Total Points

This project is 13% of your Final grade.
There are 160 points for Undergraduates.
There are 180 points for Graduate and 4+1 Students.

Special Instructions

You must include a **README.txt** and add any details that would help with testing/running your script. I should be able to **untar** the files and run **one driving script** from that directory with no problems. This set of scripts is worth 160 points for 4305 students and 180 points 5305+ students. You should **NOT** wait until the last day to complete this project. You are welcome to ask questions about approaches and collaborate. However, the final scripts must be your own work (so be careful with sharing your work). Any duplicated work results in a zero for both parties and this project is 13% of your total grade.

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 (Do not make me run several scripts).

- This is a **comprehensive project**, you will use what you have learned through the semester to complete it.
- 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 covered in this class. **Do not** use programming languages we have not covered in class (python, perl, ruby, etc), this will result in a **zero** for this project. Try not to use the same utility/program to solve each problem, but a variety of different utilities. However, 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.
- Each numbered step should be written as a separate script or tool within your script. In other words do not write one big piped program to perform all the steps. The goal is to have separate smaller programs within your script 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 been 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) remote-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.
  - 5305/6305 (Graduate and 4+1) MariaDB Parameters - requirements or bonus points for 4305 level students

    ($4) mysql-user-id: sql user id

($5) mysql-database: sql database name

- 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.
- Examples of this can be found in the section **Testing your script**
- Sort - General Info
  - Unless otherwise specified use the default sort order.
  - Pay close attention to source data. Data that is exclusively numeric must be sorted as such.
- **Special Note** on Source File **MOCK_MIX_v2.1.csv.bz2**
  - The filename of the source file can be different (make sure not to hard code the filename)
  - The location (file path) of the source file can be different (make sure not to hard code the file path)
  - The file type of the source file will always be a .csv
  - The file compression of the .csv file will always be a .bz2

Project Details

Server IP Address: **40.69.135.45**
Source file name and location: **/home/shared/MOCK_MIX_v2.1.csv.bz2**

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

1. Transfer the source file using the scp command to your project directory.
   - The scp command should accept the **remote-server**, **remote-usrid**, and **remote-file** variables as arguments for its required input.
   - The source file contains fields (customer_id, first_name, last_name, email, gender, purchase_amount, credit_card, transaction_id, transaction_date, street, city, state, zip, phone)
   - Refer to the source files header for the definitive format.
   - The source file will be refereed to as the **transaction** file from this point forward and should be named so in your code.
2. Unzip the transaction file.
3. Remove the header record from the transaction file.
4. Convert all text in the transaction file to lower case.
5. The "gender" field could contain values "f", "m", "female", "male", "1", "0", "u", " ", "etc..." - convert them as follows:
   - "1" to "f"
   - "0" to "m".
   - "male" to "m"
   - "female" to "f".
   - "u" for all other fields that do not match the above criteria.
   - The Gender field should **ONLY** have "m","f",or "u" after this step.

6. Filter all records out of the transaction file from the "state" field that do not have a state or contain "NA". Place these records in an exceptions file named **exceptions.csv**.

   NOTE: These exceptions should no longer be located in the transaction file.

7. Remove the $ sign in the transaction file from the purchase_amt field.

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 named **transaction.csv**.

9. Generate a summary file using the **transaction.csv** file. Accumulate the total purchase amount for each "customerID" and produce a new file with a single record per customerID and the total amount over all records for that customer. Use commas as your field delimiter.

   (a) The fields in this file should be in this order:

      1. customerID
      2. state
      3. zip
      4. lastname
      5. firstname
      6. total purchase amount

   (b) Sort the summary file based upon (be careful with this step, the key to this is the sort order "**priority sort**"):

      1. state
      2. zip (decending order)
      3. lastname
      4. firstname

      Keep the file format the same as specified in step 9(a). Only the sort order should be different. The final summary file should be named **summary.csv**.

10. Generate the following two reports using the **transaction.csv** file.

    (a) **Transaction Report** - Show the number of transactions by state abbreviation. The state abbreviation should be uppercase. The report should have two titles at the top "*Report by: [FirstName LastName]*" and "*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.
    Name this report **transaction.rpt**.

    See "*Example Transaction Report*" under *Sample Output* section for detailed format of the file.

    (b) **Purchase Report** – Show the total purchases by gender and state. The state abbreviation and gender needs to be in uppercase. The report should have two titles at the top "*Report by: [FirstName LastName]*" "*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. **NOTE**: the Purchase Amount field has numbers rounded to the nearest hundreds.
    Name this report **purchase.rpt**.

    See "*Example Purchase Report*" under *Sample Output* section for detailed format of the file.

11. 5305/6305 (Graduate and 4+1) requirements or bonus points for undergraduate:
    This is a research project, therefore you will need to research the implementation process on how to accomplish this task (I give a few hints on where to start).

You will need to install **mariadb-server** onto your Linux distribution. There are many tutorials online to help you with this process.

After your script generates the two reports above, your script should prompt for a database password. **NOTE:** The user should not be able to see what password is entered. The password should be saved to a variable that is later used in your mysql `--password` command. Your task is to load files into mysql. Don't forget to use the parameters passed in for **mysql-user-id** and **mysql-database** as the arguments supplied to your mysql commands. The import must use a mysql password (do not use a blank or hard coded clear text password). You may want to look at using **mysqlimport** for the implementation within 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.

(a) Load "**transaction.csv**" file into the "**TRANSACTION**" table in the mysql-database in MySQL/MariaDB
- MySQL data types for fields:
VARCHAR - most fields, DECIMAL(13,2) - purchase amount, DATE - transaction date

(b) Load "**summary.csv**" file into the "**SUMMARY**" table in the mysql-database in MySQL/MariaDB.
- Data types for fields:
VARCHAR for most fields, DECIMAL(13,2)- purchase amount.

12. After executing your script all intermediate working files should be **removed**. The only files remaining should be the final **transaction.csv**, and **exception.csv**, **summary.csv**, **transaction.rpt**, **purchase.rpt** files. These files should **NOT** be deleted if the scripts exits with an error.

Other Rubric items

- Coding standards and re-usability for all scripts

  Includes usage statement and other coding standards

- Readability/Documentation for all scripts

  README: How to run your script should be included in the README.txt file.

  Comments!!! You must comment your code

- Anything else specified in coding rubric.

Testing your script

- Try running script with a bad file transfer credentials.
- Try running script with non-existent source file.
- Build test file with the following errors in some of the fields
  - "Gender" value of blank and "x" (convert to **"u"**)
  - "state" value of blank and "NA" (this would be an exception)
  - Mixed case in name and street fields
- Confirm accumulated total purchases
- Confirm reports
- Confirm transaction and summary files loaded into MySQL (5305/6305)

Note

- You may want to scp the file to your local machine for localized testing in case the server is unreachable or for offline testing
- All currency must be in a 2 deciaml point format! Example: **33734.33**
- Make sure your script is inside a folder with your name on it.

Sample Output

This section contains two output files. **Note**: this is the first 7 lines of **actual output** (correct answers) your report should generate. Replace the placeholder ([**Your Name**]) with your First and Last name.

Example Transaction Report

```
$ head -n 7 transaction-rpt
Report by: [Your Name]
Transaction Count Report

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

Example Transaction Report

```
$ head -n 7 purchase-rpt
Report by: [Your Name]
Purchase Summary Report

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

Submission Details

Submitted Files: Only submit the following files:

- README.txt
- etl.sh
- Any supporting directory/scripts to etl.sh

Do **NOT** submit the following files, as your script should generate these:

- `MOCK_MIX*.tar.bz2, transactions.csv, exceptions.csv, summary.csv, transaction.rpt, purchase.rpt`

Place all required files in a gzip compressed tarball that follows our class conventions (LastName-FirstName-Project.tar.gz) and upload to blackboard.

NOTE: Make sure to answer the questions on the Semester Project Assignment Information page on Blackboard before submitting your project. You must complete the Write Submission along with the project.

General Requirements for all lab/assignments/projects

The following are required for all assignments and are included in the rubric for grading (unless otherwise specified):

- Submit work in the format specified in the instructions. If nothing is specified then any approved file type will do.
- All submitted work must be in the following file-name format:
  - "lastname-firstname-filename.fileExtension" (Example: corkran-jacob-assignment5.tar.gz)
- Name scripts as described in each step.
- Programs need to have the exact output, unless otherwise stated.
- Make sure to use spaces and newlines as required, proper formatting is required.
- Source files must have comments that explain your implementation.
- Source files must include the following set of comments at the top of the scripts.

```
# FirstName LastName
# Assignment/Lab/Project Name (Example: Assignment 5)
```

Warning: You may only submit coding assignments in the Programming Languages covered in this course: Bash, AWK, Sed. All other languages will result in a zero for the assignment.