=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 3

--- Loan Details ---
Enter Loan ID: LOAN008

================================================
Loan ID:         LOAN008
Type:            Home Loan
Principal:       ₹2,000,000.00
Interest Rate:   8.5% p.a.
Tenure:          12.0 months
EMI:             ₹174,439.56
Outstanding:     ₹1,839,727.11
Status:          Active
Repaid:          ₹160,272.89 (8.0%)
================================================

Payment History:
PaymentDate Date  AmountPaid Amount PrincipalPart InterestPart
 2025-12-14  NaN ₹174,439.56    N/A   ₹160,272.89   ₹14,166.67
```

OUTPUTS

```
        COREBANK MANAGEMENT SYSTEM

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 8
  Balance:   ₹9,106,762,000.00
  Loans:     7        Outstanding: ₹6,708,999.00
  Time:      2025-12-14 17:07:11
──────────────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 8

=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 1

--- Apply Loan ---
Customer ID: CUST013
Loan Types: Home Loan, Personal Loan, Car Loan, Education Loan, Busine
Type: Home Loan
Amount: 2000000
Tenure (months): 12
✓ Loan LOAN008 approved! EMI: ₹174439.56
```

```
=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 2

--- Pay Loan EMI ---

Active Loans:
 LoanID       LoanType       EMI  OutstandingAmount
LOAN001  Personal Loan   8886.00           350000.0
LOAN002      Home Loan  25000.00          2300000.0
LOAN003       Car Loan  12500.00           600000.0
LOAN004 Education Loan  14500.00           850000.0
LOAN005  Business Loan  45000.00          2500000.0
LOAN006  Personal Loan   5330.00           100000.0
LOAN007 Education Loan    327.85             8999.0
LOAN008      Home Loan 174439.56          2000000.0

Enter Loan ID: LOAN008

EMI Amount: ₹174439.56
Outstanding: ₹2000000.00
/Users/raghav/Developer/IP Project/./bank_management_system.py:
710: FutureWarning: The behavior of DataFrame concatenation wit
h empty or all-NA entries is deprecated. In a future version, t
his will no longer exclude empty or all-NA columns when determi
ning the result dtypes. To retain the old behavior, exclude the
 relevant entries before the concat operation.
  data['loan_payments'] = pd.concat([data['loan_payments'], pd.
DataFrame([payment])], ignore_index=True)
✓ EMI paid! Outstanding: ₹1839727.11

=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 3

--- Loan Details ---
Enter Loan ID: LOAN008

================================================
Loan ID:         LOAN008
Type:            Home Loan
Principal:       ₹2,000,000.00
Interest Rate:   8.5% p.a.
Tenure:          12.0 months
EMI:             ₹174,439.56
Outstanding:     ₹1,839,727.11
Status:          Active
Repaid:          ₹160,272.89 (8.0%)
================================================

Payment History:
PaymentDate Date  AmountPaid Amount PrincipalPart InterestPart
 2025-12-14  NaN ₹174,439.56    N/A   ₹160,272.89   ₹14,166.67
```

```
=== LOAN MANAGEMENT ===
1. Apply for Loan
2. Pay Loan EMI
3. View Loan Details
4. View All Loans
5. Back to Main Menu

Select: 4
 LoanID CustomerID       LoanType  PrincipalAmount       EMI  OutstandingAmount Status
LOAN001    CUST001  Personal Loan         500000.0   8886.00          350000.00 Active
LOAN002    CUST002      Home Loan        2500000.0  25000.00         2300000.00 Active
LOAN003    CUST003       Car Loan         800000.0  12500.00          600000.00 Active
LOAN004    CUST005 Education Loan        1000000.0  14500.00          850000.00 Active
LOAN005    CUST007  Business Loan        3000000.0  45000.00         2500000.00 Active
LOAN006    CUST004  Personal Loan         300000.0   5330.00          100000.00 Active
LOAN007    CUST003 Education Loan           8999.0    327.85            8999.00 Active
LOAN008    CUST013      Home Loan        2000000.0 174439.56         1839727.11 Active
```

OUTPUTS

```
Select Option: 9

=== CARD MANAGEMENT ===
1. Issue New Card
2. View My Cards
3. Block/Unblock Card
4. View Card Transactions
5. Change Card PIN
6. Back to Main Menu

Select: 1

--- Issue New Card ---
Customer ID: CUST013

Customer Accounts:
AccountNumber   AccountType   Balance
     ACC10006 Fixed Deposit 7500000.0

Link to Account Number: ACC10006

Card Type:
1. Debit Card
2. Credit Card
Select: 1
Set 4-digit PIN: 1234

✓ Card Issued Successfully!
==========================================
Card Number: XXXX-XXXX-XXXX-2077
Card Type:   Debit
Linked A/C:  ACC10006
Valid Till:  2030-12-13
CVV:         084 (Keep Secret!)
==========================================
Note: Full card number will be printed on physical card.

=== CARD MANAGEMENT ===
1. Issue New Card
2. View My Cards
3. Block/Unblock Card
4. View Card Transactions
5. Change Card PIN
6. Back to Main Menu

Select: 2

--- View Cards ---
Customer ID: CUST013

============================================================
Card:    XXXX-XXXX-XXXX-2077
Type:    Debit
Account: ACC10006
Expiry:  2030-12-13
Status:  Active
------------------------------------------------------------
```

```
=== CARD MANAGEMENT ===
1. Issue New Card
2. View My Cards
3. Block/Unblock Card
4. View Card Transactions
5. Change Card PIN
6. Back to Main Menu

Select: 2

--- View Cards ---
Customer ID: CUST013

========================================
Card:    XXXX-XXXX-XXXX-2077
Type:    Debit
Account: ACC10006
Expiry:  2030-12-13
Status:  Active
----------------------------------------
Card:    XXXX-XXXX-XXXX-1288
Type:    Credit
Account: ACC10006
Expiry:  2030-12-13
Status:  Active
Limit:   ₹5,000,000.00
----------------------------------------

=== CARD MANAGEMENT ===
1. Issue New Card
2. View My Cards
3. Block/Unblock Card
4. View Card Transactions
5. Change Card PIN
6. Back to Main Menu

Select: 3

--- Block/Unblock Card ---
Enter last 4 digits of card: 2077
Current Status: Active
Block this card? (yes/no): yes
✓ Card blocked successfully

=== CARD MANAGEMENT ===
1. Issue New Card
2. View My Cards
3. Block/Unblock Card
4. View Card Transactions
5. Change Card PIN
6. Back to Main Menu

Select: 2

--- View Cards ---
Customer ID: CUST013

========================================
Card:    XXXX-XXXX-XXXX-2077
Type:    Debit
Account: ACC10006
Expiry:  2030-12-13
Status:  Blocked
----------------------------------------
Card:    XXXX-XXXX-XXXX-1288
Type:    Credit
Account: ACC10006
Expiry:  2030-12-13
Status:  Active
Limit:   ₹5,000,000.00
```

OUTPUTS

```
Select Option: 10

=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 1

--- Issue Cheque ---
From Account Number: ACC10006
Payee Name: Manvik
Amount: ₹502

✓ Cheque Issued
Cheque Number: 489123
Amount: ₹502.00
Payee: Manvik

=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 2

--- Deposit Cheque ---
Cheque Number: 489123
Deposit to Account Number: ACC10004

✓ Cheque Cleared Successfully!
Amount: ₹502.00
Deposited to: ACC10004

=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 3

--- Cheque Status ---
Cheque Number: 489123

Cheque Number: 489123
From Account: ACC10006
Payee:        Manvik
Amount:       ₹502.00
Issue Date:   2025-12-14
Status:       Cleared
Cleared On:   2025-12-14
```

```
=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 4

--- Cancel Cheque ---
Cheque Number: 489123
Only issued cheques can be cancelled

=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 5

--- All Cheques ---
Account Number: ACC10006
ChequeNumber IssuedTo  Amount  Status  IssueDate
      489123   Manvik   502.0 Cleared 2025-12-14

=== CHEQUE PROCESSING ===
1. Issue New Cheque
2. Deposit Cheque
3. View Cheque Status
4. Cancel Cheque
5. View All Cheques
6. Back to Main Menu

Select: 6
```

```
Select Option: 11

--- Account Statement ---
Account Number: ACC10006

============================================================
                  COREBANK ACCOUNT STATEMENT
============================================================

Account Holder: Ashsish Chanchalani
Account Number: ACC10006
Account Type: Fixed Deposit
Statement Date: 2025-12-14

Account Summary:
  Opening Balance: ₹7,499,498.00
  Current Balance: ₹7,499,498.00
  Interest Rate: 7.50%
  Minimum Balance: ₹0.00
  Status: Active

Recent Transactions (Last 10):
------------------------------------------------------------
2025-12-14   Account Op -₹5,000,000.00 Balance: ₹5,000,000.00
2025-12-14   Deposit    +₹5,000,000.00 Balance: ₹10,000,000.00
2025-12-14   Withdrawal -₹2,500,000.00 Balance: ₹7,500,000.00
2025-12-14   Cheque Deb -₹    502.00 Balance: ₹7,499,498.00
------------------------------------------------------------

Select Option: 12

--- Interest Calculator ---
Account Number: ACC10006
Number of months to calculate: 18

Interest Calculation
Current Balance: ₹7,499,498.00
Annual Interest Rate: 7.50%
Period: 18 months

Results:
  Simple Interest: ₹843,693.53
  Final Balance (Simple): ₹8,343,191.53

  Compound Interest: ₹884,244.37
  Final Balance (Compound): ₹8,383,742.37
```

```
Select Option: 13

--- Customer Financial Dashboard ---
Customer ID: CUST013

============================================================
                    FINANCIAL DASHBOARD
============================================================

Customer Name: Ashsish Chanchalani
Customer ID: CUST013
Email:
Phone: 9876543210

Portfolio Overview:
  Total Accounts: 1
  Total Balance: ₹7,499,498.00
  Total Loans: 1
  Total Outstanding: ₹1,839,727.11
  Cards Issued: 2
  Total Transactions: 4

Net Worth:
  Assets (Balance): ₹7,499,498.00
  Liabilities (Loans): ₹1,839,727.11
  Net Position: ₹5,659,770.89
  Debt-to-Assets Ratio: 19.70%

Account Breakdown:
  ACC10006   Fixed Deposit ₹7,499,498.00 Active

Credit Score: 772
Rating: Excellent
```

```
Select Option: 14

--- Loan Comparison Tool ---
Loan Amount (₹): 1400000
Tenure (months): 24

Loan Comparison:
Amount: ₹1,400,000.00
Tenure: 24 months (2.0 years)

Loan Type            Interest     Monthly EMI     Total Amount    Total Interest
------------------------------------------------------------------------------
Home Loan              8.50%      63,637.94   1,527,310.56     127,310.56
Personal Loan         12.00%      65,902.86   1,581,668.64     181,668.64
Car Loan               9.50%      64,280.29   1,542,726.96     142,726.96
Education Loan         7.50%      62,999.43   1,511,986.32     111,986.32
Business Loan         11.00%      65,250.97   1,566,023.28     166,023.28
```

```
Select Option: 15

=== REPORTS & ANALYTICS ===

Text Reports:
1.  Transaction History
2.  Bank Financial Summary
3.  Customer Balances Report
4.  Daily Transactions Summary
5.  Loan Portfolio Analysis
6.  View Audit Trail
7.  Customer Credit Score
8.  Database Summary (All Tables)

Visual Charts (Matplotlib):
9.  Account Distribution (Pie Chart)
10. Loan Portfolio (Bar Chart)
11. Monthly Transaction Trend
12. Customer Growth Chart
13. Balance Distribution Analysis
14. Transaction Types Breakdown
15. Loan EMI Analysis (4 Charts)
16. Daily Activity Timeline
17. Comprehensive Dashboard (6 Charts)

18. Back to Main Menu

Select Report: 1

--- Transaction History ---
Enter Account Number: ACC10006

History for ACC10006:
--------------------------------------------------------------------------------
Date         Type                      Amount Dr/Cr           Balance
--------------------------------------------------------------------------------
2025-12-14   Account Opening      ₹5,000,000.00 Credit   ₹5,000,000.00
2025-12-14   Deposit              ₹5,000,000.00 Credit   ₹10,000,000.00
2025-12-14   Withdrawal           ₹2,500,000.00 Debit    ₹7,500,000.00
2025-12-14   Cheque Debit         ₹    502.00 Debit    ₹7,499,498.00
--------------------------------------------------------------------------------

Select Report: 3

--- Customer Balances Report ---

ID          Name                      Total Balance   Accounts Loans
------------------------------------------------------------------------
CUST001     Raghav Agarwal            ₹505,622,000.00 4        1
CUST002     Priya Sharma              ₹320,000.00     1        1
CUST003     Amit Kumar                ₹180,000.00     1        2
CUST004     Neha Patel                ₹500,000.00     1        1
CUST005     Vikram Singh              ₹450,000.00     1        1
CUST006     Anjali Verma              ₹280,000.00     1        0
CUST007     Rohan Desai               ₹620,000.00     1        1
CUST008     Isha Gupta                ₹340,000.00     1        0
CUST009     Arjun Nair                ₹200,000.00     1        0
CUST010     Divya Iyer                ₹750,000.00     1        0
CUST011     Goru                      ₹90,000,000.00  1        0
CUST012     Manvink Khatri            ₹8,500,000,502.00 1        0
CUST013     Ashsish Chanchalani       ₹7,499,498.00   1        1
```

```
Select Report: 2

--- Bank Financial Summary ---

==================================================
FINANCIAL OVERVIEW
==================================================
Total Deposits Held:     ₹9,106,762,000.00
Total Loans Outstanding: ₹   8,548,726.11
Net Liquidity:           ₹9,098,213,273.89
==================================================
TRANSACTION VOLUME
==================================================
Total Credit Volume:     ₹9,606,233,502.00
Total Debit Volume:      ₹ 503,423,502.00
Net Flow:                ₹9,102,810,000.00
==================================================
STATISTICS
==================================================
Total Customers:                        13
Total Accounts:                         16
Active Loans:                            8
Cards Issued:                           10
==================================================
Select Report: 4

--- Daily Transaction Summary ---

Date: 2025-12-14
================================================
Total Transactions: 18
================================================
Credits: 11 transactions, ₹9,605,665,502.00
Debits:  7 transactions, ₹502,943,502.00
================================================

Transaction Details:
TransactionID AccountNumber TransactionType       Amount DebitCredit
     TXN00074       ACC1002    Cheque Debit     120000.0       Debit
     TXN00075       ACC1001   Cheque Credit     120000.0      Credit
     TXN00076      ACC10002 Account Opening     300000.0      Credit
     TXN00077       ACC1001   Fund Transfer      45000.0       Debit
     TXN00078      ACC10002   Fund Transfer      45000.0      Credit
     TXN00079       ACC1001      Withdrawal      78000.0       Debit
     TXN00080      ACC10003 Account Opening   90000000.0      Credit
     TXN00081      ACC10004 Account Opening 9000000000.0      Credit
     TXN00082      ACC10004    Cheque Debit  500000000.0       Debit
     TXN00083       ACC1001   Cheque Credit  500000000.0      Credit
     TXN00084      ACC10005 Account Opening    5000000.0      Credit
     TXN00085      ACC10006 Account Opening    5000000.0      Credit
     TXN00086      ACC10006         Deposit    5000000.0      Credit
     TXN00087      ACC10006      Withdrawal    2500000.0       Debit
     TXN00088       ACC1001   Fund Transfer     200000.0       Debit
     TXN00089       ACC1002   Fund Transfer     200000.0      Credit
     TXN00090      ACC10006    Cheque Debit        502.0       Debit
     TXN00091      ACC10004   Cheque Credit        502.0      Credit
```

```
Select Report: 5

--- Loan Portfolio Analysis ---

============================================================
LOAN PORTFOLIO SUMMARY
============================================================

By Status:
        Count  PrincipalAmount  OutstandingAmount
Status
Active      8       10108999.0         8548726.11


By Loan Type:
                Count  PrincipalAmount  OutstandingAmount  Avg Rate
LoanType
Business Loan       1        3000000.0          2500000.00     11.00
Car Loan            1         800000.0           600000.00      9.50
Education Loan      2        1008999.0           858999.00      7.25
Home Loan           2        4500000.0          4139727.11      8.50
Personal Loan       2         800000.0           450000.00     12.00
```

```
Select Report: 6

--- Audit Trail ---
1. View all logs
2. Filter by action type
3. Filter by date

Select: 1

------------------------------------------------------------------------------------------------
Timestamp           Action              Details                                      Status
------------------------------------------------------------------------------------------------
2025-10-23 14:02:28  None               Operation 65 executed successfully           Success
2025-12-11 07:02:28  None               Operation 66 executed successfully           Success
2025-10-18 05:02:28  None               Operation 67 executed successfully           Success
2025-11-21 04:02:28  None               Operation 68 executed successfully           Success
2025-11-10 14:02:28  None               Operation 69 executed successfully           Success
2025-12-04 08:02:28  None               Operation 70 executed successfully           Success
2025-10-11 03:02:28  None               Operation 71 executed successfully           Success
2025-10-15 18:02:28  None               Operation 72 executed successfully           Success
2025-11-23 08:02:28  None               Operation 73 executed successfully           Success
2025-11-25 21:02:28  None               Operation 74 executed successfully           Success
2025-12-02 12:02:28  None               Operation 75 executed successfully           Success
2025-11-22 07:02:28  None               Operation 76 executed successfully           Success
2025-09-27 11:02:28  None               Operation 77 executed successfully           Success
2025-11-16 01:02:28  None               Operation 78 executed successfully           Success
2025-11-22 03:02:28  None               Operation 79 executed successfully           Success
2025-10-08 18:02:28  None               Operation 80 executed successfully           Success
2025-09-17 07:02:28  None               Operation 81 executed successfully           Success
2025-09-15 22:02:28  None               Operation 82 executed successfully           Success
2025-10-08 09:02:28  None               Operation 83 executed successfully           Success
2025-10-01 23:02:28  None               Operation 84 executed successfully           Success
2025-09-25 14:02:28  None               Operation 85 executed successfully           Success
2025-11-27 07:02:28  None               Operation 86 executed successfully           Success
2025-10-14 07:02:28  None               Operation 87 executed successfully           Success
2025-12-10 06:02:28  None               Operation 88 executed successfully           Success
2025-12-02 08:02:28  None               Operation 89 executed successfully           Success
2025-11-08 08:02:28  None               Operation 90 executed successfully           Success
2025-11-13 05:02:28  None               Operation 91 executed successfully           Success
2025-09-18 12:02:28  None               Operation 92 executed successfully           Success
2025-10-25 16:02:28  None               Operation 93 executed successfully           Success
2025-11-17 04:02:28  None               Operation 94 executed successfully           Success
2025-10-10 10:02:28  None               Operation 95 executed successfully           Success
2025-09-15 00:02:28  None               Operation 96 executed successfully           Success
2025-10-24 01:02:28  None               Operation 97 executed successfully           Success
2025-10-05 15:02:28  None               Operation 98 executed successfully           Success
2025-11-04 16:02:28  None               Operation 99 executed successfully           Success
2025-09-15 09:02:28  None               Operation 100 executed successfully          Success
2025-12-14 09:29:36  CHEQUE_ISSUED      Cheque 631773 for ₹120000.0                  Success
2025-12-14 09:30:23  CHEQUE_CLEARED     Cheque 631773 for ₹120000.0                  Success
2025-12-14 09:34:58  FUND_TRANSFER      ₹45000.0 from ACC1001 to ACC10002            Success
2025-12-14 09:36:57  LOAN_APPLIED       Loan LOAN007 for ₹8999.0                     Success
2025-12-14 10:10:35  CHEQUE_ISSUED      Cheque 811153 for ₹500000000.0               Success
2025-12-14 10:11:28  CHEQUE_CLEARED     Cheque 811153 for ₹500000000.0               Success
2025-12-14 17:06:17  FUND_TRANSFER      ₹200000.0 from ACC1001 to ACC1002            Success
2025-12-14 17:10:18  LOAN_APPLIED       Loan LOAN008 for ₹2000000.0                  Success
2025-12-14 17:11:43  EMI_PAID           EMI ₹174439.56 for LOAN008                   Success
2025-12-14 17:21:34  CARD_ISSUED        Debit card for CUST013                       Success
2025-12-14 17:24:55  CARD_ISSUED        Credit card for CUST013                      Success
2025-12-14 17:25:42  CARD_BLOCKED       Card ending 2077                             Success
2025-12-14 17:29:20  CHEQUE_ISSUED      Cheque 489123 for ₹502.0                     Success
2025-12-14 17:30:41  CHEQUE_CLEARED     Cheque 489123 for ₹502.0                     Success
```

```
Select Report: 5

--- Loan Portfolio Analysis ---

============================================================
LOAN PORTFOLIO SUMMARY
============================================================

By Status:
        Count  PrincipalAmount  OutstandingAmount
Status
Active      8       10108999.0         8548726.11


By Loan Type:
                Count  PrincipalAmount  OutstandingAmount  Avg Rate
LoanType
Business Loan       1        3000000.0         2500000.00     11.00
Car Loan            1         800000.0          600000.00      9.50
Education Loan      2        1008999.0          858999.00      7.25
Home Loan           2        4500000.0         4139727.11      8.50
Personal Loan       2         800000.0          450000.00     12.00
```

```
Select Report: 6

--- Audit Trail ---
1. View all logs
2. Filter by action type
3. Filter by date

Select: 1

------------------------------------------------------------------------------------------------
Timestamp           Action              Details                                    Status
------------------------------------------------------------------------------------------------
2025-10-23 14:02:28  None               Operation 65 executed successfully         Success
2025-12-11 07:02:28  None               Operation 66 executed successfully         Success
2025-10-18 05:02:28  None               Operation 67 executed successfully         Success
2025-11-21 04:02:28  None               Operation 68 executed successfully         Success
2025-11-10 14:02:28  None               Operation 69 executed successfully         Success
2025-12-04 08:02:28  None               Operation 70 executed successfully         Success
2025-10-11 03:02:28  None               Operation 71 executed successfully         Success
2025-10-15 18:02:28  None               Operation 72 executed successfully         Success
2025-11-23 08:02:28  None               Operation 73 executed successfully         Success
2025-11-25 21:02:28  None               Operation 74 executed successfully         Success
2025-12-02 12:02:28  None               Operation 75 executed successfully         Success
2025-11-22 07:02:28  None               Operation 76 executed successfully         Success
2025-09-27 11:02:28  None               Operation 77 executed successfully         Success
2025-11-16 01:02:28  None               Operation 78 executed successfully         Success
2025-11-22 03:02:28  None               Operation 79 executed successfully         Success
2025-10-08 18:02:28  None               Operation 80 executed successfully         Success
2025-09-17 07:02:28  None               Operation 81 executed successfully         Success
2025-09-15 22:02:28  None               Operation 82 executed successfully         Success
2025-10-08 09:02:28  None               Operation 83 executed successfully         Success
2025-10-01 23:02:28  None               Operation 84 executed successfully         Success
2025-09-25 14:02:28  None               Operation 85 executed successfully         Success
2025-11-27 07:02:28  None               Operation 86 executed successfully         Success
2025-10-14 07:02:28  None               Operation 87 executed successfully         Success
2025-12-10 06:02:28  None               Operation 88 executed successfully         Success
2025-12-02 08:02:28  None               Operation 89 executed successfully         Success
2025-11-08 08:02:28  None               Operation 90 executed successfully         Success
2025-11-13 05:02:28  None               Operation 91 executed successfully         Success
2025-09-18 12:02:28  None               Operation 92 executed successfully         Success
2025-10-25 16:02:28  None               Operation 93 executed successfully         Success
2025-11-17 04:02:28  None               Operation 94 executed successfully         Success
2025-10-10 10:02:28  None               Operation 95 executed successfully         Success
2025-09-15 00:02:28  None               Operation 96 executed successfully         Success
2025-10-24 01:02:28  None               Operation 97 executed successfully         Success
2025-10-05 15:02:28  None               Operation 98 executed successfully         Success
2025-11-04 16:02:28  None               Operation 99 executed successfully         Success
2025-09-15 09:02:28  None               Operation 100 executed successfully        Success
2025-12-14 09:29:36  CHEQUE_ISSUED      Cheque 631773 for ₹120000.0                Success
2025-12-14 09:30:23  CHEQUE_CLEARED     Cheque 631773 for ₹120000.0                Success
2025-12-14 09:34:58  FUND_TRANSFER      ₹45000.0 from ACC1001 to ACC10002          Success
2025-12-14 09:36:57  LOAN_APPLIED       Loan LOAN007 for ₹8999.0                   Success
2025-12-14 10:10:35  CHEQUE_ISSUED      Cheque 811153 for ₹500000000.0             Success
2025-12-14 10:11:28  CHEQUE_CLEARED     Cheque 811153 for ₹500000000.0             Success
2025-12-14 17:06:17  FUND_TRANSFER      ₹200000.0 from ACC1001 to ACC1002          Success
2025-12-14 17:10:18  LOAN_APPLIED       Loan LOAN008 for ₹2000000.0                Success
2025-12-14 17:11:43  EMI_PAID           EMI ₹174439.56 for LOAN008                 Success
2025-12-14 17:21:34  CARD_ISSUED        Debit card for CUST013                     Success
2025-12-14 17:24:55  CARD_ISSUED        Credit card for CUST013                    Success
2025-12-14 17:25:42  CARD_BLOCKED       Card ending 2077                           Success
2025-12-14 17:29:20  CHEQUE_ISSUED      Cheque 489123 for ₹502.0                   Success
2025-12-14 17:30:41  CHEQUE_CLEARED     Cheque 489123 for ₹502.0                   Success
```

OUTPUTS

```
Select Report: 5

--- Loan Portfolio Analysis ---

============================================================
LOAN PORTFOLIO SUMMARY
============================================================

By Status:
        Count  PrincipalAmount  OutstandingAmount
Status
Active      8       10108999.0         8548726.11


By Loan Type:
                Count  PrincipalAmount  OutstandingAmount  Avg Rate
LoanType
Business Loan       1        3000000.0          2500000.00     11.00
Car Loan            1         800000.0           600000.00      9.50
Education Loan      2        1008999.0           858999.00      7.25
Home Loan           2        4500000.0          4139727.11      8.50
Personal Loan       2         800000.0           450000.00     12.00
```

```
Select Report: 6

--- Audit Trail ---
1. View all logs
2. Filter by action type
3. Filter by date

Select: 1

------------------------------------------------------------------------------------------------------------
Timestamp           Action              Details                                          Status
------------------------------------------------------------------------------------------------------------
2025-10-23 14:02:28  None               Operation 65 executed successfully               Success
2025-12-11 07:02:28  None               Operation 66 executed successfully               Success
2025-10-18 05:02:28  None               Operation 67 executed successfully               Success
2025-11-21 04:02:28  None               Operation 68 executed successfully               Success
2025-11-10 14:02:28  None               Operation 69 executed successfully               Success
2025-12-04 08:02:28  None               Operation 70 executed successfully               Success
2025-10-11 03:02:28  None               Operation 71 executed successfully               Success
2025-10-15 18:02:28  None               Operation 72 executed successfully               Success
2025-11-23 08:02:28  None               Operation 73 executed successfully               Success
2025-11-25 21:02:28  None               Operation 74 executed successfully               Success
2025-12-02 12:02:28  None               Operation 75 executed successfully               Success
2025-11-22 07:02:28  None               Operation 76 executed successfully               Success
2025-09-27 11:02:28  None               Operation 77 executed successfully               Success
2025-11-16 01:02:28  None               Operation 78 executed successfully               Success
2025-11-22 03:02:28  None               Operation 79 executed successfully               Success
2025-10-08 18:02:28  None               Operation 80 executed successfully               Success
2025-09-17 07:02:28  None               Operation 81 executed successfully               Success
2025-09-15 22:02:28  None               Operation 82 executed successfully               Success
2025-10-08 09:02:28  None               Operation 83 executed successfully               Success
2025-10-01 23:02:28  None               Operation 84 executed successfully               Success
2025-09-25 14:02:28  None               Operation 85 executed successfully               Success
2025-11-27 07:02:28  None               Operation 86 executed successfully               Success
2025-10-14 07:02:28  None               Operation 87 executed successfully               Success
2025-12-10 06:02:28  None               Operation 88 executed successfully               Success
2025-12-02 08:02:28  None               Operation 89 executed successfully               Success
2025-11-08 08:02:28  None               Operation 90 executed successfully               Success
2025-11-13 05:02:28  None               Operation 91 executed successfully               Success
2025-09-18 12:02:28  None               Operation 92 executed successfully               Success
2025-10-25 16:02:28  None               Operation 93 executed successfully               Success
2025-11-17 04:02:28  None               Operation 94 executed successfully               Success
2025-10-10 10:02:28  None               Operation 95 executed successfully               Success
2025-09-15 00:02:28  None               Operation 96 executed successfully               Success
2025-10-24 01:02:28  None               Operation 97 executed successfully               Success
2025-10-05 15:02:28  None               Operation 98 executed successfully               Success
2025-11-04 16:02:28  None               Operation 99 executed successfully               Success
2025-09-15 09:02:28  None               Operation 100 executed successfully              Success
2025-12-14 09:29:36  CHEQUE_ISSUED      Cheque 631773 for ₹120000.0                      Success
2025-12-14 09:30:23  CHEQUE_CLEARED     Cheque 631773 for ₹120000.0                      Success
2025-12-14 09:34:58  FUND_TRANSFER      ₹45000.0 from ACC1001 to ACC10002                Success
2025-12-14 09:36:57  LOAN_APPLIED       Loan LOAN007 for ₹8999.0                         Success
2025-12-14 10:10:35  CHEQUE_ISSUED      Cheque 811153 for ₹500000000.0                   Success
2025-12-14 10:11:28  CHEQUE_CLEARED     Cheque 811153 for ₹500000000.0                   Success
2025-12-14 17:06:17  FUND_TRANSFER      ₹200000.0 from ACC1001 to ACC1002                Success
2025-12-14 17:10:18  LOAN_APPLIED       Loan LOAN008 for ₹2000000.0                      Success
2025-12-14 17:11:43  EMI_PAID           EMI ₹174439.56 for LOAN008                       Success
2025-12-14 17:21:34  CARD_ISSUED        Debit card for CUST013                           Success
2025-12-14 17:24:55  CARD_ISSUED        Credit card for CUST013                          Success
2025-12-14 17:25:42  CARD_BLOCKED       Card ending 2077                                 Success
2025-12-14 17:29:20  CHEQUE_ISSUED      Cheque 489123 for ₹502.0                         Success
2025-12-14 17:30:41  CHEQUE_CLEARED     Cheque 489123 for ₹502.0                         Success
```

```
=== REPORTS & ANALYTICS ===

Text Reports:
1.  Transaction History
2.  Bank Financial Summary
3.  Customer Balances Report
4.  Daily Transactions Summary
5.  Loan Portfolio Analysis
6.  View Audit Trail
7.  Customer Credit Score
8.  Database Summary (All Tables)

Visual Charts (Matplotlib):
9.  Account Distribution (Pie Chart)
10. Loan Portfolio (Bar Chart)
11. Monthly Transaction Trend
12. Customer Growth Chart
13. Balance Distribution Analysis
14. Transaction Types Breakdown
15. Loan EMI Analysis (4 Charts)
16. Daily Activity Timeline
17. Comprehensive Dashboard (6 Charts)

18. Back to Main Menu

Select Report: 7

--- Customer Credit Score ---
Enter Customer ID: CUST013

================================================
Customer: Ashsish Chanchalani
================================================
Credit Score: 708
Rating:       Good
================================================

Score Factors:
  Account Age:       0 days
  Total Transactions: 4
  Total Balance:     ₹7,499,498.00
  Loan Defaults:     0
```

```
Select Report: 8

--- Database Summary Export ---

                    COREBANK DATABASE SUMMARY


=== CUSTOMERS TABLE ===
Total Records: 13
CustomerID                Name      City      Phone Status
   CUST001      Raghav Agarwal     Delhi 9876543210 Active
   CUST002        Priya Sharma    Mumbai 8765432109 Active
   CUST003          Amit Kumar Bangalore 7654321098 Active
   CUST004          Neha Patel Hyderabad 6543210987 Active
   CUST005        Vikram Singh      Pune 5432109876 Active
   CUST006        Anjali Verma Ahmedabad 4321098765 Active
   CUST007         Rohan Desai     Surat 3210987654 Active
   CUST008          Isha Gupta    Jaipur 2109876543 Active
   CUST009          Arjun Nair     Kochi 1098765432 Active
   CUST010          Divya Iyer   Chennai 9988776655 Active
   CUST011                Goru     Ajmer 8989898989 Active
   CUST012      Manvink Khatri     Ajmer 3293283232 Active
   CUST013 Ashsish Chanchalani    Mumbai 9876543210 Active

=== ACCOUNTS TABLE ===
Total Records: 16
AccountNumber CustomerID   AccountType           Balance Status
      ACC1001    CUST001       Savings   ₹500,047,000.00 Active
      ACC1002    CUST001       Current       ₹230,000.00 Active
      ACC2001    CUST002       Savings       ₹320,000.00 Active
      ACC3001    CUST003       Savings       ₹180,000.00 Active
      ACC4001    CUST004       Current       ₹500,000.00 Active
      ACC5001    CUST005       Savings       ₹450,000.00 Active
      ACC6001    CUST006       Savings       ₹280,000.00 Active
      ACC7001    CUST007       Current       ₹620,000.00 Active
      ACC8001    CUST008       Savings       ₹340,000.00 Active
      ACC9001    CUST009       Savings       ₹200,000.00 Active
     ACC10001    CUST010       Current       ₹750,000.00 Active
     ACC10002    CUST001       Current       ₹345,000.00 Active
     ACC10003    CUST011 Fixed Deposit    ₹90,000,000.00 Active
     ACC10004    CUST012       Current ₹8,500,000,502.00 Active
     ACC10005    CUST001       Savings     ₹5,000,000.00 Active
     ACC10006    CUST013 Fixed Deposit     ₹7,499,498.00 Active

=== TRANSACTIONS TABLE ===
Total Records: 91
TransactionID AccountNumber TransactionType     Amount       Date
     TXN00001       ACC1001         Deposit ₹25,000.00 2025-11-19
     TXN00002       ACC1001         Deposit  ₹8,000.00 2025-11-29
     TXN00003       ACC1001      Withdrawal ₹10,000.00 2025-11-28
     TXN00004       ACC1001         Deposit  ₹5,000.00 2025-11-20
     TXN00005       ACC1001      Withdrawal  ₹8,000.00 2025-12-13
     TXN00006       ACC1002         Deposit ₹25,000.00 2025-12-12
     TXN00007       ACC1002      Withdrawal  ₹8,000.00 2025-09-20
     TXN00008       ACC1002      Withdrawal ₹10,000.00 2025-10-10
     TXN00009       ACC1002         Deposit  ₹5,000.00 2025-10-18
     TXN00010       ACC1002         Deposit ₹20,000.00 2025-10-09
     TXN00011       ACC1002      Withdrawal ₹30,000.00 2025-11-27
     TXN00012       ACC2001         Deposit  ₹8,000.00 2025-11-09
     TXN00013       ACC2001         Deposit  ₹5,000.00 2025-10-20
     TXN00014       ACC2001      Withdrawal ₹10,000.00 2025-11-17
     TXN00015       ACC2001      Withdrawal  ₹5,000.00 2025-11-12
     TXN00016       ACC2001         Deposit  ₹8,000.00 2025-09-21
     TXN00017       ACC2001         Deposit  ₹8,000.00 2025-09-29
     TXN00018       ACC3001         Deposit ₹20,000.00 2025-11-28
     TXN00019       ACC3001      Withdrawal  ₹5,000.00 2025-12-10
     TXN00020       ACC3001         Deposit ₹30,000.00 2025-10-14
  ... and 71 more records
```

=== LOANS TABLE ===
Total Records: 8

LoanID	CustomerID	LoanType	PrincipalAmount	EMI	OutstandingAmount	Status
LOAN001	CUST001	Personal Loan	₹500,000.00	₹8,886.00	₹350,000.00	Active
LOAN002	CUST002	Home Loan	₹2,500,000.00	₹25,000.00	₹2,300,000.00	Active
LOAN003	CUST003	Car Loan	₹800,000.00	₹12,500.00	₹600,000.00	Active
LOAN004	CUST005	Education Loan	₹1,000,000.00	₹14,500.00	₹850,000.00	Active
LOAN005	CUST007	Business Loan	₹3,000,000.00	₹45,000.00	₹2,500,000.00	Active
LOAN006	CUST004	Personal Loan	₹300,000.00	₹5,330.00	₹100,000.00	Active
LOAN007	CUST003	Education Loan	₹8,999.00	₹327.85	₹8,999.00	Active
LOAN008	CUST013	Home Loan	₹2,000,000.00	₹174,439.56	₹1,839,727.11	Active

=== CARDS TABLE ===
Total Records: 10

MaskedCard	CustomerID	CardType	Status
XXXX-6783	CUST001	Credit	Blocked
XXXX-9605	CUST002	Debit	Active
XXXX-7026	CUST003	Debit	Blocked
XXXX-7098	CUST004	Premium Credit	Active
XXXX-3748	CUST005	Debit	Active
XXXX-6733	CUST006	Premium Credit	Active
XXXX-2204	CUST007	Debit	Active
XXXX-2184	CUST008	Premium Credit	Blocked
XXXX-2077	CUST013	Debit	Blocked
XXXX-1288	CUST013	Credit	Active

=== CHEQUES TABLE ===
Total Records: 18

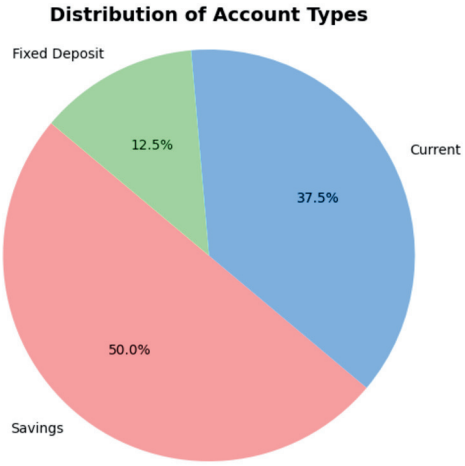
ChequeNumber	AccountNumber	IssuedTo	Amount	Status
835941	ACC1001	Payee 1	₹150,000.00	Cleared
310070	ACC1001	Payee 2	₹75,000.00	Bounced
283155	ACC2001	Payee 3	₹150,000.00	Cleared
720429	ACC2001	Payee 4	₹50,000.00	Bounced
280819	ACC2001	Payee 5	₹75,000.00	Bounced
768575	ACC3001	Payee 6	₹50,000.00	Issued
911880	ACC3001	Payee 7	₹75,000.00	Cleared
622024	ACC4001	Payee 8	₹50,000.00	Cleared
968793	ACC4001	Payee 9	₹50,000.00	Issued
269516	ACC4001	Payee 10	₹75,000.00	Issued
899550	ACC4001	Payee 11	₹100,000.00	Issued
981358	ACC5001	Payee 12	₹75,000.00	Bounced
840778	ACC5001	Payee 13	₹50,000.00	Issued
939746	ACC5001	Payee 14	₹200,000.00	Bounced
415547	ACC5001	Payee 15	₹50,000.00	Bounced
631773	ACC1002	Raghav	₹120,000.00	Cleared
811153	ACC10004	Raghav	₹500,000,000.00	Cleared
489123	ACC10006	Manvik	₹502.00	Cleared

=== TRANSFERS TABLE ===
Total Records: 12

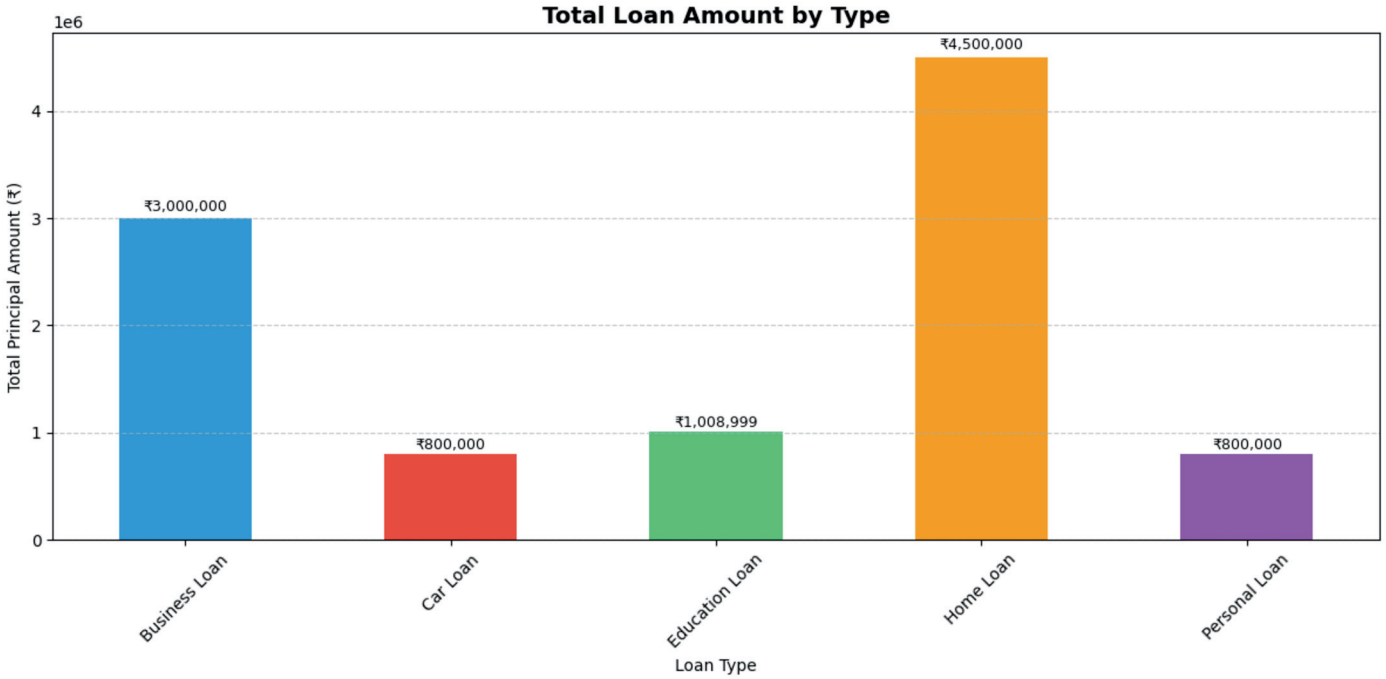
TransferID	FromAccount	ToAccount	Amount	Date	Status
TRF00001	ACC4001	ACC6001	₹20,000.00	2025-10-27	Success
TRF00002	ACC2001	ACC8001	₹30,000.00	2025-10-25	Success
TRF00003	ACC2001	ACC8001	₹30,000.00	2025-10-25	Success
TRF00004	ACC2001	ACC8001	₹10,000.00	2025-10-23	Success
TRF00005	ACC3001	ACC9001	₹30,000.00	2025-10-20	Success
TRF00006	ACC1001	ACC6001	₹10,000.00	2025-11-15	Success
TRF00007	ACC5001	ACC8001	₹50,000.00	2025-12-12	Success
TRF00008	ACC2001	ACC6001	₹10,000.00	2025-11-22	Success
TRF00009	ACC2001	ACC7001	₹10,000.00	2025-11-14	Success
TRF00010	ACC1001	ACC9001	₹10,000.00	2025-10-20	Success
XFER00011	ACC1001	ACC10002	₹45,000.00	2025-12-14	Success

```
=== AUDIT LOGS TABLE ===
Total Records: 114
    LogID         Action           Timestamp  Status
LOG000105  CHEQUE_ISSUED 2025-12-14 10:10:35 Success
LOG000106 CHEQUE_CLEARED 2025-12-14 10:11:28 Success
LOG000107  FUND_TRANSFER 2025-12-14 17:06:17 Success
LOG000108   LOAN_APPLIED 2025-12-14 17:10:18 Success
LOG000109       EMI_PAID 2025-12-14 17:11:43 Success
LOG000110    CARD_ISSUED 2025-12-14 17:21:34 Success
LOG000111    CARD_ISSUED 2025-12-14 17:24:55 Success
LOG000112   CARD_BLOCKED 2025-12-14 17:25:42 Success
LOG000113  CHEQUE_ISSUED 2025-12-14 17:29:20 Success
LOG000114 CHEQUE_CLEARED 2025-12-14 17:30:41 Success
  (Showing last 10 entries)

================================================================
DATABASE STATISTICS
================================================================

  Total Customers:     13
  Total Accounts:      16
  Total Transactions:  91
  Total Loans:         8
  Total Cards:         10
  Total Cheques:       18
  Total Transfers:     12
  Total Audit Logs:    114

  GRAND TOTAL: 375 records

=== REPORTS & ANALYTICS ===

Text Reports:
1.  Transaction History
2.  Bank Financial Summary
3.  Customer Balances Report
4.  Daily Transactions Summary
5.  Loan Portfolio Analysis
6.  View Audit Trail
7.  Customer Credit Score
8.  Database Summary (All Tables)

Visual Charts (Matplotlib):
9.  Account Distribution (Pie Chart)
10. Loan Portfolio (Bar Chart)
11. Monthly Transaction Trend
12. Customer Growth Chart
13. Balance Distribution Analysis
14. Transaction Types Breakdown
15. Loan EMI Analysis (4 Charts)
16. Daily Activity Timeline
17. Comprehensive Dashboard (6 Charts)

18. Back to Main Menu

Select Report:
```

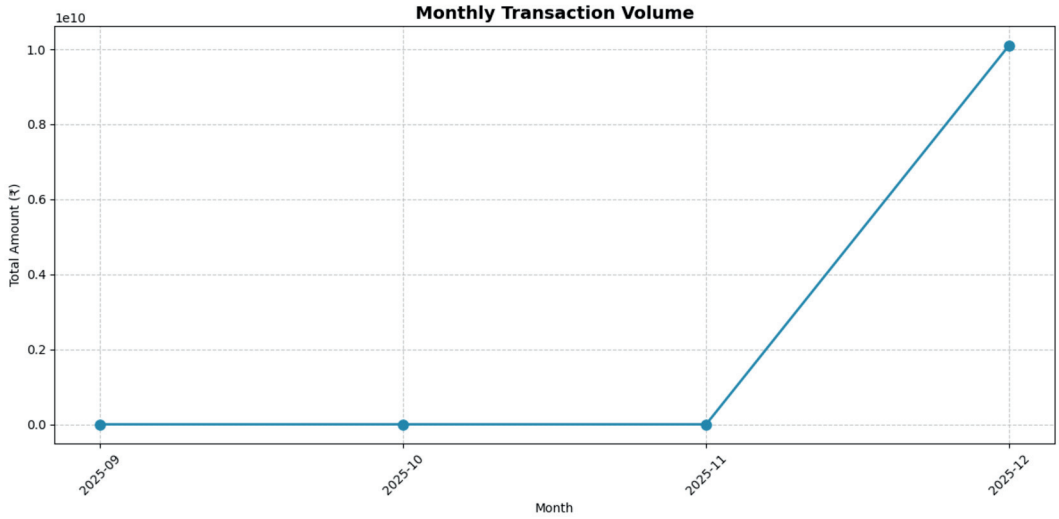
```
Select Report: 9

--- Generating Account Distribution Chart ---
Displaying chart window...
2025-12-14 18:04:19.678 Python[35112:1728942] The class 'NSSavePanel' overrides the method identifier.  This method is implemented by class 'NSWindow'
```



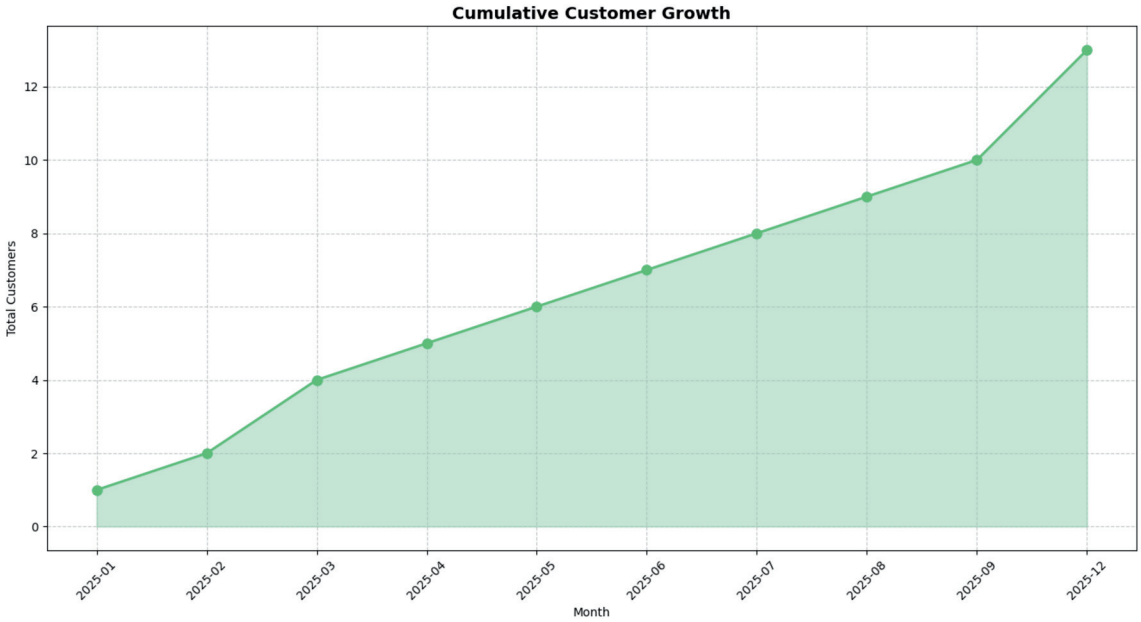

```
Select Report: 10

--- Generating Loan Status Chart ---
Displaying chart window...
```



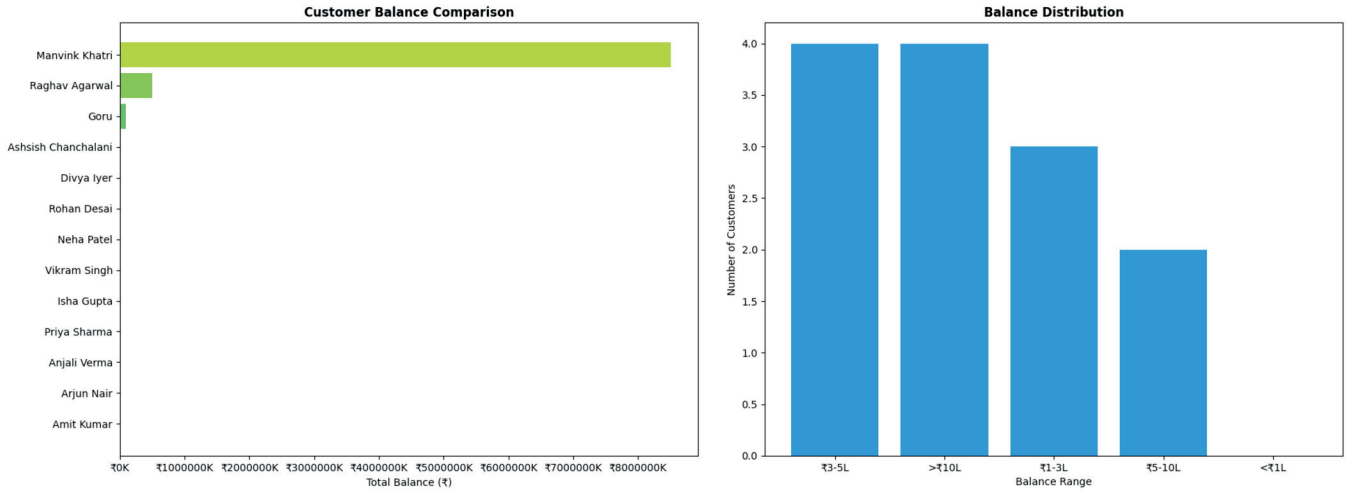

```
Select Report: 11

--- Generating Monthly Transaction Trend ---
Displaying chart window...
```



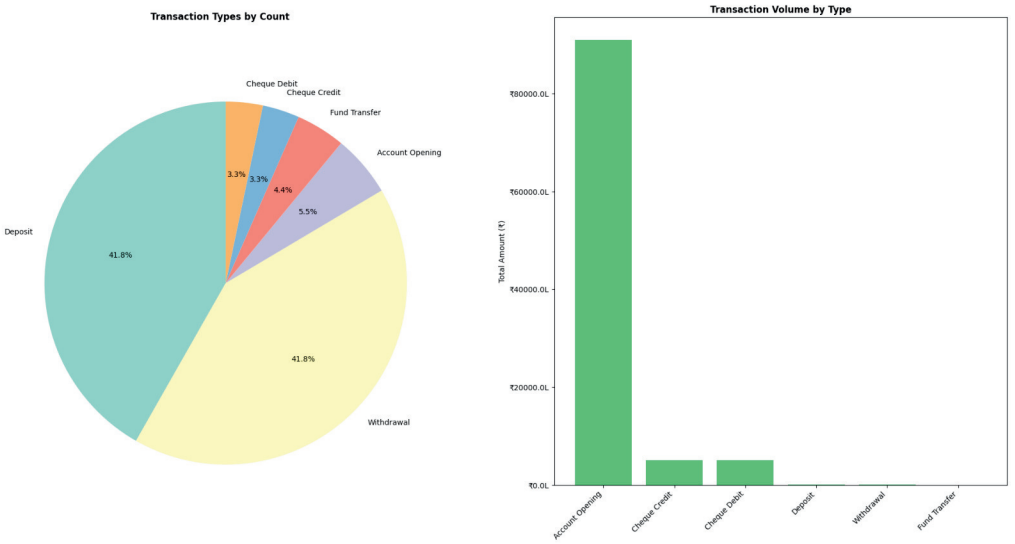
```
Select Report: 12

--- Generating Customer Growth Chart ---
Displaying chart window...
```



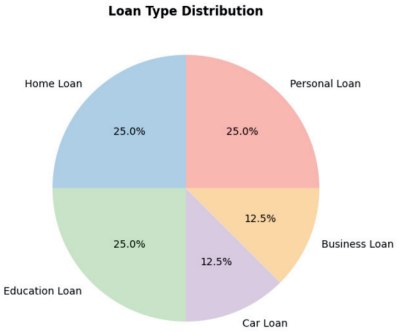

```
Select Report: 13

--- Balance Distribution Analysis ---
Displaying chart window...
```

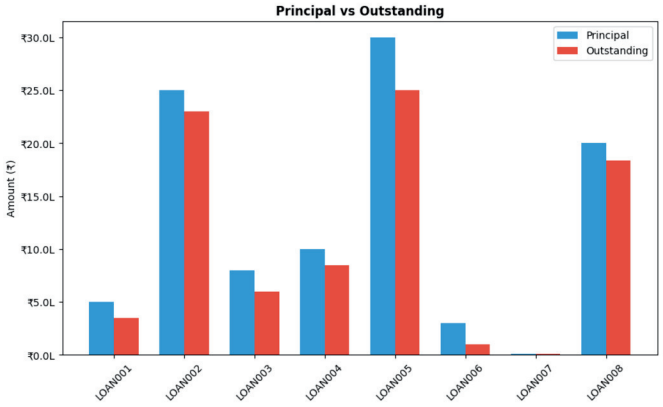


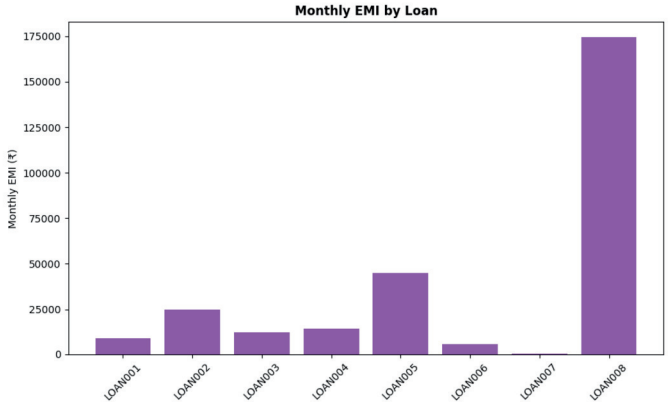

```
Select Report: 14

--- Transaction Types Analysis ---
Displaying chart window...
```

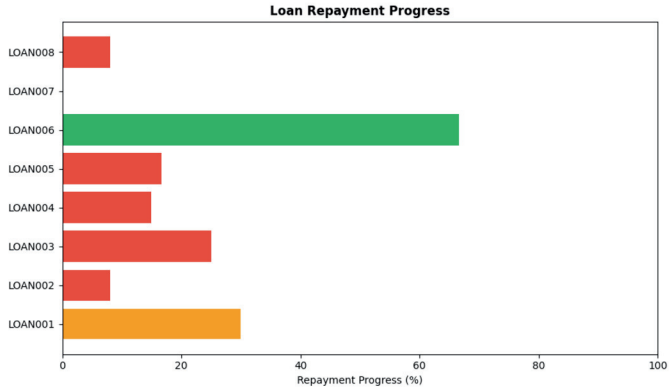


```
Select Report: 15

--- Loan EMI Analysis ---
Displaying chart window...
```

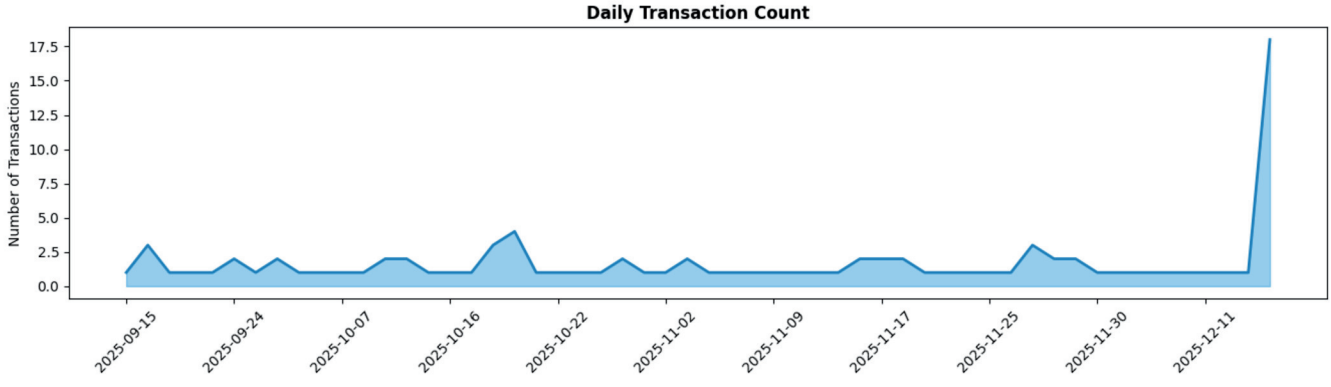


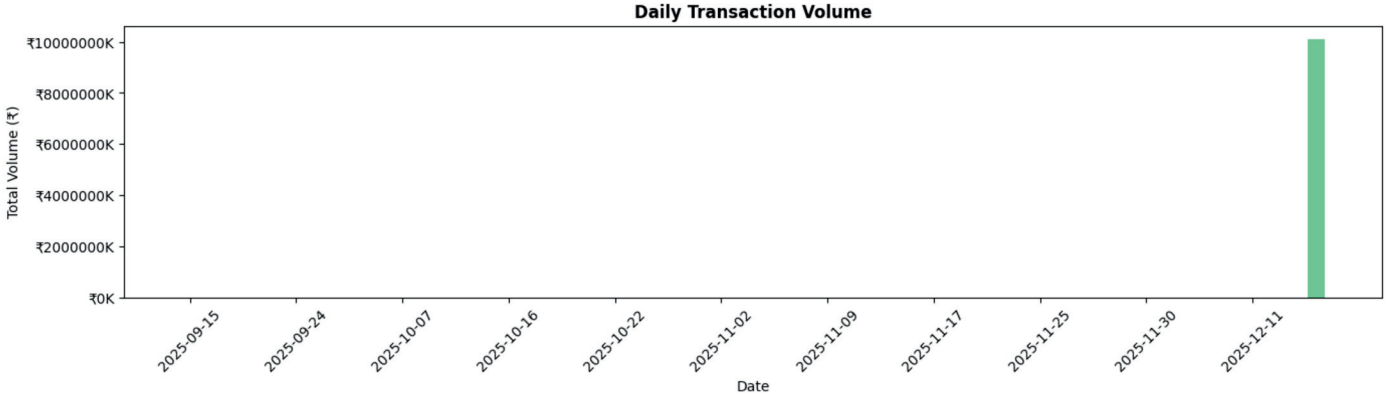







```
Select Report: 16

--- Daily Activity Analysis ---
Displaying chart window...
```





```
        COREBANK MANAGEMENT SYSTEM

Quick Stats:
  Customers: 13       Accounts: 16       Cards: 10
  Balance:   ₹9,106,762,000.00
  Loans:     8        Outstanding: ₹8,548,726.11
  Time:      2025-12-14 18:29:11
──────────────────────────────────────────────────

═══ MAIN MENU ═══

Customer & Account:
  1.  Add Customer
  2.  View Customers
  3.  Open Account
  4.  Check Balance

Transactions:
  5.  Deposit Money
  6.  Withdraw Money
  7.  Fund Transfer

Loan Management:
  8.  Loan Menu

Card & Cheque:
  9.  Card Management
  10. Cheque Processing

Advanced Features:
  11. Account Statement
  12. Interest Calculator
  13. Financial Dashboard
  14. Compare Loan Offers

Reports & Utilities:
  15. Reports & Analytics
  16. Backup Data
  17. Search Customer

  18. Exit

Select Option: 16

--- Creating Data Backup ---
✓ Backup created: backups/bank_database_20251214_183216.csv
```

LIMITATIONS OF PROJECT AND FUTURE IMPLEMENTATION

CURRENT LIMITATIONS:

• **CLI Only –** Text-based terminal interface without GUI
• **Single User –** No concurrent user support
• **CSV-Based Database –** Limited scalability for large datasets
• **No Encryption –** Passwords hashed but not encrypted at rest
• **No Network –** Local operations only, no online/mobile access
• **No Multi-Branch –** Single bank branch support only
• **Limited Reporting –** Basic analytics without advanced BI tools
• **No Integration –** Standalone system without third-party integration

FUTURE ENHANCEMENTS:•

Phase 1: Graphical Interface – Develop Tkinter GUI for better UX + Create web interface using Flask/Django
• **Phase 2: Database Migration –** Migrate to MySQL for scalability
• **Phase 3: Advanced Features –** Multi-branch, fraud detection, ML scoring
• **Phase 4: Integration –** API development for third-party integration
• **Phase 5: Cloud Deployment –** Deploy on cloud platforms

BIBLIOGRAPHY

1. Python Software Foundation. (2024). Python 3.x Documentation. Retrieved from https://docs.python.org/

2. Pandas Development Team. (2024). Pandas – Python Data Analysis Library. Retrieved from https://pandas.pydata.org/

3. Matplotlib Development Team. (2024). Matplotlib – Visualization with Python. Retrieved from https://matplotlib.org/

4. Real Python. (2024). Python Tutorials – Complete Guide. Retrieved from https://realpython.com/

6. GeeksforGeeks. (2024). Python Data Structures and Algorithms. Retrieved from https://www.geeksforgeeks.org/

7. Stack Overflow Community. (2024). Python Programming Q&A. Retrieved from https://stackoverflow.com/questions/tagged/python

8. GitHub. (2024). Python Open Source Projects. Retrieved from https://github.com/

9. Tutorialspoint. (2024). Python & Finance Learning. Retrieved from https://www.tutorialspoint.com/