COMP4003 Assignment #5

JDBC & PL/SQL

Due: December 7

Instruction

1. You should do the assignments independently. Copying is not allowed.
2. Submit your assignment as a single word/PDF document on culearn.
3. Put the programs in Parts 1, 2, 3 into the file to be submitted.
4. Put the screenshots of the execution in Parts 4 into the file to be submitted.
5. The total mark is 106.

Part 1 SQL DDL (10)

Use Oracle VM to create three tables for a bank application: customer, branch, and account with primary keys, foreign key, and other integrity constraints properly defined.

1. A customer has a 5 digit customer number starting from '00000'.
2. A branch has a 3 digit branch number starting from '000', and an address.
3. An account has a 7 digit account number that includes 3 digit branch number plus 4 digit local account number starting from '0000000', a customer number, and a balance which must never be negative.
4. Customer names and bank addresses are unique and cannot be null.
5. A branch or customer cannot be deleted if the customer has an account in the branch.
6. A customer can only have one account in a branch but can have account in several branches.

Part 2 PL/SQL (30)

Create a PL/SQL package *bank* that contains the following subprograms.

1. branch(address): return the branch number if there is one there; otherwise, return null.
2. open_branch(address): first check if there is a branch there and raise an exception if so; otherwise check if there is a branch number unused (closed branch) and just uses the first unused number if so; otherwise, generate a branch number that is the highest branch number plus one and create this branch and return the new branch number.
3. close_branch(address): delete the branch if it exists; otherwise, raise an exception.
4. create_customer(name): first check it the customer is there and raise an exception if so; otherwise, obtain the highest customer number, increase it by 1, and use it as the customer number.
5. remove_customer(name): first check if the customer is there and delete the customer if so; otherwise, raise an exception.
6. open_account(customer, address, amount): first check if the amount is positive, and the customer and branch exist. If so, generate an account number whose first 3 digits are branch number and whose remaining 4 digits are the highest account number of this branch plus 1. If there is an unused account number, it should use it. If not, generate a number that is the highest local account number plus one, and insert the information into the table.
7. close_account(a#) : check if the account exists and the balance is 0; otherwise raise an exception. .
8. withdraw(account, amount): check if the account exists and the balance is more than amount; otherwise, raise an exception.
9. deposit(account, amount): check if the account exists and the amount is positive; otherwise, raise an exception.
10. transfer(account1, account2, amount): if the accounts exist and amount is positive, transfer the amount from account to account2; otherwise, raise an exception.
11. show_branch(address): check if the branch exists and display the accounts in the branch and the balance of each account and the total balance if it exists; otherwise, raise an exception.
12. show_all_branches(): invoke show_branch method for all branches in the system.
13. show_customer(customer): check if the customer exists and display customer number, account numbers and balance in each account and the total balance.

You need to use procedures or functions for these subprograms based on the descriptions given above and display some proper message when raising an exception.

Part 3 JDBC Programming (20)

Write one JDBC program that allows bank employees to perform banking tasks. The program should provide a simple menu to allow the user to select the option and have several methods based on the tasks in Part 4 that require more than one PL/SQL procedures and functions to perform the task and display the result.

Part 3 JDBC Database Populating and Testing (40)

Use the JDBC program to populate the tables with exactly the following information

1. open a branch in London
2. open a branch in Munich
3. open a branch in New York
4. open a branch in Toronto
5. create a customer Adams,
6. create a customer Blake,
7. create a customer Henry,
8. create a customer Jones,
9. create a customer Smith
10. open an account for Adams in London branch with an initial deposit of $1000
11. open an account for Adams in Munich branch with an initial deposit of $1,000
12. open an account for Adams in New York branch with an initial deposit of $1,000
13. open an account for Adams in Toronto branch with an initial deposit of $1,000
14. open an account for Blake in London branch with an initial deposit of $1,000
15. open an account for Blake in Munich branch with an initial deposit of $2000
16. open an account for Blake in New York branch with an initial deposit of $3000
17. open an account for Henry in London branch with an initial deposit of $2,000
18. open an account for Henry in Munich branch with an initial deposit of $1,000
19. open an account for Jones in Toronto branch with an initial deposit of $5,000
20. show customer Adams (not just customer table info, use PL/SQL subprograms)
21. show customer Blake
22. show customer Henry
23. show customer Jones
24. show customer Smith
25. show London branch
26. show Munich branch
27. show New York branch
28. show Toronto Branch
29. show all branches
30. deposit $1000 to Smith's Toronto account
31. transfer $1000 from Smith's London account to Toronto account
32. transfer $1000 from Henry's Munich account to London account
33. transfer $3000 from Henry's London account to Jones' Toronto account
34. transfer $1000 from Adam's London account to Munich account
35. transfer $1000 from Adam's New York account to Toronto account
36. delete Smith as a customer
37. close all accounts that have balance 0 (Henry's two account)
38. Show all branches (Henry's account should be gone)
39. open an account for Jones in London branch
40. show customer Jones (the account # should be the one Henry had)