Collect Data From SQLite Databases

What is SQLite

A file with the .sqlite extension is a lightweight SQL database file created with the SQLite software. It is a database in a file itself and implements a self-contained, full-featured, highly-reliable SQL database engine.

We use SQLite to demonstrate the approach to access SQL databases. They follow similar steps. You just need to setup your account credentials in the `connect` so you can connect the server.

Read an SQLite Database in Python

We use a Python package, sqlite3, to deal with SQLite databases. Once the sqlite3 package is imported, the general steps are:

1. Create a connection object that connects the SQLite database.
2. Create a cursor object
3. Create a query statement
4. Execute the query statement
5. Fetch the query result to result
6. If all work is done, close the connection.

We use the built-in SQLite database Chinook as the example here. We connect with the database, and show all the tables it contains.

```
import sqlite3

connection = sqlite3.connect('/content/ds_salaries.sqlite')
cursor = connection.cursor()

query = '''
SELECT name FROM sqlite_master
WHERE type='table';
'''

cursor.execute(query)
results = cursor.fetchall()
results
```

```
[('ds_salaries',)]
```

Play with the SQLite Databases

Using SQL statements, you can play with the SQLite Databases and get the data you need.

```
query = '''SELECT *
FROM ds_salaries'''

cursor.execute(query)
results = cursor.fetchall()
results
```

```
[(None,
  'work_year',
  'experience_level',
  'employment_type',
  'job_title',
  'salary',
  'salary_currency',
  'salary_in_usd',
  'employee_residence',
  'remote_ratio',
  'company_location',
  'company_size'),
 (0,
  '2020',
  'MI',
  'FT',
  'Data Scientist',
  '70000',
  'EUR',
  '79833',
```

```
 '79833',
 'DE',
 '0',
 'DE',
 'L'),
(1,
 '2020',
 'SE',
 'FT',
 'Machine Learning Scientist',
 '260000',
 'USD',
 '260000',
 'JP',
 '0',
 'JP',
 'S'),
(2,
 '2020',
 'SE',
 'FT',
 'Big Data Engineer',
 '85000',
 'GBP',
 '109024',
 'GB',
 '50',
 'GB',
 'M'),
(3,
 '2020',
 'MI',
 'FT',
 'Product Data Analyst',
 '20000',
 'USD',
 '20000',
 'HN',
 '0',
 'HN',
 'S')
```

Save Data to CSV Files

Since CSV file is much more convenient to process, we still use pandas to convert and to write to CSV files.

```
import pandas as pd

df = pd.DataFrame(results)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 608 entries, 0 to 607
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       607 non-null    float64
 1   1       608 non-null    object 
 2   2       608 non-null    object 
 3   3       608 non-null    object 
 4   4       608 non-null    object 
 5   5       608 non-null    object 
 6   6       608 non-null    object 
 7   7       608 non-null    object 
 8   8       608 non-null    object 
 9   9       608 non-null    object 
 10  10      608 non-null    object 
 11  11      608 non-null    object 
dtypes: float64(1), object(11)
memory usage: 57.1+ KB
```

```
df.iloc[0]
```

```
0                    NaN
1              work_year
2       experience_level
3        employment_type
4              job_title
5                 salary
6        salary_currency
7          salary_in_usd
8     employee_residence
9           remote_ratio
10      company_location
11          company_size
Name: 0, dtype: object
```

```
cols = list(df.iloc[0])
cols
```

```
[nan,
 'work_year',
 'experience_level',
 'employment_type',
 'job_title',
 'salary',
 'salary_currency',
 'salary_in_usd',
 'employee_residence',
 'remote_ratio',
 'company_location',
 'company_size']
```

```
df.columns = cols
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 608 entries, 0 to 607
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   nan                 607 non-null    float64
 1   work_year           608 non-null    object 
 2   experience_level    608 non-null    object 
 3   employment_type     608 non-null    object 
 4   job_title           608 non-null    object 
 5   salary              608 non-null    object 
 6   salary_currency     608 non-null    object 
 7   salary_in_usd       608 non-null    object 
 8   employee_residence  608 non-null    object 
 9   remote_ratio        608 non-null    object 
 10  company_location    608 non-null    object 
 11  company_size        608 non-null    object 
dtypes: float64(1), object(11)
memory usage: 57.1+ KB
```

```
df.drop(0, inplace = True)
df
```

	NaN	work_year	experience_level	employment_type	job_title	salary	sal
1	0.0	2020	MI	FT	Data Scientist	70000	
2	1.0	2020	SE	FT	Machine Learning Scientist	260000	
3	2.0	2020	SE	FT	Big Data Engineer	85000	
4	3.0	2020	MI	FT	Product Data Analyst	20000	
5	4.0	2020	SE	FT	Machine Learning Engineer	150000	
...	...	...	...	...	...	...	
603	602.0	2022	SE	FT	Data Engineer	154000	
604	603.0	2022	SE	FT	Data Engineer	126000	
605	604.0	2022	SE	FT	Data Analyst	129000	
606	605.0	2022	SE	FT	Data Analyst	150000	
607	606.0	2022	MI	FT	AI Scientist	200000	

607 rows × 12 columns

```
cursor.close()
connection.close()
```

Colab paid products - Cancel contracts here