CPRG 307
Assignment 3 (Modules 1 – 10)

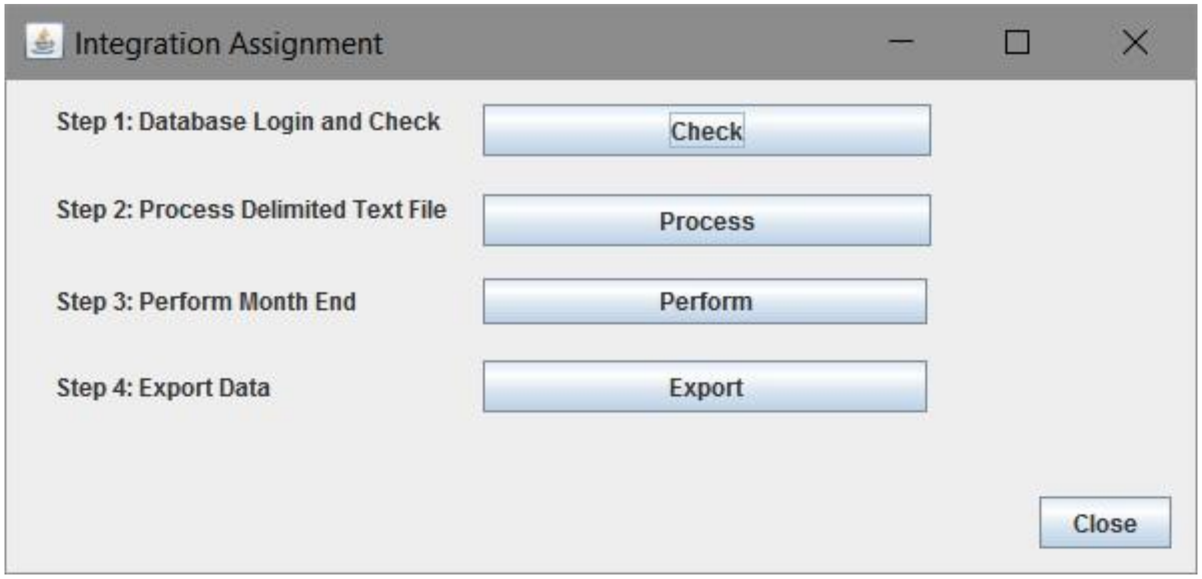
Mark: _______ / 26

Problem

This assignment is based on the We Keep It Storage (WKIS) Company's system, an accounting system, and is an extension from what we did in Assignment 2 (you do **not** have to have Assignment 2 completed and working to do Assignment 3). WKIS outsources its payroll to an external company called Payroll Systems Inc. (PSI). After each payroll run, PSI provides WKIS with a delimited text file with the payroll transaction information in it. WKIS needs this payroll information to maintain its accounting system transactions.

Your task is to process this delimited text file, perform a month end process, and export database data. There are four distinctive steps (with multiple tasks each) that are performed. This application has a Java GUI front end. You have been provided with an executable JAR of this
Java program (located in D2L Module 9). The only code pieces you are directed to create are the PL/SQL stored programs. The executable JAR has been provided to you so you can perform integration tests to ensure your code is compatible with the front end.

This is what the front end looks like:


Integration Assignment
Step 1: Database Login and Check
Check
Step 2: Process Delimited Text File
Process
Step 3: Perform Month End
Perform
Step 4: Export Data
Export
Close

Before writing your code, your CPRG307 user needs to have permissions added in order to use this user to test parts of the assignment. The following steps only needs to be performed once:

1) Connect to the database as *sys as sysdba*
   a. The password will be the same as the password for *system*
2) Issue the following commands (*cprg307* is the name of the user):
   a. `GRANT execute ON utl_file TO cprg307;`
   b. `GRANT create any directory TO cprg307;`
   c. `GRANT drop any directory TO cprg307;`

The four steps of the application (guidelines for your application):

1) Connect to the database and check permissions.

a. What the Java code does:

i. Prompt the user for the name of the database. Should be: **XE** ii. Prompt the user for their listener port

number. Should be: **1521**

iii. Prompt the user for database login credentials (the *username* and *password*).

iv. Connects to the database. If the user login failed because of an incorrect username or password, an error message is displayed to the screen to inform the user who can then try again if they so wish.

v. Will use JDBC to call a stored function.

vi. If *'N'* is returned by the function, the Java program will provide a message to the user indicating they do not have the required permissions to continue and the user should be disconnected from the database.

b. What you must create:

i. Create a stored VARCHAR2 function called *func_permissions_okay* that checks if the user, who is logged into the database, has permission to complete all of the steps ○ This function will have no input values

ii. This stored function will check that this user (the one currently connected to the database) has been given permission to *execute* the *UTL_FILE* package.

iii. The function will return *'Y'* if the user has this permission or it will return *'N'* if they do not.

- To find out if the user has this permission, you will need to look at the USER_TAB_PRIVS data dictionary view. If the user has this permission, there will be an entry in this view for a **privilege of *EXECUTE*** for the **package (in this case the column is TABLE_NAME) *UTL_FILE***.

‘EXECUTE’ and ‘UTL_FILE’ can be hard coded <u>for this</u> <u>step</u>. 2) Process the delimited text file provided.

a. A delimited text file has been given to you (available in D2L).

b. This file (payroll.txt) has been created with a “;” as a delimiter.

c. What the Java code does:

   i. Prompts the user for the path and name of the delimited file.

   ii. Prompts the user for the path and name of the control file to be created (this is an information file that SQL*Loader requires).

   iii. Creates the SQL*Loader control file.

   iv. Prompts the user for the path and name of the SQL*Loader log file that will be generated by SQL*Loader in which errors will be placed when this utility processes the delimited file.

   v. Will call an Oracle utility call SQL*Loader which will take each row in the delimited file and will insert them into the PAYROLL_LOAD table.

   vi. **NOTE: Folder path must be created by you and must exist. vii. NOTE: Folder paths and file names cannot contain spaces.**

d. What you must create:

i. A DML trigger.

ii. Each new payroll entry being loaded should be processed by the database. To do this a trigger (<u>one</u> DML trigger, not a compound trigger) should fire on the PAYROLL_LOAD table. This trigger will:

- Create a new transaction for each payroll entry and put this transaction into the NEW_TRANSACTIONS table. Remember that we are using double-entry accounting. For each row in PAYROLL_LOAD, a **complete transaction** must be added to the NEW_TRANSACTIONS table. This means NEW_TRANSACTIONS would need two entries – the first entry is the credit to the <u>accounts payable account</u> and the second entry is the debit to the <u>payroll expense account</u>.

- As these two accounts will never change, you can hardcode these account numbers <u>for this step</u>.

- Every entry that goes into the NEW_TRANSACTIONS table will put a value into each column. Use the sequence available with the WKIS database to produce the transaction number.

- The STATUS for the associated row in PAYROLL_LOAD should be changed to *G* if it was processed above with no errors or *B* if there were errors.

3) Perform the month end process.

a. What the Java code does:

i. Will use JDBC to call a stored procedure.

b. **What you must create:**

i. A stored procedure called *proc_month_end* that will zero out all temporary accounts:

- This stored procedure will have no input values.
- Temporary accounts are considered the revenue and expense accounts.

  These two account types (‘RE’, ‘EX’) can be hard coded <u>for this step</u>.

- By “zero out”, we do not mean changing the balance for these accounts to zero directly (i.e. the ACCOUNT table is never modified). We need to create a transaction (debit and credit) that will do this.
- For any revenue account, to zero it out, create a transaction (placing it in the NEW_TRANSACTIONS table), with the revenue account having a debit for the balance amount and a credit to the <u>owners equity account</u> for the same amount.

  The owner’s equity account number can be hard coded <u>for</u> <u>this step</u>.

  The revenue accounts should be dynamically retrieved from the database.

- For any expense account, to zero it out, create a transaction (placing it in the NEW_TRANSACTIONS table), with the expense account having a credit for the balance amount and a debit to the <u>owners equity account</u> for the same amount.

  The owner’s equity account number can be hard coded <u>for</u> <u>this step</u>.

  The expense accounts should be dynamically retrieved from the database.

- o Every entry that goes into the NEW_TRANSACTIONS table will put a value into each column. Use the sequence available with the WKIS database to produce the transaction number.

4) Export transactional data to a delimited file.

   a. Once all of the new transactions have been created, WKIS wants to be able to pull this data from the database and put it into a comma delimited file that will be used by an external application for analysis purposes.

   b. What the Java code does:

      i. Will prompt the user for three pieces of information:

         - o The complete directory path that the export file will go to. o The name of the delimited file to be

           created.

         - o The "alias" for the directory path.

      ii. Will JDBC to issue the command to the database to create the alias (pointer to the directory path).

      iii. Will use JDBC to call a stored procedure.

   c. What you must create:

      i. A stored procedure to create and populate the delimited file.

      ii. The store procedure called *proc_export_csv* will take in two parameters (the directory alias and the file name).

         - o Programming Note: Inside the procedure, the directory alias (the value of the input parameter, not the parameter) must be in upper case only or it will not work correctly.

iii. The delimiter that should be used is a comma.

iv. The table to be exported is the NEW_TRANSACTIONS table.

v. All columns and rows for this table should be included in this export.

vi. To create and populate the file correctly, look at the UTL_FILE package: http://docs.oracle.com/cd/B19306_01/appdev.102/b14258/u_file.htm

- You will need to do a little research and testing to get this feature figured out and working.

- Stored programs that you may need would include:

  UTL_FILE.FOPEN

  UTL_FILE.PUT

  UTL_FILE.NEW_LINE

  UTL_FILE.FCLOSE

Before starting your application coding, make sure to:

1. Ensure you have created the WKIS database tables successfully. Use the WKIS scripts provided with this assignment located in *Module 9* on D2L below this assignment document.

2. Design and write a PL/SQL program using the information outlined above.
   a. Do not worry about invalid data going into the database from the delimited text file that was processed (other than as noted with the status flag in your trigger). Data errors will be handled when the transactions in the NEW_TRANSACTIONS table are processed (this is the process you created in Assignment 2).

    b. The Java code provided assumes **<u>all</u>** data being prompted for is clean. Bad data will crash the program.

3. There should be no hard coding of values except where specifically indicated to do so in the assignment.