**Tech Exercise**

**Name:** Manikandan Ganesh

**Email:** [karthickgm27@gmail.com](mailto:karthickgm27@gmail.com)

**Phone:** 312-874-8660

**LinkedIn:** <https://www.linkedin.com/in/manikandan-ganesh-02919963>

**GitHub:** <https://github.com/manikandanganesh2791>

**Objective**

The purpose of this task is to create a clean attendance record dataset which is likely to be loaded into a new sole system.

The word” clean” indicates four things here:

1. The final data set must be simple and all the department in the company should find it easy to use.
2. The structure of my design must be flexible so that, if the business logic changes, the design can be changed easily.
3. The solution must open doors for more data.
4. The data set must be consistent across all the regions.

Making these assumptions, I decided to work on the given unclean data set from different systems (c, k, s) and transform them into a single clean data set satisfying the ACID properties in the context of database.

**Data Cleaning**

I first started off by looking at the data set of all the given three different systems and I could observe the following inconsistencies and unclean records.

**Removing Duplicates in attendance\_s.csv file**

I observed the presence of duplicate values in attendance\_s.csv file.

For instance, I could observe the presence of redundant data (like the one given below) carrying the same ID value in the attendance\_s.csv file.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **CardSourceID** | **CardHolderID** | **DateIn** | **ReportField1** | **ReportField2** | **DateOut** | **HoursWorked** |
| 4071 | 73 | 218696326 | 2/27/2017 10:38 |  |  | NULL | 0 |
| 4071 | 73 | 218696326 | 2/27/2017 10:38 |  |  | NULL | 0 |
| 4072 | 73 | 201706009 | 2/27/2017 10:39 |  |  | NULL | 0 |
| 4072 | 73 | 201706009 | 2/27/2017 10:39 |  |  | NULL | 0 |

So, I removed these duplicate entries from the data set first. Also, I restructured the table format by having separate fields for Date and Time. My final table after cleaning the entire data set had the following structure:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **CardSourceID** | **CardHolderID** | **DateIn** | **InTime** | **ReportField1** | **ReportField2** | **DateOut** | **Out Time** | **HoursWorked** |
| 4839 | 73 | 226730190 | 3/21/2017 | 9:09 |  |  | NULL | NULL | 0 |
| 4838 | 73 | 223198193 | 3/21/2017 | 9:05 |  |  | 3/23/2017 | 15:48 | 54.717 |
| 7045 | 73 | 221142714 | NULL | NULL |  |  | 4/6/2017 | 15:54 | 0 |

I used Python to scrap and cleanse the data set to obtain a new clean source record from the “s” system, in the prescribed format. The code I used to clean the set is given below:

**Python Code:**

1. target\_file = open('attendance\_s\_clean.csv', 'w')
2. attendance\_s = open('attendance\_s.csv', 'r')
3. row\_s = attendance\_s.read().splitlines()
4. row\_list = []
5. record\_dict = {}
6. date\_split\_list = []
7. date\_val = ""
8. target\_line = ""
10. #Method list\_val is used to clean the attendance\_s.csv file by returning only
11. #the set of rows having unique Id and eliminates the redundant Id values.
12. #Also, this method parses the datein and dateout column values by stripping
13. #the date of the format mm/dd/yyyy and removing the time values from it.
15. **def** list\_val(row\_list):
16. result = []
17. **for** i **in** range(len(row\_list)):
18. date\_val = ""
19. time\_val = ""
20. date\_split\_list = []
21. **if**(i != 0):
22. **if**(i==3):
23. **if**(row\_list[i] != 'NULL'):
24. date\_split\_list=row\_list[i].split(" ")
25. date\_val=date\_split\_list[0]
26. time\_val=date\_split\_list[1]
27. result.append(date\_val)
28. result.append(time\_val)
29. **else**:
30. result.append(row\_list[i])
31. result.append(row\_list[i])
32. **elif**(i==6):
33. **if**(row\_list[i] != 'NULL'):
34. date\_split\_list=row\_list[i].split(" ")
35. date\_val=date\_split\_list[0]
36. time\_val=date\_split\_list[1]
37. result.append(date\_val)
38. result.append(time\_val)
39. **else**:
40. result.append(row\_list[i])
41. result.append(row\_list[i])
42. **else**:
43. result.append(row\_list[i])
44. **return** result
46. #Loop that forms the dictionary 'record\_dict' to store the records from
47. #attendance\_s.csv file having the key as Id and values as a list carrying
48. #CardSourceId, CardHolderId, DateIn, DateOut and HoursWorked values
49. #for CardSourceId '73'
50. **for** line **in** row\_s[1:]:
51. row\_list = line.split(",")
52. **if**(row\_list[1] == '73'):
53. **if**(row\_list[0] **in** record\_dict):
54. **continue**
55. record\_dict[row\_list[0]] = list\_val(row\_list)
56. column\_header = 'ID'+','+'CardSourceID'+','+"CardHolderID"+','+"DateIn"+','+"InTime"+','+"ReportField1"+','+"ReportField2"+','+"DateOut"+','+"OutTime"+','+"HoursWorked"
57. target\_file.write(column\_header)
58. target\_file.write("\n")
59. **for** key, value **in** record\_dict.items():
60. target\_line = ""
61. target\_line+=key
62. **for** i **in** range(len(value)):
63. target\_line+=","+value[i]
64. target\_file.write(target\_line)
65. target\_file.write("\n")
66. target\_file.close()

After cleaning the data set, I decided to dump the entries into a MySQL table exclusively for attendance\_s.csv entries. Before creating a table, I first created a database named “**attendance**” inside the MySQL engine using the script –

**CREATE DATABASE attendance;**

The purpose of creating the database is to have a common store and easy accessibility for all the tables I will be creating in this exercise.

I created a table exclusive for attendance\_s.csv file entries inside the **attendance** database using the script,

**drop table if exists attendance\_s;**

**create table attendance\_s(**

**ID int,**

**CardSourceID int,**

**CardHolderID int,**

**DateIn varchar(12),**

**InTime varchar(5),**

**ReportField1 varchar(100),**

**ReportField2 varchar(100),**

**DateOut varchar(12),**

**OutTime varchar(5),**

**HoursWorked int);**

I dumped the values from the attendance\_s\_clean.csv (cleaned version of attendance\_s.csv) into the MySQL table using the following command:

**LOAD DATA LOCAL INFILE '~/attendance\_s\_clean.csv' INTO TABLE attendance\_s**

**FIELDS TERMINATED BY ','**

**ENCLOSED BY '"'**

**LINES TERMINATED BY '\n'**

**IGNORE 1 LINES;**

**Removing Duplicates in attendance\_c.csv file**

Like attendance\_s.csv file, attendance\_c.csv file also had duplicate entries. In comparison to attendance\_s.csv, attendance\_c.csv file was less complex in terms of number of fields and the type entries. So, I did not make any changes to the table structure. I used the following Python code for data scraping and data cleansing:

1. target\_file\_c = open('attendance\_c\_clean.csv', 'w')
2. attendance\_c = open('attendance\_c.csv', 'r')
3. row\_c = attendance\_c.read().splitlines()
4. c\_entry\_set = set()
5. line\_list=[]
6. iout=0
7. **for** i **in** row\_c[1:]:
8. line\_list = i.split(",")
9. **if** line\_list[0]+"|"+line\_list[1] **not** **in** c\_entry\_set:
10. target\_file\_c.write(i)
11. target\_file\_c.write("\n")
12. c\_entry\_set.add(line\_list[0]+"|"+line\_list[1])
13. target\_file\_c.close()

I followed the same process of creating a MySQL table and dumping the entries of attendance\_c\_clean.csv file into the table. The scripts are given below:

**drop table if exists attendance\_c;**

**create table attendance\_c(**

**ID int not null auto\_increment,**

**CardHolderID int,**

**Date varchar(12),**

**Status varchar(25),**

**PRIMARY KEY(ID));**

**LOAD DATA LOCAL INFILE '/vagrant\_data/attendance\_c\_clean.csv'**

**INTO TABLE attendance\_c**

**FIELDS TERMINATED BY ','**

**ENCLOSED BY '"'**

**LINES TERMINATED BY '\n'**

**(@col1,@col2,@col3) set CardHolderID=@col1,Date=@col2,Status=@col3;**

One unique condition to be noted down while dumping the contents from csv into the table is, unlike the attendance\_s.csv file, attendance\_c.csv file did not have a primary key field. To overcome the problem, I used an **AUTO INCREMENTED** primary key field and customized the process of dumping using temporary variables (@col1, @col2, @col3).

**Removing Duplicates in attendance\_k.csv file**

Like attendance\_c.csv, attendance\_k.csv file was less complex. The python code used for cleaning the data set is given below:

1. target\_file\_k = open('attendance\_k\_clean.csv', 'w')
2. target\_file\_k\_empty = open('attendance\_k\_empty.csv','w')
3. attendance\_k = open('attendance\_k.csv','r')
4. row\_k = attendance\_k.read().splitlines()
5. row\_k\_list =[]
6. empty\_id\_list=[]
7. k\_entry\_set = set()
8. **for** i **in** row\_k[1:]:
9. row\_k\_list = i.split(",")
10. **if**(row\_k\_list[0] != ""):
11. **if**(row\_k\_list[0] + "|" + row\_k\_list[3] **not** **in** k\_entry\_set):
12. target\_file\_k.write(i)
13. target\_file\_k.write("\n")
14. k\_entry\_set.add(row\_k\_list[0] + "|" + row\_k\_list[3])
15. **else**:
16. target\_file\_k\_empty.write(i)
17. target\_file\_k\_empty.write("\n")
18. target\_file\_k.close()
19. target\_file\_k\_empty.close()

The following SQL scripts are used for table creation and insertion of records from csv file:

**drop table if exists attendance\_k;**

**create table attendance\_k(**

**ID int not null auto\_increment,**

**CardHolderID int,**

**Date varchar(12),**

**Status varchar(25),**

**PRIMARY KEY(ID));**

**LOAD DATA LOCAL INFILE '/vagrant\_data/attendance\_k\_clean.csv'**

**INTO TABLE attendance\_k**

**FIELDS TERMINATED BY ','**

**ENCLOSED BY '"'**

**LINES TERMINATED BY '\n'**

**(@col1,@col2,@col3,@col4) set CardHolderID=@col1,Status=@col2,Date=@col4;**

Once I dumped all the records from the csv files into the corresponding database tables, I decided to design the system which is consistent and at the same time, flexible to any additional business logic.