PSET-4 - Managing Data Excercise 2

March 3, 2025

```
<h2 style ="text-align:center; padding-top:5px;"> CS 101 - Foundation of Data Science and Engi
<p style="text-align:center;padding:5px; fontt-size:14px"><b> PSET-4 - Managing Data Excercise-
```

0.0.1 This is an individual assignment. No collaboration is allowed.

0.0.2 Assignment Goal:

Part-1 : Explore Pandas, perform data cleaning using Pandas.

Part-2 : Generate random sample data in SQL

Part-3 : Practice writing SQL queries

Start by reviewing the provided file nj_teachers_salaries_pset4.csv. Examine the column names, data types of this data file. After reviewing this file please provide your solutions for the questions below.

Note: The file has identical columns that you worked on PSET-3, however all the data are not identical

Resources: https://pandas.pydata.org/docs/reference/frame.html Module 4 & Module 5 Lectures

Please feel free to create new cells in your notebook for completing the assignment.

1 Part-1 (60 points)

In this part you will be working with Pandas to explore and clean data. For each of the questions, please make sure that you show your work on what was done in each step.

For Example if you drop rows, be sure to show the how many rows were dropped at each step. You can use df.shape to show before and after count.

For Questions 3-5 that involve modifying your values, you need to show us few rows where the modification was done. As an example you are looking at df['experience_total'] column and you discover that the column has values that are not numerical. You go ahead and set the values as np.NAN. You should show that those values were indeed set as nan. You can use print statements or simply create a new cell and show some example rows. Please display relevant rows and not the full dataframe.

[1]:
```
import pandas as pd
import numpy as np
import mysql.connector as sq
```

1.1 Question-1 (1 pts)

1.1.1 Create a dataframe called df using the provided csv file nj_teachers_salaries_pset4.csv. Use df.info() to get the information about the columns, non-null values, and data type inferred by Pandas for each column.

Pandas tries to infer the data type of each column. However if you have a numerical column, with an invalid value (such as a string), it will infer it as an object. String values are inferred as object data type.

[9]:
```
# Create a DataFrame
df = pd.read_csv('/Users/kt/Harvard/CS101/PSET/pset4/nj_teachers_salaries_pset4.
 ↪csv')
```

```
/var/folders/0x/msstf8r11wg2nt4pzb707v480000gn/T/ipykernel_26910/1191264309.py:2
: DtypeWarning: Columns (7,8,13,14,15) have mixed types. Specify dtype option on
import or set low_memory=False.
  df =
pd.read_csv('/Users/kt/Harvard/CS101/PSET/pset4/nj_teachers_salaries_pset4.csv')
```

[11]:
```
#Display info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100005 entries, 0 to 100004
Data columns (total 16 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   id                   99998 non-null   float64
 1   last_name            100003 non-null  object
 2   first_name           100003 non-null  object
 3   county               100003 non-null  object
 4   district             100003 non-null  object
 5   school               100003 non-null  object
 6   primary_job          100003 non-null  object
 7   fte                  100003 non-null  object
 8   salary               99983 non-null   object
 9   certificate          100003 non-null  object
 10  subcategory          100003 non-null  object
 11  teaching_route       100003 non-null  object
 12  highly_qualified     100003 non-null  object
 13  experience_district  100003 non-null  object
 14  experience_nj        100003 non-null  object
 15  experience_total     99983 non-null   object
dtypes: float64(1), object(15)
```

```
memory usage: 12.2+ MB
```

1.2 Question-2 (1 pts)

1.2.1 Drop rows that have all values as NaN. (Recall from lecture that you have to set the parameter how='all')

[15]:
```
# Drop rows that have all values as NaN
df.dropna(how="all", inplace=True)
```

1.3 Question-3 (20 pts)

1.3.1 Numerical Columns :

1.3.2 Identify numerical columns excluding id column, remove any invalid characters from numerical columns by first setting it to np.NAN , and finally drop rows containing NaN values. (5))

1.3.3 Set the correct data type for each of the numerical columns (i.e. int , float) (1)

1.4 Check the id column. Set the correct id number for rows that are NA/NaN. Set the correct dtype.(5)

1.4.1 At the end of this step your dataframe should not contain any invalid values for numerical values. Only invalid/missing values should have been dropped. (5)

1.4.2 Please be sure to show your work, meaning , show few example rows that were actually modified. (4)

1.4.3 Note : do not reset the index of the dataframe at any point.

[19]:
```
# Identify numerical columns (excluding 'id')
numerical_columns = ['fte', 'salary', 'experience_district', 'experience_nj',␣
 ↪'experience_total']

# Replace non-numeric values with NaN
df[numerical_columns] = df[numerical_columns].replace(r'[^0-9.]', np.nan,␣
 ↪regex=True)

# Drop rows with NaN values in any numerical column
df.dropna(subset=numerical_columns, inplace=True)
```

[21]:
```
# Convert numerical columns to the correct types
df['fte'] = df['fte'].astype(float)  # Float
df['salary'] = df['salary'].astype(float)  # Float
df['experience_district'] = df['experience_district'].astype(float)  # Float
df['experience_nj'] = df['experience_nj'].astype(float)  # Float
df['experience_total'] = df['experience_total'].astype(float)  # Float
```

[23]:
```
# Fill missing ID values with sequential numbers starting from max existing ID␣
 ↪+ 1
```

```
next_id = int(df['id'].max()) + 1
df.loc[df['id'].isna(), 'id'] = range(next_id, next_id + df['id'].isna().sum())

# Convert ID column to integer
df['id'] = df['id'].astype(int)
```

[25]:
```
# Show a few examples of fixed rows
print("\nExamples of fixed 'id' values:")
print(df.loc[df['id'].isna()].head(5))  # Should return an empty DataFrame if␣
 ↪all NaNs were fixed

# Final check to ensure no missing values in numerical columns
print("\nFinal Check: Any remaining NaNs in numerical columns?")
print(df[['id', 'fte', 'salary', 'experience_district', 'experience_nj',␣
 ↪'experience_total']].isna().sum())
```

```
Examples of fixed 'id' values:
Empty DataFrame
Columns: [id, last_name, first_name, county, district, school, primary_job, fte,
salary, certificate, subcategory, teaching_route, highly_qualified,
experience_district, experience_nj, experience_total]
Index: []

Final Check: Any remaining NaNs in numerical columns?
id                    0
fte                   0
salary                0
experience_district   0
experience_nj         0
experience_total      0
dtype: int64
```

1.5 Question-4 (5 pts)

1.5.1 String Columns:

1.5.2 Identify string/object columns. Remove any leading and trailing spaces. This can be applied to all string columns (3)

1.5.3 Show example rows/columns where leading and trailing spaces were removed.Hint : first_name,last_name have data values with leading and trailing spaces. Show at least 2 such examples where data values were modified for these columns. (2)

1.5.4 No rows should be dropped.

[29]:
```
# Identify string/object columns
string_columns = df.select_dtypes(include=['object']).columns

# Remove leading and trailing spaces from all string columns
df[string_columns] = df[string_columns].apply(lambda x: x.str.strip() if x.
 ↪dtype == "object" else x)
```

[31]:
```
# Show examples where spaces were removed
modified_names = df[
    (df['first_name'].str.startswith(' ')) | (df['first_name'].str.endswith('␣
 ↪')) |
    (df['last_name'].str.startswith(' ')) | (df['last_name'].str.endswith(' '))
]

# Display a few modified rows
print("\nExamples where leading/trailing spaces were removed (first_name,␣
 ↪last_name):")
print(modified_names[['first_name', 'last_name']].head(2))  # Show at least 2␣
 ↪examples

# Final check to confirm no rows were dropped
print("\nFinal Check: Number of rows before and after cleaning")
print(f"Original Row Count: {len(df)}")
```

```
Examples where leading/trailing spaces were removed (first_name, last_name):
Empty DataFrame
Columns: [first_name, last_name]
Index: []

Final Check: Number of rows before and after cleaning
Original Row Count: 99959
```

1.6 Question-5 (20 pts)

1.6.1 Additional Cleaning - String Column :

1.6.2 Perform additional cleaning on string columns. Remove any special/invalid characters from the string columns.

1.6.3 Example :

1.6.4 df['primary_job'] contains a value 'Family & Consumer Sciences â€" Apparel, Textiles And Interiors'.

1.6.5 The special character should be removed to give the value 'Family & Consumer Sciences Apparel, Textiles And Interiors' (2.5 pts)

1.6.6 Perform data cleaning on at least 3 string columns. You will have to identify data values in your string columns, and remove any special characters. (7.5) pts

1.6.7 You should try to avoid setting string columns to np.NAN , and dropping it. However, it is ok if you set some rows to np.NAN and drop it for which values are completely invalid. In the end you should have approximately the same number of rows that you had after finishing Question 3.

1.6.8 We are not looking for a perfect solution. The data may still consist of invalid values. We are more interested in seeing how you have applied your learning to this assignment.

1.6.9 In all cases please show your work, meaning show us few example rows/columns where the data values were actually modified. (10 pts)

1.6.10 Note : In general letters, numbers, punctuations , & , /, , () , - , :, s'_,.,?!&/-:#@ are considered valid. You can choose to include more characters. However, for first name and last name, teaching_route, subcategory you will want to choose only specific characters to be considered valid.

[35]:
```
# Identify string columns
string_columns = df.select_dtypes(include=['object']).columns
```

[37]:
```
# Define valid character patterns for different string columns (using raw␣
 ↪strings r"")
valid_patterns = {
    'primary_job': r"[^A-Za-z0-9\s&\-,/]",  # Letters, numbers, spaces, &, -, /
    'teaching_route': r"[^A-Za-z\s\-']",  # Letters, spaces, -, '
    'subcategory': r"[^A-Za-z\s\-,.:]",  # Letters, spaces, -, ., :
    'first_name': r"[^A-Za-z\-' ]",  # Letters, hyphens, and apostrophes
    'last_name': r"[^A-Za-z\-' ]"  # Letters, hyphens, and apostrophes
}
```

[39]:
```
# Apply regex cleaning to respective columns
for col, pattern in valid_patterns.items():
    df[col] = df[col].str.replace(pattern, '', regex=True)
```

[41]:
```
# Identify modified rows
modified_rows = df[string_columns].copy()
for col in valid_patterns.keys():
    modified_rows[col] = df[col] != df[col].str.replace(valid_patterns[col],␣
 ↪'', regex=True)
```

[43]:
```
# Show examples of modified rows
modified_rows = df[modified_rows.any(axis=1)]
```

[45]:
```
# Display a few examples of modified rows
print("\nExamples of modified string column values:")
print(modified_rows[['primary_job', 'teaching_route', 'subcategory',␣
 ↪'first_name', 'last_name']].head(5))
```

```
Examples of modified string column values:
                       primary_job teaching_route subcategory first_name  \
0    Elementary School Teacher K-5    Traditional  General ed    William
1                              Art    Traditional  General ed      Kelly
2                     Kindergarten      Alternate  General ed  Crystal A
3  Elementary Kindergraten-8 Grade    Traditional  Special ed     Isaiah
4           English Non-elementary    Traditional  General ed     Dustin

  last_name
0   Heckman
1      Bird
2    Aikens
3   Leonard
4    Hinton
```

[47]:
```
# Final row count check to ensure minimal data loss
print("\n Final Check: Number of rows before and after cleaning")
print(f"Row count after Question 3: {len(df)}")
```

```
 Final Check: Number of rows before and after cleaning
Row count after Question 3: 99959
```

1.7 Question -7

1.7.1 Save your cleaned dataframe as cleaned_data.csv. Be sure to set the parameter index = False to avoid saving the index as an extra column

ex: df.to_csv(“cleaned_data.csv”, index=False)

[51]:
```
# Save the cleaned dataframe as 'cleaned_data.csv'
df.to_csv("cleaned_data.csv", index=False)

print("\nCleaned data has been successfully saved as 'cleaned_data.csv'.")
```

```
Cleaned data has been successfully saved as 'cleaned_data.csv'.
```

1.8 Question -8

1.8.1 Similar to PSET-3

1.8.2 8.1 Connect to your MySql database using your username and password. Name the cursor returned from the mysql connection object as mycursor. (1 pts)

1.8.3 8.2 Use the same database as PSET-3 nj_state_teachers_salaries, or if you have deleted it create a database called nj_state_teachers_salaries

1.8.4 8.3 Create a table called teachers_salaries_pset4 with all the columns in your cleaned_data.csv. For this part ,be sure to use appropriate data type for all the columns. If you are facing difficulty creating a column with Float or bool or int , it is ok to store it as TEXT. (MAX 2 allowed for numerical columns being stored as TEXT) (3 pts)

1.8.5 8.4 Using LOAD DATA statement (as discussed in Module 4 lectures) load the data from cleaned_data.csv to your table created in 8.3. Use of OPTIONALLY ENCLOSED BY clause and TERMINATED by clause is recommended. (3 pts)

[55]:
```
# MySQL connection
mydb = sq.connect(
    host="localhost",
    user="cs101",
    password="dataisfun"
)

# Create a cursor object
mycursor = mydb.cursor()
```

[57]:
```
# Create the database if it doesn't exist
mycursor.execute("CREATE DATABASE IF NOT EXISTS nj_state_teachers_salaries")

# Use the database
mycursor.execute("USE nj_state_teachers_salaries")
```

[59]:
```
# Drop table if it already exists to avoid conflicts
mycursor.execute("DROP TABLE IF EXISTS teachers_salaries_pset4")

# Create table with appropriate data types
create_table_query = """
CREATE TABLE teachers_salaries_pset4 (
    id INT PRIMARY KEY,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
    county VARCHAR(50),
    district VARCHAR(100),
```

```
    school VARCHAR(100),
    primary_job VARCHAR(150),
    fte FLOAT,  -- Float data type
    salary FLOAT,  -- Float data type
    certificate VARCHAR(50),
    subcategory VARCHAR(50),
    teaching_route VARCHAR(50),
    highly_qualified VARCHAR(100),
    experience_district FLOAT,  -- Float because we kept decimals
    experience_nj FLOAT,  -- Float because we kept decimals
    experience_total FLOAT  -- Float because we kept decimals
);
"""
# Execute the table creation query
mycursor.execute(create_table_query)

print("\nTable 'teachers_salaries_pset4' created successfully.")
```

```
Table 'teachers_salaries_pset4' created successfully.
```

[64]:
```
# Load data from cleaned_data.csv into the MySQL table
load_data_query = """
LOAD DATA INFILE '/Users/kt/Harvard/CS101/PSET/pset4/cleaned_data.csv'
INTO TABLE teachers_salaries_pset4
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
"""

# Execute the data loading query
mycursor.execute(load_data_query)

# Commit the changes
mydb.commit()

print("\nData loaded successfully into 'teachers_salaries_pset4'.")
```

```
Data loaded successfully into 'teachers_salaries_pset4'.
```

1.8.6 Question 9 - For this question you are only required to run the cells. To get credit your code from Question 8 must have been successfully run, and executed. No credit will be awarded if data was loaded using MySQL workbench.

1.9 Question 9 (5 pts)

Run the 2 cells below. The code checks if all the data rows and columns were stored in the database.

The code below assumes that you named your cursor object as mycursor(As specified in Question-8). If you named it differently, you can rename mycursor to match the variable name.

[66]:
```
cmd = "select count(*) from \
                 nj_state_teachers_salaries.teachers_salaries_pset4 "
mycursor.execute(cmd)
count = mycursor.fetchone()[0]

print(f"Number of rows in teachers_salaries table : {count}")
```

```
Number of rows in teachers_salaries table : 99959
```

[68]:
```
cmd = """SELECT COUNT(*) \
                FROM INFORMATION_SCHEMA.COLUMNS \
                WHERE table_schema = 'nj_state_teachers_salaries' \
                AND table_name = 'teachers_salaries_pset4'"""
mycursor.execute(cmd)
count = mycursor.fetchone()[0]
print(f"Number of columns in teachers_salaries table : {count}")
```

```
Number of columns in teachers_salaries table : 16
```

2 End of Part-1

[ ]:
```

```

2.0.1 For both Part-2 and Part-3 you will need to work on MySQL workbench. For both parts you must submit .sql files. More information below.

3 Part-2 (10 pts)

For this part you will generate a random sample data from the table you created in Part-1 and save it as a csv file. Generating random samples have many use cases in the real world. For example, you are a developer who is working on a software application that requires access to a critical database. Instead you maybe given only a sample of data to work with to develop your application. Another use case is bootstrapping in statistics, or when you test your models with samples of data.

3.1 Question 1 (8 pts)

3.1.1 Use a SELECT statement to generate and output a random sample to :

3.1.2 Include all columns

3.1.3 Include field (column) headings

3.1.4 Randomly select 777 records with a seed value of 7

3.1.5 Output results to a csv file named sample.csv

3.1.6 save your sql as output.sql . You will submit this file as a part of this assignment.

You will find module 5 lecture on SQL Random Sample Generation useful

[ ]:

3.2 Question 2 (2 pts)

3.2.1 Create a dataframe using sample.csv generated from Question-1. Display the first 5 rows, and last 5 rows. Print the shape of the dataframe.

[70]:
```
df = pd.read_csv('/Users/kt/Harvard/CS101/PSET/pset4/sample.csv')
```

[72]:
```
# Display the first 5 rows
print("First 5 rows:")
print(df.head(5))

# Display the last 5 rows
print("\nLast 5 rows:")
print(df.tail(5))

# Print the shape of the dataframe
print("\nShape of the dataframe:", df.shape)
```

```
First 5 rows:
    5907  Velasquez    Carl Hunterdon  South Hunterdon Regional  \
0  91649      Evans   James    Mercer              Hamilton Twp
1  33656      Davis  Amanda    Mercer  Hopewell Valley Regional
2  37479  Mcconnell  Robert    Mercer  Hopewell Valley Regional
3  63816    Francis  Elijah  Monmouth               Neptune Twp
4  29195     Fisher   David    Camden               Camden City

                West Amwell Twp School  \
0            Hamilton North-nottingham
1   Hopewell Valley Central High School
2              Timberlane Middle School
3   Shark River Hills Elementary School
4  Creative Arts Morgan Village Academy

           Elementary Kindergraten-8 Grade  0.5   74437  Standard certificate  \
```

```
0            Elementary School Teacher K-5  0.8   67426                  CEAS
1              Health & Physical Education  0.8   70399  Standard certificate
2                Resource Program In-class  1.0  110165                  CEAS
3                             Music Vocal  0.8  105773  Standard certificate
4  Reading Development/remedial Elementary  0.5   58229                  CEAS

   Special ed  Traditional                Not highly qualified   9  9.1  9.2
0  Special ed  Traditional  Doesn't need to be highly qualified  18   18   18
1  General ed  Traditional  Doesn't need to be highly qualified  13    8   23
2  Special ed  Traditional  Doesn't need to be highly qualified  19   31   35
3  General ed    Alternate                Not highly qualified  13   11   31
4  General ed    Alternate                Not highly qualified  20   35   32

Last 5 rows:
      5907 Velasquez         Carl   Hunterdon South Hunterdon Regional  \
771  76099    Taylor      Monique      Camden            Waterford Twp
772  13819  Robinson        Laura      Bergen            Fort Lee Boro
773  32639    Mclean       Steven  Burlington            Riverside Twp
774  87334     David     Jennifer       Essex        Essex Co Voc-tech
775  83195   Serrano  Christopher      Bergen             Oakland Boro

          West Amwell Twp School  Elementary Kindergraten-8 Grade  0.5  \
771   Thomas Richards Elementary  Lang Arts/literacy Grades 5 - 8  0.8
772                School No. 1  Assistant Principal High School  0.8
773  Riverside Elementary School  Elementary Kindergraten-8 Grade  0.8
774           West Caldwell Tech              Math Non-elementary  1.0
775         Valley Middle School    Elementary School Teacher K-5  1.0

      74437  Standard certificate  Special ed  Traditional  \
771  110102                  CEAS  Special ed    Alternate
772   66928                  CEAS  Special ed    Alternate
773   92725                  CEAS  General ed    Alternate
774   93221                  CEAS  General ed  Traditional
775   92030  Standard certificate  Special ed    Alternate

                   Not highly qualified   9  9.1  9.2
771                Not highly qualified  37   33   39
772  Doesn't need to be highly qualified   9    9    9
773  Doesn't need to be highly qualified  18   18   18
774  Doesn't need to be highly qualified  23   27   23
775                Not highly qualified  11   17   17

Shape of the dataframe: (776, 16)
```

4 Part-3 (30 pts)

For this part you will work on sql queries. You will write your queries for the provided dataset teachersample.csv. We could have asked you to write the queries based on the existing table nj_state_teachers_salaries.teachers_salaries_pset4 , however everyone's data cleaning process will be different resulting in different dataset.

All work need to be done in MySQL workbench

4.1 Question 1

4.1.1 Create a table called salaries within the nj_state_teachers_salaries database. Load the data in to the table from the provided file teachersample.csv. The teachersample.csv does not contain the id column. Please modify your code to work with this csv file.

4.1.2 You don't need to submit the code for this. This table is intended only for queries in Question-2.

[ ]:

4.2 Question 2 (30 pts)

4.3 Each query is worth 3 pts.

4.3.1 Write the following queries in MySQL workbench, and name the file queries.sql. The file you submit should have the exact name for you to get credit. We will run your query, so you don't need to capture the output. The file should include only the 10 queries. Be sure to test it before submission.

Example Query for your reference:

select count(*) from nj_state_teachers_salaries.salaries;

Note : Please include the name of the database and the table in each query as shown in the above example. End each query with a semicolon as shown in example.Your file queries.sql should be able to execute any any machine that has the nj_state_teachers_salaries database and the salaries table. We will deduct upto 10 pts if queries.sql does not execute.

[ ]:

4.3.2 1. Calculate the average salary

4.3.3 2. Calculate the number of people whose salary is more than 150,000.

4.3.4 3. Get the last name of the ones who make more than 150,000 but have less than 5 years of total experience

[ ]:

4.3.5 4. Get the highest salary for Preschool, School Counselor, Principal (anyone with the word Principal in the title), School Psychologist, and Kindergarten. (These are individual queries. You should have 5 separate queries.)

[ ]:

4.3.6 5. Get the last name, first name, and salary of the lowest earner who works in Atlantic City

4.3.7 6. Get the total number of employees working in Passaic City with more than ten years of total experience.

[ ]:

[ ]:

4.4 Submission on Gradescope

Gradescope canvas left menu -> Gradescop -> PSET 4: Managing Data Excercise 2

Submission :

Part -1 : This jupyter notebook, and a pdf of this notebook.

Part -2 : output.sql and sample.csv

Part -3 : queries.sql containing all your queries. This file should only include the sql queries. Please don't include the code that created the salaries table.

To create a pdf of this notebook : In your browser open print, and save as pdf. Name the pdf LastNameFirstName.pdf example: DoeJohn.pdf

Name this jupyter notebook with the same format LastNameFirstName.ipynb

Make sure that your notebook has been run before creating pdf. Any outputs from running the code needs to be clearly visible. We need all the files from Part-1, Part-2, and Part-3 to assign you grades.

Drop all the files in gradescope under PSET 4: Managing Data Exercise 2.

[ ]:

4.4.1 Submission Note (Please read)

After submitting your files on Gradescope , You may an error that says

"The autograder failed to execute correctly. Contact your course staff for help in debugging this issue. Make sure to include a link to this page so that they can help you most effectively."

4.4.2 You don't have to take any action , and you do not need to contact us. The error is beacuse of some internal setup on Gradescope. As long as you have followed the specs, and submitted all the required files, you are good.

[ ]: