

# **CIS 8040-Database Management**

**Assignment 1: Design and Implementation of a Relational Database**

**Team 2: Gaurav Mishra, Kelly Duong, Manoj Potnuru, Md Kamruzzaman Kamrul, Purvi Bharani**

**Question 1:** Select an application for which a database management system is needed. Describe the application and justify why it is an important application from a management perspective.

**Ans:**

**Application Description:**

The Bank Account Management System is a comprehensive application designed to facilitate a wide range of banking transactions for customers. This system enables customers to create and manage accounts, perform financial transactions like deposits and withdrawals, and access account reports. It can be accessed through various channels such as ATMs, telephone banking, online banking via computers, and mobile banking through smartphones.

**Importance from a Management Perspective**

* **Operational Efficiency:** Automates routine banking transactions, reducing the need for manual intervention and thereby increasing operational efficiency.
* **Customer Service Improvement:** Provides 24/7 banking facilities, enhancing customer convenience and satisfaction.
* **Financial Tracking and Reporting:** Helps in monitoring transactions and generating reports, which are vital for financial analysis and decision-making.
* **Security and Compliance:** Ensures adherence to financial regulations and enhances security measures to protect customer data and prevent fraud.
* **Market Competitiveness:** Offers a competitive edge in the banking industry by providing advanced and user-friendly banking solutions.

**Question 2:** State three important business rules that the database needs to be able to support. Explain how the database will support these business rules. An example is: employees must be designated as being either full-time or part-time employees. Note: Stating mapping ratios or min/max cardinalities is not a business rule.

**Ans:**

1. **Transaction Limits with Managerial Approval**

**Rule:** Transactions exceeding a certain amount require manager approval.

**Database Support:** Implement triggers to flag transactions above the threshold. These triggers can send automated alerts to managers for approval, ensuring compliance and oversight for large transactions.

1. **Account Security Through Locking Mechanisms**

**Rule:** Accounts with multiple failed login attempts are temporarily locked.

**Database Support:** The system tracks login attempts and utilizes a security protocol to lock accounts after a specified number of failed attempts. This information is logged and managed within the database to enhance security measures against unauthorized access.

1. **Restricted Access to Account Information**

**Rule:** Only authorized users can access and modify account information.

**Database Support:** Implement robust user authentication systems within the database, such as storing encrypted login credentials. The system could also incorporate multi- factor authentication processes, ensuring that only authorized personnel have access to sensitive account information.

1. **Accurate and Timely Record-Keeping**

**Rule:** All transactions must be recorded in real-time with time-stamped entries.

**Database Support:** The database system should be configured to automatically log every transaction with a timestamp. This ensures accurate historical data and aids in audits and compliance checks.

1. **Compliance with Regulatory Standards**

Rule: The system must comply with financial regulatory standards, including data privacy laws and anti-money laundering protocols.

**Database Support:** Ensure that the database management system is compliant with relevant laws and regulations. This includes data encryption, regular audits, and implementing protocols for anti-money laundering checks.

**Question 3:** Create a conceptual model for this application. Include proper names for entities, attributes, and relationships. Identify min/max cardinalities. You may use either the Chen or Crow’s Feet representation. Use the notation presented in class. There should be 5-8 entities in the conceptual model.

**Ans:**

**Conceptual Model Overflow:**

**Entities and Attributes:**

* Customer
  + Attributes: CustomerID, Name, Address, Phone, Email
  + Relationships: The Customer is optional to have an account or more accounts and undergo Authentication.
* Account
  + Attributes: AccountID, Balance, DateOpened, TransactionLimit
  + Relationships: Owned by Customer, has Transactions, has Branch.
* Branch
  + Attributes: BranchID, Address, Phone.
  + Relationships: Services type at the branch, related to Transaction.
* Transaction
  + Attributes: TransactionID, Amount, Date, Type
  + Relationships: Occurs in an Account, and may require Manager Approval
* Manager
  + Attributes: ManagerID, Name
  + Relationships: Approves high-value Transactions
* Authentication
  + Attributes: CustomerID, AuthenticationMethod, LastLogin, FailedAttempts
  + Relationships: Validates Customer identity, linked to Customer

**Relationships and Cardinalities:**

* Customer-Account
  + A customer is optional to own an account or have multiple accounts (0, N).
  + But many Accounts are owned by one customer (1, N)
* Account-Branch
  + An account can have one branch specific for service (1,1).
  + Many branches may or do not have accounts (0, N).
* Branch-Manager
* Branches must have one manager (1,1).
  + One manager working at one branch (1,1).
* Manager-Transaction
  + A transaction may require zero or one manager's approval (0,1).
  + A manager may not need to approve or can approve multiple transactions (0, N).
* Customer-Authentication
  + Each customer has one authentication record (1,1).
  + Authentication is specific to one customer (1,1).

**Diagram Representation (Crow’s Foot Notation):**

*Link:* **Lucid.app**

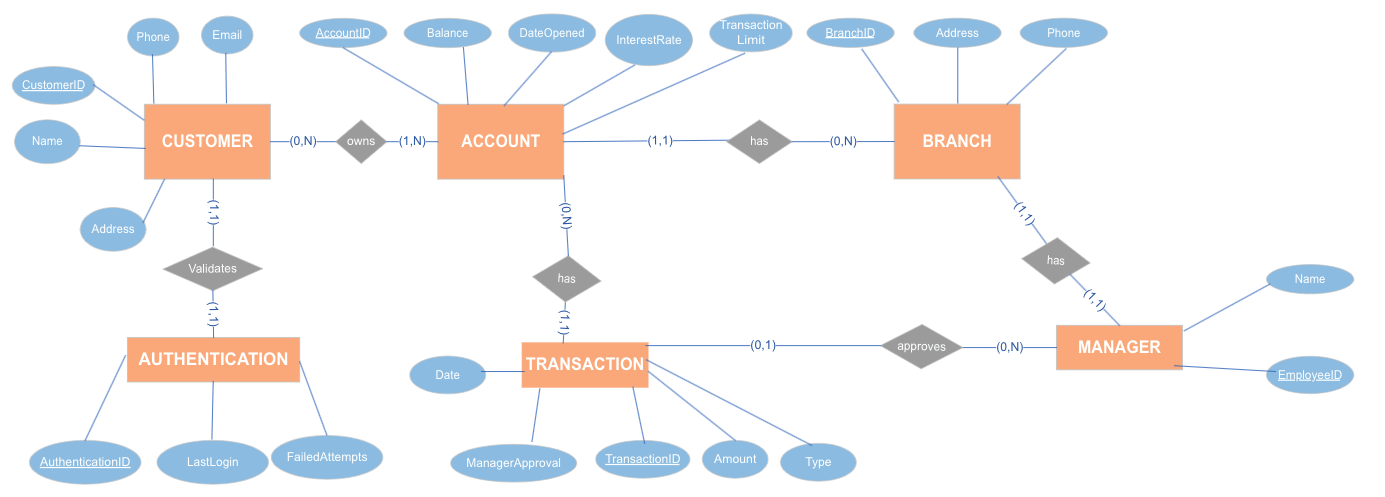
**https://lucid.app/lucidchart/a4af08ec-4321-4507-84c6-7654cbc209b1/edit?invitationId=inv\_22e5fe0e-54de-4fc0-b0bc-801ef7a94bd5**

A diagram of a bank account

Description automatically generated

**Diagram Representation (Chen Notation)**

*Link:*[**Chen's Notation**](https://studentgsu-my.sharepoint.com/:u:/g/personal/mpotnuru1_student_gsu_edu/ETdxPqDno4JFi7zgezQsouYBHGv8Q4osG5DqqK7kzUiLQA?e=wiF893)



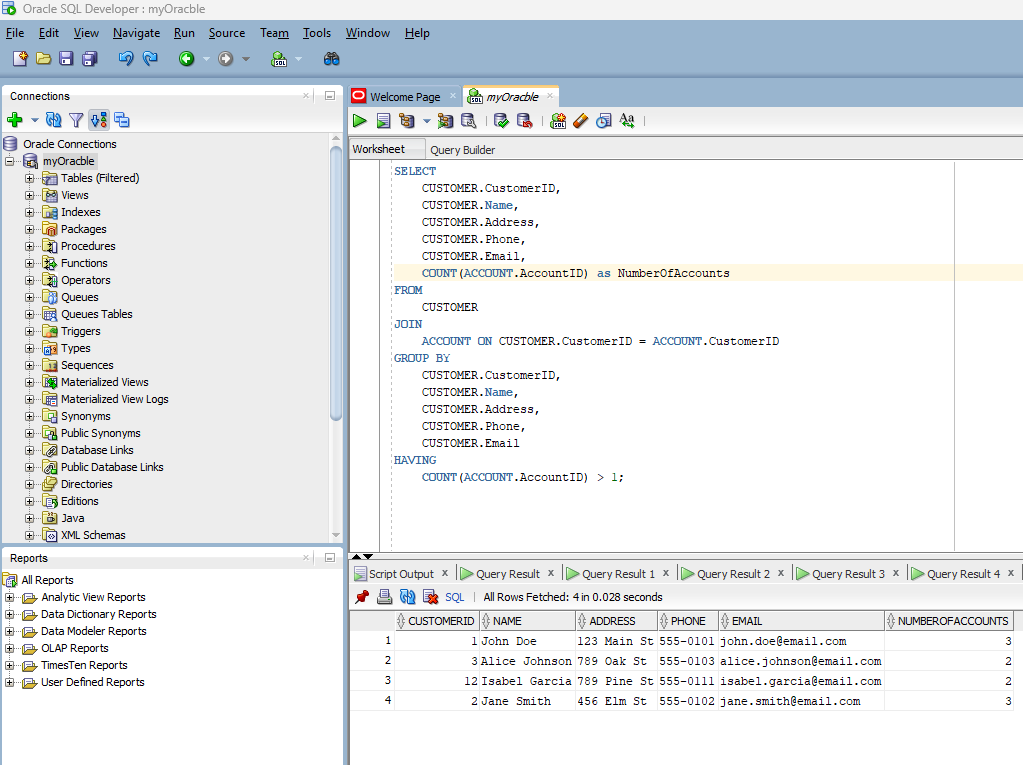
**Question 4:** List 5 non-trivial queries that you would want to run against a populated database. For each query, justify why it would be important for the operations of a company.

**Ans:**

**Query 1 - Identify Customers with Multiple Accounts:**

**Description:** This query retrieves a list of customers who have multiple accounts. It joins the CUSTOMER and ACCOUNT tables, counting the number of accounts each customer has. Customers with more than one account are included in the result.

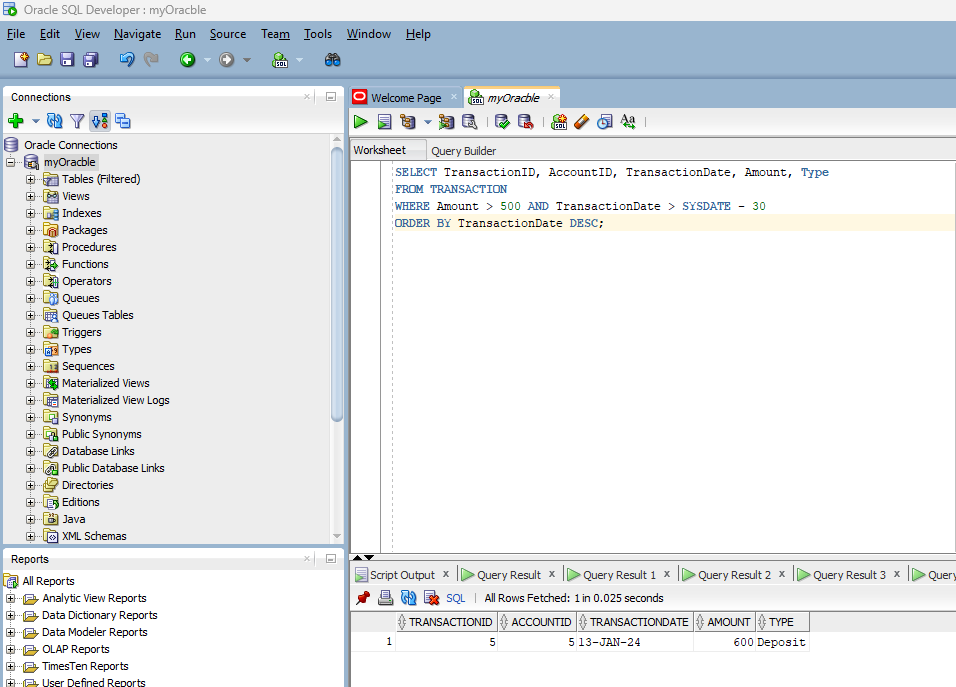
**Justification:** This query is useful for customer relationship management. Identifying customers with multiple accounts can help the company tailor its services, promotions, and communication to better serve these valuable customers. It also allows for a better understanding of customer behavior and preferences.



**Query 2 - Transaction above five hundred**

**Description:** This query retrieves transactions with an amount greater than five hundred and which have occurred within the last 30 days. It selects TransactionID, AccountID, TransactionDate, Amount, and Type from the TRANSACTION table, filtering based on the specified criteria.

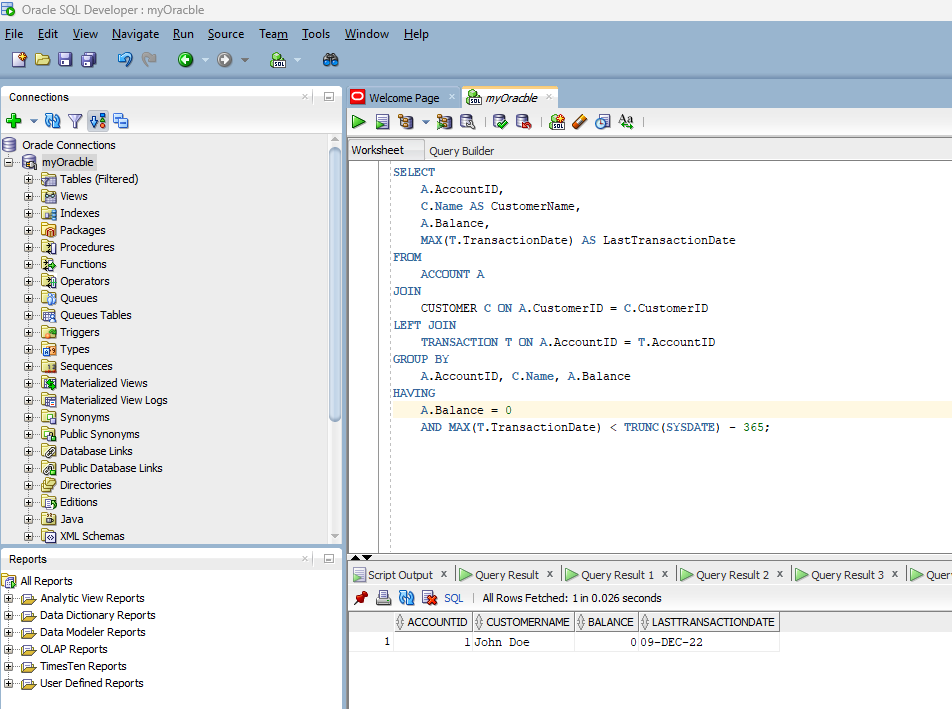
**Justification:** This query is important for monitoring high-value transactions. It helps the company identify significant financial activities within a recent time frame. Detecting large transactions can be crucial for fraud prevention and financial analysis.



**Query 3: List Inactive Accounts**

**Description:** This query lists accounts that are considered inactive. It identifies accounts with a balance of zero and no transactions within the last 365 days. It includes AccountID, CustomerName, Balance, and the last transaction date.

**Justification:** Identifying inactive accounts is essential for managing account resources efficiently. It allows the company to identify accounts that may need attention, such as closure or reactivation. This query helps in maintaining a clean and updated account database.

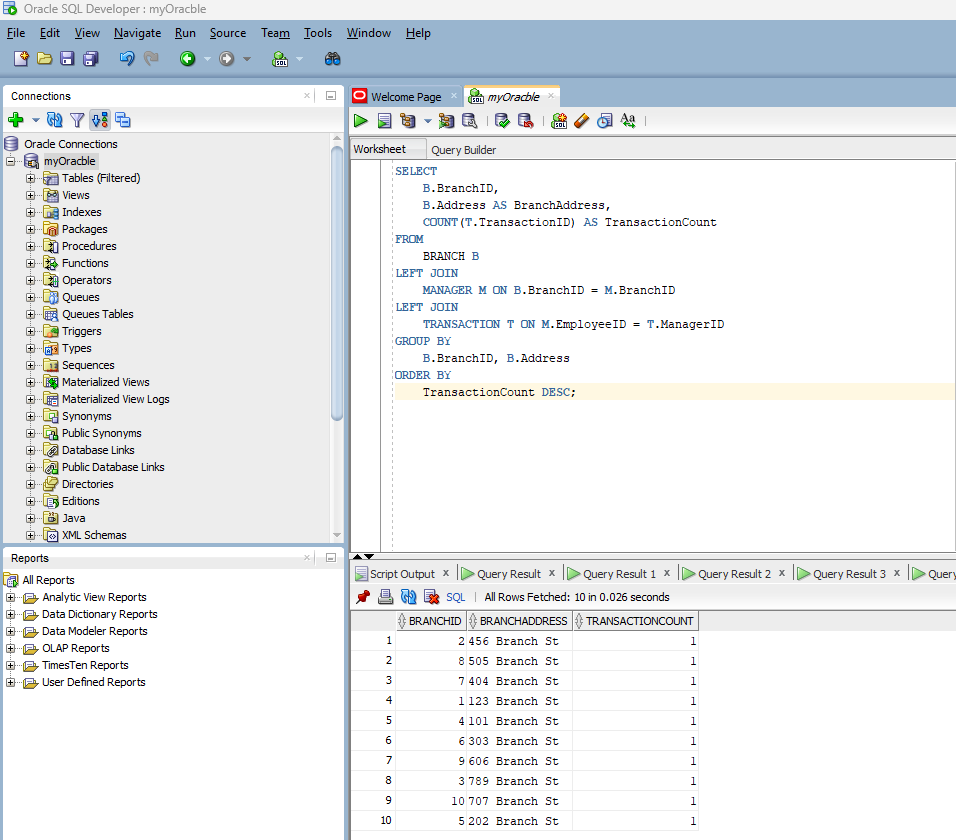


**Justification:** This ensures that transactions awaiting managerial approval are processed in a timely manner, maintaining operational efficiency and customer satisfaction. It also helps managers prioritize their tasks.

**Query 4 - Determine Branch Activity Level**

**Description:** This query determines the activity level of branches by counting the number of transactions associated with each branch. It retrieves BranchID, BranchAddress, and TransactionCount, ordering the results by transaction count in descending order.

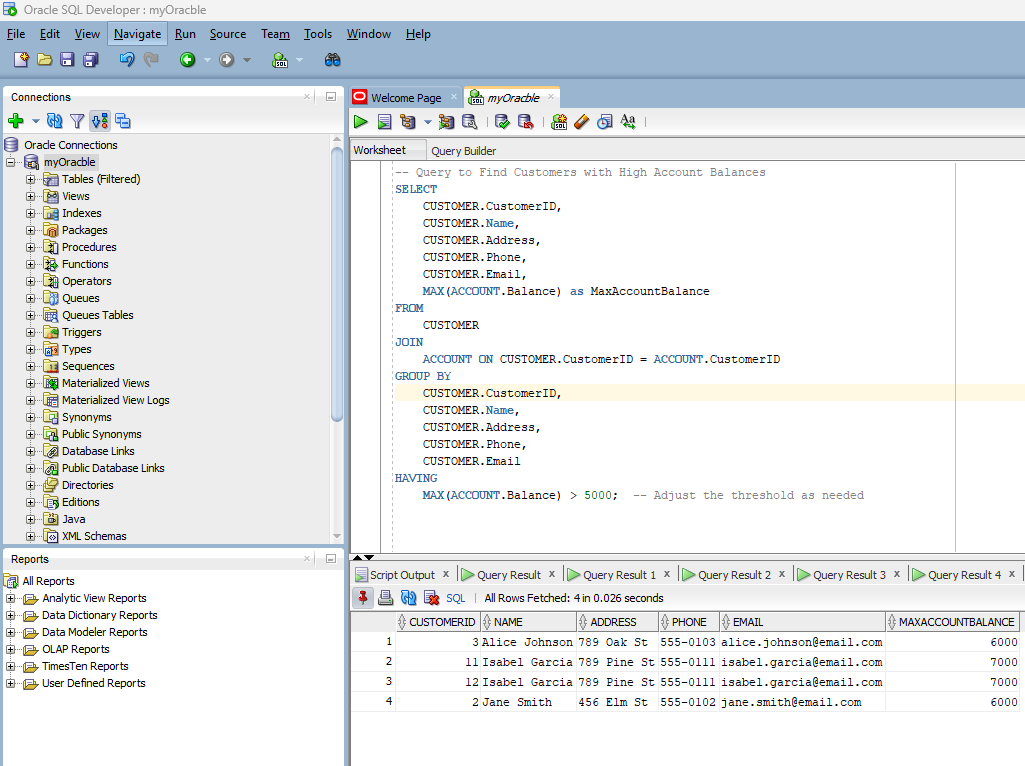
**Justification:** Branch activity level is vital for assessing the performance of different branches. It helps the company allocate resources, identify high-performing branches, and improve customer service. This query provides insights into branch operations.



**Query 5:** Query to Find Customers with High Account Balances

**Description:** This query identifies customers with high account balances by joining the CUSTOMER and ACCOUNT tables, calculating their maximum account balance, and filtering for those exceeding a specified threshold, set at five thousand in the HAVING clause.

**Justification:** Valuable for tailoring services, marketing, and financial advice to high-balance customers, potentially increasing revenue. Aids in risk assessment and evaluating customer financial health. Threshold in HAVING clause adjustable for business goals and segmentation.



The queries can be found in the following link –

https://colab.research.google.com/drive/1l-dwA82qNIfupYu32bWPmpYUHCasQtja?usp=sharing

**Question 5:** For each entity, create a corresponding entity relation. Implement the entity relations in Oracle using your Oracle account provided for this course. Populate the relations with data. You will need to make up the data, which should be a reasonable reflection of the data that could occur in such an application. Show the populated relations. You can do this by using the “select \* command. There should be at least 5-10 entries for each relation.

**Ans:**

In response to the assignment's instruction to "For each entity, create a corresponding entity relation," we have defined tables in the Oracle database to represent each entity within the database schema. Below is an overview of the tables (entity relations) created:

1. CUSTOMER Table:
   * Represents the CUSTOMER entity.
   * Contains columns such as CustomerID, Name, Address, Phone, and Email.
   * CustomerID is set as the primary key.
2. BRANCH Table:
   * Represents the BRANCH entity.
   * Contains columns like BranchID, Address, and Phone.
   * BranchID is set as the primary key.
3. ACCOUNT Table:
   * Represents the ACCOUNT entity.
   * Contains columns including AccountID, CustomerID, BranchID, Balance, DateOpened, InterestRate, and TransactionLimit.
   * CustomerID and BranchID are foreign keys that reference the CUSTOMER and BRANCH tables.
   * AccountID is set as the primary key.
4. MANAGER Table:
   * Represents the MANAGER entity.
   * Includes columns like EmployeeID, Name, and BranchID.
   * BranchID is a foreign key referencing the BRANCH table.
   * EmployeeID is set as the primary key.
5. TRANSACTION Table:
   * Represents the TRANSACTION entity.
   * Consists of columns such as TransactionID, AccountID, ManagerID, TransactionDate, Amount, and Type.
   * AccountID and ManagerID are foreign keys referring to the ACCOUNT and MANAGER tables, respectively.
   * TransactionID is set as the primary key.
6. AUTHENTICATION Table:
   * Represents the AUTHENTICATION entity.
   * Contains columns like AuthenticationID, CustomerID, LastLogin, and FailedAttempts.
   * CustomerID is a foreign key linking to the CUSTOMER table.
   * AuthenticationID is set as the primary key.

**Populating Data:**

1. **Creating & Populating Data in Customer Table**

**A screenshot of a computer

Description automatically generated**

1. **Creating & Populating Data in Branch Table**

A screenshot of a computer

Description automatically generated

1. **Creating & Populating Data in Account Table**

A screenshot of a computer

Description automatically generated

1. **Creating & Populating Data in Manager Table**

A screenshot of a computer

Description automatically generated

1. **Creating & Populating Data in Transaction Table**

A screenshot of a computer

Description automatically generated

1. **Creating & Populating Data in Authentication Table**

**A screenshot of a computer

Description automatically generated**

**Question 6:** Describe how you would expand your conceptual model if you were to implement it for a real-world application. Identify what additional constructs (entities, relationships, attributes) you would add and the purpose of each. Based on the expanded model, what additional queries would you be able to answer?

**Ans:**

We can add additional features like Loan, Investment, ATM for a banking application as entities, explain the purpose, and discuss additional queries.

**Additional Constructs for Expansion:**

* **Loan**
  + **Attributes:** LoanID, Amount, InterestRate, StartDate, EndDate, CustomerID, LoanTypeID
  + **Purpose:** To manage customer loans, track loan amounts, interest rates, and repayment schedules.
  + **Additional Queries:**
    - Find customers with outstanding loans above a certain amount.
    - Generate reports on loan repayments due within the next month.
* **Investment**
  + **Attributes:** InvestmentID, Type, Amount, Date, CustomerID, RiskLevel
  + **Purpose:** To track customer investments, types of investments (stocks, bonds, etc.), and associated risks.
  + **Additional Queries:**
    - Identify high-risk investments made by customers.
    - Calculate the total investment value per customer.
* **ATM**
  + **Attributes:** ATMID, Location, Status (Active/Out of Service), CashAvailable
  + **Purpose:** To manage ATM locations and statuses, including cash availability.
  + **Additional Queries:**
    - Locate ATMs that are running low on cash.
    - Find ATMs in a specific geographic area.
* **Loan Type**
  + **Attributes:** LoanTypeID, Description, TypicalTerm, TypicalInterestRate
  + **Purpose:** To categorize distinct types of loans (e.g., personal, mortgage, auto) and their typical terms.
  + **Additional Queries:**
    - Determine the most popular loan type based on the number of loans issued.
* **Account-Loan Relationship**
  + **Attributes:** AccountID, LoanID
  + **Purpose:** To link customer accounts to their loans for easy tracking of loan payments and account deductions.
  + **Additional Queries:**
    - List all loans linked to a specific account.
* **Customer-Investment Relationship**
  + **Attributes:** CustomerID, InvestmentID
  + **Purpose:** To associate customers with their investments.
  + **Additional Queries:**
    - Summarize total investment values by customer.
* **ATM-Transaction**
  + **Attributes:** ATMID, TransactionID
  + **Purpose:** To track which transactions are made at which ATMs.
  + **Additional Queries:**
    - Analyze the frequency of transactions at each ATM.

### **Impact of Expansion on Queries**

The expansion of the conceptual model allows for a more comprehensive analysis of banking operations. For instance, with the addition of loans and investments, the bank can better understand its financial exposure and customer investment behavior. The integration of ATM data aids in operational management and customer service improvements.

These additional entities and relationships provide a deeper insight into the banking ecosystem, allowing for more sophisticated queries that can inform decision-making, risk management, customer relationship management, and strategic planning. The ability to analyze diverse aspects of banking operations holistically is crucial for a modern, dynamic financial institution.