**Data Engineering Day 09**

**The credit for this course goes to Coursera. Click More**

**Another link : Azure data Engineer**

**Views in MySQL | Advanced Data Engineering**

## Using Line Magic Statements

| Line Magics | Uses |
|---|---|
| %pwd | prints the current working directory |
| %ls | lists all files in the current directory |
| %history | shows the command history |
| %reset | resets the namespace by removing all names defined by the user |
| %who | lists all variables in the namespace |
| %whos | provides more detailed information about all variables in the namespace |
| %matplotlib inline | makes matplotlib plots appear within the notebook |
| %timeit | times the execution of a single statement |
| %lsmagic | lists all available line magics |

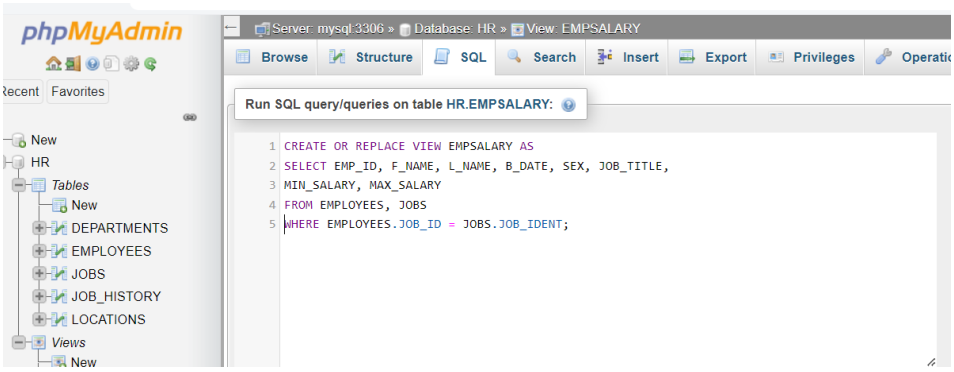
# Creating Views in SQL:

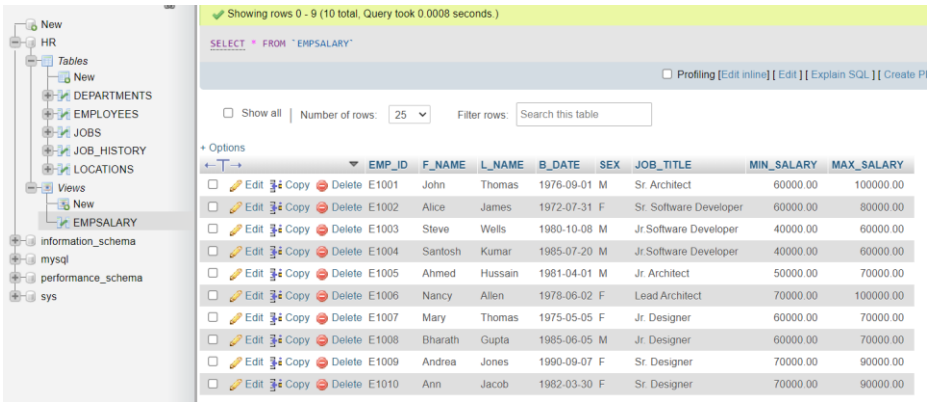
- Views are the virtual table in SQL.
- Views can be created within a table or more than one table.

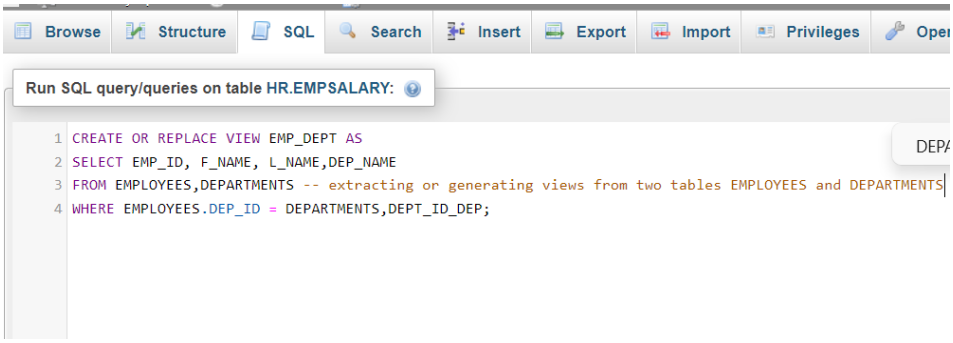
# advantages of creating views:

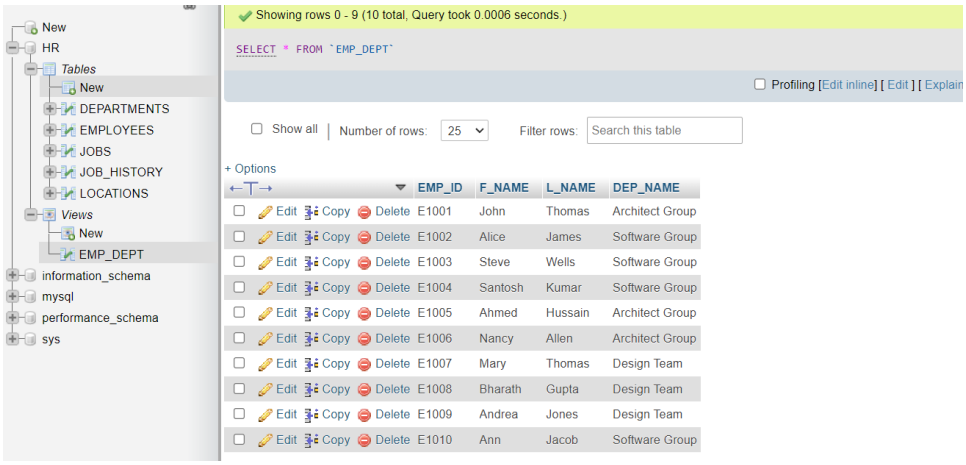
- Security
- Make complex analysis easy.
- Provide independent data.

Example of creating views:






# Stored Procedures:

- A sets of SQL statements stored and executed in SQL server.

# benefits of Stored Procedures:

- Network traffic reductions.
- Improvements in performances
- Reuses of codes.
- More secure

# Analyzing a real world data-set with SQL and Python

Estimated time needed: **15** minutes

## Objectives

After completing this lab you will be able to:

- Understand a dataset of selected socioeconomic indicators in Chicago
- Learn how to store data in an SQLite database.
- Solve example problems to practice your SQL skills

## Selected Socioeconomic Indicators in Chicago

The city of Chicago released a dataset of socioeconomic data to the Chicago City Portal. This dataset contains a selection of six socioeconomic indicators of public health significance and a “hardship index,” for each Chicago community area, for the years 2008 – 2012.

Scores on the hardship index can range from 1 to 100, with a higher index number representing a greater level of hardship.

A detailed description of the dataset can be found on the city of Chicago's website, but to summarize, the dataset has the following variables:

- **Community Area Number** ( `ca` ): Used to uniquely identify each row of the dataset
- **Community Area Name** ( `community_area_name` ): The name of the region in the city of Chicago
- **Percent of Housing Crowded** ( `percent_of_housing_crowded` ): Percent of occupied housing units with more than one person per room
- **Percent Households Below Poverty** ( `percent_households_below_poverty` ): Percent of households living below the federal poverty line
- **Percent Aged 16+ Unemployed** ( `percent_aged_16_unemployed` ): Percent of persons over the age of 16 years that are unemployed

- **Percent Aged 25+ without High School Diploma** ( `percent_aged_25_without_high_school_diploma` ): Percent of persons over the age of 25 years without a high school education
- **Percent Aged Under** 18 or Over 64:Percent of population under 18 or over 64 years of age ( `percent_aged_under_18_or_over_64` ): (ie. dependents)
- **Per Capita Income** ( `per_capita_income_` ): Community Area per capita income is estimated as the sum of tract-level aggragate incomes divided by the total population
- **Hardship Index** ( `hardship_index` ): Score that incorporates each of the six selected socioeconomic indicators

In this Lab, we'll take a look at the variables in the socioeconomic indicators dataset and do some basic analysis with Python.

## Connect to the database

Let us first load the SQL extension and establish a connection with the database

**The syntax for connecting to magic sql using sqllite is**

**%sql sqlite://DatabaseName**

where DatabaseName will be your **.db** file

In [1]:
```
%load_ext sql
```

In [4]:
```
import csv, sqlite3

con = sqlite3.connect("socioeconomic.db") # this code creates the connections with
cur = con.cursor()
!pip install -q pandas==1.1.5
```

In [3]:
```
%sql sqlite:///socioeconomic.db
```

Out[3]:
```
'Connected: @socioeconomic.db'
```

## Store the dataset in a Table

**In many cases the dataset to be analyzed is available as a .CSV (comma separated values) file, perhaps on the internet. To analyze the data using SQL, it first needs to be stored in the database.**

**We will first read the csv files from the given url into pandas dataframes**

**Next we will be using the df.to_sql() function to convert each csv file to a table in sqlite with the csv data loaded in it.**

In [27]:
```
import pandas
df = pandas.read_csv('https://data.cityofchicago.org/resource/jcxq-k9xf.csv')
```

```
df.to_sql("chicago_socioeconomic_data", con, if_exists='replace', index=False,metho
```

You can verify that the table creation was successful by making a basic query like:

In [14]:
```
%sql SELECT * FROM chicago_socioeconomic_data limit 5; ---this will render top 5 d
```

In [15]:
```
# %sql SELECT * FROM chicago_socioeconomic_data; ---all data will be displayed
```

# Problems

## Problem 1

How many rows are in the dataset?

In [16]:
```
# to check the numbers of rows in the data
%sql SELECT COUNT(*) FROM chicago_socioeconomic_data;
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[16]:

| COUNT(*) |
|---|
| 78 |

▶ Click here for the solution

## Problem 2

How many community areas in Chicago have a hardship index greater than 50.0?

In [17]:
```
%sql SELECT COUNT(*) FROM chicago_socioeconomic_data WHERE hardship_index > 50.0;
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[17]:

| COUNT(*) |
|---|
| 38 |

In [23]:
```
  # %sql SELECT COUNT(*) FROM chicago_socioeconomic_data WHERE hardship_index < 50.
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[23]:

| COUNT(*) |
|---|
| 38 |

In [22]:
```
  # %sql SELECT COUNT(*) FROM chicago_socioeconomic_data WHERE hardship_index BETWE
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[22]:

| COUNT(*) |
|---|
| 39 |

▶ Click here for the solution

## Problem 3

**What is the maximum value of hardship index in this dataset?**

In [26]:
```
%sql SELECT MAX(hardship_index) FROM chicago_socioeconomic_data;
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[26]:

| MAX(hardship_index) |
|---|
| 98.0 |

▶ Click here for the solution

## Problem 4

**Which community area which has the highest hardship index?**

In [28]:
```
%sql SELECT community_area_name FROM chicago_socioeconomic_data where hardship_inde
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[28]:

| community_area_name |
|---|
| Riverdale |

▶ Click here for the solution

## Problem 5

**Which Chicago community areas have per-capita incomes greater than $60,000?**

In [30]:
```
%sql SELECT community_area_name FROM chicago_socioeconomic_data WHERE per_capita_in
```

```
 * sqlite:///socioeconomic.db
Done.
```

Out[30]:

| community_area_name |
|---|
| Lake View |
| Lincoln Park |
| Near North Side |
| Loop |

▶ Click here for the solution

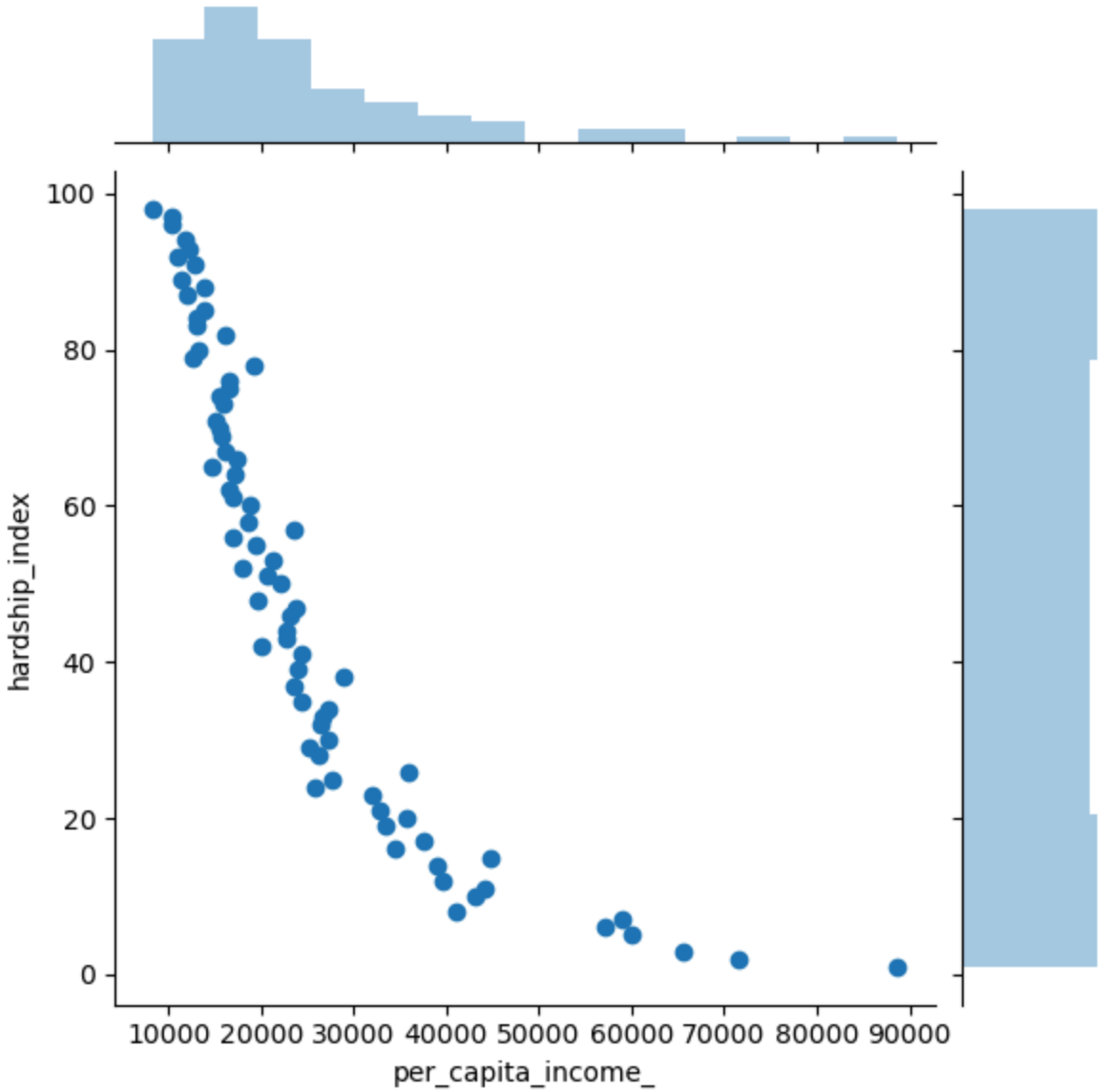
## Problem 6

**Create a scatter plot using the variables `per_capita_income_` and `hardship_index`. Explain the correlation between the two variables.**

In [37]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

income_vs_hardship = %sql SELECT per_capita_income_, hardship_index FROM chicago_so
plot = sns.jointplot(x='per_capita_income_',y='hardship_index',data=income_vs_hards
```

```
 * sqlite:///socioeconomic.db
Done.
```




▶ Click here for the solution

## Conclusion

Now that you know how to do basic exploratory data analysis using SQL and python visualization tools, you can further explore this dataset to see how the variable `per_capita_income_` is related to `percent_households_below_poverty` and `percent_aged_16_unemployed`. Try to create interesting visualizations!

# Summary

In this lab you learned how to store a real world data set from the internet in a database, gain insights into data using SQL queries. You also visualized a portion of the data in the database to see what story it tells.

# Author

Rav Ahuja

# Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2022-03-04 | 2.3 | Lakshmi Holla | Made changes in markdown cells |
| 2021-07-09 | 2.2 | Malika | Updated connection string |
| 2021-05-06 | 2.1 | Malika Singla | Added libraries |
| 2020-08-28 | 2.0 | Lavanya | Moved lab to course repo in GitLab |

---

© IBM Corporation 2020. All rights reserved.