**Data Analysis with Python**

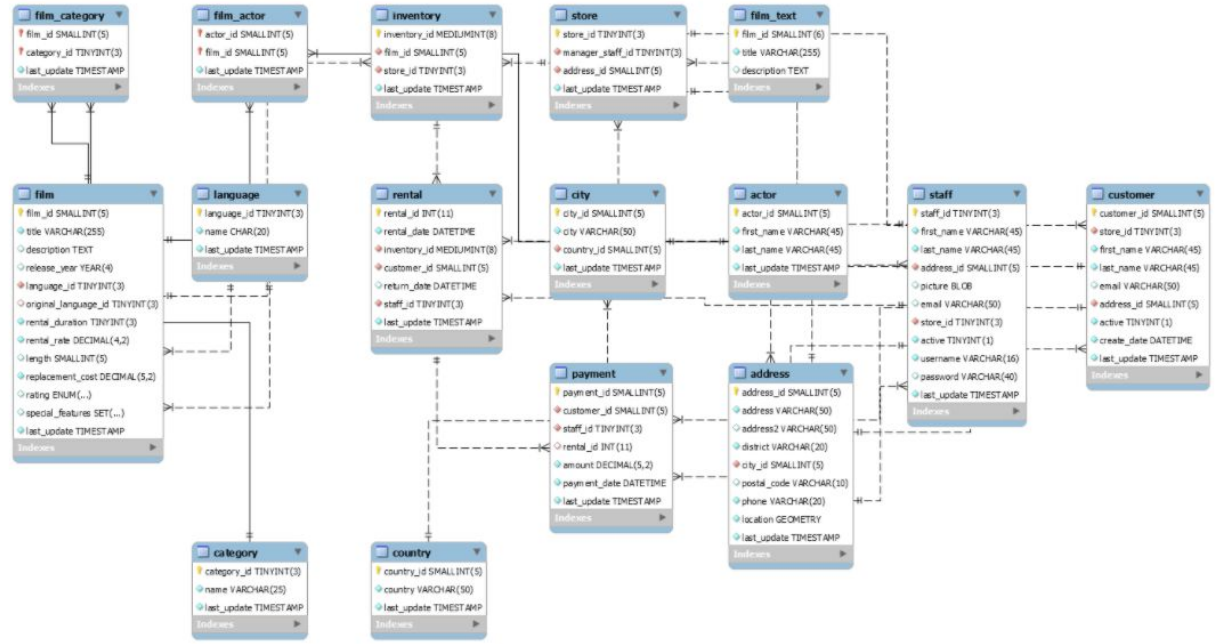
**The Sakila Database**

**https://dev.mysql.com/doc/sakila/en/**

**https://jupyter-notebook.readthedocs.io/en/stable/notebook.html#trusting-notebooks**

Recall: Standard Deviation and Variance and quantiles (25%, 50%, 75%) and boxplot and density (graphs / plots)

Now, instead of getting data from a csv file, will get data from a database



In [2]:
```
# The Sakila Database
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sqlite3

%matplotlib inline
```

In [4]:
```
import os
```

In [5]:
```
#os.getcwd()
os.chdir('C:\\Users\\maullonv\\Desktop\\dataAnalysisPython')
```

In [6]:
```
# Load the data

conn = sqlite3.connect('sakila.db')

df = pd.read_sql('''
    SELECT
        rental.rental_id, rental.rental_date, rental.return_date,
        customer.last_name AS customer_lastname,
        store.store_id,
        city.city AS rental_store_city,
        film.title AS film_title, film.rental_duration AS film_rental_duration,
        film.rental_rate AS film_rental_rate, film.replacement_cost AS film_replacement_cost,
        film.rating AS film_rating
    FROM rental
    INNER JOIN customer ON rental.customer_id == customer.customer_id
    INNER JOIN inventory ON rental.inventory_id == inventory.inventory_id
    INNER JOIN store ON inventory.store_id == store.store_id
    INNER JOIN address ON store.address_id == address.address_id
    INNER JOIN city ON address.city_id == city.city_id
    INNER JOIN film ON inventory.film_id == film.film_id
    ;
''', conn, index_col = 'rental_id', parse_dates = ['rental_date', 'return_date'])
```

- SQL queries
    - Basically pulling the data from the database
    - MySQL
    - Basically converted the data into a dataframe

## Understand the Structure of the Data

```
In [7]:
# The data at a glance.
df.head()
```

| rental_id | rental_date | return_date | customer_lastname | store_id | rental_store_city | film_title | film_rental_duration | fil |
|---|---|---|---|---|---|---|---|---|
| 1 | 2005-05-24 22:53:30 | 2005-05-26 22:04:30 | HUNTER | 1 | Lethbridge | BLANKET BEVERLY | 7 | 2.9 |
| 2 | 2005-05-24 22:54:33 | 2005-05-28 19:40:33 | COLLAZO | 2 | Woodridge | FREAKY POCUS | 7 | 2.9 |
| 3 | 2005-05-24 23:03:39 | 2005-06-01 22:12:39 | MURRELL | 2 | Woodridge | GRADUATE LORD | 7 | 2.9 |
| 4 | 2005-05-24 23:04:41 | 2005-06-03 01:43:41 | PURDY | 1 | Lethbridge | LOVE SUICIDES | 6 | 0.9 |
| 5 | 2005-05-24 23:05:21 | 2005-06-02 04:33:21 | HANSEN | 2 | Woodridge | IDOLS SNATCHERS | 5 | 2.9 |

df.head()

```
In [8]:
df.shape
# (# of rows, # of columns)

(16044, 10)
```

df.shape

```
In [9]:
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 16044 entries, 1 to 16049
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   rental_date           16044 non-null  datetime64[ns]
 1   return_date           15861 non-null  datetime64[ns]
 2   customer_lastname     16044 non-null  object
 3   store_id              16044 non-null  int64
 4   rental_store_city     16044 non-null  object
 5   film_title            16044 non-null  object
 6   film_rental_duration  16044 non-null  int64
 7   film_rental_rate      16044 non-null  float64
 8   film_replacement_cost 16044 non-null  float64
 9   film_rating           16044 non-null  object
dtypes: datetime64[ns](2), float64(2), int64(2), object(4)
memory usage: 1.1+ MB
```

df.info()

- More info on the rows and columns, data types

```
In [10]:
# Numerical analysis and visualization

# Analyze the film_rental_rate column
df['film_rental_rate'].describe()
```

```
count    16044.000000
mean         2.942630
std          1.649678
min          0.990000
25%          0.990000
50%          2.990000
75%          4.990000
max          4.990000
Name: film_rental_rate, dtype: float64
```

df['film_rental_rate'].describe()

- Statistical information about the data

```
In [11]:
df['film_rental_rate'].mean()
```

```
2.9426302667663933
```

df['film_rental_rate'].mean()

```
In [12]:
df['film_rental_rate'].median()
```

```
2.99
```

df['film_rental_rate'].median()

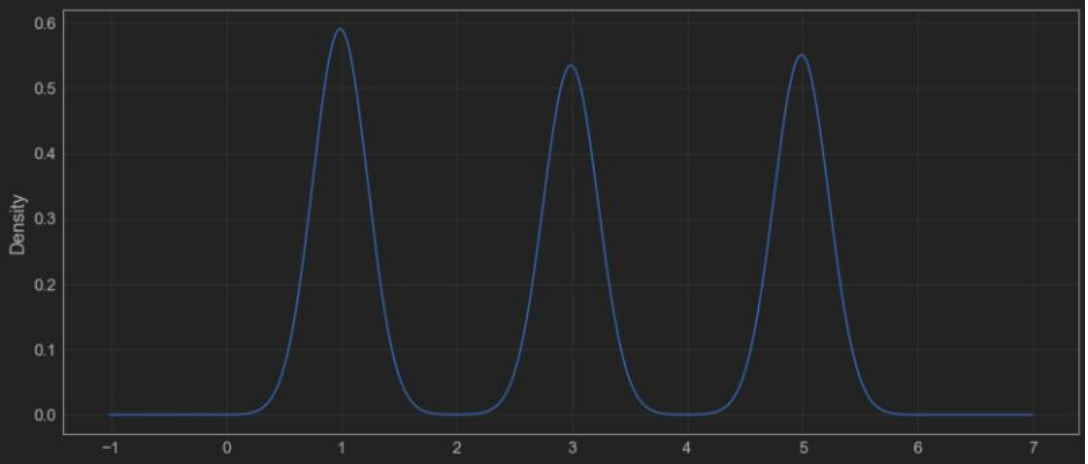
```
In [13]: from jupyterthemes import jtplot
         jtplot.style(theme='monokai')

In [14]: df['film_rental_rate'].plot(kind = 'box', vert = False, figsize = (14, 6))
```

```
<AxesSubplot:>
```


df[‘film_rental_rate’].plot(kind = ‘box’, vert = False, figsize = (14, 6))

```
In [15]: df['film_rental_rate'].plot(kind = 'density', figsize = (14, 6))
```

```
<AxesSubplot:ylabel='Density'>
```



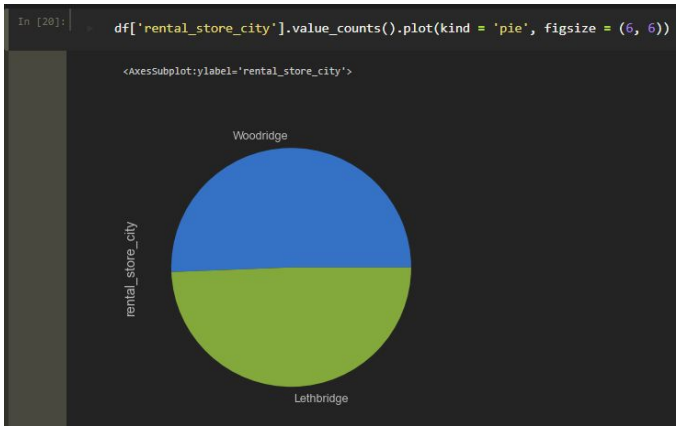

df[‘film_rental_rate’].plot(kind = ‘density’, figsize = (14, 6))

**Categorical Analysis regarding Distribution of *rental_store_city***

```
In [19]: # Categorical Analysis and Visualization

         # Will analyze rental_store_city column
         df['rental_store_city'].value_counts()

Woodridge     8121
Lethbridge    7923
Name: rental_store_city, dtype: int64
```

df[‘rental_store_city’].value_counts()

- Notice: the two cities are about even




df[‘rental_store_city’].value_counts().plot(kind = ‘pie’, figsize = (6, 6))

- Need *.value_counts()*
  - Pie chart will not be the output without it; will instead get an error
  - I.e. df[‘rental_store_city’].plot(kind = ‘pie’, figsize = (6, 6))
    - This will NOT work
      - b/c NOTE the values within this column are of type *str*

ax = df[‘rental_store_city’].value_counts().plot(kind = ‘bar’, figsize = (14, 6))
ax.set_ylabel(Number of Rentals’)

**Column Wrangling**

- Can also create new columns or modify existing ones
- Want: the rental rate of return of each film
- Will add and calculate a new *rental_rate_return* column

  *rental_gain_return* = (*film_rental_rate* / *film_replacement_cost*) * 100
- Return of rentals
  - I.e. which film rentals will be more profitable for the company?

```
In [23]: df['rental_gain_return'] = df['film_rental_rate'] / df['film_replacement_cost'] * 100
```

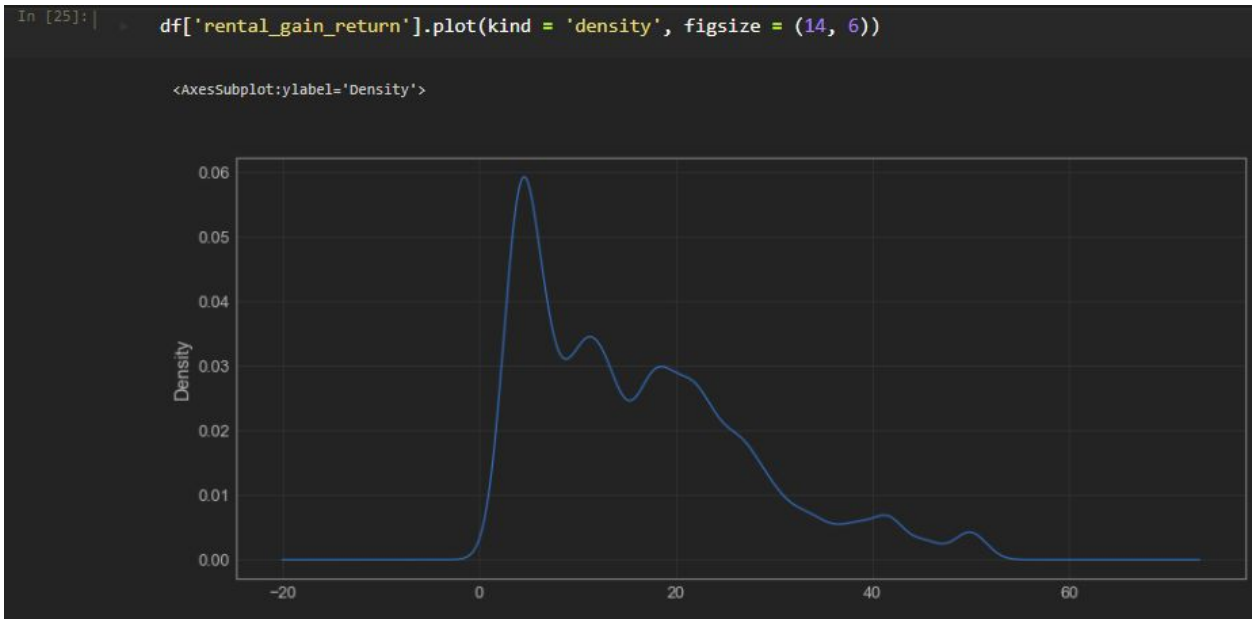
df[‘rental_gain_return’] = df[‘film_rental_rate’] / df[‘film_replacement_cost’] * 100

- Rental rate (how much we charge) / the cost to acquire the film

```
In [24]: df['rental_gain_return'].head()

rental_id
1    13.597090
2    17.598587
3    19.946631
4     4.502046
5     9.969990
Name: rental_gain_return, dtype: float64
```

df[‘rental_gain_return’].head()

```
In [25]: df['rental_gain_return'].plot(kind = 'density', figsize = (14, 6))

<AxesSubplot:ylabel='Density'>
```




df[‘rental_gain_return’].plot(kind = ‘density’, figsize = (14, 6))

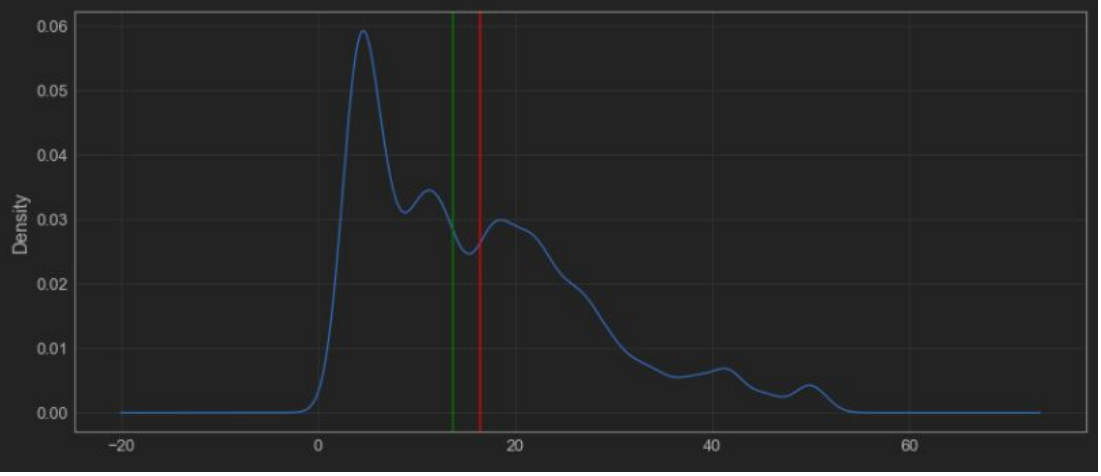
- Rentals nearly zero, then more profitable rentals
  - I.e. making up to 60% of the rental

```
In [26]: df['rental_gain_return'].mean().round(2)

16.34

In [29]: df['rental_gain_return'].median().round(2)
         # Each rental; represents 13.6% of film cost

13.6
```

df[‘rentail_gain_return’].mean().round(2)

def[‘rentail_gain_return’].median().round(2)

```
In [30]: ax = df['rental_gain_return'].plot(kind = 'density', figsize = (14, 6))
         ax.axvline(df['rental_gain_return'].mean(), color = 'red')
         ax.axvline(df['rental_gain_return'].median(), color = 'green')

<matplotlib.lines.Line2D at 0x2a07e850>
```




ax = df[‘rental_gain_return’].plot(kind = ‘density’, figsize = (14, 6))
ax.axvline(df[‘rental_gain_return’].mean(), color = ‘red’)
ax.axvline(df[‘rental_gain_return’].median().color = ‘green’)

```
In [31]: 100 / 13.6
         # So this output (7.35) is the number of rentals needed to recover film market price
         # (film_replacement_cost).
         # recall from above, the median of rental_gain_return is 13.6

7.352941176470589
```

100 / 13.6

- 100 / median of *rental_gain_return*

- 7.35 ~ 7 is the number of rentals needed to recover film market price
    - This is the film replacement cost

```
In [35]:
# While on average, each film is rented 16.74 times.
df['film_title'].value_counts().mean()

16.747390396659707
```

df['film_title'].value_counts().mean()

- Whenever the value within a column is type *str* ALWAYS include *value_counts()*

**Selection and Indexing**

**df.loc[[...] == ...]**

- This will always be used when filtering items (selection & indexing)
- Zooming into a particular characteristic within the data

```
In [36]:
# Selection & Indexing
# Get the rental records of the customer with lastname HANSEN

df.loc[df['customer_lastname'] == 'HANSEN']
```

| rental_id | rental_date | return_date | customer_lastname | store_id | rental_store_city | film_title | film_rental_duration | fil |
|---|---|---|---|---|---|---|---|---|
| 5 | 2005-05-24 23:05:21 | 2005-06-02 04:33:21 | HANSEN | 2 | Woodridge | IDOLS SNATCHERS | 5 | 2.9 |
| 134 | 2005-05-25 21:48:41 | 2005-06-02 18:28:41 | HANSEN | 2 | Woodridge | JUMPING WRATH | 4 | 0.9 |
| 416 | 2005-05-27 15:02:10 | 2005-05-29 10:34:10 | HANSEN | 2 | Woodridge | LESSON CLEOPATRA | 3 | 0.9 |
| 809 | 2005-05-29 19:10:20 | 2005-06-05 19:05:20 | HANSEN | 2 | Woodridge | INDIAN LOVE | 4 | 0.9 |
| 1006 | 2005-05-31 00:57:08 | 2005-06-02 22:35:08 | HANSEN | 2 | Woodridge | SALUTE APOLLO | 4 | 2.9 |

df.loc[df['customer_lastname'] == 'HANSEN']

```
In [37]:
# Create a list of all the films with the highest replacement cost
df['film_replacement_cost'].max()

29.99
```

df['film_replacement_cost'].max()

```
In [60]: ilmList = df.loc[df['film_replacement_cost'] == df['film_replacement_cost'].max(), 'film_title'].unique()

         ilmList

array(['IDOLS SNATCHERS', 'LAWLESS VISION', 'SONG HEDWIG',
       'LOATHING LEGALLY', 'PATIENT SISTER', 'RESERVOIR ADAPTATION',
       'JEEPERS WEDDING', 'GOLDFINGER SENSIBILITY', 'CHARIOTS CONSPIRACY',
       'HONEY TIES', 'GRAFFITI LOVE', 'SLACKER LIAISONS', 'DIRTY ACE',
       'BLINDNESS GUN', 'WYOMING STORM', 'FEUD FROGMEN', 'SALUTE APOLLO',
       'JINGLE SAGEBRUSH', 'HILLS NEIGHBORS', 'UNCUT SUICIDES',
       'EVERYONE CRAFT', 'FLATLINERS KILLER', 'BALLROOM MOCKINGBIRD',
       'RIVER OUTLAW', 'ARABIA DOGMA', 'VIRGIN DAISY', 'JERICHO MULAN',
       'SASSY PACKER', 'TRACY CIDER', 'LOVER TRUMAN', 'DOCTOR GRAIL',
       'GILMORE BOILED', 'PRINCESS GIANT', 'CRUELTY UNFORGIVEN',
       'REIGN GENTLEMEN', 'WEST LION', 'BONNIE HOLOCAUST', 'EARTH VISION',
       'RANDOM GO', 'CLOCKWORK PARADISE', 'FANTASIA PARK', 'RIGHT CRANES',
       'CUPBOARD SINNERS', 'OSCAR GOLD', 'SMILE EARRING',
       'HOLLYWOOD ANONYMOUS', 'POSEIDON FOREVER',
       'EXTRAORDINARY CONQUERER', 'QUEST MUSSOLINI', 'JAPANESE RUN',
       'CLYDE THEORY', 'DESPERATE TRAINSPOTTING'], dtype=object)

In [61]: len(filmList)

52
```

df.loc[df['film_replacement_cost'] == df['film_replacement_cost'].max(), 'film_title'].unique()

- The list of films that have the highest replacement cost

```
In [ ]: # How many PG OR PG-13 rating films were rented?

In [50]: df.loc[(df['film_rating'] == 'PG') | (df['film_rating'] == 'PG-13')]
         # These are the PG or PG-13 rating films that were rented.
```

| rental_id | rental_date | return_date | customer_lastname | store_id | rental_store_city | film_title | film_rental_duration | fil |
|---|---|---|---|---|---|---|---|---|
| 7 | 2005-05-24 23:11:53 | 2005-05-29 20:34:53 | WALTERS | 2 | Woodridge | SWARM GOLD | 4 | 0.9 |
| 9 | 2005-05-25 00:00:40 | 2005-05-28 00:22:40 | SIMPSON | 1 | Lethbridge | MATRIX SNOWMAN | 6 | 4.9 |
| 11 | 2005-05-25 00:09:02 | 2005-06-02 20:56:02 | BURNS | 1 | Lethbridge | WHALE BIKINI | 4 | 4.9 |
| 12 | 2005-05-25 00:19:27 | 2005-05-30 05:44:27 | BYRD | 1 | Lethbridge | GAMES BOWFINGER | 7 | 4.9 |
| 15 | 2005-05-25 00:39:22 | 2005-06-03 03:30:22 | WEINER | 1 | Lethbridge | PELICAN COMFORTS | 4 | 4.9 |

df.loc[(df['film_rating'] == 'PG') | df['film_rating'] == 'PG-13']

```
In [52]: # To count how many...
         len(df.loc[(df['film_rating'] == 'PG') | (df['film_rating'] == 'PG-13')])

6797

In [53]: # OR
         df.loc[(df['film_rating'] == 'PG') | (df['film_rating'] == 'PG-13')].shape[0]

6797
```

df.loc[(df['film_rating'] == 'PG') | (df['film_rating'] == 'PG-13')].shape[0]

**Process:**

- Load the data
- Reshape it somehow
- Create columns
- Cleaning, reshaping, creating new columns, combining data, creating visualizations...