databricks Final

(https://databricks.com)

Overview

This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS (https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html) is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

Bringing in DATAFRAME

```
# File location and type
file_location = "/FileStore/tables/heart_disease_uci.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Table

	id	age	sex	dataset	cp	trestbps
1	1	63	Male	Cleveland	typical angina	145
2	2	67	Male	Cleveland	asymptomatic	160
3	3	67	Male	Cleveland	asymptomatic	120
4	4	37	Male	Cleveland	non-anginal	130
5	5	41	Female	Cleveland	atypical angi...	130
6	6	56	Male	Cleveland	atypical angi...	120
7	7	62	Female	Cleveland	asymptomatic	140
8	8	57	Female	Cleveland	asymptomatic	120
9	9	63	Male	Cleveland	asymptomatic	130
10	10	53	Male	Cleveland	asymptomatic	140
11	11	57	Male	Cleveland	asymptomatic	140
12	12	56	Female	Cleveland	atypical angi...	140
13	13	56	Male	Cleveland	non-anginal	130
14	14	44	Male	Cleveland	atypical angi...	120
15	15	52	Male	Cleveland	non-anginal	172

920 rows

Describe the Variables

```
# Describe the variables
variables_description = {
    'age': 'Age of the patient',
    'sex': 'Sex of the patient (1 = male, 0 = female)',
    'cp': 'Chest pain type (1 to 4)',
    'trestbps': 'Resting blood pressure (in mm Hg)',
    'chol': 'Serum cholesterol (in mg/dl)'
}

variables_description
```

```
Out[28]: {'age': 'Age of the patient',
 'sex': 'Sex of the patient (1 = male, 0 = female)',
 'cp': 'Chest pain type (1 to 4)',
 'trestbps': 'Resting blood pressure (in mm Hg)',
 'chol': 'Serum cholesterol (in mg/dl)'}
```

Converting to Pandas

```
import pandas as pd

df = df.toPandas()
print(pandas_df.head())
```

```
  id age     sex    dataset               cp trestbps chol    fbs  \
0  1  63    Male  Cleveland   typical angina      145  233   TRUE
1  2  67    Male  Cleveland     asymptomatic      160  286  FALSE
2  3  67    Male  Cleveland     asymptomatic      120  229  FALSE
3  4  37    Male  Cleveland      non-anginal      130  250  FALSE
4  5  41  Female  Cleveland  atypical angina      130  204  FALSE

          restecg thalch  exang oldpeak        slope ca               thal num
0  lv hypertrophy    150  FALSE     2.3  downsloping  0       fixed defect   0
1  lv hypertrophy    108   TRUE     1.5         flat  3             normal   2
2  lv hypertrophy    129   TRUE     2.6         flat  2  reversable defect   1
3          normal    187  FALSE     3.5  downsloping  0             normal   0
4  lv hypertrophy    172  FALSE     1.4    upsloping  0             normal   0
```

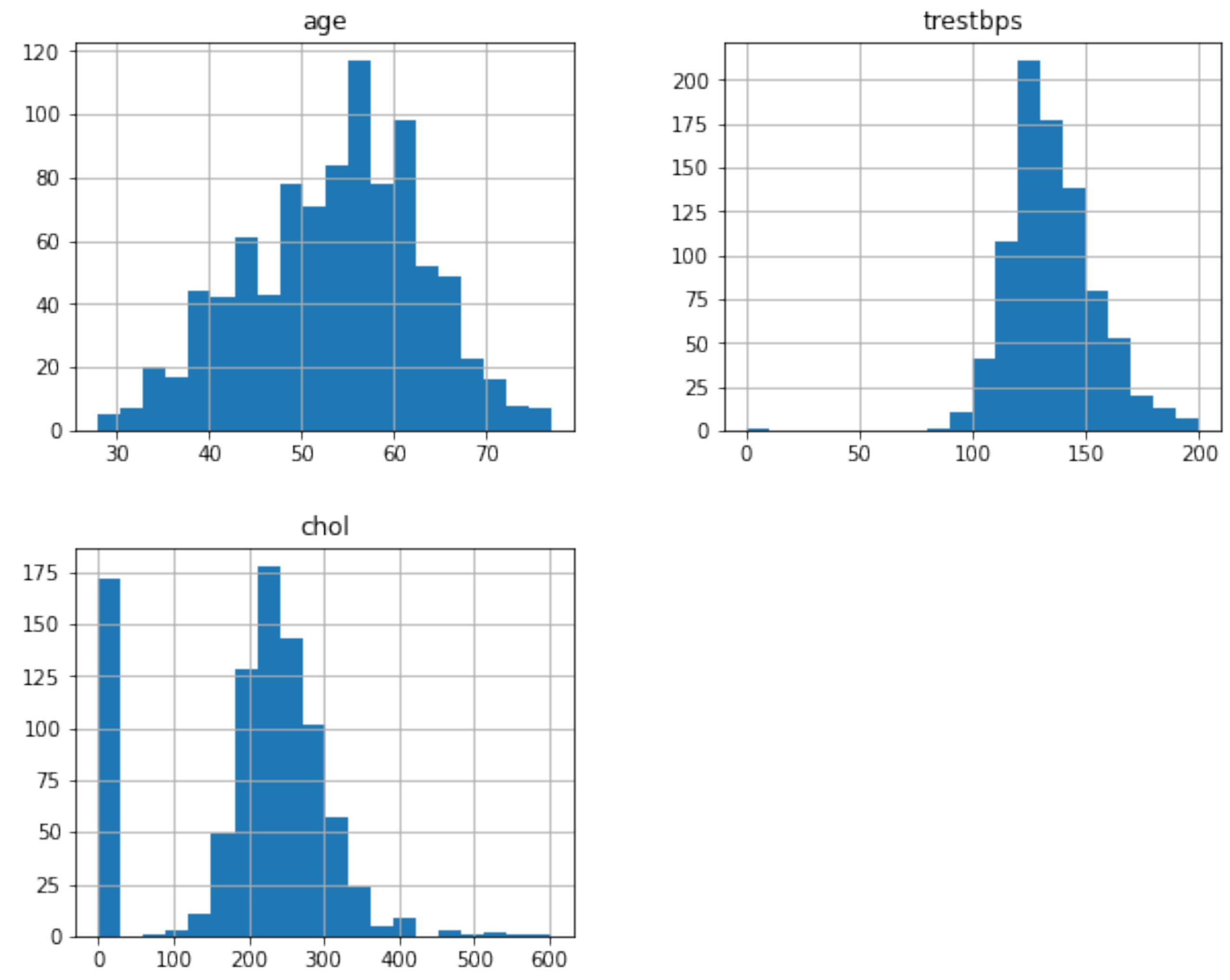
Selecting Certain Variables

```
variables = ['age', 'sex', 'cp', 'trestbps', 'chol']
df_selected = df[variables]

# Display the selected variables
print(df_selected.head())
```

```
   age     sex               cp  trestbps   chol
0   63    Male   typical angina     145.0  233.0
1   67    Male     asymptomatic     160.0  286.0
2   67    Male     asymptomatic     120.0  229.0
3   37    Male      non-anginal     130.0  250.0
4   41  Female  atypical angina     130.0  204.0
```

Histograms and Outliers

```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Plot histograms
df[['age', 'sex', 'cp', 'trestbps', 'chol']].hist(figsize=(10, 8), bins=20)
plt.show()

# Identifying outliers
df[['age', 'trestbps', 'chol']].describe()
```



	age	trestbps	chol
count	920.000000	861.000000	890.000000
mean	53.510870	132.132404	199.130337
std	9.424685	19.066070	110.780810
min	28.000000	0.000000	0.000000
25%	47.000000	120.000000	175.000000
50%	54.000000	130.000000	223.000000
75%	60.000000	140.000000	268.000000
max	77.000000	200.000000	603.000000

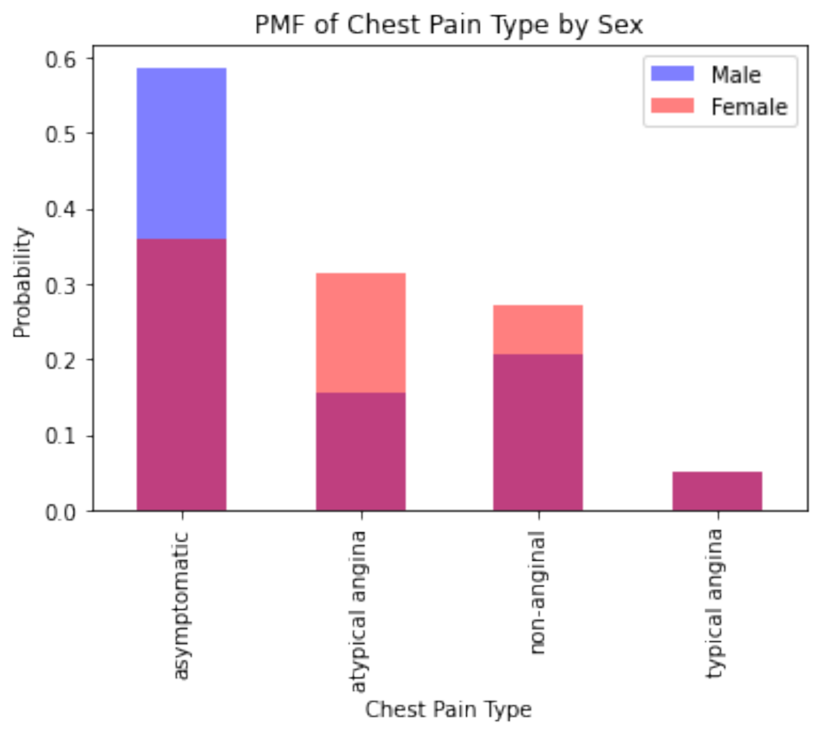
Descriptive Statistics

```
# Calculating and displaying descriptive statistics
descriptive_stats = df_selected.describe()
print(descriptive_stats)
```

```
              age    trestbps        chol
count  920.000000  861.000000  890.000000
mean    53.510870  132.132404  199.130337
std      9.424685   19.066070  110.780810
min     28.000000    0.000000    0.000000
25%     47.000000  120.000000  175.000000
50%     54.000000  130.000000  223.000000
75%     60.000000  140.000000  268.000000
max     77.000000  200.000000  603.000000
```

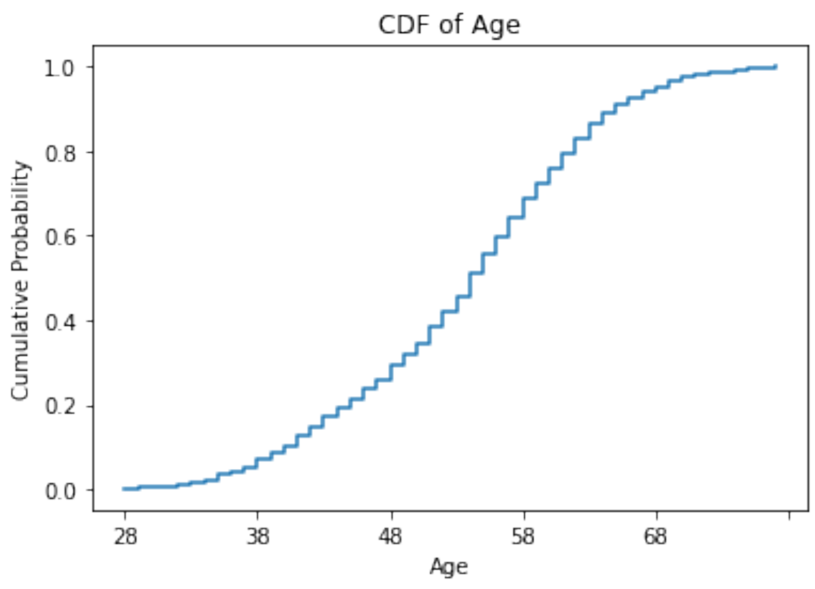
PMF Comparison

```
pmf_cp_male = df[df['sex'] == "Male"]
['cp'].value_counts(normalize=True).sort_index()
pmf_cp_female = df[df['sex'] == "Female"]
['cp'].value_counts(normalize=True).sort_index()

# Plotting the PMF comparison
pmf_cp_male.plot(kind='bar', color='blue', alpha=0.5, label='Male')
pmf_cp_female.plot(kind='bar', color='red', alpha=0.5, label='Female')
plt.title('PMF of Chest Pain Type by Sex')
plt.xlabel('Chest Pain Type')
plt.ylabel('Probability')
plt.legend()
plt.show()
```



CDF

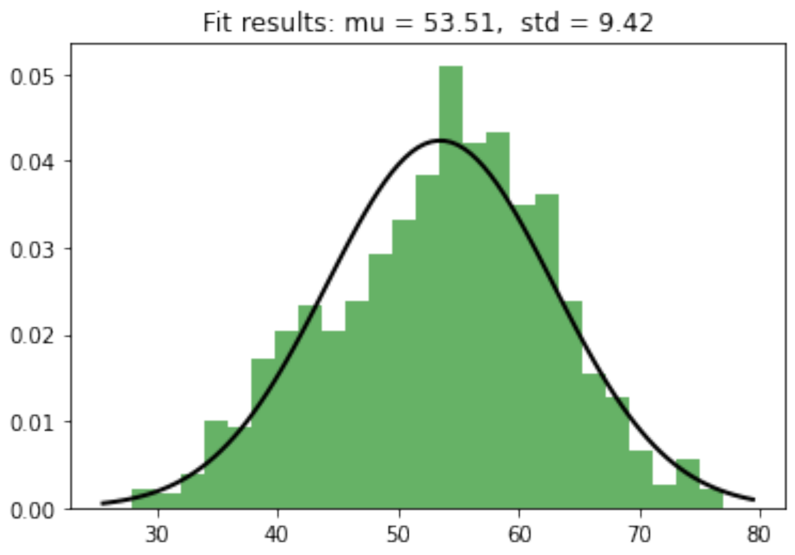
```
age_cdf = df['age'].value_counts().sort_index().cumsum() / len(df)
age_cdf.plot(drawstyle='steps-post')
plt.title('CDF of Age')
plt.xlabel('Age')
plt.ylabel('Cumulative Probability')
plt.show()
```



Analytical Distribution

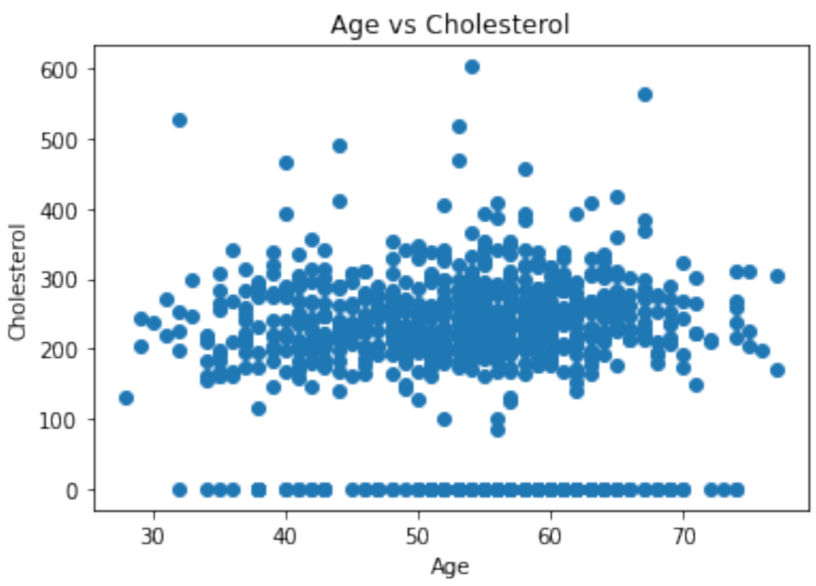
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# Ensure the 'age' column is numeric
df_selected['age'] = pd.to_numeric(df_selected['age'], errors='coerce')

# Drop any rows with NaN values in the 'age' column
df_selected = df_selected.dropna(subset=['age'])

# Fit a normal distribution to the 'age' data
mu, std = stats.norm.fit(df_selected['age'])

# Plot the histogram and the fitted distribution
plt.hist(df_selected['age'], bins=25, density=True, alpha=0.6, color='g')

# Plot the probability density function (PDF)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.title(f"Fit results: mu = {mu:.2f},  std = {std:.2f}")
plt.show()
```

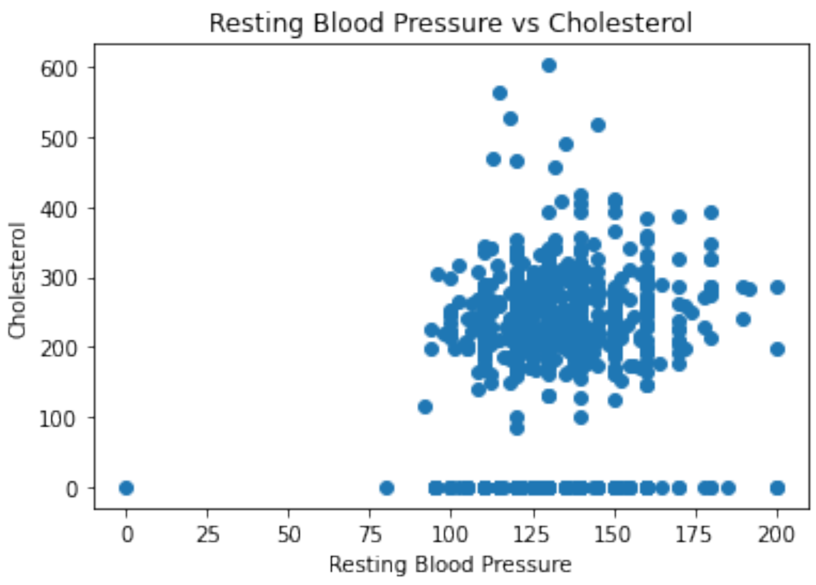
```
<command-4255733837626512>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_selected['age'] = pd.to_numeric(df_selected['age'], errors='coerce')
```



Scatter Plots

```
plt.scatter(df['age'], df['chol'])
plt.title('Age vs Cholesterol')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.show()

# Scatter plot for resting blood pressure vs cholesterol
plt.scatter(df['trestbps'], df['chol'])
plt.title('Resting Blood Pressure vs Cholesterol')
plt.xlabel('Resting Blood Pressure')
plt.ylabel('Cholesterol')
plt.show()
```





Hypothesis Testing

```
from scipy.stats import ttest_ind

# Check for NaN values and drop them
chol_male = df[df['sex'] == "Male"]['chol'].dropna()
chol_female = df[df['sex'] == "Female"]['chol'].dropna()

# Check that the groups have valid data
if len(chol_male) == 0 or len(chol_female) == 0:
    print("One of the groups has no data. Hypothesis test cannot be performed.")
elif chol_male.mean() == chol_female.mean():
    print("The two groups have identical means. Hypothesis test results will be
NaN.")
else:
    # Perform the hypothesis test
    t_stat, p_value = ttest_ind(chol_male, chol_female)
    print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

```
T-statistic: -5.9967029323480725, P-value: 2.927317272597984e-09
```

Regression Analysis

```
Df Residuals:                     830   BIC:                             2436.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.0224      0.294     -3.474      0.001      -1.600      -0.445
age            0.0354      0.004      8.929      0.000       0.028       0.043
trestbps       0.0045      0.002      2.291      0.022       0.001       0.008
chol          -0.0024      0.000     -7.257      0.000      -0.003      -0.002
==============================================================================
Omnibus:                       86.106   Durbin-Watson:                   1.728
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              110.970
Skew:                           0.875   Prob(JB):                     8.00e-25
Kurtosis:                       3.363   Cond. No.                     2.17e+03
==============================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 2.17e+03. This might indicate that there are
```