Blockchain Finance

Database Final Project
COSC 641 Part 1

You are employed to work on an existing project. This new financial organization (bank) is based on Bitcoin. The base tables and some data are already provided for you. The ER diagram and structure of tables are in a supplemental document.

Please number your chapters correctly. I grade one chapter at a time.

The data and based tables are under PROJECT2020 account. You can access each table by using PROJECT2020.table_name. You can copy each table into your Oracle account like:

```
CREATE TABLE branch
AS
SELECT      *
FROM        PROJECT2020.branch;
```

Check your data:
```
SELECT *
FROM branch;
```

Check the attributes:
```
DESC branch;
```

Note 1: Frequently it is required to make changes to the base tables (add or delete fields). You can use ALTER TABLE. To add/delete fields to your tables. The added fields will have value of NULL for now.
Note 2: All of the data must be in your based tables. We will not create FORCED view.
Note 3: For complex queries you can create intermediate views and create view from view.
Note 4: We will use the views and sequences later on in the project. For now, just create them.
Note 4: The data in the tables are test data. By no means they are complete. You welcome to add additional records to the tables.

Please try to keep the names tables as close to the original names as you can. If there is any missing data, feel free to add it to your tables.

List of tables are:

1. BRANCH
2. DEPARTMENT
3. CAR
4. DRIVER (emp drivers' car)
5. CUSTOMER
6. BANK_EMPLOYEE
7. JOB
8. EMP_ANNUAL_DATA
9. BRANCH_EMPLOYEE (emp work at branch)
10. ATM
11. BRANCH_ACCESS_POINTS (branch has access points)
12. BRANCH_MANAGER (emp manager branch)
13. LOAN
14. LOAN_PROJECT

15. DEPOSIT_ACC
16. DEPOSIT_ACC_PRODUCT
17. CD_ACCOUNT
18. CD_PRODUCT
19. CREDIT_ACCOUNT
20. CREDIT_PRODUCT
21. LOAN_PAYMENT
22. DEPOSIT_ACCT_TRANSACTION
23. CREDIT_ACCT_TRANSACTION

Chapter 1:
Create the following views: Please show you **code** (query) and **result** of running your code. Such as:

1a:
```
SELECT  *
FROM   EMPLOYEE_DATA;
```

a. Employee_data with the following attributes:

Name of Employee (first, middle, last)
Address
Zip code of Employee Address
SSN
Title
Current Year
Current Yearly Salary
Current Tax Deduction Rate
The date s/he was employed at the Current Branch
Birth Date
Age of Employee
Employee Branch Phone Extension s/he Works at
Branch Phone Number
Branch Name (Employee Works at)
Highest Degree
Highest Degree date

b. Employee_salary with the following attributes:

Name
Current Year
SSN
Current Salary
Branch Employee Works at
Total Cost of Employee Salaries at the branch s/he works
Highest salary at his/her branch
Average salary at his/her branch

c. Branch_data with the following attributes:

Branch ID
Branch Name

Address
Phone Number
Fax Number to be inputted later
Number of Employee at this Branch
Category
Manager Name
Total Transactions Done at this Branch for year 2020

d. Valued_Customers with the following attributes:
Describe how you select the important customers in this bank.

SSN
Name
Age
Home Phone
Work Phone
Address
Zip Code
Email to be inputted later
State they live in
Total number of transactions the customer has done in a given year
Total amount (in bitcoin) of transactions the customer has done in a given year (you choose the year)

e. Statistics_by_Branch with the following attributes (Read only view):

Branch Id
Branch name
Year
Total deposit in that year for this location
Total number of transactions
Total number of employees at this branch

f, g-Create two more views that can be used by **customers**. (Make sure it is useful to the customers. You will be graded based on the usefulness of the views)

h, i-Create two more views that can be used by **management**. (Make sure it is useful for managerial decisions)

Chapter 2:

a- Create a sequence called ID_generator to be used for Account ID.
Start with 1111
Generate only odd numbers for security
Cache 50 numbers at a time

b- Create a sequence to be used for the Transaction ID. (Make your own assumption).

Chapter 3:

Write Subprograms with exception handling: (make sure your subprograms have appropriate exception handling).

A. Procedure to transfer $x from one account to another account. You will pass the amount of transfer, and two account numbers. Also, write the FROM account number, to TO account number, the amount and the date of transaction into a Transaction_Log file (Create this table). Show your work. You may use CREDIT_ACCOUNT table for this procedure.

B. Create a subprogram called **Birthday_sub** that accepts today's date as default and writes the first name, last name, and address of a customer whose birth date (day and month only) is 15 days from today's date. Write them into a file (create a table called B_C_File). You may add additional records to your customer table.

C. Procedure to write all daily (today) transactions (credits and debits) into a database table called **Today_Transaction.** (all deposits and withdraw for that day). You may use CREDIT_ACCT_TRANSACTION table

   a. Use SYSDATE to get today's date.
   b. The structure of Today_Transaction is:
      i. (Date & Time, Account number, Account Type, Amount, deposit/withdraw)

D. Create a function called CustomerInfo to accept a customer account number and return the total deposit for the customer account.
   IN : CUSTOMER ACCOUNT NUMBER
   OUT : TOTAL DEPOSIT

E. Create a function with the same name CustomerInfo (overload) to accept a customer account number and a date; and return the total deposit of that customer account for that date.
   IN : CUSTOMER ACCOUNT NUMBER, DATE
   OUT : TOTAL DEPOSIT on supplied DATE

F. Create a function with the same name CustomerInfo (overload) to accept a customer ID, a date and a co-owner account number, and return the total deposit of that joint account for that date.
   => NOTE: Assumed Customer Id = Primary
   IN : CUSTOMER ID, DATE, COOWNER ACCOUNT NUMBER
   OUT : TOTAL DEPOSIT on supplied DATE

G. Procedure to list the last 10 transactions of a customer by passing Customer Id.
   => NOTE: Assumed Customer Id = Primary
   IN : CUSTOMER ID
   OUT: LAST 10 TRANSACTIONS

H. Create a subprogram to accept a customer ID and output name (first, mid, and last), loan no for this customer, and amount of loan. (a customer may have more than one loan.
   => NOTE: Assumed Customer Id = Primary
   IN : CUSTOMER ID
   OUT : CUSTOMER NAME (first, mid, and last), LOAN NUMBER, LOAN AMOUNT

I. Create a subprogram to write the following information for each employee into a database called Emp_list only for employees that older than 30 years old. Name (first, mid, and last), address, date of birth, and their salary.

Chapter 4:

Create the following packages with exception handling: Please show your code.

A. Create a package with the following functions called **BankP** for each customer (customer id):
   a. Function to return the current balance.
   b. Function to return the last deposit from checking.
   c. Function to return the last deposit from saving
   d. Function to return the last withdraw from checking.
   e. Function to return the last withdraw from saving.

B. Create a package with the following functions & procedures called **BranchP** by branch id.
   a. Function to return the current branch address.
   b. Function to return the current branch phone number.
   c. Procedure to output the name of employees working at that branch.

C. Create a package called **Insert_pkg** with subprograms to automate the insert, delete, and update of data in your database.
   a. Subprogram to insert a row in table customer (make sure you use the sequence for generating ID's)
   b. Subprogram to insert a row in table bank employee
   c. Subprogram to insert a row in table CD
   d. Subprogram to insert a row in table driver.
   e. Subprogram to delete a row from table driver.
   f. Subprogram to update a row in table driver.

D. Create a useful package of your choice with functions, procedures, and datatype of your own.

Chapter 5:

Create the following triggers:

a. Any deletion from employee file, trigger to write OLD attributes into an **Employee_History** file.
b. Any modification to the customer accounts, write who, date, and the nature of (OLD and NEW) modification into a **Cust_Mod_Log** file.
c. Any deposit larger than $5000 to any account will also be written in a **Large_Dep_Log** file.
d. Any withdraw larger than $10000 from any account also will be written in a **Large_With_Log** file.