CSCI 4305/5305 – Linux
Project 1: Due EOD on December 6, 2018

Total points

This project is 15% of your final grade.
180 total possible points for Undergraduates.
200 total possible points for Graduate and 4+1 Students.

Submission

You should submit your script (etl.sh) and any supporting scripts in a tarball. You may also include a README text file if you would like to add any details that would help with testing/running your script. I should be able to untar the files and run the scripts from that directory with no problems. This set of scripts is worth 180-200 points. You should **NOT** wait until the last day to write it. You are welcome to ask questions about approaches and collaborate. However, the final scripts must be your own work.

Introduction

In this project, you will do work similar to an ETL engine using Linux tools and utilities. All of the following should be automated with one driving bash script that runs each step or multiple steps.

- Use a variety of different Linux utilities/filters to accomplish the tasks (tr, awk, sed, bash, sort, cut, etc). You may only use standard Linux utilities, bash scripts that act as filters, and c programs that act as filters**. Do not** use programming languages we have not covered in class (python, perl, ruby, etc) and try not to use the same program to solve each problem. However, **do** use the program that makes the most sense for your problem. For example, awk is a good utility for reports and a good candidate for several other steps as well. *If you use c filters, you must provide a makefile that will compile the code on my computer when I type "make".*
- Each step should be written using a separate script or tool. Do not write one big program to perform all the steps. The goal is to have separate programs that could be reused later for other ETL problems.
- The script should trap reasonable errors and print useful error messages to help the user resolve problems.
- The script should print messages to standard output to indicate which steps of the ETL process have completed.
- If no parameters are passed to your script, it should print a usage statement that makes it easy for the user to figure out how to run the script***. If I can't figure out how to run your script, I can't grade it!***
- The bash script should accept parameters for the following:
  - Remote file transfer parameters:
    - ***($1) remote-server***: server name or ip address
    - ***($2) remove-userid***: userid to login to the remote machine. (assume you are using ssh keys so that a password is not required)
    - ***($3) remote-file***: full path to the remote file on the remote server
  - ***For graduate and 4+1 students only or bonus points for undergraduate:***
    - MariaDB parameters
      - ***($4) mysql-user-id***
      - ***($5) mysql-database***
- Keep in mind that for grading, this script will be tested with a different file from the test file. Try to think about what type of errors you could run into and handle them in your script.

Project Details

Your ETL script (etl.sh) should execute scripts or Linux utilities that perform the following steps:

1. Transfer test file using sftp, scp, or rsync . Transfer ***remote-file*** from ***remote-server*** using ***remote-userid***. File contains fields: (customerID, firstname, middleinitial, lastname, street, city, state, zip, gender, storeID, amount, date). This is the called the **transaction** file. (12)
2. Unzip the **transaction** file. (8)
3. Remove header record from the file. (6)
4. Convert all text in the file to lower case. (6)
5. The "gender" field has values { "f", "m", “female”, “male”, "1", "0" } - convert "1" to "f" and "0" to "m". Convert “male” to “m” and “female” to “f”. Place “u” in fields that do not have a valid value in them. This field should ONLY have “m”,”f”,or “u” after this step. (18)
6. Filter records out of the transaction file that do not have state or contain “NA”. Place these records in an exceptions file called **exceptions.csv**. (12)
7. Remove the $ sign from the purchase_amt field. (8)
8. Sort transaction file by customerID. The format of the transaction file should not change. Only the sort order should be different. The final transaction file should be called **transaction.csv**. (12)
9. Accumulate total purchase amount for each "customerID" and produce new file with a single record per customerID and the total amount over all records for that customer. Use commas as your field delimiter.
   The fields in this file should be in this order
   - customerID
   - state
   - zip
   - lastname
   - firstname
   - total purchase amount

   Make sure you use non-blank fields for the summary record when they are available. This is your summary file. (20)
10. Sort the summary file based upon
    - state
    - zip
    - lastname
    - firstname

    Keep the file format the same as specified in step 8. Only the sort order should be different. The final summary file should be called **summary.csv**. (12)
11. Generate the following reports:
    - Transaction Report - showing the number of transactions by state abbreviation. **transaction-rpt** should be the name for the file containing this report. The state abbreviation should be uppercase. You should have this title on the report “Transaction Count Report”. You should also have column headers for each column (State and Transaction Count). The report should be sorted by number of transactions in descending order followed by state. (10)
      Example Report (Just the first few lines):

```
Transaction Count Report

State    Transaction Count
TX       131
CA       103
FL       96
NY       60
```

- Purchase Report – showing the total purchases by gender and state. The state abbreviation and gender to be in uppercase. You should have this title on the report "Purchase Total Report". You should also have column headers for each column. (State, Gender, Purchase Amount) The report should be sorted by total number of purchases in descending order, then by state, and finally by gender. **purchase-rpt** should be the name for the file containing this report. (10)

  Example Report (Just the first few lines):

```
Purchase Summary Report

State    Gender  Purchase Amount
TX       F       33734.33
CA       F       23911.61
TX       M       23043.64
FL       M       18846.49
```

12. ***Graduate and 4+1 students only or bonus points for undergraduate: (20)***
    Load files into mysql. Don't forget to use the parameter passed in for user id and database. *You may want to look at using mysqlimport from your etl.sh script*. You may also want to use the -local option on *mysqlimport*. Table layouts should match csv layouts. You may create the tables before running the script and assume the tables exists before running the import. For an extra challenge, you can drop and create the tables from this script as well.
    - Load "**transaction.csv**" file into the "TRANSACTION" table in the ***mysql-database*** in MySQL/MariaDB (10)
      - MySQL data types for fields: VARCHAR for most fields, purchase amount – DECIMAL(13,2), transaction date – DATE
    - Load "**summary.csv**" file into the "SUMMARY" table in the ***mysql-database*** in MySQL/MariaDB. (10)
      - Data types for fields VARCHAR for most fields, purchase amount – DECIMAL(13,2)
13. Remove any intermediate working files. The only files left should be the final transaction and summary files, the exception file, and the report files. These files should NOT be deleted if the scripts exits with an error. (10)

Other rubric items:

- Coding standards and reusability for all scripts (36 points)
  - Includes usage statement and other coding standards
- Readability/Documentation for all scripts (36 points)
  - Script documentation in README file
  - Comments in code
  - Anything else specified in coding rubric.

Testing

1. Try script with bad file transfer credentials.
2. Try script with non-existent source file.
3. Build test file with following errors in some fields
    a. "Gender" value of blank and "x"
    b. "State" value of blank and "NA"
    c. "Amount" value of blank
    d. Mixed case in name and street fields
4. Confirm accumulated total purchases
5. Confirm reports.
6. Confirm transaction and summary files loaded into MySQL. (Graduate and 4+1)

Location of transaction file for testing:

The transaction file is located on our class server. You can only access this server when you are on the UCA campus, so you may want to scp the file to your local machine for localized testing.
161.31.72.29:/fall2018/MOCK_MIX_v2.1.csv.bz