Deepnote CMSC 320 / HomeWork3 Published at Feb 24, 2023 Unlisted

Homework #3: SQL

To start with, if you felt the class was unclear, check out the following tutorial: https://mode.com/sql-tutorial/introduction-to-sql/

Now! We'll be using sqlite to access a database. Start by downloading the sql lite file and putting it in the same directory as this notebook: https://www.kaggle.com/datasets/kaggle/sf-salaries (hit the 'download' button in the upper right). Check out the description of the data so you know the table / column names.

The following code will use sqlite to create a database connection.

```
import sqlite3
import pandas as pd

conn = sqlite3.connect("database.sqlite")
crsr = conn.cursor()
```

Exploration

Problem 1:

Try to create a query that gives you a data frame of the EmployeeName, JobTitle, and BasePay from the salaries table.

```
query = 'SELECT EmployeeName,JobTitle,BasePay FROM salaries'

df = pd.read_sql(query, conn)
df.head()
```

	EmployeeName o...	JobTitle object	BasePay object
0	NATHANIEL FORD	GENERAL MANAGER-...	167411.18
1	GARY JIMENEZ	CAPTAIN III (POLICE...	155966.02
2	ALBERT PARDINI	CAPTAIN III (POLICE...	212739.13
3	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE...	77916
4	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,...	134401.6

Problem 2.

Modify your query to limit it to the year 2012.

```
query = 'SELECT * FROM salaries  WHERE Year =  2012'

df = pd.read_sql(query, conn)
df.head()
```

	Id int64	EmployeeName o...	JobTitle object	BasePay float64	OvertimePay floa...	OtherPay float64	Benefits float64	TotalPay float64
0	36160	Gary Altenberg	Lieutenant, Fire Suppression	128808.87	220909.48	13126.31	44430.12	362844.6
1	36161	Gregory Suhr	Chief of Police	302578.0	0.0	18974.11	69810.19	321552.1
2	36162	Khoa Trinh	Electronic Maintenance Tech	111921.0	146415.32	78057.41	53102.29	336393.7

3	36163	Joanne Hayes-White	Chief, Fire Department	296943.01	0.0	17816.59	72047.88	314759.
4	36164	Frederick Binkley	EMT/Paramedic/Firefighter	126863.19	192424.49	17917.18	44438.25	337204.8

Problem 3:

Further limit the table to the year 2012, employees making under 150,000, and sort in descending order by year.

```
query = 'SELECT * FROM salaries    WHERE TotalPay < 150000 AND Year =  2012 ORDER BY Year DESC'

df = pd.read_sql(query, conn)
df.head()
```

	Id int64	EmployeeName o...	JobTitle object	BasePay float64	OvertimePay floa...	OtherPay float64	Benefits float64	TotalPay float64
1	37999	Tristan Levardo	Manager IV	145592.63	0.0	1500.0	61386.3	147092.63
0	37556	Marcia Bell	Law Librarian	140485.6	0.0	8387.1	69196.17	148872.7
3	38049	Gerardo Fries	Manager IV	145448.07	0.0	3486.0	58289.15	148934.07
2	38041	Paul Gambon	Manager V	145592.65	0.0	3486.0	58293.37	149078.65
4	38054	Masood Ordikhani	Manager IV	145606.89	0.0	3486.0	58007.96	149092.89

Aggregation

Problem 4:

Select the average base pay from the table.

```
query = 'SELECT AVG(BasePay) FROM Salaries'

df = pd.read_sql(query, conn)
df.head()
```

	AVG(BasePay) fl...
0	66053.72928809702

Problem 5:

Produce and print the head of a dataframe that shows the average pay for each year (only use a single, simple query). Your result should have a column for the year and a column for the average base pay.

```
query = 'SELECT Year, AVG(BasePay) FROM Salaries GROUP BY Year'

df = pd.read_sql(query, conn)
df.head()
```

	Year int64	AVG(BasePay) fl...
0	2011	63595.956516774524
1	2012	65436.40685742255
2	2013	68509.83215550765

3	2014	66557.4377499144 8

Problem 6:

Create a dataframe with average base pay, benefits, and overtime for each job title, as well as a column with the total average.

```
query = 'SELECT AVG(BasePay),  AVG(Benefits), AVG(OvertimePay) FROM Salaries GROUP BY JobTitle'
df = pd.read_sql(query, conn)
df["TotalAveragePay"] = df['AVG(BasePay)'] + df['AVG(OvertimePay)'] + df['AVG(Benefits)']
df.head()
```

	AVG(BasePay) fl...	AVG(Benefits) flo...	AVG(OvertimePa...	TotalAveragePay f..
0	43300.806506024 106	0.0	373.20084337349 4	43674.007349397 6
1	46643.172	0.0	0.0	46643.172
2	28732.663958333 33	0.0	24.430625000000 003	28757.094583333 33
3	62290.78	17975.59	0.0	80266.37
4	66374.4	0.0	0.0	66374.4

Table Creation

Problem 7:

Now we'll create our own table in our database. Separate the Salaries table by Year, and add it back to the database

```
for y in ['2011','2012','2013','2014']:
    query = "SELECT * FROM salaries GROUP BY Year"

    df = pd.read_sql(query, conn)
    df.to_sql(name='Y'+y, con=conn, if_exists='replace')
```

Table Joining

Problem 8:

We'll move on to a new dataset for the next steps. Download the dataset from here (https://www.kaggle.com/datasets/luizpaulodeoliveira/imdb-project-sql) and load the sqlite file same as before. Start by just selecting everything in the movie to see what it looks like.

```
conn = sqlite3.connect("movies.sqlite")
query = 'SELECT * FROM movies'

df = pd.read_sql(query, conn)
df.head()
```

	id int64	original_title object	budget int64	popularity int64	release_date obj...	revenue int64	title object	vote_average flo...
0	43597	Avatar	237000000	150	2009-12-10	2787965087	Avatar	7.2
1	43598	Pirates of the Caribbean: At...	300000000	139	2007-05-19	961000000	Pirates of the Caribbean: At...	6.9
2	43599	Spectre	245000000	107	2015-10-26	880674609	Spectre	6.3
3	43600	The Dark Knight Rises	250000000	112	2012-07-16	1084939099	The Dark Knight Rises	7.6
4	43601	John Carter	260000000	43	2012-03-07	284139100	John Carter	6.1

Problem 9:

Create a dataframe that includes the entire contents of movies as well as the director's name.

```
query = 'SELECT movies.*, directors.name FROM movies JOIN directors ON movies. director_id = directors.id'
df = pd.read_sql(query, conn)

print(df.head(5))
df.size
```

```
3   2012-07-16  1084939099                     The Dark Knight Rises
4   2012-03-07   284139100                               John Carter

   vote_average  vote_count  \
0           7.2       11800
1           6.9        4500
2           6.3        4466
3           7.6        9106
4           6.1        2124

                                            overview  \
0  In the 22nd century, a paraplegic Marine is di...
1  Captain Barbossa, long believed to be dead, ha...
2  A cryptic message from Bond’s past sends him o...
3  Following the death of District Attorney Harve...
4  John Carter is a war-weary, former military ca...

                                          tagline     uid  director_id  \
0                     Enter the World of Pandora.   19995         4762
1  At the end of the world, the adventure begins.     285         4763
2                           A Plan No One Escapes  206647         4764
3                                 The Legend Ends   49026         4765
4            Lost in our world, found in another.   49529         4766

                name
0      James Cameron
1     Gore Verbinski
2         Sam Mendes
3  Christopher Nolan
4     Andrew Stanton
```

```
66822
```

Analysis

The next few problems will be more involved! You'll need to combine some concepts you've learned. For each cell, show your work.

Problem 10:

What is the average budget used for the top 10 grossing movies?

```
query = 'SELECT budget FROM movies ORDER by revenue DESC   LIMIT 10;'

df = pd.read_sql(query, conn)


df['budget'].mean()
```

```
195100000.0
```

Problem 11:

Which directors have the highest voting average? - show the top 5 directors' name and their average rating

```
query = '''
        SELECT directors.name, movies.vote_average AS avg_rating
        FROM movies
        JOIN directors ON movies.director_id = directors.id
        GROUP BY movies.director_id
        ORDER BY avg_rating DESC
        LIMIT 5

        '''
df = pd.read_sql(query, conn)

df
```

	name object	avg_rating float64
0	Gary Sinyor	10.0
1	Rohit Jugraj	9.5
2	Lance Hool	9.3
3	Floyd Mutrux	8.5
4	John Cromwell	8.4

Problem 12:

What are the top five directors by average budget?

```
query = '''
        SELECT AVG(movies.budget) AS avg_budget, directors.name
        FROM movies
        JOIN directors ON movies.director_id = directors.id
        GROUP BY directors.name
        ORDER BY AVG(movies.budget) DESC

        LIMIT 5 '''
df = pd.read_sql(query, conn)

print(df.head(5))
```

```
     avg_budget            name
0  2.600000e+08    Byron Howard
1  2.000000e+08     Dan Scanlon
2  2.000000e+08     Lee Unkrich
3  1.933333e+08     David Yates
4  1.850000e+08  Brenda Chapman
```