Tutorial 3.2 - Pandas for CEO Data Management

Management Science - Mastering Business Data

Introduction

Welcome back to the CEO suite at Bean Counter!

Your Data Challenge as CEO

Now that you've mastered NumPy for numerical analysis, you face a new challenge: Bean Counter's data comes in spreadsheets, CSVs, and databases. You have:

- Sales reports from 50+ locations
- Product catalogs with thousands of items
- Customer data with demographics and preferences
- Supplier information across multiple regions

The Problem: This data has labels, categories, dates, and mixed types. NumPy arrays aren't enough, you need something more powerful!

Your Tool: Pandas - Think of it as Excel but better, capable of handling millions of rows.

In this tutorial, you'll learn to wrangle business data, turning spreadsheets into actionable insights.

> Tip
>
> Excel cannot have more than 1 million rows!

> How to Use This Tutorial
>
> First import pandas and numpy. Work through exercises marked "YOUR CODE BELOW".

Section 1 - DataFrames: Your CEO's Digital Spreadsheet

A DataFrame is like a better Excel spreadsheet in Python. Let's create one for Bean Counter's store performance.

```
import pandas as pd
import numpy as np

# Create a DataFrame from a dictionary
store_data = {
```

```
    'store_id': [101, 102, 103, 104, 105],
    'location': ['Downtown', 'Airport', 'University', 'Beach', 'Mall'],
    'monthly_sales': [450000, 620000, 380000, 290000, 510000],
    'staff_count': [45, 62, 38, 28, 52],
    'customer_rating': [4.8, 4.5, 4.9, 4.7, 4.6]
}

df = pd.DataFrame(store_data)
print("Bean Counter Store Performance:")
print(df)
print(f"\nDataFrame shape: {df.shape} (rows, columns)")
```

```
Bean Counter Store Performance:
   store_id    location  monthly_sales  staff_count  customer_rating
0       101    Downtown         450000           45              4.8
1       102     Airport         620000           62              4.5
2       103  University         380000           38              4.9
3       104       Beach         290000           28              4.7
4       105        Mall         510000           52              4.6

DataFrame shape: (5, 5) (rows, columns)
```

DataFrames vs NumPy Arrays

- NumPy arrays: Great for numerical calculations
- Pandas DataFrames: Perfect for labeled, mixed-type data
- Think of DataFrames as spreadsheets!

Exercise 1.1 - Create Product Catalog DataFrame

Build a DataFrame for Bean Counter's top products.

```
import pandas as pd

# YOUR CODE BELOW
# Create a dictionary with product data
product_data = {
    'product_name': ['Espresso', 'Latte', 'Cappuccino', 'Americano',
'Mocha'],
    'price': [2.50, 4.50, 4.00, 3.00, 5.00],
    'cost': [0.75, 1.50, 1.25, 0.90, 1.80],
    'units_sold_daily': [450, 320, 280, 210, 190]
}

# Create DataFrame
products_df =
```

```
# Test your DataFrame
assert products_df.shape == (5, 4), "Should have 5 products and 4 columns"
```

```
assert list(products_df.columns) == ['product_name', 'price', 'cost',
'units_sold_daily']
assert products_df['price'].sum() == 19.00, "Total price should be 19.00"
print("Bean Counter Product Catalog:")
print(products_df)
print("Perfect! Your product catalog is ready for analysis!")
```

Section 2 - Exploring Your Business Data

As CEO, you need to quickly understand your data. Pandas provides powerful exploration tools.

```
import pandas as pd
import numpy as np

# Create sample sales data
np.random.seed(42)
sales_data = pd.DataFrame({
    'date': pd.date_range('2024-01-01', periods=100),
    'store_id': np.random.choice([101, 102, 103, 104, 105], 100),
    'daily_revenue': np.random.uniform(8000, 15000, 100),
    'customers': np.random.randint(200, 500, 100),
    'avg_ticket': np.random.uniform(15, 35, 100)
})

# Key exploration methods
print("First 5 rows:")
print(sales_data.head())

print("\n\nData Info:")
print(sales_data.info())

print("\n\nStatistical Summary:")
print(sales_data.describe().round(2))
```

```
First 5 rows:
        date  store_id  daily_revenue  customers  avg_ticket
0 2024-01-01       104   14404.717729        466   29.178220
1 2024-01-02       105   13950.270045        350   26.056400
2 2024-01-03       103   11146.154719        497   20.930203
3 2024-01-04       105    8667.870815        298   23.395617
4 2024-01-05       105   10595.727765        462   20.124139


Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           100 non-null    datetime64[ns]
```

```
 1   store_id       100 non-null    int64
 2   daily_revenue  100 non-null    float64
 3   customers      100 non-null    int64
 4   avg_ticket     100 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(2)
memory usage: 4.0 KB
None


Statistical Summary:
                      date  store_id  daily_revenue  customers  avg_ticket
count                  100    100.00         100.00     100.00      100.00
mean   2024-02-19 12:00:00    103.07       11401.73     353.45       24.58
min    2024-01-01 00:00:00    101.00        8003.64     201.00       15.10
25%    2024-01-25 18:00:00    102.00        9788.71     298.00       18.84
50%    2024-02-19 12:00:00    103.00       11243.92     351.50       24.22
75%    2024-03-15 06:00:00    104.00       12869.13     426.75       29.37
max    2024-04-09 00:00:00    105.00       14984.18     498.00       34.64
std                    NaN      1.40        1899.23      85.48        6.08
```

Exercise 2.1 - Explore Store Performance

Use pandas exploration methods to understand store performance data.

```
import pandas as pd
import numpy as np

# Store performance data
performance_df = pd.DataFrame({
    'store': ['Plaza', 'Station', 'Park', 'Beach', 'Airport', 'Mall',
'Downtown', 'University'],
    'quarterly_revenue': [1250000, 980000, 870000, 650000, 1450000,
1100000, 1350000, 920000],
    'profit_margin': [32.5, 28.7, 30.1, 25.4, 35.2, 31.8, 33.9, 29.5],
    'customer_count': [45000, 38000, 34000, 28000, 52000, 41000, 48000,
36000],
    'satisfaction': [4.7, 4.5, 4.6, 4.8, 4.4, 4.6, 4.8, 4.9]
})

# YOUR CODE BELOW
# 1. Display the first 3 rows (print them)

# 2. Get the shape (rows, columns) and save it
data_shape =

# 3. Get summary statistics for numerical columns (print them)
```

```
# Test your exploration
assert data_shape == (8, 5), "Should have 8 stores and 5 columns"
print("Great exploration! You understand your data structure!")
```

Section 3 - Selecting and Filtering CEO Reports

As CEO, you need to slice and dice data to answer specific questions.

```
import pandas as pd

# Sample store data
df = pd.DataFrame({
    'store': ['Downtown', 'Airport', 'Beach', 'Mall', 'University'],
    'revenue': [450000, 620000, 290000, 510000, 380000],
    'profit': [135000, 155000, 65000, 140000, 95000],
    'rating': [4.8, 4.5, 4.7, 4.6, 4.9]
})

# Selecting columns
print("Revenue column:")
print(df['revenue'])

print("\n\nMultiple columns:")
print(df[['store', 'profit']])

# Filtering with conditions
print("\n\nHigh performers (revenue > 400k):")
high_performers = df[df['revenue'] > 400000]
print(high_performers)
```

```
Revenue column:
0    450000
1    620000
2    290000
3    510000
4    380000
Name: revenue, dtype: int64


Multiple columns:
        store  profit
0    Downtown  135000
1     Airport  155000
2       Beach   65000
3        Mall  140000
4  University   95000


High performers (revenue > 400k):
      store  revenue  profit  rating
0  Downtown   450000  135000     4.8
1   Airport   620000  155000     4.5
3      Mall   510000  140000     4.6
```

⚠ Single vs Double Brackets

- `df['column']` returns a Series (single column)
- `df[['column']]` returns a DataFrame (even with one column)
- `df[['col1', 'col2']]` selects multiple columns

Exercise 3.1 - Filter Strategic Locations

Identify stores meeting specific CEO criteria.

```
import pandas as pd

stores_df = pd.DataFrame({
    'location': ['Plaza', 'Station', 'Park', 'Beach', 'Airport', 'Mall',
'Downtown', 'University'],
    'monthly_revenue': [125000, 98000, 87000, 65000, 145000, 110000,
135000, 92000],
    'growth_rate': [5.2, -2.1, 3.8, -0.5, 8.7, 4.1, 6.3, 2.9],
    'staff': [25, 20, 18, 15, 32, 23, 28, 19]
})

# YOUR CODE BELOW
# 1. Select only location and monthly_revenue columns
revenue_report =

# 2. Filter stores with revenue > 100000
high_revenue_stores =

# 3. Filter stores with positive growth
growing_stores =
```

```
# Test your filtering
assert len(high_revenue_stores) == 5, "Should find 5 high revenue stores"
print("Excellent! You've identified your star performers!")
```

Section 4 - Creating Strategic Insights

As CEO, you need to create new metrics and sort data for decision-making.

```
import pandas as pd

# Product performance data
products = pd.DataFrame({
    'product': ['Espresso', 'Latte', 'Cappuccino', 'Americano', 'Mocha'],
    'revenue': [11250, 14400, 11200, 6300, 9500],
    'cost': [3375, 4800, 3500, 1890, 3420],
    'units_sold': [4500, 3200, 2800, 2100, 1900]
})
```

```
# Create new calculated columns
products['profit'] = products['revenue'] - products['cost']
products['margin_percent'] = (products['profit'] / products['revenue'] *
100).round(1)
products['price_per_unit'] = (products['revenue'] /
products['units_sold']).round(2)

# Sort by profit (descending)
products_sorted = products.sort_values('profit', ascending=False)

print("Product Profitability Analysis:")
print(products_sorted[['product', 'profit', 'margin_percent']])
```

```
Product Profitability Analysis:
      product  profit  margin_percent
1       Latte    9600            66.7
0    Espresso    7875            70.0
2  Cappuccino    7700            68.8
4       Mocha    6080            64.0
3   Americano    4410            70.0
```

Exercise 4.1 - CEO Performance Dashboard

Create a comprehensive performance analysis with calculated metrics.

```
import pandas as pd

# Store operational data
operations_df = pd.DataFrame({
    'store': ['Plaza', 'Airport', 'Beach', 'Mall', 'Downtown'],
    'revenue': [125000, 145000, 65000, 110000, 135000],
    'costs': [87500, 94250, 48750, 77000, 87750],
    'customers': [4500, 5200, 2800, 4100, 4800],
    'staff': [25, 32, 15, 23, 28]
})

# YOUR CODE BELOW
# 1. Calculate profit for each store
operations_df['profit'] =

# 2. Calculate profit margin percentage
operations_df['profit_margin'] =

# 3. Calculate revenue per customer
operations_df['revenue_per_customer'] =

# 4. Calculate customers per staff (efficiency)
operations_df['efficiency'] =

# 5. Sort by profit (highest to lowest)
top_performers =
```

```
# Test your dashboard
assert operations_df['profit'].sum() == 184750, "Total profit should be
184,750"
assert operations_df['profit_margin'].max() == 35.0, "Highest margin should
be 35%"
assert top_performers.iloc[0]['store'] == 'Airport', "Airport should be
most profitable"
print("CEO Dashboard - Top Performers by Profit:")
print(top_performers[['store', 'profit', 'profit_margin', 'efficiency']])
print("Great CEO dashboard! You have visibility of performance!")
```

Section 5 - Reading Real Business Data

As CEO, you'll work with data from various sources. Let's load real files!

```
import pandas as pd
import io

# Simulate reading a CSV file (in practice, you'd use
pd.read_csv('filename.csv'))
csv_data = """store_id,location,jan_sales,feb_sales,mar_sales
101,Downtown,125000,132000,118000
102,Airport,145000,152000,148000
103,Beach,82000,79000,85000
104,Mall,98000,102000,105000
105,University,76000,81000,79000"""

df = pd.read_csv(io.StringIO(csv_data))
print("Quarterly Sales Report (loaded from CSV):")
print(df)

# Calculate Q1 totals
df['q1_total'] = df['jan_sales'] + df['feb_sales'] + df['mar_sales']
print("\nWith Q1 Totals:")
print(df[['location', 'q1_total']])
```

```
Quarterly Sales Report (loaded from CSV):
   store_id    location  jan_sales  feb_sales  mar_sales
0       101    Downtown     125000     132000     118000
1       102     Airport     145000     152000     148000
2       103       Beach      82000      79000      85000
3       104        Mall      98000     102000     105000
4       105  University      76000      81000      79000

With Q1 Totals:
     location  q1_total
0    Downtown    375000
1     Airport    445000
2       Beach    246000
3        Mall    305000
4  University    236000
```

Reading Different File Types

```
# CSV files
df = pd.read_csv('sales_data.csv')

# Excel files
df = pd.read_excel('report.xlsx', sheet_name='Sales')

# Specify columns to read
df = pd.read_csv('data.csv', usecols=['date', 'revenue'])
```

Exercise 5.1 - Load and Analyze Sales Data

Process a CSV file containing Bean Counter's sales data.

```
import pandas as pd
import io

# Simulated CSV data (in practice, you'd read from a file)
csv_content = """product,category,price,units_sold,customer_rating
Espresso,Coffee,2.50,4500,4.7
Latte,Coffee,4.50,3200,4.8
Cappuccino,Coffee,4.00,2800,4.6
Croissant,Food,3.50,1200,4.5
Muffin,Food,2.75,1800,4.4
Sandwich,Food,6.50,900,4.7
Mocha,Coffee,5.00,1900,4.5
Americano,Coffee,3.00,2100,4.6"""

# YOUR CODE BELOW
# 1. Read the CSV data
sales_df = pd.read_csv(io.StringIO(csv_content))

# 2. Calculate revenue for each product
sales_df['revenue'] =

# 3. Filter for only Coffee products
coffee_df =

# 4. Sort coffee products by units_sold (highest first)
coffee_sorted =

# 5. Calculate total coffee revenue
total_coffee_revenue =
```

```
# Test your data loading and analysis
assert len(coffee_df) == 5, "Should have 5 coffee products"
assert coffee_sorted.iloc[0]['product'] == 'Espresso', "Espresso should be
top seller"
assert total_coffee_revenue == 65750.0, f"Coffee revenue should be 65,750"
```

```
print("Top Coffee Products by Volume:")
print(coffee_sorted[['product', 'units_sold', 'revenue']])
print(f"\nTotal Coffee Revenue: ${total_coffee_revenue:,.2f}")
print("Perfect! You can now load and analyze real business data!")
```

Conclusion

Congratulations! You've learned the first steps in Pandas for management!

You've learned:

- DataFrames - Creating and working with structured business data
- Exploration - Using head(), info(), describe() for quick insights
- Selection - Accessing specific columns and rows of data
- Filtering - Finding data that meets business criteria
- Calculated Columns - Creating new metrics from existing data
- Sorting - Ranking data by any metric
- File I/O - Loading data from CSV and Excel files

Your Bean Counter CEO data toolkit now includes:

- Ability to work with spreadsheet-like data in Python
- Tools to explore and understand large datasets quickly
- Skills to filter and find exactly the data you need
- Power to create custom metrics and KPIs
- Capability to process data from multiple file formats

Remember:

- DataFrames are like Excel spreadsheets, but better
- Use `df['column']` for single columns, `df[['col1','col2']]` for multiple
- Filter with boolean conditions: `df[df['revenue'] > 100000]`
- Create new columns with calculations: `df['profit'] = df['revenue'] - df['cost']`
- Sort with `sort_values()` to rank your data
- Load real data with `pd.read_csv()` and `pd.read_excel()`

What's Next: In the final session of the Python introduction, you'll combine NumPy and Pandas with visualization to create compelling charts and graphs that will wow the board of directors and drive strategic decisions!

Solutions

You will likely find solutions to most exercises online. However, I strongly encourage you to work on these exercises independently without searching explicitly for the exact answers to the exercises. Understanding someone else's solution is very different from developing your own. Use the lecture notes and try to solve the exercises on your own. This approach will significantly enhance your learning and problem-solving skills.

Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities. If you encounter difficulties, review the lecture materials, experiment with different approaches, and don't hesitate to ask for clarification during class discussions.

Bibliography