COLLABORATE19

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

SQL Vs Python Pandas

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Pandas

Session ID:

11033

Prepared by:

Rama Koganti

Remember to complete your evaluation for this session within the app!

About Me

Rama Koganti

- 2 years of Python Experience
- 9+ Years Oracle ERP Experience (Functional and Technical)
- Masters in Computer Science.
- APICS:Certified in Production and Inventory Management.

How many of you

- Use Python
- Use Pandas
- Interact with multiple systems/databases/ Excel..

Agenda

❑ About Python & Pandas

❑ SQL like
  - ❑ Select & Where
  - ❑ Joins
  - ❑ Aggregation (Group by, Pivot)

❑ Visualization

❑ Installations

Pandas

Why use Python or Pandas

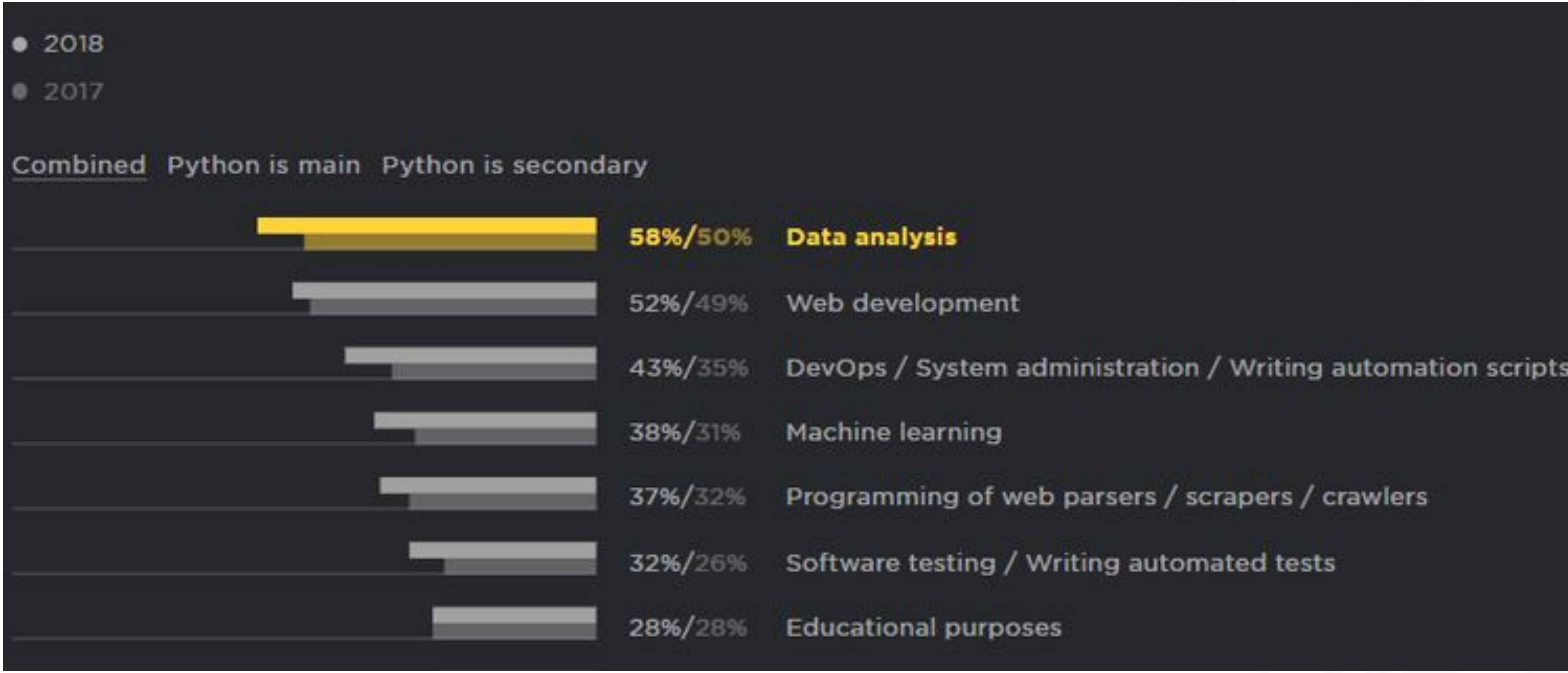
Python

Speed of Code

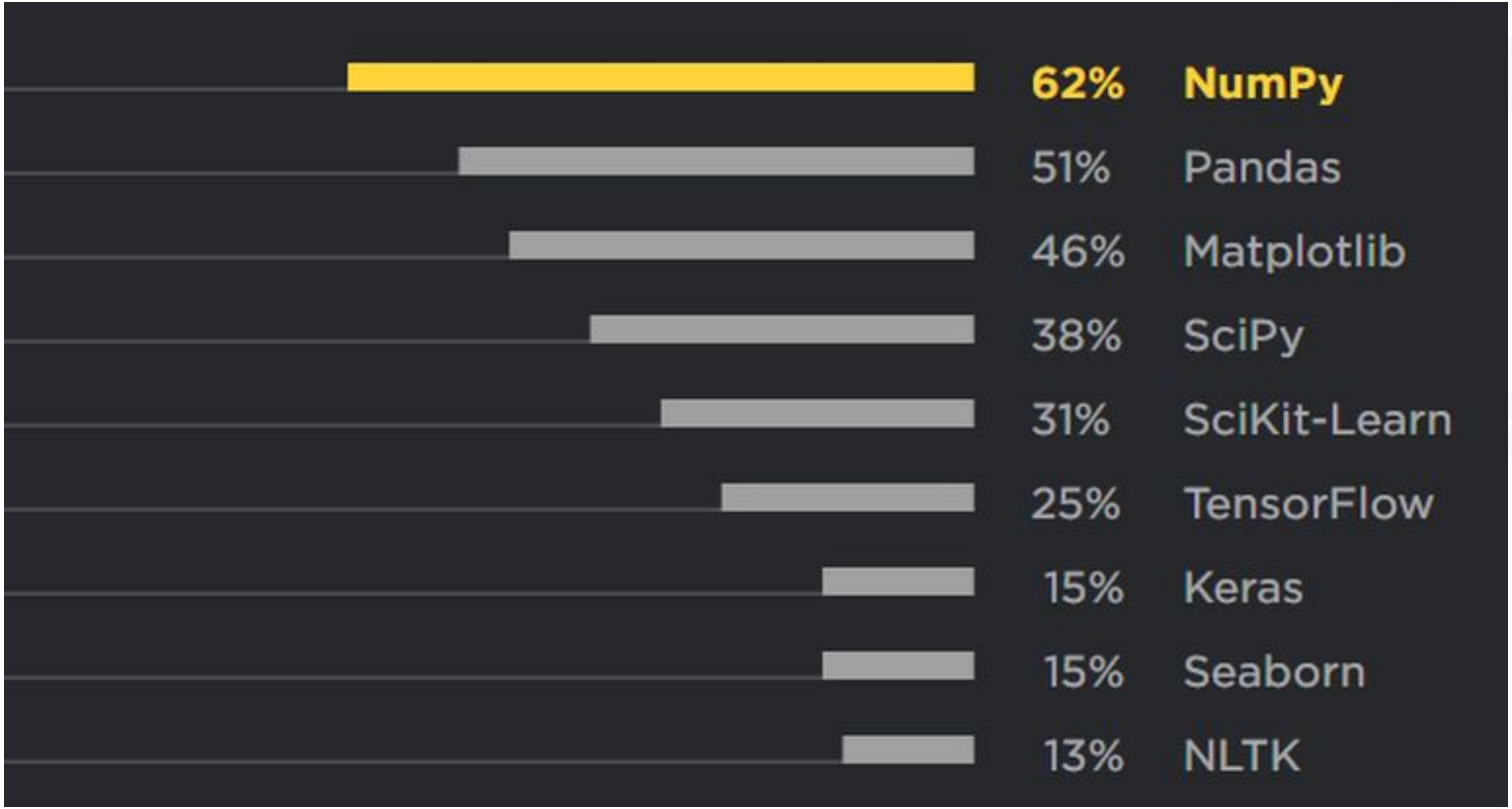
Vs

Speed of Development

What Python is used for?



Most popular Data science frameworks for Python



? Pandas

	Col 1	Col 2	Col 3	Col 4
one	5	4	2	6
two	4	6	4	NA
three	8	6	4	9
four	1	3	8	4
five	4	3	NA	3

- Python In-Memory Table.
- Perform
  - Data Import / Export
  - Data Selection
  - Data Reshaping (Pivot, Group by)
  - Visualizations.

} SQL Like Operations in Python

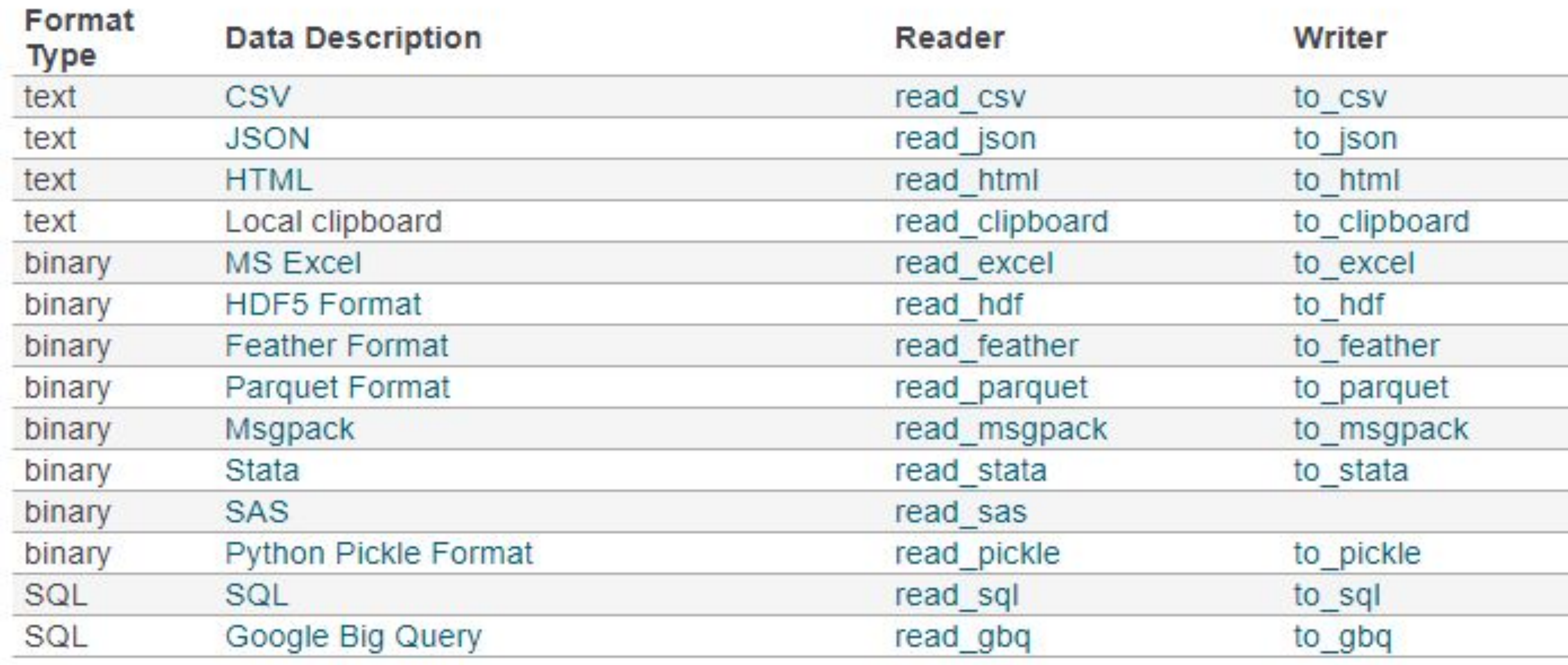
Supported Data Sources

- DB: Oracle, Microsoft SQL, MySQL, Mongodb …
- Excel,CSV
- JSON, HTML
- SAS
- Google Bigquery.
- HDF5 (Hadoop File System)

Pandas Dataframe

	Col 1	Col 2	Col 3	Col 4
one	5	4	2	6
two	4	6	4	NA
three	8	6	4	9
four	1	3	8	4
five	4	3	NA	3

Pandas Data IO : Full list



Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

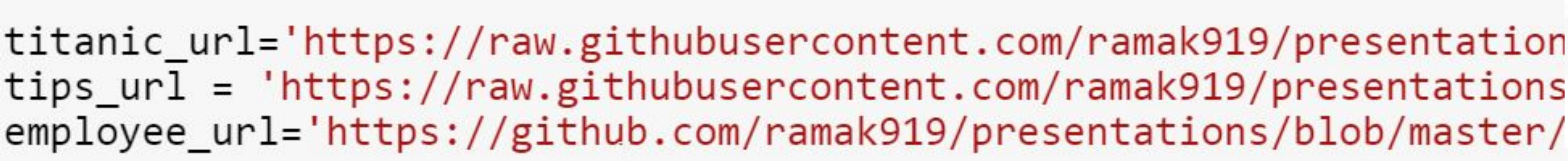
COLLABORATE19

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

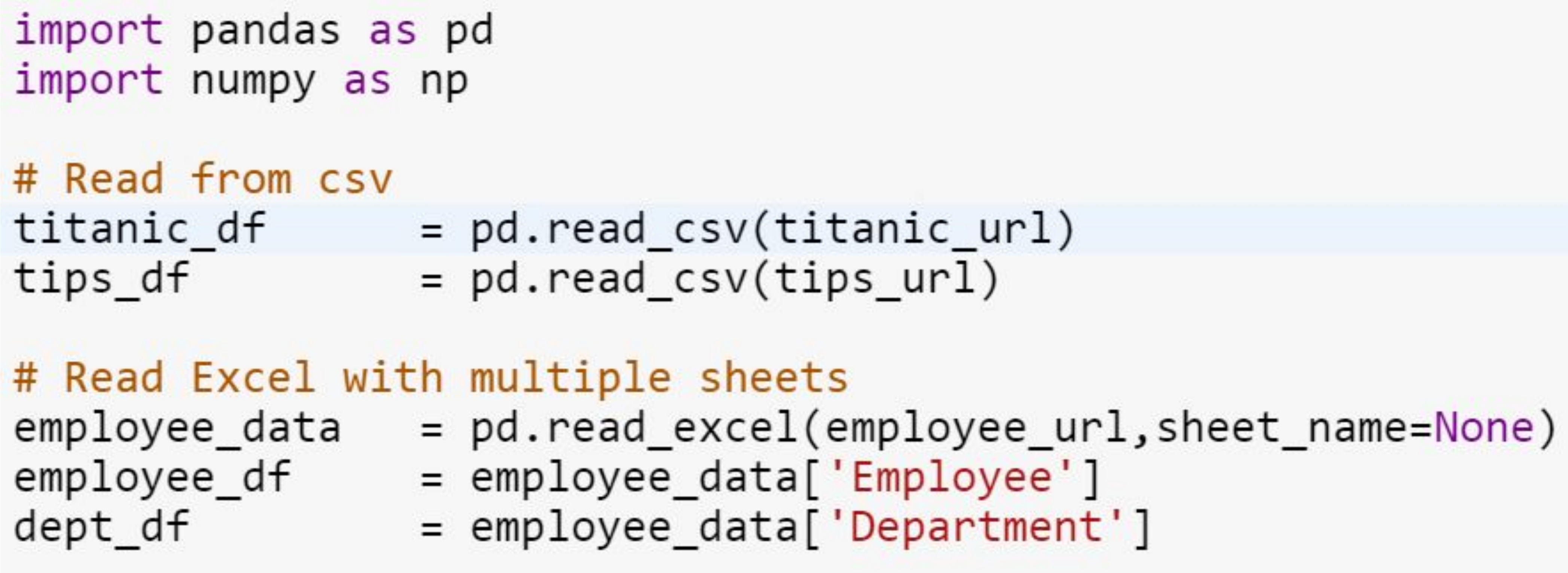
Create / Load DataFrame

#C19TX

Create/Load Data Frame



```
titanic_url='https://raw.githubusercontent.com/ramak919/presentation
tips_url = 'https://raw.githubusercontent.com/ramak919/presentations
employee_url='https://github.com/ramak919/presentations/blob/master/
```

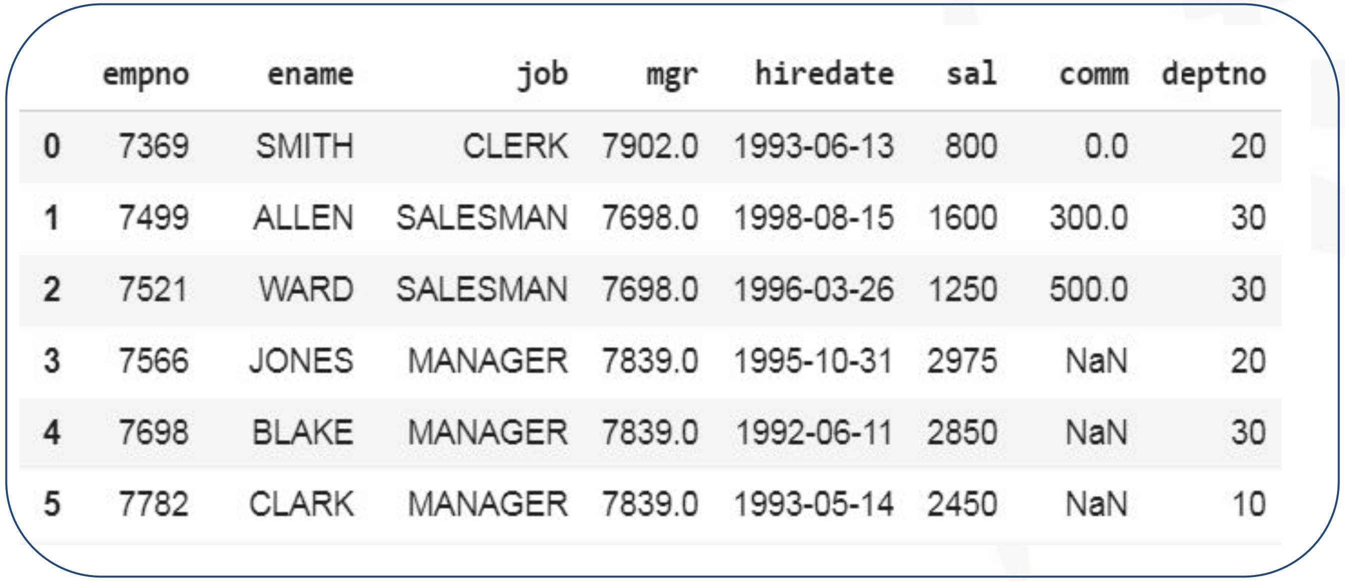


```
import pandas as pd
import numpy as np

# Read from csv
titanic_df        = pd.read_csv(titanic_url)
tips_df           = pd.read_csv(tips_url)

# Read Excel with multiple sheets
employee_data     = pd.read_excel(employee_url,sheet_name=None)
employee_df       = employee_data['Employee']
dept_df           = employee_data['Department']
```

Data used is available publicly through various Data Analysis & Machine Learning Teachings

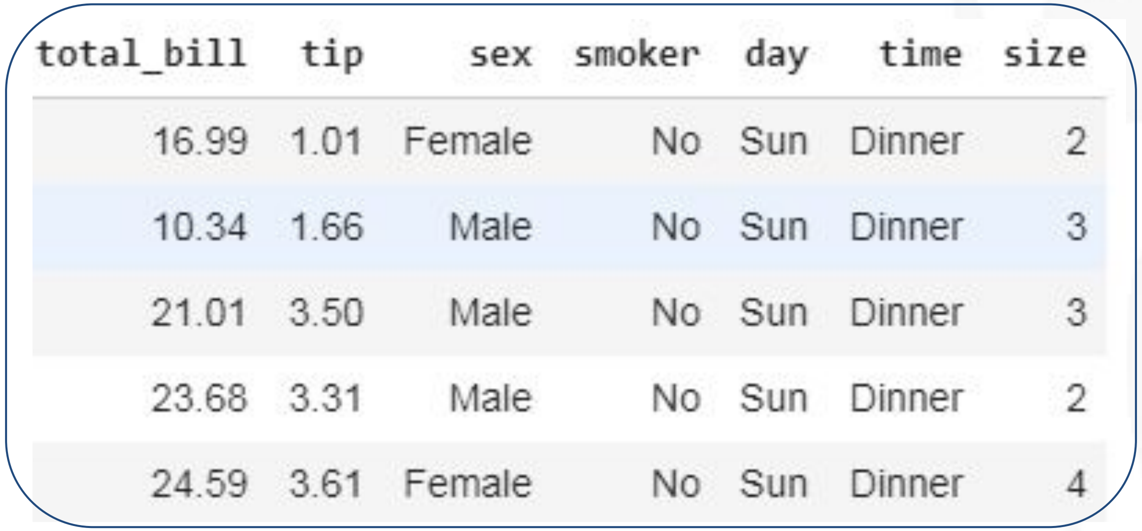
Dataframe

Employee Data Frame



	empno	ename	job	mgr	hiredate	sal	comm	deptno
0	7369	SMITH	CLERK	7902.0	1993-06-13	800	0.0	20
1	7499	ALLEN	SALESMAN	7698.0	1998-08-15	1600	300.0	30
2	7521	WARD	SALESMAN	7698.0	1996-03-26	1250	500.0	30
3	7566	JONES	MANAGER	7839.0	1995-10-31	2975	NaN	20
4	7698	BLAKE	MANAGER	7839.0	1992-06-11	2850	NaN	30
5	7782	CLARK	MANAGER	7839.0	1993-05-14	2450	NaN	10

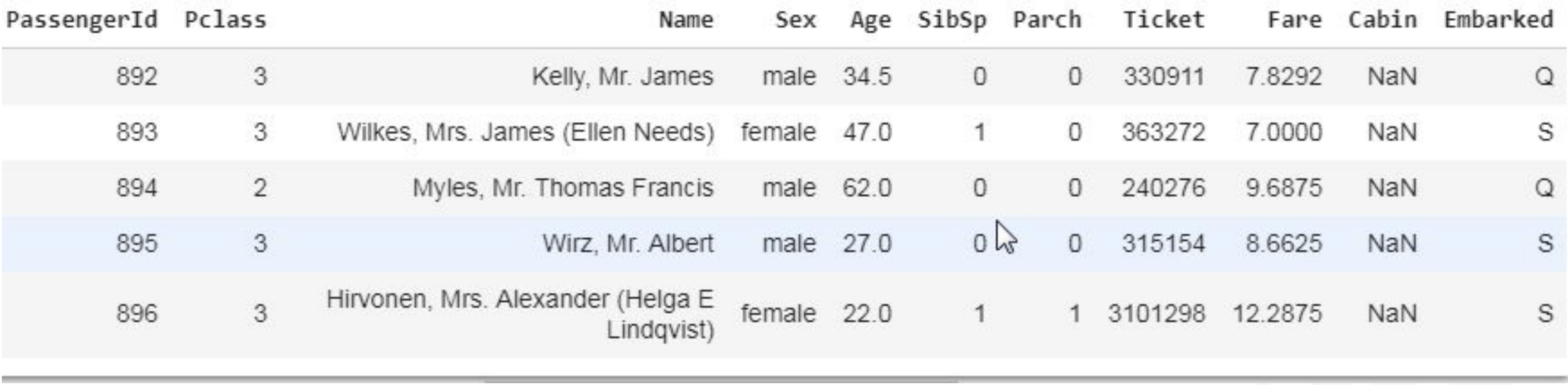
Tips Data Frame



total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.50	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4

Dataframe

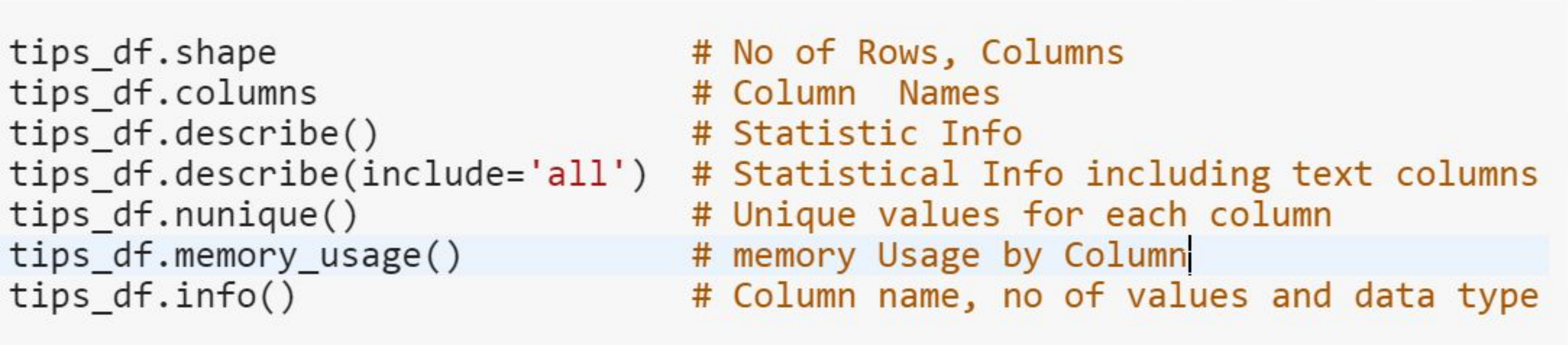
Titanic Data Frame



PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

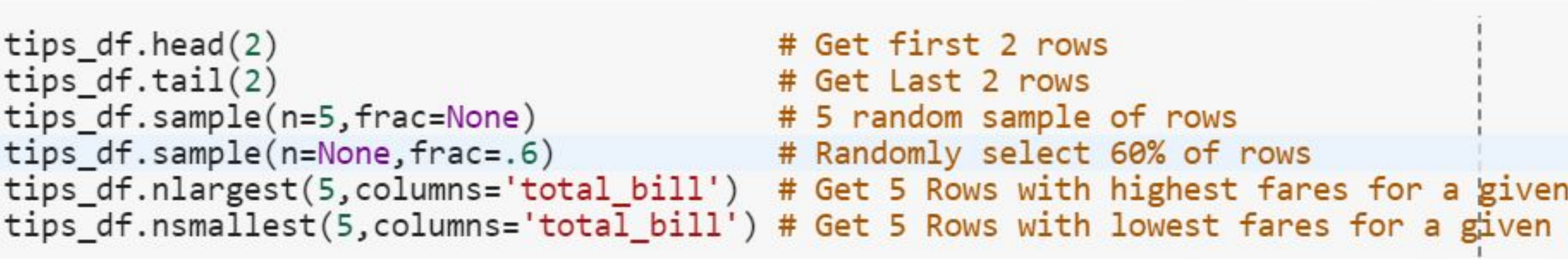
SQL Select & Where

Descriptive Info

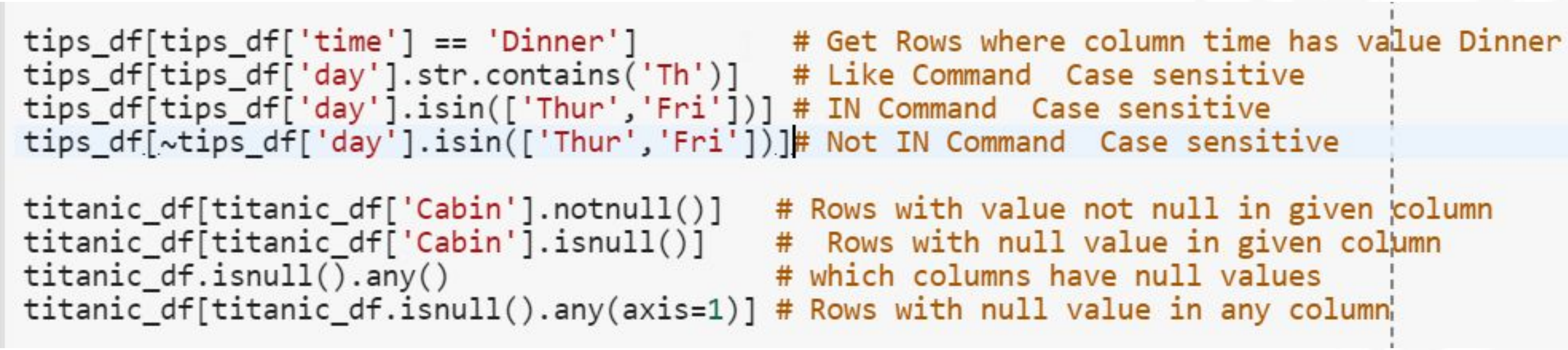


```
tips_df.shape                     # No of Rows, Columns
tips_df.columns                   # Column  Names
tips_df.describe()                # Statistic Info
tips_df.describe(include='all')   # Statistical Info including text columns
tips_df.nunique()                 # Unique values for each column
tips_df.memory_usage()            # memory Usage by Column
tips_df.info()                    # Column name, no of values and data type
```

Where Condition

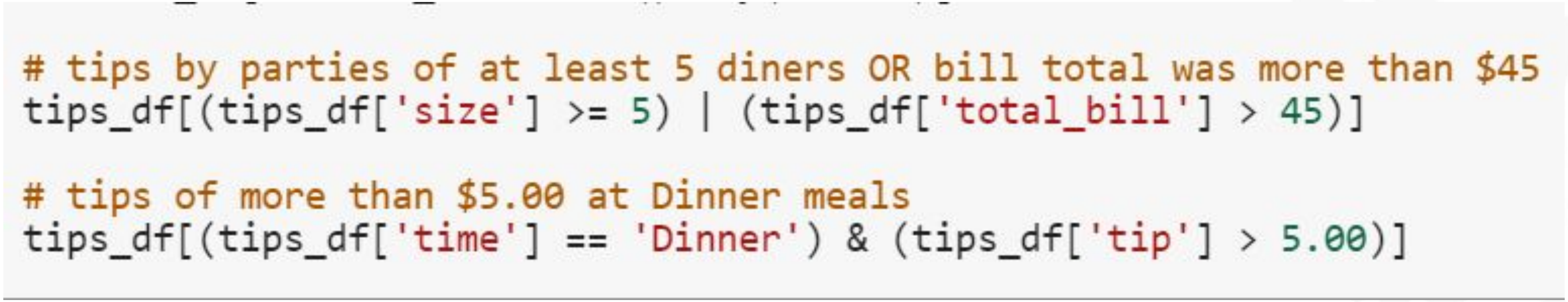


```
tips_df.head(2)                           # Get first 2 rows
tips_df.tail(2)                           # Get Last 2 rows
tips_df.sample(n=5,frac=None)             # 5 random sample of rows
tips_df.sample(n=None,frac=.6)            # Randomly select 60% of rows
tips_df.nlargest(5,columns='total_bill')  # Get 5 Rows with highest fares for a given
tips_df.nsmallest(5,columns='total_bill') # Get 5 Rows with lowest fares for a given
```



```
tips_df[tips_df['time'] == 'Dinner']         # Get Rows where column time has value Dinner
tips_df[tips_df['day'].str.contains('Th')]   # Like Command  Case sensitive
tips_df[tips_df['day'].isin(['Thur','Fri'])] # IN Command  Case sensitive
tips_df[~tips_df['day'].isin(['Thur','Fri'])]# Not IN Command  Case sensitive

titanic_df[titanic_df['Cabin'].notnull()]   # Rows with value not null in given column
titanic_df[titanic_df['Cabin'].isnull()]    #  Rows with null value in given column
titanic_df.isnull().any()                   # which columns have null values
titanic_df[titanic_df.isnull().any(axis=1)] # Rows with null value in any column
```

Where Condition Page 2



```
# tips by parties of at least 5 diners OR bill total was more than $45
tips_df[(tips_df['size'] >= 5) | (tips_df['total_bill'] > 45)]

# tips of more than $5.00 at Dinner meals
tips_df[(tips_df['time'] == 'Dinner') & (tips_df['tip'] > 5.00)]
```

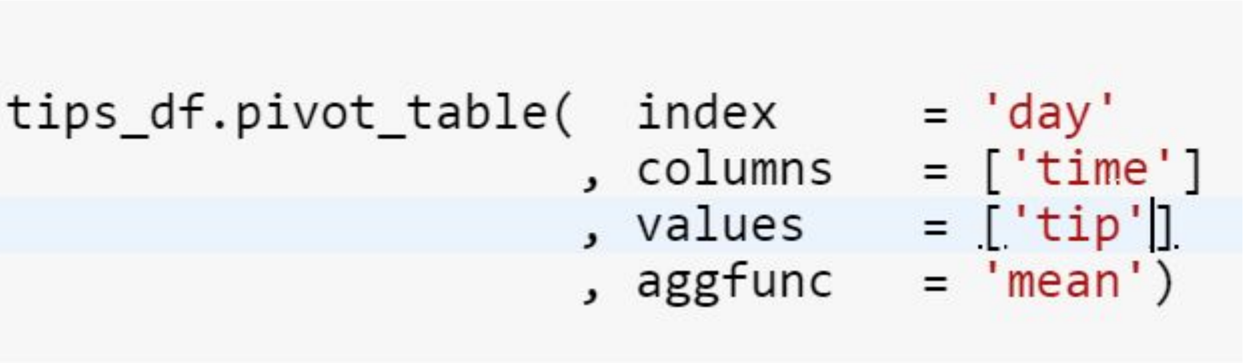
SQL Aggregation

Aggregation Agenda

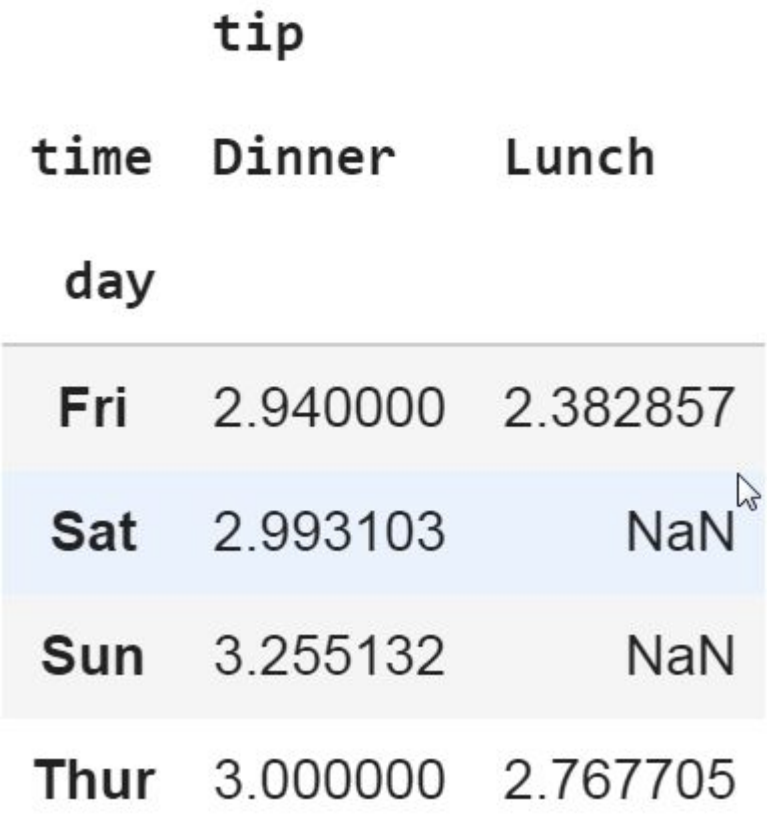
Pivot

Group by

Basic Pivot

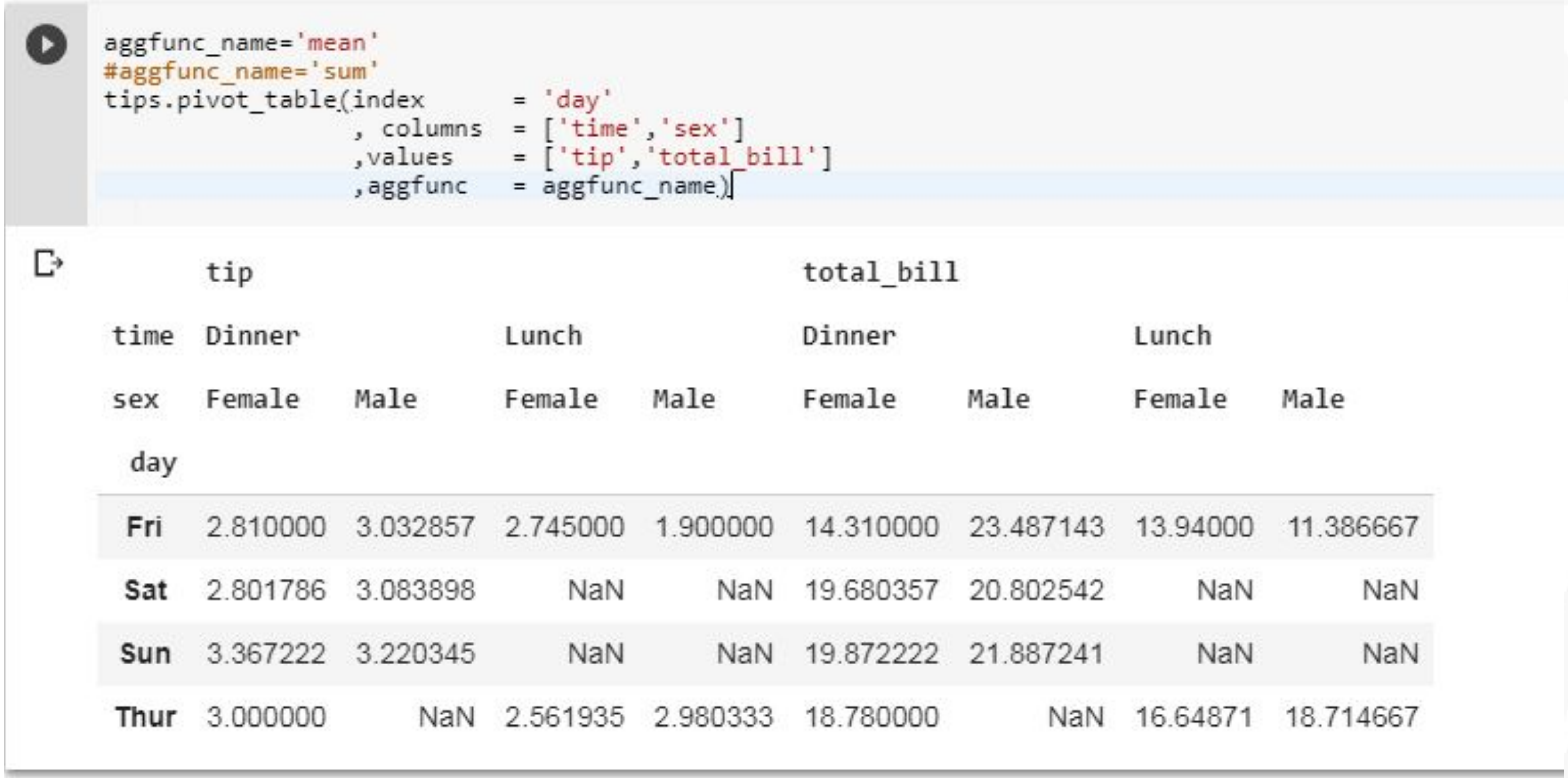


```
tips_df.pivot_table(  index      = 'day'
                    , columns    = ['time']
                    , values     = ['tip']
                    , aggfunc    = 'mean')
```



	tip	
time	Dinner	Lunch
day		
Fri	2.940000	2.382857
Sat	2.993103	NaN
Sun	3.255132	NaN
Thur	3.000000	2.767705

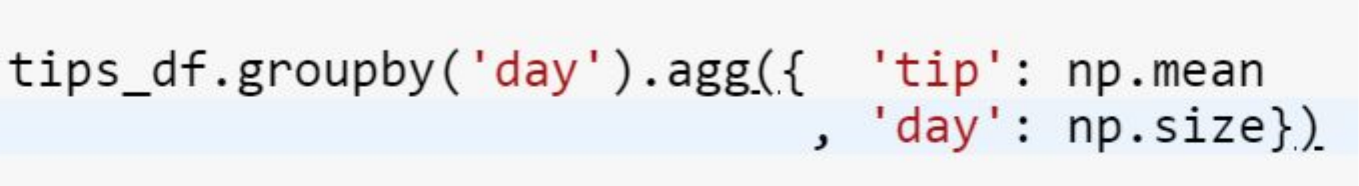
Advanced Pivot



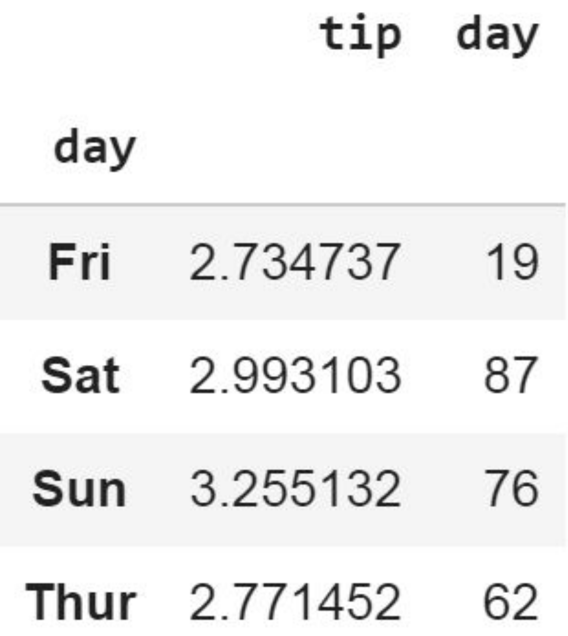
```
aggfunc_name='mean'
#aggfunc_name='sum'
tips.pivot_table(index       = 'day'
                , columns    = ['time','sex']
                ,values      = ['tip','total_bill']
                ,aggfunc     = aggfunc_name)
```

	tip				total_bill			
time	Dinner		Lunch		Dinner		Lunch	
sex	Female	Male	Female	Male	Female	Male	Female	Male
day								
Fri	2.810000	3.032857	2.745000	1.900000	14.310000	23.487143	13.94000	11.386667
Sat	2.801786	3.083898	NaN	NaN	19.680357	20.802542	NaN	NaN
Sun	3.367222	3.220345	NaN	NaN	19.872222	21.887241	NaN	NaN
Thur	3.000000	NaN	2.561935	2.980333	18.780000	NaN	16.64871	18.714667

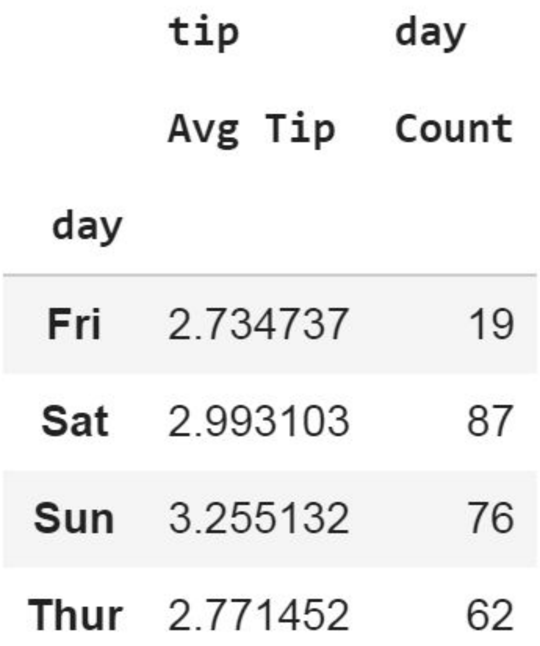
Group By



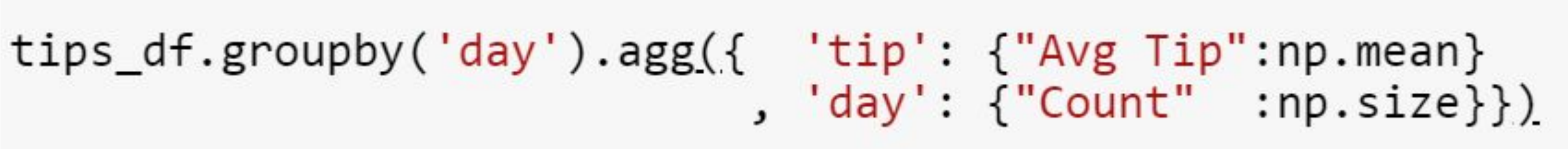
```
tips_df.groupby('day').agg({  'tip': np.mean
                            , 'day': np.size})
```



	tip	day
day		
Fri	2.734737	19
Sat	2.993103	87
Sun	3.255132	76
Thur	2.771452	62

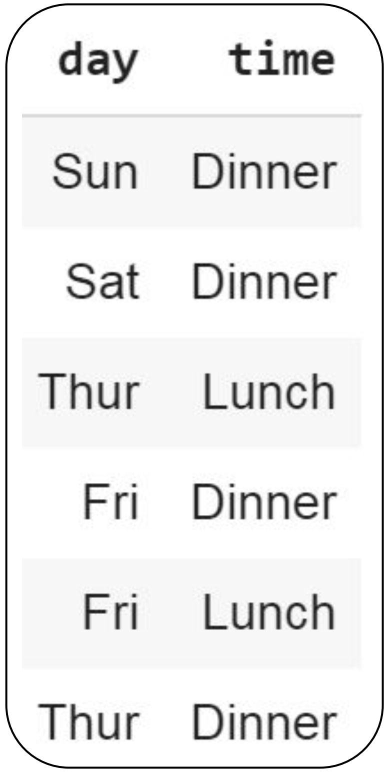


	tip	day
	Avg Tip	Count
day		
Fri	2.734737	19
Sat	2.993103	87
Sun	3.255132	76
Thur	2.771452	62

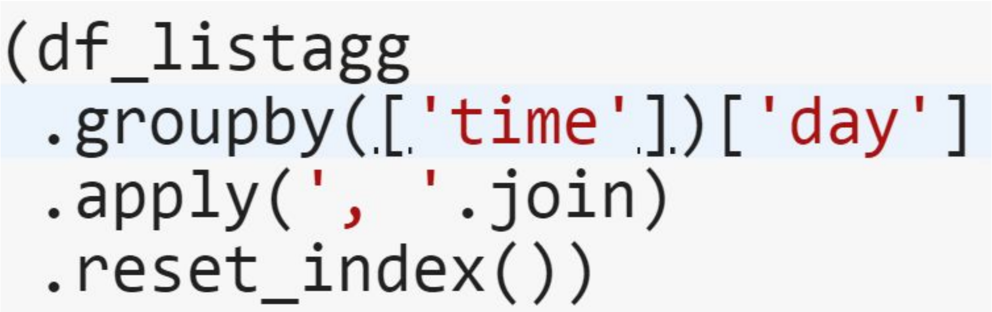


```
tips_df.groupby('day').agg({  'tip': {"Avg Tip":np.mean}
                            , 'day': {"Count"  :np.size}})
```

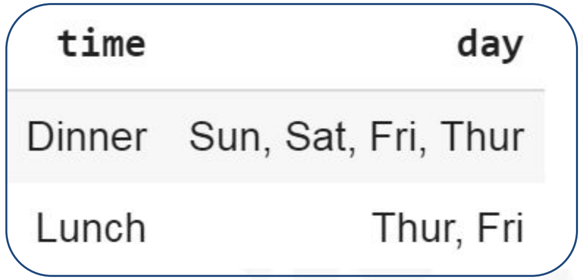
Group By: List Aggregation



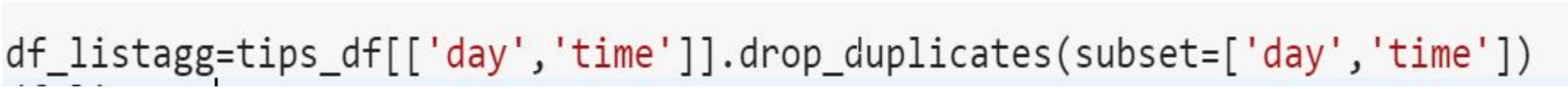
day	time
Sun	Dinner
Sat	Dinner
Thur	Lunch
Fri	Dinner
Fri	Lunch
Thur	Dinner



```
(df_listagg
 .groupby(['time'])['day']
 .apply(', '.join)
 .reset_index())
```



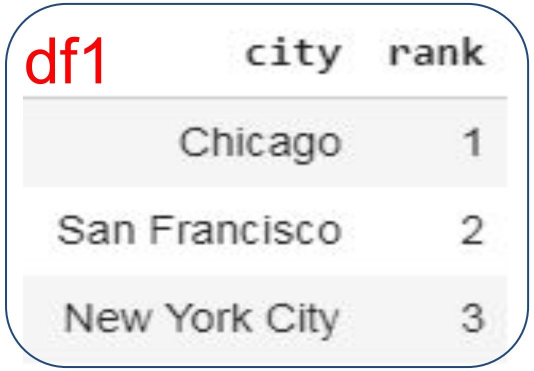
time	day
Dinner	Sun, Sat, Fri, Thur
Lunch	Thur, Fri



```
df_listagg=tips_df[['day','time']].drop_duplicates(subset=['day','time'])
```

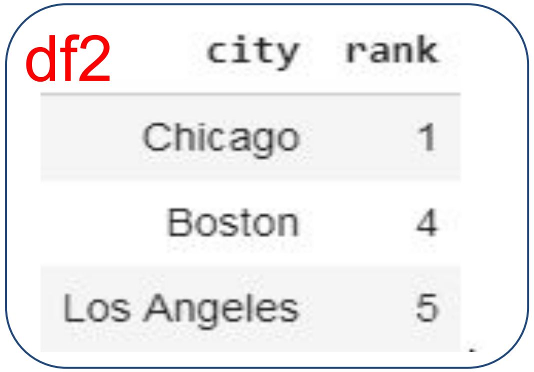
UNION

Union



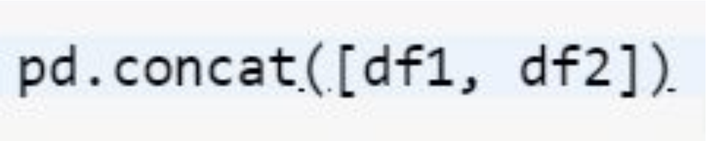
df1

city	rank
Chicago	1
San Francisco	2
New York City	3

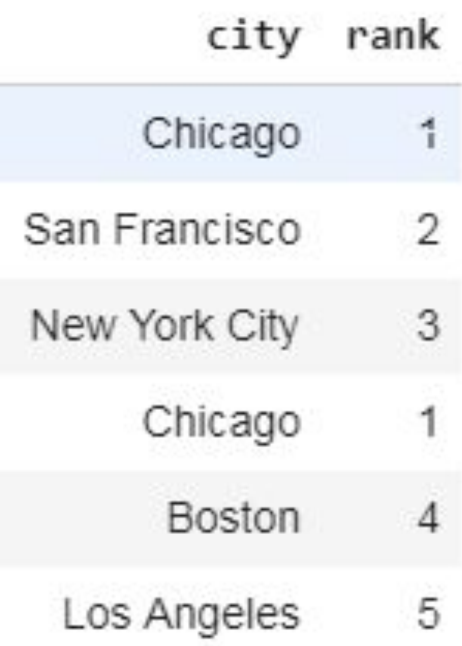


df2

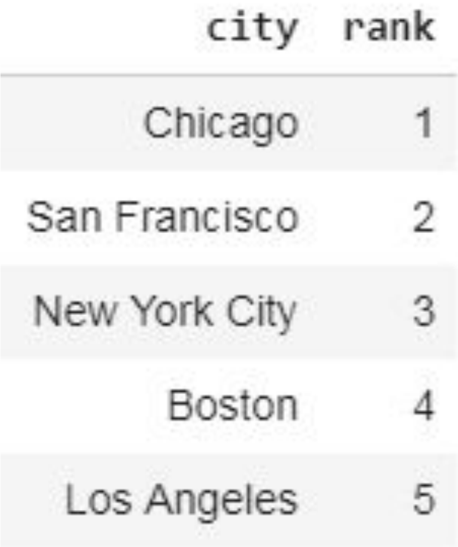
city	rank
Chicago	1
Boston	4
Los Angeles	5



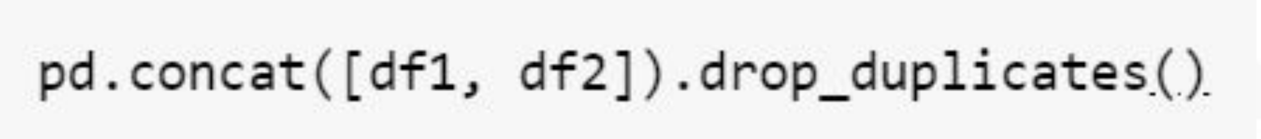
```
pd.concat([df1, df2])
```



city	rank
Chicago	1
San Francisco	2
New York City	3
Chicago	1
Boston	4
Los Angeles	5



city	rank
Chicago	1
San Francisco	2
New York City	3
Boston	4
Los Angeles	5

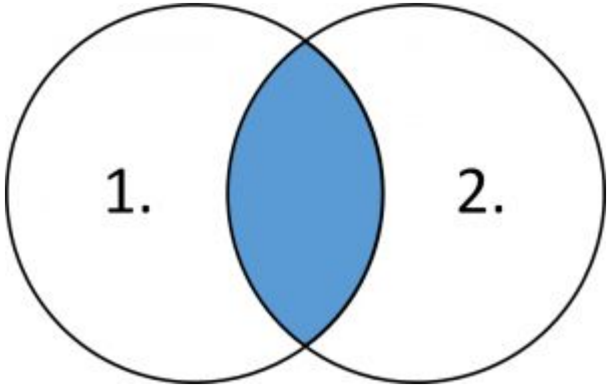


```
pd.concat([df1, df2]).drop_duplicates()
```

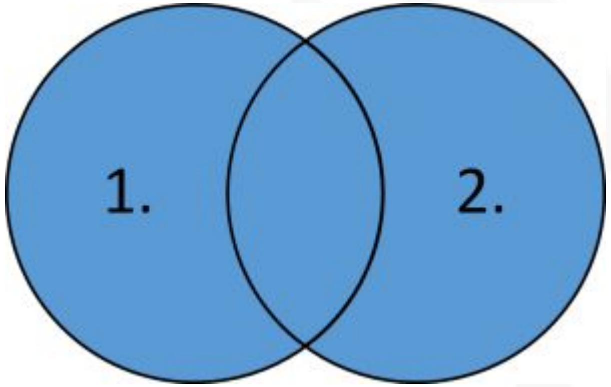
Joins

Join Types

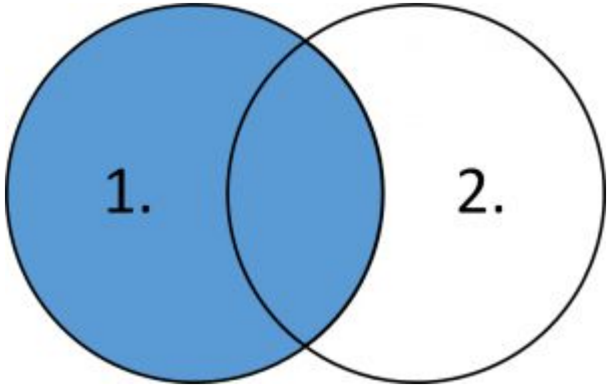
Inner



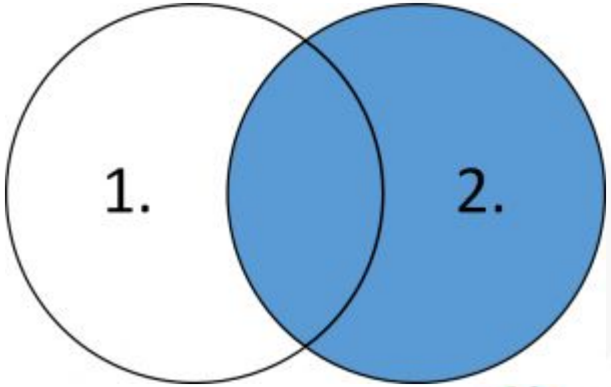
Full Outer



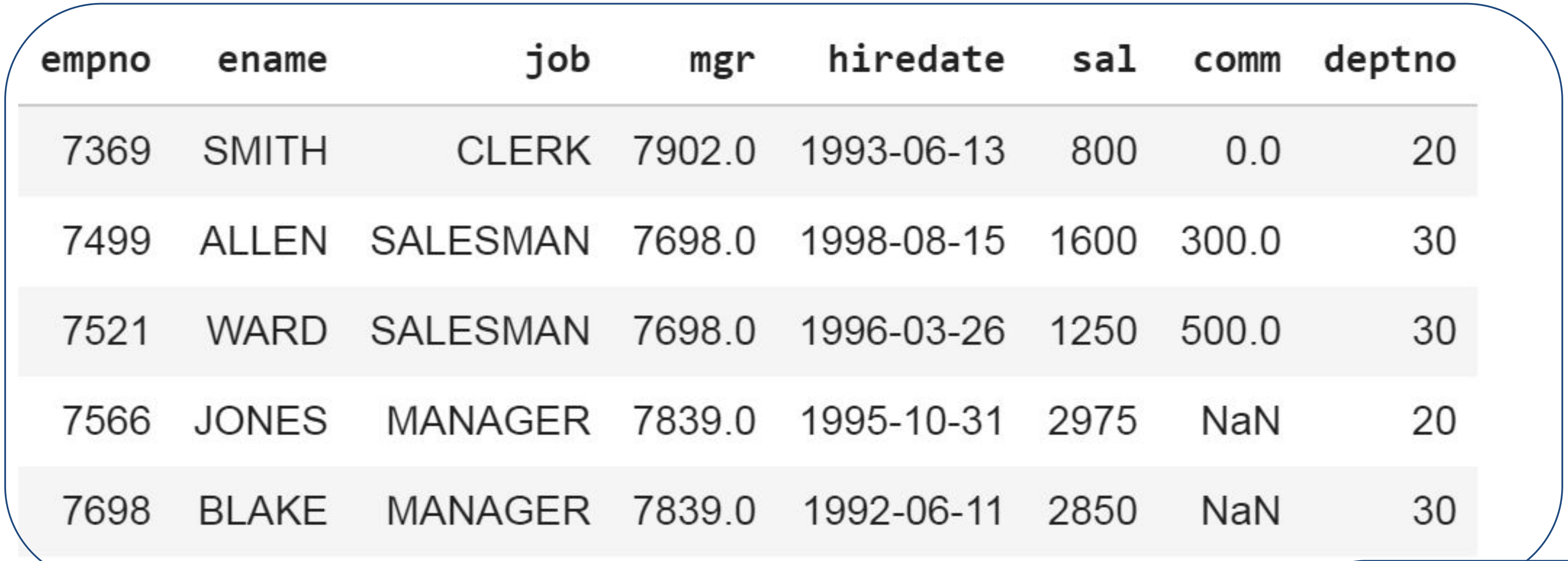
Left



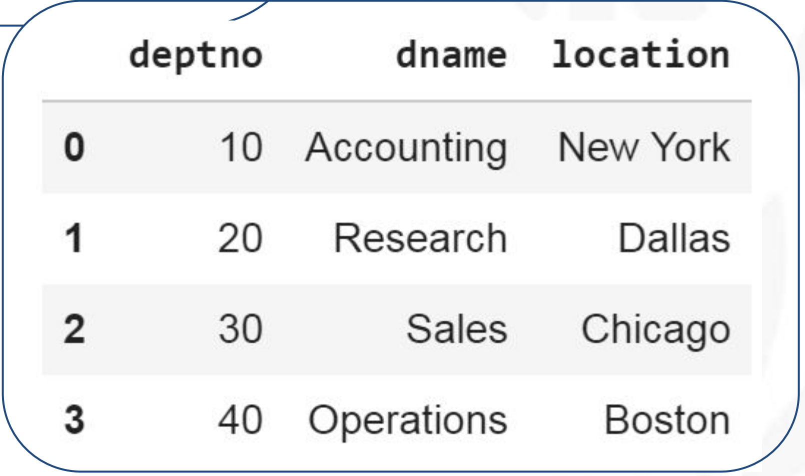
Right



Join Data

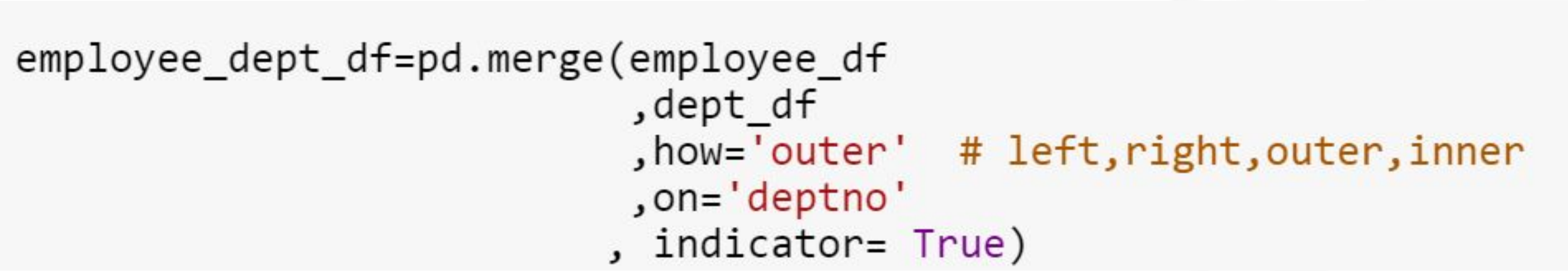


empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902.0	1993-06-13	800	0.0	20
7499	ALLEN	SALESMAN	7698.0	1998-08-15	1600	300.0	30
7521	WARD	SALESMAN	7698.0	1996-03-26	1250	500.0	30
7566	JONES	MANAGER	7839.0	1995-10-31	2975	NaN	20
7698	BLAKE	MANAGER	7839.0	1992-06-11	2850	NaN	30

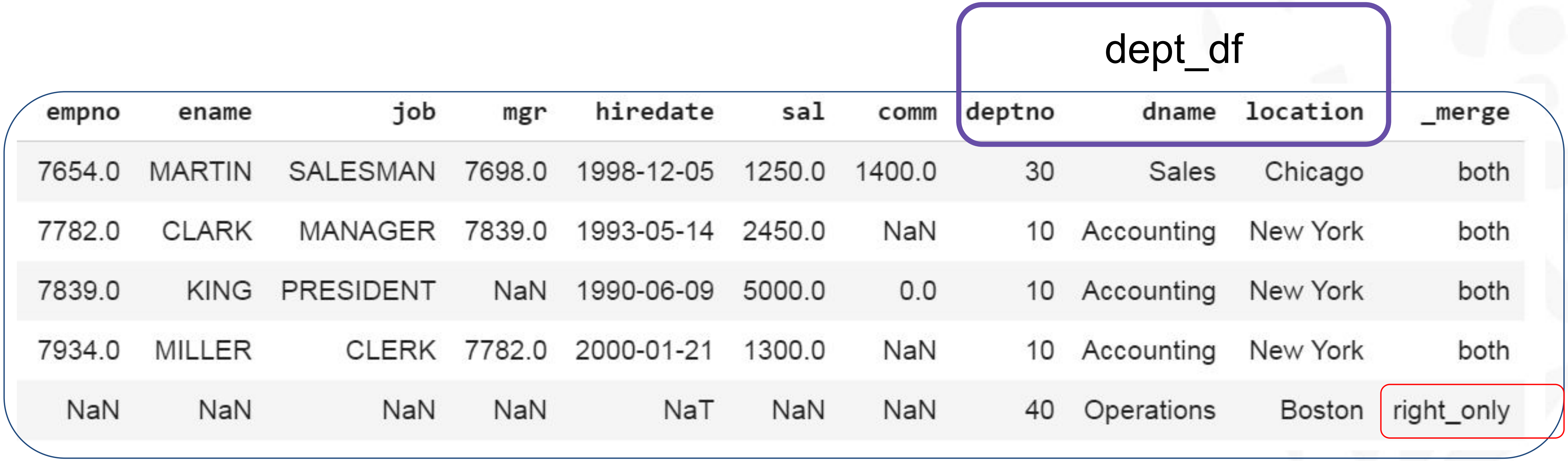


	deptno	dname	location
0	10	Accounting	New York
1	20	Research	Dallas
2	30	Sales	Chicago
3	40	Operations	Boston

Join



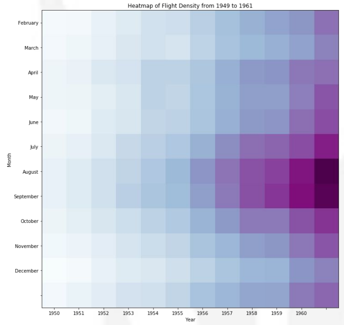
```
employee_dept_df=pd.merge(employee_df
                          ,dept_df
                          ,how='outer'  # left,right,outer,inner
                          ,on='deptno'
                         , indicator= True)
```



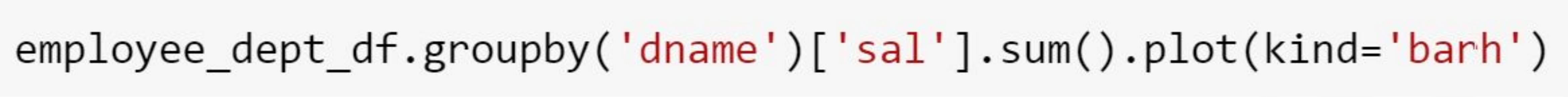
dept_df

empno	ename	job	mgr	hiredate	sal	comm	deptno	dname	location	_merge
7654.0	MARTIN	SALESMAN	7698.0	1998-12-05	1250.0	1400.0	30	Sales	Chicago	both
7782.0	CLARK	MANAGER	7839.0	1993-05-14	2450.0	NaN	10	Accounting	New York	both
7839.0	KING	PRESIDENT	NaN	1990-06-09	5000.0	0.0	10	Accounting	New York	both
7934.0	MILLER	CLERK	7782.0	2000-01-21	1300.0	NaN	10	Accounting	New York	both
NaN	NaN	NaN	NaN	NaT	NaN	NaN	40	Operations	Boston	right_only

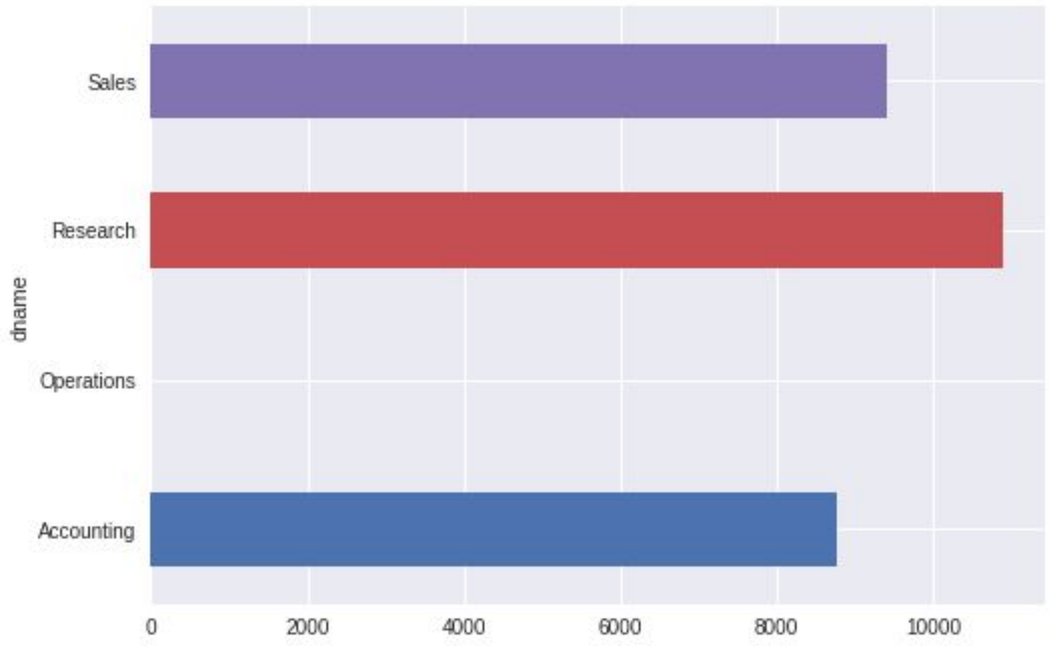
Visualization



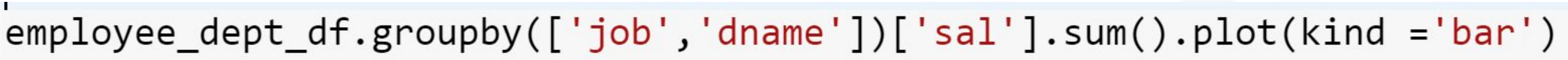
Bar Horizantal



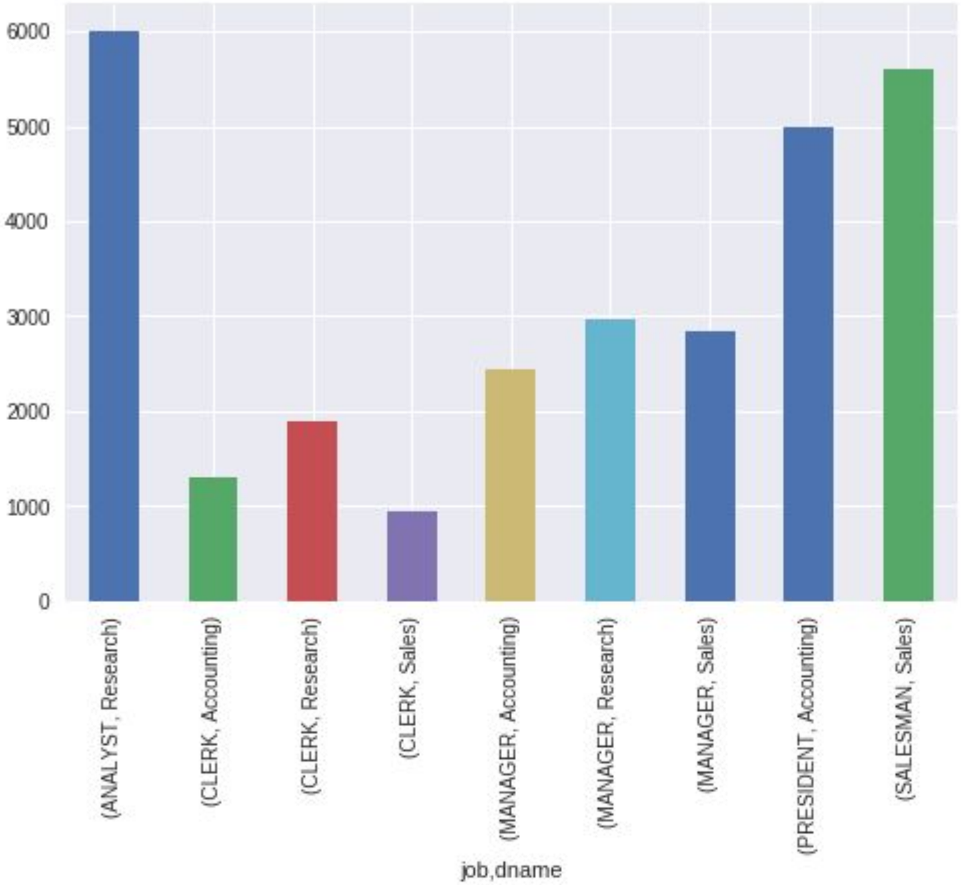
```
employee_dept_df.groupby('dname')['sal'].sum().plot(kind='barh')
```



Bar Vertical

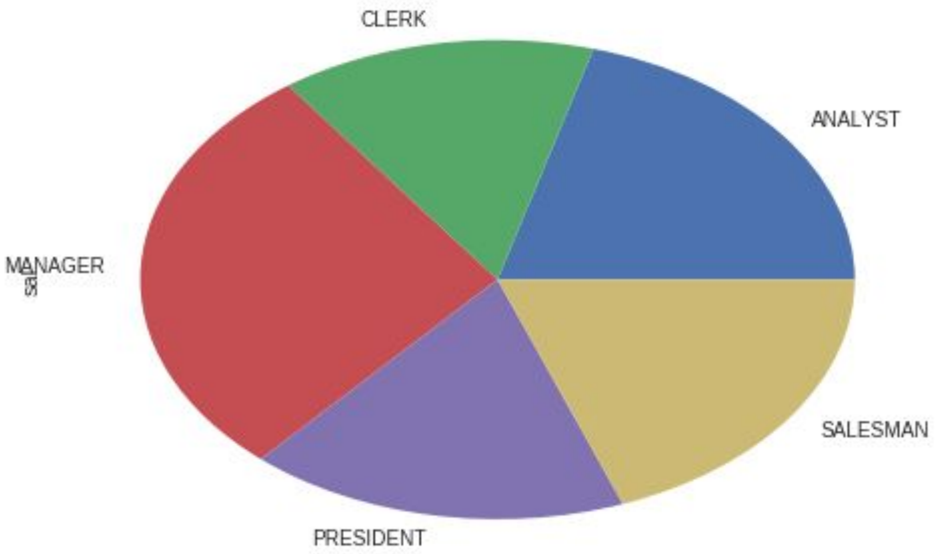


```
employee_dept_df.groupby(['job','dname'])['sal'].sum().plot(kind ='bar')
```

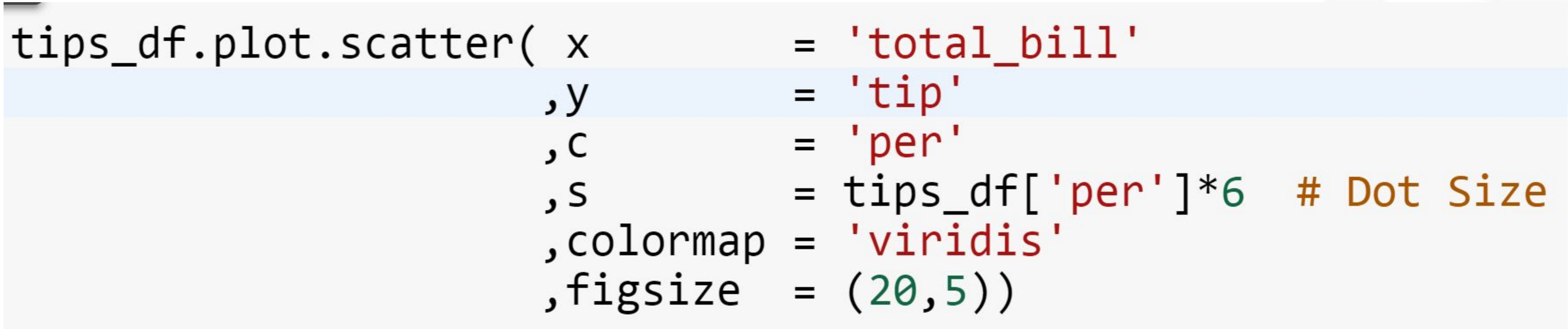


Pie

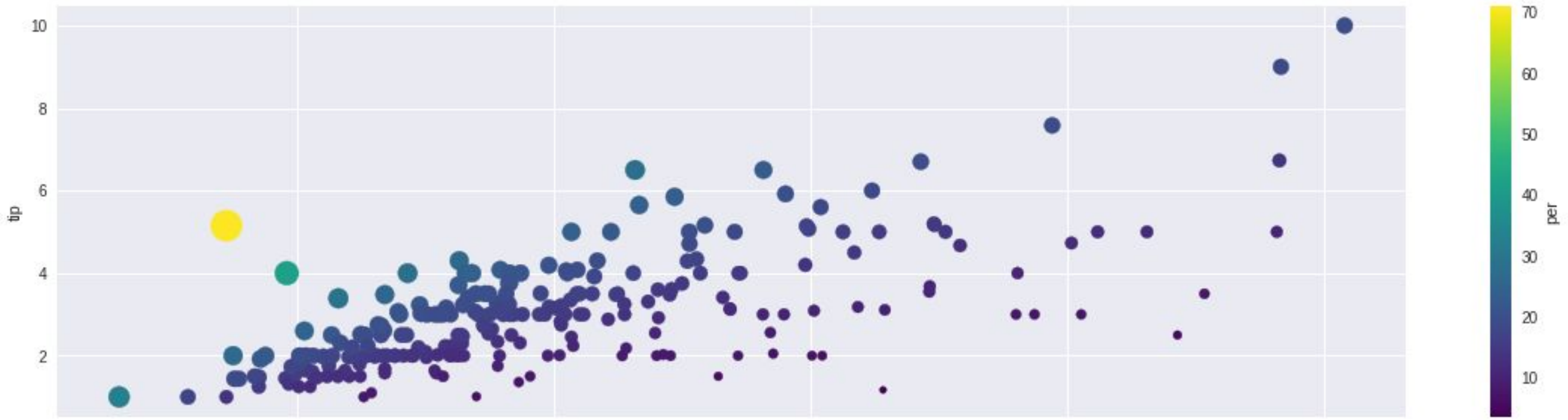
```
employee_dept_df.groupby('job')['sal'].sum().plot(x='job',y='sal',kind='pie')
```



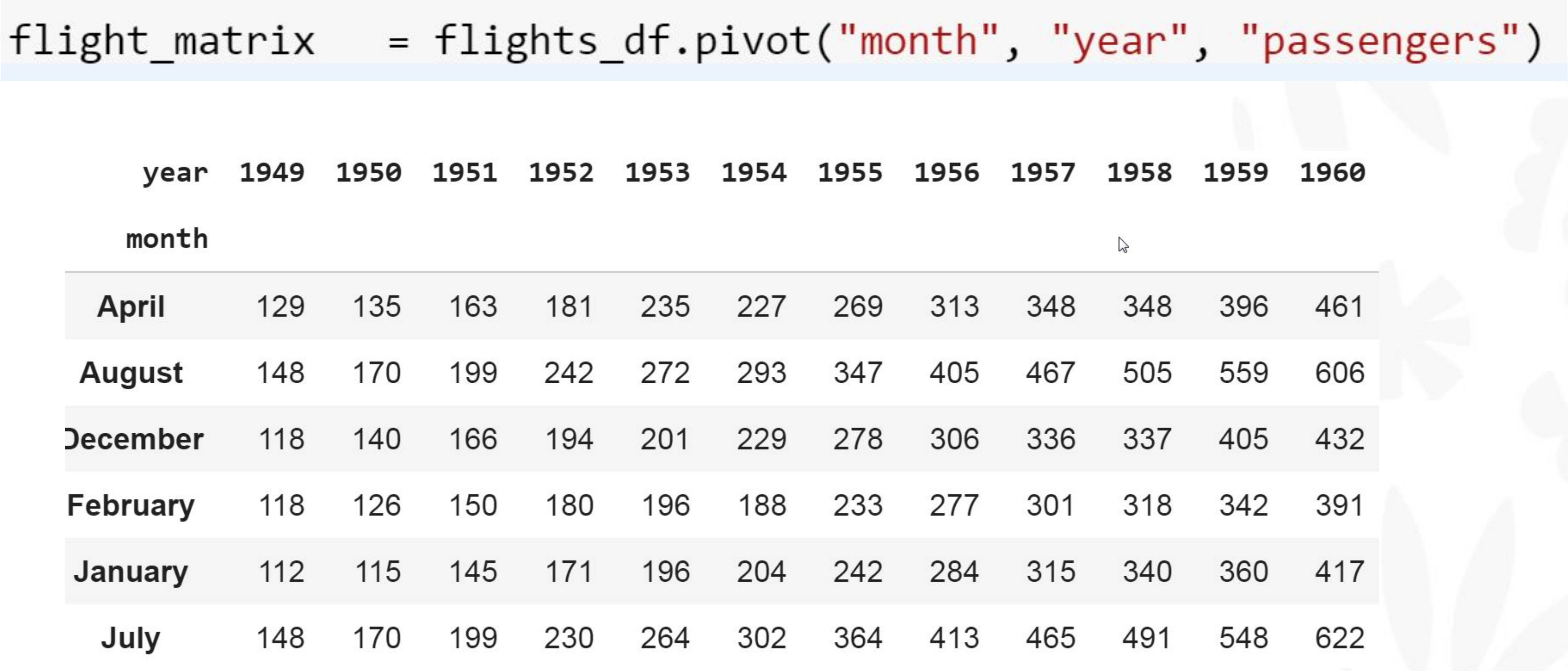
Scatter



```
tips_df.plot.scatter( x        = 'total_bill'
                     ,y        = 'tip'
                     ,c        = 'per'
                     ,s        = tips_df['per']*6  # Dot Size
                     ,colormap = 'viridis'
                     ,figsize  = (20,5))
```



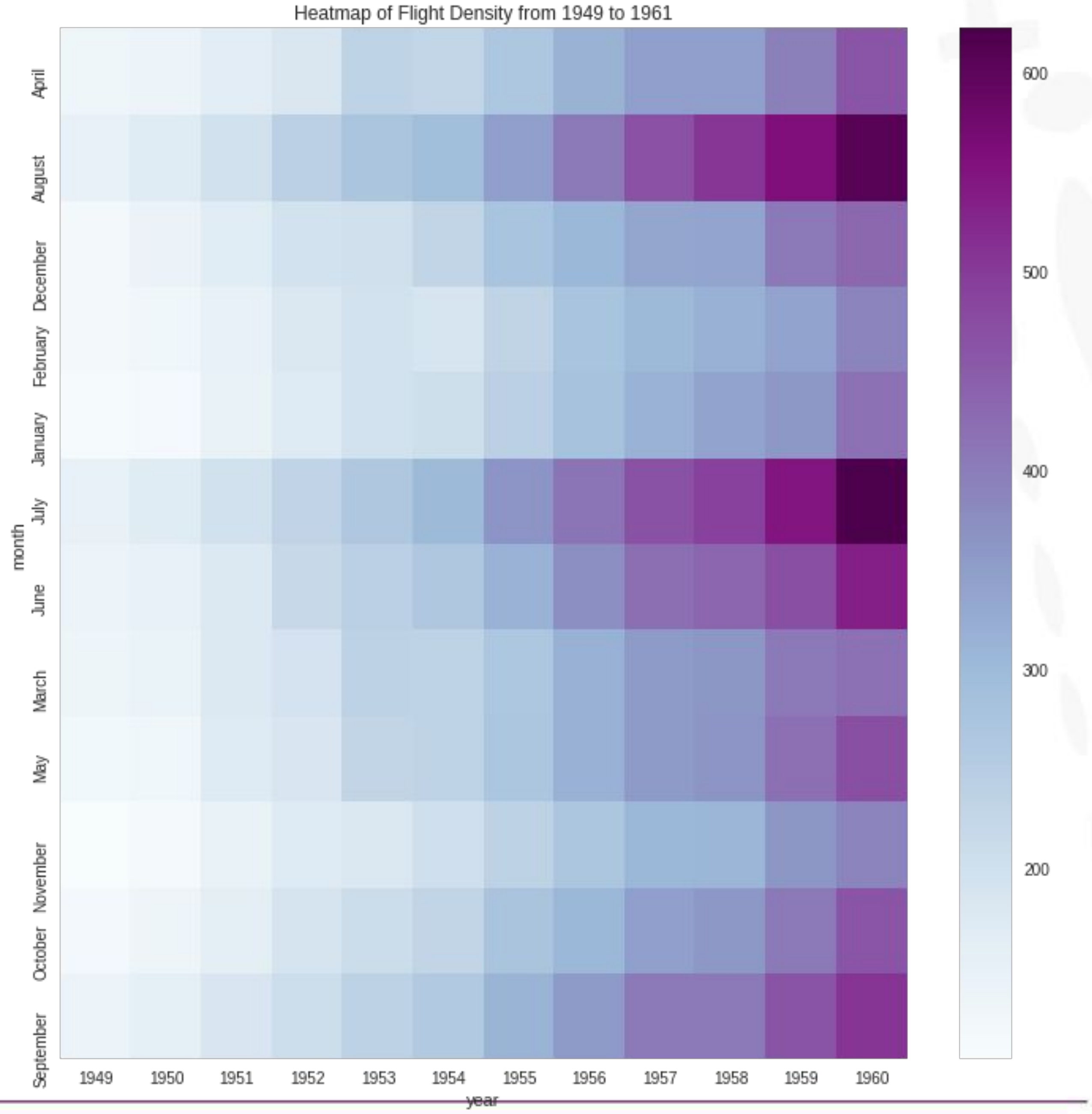
Heat Map



```
flight_matrix   = flights_df.pivot("month", "year", "passengers")
```

year month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
April	129	135	163	181	235	227	269	313	348	348	396	461
August	148	170	199	242	272	293	347	405	467	505	559	606
December	118	140	166	194	201	229	278	306	336	337	405	432
February	118	126	150	180	196	188	233	277	301	318	342	391
January	112	115	145	171	196	204	242	284	315	340	360	417
July	148	170	199	230	264	302	364	413	465	491	548	622

Heat Map


Heatmap of Flight Density from 1949 to 1961
month
April
August
December
February
January
July
June
March
May
November
October
September
year
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
600
500
400
300
200

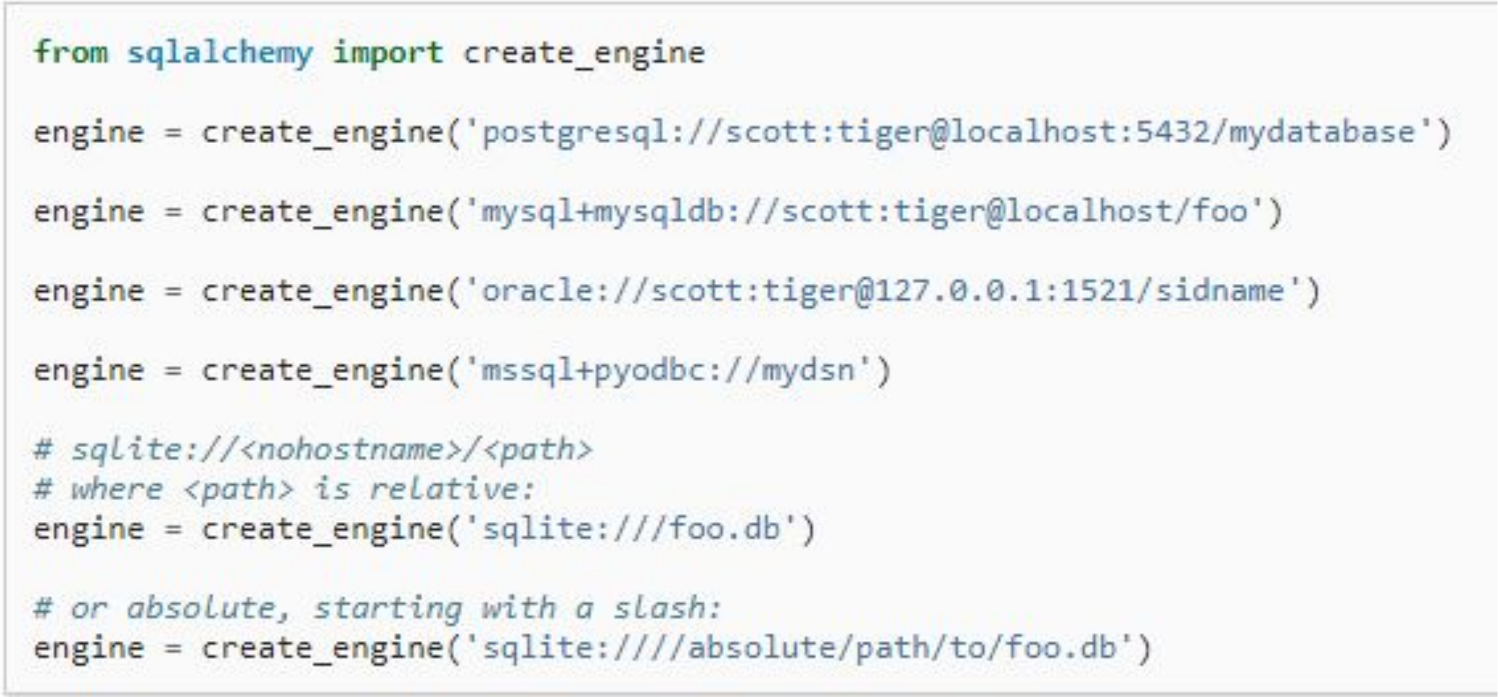
COLLABORATE19

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

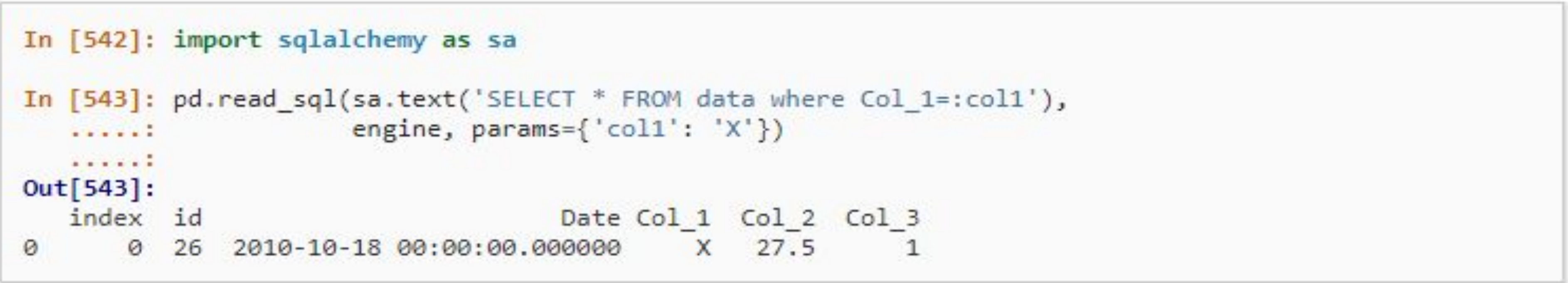
DB Connections

#C19TX

Connect to db examples

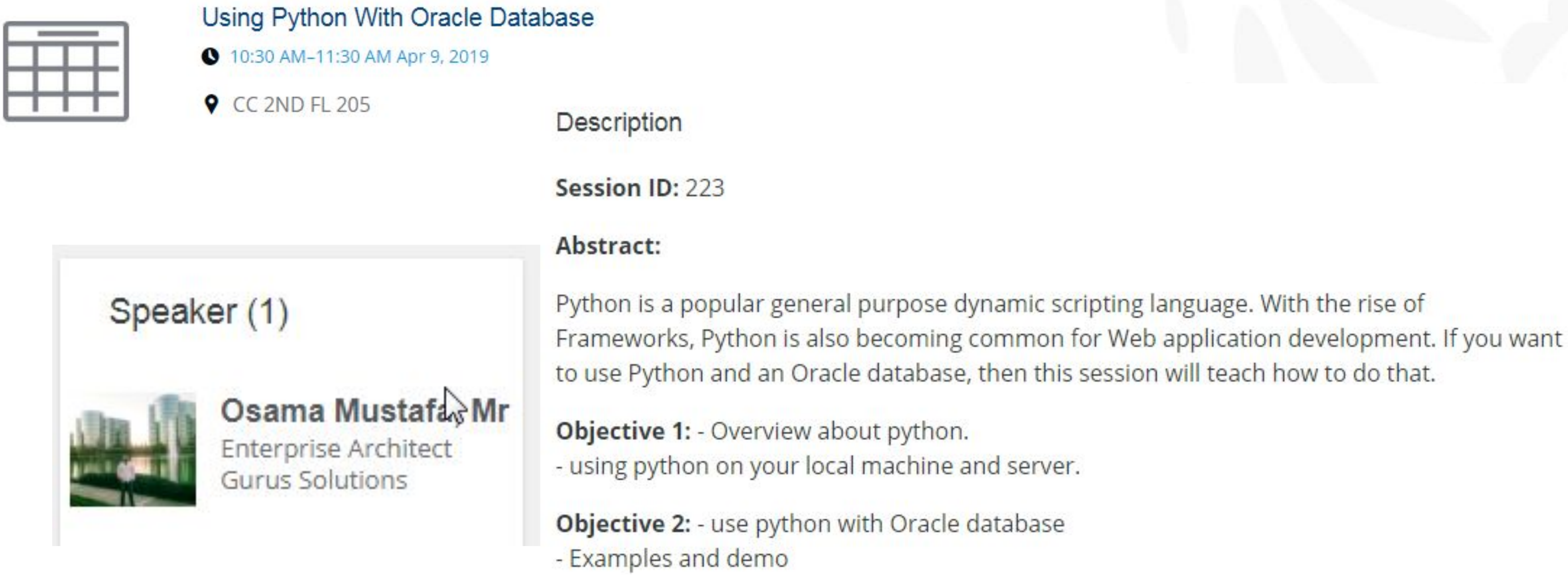


```
from sqlalchemy import create_engine

engine = create_engine('postgresql://scott:tiger@localhost:5432/mydatabase')

engine = create_engine('mysql+mysqldb://scott:tiger@localhost/foo')

engine = create_engine('oracle://scott:tiger@127.0.0.1:1521/sidname')

engine = create_engine('mssql+pyodbc://mydsn')

# sqlite://<nohostname>/<path>
# where <path> is relative:
engine = create_engine('sqlite:///foo.db')

# or absolute, starting with a slash:
engine = create_engine('sqlite:////absolute/path/to/foo.db')
```



```
In [542]: import sqlalchemy as sa

In [543]: pd.read_sql(sa.text('SELECT * FROM data where Col_1=:col1'),
   .....:             engine, params={'col1': 'X'})
   .....:
Out[543]:
   index  id                        Date Col_1  Col_2  Col_3
0      0  26  2010-10-18 00:00:00.000000     X   27.5      1
```

For more DB Connect OAUG 2019 Session



Using Python With Oracle Database

10:30 AM–11:30 AM Apr 9, 2019

CC 2ND FL 205

Speaker (1)

Osama Mustafa Mr

Enterprise Architect

Gurus Solutions

Description

Session ID: 223

Abstract:

Python is a popular general purpose dynamic scripting language. With the rise of Frameworks, Python is also becoming common for Web application development. If you want to use Python and an Oracle database, then this session will teach how to do that.

Objective 1: - Overview about python.
- using python on your local machine and server.

Objective 2: - use python with Oracle database
- Examples and demo

COLLABORATE19

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Installation

#C19TX

Tools

jupyter

ANACONDA.

Google colab

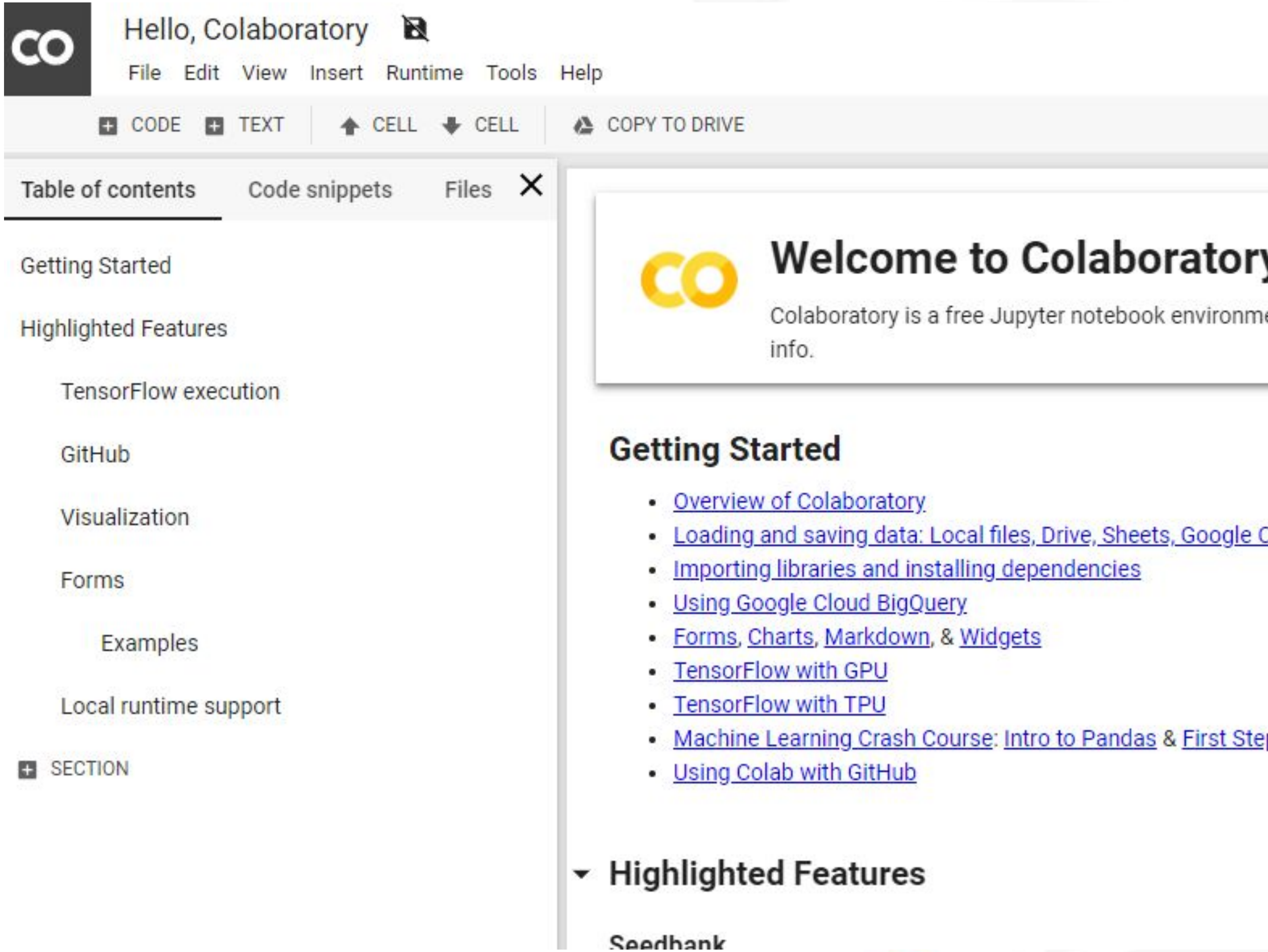
Google Colaboratory

- Demo was performed using Google Colaboratory

❖ No installation required.

❖ Just need google Account.

❖ Free to use.



CO
Hello, Colaboratory
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL COPY TO DRIVE
Table of contents Code snippets Files

Getting Started

Highlighted Features

TensorFlow execution

GitHub

Visualization

Forms

Examples

Local runtime support

Welcome to Colaboratory

Colaboratory is a free Jupyter notebook environm
info.

Getting Started

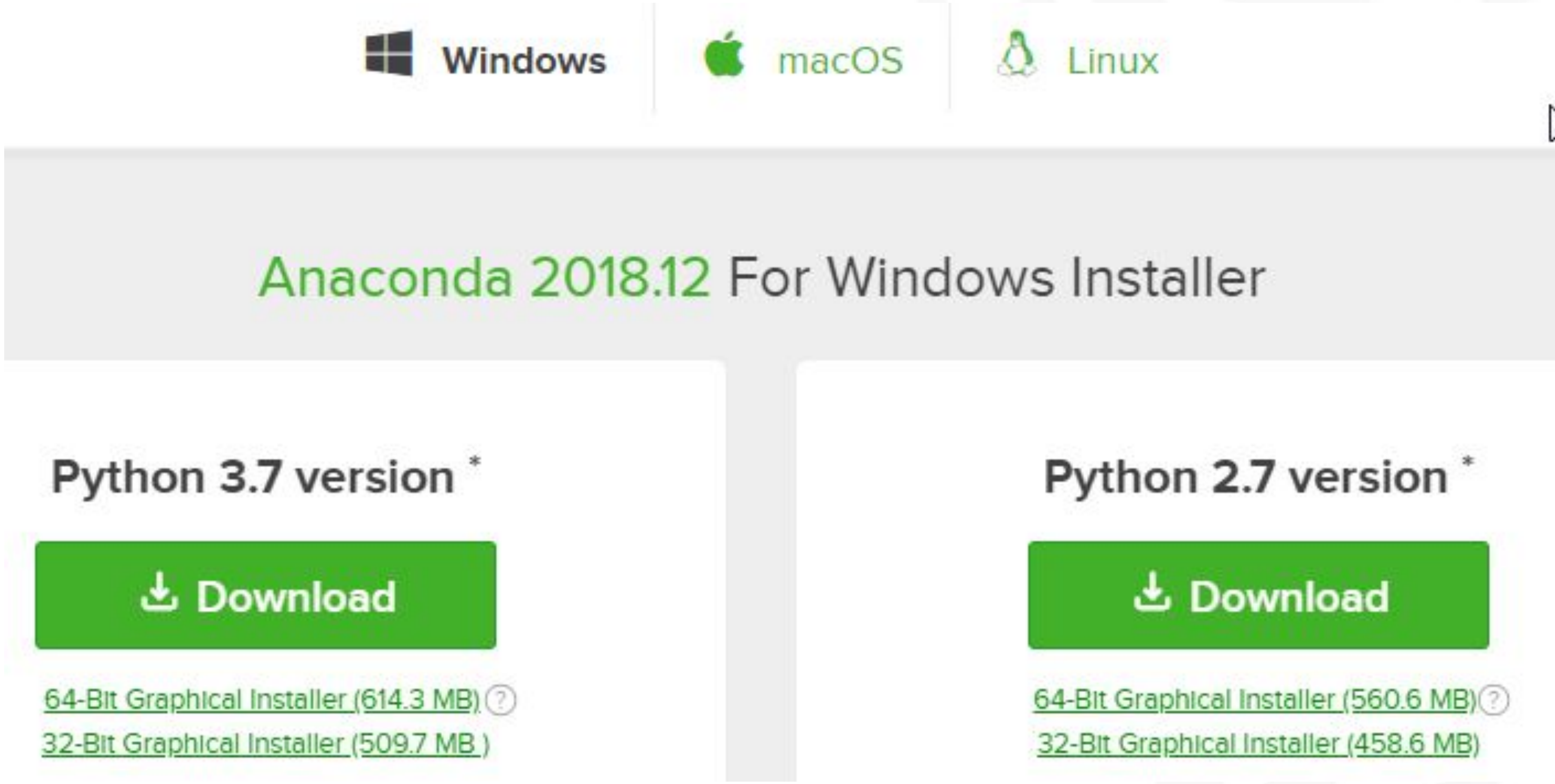
- Overview of Colaboratory
- Loading and saving data: Local files, Drive, Sheets, Google C
- Importing libraries and installing dependencies
- Using Google Cloud BigQuery
- Forms, Charts, Markdown, & Widgets
- TensorFlow with GPU
- TensorFlow with TPU
- Machine Learning Crash Course: Intro to Pandas & First Ste
- Using Colab with GitHub

Highlighted Features

Seedbank

ANACONDA

- Install Anaconda
  - ❖ Prepackaged Installation.
  - ❖ Ready for Data Connection, Analysis, Visualization & Machine Learning.
  - ❖ Free to download.



Anaconda 2018.12 For Windows Installer

Python 3.7 version *

64-Bit Graphical Installer (614.3 MB)

32-Bit Graphical Installer (509.7 MB)

Python 2.7 version *

64-Bit Graphical Installer (560.6 MB)

32-Bit Graphical Installer (458.6 MB)

Why use Pandas

???

Q &A

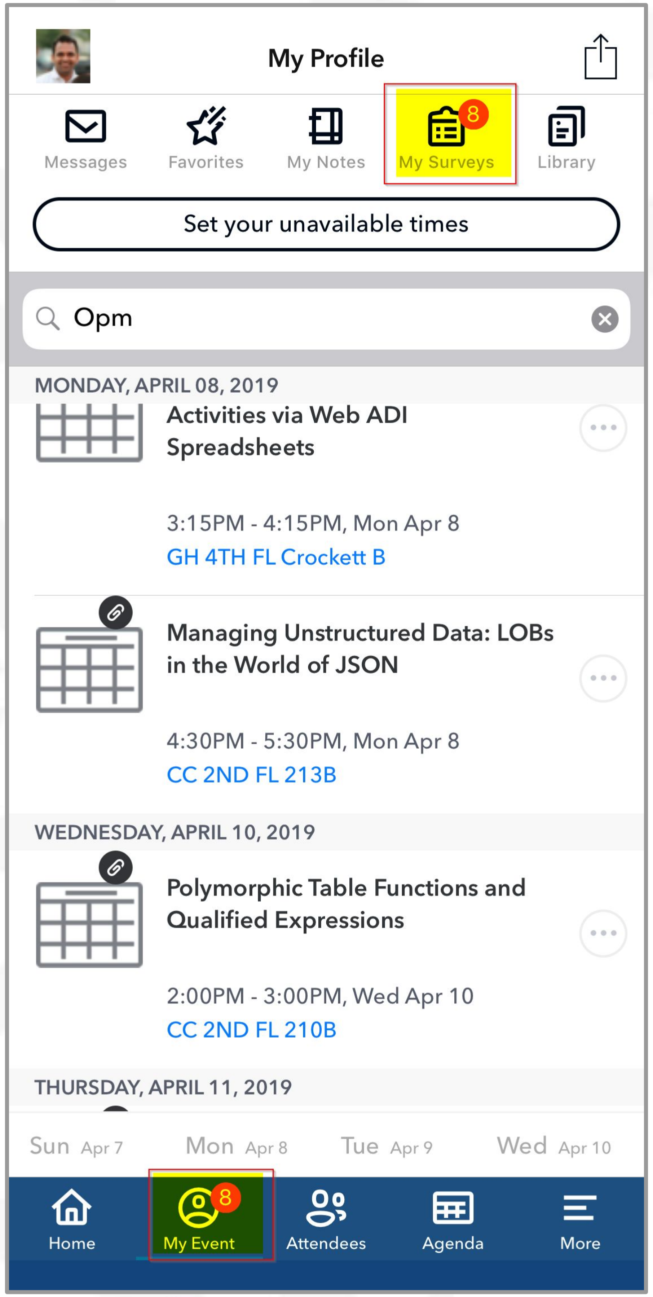
Rama Koganti

linkedin.com/in/ramak919/

Session ID:

11033

Remember to complete your evaluation for this session within the app!



MONDAY, APRIL 08, 2019

Activities via Web ADI Spreadsheets

3:15PM - 4:15PM, Mon Apr 8

GH 4TH FL Crockett B

Managing Unstructured Data: LOBs in the World of JSON

4:30PM - 5:30PM, Mon Apr 8

CC 2ND FL 213B

WEDNESDAY, APRIL 10, 2019

Polymorphic Table Functions and Qualified Expressions

2:00PM - 3:00PM, Wed Apr 10

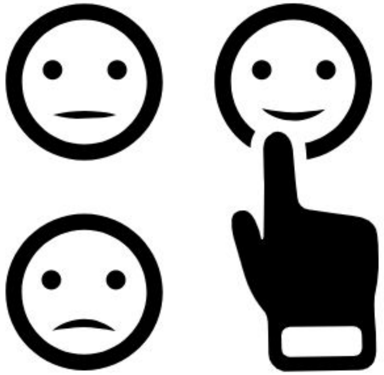
CC 2ND FL 210B

THURSDAY, APRIL 11, 2019

COLLABORATE19

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Feedback



Mobile App

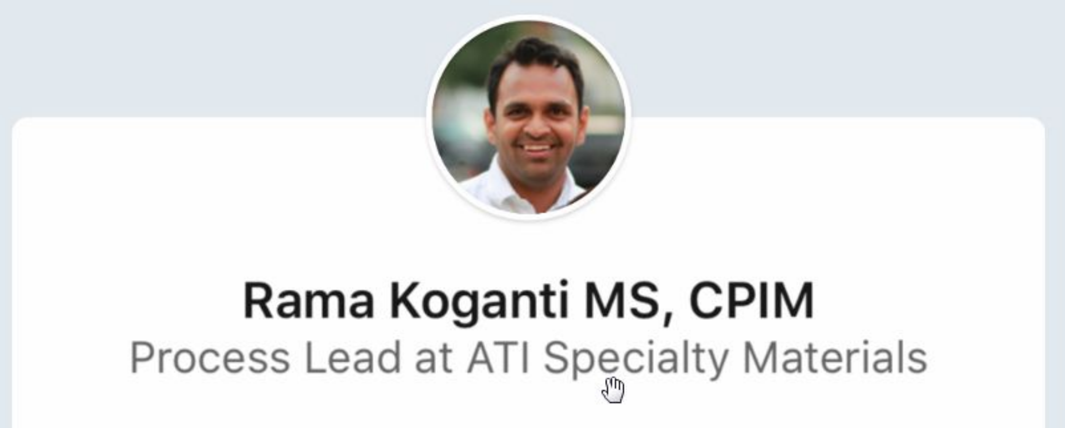
Session ID:

11033

Remember to complete your evaluation for this session within the app!

#C19TX

Scan me to Linkedin



Rama Koganti MS, CPIM

Process Lead at ATI Specialty Materials