College of Professional Studies
Northeastern University San Jose

MPS Analytics

Course: ALY6110 – Data Management and Big Data

Assignment:

Module 4 Lab 1

Submitted on:

June 25th, 2023

Submitted to:

Professor: BEHZAD AHMADI

Submitted by:

NIKSHITA RANGANATHAN

INTRODUCTION

Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance.

Spark is designed to be fast and general-purpose, and it can be used for a variety of data processing tasks. It enables batch processing, stream processing, SQL queries, and machine learning tasks on large datasets.

Here are some of the benefits of using Spark:

- Speed: Spark is much faster than Hadoop MapReduce. This is because Spark uses in-memory caching, which allows it to access data much faster than Hadoop, which stores data on disk.
- Scalability: Spark can scale to handle trillions of records. This makes it a good choice for large-scale data processing applications.
- Fault tolerance: Spark is fault-tolerant. This means that it can continue to operate even if some of the nodes in a cluster fail.
- Flexibility: Spark provides a wide range of APIs, making it a good choice for a variety of use cases.

About this Dataset:

Dataset – 1: Census Income

The US Adult Income dataset is a publicly available dataset that was extracted from the 1994 US Census database. The dataset contains 48,842 rows and 14 columns of anonymized information about individuals, such as their age, education, race, occupation, and income. The target variable is "income," indicating whether an individual earns more than $50,000 per year or not.

The dataset is a valuable resource for everyone who is interested in studying the factors that affect income. It can be used to build predictive models that can be used to predict a person's income based on their personal characteristics.

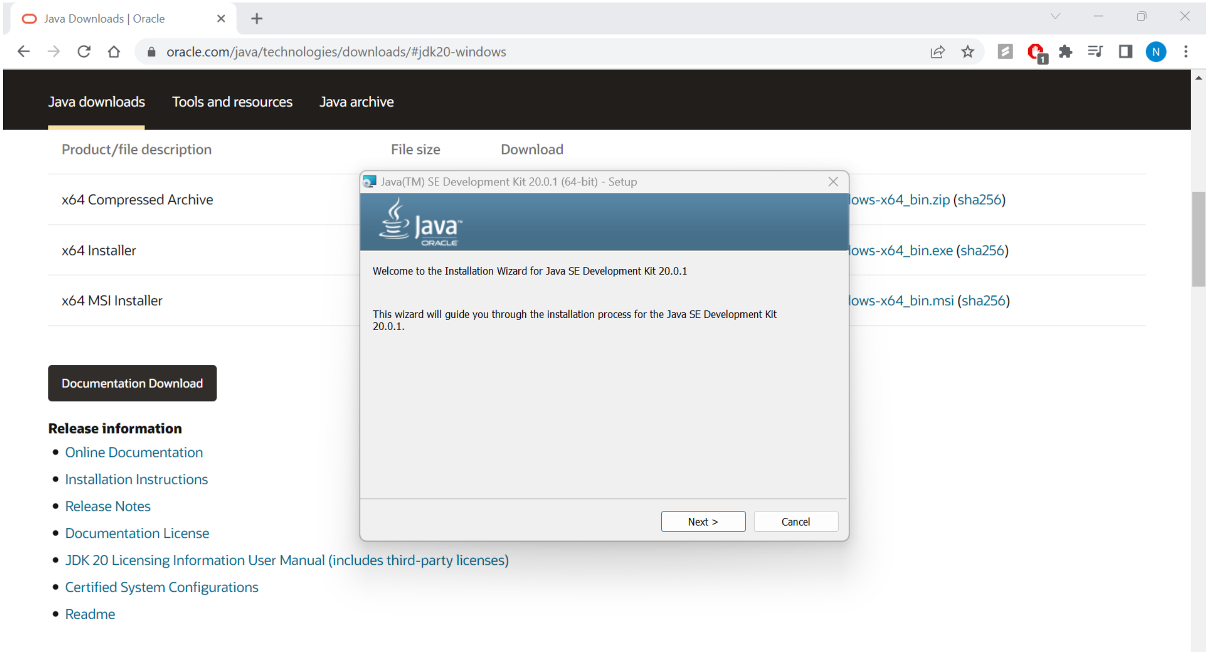
Dataset – 2: Zillow

The Zillow dataset provides valuable information on real estate market trends and housing prices across different regions in the United States.

The dataset contains several columns that offer insights into various aspects of the housing market. It is valuable for conducting market research, predicting home values, and understanding the dynamics of real estate in different regions over time.

ANALYSIS & RESULTS

1. Installation of Java, Python and Apache Spark




```
(pyspark-env) C:\Users\14086>python
Python 3.11.3 | packaged by Anaconda, Inc. | (main, May 15 2023, 15:41:31) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyspark
>>> from pyspark
  File "<stdin>", line 1
    from pyspark
                ^
SyntaxError: invalid syntax
>>> from pyspark.sql import SparkSession
>>> exit()
```

```
Anaconda Prompt (anaconda

(base) C:\Users\14086>java --version
java 20.0.1 2023-04-18
Java(TM) SE Runtime Environment (build 20.0.1+9-29)
Java HotSpot(TM) 64-Bit Server VM (build 20.0.1+9-29, mixed mode, sharing)

(base) C:\Users\14086>pyspark
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/06/20 13:07:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where a
pplicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.4.0
      /_/

Using Python version 3.9.12 (main, Apr  4 2022 05:22:27)
Spark context Web UI available at http://10.0.0.60:4040
Spark context available as 'sc' (master = local[*], app id = local-1687291672509).
SparkSession available as 'spark'.
>>> 23/06/20 13:08:02 WARN GarbageCollectionMetrics: To enable non-built-in garbage collector(s) List(G1 Concurrent GC), users should
 configure it(them) to spark.eventLog.gcMetrics.youngGenerationGarbageCollectors or spark.eventLog.gcMetrics.oldGenerationGarbageColl
ectors
```

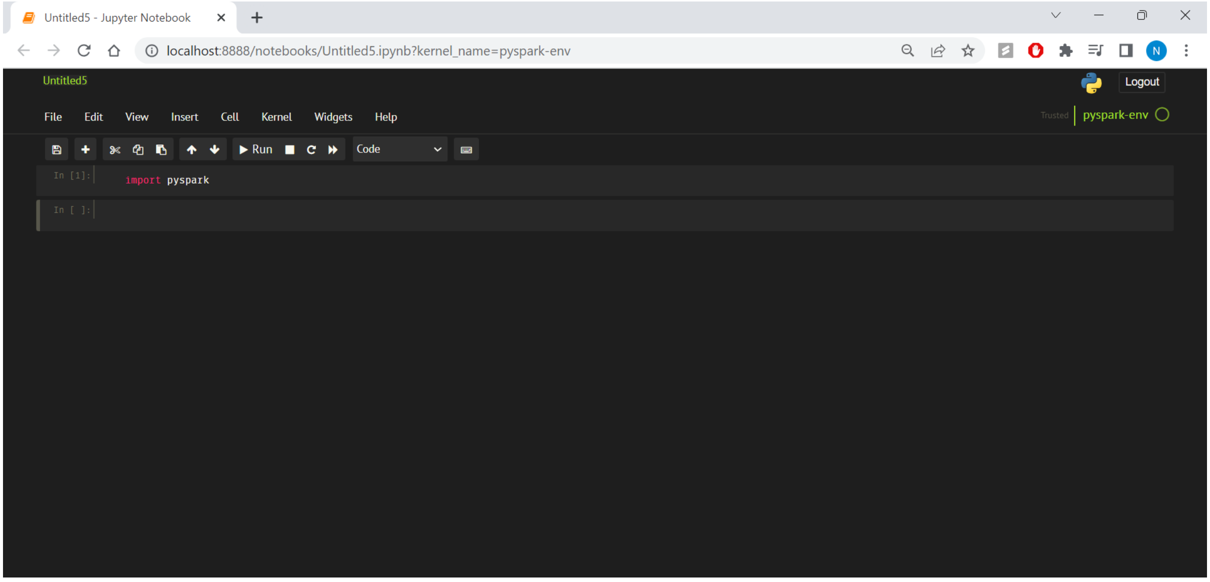
2. Installation of pyspark in Anaconda

```
(base) C:\Users\14086>conda activate pyspark-env

(pyspark-env) C:\Users\14086>conda install pyspark
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.5.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: C:\Users\14086\anaconda3\envs\pyspark-env

  added / updated specs:
    - pyspark


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    abseil-cpp-20211102.0      |       hd77b12b_0         1.7 MB
    arrow-cpp-11.0.0           |       ha81ea56_1         7.3 MB
    aws-c-common-0.6.8         |       h2bbff1b_1         168 KB
    aws-c-event-stream-0.1.6   |       hd77b12b_6          28 KB
    aws-checksums-0.1.11       |       h2bbff1b_2          52 KB
    aws-sdk-cpp-1.8.185        |       hd77b12b_1         2.9 MB
```

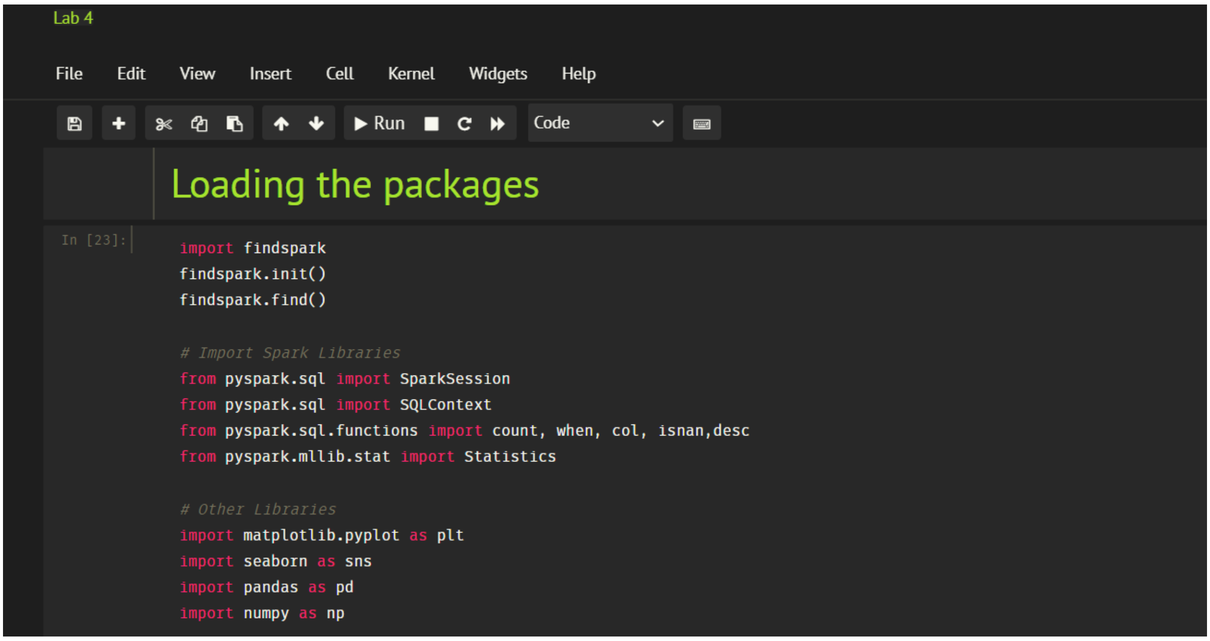
3. Installation of package "findspark" in the Anaconda environment

```
(pyspark-env) C:\Users\14086>conda install -c conda-forge findspark
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.5.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: C:\Users\14086\anaconda3\envs\pyspark-env

  added / updated specs:
    - findspark
```

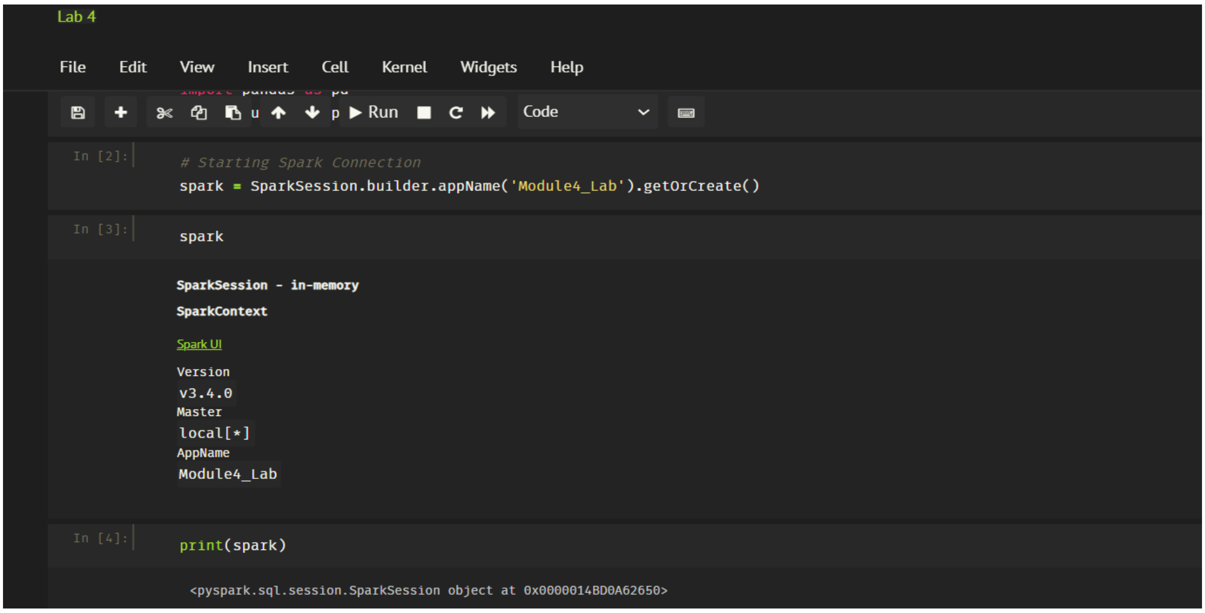
4. Checking if Pyspark has been installed properly



5. Importing Libraries of Spark, Visualization, and Pandas-NumPy



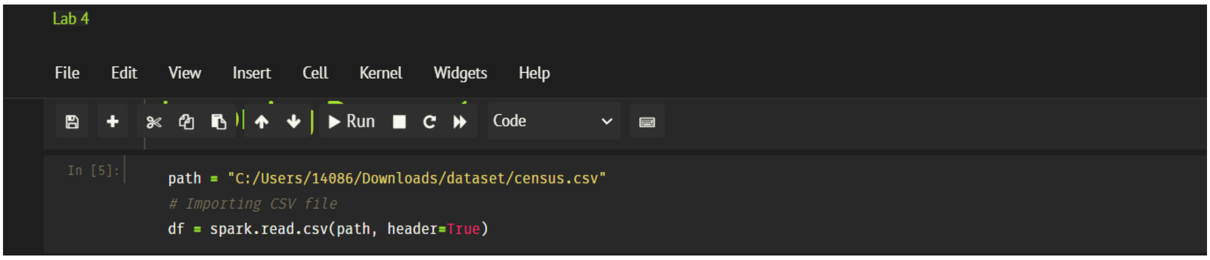
6. Starting "Apache Spark" connection and checking its setup and configuration details




```
In [2]:
# Starting Spark Connection
spark = SparkSession.builder.appName('Module4_Lab').getOrCreate()

In [3]:
spark

SparkSession - in-memory
SparkContext
Spark UI
Version
v3.4.0
Master
local[*]
AppName
Module4_Lab

In [4]:
print(spark)

<pyspark.sql.session.SparkSession object at 0x0000014BD0A62650>
```

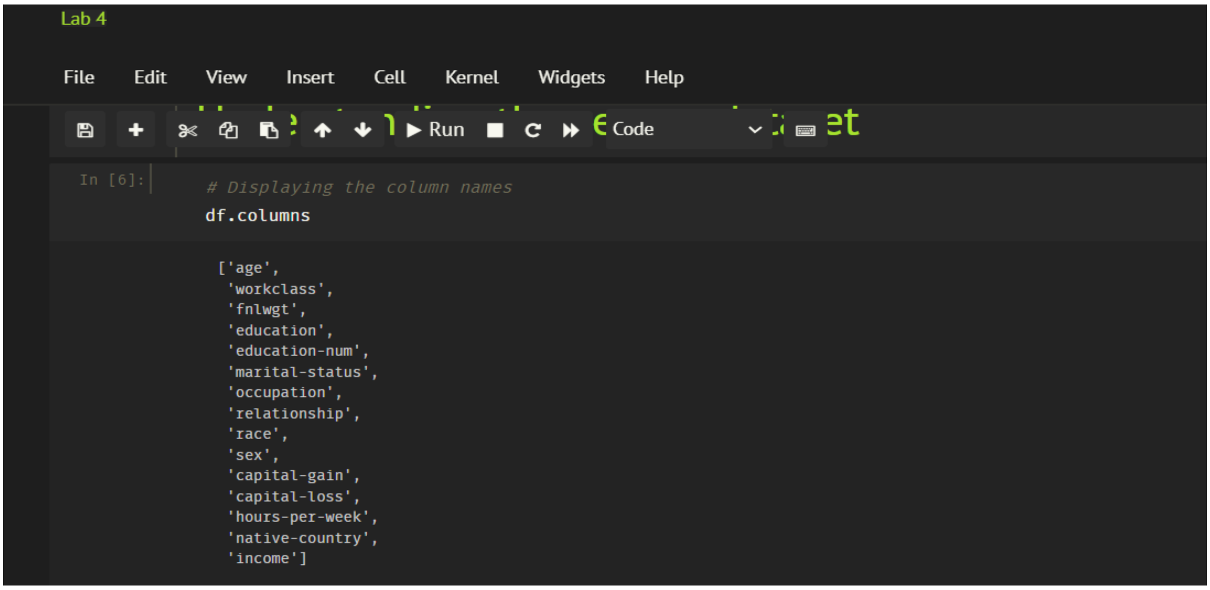
Dataset - 1

7. Importing Census Income data




```
In [5]:
path = "C:/Users/14086/Downloads/dataset/census.csv"
# Importing CSV file
df = spark.read.csv(path, header=True)
```

8. Columns of the census dataset




```
In [6]:
# Displaying the column names
df.columns

['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country',
 'income']
```

9. Schema of census dataset

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [7]:
```python
# schema of df
df.printSchema()
```

```
root
 |-- age: string (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: string (nullable = true)
 |-- education: string (nullable = true)
 |-- education-num: string (nullable = true)
 |-- marital-status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital-gain: string (nullable = true)
 |-- capital-loss: string (nullable = true)
 |-- hours-per-week: string (nullable = true)
 |-- native-country: string (nullable = true)
 |-- income: string (nullable = true)
```

10. Count of rows and columns

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [9]:
```python
# Checking the number of rows
df.count()
```

```
48842
```

In [10]:
```python
# Get the number of columns
num_columns = len(df.columns)

print("Number of columns:", num_columns)
```

```
Number of columns: 15
```

11. First few records of census dataset

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [11]:
```python
# The first 5 rows of df
df.show(5)
```

```
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
|age|workclass|fnlwgt|   education|education-num|    marital-status|       occupation|relationship| race|   sex|capital-gain|capital-loss|hours-per-week|native-country|income|
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
| 25|  Private|226802|        11th|            7|     Never-married|Machine-op-inspct|   Own-child|Black|  Male|           0|           0|            40| United-States| <=50K|
| 38|  Private| 89814|     HS-grad|            9|Married-civ-spouse|  Farming-fishing|     Husband|White|  Male|           0|           0|            50| United-States| <=50K|
| 28|Local-gov|336951|  Assoc-acdm|           12|Married-civ-spouse|  Protective-serv|     Husband|White|  Male|           0|           0|            40| United-States|  >50K|
| 44|  Private|160323|Some-college|           10|Married-civ-spouse|Machine-op-inspct|     Husband|Black|  Male|        7688|           0|            40| United-States|  >50K|
| 18|        ?|103497|Some-college|           10|     Never-married|                ?|   Own-child|White|Female|           0|           0|            30| United-States| <=50K|
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
only showing top 5 rows
```

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [12]:
```
df.limit(10).toPandas().T
```

	0	1	2	3	4	5	6	7	8	9
age	25	38	28	44	18	34	29	63	24	55
workclass	Private	Private	Local-gov	Private	?	Private	?	Self-emp-not-inc	Private	Private
fnlwgt	226802	89814	336951	160323	103497	198693	227026	104626	369667	104996
education	11th	HS-grad	Assoc-acdm	Some-college	Some-college	10th	HS-grad	Prof-school	Some-college	7th-8th
education-num	7	9	12	10	10	6	9	15	10	4
marital-status	Never-married	Married-civ-spouse	Married-civ-spouse	Married-civ-spouse	Never-married	Never-married	Never-married	Married-civ-spouse	Never-married	Married-civ-spouse
occupation	Machine-op-inspct	Farming-fishing	Protective-serv	Machine-op-inspct	?	Other-service	?	Prof-specialty	Other-service	Craft-repair
relationship	Own-child	Husband	Husband	Husband	Own-child	Not-in-family	Unmarried	Husband	Unmarried	Husband
race	Black	White	White	Black	White	White	Black	White	White	White
sex	Male	Male	Male	Male	Female	Male	Male	Male	Female	Male
capital-gain	0	0	0	7688	0	0	0	3103	0	0
capital-loss	0	0	0	0	0	0	0	0	0	0
hours-per-week	40	50	40	40	30	30	40	32	40	10
native-country	United-States	United-States	United-States	United-States	United-States	United-States	United-States	United-States	United-States	United-States
income	<=50K	<=50K	>50K	>50K	<=50K	<=50K	<=50K	>50K	<=50K	<=50K

12. Descriptive statistics of the census dataset

Age:

- The average age is approximately 38.64 years.
- The age values range from 17 to 90 years.

Capital Gain:

- The average capital gain is approximately 1,079.07.
- The capital gain values range from 0 to 99,999.

Capital Loss:

- The average capital loss is approximately 87.50.
- The capital loss values range from 0 to 974.

Hours Per Week:

- The average hours worked per week is approximately 40.42.
- The hours worked per week values range from 1 to 99.

In [13]:
```
df.describe("age","fnlwgt","education-num","capital-gain","capital-loss","hours-per-week").show()
```

```
+-------+-----------------+------------------+------------------+------------------+-----------------+-----------------+
|summary|              age|            fnlwgt|     education-num|      capital-gain|     capital-loss|   hours-per-week|
+-------+-----------------+------------------+------------------+------------------+-----------------+-----------------+
|  count|            48842|             48842|             48842|             48842|            48842|            48842|
|   mean| 38.64358543876172|189664.13459727284|10.078088530363212|1079.0676262233324| 87.50231358257237|40.422382375824085|
| stddev|13.710509934443518|105604.02542315764|2.5709727555922575| 7452.019057655416|403.00455212435935| 12.3914440242523|
|    min|               17|            100009|                 1|                 0|                0|                1|
|    max|               90|             99987|                 9|             99999|              974|               99|
+-------+-----------------+------------------+------------------+------------------+-----------------+-----------------+
```

13. Data Cleaning – Handling Null / NA / "?" values

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

In [14]:

```
df1 = df.select([count(when(col(c).contains('None') | \
                            col(c).contains('NULL') | \
                            (col(c) == '' ) | \
                            col(c).isNull() | \
                            isnan(c), c
                           )).alias(c)
                    for c in df.columns])
df1.show()
df = df.dropna()
```

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

In [15]:

```
df2 = df.select([count(when(col(c).contains('?'), c
                           )).alias(c)
                    for c in df.columns])
df2.show()
```

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	2799	0	0	0	0	2809	0	0	0	0	0	0	857	0

14. Data Cleaning – Handling duplicate rows

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

Handling Duplicate Rows

In [16]:

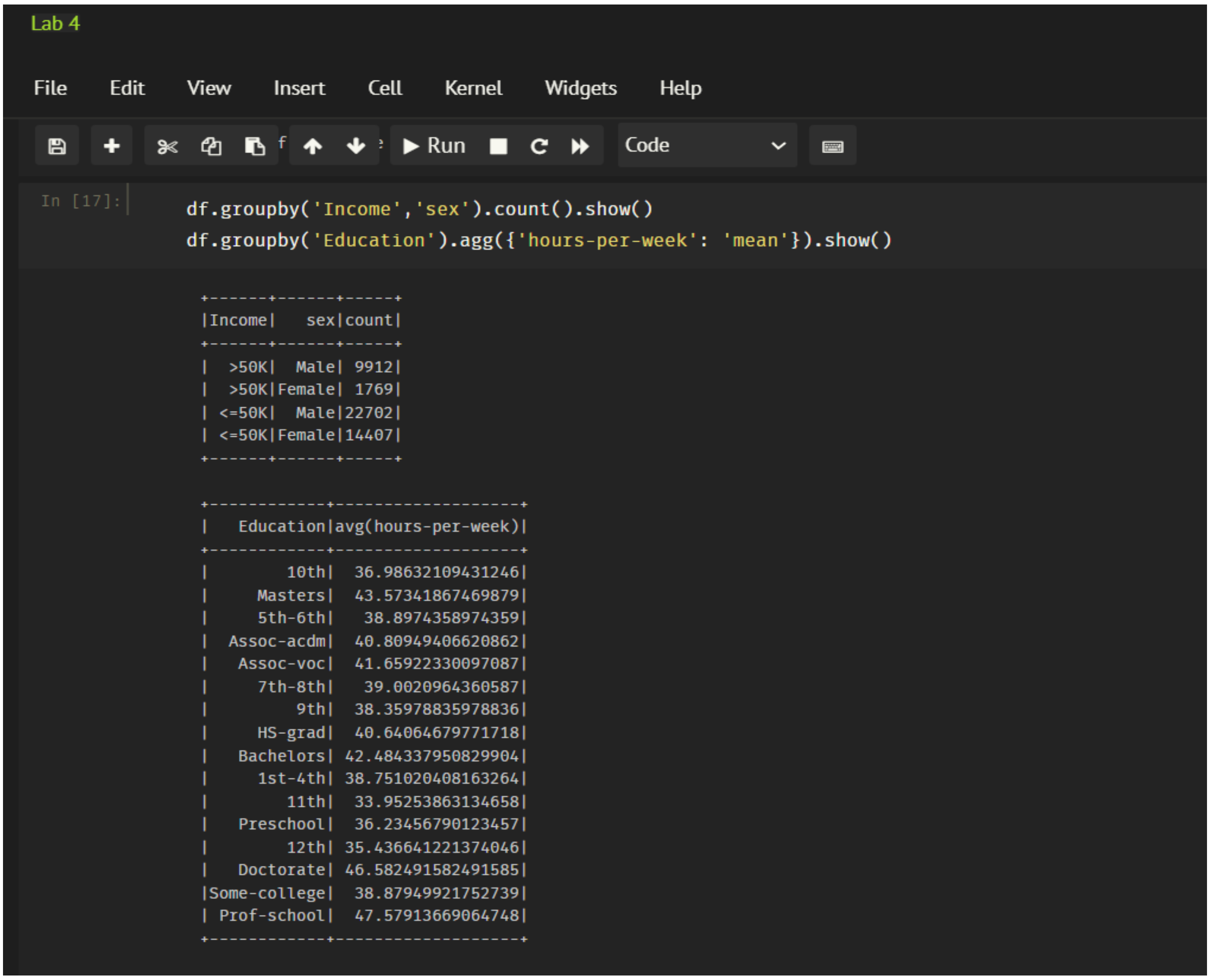
```
df_distinct = df.distinct()

# Get the count of duplicate rows
num_duplicates = df.count() - df_distinct.count()

# Print the number of duplicate rows
print("Number of duplicate rows:", num_duplicates)

# Removing duplicate rows
df = df.dropDuplicates()
```

```
Number of duplicate rows: 52
```

15. Groupby

This table suggests that there is a higher proportion of males with an income above $50,000 than females. This could be due to a number of factors, such as differences in education, occupation, or experience.

The education level "Prof-school" has the highest average hours worked per week with 47.58 hours. This suggests that individuals with professional degrees, such as those in law or medicine, tend to have demanding work schedules that require longer hours.



16. Correlation matrix and plot

Lab 4

File Edit View Insert Cell Kernel Widgets Help

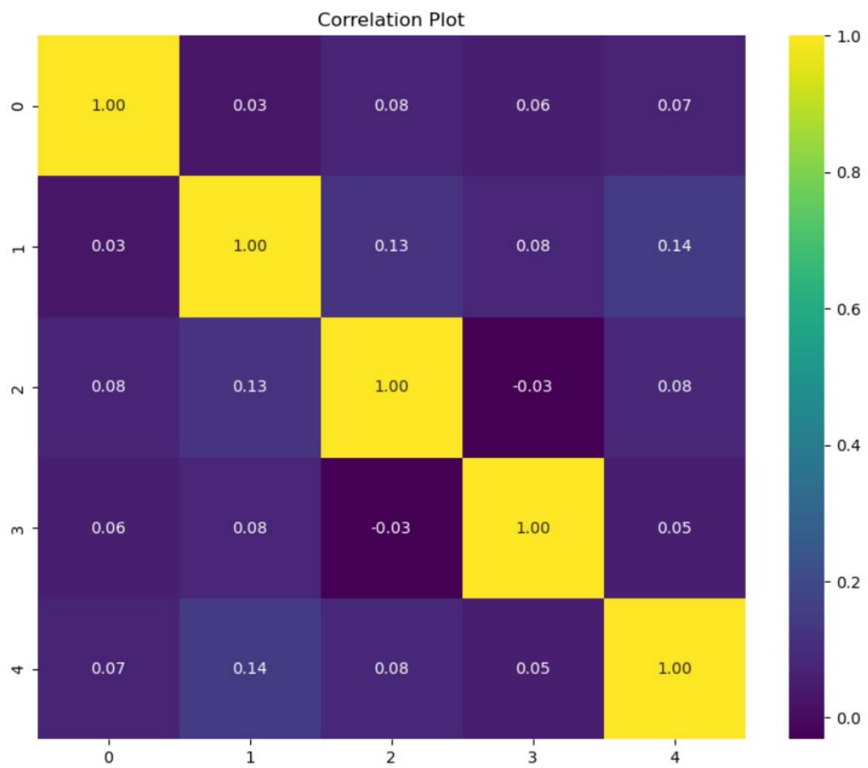
Correlation Matrix

In [18]:

```python
# select variables to check correlation
df_features = df.select("age", "education-num", "capital-gain", "capital-loss", "hours-per-week")
df_features

# create RDD table for correlation calculation
rdd_table = df_features.rdd.map(lambda row: row[0:])

# get the correlation matrix
corr_mat=Statistics.corr(rdd_table, method="pearson")
corr_mat
```

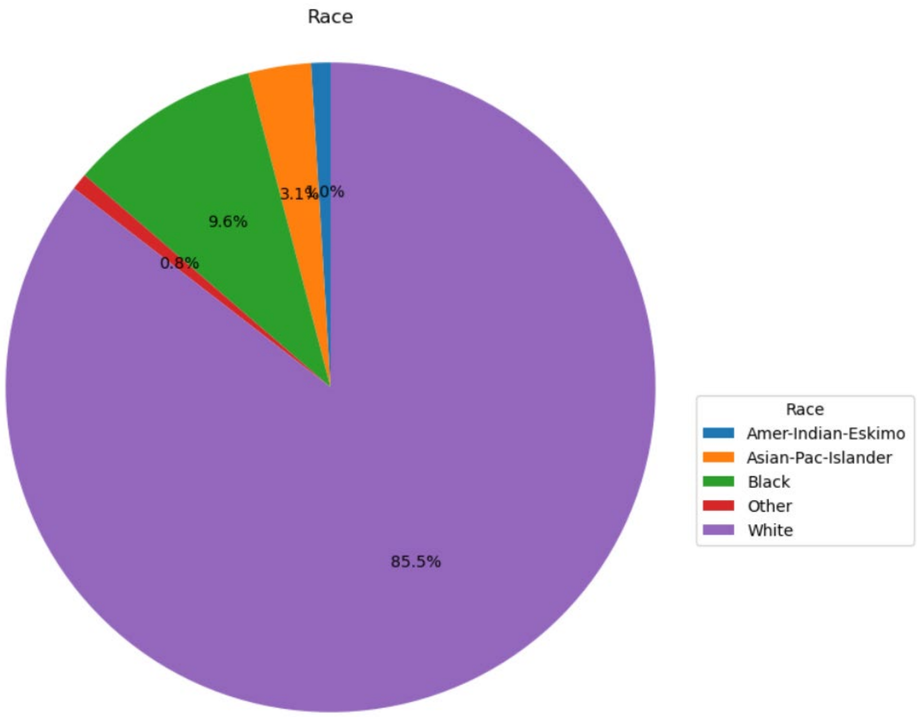
```
array([[ 1.        ,  0.03063457,  0.07718504,  0.05683798,  0.07122349],
       [ 0.03063457,  1.        ,  0.12521918,  0.08098558,  0.14391532],
       [ 0.07718504,  0.12521918,  1.        , -0.03147542,  0.08215248],
       [ 0.05683798,  0.08098558, -0.03147542,  1.        ,  0.05443052],
       [ 0.07122349,  0.14391532,  0.08215248,  0.05443052,  1.        ]])
```

In this dataset, we can observe that there is no significant correlation among the variables.

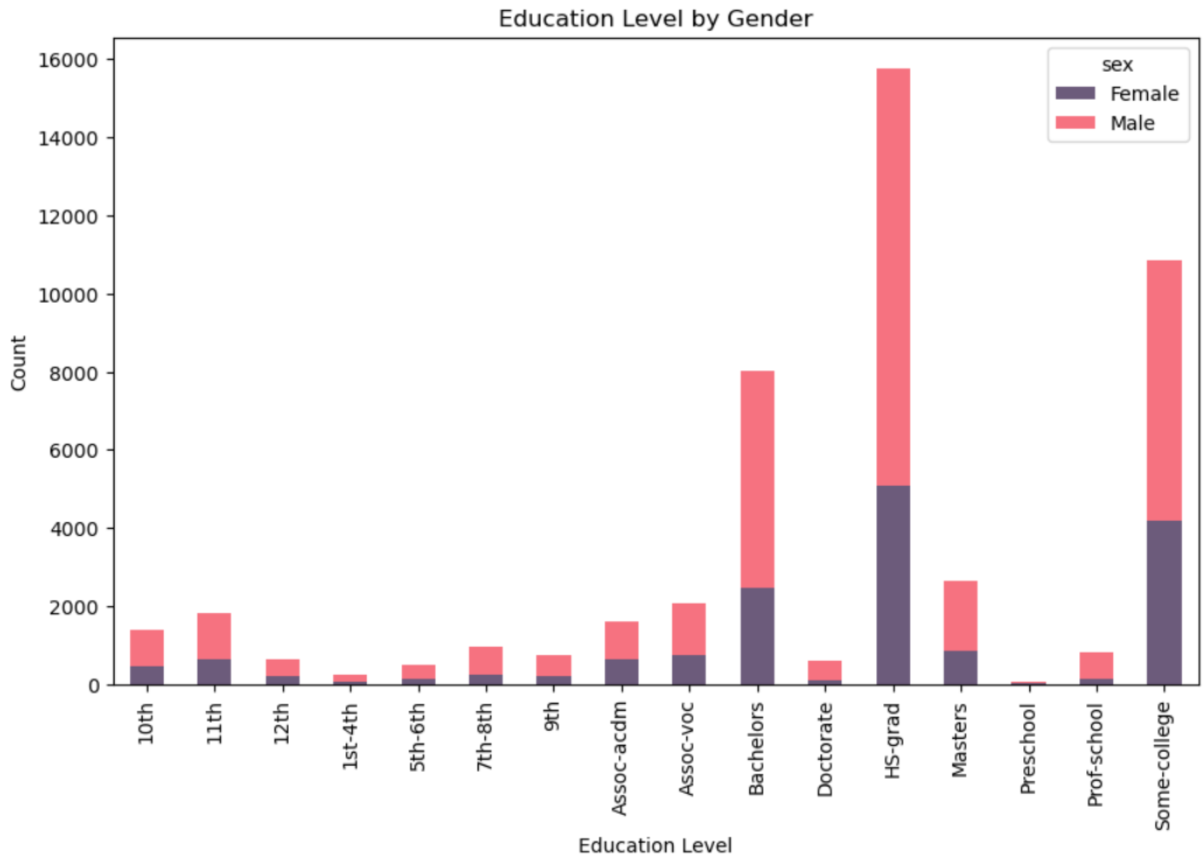


17. Data Visualizations

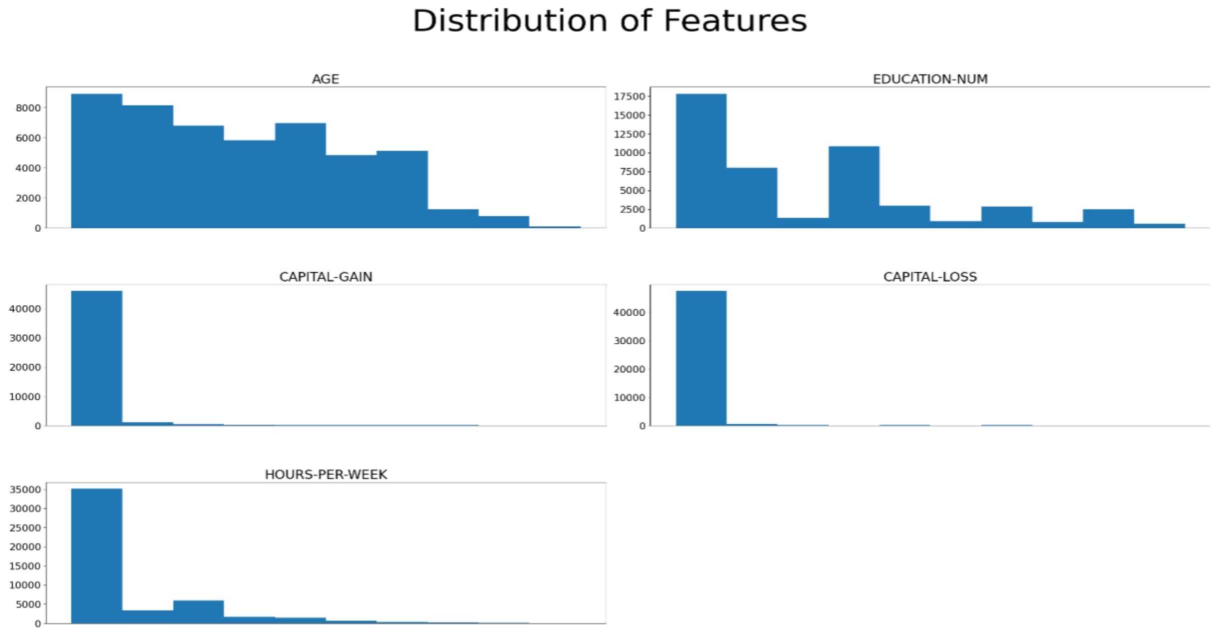
- **Pie chart with the distribution of race**



- **Barplot (Education level by gender)**



- **Histogram (numerical features)**



<u>Dataset - 2</u>

18. Importing the Zillow dataset

```
In [24]:  path = "C:/Users/14086/Downloads/dataset/zillow.csv"
          # Import CSV file
          zillow_df = spark.read.csv(path, header=True)
```

19. Columns of the Zillow dataset

```
In [26]:  # Displaying the column names
          zillow_df.columns

          ['Date',
           'RegionID',
           'RegionName',
           'State',
           'Metro',
           'County',
           'City',
           'SizeRank',
           'Zhvi',
           'MoM',
           'QoQ',
           'YoY',
           '5Year',
           '10Year',
           'PeakMonth',
           'PeakQuarter',
           'PeakZHVI',
           'PctFallFromPeak',
           'LastTimeAtCurrZHVI']
```

20. Schema of Zillow dataset

```
In [27]:  #schema of dataset 2
          zillow_df.printSchema()

root
 |-- Date: string (nullable = true)
 |-- RegionID: string (nullable = true)
 |-- RegionName: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Metro: string (nullable = true)
 |-- County: string (nullable = true)
 |-- City: string (nullable = true)
 |-- SizeRank: string (nullable = true)
 |-- Zhvi: string (nullable = true)
 |-- MoM: string (nullable = true)
 |-- QoQ: string (nullable = true)
 |-- YoY: string (nullable = true)
 |-- 5Year: string (nullable = true)
 |-- 10Year: string (nullable = true)
 |-- PeakMonth: string (nullable = true)
 |-- PeakQuarter: string (nullable = true)
 |-- PeakZHVI: string (nullable = true)
 |-- PctFallFromPeak: string (nullable = true)
 |-- LastTimeAtCurrZHVI: string (nullable = true)
```

21. Count of rows and columns

Lab 4

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [30]:  zillow_df.count()

30134

In [31]:  # Get the number of columns
          num_columns = len(zillow_df.columns)

          print("Number of columns:", num_columns)

Number of columns: 19
```

22. First few records of Zillow dataset

```
In [29]:  zillow_df.show(5)

+----------+--------+----------+-----+--------------------+-------------+--------+--------+-------+--------------------+----+----+-----+------+---------+-----------+--------+-------------------+---
---------------+
|      Date|RegionID|RegionName|State|               Metro|       County|    City|SizeRank|   Zhvi|                 MoM| QoQ| YoY|5Year|10Year|PeakMonth|PeakQuarter|PeakZHVI|    PctFallFromPeak|Las
tTimeAtCurrZHVI|
+----------+--------+----------+-----+--------------------+-------------+--------+--------+-------+--------------------+----+----+-----+------+---------+-----------+--------+-------------------+---
---------------+
|2020-03-31|   61639|     10025|   NY|New York-Newark-J...|New York County|New York|       0|1085142|-0.00033274248863...|null|null| null|  null|  2016-12|    2016-Q4| 1274615|  -0.148651161331069|
2014-04|
|2020-03-31|   84654|     60657|   IL|Chicago-Napervill...|    Cook County| Chicago|       1| 491246|0.000035953358747...|null|null| null|  null|  2018-12|    2018-Q4|  499880| -0.0172721453148756|
2017-12|
|2020-03-31|   61637|     10023|   NY|New York-Newark-J...|New York County|New York|       2|1221711|0.000002439410891...|null|null| null|  null|  2017-06|    2017-Q2| 1316646| -0.0721036633992736|
2015-05|
|2020-03-31|   91982|     77494|   TX|Houston-The Woodl...|  Harris County|    Katy|       3| 336286|0.000013166780894...|null|null| null|  null|  2015-10|    2015-Q4|  339339|-0.00899690280221254|
2015-06|
|2020-03-31|   84616|     60614|   IL|Chicago-Napervill...|    Cook County| Chicago|       4| 638027|0.000005449169548...|null|null| null|  null|  2018-12|    2018-Q4|  653091| -0.0230656983483159|
2017-03|
+----------+--------+----------+-----+--------------------+-------------+--------+--------+-------+--------------------+----+----+-----+------+---------+-----------+--------+-------------------+---
---------------+
only showing top 5 rows
```

```
In [32]: zillow_df.limit(10).toPandas().T
```

	0	1	2	3	4	5
Date	2020-03-31	2020-03-31	2020-03-31	2020-03-31	2020-03-31	2020-03-31
RegionID	61639	84654	61637	91982	84616	91940	61616
RegionName	10025	60657	10023	77494	60614	77449	10002
State	NY	IL	NY	TX	IL	TX	NY
Metro	New York-Newark-Jersey City	Chicago-Naperville-Elgin	New York-Newark-Jersey City	Houston-The Woodlands-Sugar Land	Chicago-Naperville-Elgin	Houston-The Woodlands-Sugar Land	New York-Newark-Jersey City
County	New York County	Cook County	New York County	Harris County	Cook County	Harris County	New York County
City	New York	Chicago	New York	Katy	Chicago	Katy	New York
SizeRank	0	1	2	3	4	5	6
Zhvi	1085142	491246	1221711	336286	638027	191404	1096075
MoM	-0.000332742488632599	0.0000359533587470651	0.0000024394108913838 9	0.0000131667808940013	0.0000054491695482283 8	0.0000066051791654899 7	-0.00001325668611874
QoQ	None	None	None	None	None	None	None
YoY	None	None	None	None	None	None	None
5Year	None	None	None	None	None	None	None
10Year	None	None	None	None	None	None	None
PeakMonth	2016-12	2018-12	2017-06	2015-10	2018-12	2020-03	2019-10
PeakQuarter	2016-Q4	2018-Q4	2017-Q2	2015-Q4	2018-Q4	2020-Q1	2019-Q4
PeakZHVI	1274615	499880	1316646	339339	653091	191404	1101521
PctFallFromPeak	-0.148651161331069	-0.0172721453148756	-0.0721036633992736	-0.00899690280221254	-0.0230656983483159	0	-0.00494407278662867
LastTimeAtCurrZHVI	2014-04	2017-12	2015-05	2015-06	2017-03	2020-03	2019-09

23. Descriptive statistics of the Zillow dataset

SizeRank:

- The SizeRank values range from 0 to 9,999.

ZHVI (Zillow Home Value Index):

- The average ZHVI is approximately 220,009.39.
- The ZHVI values range from 100,003 to 99,999.

MoM (Month-over-Month):

- The average MoM value is approximately 2.93e-5.
- The MoM values range from -0.00000000157936 to 0.000457225603361925.

PeakZHVI (Peak Zillow Home Value Index):

- The average PeakZHVI is approximately 229,228.33.
- The PeakZHVI values range from 100,003 to 99,995.

```
In [33]: zillow_df.describe("SizeRank","Zhvi","MoM","PeakZHVI","PctFallFromPeak").show()

+-------+-----------------+------------------+--------------------+-----------------+--------------------+
|summary|         SizeRank|              Zhvi|                 MoM|         PeakZHVI|     PctFallFromPeak|
+-------+-----------------+------------------+--------------------+-----------------+--------------------+
|  count|            30134|             30134|               30134|            30134|               30134|
|   mean|          15066.5| 220009.3946704719|2.927406339352759...|229228.32786885247|-0.03261201551657523|
| stddev|8699.080842249945|214881.17667935032|6.421417515461265E-5|226264.79354344108| 0.06616423005174453|
|    min|                0|            100003|-0.00000000157936...|           100003|-0.00000382656477...|
|    max|             9999|             99999|0.000457225603361925|            99995|                   0|
+-------+-----------------+------------------+--------------------+-----------------+--------------------+
```

24. Data Cleaning – Handling Null values

```
In [34]:
df3 = zillow_df.select([
    count(when(col(c).contains('None') |
               col(c).contains('NULL') |
               (col(c) == '') |
               col(c).isNull() |
               isnan(col(c)), c)
          ).alias(c)
    for c in zillow_df.columns
])

df3.show()
zillow_df = zillow_df.na.drop(subset=['Metro'])
zillow_df = zillow_df.na.drop(subset=['LastTimeAtCurrZHVI'])
```

```
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
|Date|RegionID|RegionName|State|Metro|County|City|SizeRank|Zhvi|MoM|  QoQ|  YoY|5Year|10Year|PeakMonth|PeakQuarter|PeakZHVI|PctFallFromPeak|LastTimeAtCurrZHVI|
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
|   0|       0|         0|    0| 7149|     0|   0|       0|   0|  0|30134|30134|30134| 30134|        0|          0|       0|              0|              1423|
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
```

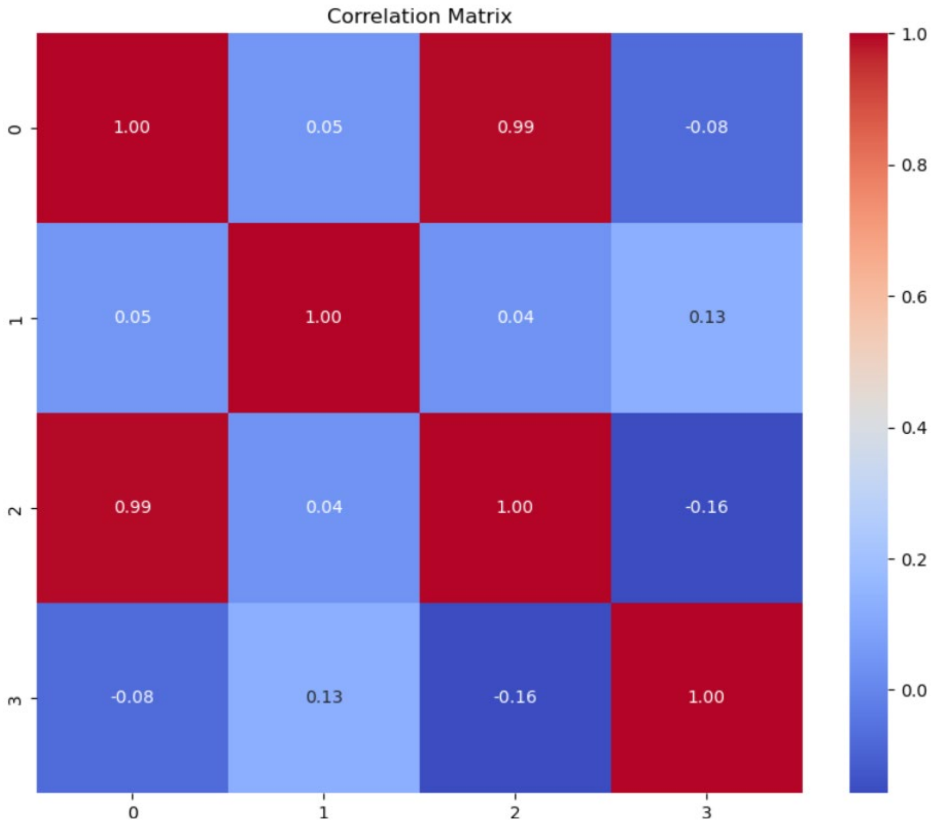
```
In [35]:
from pyspark.sql.functions import count, when, col, isnan

df3 = zillow_df.select([
    count(when(col(c).contains('None') |
               col(c).contains('NULL') |
               (col(c) == '') |
               col(c).isNull() |
               isnan(col(c)), c)
          ).alias(c)
    for c in zillow_df.columns
])

df3.show()
```

```
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
|Date|RegionID|RegionName|State|Metro|County|City|SizeRank|Zhvi|MoM|  QoQ|  YoY|5Year|10Year|PeakMonth|PeakQuarter|PeakZHVI|PctFallFromPeak|LastTimeAtCurrZHVI|
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
|   0|       0|         0|    0|    0|     0|   0|       0|   0|  0|22038|22038|22038| 22038|        0|          0|       0|              0|                 0|
+----+--------+----------+-----+-----+------+----+--------+----+---+-----+-----+-----+------+---------+-----------+--------+---------------+------------------+
```

25. Correlation matrix and plot

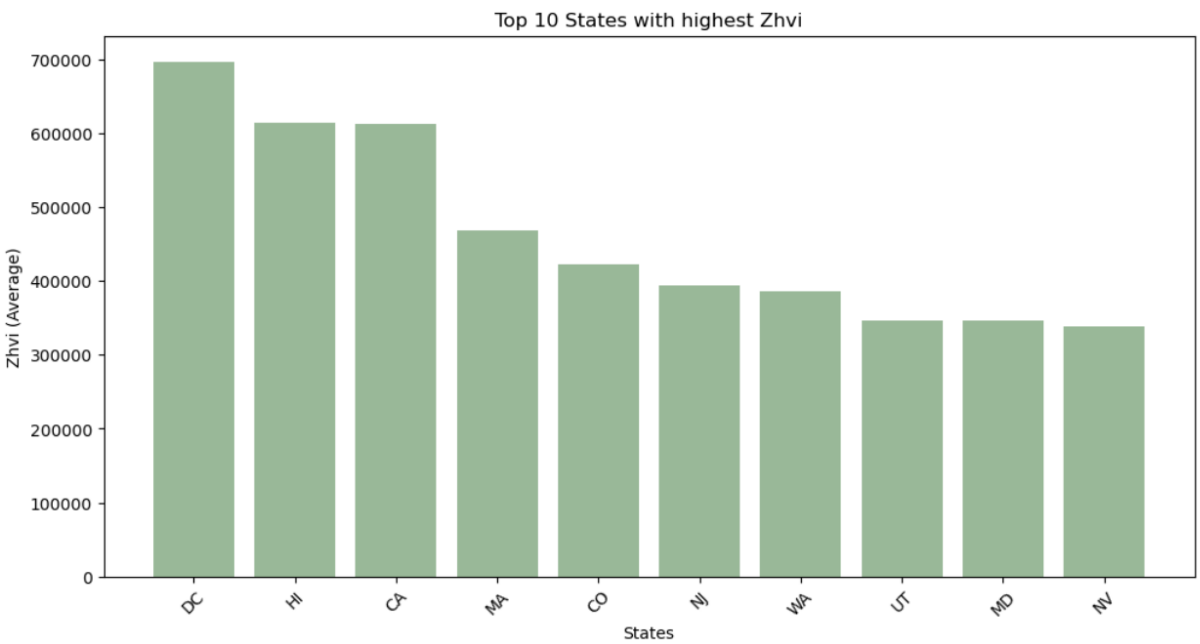
```
In [36]:
# select variables to check correlation
zillow_df_features = zillow_df.select('Zhvi','MoM','PeakZHVI','PctFallFromPeak')
zillow_df_features

# create RDD table for correlation calculation
rdd_table_2 = zillow_df_features.rdd.map(lambda row: row[0:])

# get the correlation matrix
corr_mat_2=Statistics.corr(rdd_table_2, method="pearson")
corr_mat_2
```

```
array([[ 1.        ,  0.04895401,  0.99467205, -0.07579185],
       [ 0.04895401,  1.        ,  0.03716116,  0.12973526],
       [ 0.99467205,  0.03716116,  1.        , -0.15757987],
       [-0.07579185,  0.12973526, -0.15757987,  1.        ]])
```

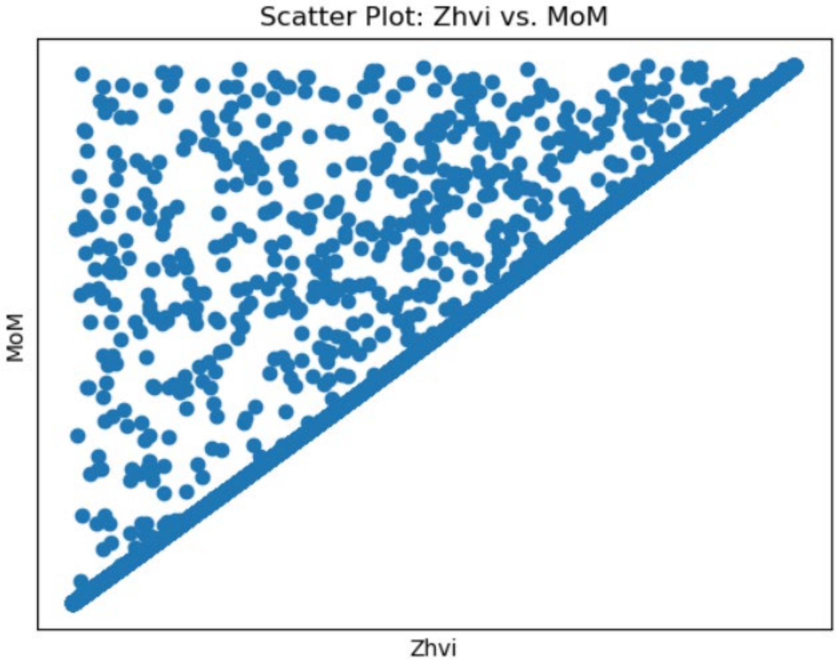


26. Data Visualizations

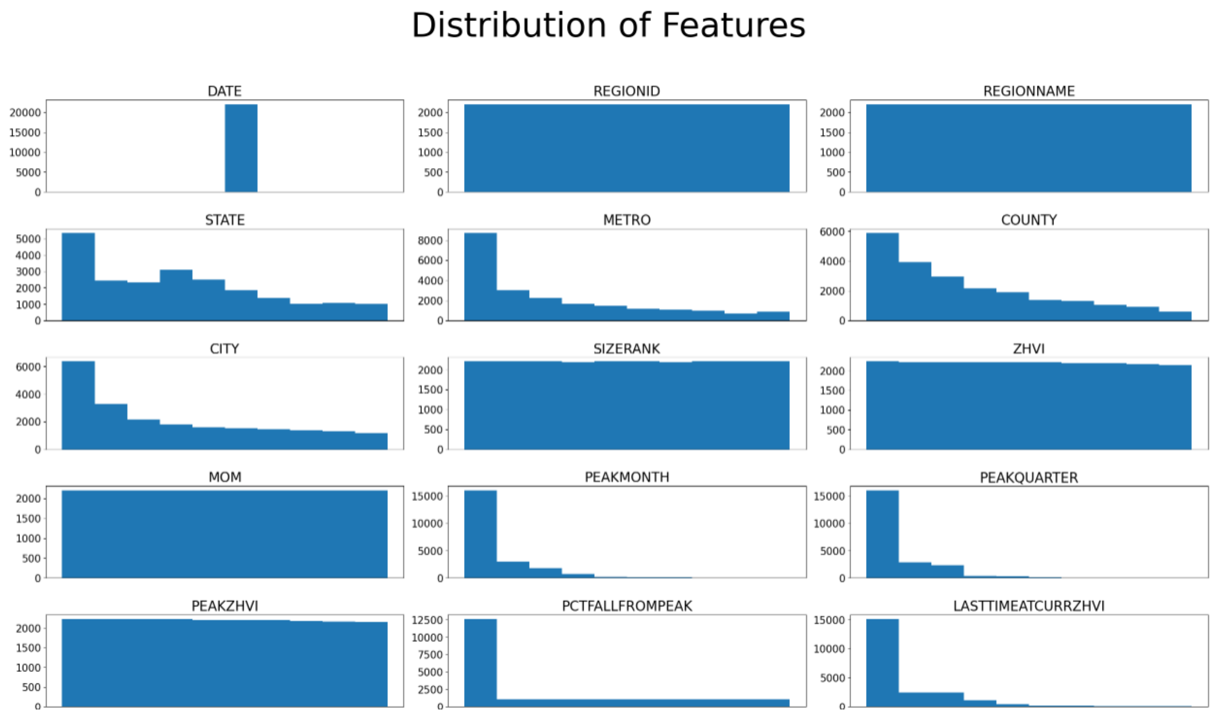
- **States with high ZHVI(Top-10)**



- **ZHVI vs MoM**



- **Histogram (All features)**



INSIGHTS

Dataset - 1

- Majority of people in the dataset (58.2%) have an income of less than $50,000.
- A significant finding is that the dataset contains a high percentage of individuals in three education categories: High School Graduate (HS-Grad), Bachelors, and Some College.
- The following variables are most correlated with income:
    1. Education is the most important factor in determining income level.
    2. Age is also a significant factor, but its impact on income level is nonlinear.
    3. Gender and race/ethnicity also play a role in determining income level, but their impact is smaller than that of education.

The above insights shed light on valuable information for understanding income disparities and making informed decisions regarding education, career choices, and policies related to income inequality.

Dataset - 2

- Homes in desirable locations in the District of Columbia (DC), Hawaii (HI), and California (CA) tend to experience larger increases in value than homes in less desirable locations.
- We can observe that all data points fall in the upper diagonal of the graph in the scatterplot between ZHVI and MoM. This indicates a positive correlation between the two, suggesting that higher-priced properties tend to have shorter periods on the market.
- The housing market and economic conditions play a role in determining home value in these states, but their impact is smaller than that of location.

REFERENCES

Census Income Data Set. (2019, December 18). Kaggle. *https://www.kaggle.com/datasets/vivamoto/us-adult-income-update*

Markumreed. (n.d.). *data_science_for_everyone/pyspark_examples/pima_indians_diabetes_eda_pyspark.ipynb at main · markumreed/data_science_for_everyone*. GitHub. https://github.com/markumreed/data_science_for_everyone/blob/main/pyspark_examples/pima_indians_diabetes_eda_pyspark.ipynb

Naveen. (2022). PySpark – Find Count of null, None, NaN Values. *Spark by {Examples}*. https://sparkbyexamples.com/pyspark/pyspark-find-count-of-null-none-nan-values/

APPENDIX

Loading the packages

```
import findspark
findspark.init()
findspark.find()

# Import Spark Libraries
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql.functions import count, when, col, isnan,desc
from pyspark.mllib.stat import Statistics

# Other Libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
# Starting Spark Connection
spark = SparkSession.builder.appName('Module4_Lab').getOrCreate()
spark
print(spark)
```

Importing Dataset 1

```
path = "C:/Users/14086/Downloads/dataset/census.csv"
# Importing CSV file
df = spark.read.csv(path, header=True)
```

Understanding the census dataset

```
# Displaying the column names
df.columns
# schema of df
df.printSchema()
# Overview of df
df.show()
# Checking the number of rows
df.count()
# Get the number of columns
num_columns = len(df.columns)

print("Number of columns:", num_columns)
# The first 5 rows of df
df.show(5)
df.limit(10).toPandas().T
```

```
df.describe("age","fnlwgt","education-num","capital-gain","capital-loss","hours-per-week").show()
```

Null / NA / "?" values

```
# Checking the Null / NA / "?" values
df1 = df.select([count(when(col(c).contains('None') | \
                            col(c).contains('NULL') | \
                            (col(c) == '' ) | \
                            col(c).isNull() | \
                            isnan(c), c
                           )).alias(c)
                 for c in df.columns])
df1.show()
df = df.dropna()
df2 = df.select([count(when(col(c).contains('?'), c
                           )).alias(c)
                 for c in df.columns])
df2.show()
```

Handling Duplicate Rows

```
df_distinct = df.distinct()

# Get the count of duplicate rows
num_duplicates = df.count() - df_distinct.count()

# Print the number of duplicate rows
print("Number of duplicate rows:", num_duplicates)

# Removing duplicate rows
df = df.dropDuplicates()
df.groupby('Income','sex').count().show()
df.groupby('Education').agg({'hours-per-week': 'mean'}).show()
```

Correlation Matrix

```
# select variables to check correlation
df_features = df.select("age", "education-num", "capital-gain", "capital-loss", "hours-per-week")
df_features

# create RDD table for correlation calculation
rdd_table = df_features.rdd.map(lambda row: row[0:])

# get the correlation matrix
corr_mat=Statistics.corr(rdd_table, method="pearson")
corr_mat
```

Plot

```
plt.figure(figsize=(10, 8))
sns.heatmap(corr_mat, annot=True, cmap='viridis', fmt=".2f")
plt.title('Correlation Plot')
```

```
plt.show()
Visualizations

race_counts = df.groupBy('race').count().orderBy('race')
race_pd = race_counts.toPandas()

# Calculate the percentage of each race
race_pd['percentage'] = race_pd['count'] / race_pd['count'].sum()

# pie chart
fig, ax = plt.subplots(figsize=(8, 8))
wedges, _, autotexts = ax.pie(
    race_pd['percentage'],
    labels=None,
    autopct='%.1f%%',
    startangle=90,
)
ax.set_title('Race')
ax.axis('equal')

ax.legend(wedges, race_pd['race'], loc='best', bbox_to_anchor=(1, 0.5), title='Race')

plt.show()
education_income_counts = df.groupBy('education', 'sex').count().orderBy('education')
education_income_pd = education_income_counts.toPandas()
pivoted_df = education_income_pd.pivot(index='education', columns='sex', values='count')

# bivariate bar graph
ax = pivoted_df.plot(kind='bar', stacked=True, figsize=(10, 6),color=['#6C5B7B', '#F67280'])
ax.set_xlabel('Education Level')
ax.set_ylabel('Count')
ax.set_title('Education Level by Gender')

plt.show()
fig = plt.figure(figsize=(25, 15))
st = fig.suptitle("Distribution of Features", fontsize=50, verticalalignment="center")
for col, num in zip(df_features.toPandas().describe().columns, range(1,6)):
  ax = fig.add_subplot(3,2, num)
  ax.hist(df.toPandas()[col])
  plt.grid(False)
  plt.yticks(fontsize=15)
  plt.title(col.upper(), fontsize=20)
  plt.xticks([])

plt.tight_layout()
st.set_y(0.95)
```

```
fig.subplots_adjust(top=0.85, hspace=0.4)
plt.show()
```

Importing dataset 2

```
path = "C:/Users/14086/Downloads/dataset/zillow.csv"
# Import CSV file
zillow_df = spark.read.csv(path, header=True)
```

Understanding the zillow dataset

```
type(zillow_df)
# Displaying the column names
zillow_df.columns
#schema of dataset 2
zillow_df.printSchema()
#overview of dataset 2
zillow_df.show()
zillow_df.show(5)
zillow_df.count()
# Get the number of columns
num_columns = len(zillow_df.columns)

print("Number of columns:", num_columns)
zillow_df.limit(10).toPandas().T
zillow_df.describe("SizeRank","Zhvi","MoM","PeakZHVI","PctFallFromPeak").show()
```

Checking the Null / NA / "?" values

```
df3 = zillow_df.select([
    count(when(col(c).contains('None') |
            col(c).contains('NULL') |
            (col(c) == '') |
            col(c).isNull() |
            isnan(col(c)), c)
        ).alias(c)
    for c in zillow_df.columns
])

df3.show()
zillow_df = zillow_df.na.drop(subset=['Metro'])
zillow_df = zillow_df.na.drop(subset=['LastTimeAtCurrZHVI'])
from pyspark.sql.functions import count, when, col, isnan

df3 = zillow_df.select([
    count(when(col(c).contains('None') |
            col(c).contains('NULL') |
            (col(c) == '') |
            col(c).isNull() |
            isnan(col(c)), c)
        ).alias(c)
    for c in zillow_df.columns
```

```
])

df3.show()
Correlation Matrix
# select variables to check correlation
zillow_df_features = zillow_df.select('Zhvi','MoM','PeakZHVI','PctFallFromPeak')
zillow_df_features

# create RDD table for correlation calculation
rdd_table_2 = zillow_df_features.rdd.map(lambda row: row[0:])

# get the correlation matrix
corr_mat_2=Statistics.corr(rdd_table_2, method="pearson")
corr_mat_2
plt.figure(figsize=(10, 8))
sns.heatmap(corr_mat_2, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
Visualizations
fig = plt.figure(figsize=(25, 15))
st = fig.suptitle("Distribution of Features", fontsize=50, verticalalignment="center")

# Drop rows with missing values from the selected columns
selected_columns =
['Date','RegionID','RegionName','State','Metro','County','City','SizeRank','Zhvi','MoM','Peak
Month','PeakQuarter','PeakZHVI','PctFallFromPeak',
 'LastTimeAtCurrZHVI']
zillow_df_selected = zillow_df.select(selected_columns).dropna()

for col, num in zip(zillow_df_selected.columns, range(1, 20)):
    ax = fig.add_subplot(5, 3, num)
    ax.hist(zillow_df_selected.select(col).toPandas()[col])
    plt.grid(False)
    plt.yticks(fontsize=15)
    plt.title(col.upper(), fontsize=20)
    plt.xticks([])

plt.tight_layout()
st.set_y(0.95)
fig.subplots_adjust(top=0.85, hspace=0.4)
plt.show()
top_10_states = (zillow_df.select('State', 'Zhvi')
                 .withColumn('Zhvi', col('Zhvi').cast('double'))
                 .groupBy('State').avg('Zhvi')
                 .orderBy(col('avg(Zhvi)').desc()).limit(10)
                 .toPandas())
```

```
# Create the bar plot using matplotlib
plt.figure(figsize=(12, 6))
plt.bar(top_10_states['State'], top_10_states['avg(Zhvi)'], color='#99B898')
plt.xlabel('States')
plt.ylabel('Zhvi (Average)')
plt.title('Top 10 States with highest Zhvi')
plt.xticks(rotation=45)

# Display the plot
plt.show()

selected_df = zillow_df.select("Zhvi", "MoM")
selected_df = selected_df.toPandas()
plt.scatter(selected_df['Zhvi'], selected_df['MoM'],color='#4d648d')
plt.xlabel('Zhvi')
plt.ylabel('MoM')
plt.title('Scatter Plot: Zhvi vs. MoM')
plt.xticks([])
plt.yticks([])
plt.show()
selected_df = zillow_df.select("Zhvi", "MoM")
selected_df = selected_df.toPandas()
plt.scatter(selected_df['Zhvi'], selected_df['MoM'],color='#4d648d')
plt.xlabel('Zhvi')
plt.ylabel('MoM')
plt.title('Scatter Plot: Zhvi vs. MoM')
plt.xticks([])
plt.yticks([])
plt.show()
```