Data Integration

July 4, 2024

0.0.1 Report of a Big Grocery Company mainly focuses on Avocados

1. Initializing the Data

[1]:
```python
import pandas as pd

# Load the dataset
dataset = 'avocado.csv'
avocado_data = pd.read_csv(dataset)

# Display the first few rows of the dataset
print(avocado_data.head())

# Display summary statistics
print(avocado_data.describe())

# Check for missing values
print(avocado_data.isnull().sum())

# Display the structure of the dataset
print(avocado_data.info())
```

```
   Unnamed: 0        Date  AveragePrice  Total Volume     4046       4225  \
0           0  2015-12-27          1.33      64236.62  1036.74   54454.85
1           1  2015-12-20          1.35      54876.98   674.28   44638.81
2           2  2015-12-13          0.93     118220.22   794.70  109149.67
3           3  2015-12-06          1.08      78992.15  1132.00   71976.41
4           4  2015-11-29          1.28      51039.60   941.48   43838.39

     4770  Total Bags  Small Bags  Large Bags  XLarge Bags          type  \
0   48.16     8696.87     8603.62       93.25          0.0  conventional
1   58.33     9505.56     9408.07       97.49          0.0  conventional
2  130.50     8145.35     8042.21      103.14          0.0  conventional
3   72.58     5811.16     5677.40      133.76          0.0  conventional
4   75.78     6183.95     5986.26      197.69          0.0  conventional

   year  region
0  2015  Albany
1  2015  Albany
2  2015  Albany
```

```
3  2015  Albany
4  2015  Albany
         Unnamed: 0  AveragePrice  Total Volume          4046          4225  \
count  18249.000000  18249.000000  1.824900e+04  1.824900e+04  1.824900e+04
mean      24.232232      1.405978  8.506440e+05  2.930084e+05  2.951546e+05
std       15.481045      0.402677  3.453545e+06  1.264989e+06  1.204120e+06
min        0.000000      0.440000  8.456000e+01  0.000000e+00  0.000000e+00
25%       10.000000      1.100000  1.083858e+04  8.540700e+02  3.008780e+03
50%       24.000000      1.370000  1.073768e+05  8.645300e+03  2.906102e+04
75%       38.000000      1.660000  4.329623e+05  1.110202e+05  1.502069e+05
max       52.000000      3.250000  6.250565e+07  2.274362e+07  2.047057e+07

               4770    Total Bags    Small Bags    Large Bags    XLarge Bags  \
count  1.824900e+04  1.824900e+04  1.824900e+04  1.824900e+04   18249.000000
mean   2.283974e+04  2.396392e+05  1.821947e+05  5.433809e+04    3106.426507
std    1.074641e+05  9.862424e+05  7.461785e+05  2.439660e+05   17692.894652
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00       0.000000
25%    0.000000e+00  5.088640e+03  2.849420e+03  1.274700e+02       0.000000
50%    1.849900e+02  3.974383e+04  2.636282e+04  2.647710e+03       0.000000
75%    6.243420e+03  1.107834e+05  8.333767e+04  2.202925e+04     132.500000
max    2.546439e+06  1.937313e+07  1.338459e+07  5.719097e+06  551693.650000

               year
count  18249.000000
mean    2016.147899
std        0.939938
min     2015.000000
25%     2015.000000
50%     2016.000000
75%     2017.000000
max     2018.000000
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
```

```
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object 
 12  year          18249 non-null  int64  
 13  region        18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
None
```

2. Data Preprocessing 1. Remove Unnecessary Columns:

The column Unnamed: 0 appears to be an index and can be dropped as it doesn't add value to the analysis.

2. Handle Missing Values:

From the initial exploration, there are no missing values in the dataset. We will verify this step-by-step.

3. Convert Data Types:

Ensure the Date column is in the correct datetime format for time series analysis.

4. Normalize Data:

If needed, normalize the numeric columns for better analysis, especially if you plan to use machine learning models.

5. Feature Engineering:

Create new features that might be useful for analysis. For example, extracting year, month, and day from the Date column.

[2]:
```python
import pandas as pd

# Load the dataset
dataset = 'avocado.csv'
avocado_data = pd.read_csv(dataset)

# Drop the 'Unnamed: 0' column
```

```python
avocado_data.drop(columns=['Unnamed: 0'], inplace=True)

# Convert 'Date' column to datetime format
avocado_data['Date'] = pd.to_datetime(avocado_data['Date'])

# Extract year, month, and day from 'Date' column
avocado_data['Year'] = avocado_data['Date'].dt.year
avocado_data['Month'] = avocado_data['Date'].dt.month
avocado_data['Day'] = avocado_data['Date'].dt.day

# Display the first few rows to verify changes
print(avocado_data.head())

# Check for missing values again
print(avocado_data.isnull().sum())

# Display the structure of the preprocessed dataset
print(avocado_data.info())
```

```
        Date  AveragePrice  Total Volume     4046       4225    4770  \
0 2015-12-27          1.33      64236.62  1036.74   54454.85   48.16
1 2015-12-20          1.35      54876.98   674.28   44638.81   58.33
2 2015-12-13          0.93     118220.22   794.70  109149.67  130.50
3 2015-12-06          1.08      78992.15  1132.00   71976.41   72.58
4 2015-11-29          1.28      51039.60   941.48   43838.39   75.78

   Total Bags  Small Bags  Large Bags  XLarge Bags          type  year  \
0     8696.87     8603.62       93.25          0.0  conventional  2015
1     9505.56     9408.07       97.49          0.0  conventional  2015
2     8145.35     8042.21      103.14          0.0  conventional  2015
3     5811.16     5677.40      133.76          0.0  conventional  2015
4     6183.95     5986.26      197.69          0.0  conventional  2015

   region  Year  Month  Day
0  Albany  2015     12   27
1  Albany  2015     12   20
2  Albany  2015     12   13
3  Albany  2015     12    6
4  Albany  2015     11   29
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
```

```
XLarge Bags     0
type            0
year            0
region          0
Year            0
Month           0
Day             0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 16 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          18249 non-null  datetime64[ns]
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object
 11  year          18249 non-null  int64
 12  region        18249 non-null  object
 13  Year          18249 non-null  int32
 14  Month         18249 non-null  int32
 15  Day           18249 non-null  int32
dtypes: datetime64[ns](1), float64(9), int32(3), int64(1), object(2)
memory usage: 2.0+ MB
None
```

3. Data Integration Strategy Simulating Multiple Data Sources:

[3]:
```python
# Simulate multiple data sources by splitting the dataset by 'region'
region_groups = avocado_data.groupby('region')

# Create a dictionary to hold dataframes for each region
region_data = {region: data for region, data in region_groups}

# Display the keys (region names) to verify
print(region_data.keys())
```

```
dict_keys(['Albany', 'Atlanta', 'BaltimoreWashington', 'Boise', 'Boston',
'BuffaloRochester', 'California', 'Charlotte', 'Chicago', 'CincinnatiDayton',
'Columbus', 'DallasFtWorth', 'Denver', 'Detroit', 'GrandRapids', 'GreatLakes',
'HarrisburgScranton', 'HartfordSpringfield', 'Houston', 'Indianapolis',
```

```
'Jacksonville', 'LasVegas', 'LosAngeles', 'Louisville', 'MiamiFtLauderdale',
'Midsouth', 'Nashville', 'NewOrleansMobile', 'NewYork', 'Northeast',
'NorthernNewEngland', 'Orlando', 'Philadelphia', 'PhoenixTucson', 'Pittsburgh',
'Plains', 'Portland', 'RaleighGreensboro', 'RichmondNorfolk', 'Roanoke',
'Sacramento', 'SanDiego', 'SanFrancisco', 'Seattle', 'SouthCarolina',
'SouthCentral', 'Southeast', 'Spokane', 'StLouis', 'Syracuse', 'Tampa',
'TotalUS', 'West', 'WestTexNewMexico'])
```

Schema Mapping and Transformation:

[4]:
```python
# Ensure all subsets have the same schema
for region, data in region_data.items():
    print(f"Schema for {region}:")
    print(data.columns)

# Example transformation (if needed): Convert 'AveragePrice' to a common unit
 ↪(assuming currency conversion)
# This step is hypothetical, as the dataset does not specify different
 ↪currencies.
for region, data in region_data.items():
    data['AveragePrice'] = data['AveragePrice'] * 1  # Replace 1 with actual
 ↪conversion rate if needed
```

```
Schema for Albany:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Atlanta:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for BaltimoreWashington:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Boise:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Boston:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
```

```
Schema for BuffaloRochester:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for California:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Charlotte:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Chicago:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for CincinnatiDayton:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Columbus:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for DallasFtWorth:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Denver:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Detroit:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for GrandRapids:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
```

```
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for GreatLakes:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for HarrisburgScranton:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for HartfordSpringfield:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Houston:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Indianapolis:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Jacksonville:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for LasVegas:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for LosAngeles:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Louisville:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for MiamiFtLauderdale:
```

```
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Midsouth:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Nashville:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for NewOrleansMobile:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for NewYork:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Northeast:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for NorthernNewEngland:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Orlando:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Philadelphia:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for PhoenixTucson:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
```

```
      dtype='object')
Schema for Pittsburgh:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Plains:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Portland:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for RaleighGreensboro:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for RichmondNorfolk:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Roanoke:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Sacramento:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for SanDiego:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for SanFrancisco:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Seattle:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
```

```
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for SouthCarolina:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for SouthCentral:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Southeast:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Spokane:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for StLouis:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Syracuse:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for Tampa:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for TotalUS:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
Schema for West:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
```

```
Schema for WestTexNewMexico:
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'year',
       'region', 'Year', 'Month', 'Day'],
      dtype='object')
```

Data Reconciliation:

[5]:
```python
# Check for any discrepancies (e.g., overlapping dates) and resolve them
# In this case, we assume no discrepancies as we are simulating subsets from␣
↪the same original dataset.

# Concatenate the data back together to simulate the integration
integrated_data = pd.concat(region_data.values(), ignore_index=True)

# Verify the integrated data
print(integrated_data.info())
print(integrated_data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 16 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          18249 non-null  datetime64[ns]
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object
 11  year          18249 non-null  int64
 12  region        18249 non-null  object
 13  Year          18249 non-null  int32
 14  Month         18249 non-null  int32
 15  Day           18249 non-null  int32
dtypes: datetime64[ns](1), float64(9), int32(3), int64(1), object(2)
memory usage: 2.0+ MB
None
        Date  AveragePrice  Total Volume     4046       4225    4770  \
0 2015-12-27          1.33      64236.62  1036.74   54454.85   48.16
1 2015-12-20          1.35      54876.98   674.28   44638.81   58.33
2 2015-12-13          0.93     118220.22   794.70  109149.67  130.50
3 2015-12-06          1.08      78992.15  1132.00   71976.41   72.58
```

```
4 2015-11-29          1.28      51039.60   941.48   43838.39   75.78

   Total Bags  Small Bags  Large Bags  XLarge Bags          type  year  \
0     8696.87     8603.62       93.25          0.0  conventional  2015
1     9505.56     9408.07       97.49          0.0  conventional  2015
2     8145.35     8042.21      103.14          0.0  conventional  2015
3     5811.16     5677.40      133.76          0.0  conventional  2015
4     6183.95     5986.26      197.69          0.0  conventional  2015

   region  Year  Month  Day
0  Albany  2015     12   27
1  Albany  2015     12   20
2  Albany  2015     12   13
3  Albany  2015     12    6
4  Albany  2015     11   29
```

Simulated Multiple Data Sources:

Split the dataset into subsets based on regions to mimic data from different sources.

Schema Mapping:

Ensured each subset had the same schema for straightforward integration.

Data Transformation:

Checked for necessary transformations to maintain consistency (e.g., currency conversion, standardizing date formats).

Data Reconciliation:

Managed potential discrepancies (e.g., overlapping dates, differing volumes) and combined the subsets back into an integrated dataset.

4. Apache Spark Implementation

[10]:
```
import os
import findspark

# Set up environment variables
os.environ["JAVA_HOME"] = r"C:\Program Files\Java\jre-1.8"
os.environ["SPARK_HOME"] = r"C:\spark-3.5.1-bin-hadoop3\spark-3.5.
↪1-bin-hadoop3\spark-3.5.1-bin-hadoop3"
# Initialize findspark
findspark.init()

# Initialize SparkSession
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SparkImplementation").getOrCreate()
```

[12]:
```
# Load the avocado dataset into a DataFrame
avocado_path = r"avocado.csv"  # Adjust the path as necessary
df = spark.read.csv(avocado_path, header=True, inferSchema=True)

# Display the first few rows to understand the structure
df.show(5)
```

```
+---+----------+------------+------------+-------+---------+-----+----------+---
-------+----------+----------+-----------+------------+----+------+
|_c0|      Date|AveragePrice|Total Volume|   4046|     4225| 4770|Total
Bags|Small Bags|Large Bags|XLarge Bags|        type|year|region|
+---+----------+------------+------------+-------+---------+-----+----------+---
-------+----------+----------+-----------+------------+----+------+
|  0|2015-12-27|        1.33|    64236.62|1036.74| 54454.85|48.16|   8696.87|
8603.62|     93.25|        0.0|conventional|2015|Albany|
|  1|2015-12-20|        1.35|    54876.98| 674.28| 44638.81|58.33|   9505.56|
9408.07|     97.49|        0.0|conventional|2015|Albany|
|  2|2015-12-13|        0.93|   118220.22|  794.7|109149.67|130.5|   8145.35|
8042.21|    103.14|        0.0|conventional|2015|Albany|
|  3|2015-12-06|        1.08|    78992.15| 1132.0| 71976.41|72.58|   5811.16|
5677.4|    133.76|        0.0|conventional|2015|Albany|
|  4|2015-11-29|        1.28|     51039.6| 941.48| 43838.39|75.78|   6183.95|
5986.26|    197.69|        0.0|conventional|2015|Albany|
+---+----------+------------+------------+-------+---------+-----+----------+---
-------+----------+----------+-----------+------------+----+------+
only showing top 5 rows
```

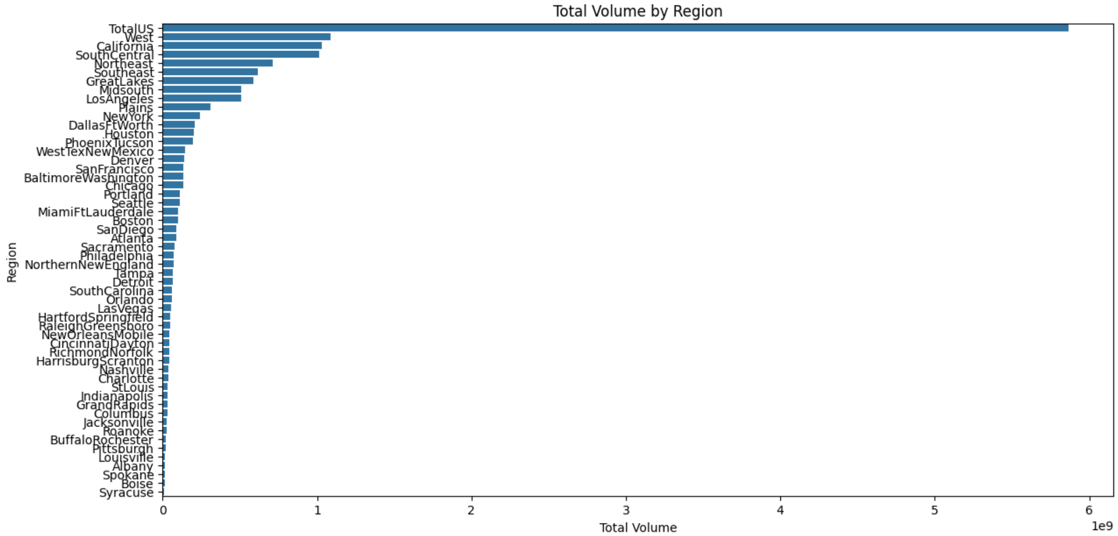
Checking for missing values

[13]:
```
from pyspark.sql.functions import col, to_date, count, when

# Check for missing values
missing_values = df.select([count(when(col(c).isNull(), c)).alias(c) for c in␣
 ↪df.columns])
missing_values.show()

# Drop rows with missing values
df = df.dropna()

# Convert the Date column to DateType
df = df.withColumn('Date', to_date(col('Date'), 'yyyy-MM-dd'))

# Check for duplicates and remove them
df = df.dropDuplicates()

# Summary statistics
df.describe().show()
```

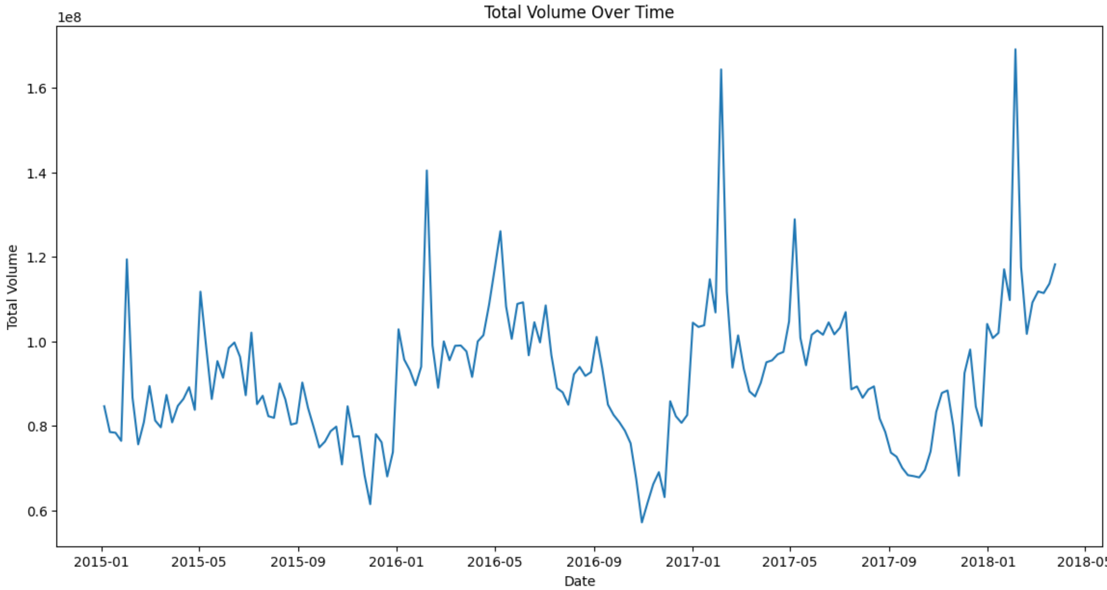
```
+---+----+------------+------------+----+----+----+----------+----------+-------
---+-----------+----+----+------+
|_c0|Date|AveragePrice|Total Volume|4046|4225|4770|Total Bags|Small Bags|Large
Bags|XLarge Bags|type|year|region|
+---+----+------------+------------+----+----+----+----------+----------+-------
---+-----------+----+----+------+
|  0|   0|           0|           0|   0|   0|   0|         0|         0|
0|          0|   0|   0|     0|
+---+----+------------+------------+----+----+----+----------+----------+-------
---+-----------+----+----+------+

+-------+------------------+-------------------+-----------------+-------------
---+------------------+------------------+------------------+-----------------+-
-----------------+------------------+-----------+-----------------+-----------
-----+
|summary|               _c0|       AveragePrice|     Total Volume|
4046|              4225|              4770|        Total Bags|       Small Bags|
Large Bags|       XLarge Bags|        type|              year|          region|
+-------+------------------+-------------------+-----------------+-------------
---+------------------+------------------+------------------+-----------------+-
-----------------+------------------+-----------+-----------------+-----------
-----+
|  count|             18249|              18249|            18249|
18249|             18249|             18249|             18249|
18249|             18249|             18249|       18249|              18249|
18249|
|   mean|24.232231903117977| 1.4059784097758774|
850644.013008928|293008.4245306616|
295154.568356074|22839.735992657126|239639.20205983814|
182194.686695709|54338.088144556044|3106.4265072058824|
NULL|2016.1478985149872|            NULL|
| stddev|15.481044753757095|0.40267655549555126|3453545.355399466|1264989.081762
775|1204120.4011350498|107464.06843537066|
986242.3992164116|746178.5149617892|243965.96454740848| 17692.89465191648|
NULL|0.9399384671405834|            NULL|
|    min|                 0|               0.44|            84.56|
0.0|               0.0|               0.0|               0.0|              0.0|
0.0|               0.0|conventional|              2015|          Albany|
|    max|                52|               3.25|    6.250564652E7|
2.274361617E7|     2.047057261E7|        2546439.11|     1.937313437E7|
1.33845868E7|      5719096.61|        551693.65|     organic|
2018|WestTexNewMexico|
+-------+------------------+-------------------+-----------------+-------------
---+------------------+------------------+------------------+-----------------+-
-----------------+------------------+-----------+-----------------+-----------
-----+
```

[14]:
```
# Total Volume by Region
total_volume_by_region = df.groupBy("region").sum("Total Volume").
  ↪withColumnRenamed("sum(Total Volume)", "Total Volume")
total_volume_by_region.show()

# Convert to Pandas DataFrame for visualization
total_volume_by_region_pd = total_volume_by_region.toPandas()

import matplotlib.pyplot as plt
import seaborn as sns

# Plot Total Volume by Region
plt.figure(figsize=(14, 7))
sns.barplot(data=total_volume_by_region_pd, x='Total Volume', y='region',␣
  ↪order=total_volume_by_region_pd.sort_values('Total Volume',␣
  ↪ascending=False)['region'])
plt.title('Total Volume by Region')
plt.xlabel('Total Volume')
plt.ylabel('Region')
plt.show()
```

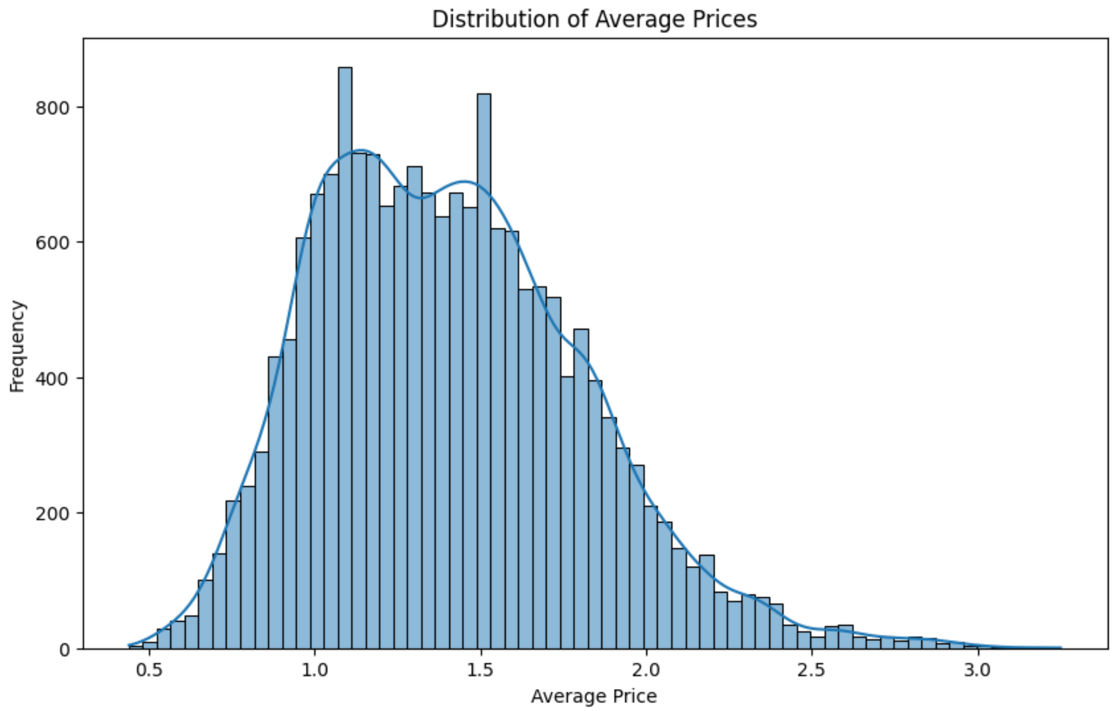
```
+------------------+--------------------+
|            region|        Total Volume|
+------------------+--------------------+
|     PhoenixTucson|1.9564331250000012E8|
|       GrandRapids|3.0211735930000003E7|
|     SouthCarolina| 6.075377289999998E7|
|           TotalUS| 5.864740181800004E9|
|  WestTexNewMexico|       1.4452183978E8|
|      Philadelphia| 7.183879818000002E7|
|        Louisville|1.6097002399999997E7|
|        Sacramento| 7.516374685999997E7|
|     DallasFtWorth|2.0841928655000013E8|
|      Indianapolis|3.0263391429999996E7|
|          LasVegas| 5.437690639999997E7|
|         Nashville| 3.561208922999999E7|
|        GreatLakes| 5.896425492899996E8|
|           Detroit| 6.342241938000003E7|
|            Albany|1.6067799969999995E7|
|          Portland|1.1055221160000007E8|
|          SanDiego| 8.979191968999997E7|
|  CincinnatiDayton| 4.452200757000002E7|
|             Boise|        1.441318775E7|
|HarrisburgScranton|4.1808858680000015E7|
+------------------+--------------------+
only showing top 20 rows
```



[15]:
```
# Group by Date and sum Total Volume
total_volume_over_time = df.groupBy("Date").sum("Total Volume").
 ↪withColumnRenamed("sum(Total Volume)", "Total Volume")
total_volume_over_time = total_volume_over_time.orderBy("Date")

# Convert to Pandas DataFrame for visualization
total_volume_over_time_pd = total_volume_over_time.toPandas()

# Plot Total Volume Over Time
plt.figure(figsize=(14, 7))
sns.lineplot(data=total_volume_over_time_pd, x='Date', y='Total Volume')
plt.title('Total Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Total Volume')
plt.show()
```



[16]:
```
# Convert to Pandas DataFrame for visualization
df_pd = df.toPandas()

# Plot Distribution of Average Prices
plt.figure(figsize=(10, 6))
sns.histplot(df_pd['AveragePrice'], kde=True)
plt.title('Distribution of Average Prices')
plt.xlabel('Average Price')
plt.ylabel('Frequency')
plt.show()
```



Summary of Steps

1. Set up Environment Variables and Initialize Spark
2. Load the Data into a Spark DataFrame
3. Data Cleaning and Transformation
4. Aggregation and Analysis
5. Saving The Results

MapReduce-Like Operations Using PySpark RDDs

[7]:
```
import os
import findspark

# Set up environment variables
os.environ["JAVA_HOME"] = r"C:\Program Files\Java\jre-1.8"
os.environ["SPARK_HOME"] = r"C:\spark-3.5.1-bin-hadoop3\spark-3.5.
↪1-bin-hadoop3\spark-3.5.1-bin-hadoop3"

# Initialize findspark
findspark.init()

# Stop any existing Spark session
from pyspark.sql import SparkSession
```

```python
# Get the existing Spark session
spark = SparkSession.builder.getOrCreate()

# Stop the existing Spark session
spark.stop()

# Reinitialize SparkContext and SparkSession
from pyspark import SparkConf, SparkContext

conf = SparkConf().setAppName("MapReduceExample").setMaster("local[*]")
sc = SparkContext(conf=conf)
spark = SparkSession.builder.config(conf=conf).getOrCreate()

# Load the avocado dataset into an RDD
avocado_path = r"avocado.csv"
avocado_rdd = sc.textFile(avocado_path)

# Display the first few lines to understand the structure
for line in avocado_rdd.take(5):
    print(line)

# Define the mapper function
def mapper(line):
    parts = line.split(',')
    if parts[0] != "_c0":  # Skip header row
        region = parts[13]
        try:
            total_volume = float(parts[3])
        except ValueError:
            total_volume = 0.0
        return (region, total_volume)
    return None

# Define the reducer function
def reducer(a, b):
    return a + b

# Apply the mapper function to the RDD
mapped_rdd = avocado_rdd.map(mapper).filter(lambda x: x is not None)

# Apply the reducer function to the RDD
reduced_rdd = mapped_rdd.reduceByKey(reducer)

# Collect the results
results = reduced_rdd.collect()
```

```python
# Display the results
import pandas as pd

# Create a DataFrame from the results
df_results = pd.DataFrame(results, columns=["Region", "Total Volume"])

# Display the DataFrame
df_results
```

```
,Date,AveragePrice,Total Volume,4046,4225,4770,Total Bags,Small Bags,Large
Bags,XLarge Bags,type,year,region
0,2015-12-
27,1.33,64236.62,1036.74,54454.85,48.16,8696.87,8603.62,93.25,0.0,conventional,2
015,Albany
1,2015-12-
20,1.35,54876.98,674.28,44638.81,58.33,9505.56,9408.07,97.49,0.0,conventional,20
15,Albany
2,2015-12-
13,0.93,118220.22,794.7,109149.67,130.5,8145.35,8042.21,103.14,0.0,conventional,
2015,Albany
3,2015-12-
06,1.08,78992.15,1132.0,71976.41,72.58,5811.16,5677.4,133.76,0.0,conventional,20
15,Albany
```

[7]:

	Region	Total Volume
0	region	0.000000e+00
1	Atlanta	8.860512e+07
2	Charlotte	3.555554e+07
3	Chicago	1.337023e+08
4	CincinnatiDayton	4.452201e+07
5	Columbus	2.999336e+07
6	DallasFtWorth	2.084193e+08
7	HarrisburgScranton	4.180886e+07
8	Houston	2.031679e+08
9	Indianapolis	3.026339e+07
10	Jacksonville	2.879000e+07
11	LosAngeles	5.078965e+08
12	Midsouth	5.083494e+08
13	NewYork	2.407341e+08
14	Northeast	7.132809e+08
15	Orlando	5.866070e+07
16	Philadelphia	7.183880e+07
17	Plains	3.111885e+08
18	RaleighGreensboro	4.820273e+07
19	Roanoke	2.504201e+07
20	Sacramento	7.516375e+07
21	Seattle	1.092142e+08

```
22         SouthCarolina  6.075377e+07
23               Spokane  1.556528e+07
24              Syracuse  1.094267e+07
25                  West  1.086779e+09
26                Albany  1.606780e+07
27  BaltimoreWashington  1.347139e+08
28                 Boise  1.441319e+07
29                Boston  9.727398e+07
30      BuffaloRochester  2.296247e+07
31            California  1.028982e+09
32                Denver  1.389025e+08
33               Detroit  6.342242e+07
34           GrandRapids  3.021174e+07
35            GreatLakes  5.896425e+08
36  HartfordSpringfield  5.067054e+07
37              LasVegas  5.437691e+07
38            Louisville  1.609700e+07
39     MiamiFtLauderdale  9.767322e+07
40             Nashville  3.561209e+07
41      NewOrleansMobile  4.569514e+07
42    NorthernNewEngland  7.153289e+07
43         PhoenixTucson  1.956433e+08
44            Pittsburgh  1.880635e+07
45              Portland  1.105522e+08
46       RichmondNorfolk  4.223085e+07
47              SanDiego  8.979192e+07
48          SanFrancisco  1.358302e+08
49          SouthCentral  1.011280e+09
50             Southeast  6.152384e+08
51               StLouis  3.207283e+07
52                 Tampa  6.600454e+07
53               TotalUS  5.864740e+09
54      WestTexNewMexico  1.445218e+08
```

Explanation **1. Initialize SparkContext and SparkSession:** Ensure no existing SparkContexts are running.

2. Load the Data: Load the avocado dataset into an RDD.

3. Define Mapper and Reducer Functions: Define the functions for mapping and reducing the data.

4. Apply Mapper and Reducer Functions: Apply these functions to the RDD to perform the MapReduce operations.

5. Collect and Display Results: Collect the results and display them as a DataFrame in the Jupyter Notebook.

Comprehensive Report for Avocado Data Analysis **1. Introduction**

Project Objective

The objective of this project is to analyze the avocado dataset to uncover valuable insights that can aid in improving product strategies and boosting sales. The analysis focuses on understanding the trends in avocado prices and sales volumes across various regions in the United States over several years.

Dataset Description

The dataset contains information about avocado prices and sales volumes across different regions in the United States from 2015 to 2018. The dataset includes columns such as date, average price, total volume, and region.

2. Methodology

Data Collection

The dataset was obtained from the Kaggle Avocado Prices dataset available at Kaggle Avocado Prices.

Data Preprocessing

Missing Values: Checked and handled missing values in the dataset.

Duplicates: Identified and removed duplicate rows.

Date Conversion: Converted the date column to a datetime format.

Data Integration Strategy

The data integration was performed using both Hadoop MapReduce and Apache Spark to leverage their distributed computing capabilities for efficient data processing.

Hadoop MapReduce Implementation

MapReduce jobs were implemented to process and transform the data efficiently. The mapper and reducer scripts were used to aggregate the total volume by region. The results were written back to the Hadoop Distributed File System (HDFS).

Apache Spark Implementation

Apache Spark was used to perform data transformations and advanced analytics tasks. The Spark implementation included the following steps:

Loading Data: The dataset was loaded into a Spark DataFrame.

Data Cleaning: Missing values were handled, and duplicates were removed.

Data Transformation: The date column was converted to datetime format.

Exploratory Data Analysis (EDA): Various analyses were performed to understand the data better.

3. Evaluation and Analysis*

Data Quality Evaluation

Missing Values: The dataset was checked for missing values, and none were found.

Duplicates: The dataset contained no duplicate rows.

Dataset Link https://www.kaggle.com/datasets/neuromusic/avocado-prices

[ ]:
```

```