PSET-4 - Managing Data Excercise 2

March 6, 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.

[ ]:

[174]:
```
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.

[175]:
```
df = pd.read_csv("nj_teachers_salaries_pset4.csv")
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
```

```
/var/folders/14/m1yk_rm10nx5rd9pwp0n6qtc0000gn/T/ipykernel_1491/2370806246.py:1:
DtypeWarning: Columns (7,8,13,14,15) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv("nj_teachers_salaries_pset4.csv")
```

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')

[176]:
```
print("Before dropping rows that have values as NaN:", df.shape)
df.dropna(how='all', inplace=True)
print("After dropping rows that have values as NaN:", df.shape)
```

```
Before dropping rows that have values as NaN: (100005, 16)
After dropping rows that have values as NaN: (100003, 16)
```

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.

[177]:
```
num_cols = num_cols = ["fte", "salary", "experience_district", "experience_nj",␣
 ↪"experience_total"]
print("A few rows before cleaning numeric columns:")
display(df[num_cols].head(7))
```

```
A few rows before cleaning numeric columns:
```

	fte	salary	experience_district	experience_nj	experience_total
0	1.0	98774	9.0	9.0	9
1	1.0	118415	13.0	13.0	13
2	1.0	57919	7.0	7.0	7
3	0.8	107746	26.0	26.0	26
4	0.8	54277	5.0	5.0	5
5	0.5	82772	1.0	20.0	20
6	0.8	51379	39.0	39.0	39

[178]:
```
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

```
before_drop = df.shape
df.dropna(subset=num_cols, inplace=True)
after_drop = df.shape

print("Before dropping invalid numeric rows:", before_drop)
print("After dropping invalid numeric rows:", after_drop)
```

```
Before dropping invalid numeric rows: (100003, 16)
After dropping invalid numeric rows: (99959, 16)
```

[179]:
```
df["experience_total"] = df["experience_total"].astype(int)
df["experience_district"] = df["experience_district"].astype(int)
df["experience_nj"] = df["experience_nj"].astype(int)
df["salary"] = df["salary"].astype(float)
df["fte"] = df["fte"].astype(float)
```

[180]:
```
missing_id = df["id"].isna()
print("Number of rows with missing id:", missing_id.sum())

if missing_id.sum() > 0:
    print("Rows with missing id before fixing:")
    display(df.loc[missing_id, ["id"]].head(5))
```

```
Number of rows with missing id: 5
Rows with missing id before fixing:
```

	id
30000	NaN
30001	NaN
30002	NaN
30003	NaN
30004	NaN

[181]:
```
missing_ids = df.index[df["id"].isna()].tolist()
for x in missing_ids:
    # Identify the row above
    x_above = x - 1
    id_above = df.at[x_above, "id"]
    df.at[x, "id"] = int(id_above + 1)
```

[182]:
```
df["id"] = df["id"].astype(int)
print("Rows that had missing ID after fixing:")
fixed_ids = [x for x in missing_ids if x in df.index]
display(df.loc[fixed_ids, ["id"]].head(5))
```

```
Rows that had missing ID after fixing:
```

	id
30000	30000
30001	30001

```
30002   30002
30003   30003
30004   30004
```

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.

[183]:
```
string_cols = df.select_dtypes(include="object").columns
example_cols = ["first_name"]  #using the hint
print("\nBEFORE removing leading/trailing spaces:")
for col in example_cols:
    if col in string_cols:
        spaces = df[col].str.match(r"^\s+.*|.*\s+$", na=False)
        if spaces.any():
            print("Examples in column", col,":")
            display(df.loc[spaces, [col]].head(2))
```

```
BEFORE removing leading/trailing spaces:
Examples in column first_name :
```

	first_name
40000	Christopher
40001	Angela

[184]:
```
for col in string_cols:
    df[col] = df[col].str.strip()


    for col in example_cols:
        if col in string_cols:
        # Re-check the same rows as before
            spaces = df[col].str.match(r"^\s+.*|.*\s+$", na=False)
            if spaces.any():
                print("Examples in column",col," after cleaning:")
                display(df.loc[spaces, [col]].head(2))
```

```
Examples in column first_name  after cleaning:
```

	first_name

```
40000   Christopher
40001         Angela
```

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.

[185]:
```
allowed_chars =␣
 ↪set("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789&/()-:'_.
 ↪,?!@# ""\\")
def remove_special_chars(text):
    if pd.isna(text):
        return text  #NaNs

    cleaned = []
    for ch in text:
        if ch in allowed_chars:
            cleaned.append(ch)
        #skip if not in allowed_chars
    return "".join(cleaned)
```

[186]:
```
def has_weird_chars(text):
    if pd.isna(text):
        return False
    return any(ch not in allowed_chars for ch in text)

weird_before = {}
for col in string_cols:
    weird = df[col].apply(has_weird_chars)
    if weird.any():
        weird_rows = df.loc[weird, [col]]
        weird_before[col] = weird_rows
        print("Examples of rows with special/invalid characters in ", col, "␣
 ↪before cleaning:")
        display(df.loc[weird, [col]].head(3))
```

Examples of rows with special/invalid characters in last_name before cleaning:

	last_name
50000	Rodgers â€
50001	Kline â€
50002	Cox â€

Examples of rows with special/invalid characters in school before cleaning:

	school
1849	Salome UreÃfÂ±a Elementary School
17420	Salome UreÃfÂ±a Elementary School
20127	Salome UreÃfÂ±a Elementary School

Examples of rows with special/invalid characters in primary_job before cleaning:

	primary_job
3023	Family & Consumer Sciences Ã¢â‚¬â€œ Apparel, T…
3195	Family & Consumer Sciences Ã¢â‚¬â€œ Apparel, T…
3212	Family & Consumer Sciences Ã¢â‚¬â€œ Apparel, T…

Examples of rows with special/invalid characters in subcategory before cleaning:

	subcategory
70000	Special ed ãéü
70001	General ed ãéü
70002	Special ed ãéü

Examples of rows with special/invalid characters in teaching_route before cleaning:

	teaching_route
60000	Traditional éü 0.5556085214578451
60001	Traditional éü 0.7761819498792722
60002	Alternate éü 0.11094518514028973

[187]:
```
for col in string_cols:
    df[col] = df[col].apply(remove_special_chars)
```

[188]:
```
for col in string_cols:
    if col in weird_before:
        same_rows = weird_before[col]
        if len(same_rows) > 0:
            print("Rows in ", col," after cleaning:")
            display(df.loc[same_rows.index, [col]].head(3))
```

```
Rows in  last_name  after cleaning:
```

	last_name
50000	Rodgers
50001	Kline
50002	Cox

```
Rows in  school  after cleaning:
```

	school
1849	Salome Urea Elementary School
17420	Salome Urea Elementary School
20127	Salome Urea Elementary School

```
Rows in  primary_job  after cleaning:
```

		primary_job
3023	Family & Consumer Sciences	Apparel, Textiles …
3195	Family & Consumer Sciences	Apparel, Textiles …
3212	Family & Consumer Sciences	Apparel, Textiles …

```
Rows in  subcategory  after cleaning:
```

	subcategory
70000	Special ed
70001	General ed
70002	Special ed

```
Rows in  teaching_route  after cleaning:
```

		teaching_route
60000	Traditional	0.5556085214578451
60001	Traditional	0.7761819498792722
60002	Alternate	0.11094518514028973

[189]:
```
for col in string_cols:
    empty = (df[col] == "")
    if empty.any():
        print(empty.sum()," rows in ", col, " became empty after cleaning")
        df.loc[empty, col] = np.nan

df.dropna(subset=string_cols, inplace=True)
```

1.7 Question-6 (1 pts)

1.7.1 Drop any duplicate rows. Display df.info() to shows the data types, and Non-Null count.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

[190]:
```
print("Before dropping duplicates:", df.shape)

df.drop_duplicates(inplace=True)

print("After dropping duplicates:", df.shape)
df.info()
```

```
Before dropping duplicates: (99959, 16)
After dropping duplicates: (99959, 16)
<class 'pandas.core.frame.DataFrame'>
Index: 99959 entries, 0 to 100004
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   99959 non-null  int64
 1   last_name            99959 non-null  object
 2   first_name           99959 non-null  object
 3   county               99959 non-null  object
 4   district             99959 non-null  object
 5   school               99959 non-null  object
 6   primary_job          99959 non-null  object
 7   fte                  99959 non-null  float64
 8   salary               99959 non-null  float64
 9   certificate          99959 non-null  object
 10  subcategory          99959 non-null  object
 11  teaching_route       99959 non-null  object
 12  highly_qualified     99959 non-null  object
 13  experience_district  99959 non-null  int64
 14  experience_nj        99959 non-null  int64
 15  experience_total     99959 non-null  int64
dtypes: float64(2), int64(4), object(10)
memory usage: 15.0+ MB
```

1.8 Question -7

1.8.1 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)

[191]:
```
df.to_csv("cleaned_data.csv", index=False)
```

1.9 Question -8

1.9.1 Similar to PSET-3

1.9.2 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)

1.9.3 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

1.9.4 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)

1.9.5 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)

[192]:
```
mydb = sq.connect(
    host="localhost",
    user="root",
    password="idp/dt=[H,p]",
    allow_local_infile=True,
)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE IF NOT EXISTS nj_state_teachers_salaries")
mycursor.execute("USE nj_state_teachers_salaries")
```

[193]:
```
drop_table = "DROP TABLE IF EXISTS teachers_salaries_pset4"
mycursor.execute(drop_table)
create_table = """
CREATE TABLE IF NOT EXISTS teachers_salaries_pset4 (
    id INT PRIMARY KEY,
    last_name VARCHAR(255),
    first_name VARCHAR(255),
    county VARCHAR(255),
    district VARCHAR(255),
    school VARCHAR(255),
    primary_job VARCHAR(255),
    fte FLOAT,
    salary FLOAT,
    certificate VARCHAR(255),
    subcategory VARCHAR(255),
    teaching_route VARCHAR(255),
    highly_qualified VARCHAR(255),
    experience_district INT,
    experience_nj INT,
    experience_total INT
```

```
);
"""
mycursor.execute(create_table)
```

[194]:
```
mycursor.execute("SET GLOBAL local_infile = 1")
mydb.commit()
load_data = """
LOAD DATA LOCAL INFILE 'cleaned_data.csv'
INTO TABLE teachers_salaries_pset4
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
"""
mycursor.execute(load_data)
mydb.commit()
```

[195]:
```
mycursor.execute("SELECT COUNT(*) FROM teachers_salaries_pset4")
print(f"Number of rows in table: {mycursor.fetchone()[0]}")
```

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

1.9.6 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.

1.10 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.

[196]:
```
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
```

[197]:
```
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

[198]:
```
random_sample = f"""
USE nj_state_teachers_salaries;

SELECT
    'id' AS id,
    'last_name' AS last_name,
    'first_name' AS first_name,
    'county' AS county,
    'district' AS district,
    'school' AS school,
    'primary_job' AS primary_job,
    'fte' AS fte,
    'salary' AS salary,
    'certificate' AS certificate,
    'subcategory' AS subcategory,
    'teaching_route' AS teaching_route,
```

```
    'highly_qualified' AS highly_qualified,
    'experience_district' AS experience_district,
    'experience_nj' AS experience_nj,
    'experience_total' AS experience_total
UNION ALL
SELECT
    id,
last_name,
first_name,
county,
district,
school,
primary_job,
fte,
salary,
certificate,
subcategory,
teaching_route,
highly_qualified,
experience_district,
experience_nj,
experience_total
FROM teachers_salaries_pset4
ORDER BY RAND(7)
LIMIT 777
INTO OUTFILE '/Users/alf/Documents/sample.csv'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\\n';
"""
mycursor.execute(random_sample, multi=True)
mydb.commit()

with open("output.sql", "w") as f:
    f.write(random_sample)
```

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.

[199]:
```
df_sample = pd.read_csv("sample.csv")

print("FIRST 5 ROWS:")
display(df_sample.head())

print("\nLAST 5 ROWS:")
```

```
display(df_sample.tail())


print("DataFrame shape (rows, columns):", df_sample.shape)
```

```
FIRST 5 ROWS:
```

```
      id last_name first_name      county                 district  \
0  54085     Foley        Ann   Middlesex  Middlesex Co Vocational
1    616   Collins     Joshua      Camden             Camden City
2  83855   Parsons       Ryan    Atlantic          Egg Harbor Twp
3  63733    Larson   Samantha  Gloucester          Washington Twp
4  53610    Deleon      Anita      Camden          Gloucester Twp

                                          school  \
0  Middlesex County Voc Academy Math Science &eng…
1                                 District Office
2                                 District Office
3                     Hurffville Elementary School
4                          Chews Elementary School

                                  primary_job  fte    salary  \
0                     Resource Program In-class  0.5  119914.0
1  Assistant Superintendent Curriculum Instruction  0.8   80718.0
2                Elementary Kindergraten-8 Grade  1.0   53462.0
3                Elementary Kindergraten-8 Grade  0.8  101684.0
4        Learning Disabilities Teacher Consultant  0.8   55259.0

           certificate subcategory teaching_route  \
0  Standard certificate  Special ed      Alternate
1                  CEAS  General ed    Traditional
2                  CEAS  Special ed      Alternate
3  Standard certificate  Special ed      Alternate
4                  CEAS  Special ed      Alternate

                     highly_qualified  experience_district  experience_nj  \
0                Not highly qualified                    8             15
1                Not highly qualified                   20             20
2                Not highly qualified                   20             20
3                Not highly qualified                    5             15
4  Doesn't need to be highly qualified                   23             15

   experience_total
0                26
1                20
2                20
3                15
4                25
```

LAST 5 ROWS:

```
       id last_name first_name    county                   district  \
772  83399     Perez     Ronald     Essex               Millburn Twp
773  61038    Kramer    Phillip    Mercer               Hamilton Twp
774  77525  Thompson        Ian    Morris  Parsippany-troy Hills Twp
775   6462      Lowe       Mary     Essex         City Of Orange Twp
776  91113    Harris      Shane  Atlantic     Greater Egg Harbor Reg

                             school  \
772           Millburn Middle School
773             Hamilton West-watson
774            Central Middle School
775  Lincoln Avenue Elementary School
776             Absegami High School

                                  primary_job  fte    salary  \
772                     English Non-elementary  1.0   90428.0
773                   Mathematics Grades 5 - 8  0.5   92801.0
774  Non-supervisory Coordinator Of Basic Skills  0.8  109317.0
775           Director Curriculum & Instruction  0.5   85641.0
776               Elementary School Teacher K-5  1.0   50913.0

            certificate subcategory teaching_route  \
772                CEAS  General ed    Traditional
773                CEAS  General ed      Alternate
774  Standard certificate  General ed    Traditional
775  Standard certificate  Special ed    Traditional
776                CEAS  General ed      Alternate

                      highly_qualified  experience_district  experience_nj  \
772  Doesn't need to be highly qualified                   29              0
773  Doesn't need to be highly qualified                    5             28
774  Doesn't need to be highly qualified                   28             11
775  Doesn't need to be highly qualified                   10             10
776  Doesn't need to be highly qualified                    3              3

     experience_total
772                37
773                25
774                35
775                10
776                 3
```

DataFrame shape (rows, columns): (777, 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.

[200]:
```
mycursor.execute("DROP TABLE IF EXISTS salaries")
create_salaries = """
CREATE TABLE salaries (
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    primary_job VARCHAR(255),
    experience_total INT,
    experience_district INT,
    experience_current_job INT,
    salary FLOAT,
    days_worked FLOAT,
    teaching_route VARCHAR(100),
    subcategory VARCHAR(255)
    -- Add or adjust columns if your CSV has more/less columns
);
"""
mycursor.execute(create_salaries)

load_data = """
LOAD DATA LOCAL INFILE 'teachersample.csv'
INTO TABLE salaries
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\\n'
IGNORE 1 LINES
(
  first_name,
  last_name,
  primary_job,
  experience_total,
```

```
  experience_district,
  experience_current_job,
  salary,
  days_worked,
  teaching_route,
  subcategory
);
"""
mycursor.execute(load_data)
mydb.commit()
```

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.

[201]:
```
queries_text = """
USE nj_state_teachers_salaries;
"""
```

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

4.3.6 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.)

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

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

[202]:
```
%%writefile queries.sql
-- Query 1: Calculate the average salary.
SELECT AVG(salary)
FROM nj_state_teachers_salaries.salaries;

-- Query 2: Calculate the number of people whose salary is more than 150,000.
SELECT COUNT(*)
FROM nj_state_teachers_salaries.salaries
WHERE salary > 150000;

-- Query 3: Get the last name of the ones who make more than 150,000 but have␣
 ↪less than 5 years of total experience.
SELECT last_name
FROM nj_state_teachers_salaries.salaries
WHERE salary > 150000
  AND experience_total < 5;

-- Query 4: Get the highest salary for Preschool.
SELECT MAX(salary)
FROM nj_state_teachers_salaries.salaries
WHERE primary_job LIKE '%Preschool%';

-- Query 5: Get the highest salary for School Counselor.
SELECT MAX(salary)
FROM nj_state_teachers_salaries.salaries
WHERE primary_job LIKE '%School Counselor%';

-- Query 6: Get the highest salary for Principal (any title containing␣
 ↪"Principal").
SELECT MAX(salary)
```

```
FROM nj_state_teachers_salaries.salaries
WHERE primary_job LIKE '%Principal%';

-- Query 7: Get the highest salary for School Psychologist.
SELECT MAX(salary)
FROM nj_state_teachers_salaries.salaries
WHERE primary_job LIKE '%School Psychologist%';

-- Query 8: Get the highest salary for Kindergarten.
SELECT MAX(salary)
FROM nj_state_teachers_salaries.salaries
WHERE primary_job LIKE '%Kindergarten%';

-- Query 9: Get the last name, first name, and salary of the lowest earner who␣
 ↪works in Atlantic City.
SELECT last_name, first_name, salary
FROM nj_state_teachers_salaries.salaries
WHERE city = 'Atlantic City'
ORDER BY salary ASC
LIMIT 1;

-- Query 10: Get the total number of employees working in Passaic City with␣
 ↪more than ten years of total experience.
SELECT COUNT(*)
FROM nj_state_teachers_salaries.salaries
WHERE city = 'Passaic City'
  AND experience_total > 10;
```

```
Overwriting queries.sql
```

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.

[ ]: