Module 3: Financial Data Analysis with Pandas

Lesson 1: Mastering Pandas for Financial Analysis

Welcome to the Power of Data! 🚀

Congratulations on completing Module 02! You now understand Python fundamentals. But here's where things get exciting - you're about to learn **Pandas**, the library that makes Python the #1 choice for financial analysis worldwide.

What you'll learn:

- Handle financial data like a pro (income statements, balance sheets, market data)
- Build comparable company analyses in minutes (not hours in Excel)
- Analyze time series data (stock prices, quarterly results)
- Calculate financial metrics automatically
- Import/export Excel files seamlessly
- Work with real market data from APIs

By the end of this module, you'll be able to:

- Replace complex Excel workbooks with clean Python code
- Analyze thousands of companies in seconds
- Build dynamic financial models that update automatically
- Create professional comp tables for PE Club presentations

What is Pandas? And Why Does Every Quant Use It?

Pandas = Python Data Analysis Library

Think of Pandas as **Excel on steroids**:

- **Excel**: Great for manual work, limited to ~1 million rows, prone to errors
- **Pandas**: Automates everything, handles billions of rows, reproducible and auditable

What makes Pandas perfect for finance:

1. **DataFrames** - Like Excel tables, but programmable
2. **Time Series** - Built-in understanding of dates, quarters, fiscal years
3. **Calculations** - Apply formulas to millions of rows instantly
4. **Grouping** - Analyze by sector, region, time period automatically
5. **Excel Integration** - Read and write Excel files effortlessly
6. **Speed** - Process years of data in seconds

Real-world PE/IB use cases:

- Screening thousands of potential targets
- Building comp tables from Bloomberg data

- Analyzing portfolio company performance
- Creating board presentation materials
- Backtesting investment strategies

Setup: Installing Pandas

Before you start, make sure:

1. ✅ Your virtual environment is activated (you should see `(venv)` in terminal)
2. ✅ You're in your `financial-modeling` folder
3. ✅ You completed Modules 01 and 02

Install Pandas and required libraries:

Open VS Code terminal (`Ctrl+``) and run:

```
# Install Pandas and related libraries
pip install pandas numpy openpyxl yfinance

# What each library does:
# - pandas: Data analysis and manipulation
# - numpy: Numerical computing (Pandas uses this underneath)
# - openpyxl: Read/write modern Excel files (.xlsx)
# - yfinance: Download real market data from Yahoo Finance
```

What to expect:

- Installation takes 30-60 seconds
- You'll see "Successfully installed..." messages
- No error messages = ready to go! ✅

Verify installation:

```
python -c "import pandas as pd; print(f'Pandas {pd.__version__} ready!')"
```

You should see: `Pandas 2.x.x ready!`

Your First DataFrame - Building a Comp Table

What is a DataFrame? A DataFrame is like an Excel table - rows and columns - but with superpowers. Let's build a comparable companies analysis to see it in action.

Create a new file: `Module_03_Data_Analysis/01_first_dataframe.py`

```
"""
Your First Pandas DataFrame - Comparable Companies Analysis

This is how PE analysts screen potential targets and build comp tables.
"""

import pandas as pd
import numpy as np

# Creating a DataFrame - Just like an Excel table!
# Each column is a list of values

companies_data = {
    'Company': ['Apple Inc.', 'Microsoft Corp.', 'Alphabet Inc.',
'Amazon.com'],
    'Ticker': ['AAPL', 'MSFT', 'GOOGL', 'AMZN'],
    'Revenue': [394, 211, 307, 514],  # in billions
    'EBITDA': [130, 107, 100, 66],     # in billions
    'Market_Cap': [3000, 2800, 1800, 1600]  # in billions
}

# Create the DataFrame
df = pd.DataFrame(companies_data)

# Display it
print("Tech Giants Comparable Companies:")
print("=" * 70)
print(df)
print()

# Basic DataFrame operations
print(f"Number of companies: {len(df)}")
print(f"Columns: {list(df.columns)}")
print()

# Calculate new columns (like Excel formulas!)
df['EBITDA_Margin_%'] = (df['EBITDA'] / df['Revenue']) * 100
df['EV/Revenue'] = df['Market_Cap'] / df['Revenue']
df['EV/EBITDA'] = df['Market_Cap'] / df['EBITDA']

# Display with calculated metrics
print("With Financial Metrics:")
print("=" * 70)
print(df)
print()

# Select specific columns (like choosing columns in Excel)
summary = df[['Company', 'Ticker', 'EV/EBITDA', 'EBITDA_Margin_%']]
print("Key Metrics Summary:")
print("=" * 70)
print(summary)
print()
```

```
# Statistics (Excel descriptive statistics, but automatic!)
print("Statistical Summary:")
print("=" * 70)
print(df[['Revenue', 'EBITDA', 'EV/EBITDA']].describe())
```

Run it! (Click the ▶ button or press `Ctrl+Alt+N`)

What you should see:

```
Tech Giants Comparable Companies:
======================================================================
         Company Ticker  Revenue  EBITDA  Market_Cap
0     Apple Inc.   AAPL      394     130        3000
1  Microsoft Corp.   MSFT      211     107        2800
2  Alphabet Inc.  GOOGL      307     100        1800
3    Amazon.com   AMZN      514      66        1600

Number of companies: 4
Columns: ['Company', 'Ticker', 'Revenue', 'EBITDA', 'Market_Cap']

With Financial Metrics:
======================================================================
         Company Ticker  Revenue  EBITDA  Market_Cap  EBITDA_Margin_%
EV/Revenue  EV/EBITDA
0     Apple Inc.   AAPL      394     130        3000            33.0
7.6      23.1
1  Microsoft Corp.   MSFT      211     107        2800            50.7
13.3      26.2
2  Alphabet Inc.  GOOGL      307     100        1800            32.6
5.9      18.0
3    Amazon.com   AMZN      514      66        1600            12.8
3.1      24.2

Key Metrics Summary:
======================================================================
         Company Ticker  EV/EBITDA  EBITDA_Margin_%
0     Apple Inc.   AAPL      23.1             33.0
1  Microsoft Corp.   MSFT      26.2             50.7
2  Alphabet Inc.  GOOGL      18.0             32.6
3    Amazon.com   AMZN      24.2             12.8

Statistical Summary:
======================================================================
          Revenue      EBITDA    EV/EBITDA
count     4.00000    4.000000     4.000000
mean    356.50000  100.750000    22.875000
std     131.45998   27.039258     3.467706
min     211.00000   66.000000    18.000000
25%     284.00000   90.250000    21.525000
50%     350.50000  103.500000    23.650000
75%     419.25000  114.250000    24.950000
max     514.00000  130.000000    26.200000
```

What just happened?

1. You created a comp table (like Excel, but in code!)
2. Calculated financial ratios instantly (EBITDA margin, multiples)
3. Got statistical analysis automatically (mean, median, quartiles)
4. **All reproducible** - run it again, same results every time!

Try this:

- Add another company (Meta, Tesla, Netflix)
- Calculate P/E ratio (you'll need Net Income)
- Sort by EV/EBITDA (hint: `df.sort_values('EV/EBITDA')`)

Essential DataFrame Operations for Finance

Selecting and Filtering Data - Finding Investment Targets

Create a new file: `02_selecting_filtering.py`

```
"""
Selecting and Filtering Data - PE Screening

Learn how to find companies that meet your investment criteria.
"""

import pandas as pd

# Larger dataset - potential PE targets
targets = pd.DataFrame({
    'Company': ['TechCo A', 'TechCo B', 'RetailCo', 'ManufactCo',
'ServiceCo'],
    'Sector': ['Technology', 'Technology', 'Retail', 'Manufacturing',
'Services'],
    'Revenue': [500, 1200, 800, 300, 450],
    'EBITDA': [100, 240, 120, 45, 90],
    'Debt': [200, 400, 300, 100, 150],
    'Cash': [50, 100, 40, 20, 30]
})

# Calculate metrics
targets['EBITDA_Margin'] = targets['EBITDA'] / targets['Revenue']
targets['Net_Debt'] = targets['Debt'] - targets['Cash']
targets['Net_Debt_to_EBITDA'] = targets['Net_Debt'] / targets['EBITDA']

print("All Potential Targets:")
print(targets)
print()

# SELECTING COLUMNS - like choosing which columns to see in Excel
```

```
print("Key Metrics Only:")
key_metrics = targets[['Company', 'Revenue', 'EBITDA_Margin',
'Net_Debt_to_EBITDA']]
print(key_metrics)
print()

# FILTERING ROWS - PE investment criteria
print("Filtering for Investment Criteria:")
print("=" * 70)

# Criterion 1: Revenue > $400M (scale threshold)
large_companies = targets[targets['Revenue'] > 400]
print(f"\nCompanies with Revenue > $400M: {len(large_companies)}")
print(large_companies[['Company', 'Revenue']])

# Criterion 2: EBITDA Margin > 18% (profitability threshold)
profitable = targets[targets['EBITDA_Margin'] > 0.18]
print(f"\nCompanies with EBITDA Margin > 18%: {len(profitable)}")
print(profitable[['Company', 'EBITDA_Margin']])

# Criterion 3: Net Debt/EBITDA < 3.0x (reasonable leverage)
reasonable_leverage = targets[targets['Net_Debt_to_EBITDA'] < 3.0]
print(f"\nCompanies with Net Debt/EBITDA < 3.0x:
{len(reasonable_leverage)}")
print(reasonable_leverage[['Company', 'Net_Debt_to_EBITDA']])

# MULTIPLE CRITERIA - The real screening!
print("\n" + "=" * 70)
print("QUALIFIED PE TARGETS (All criteria met):")
print("=" * 70)

qualified = targets[
    (targets['Revenue'] > 400) &
    (targets['EBITDA_Margin'] > 0.18) &
    (targets['Net_Debt_to_EBITDA'] < 3.0)
]

if len(qualified) > 0:
    print(qualified)
else:
    print("No companies meet all criteria. Adjust thresholds.")

print()
print(f"Total companies screened: {len(targets)}")
print(f"Qualified targets: {len(qualified)}")
print(f"Pass rate: {len(qualified)/len(targets):.1%}")
```

What you should see:

```
QUALIFIED PE TARGETS (All criteria met):
======================================================================
    Company      Sector  Revenue  EBITDA  ...  EBITDA_Margin  Net_Debt
```

```
Net_Debt_to_EBITDA
1  TechCo B  Technology     1200     240  ...           0.20       300
1.25

Total companies screened: 5
Qualified targets: 1
Pass rate: 20.0%
```

This is exactly how PE firms screen thousands of potential targets!

Try this:

- Add more companies to the dataset
- Change the screening criteria (tighter or looser)
- Add sector-specific criteria (e.g., only Technology)

Working with Financial Statements - Income Statement Analysis

Create a new file: `03_financial_statements.py`

```python
"""
Financial Statement Analysis with Pandas

Build and analyze multi-year income statements like a pro.
"""

import pandas as pd
import numpy as np

# Income Statement - Traditional format (line items as rows)
print("Building an Income Statement DataFrame:")
print("=" * 70)

income_statement = pd.DataFrame({
    '2021': [1000, 600, 400, 100, 300, 75, 225, 56, 169],
    '2022': [1150, 660, 490, 110, 380, 86, 294, 74, 221],
    '2023': [1320, 720, 600, 120, 480, 99, 381, 95, 286],
    '2024E': [1520, 820, 700, 135, 565, 114, 451, 113, 338],
    '2025E': [1750, 910, 840, 150, 690, 131, 559, 140, 419]
}, index=[
    'Revenue',
    'COGS',
    'Gross Profit',
    'SG&A',
    'EBITDA',
    'D&A',
    'EBIT',
    'Taxes (25%)',
    'Net Income'
])
```

```
print(income_statement)
print()

# Transpose for time series analysis (years as rows)
print("Transposed for Time Series Analysis:")
print("=" * 70)

is_time_series = income_statement.T  # T = transpose
print(is_time_series)
print()

# Calculate margins automatically
print("With Calculated Margins:")
print("=" * 70)

is_time_series['Gross_Margin_%'] = (
    is_time_series['Gross Profit'] / is_time_series['Revenue'] * 100
)
is_time_series['EBITDA_Margin_%'] = (
    is_time_series['EBITDA'] / is_time_series['Revenue'] * 100
)
is_time_series['Net_Margin_%'] = (
    is_time_series['Net Income'] / is_time_series['Revenue'] * 100
)

# Display key metrics
key_metrics = is_time_series[[
    'Revenue', 'EBITDA', 'Net Income',
    'Gross_Margin_%', 'EBITDA_Margin_%', 'Net_Margin_%'
]]
print(key_metrics)
print()

# Calculate growth rates (Year-over-Year)
print("Year-over-Year Growth Analysis:")
print("=" * 70)

is_time_series['Revenue_Growth_%'] =
is_time_series['Revenue'].pct_change() * 100
is_time_series['EBITDA_Growth_%'] = is_time_series['EBITDA'].pct_change()
* 100
is_time_series['NI_Growth_%'] = is_time_series['Net Income'].pct_change()
* 100

growth_analysis = is_time_series[[
    'Revenue', 'Revenue_Growth_%',
    'EBITDA', 'EBITDA_Growth_%',
    'Net Income', 'NI_Growth_%'
]]
print(growth_analysis)
print()

# Summary statistics
```

```
print("Summary Statistics (Historical + Projections):")
print("=" * 70)
print(is_time_series[['Revenue', 'EBITDA_Margin_%',
'Net_Margin_%']].describe())
```

What you should see:

```
With Calculated Margins:
======================================================================
        Revenue  EBITDA  Net Income  Gross_Margin_%  EBITDA_Margin_%
Net_Margin_%
2021       1000     300         169            40.0             30.0
16.9
2022       1150     380         221            42.6             33.0
19.2
2023       1320     480         286            45.5             36.4
21.7
2024E      1520     565         338            46.1             37.2
22.2
2025E      1750     690         419            48.0             39.4
23.9

Year-over-Year Growth Analysis:
======================================================================
        Revenue  Revenue_Growth_%  EBITDA  EBITDA_Growth_%  Net Income
NI_Growth_%
2021       1000               NaN     300              NaN         169
NaN
2022       1150              15.0     380             26.7         221
30.8
2023       1320              14.8     480             26.3         286
29.4
2024E      1520              15.2     565             17.7         338
18.2
2025E      1750              15.1     690             22.1         419
24.0
```

This is what PE analysts do every day! Building and analyzing historical + projected financials.

Try this:

- Add more years of projections
- Calculate CAGR (Compound Annual Growth Rate)
- Add sensitivity analysis (what if margins improve?)

Time Series Analysis - Quarterly Performance Tracking

Create a new file: `04_time_series.py`

```
"""
Time Series Analysis - Track Quarterly Performance

Portfolio companies report quarterly - learn to analyze trends over time.
"""

import pandas as pd
import numpy as np
from datetime import datetime

# Create quarterly date range
print("Creating Quarterly Revenue Data:")
print("=" * 70)

# Generate quarters from 2022 to 2024
dates = pd.date_range(start='2022-01-01', end='2024-12-31', freq='Q')

# Simulate quarterly revenue (with growth and seasonality)
np.random.seed(42)  # For reproducible results
base_revenue = 250
growth = 0.03  # 3% quarterly growth
seasonality = [1.0, 1.05, 0.95, 1.10]  # Q1, Q2, Q3, Q4 patterns

revenues = []
for i in range(len(dates)):
    seasonal_factor = seasonality[i % 4]
    revenue = base_revenue * (1 + growth) ** i * seasonal_factor
    revenues.append(revenue + np.random.uniform(-10, 10))

# Create DataFrame
quarterly_data = pd.DataFrame({
    'Revenue': revenues
}, index=dates)

print(quarterly_data)
print()

# Calculate growth metrics
print("Growth Metrics:")
print("=" * 70)

quarterly_data['QoQ_Growth_%'] = quarterly_data['Revenue'].pct_change() *
100
quarterly_data['YoY_Growth_%'] =
quarterly_data['Revenue'].pct_change(periods=4) * 100

# 4-quarter rolling sum (TTM = Trailing Twelve Months)
quarterly_data['TTM_Revenue'] =
quarterly_data['Revenue'].rolling(window=4).sum()

print(quarterly_data.tail(8))  # Show last 8 quarters
print()
```

```
# Quarterly statistics by year
print("Annual Summary (by Year):")
print("=" * 70)

annual_summary = quarterly_data.resample('Y').agg({
    'Revenue': ['sum', 'mean', 'std'],
    'QoQ_Growth_%': 'mean',
    'YoY_Growth_%': 'mean'
})

print(annual_summary)
print()

# Identify best and worst quarters
print("Performance Extremes:")
print("=" * 70)

best_quarter = quarterly_data['Revenue'].idxmax()
worst_quarter = quarterly_data['Revenue'].idxmin()

print(f"Best Quarter:  {best_quarter.strftime('%Y-Q%q')}: "
      f"${quarterly_data.loc[best_quarter, 'Revenue']:.2f}M")
print(f"Worst Quarter: {worst_quarter.strftime('%Y-Q%q')}: "
      f"${quarterly_data.loc[worst_quarter, 'Revenue']:.2f}M")
print()

# Average quarterly performance
avg_revenue = quarterly_data['Revenue'].mean()
avg_qoq = quarterly_data['QoQ_Growth_%'].mean()
avg_yoy = quarterly_data['YoY_Growth_%'].mean()

print(f"Average Quarterly Revenue:    ${avg_revenue:.2f}M")
print(f"Average QoQ Growth:           {avg_qoq:.2f}%")
print(f"Average YoY Growth:           {avg_yoy:.2f}%")
```

What you should see:

```
Growth Metrics:
======================================================================
            Revenue  QoQ_Growth_%  YoY_Growth_%  TTM_Revenue
2023-09-30   287.34          3.45         13.25      1118.92
2023-12-31   318.74         10.93         14.13      1152.62
2024-03-31   280.12        -12.12         12.85      1187.81
2024-06-30   293.91          4.92         12.23      1223.76
2024-09-30   301.45          2.57         11.54      1194.22
2024-12-31   337.98         12.12         13.45      1213.46

Performance Extremes:
======================================================================
Best Quarter:  2024-Q4: $337.98M
Worst Quarter: 2022-Q1: $245.83M
```

```
Average Quarterly Revenue:    $278.45M
Average QoQ Growth:           2.89%
Average YoY Growth:           11.24%
```

This is exactly how PE firms track portfolio companies! Quarterly performance, YoY comparisons, TTM metrics.

Loading Real Market Data - Yahoo Finance Integration

Create a new file: `05_market_data.py`

```
"""
Real Market Data with yfinance

Download actual stock prices, calculate returns, analyze volatility.
This is what quants do every day!
"""

import pandas as pd
import numpy as np
import yfinance as yf

print("Downloading Real Stock Data:")
print("=" * 70)

# Download stock data for a tech company
ticker = 'AAPL'
start_date = '2023-01-01'
end_date = '2024-12-31'

print(f"Fetching {ticker} data from {start_date} to {end_date}...\\n")

# Download the data
stock_data = yf.download(ticker, start=start_date, end=end_date,
progress=False)

print(f"Downloaded {len(stock_data)} trading days of data\\n")
print("Sample Data (first 5 days):")
print(stock_data.head())
print()

# Calculate returns
print("Calculating Returns and Volatility:")
print("=" * 70)

stock_data['Daily_Return_%'] = stock_data['Close'].pct_change() * 100
stock_data['Cumulative_Return_%'] = ((1 +
stock_data['Close'].pct_change()).cumprod() - 1) * 100

# Moving averages (technical analysis)
```

```
stock_data['MA_20'] = stock_data['Close'].rolling(window=20).mean()
stock_data['MA_50'] = stock_data['Close'].rolling(window=50).mean()

# Volatility calculation
annual_volatility = stock_data['Daily_Return_%'].std() * np.sqrt(252)
print(f"\\n{ticker} Annual Volatility: {annual_volatility:.2f}%")
print(f"Average Daily Return: {stock_data['Daily_Return_%'].mean():.3f}%")
print(f"Total Return (period):
{stock_data['Cumulative_Return_%'].iloc[-1]:.2f}%")
print()

# Best and worst trading days
print("Performance Extremes:")
print("=" * 70)

best_day = stock_data['Daily_Return_%'].idxmax()
worst_day = stock_data['Daily_Return_%'].idxmin()

print(f"Best Day:  {best_day.strftime('%Y-%m-%d')}: +
{stock_data.loc[best_day, 'Daily_Return_%']:.2f}%")
print(f"Worst Day: {worst_day.strftime('%Y-%m-%d')}:
{stock_data.loc[worst_day, 'Daily_Return_%']:.2f}%")
print()

# Monthly performance summary
print("Monthly Returns Summary:")
print("=" * 70)

monthly_returns = stock_data['Close'].resample('M').last().pct_change() *
100
print(monthly_returns.tail(12))
print()

# Recent performance
print("Recent Performance (Last 10 Days):")
print("=" * 70)
recent = stock_data[['Close', 'Daily_Return_%', 'MA_20',
'MA_50']].tail(10)
print(recent)
```

This is real market data! You just downloaded and analyzed actual stock prices.

Try this:

- Compare multiple tickers (MSFT, GOOGL, AMZN)
- Calculate Sharpe ratio (risk-adjusted returns)
- Find correlation between stocks

Comparable Company Analysis - Valuation

Create a new file: `06_comps_analysis.py`

```
"""
Comparable Company Analysis (Comps)

Build a professional comp table and value a target company.
This is PE/IB 101!
"""

import pandas as pd
import numpy as np

print("Building Comparable Company Analysis:")
print("=" * 70)

# Create comps table (realistic SaaS companies)
comps = pd.DataFrame({
    'Company': ['SaaS Co A', 'SaaS Co B', 'SaaS Co C', 'SaaS Co D', 'SaaS
Co E', 'Target Co'],
    'Revenue': [1500, 2200, 1800, 3000, 1200, 1600],
    'EBITDA': [300, 440, 360, 600, 240, 320],
    'Growth_%': [25, 30, 22, 28, 35, 27],
    'Market_Cap': [3000, 4800, 3600, 7200, 2800, np.nan],  # Target
unknown
    'Net_Debt': [200, 400, 300, 500, 150, 250]
})

# Calculate Enterprise Value
comps['Enterprise_Value'] = comps['Market_Cap'] + comps['Net_Debt']

# Calculate multiples
comps['EV/Revenue'] = comps['Enterprise_Value'] / comps['Revenue']
comps['EV/EBITDA'] = comps['Enterprise_Value'] / comps['EBITDA']
comps['EBITDA_Margin_%'] = (comps['EBITDA'] / comps['Revenue']) * 100

print("\\nFull Comp Table:")
print(comps)
print()

# Statistics (excluding target)
is_not_target = comps['Company'] != 'Target Co'
comp_stats = comps[is_not_target][['EV/Revenue', 'EV/EBITDA',
'EBITDA_Margin_%']].describe()

print("Comparable Company Statistics:")
print("=" * 70)
print(comp_stats)
print()

# Calculate valuation using multiples
print("Target Company Valuation:")
print("=" * 70)

# Get median multiples
```

```
median_ev_revenue = comps[is_not_target]['EV/Revenue'].median()
median_ev_ebitda = comps[is_not_target]['EV/EBITDA'].median()

# Get target metrics
target_revenue = comps[comps['Company'] == 'Target Co']
['Revenue'].values[0]
target_ebitda = comps[comps['Company'] == 'Target Co']['EBITDA'].values[0]
target_net_debt = comps[comps['Company'] == 'Target Co']
['Net_Debt'].values[0]

# Calculate valuations
ev_from_revenue = target_revenue * median_ev_revenue
ev_from_ebitda = target_ebitda * median_ev_ebitda
avg_ev = (ev_from_revenue + ev_from_ebitda) / 2

# Convert to equity value
equity_value_revenue = ev_from_revenue - target_net_debt
equity_value_ebitda = ev_from_ebitda - target_net_debt
avg_equity_value = avg_ev - target_net_debt

print(f"\\nMedian EV/Revenue Multiple: {median_ev_revenue:.2f}x")
print(f"Median EV/EBITDA Multiple:  {median_ev_ebitda:.2f}x")
print()
print(f"Target Revenue: ${target_revenue:,.0f}M")
print(f"Target EBITDA:  ${target_ebitda:,.0f}M")
print(f"Target Net Debt: ${target_net_debt:,.0f}M")
print()
print("Valuation Results:")
print("-" * 70)
print(f"EV using Revenue multiple:  ${ev_from_revenue:,.0f}M")
print(f"EV using EBITDA multiple:   ${ev_from_ebitda:,.0f}M")
print(f"Average EV:                 ${avg_ev:,.0f}M")
print()
print(f"Equity Value (Revenue):     ${equity_value_revenue:,.0f}M")
print(f"Equity Value (EBITDA):      ${equity_value_ebitda:,.0f}M")
print(f"Average Equity Value:       ${avg_equity_value:,.0f}M")
```

This is exactly how investment bankers value companies! Using comparable company multiples.

Try this:

- Add more companies to the comp set
- Calculate P/E ratios (need net income)
- Sort by different metrics

Excel Integration - Reading and Writing Files

Create a new file: `07_excel_integration.py`

```
"""
Excel Integration - Seamless Import/Export

Work with Excel files just like you would with Python DataFrames.
Perfect for collaborating with teams that use Excel.
"""

import pandas as pd
import numpy as np

print("Excel Integration with Pandas:")
print("=" * 70)

# CREATE SAMPLE DATA
# Income Statement
income_statement = pd.DataFrame({
    '2021': [1000, 600, 400, 100, 300],
    '2022': [1150, 660, 490, 110, 380],
    '2023': [1320, 720, 600, 120, 480],
    '2024E': [1520, 820, 700, 135, 565]
}, index=['Revenue', 'COGS', 'Gross Profit', 'OpEx', 'EBITDA'])

# Balance Sheet
balance_sheet = pd.DataFrame({
    '2021': [2000, 1200, 800],
    '2022': [2300, 1300, 1000],
    '2023': [2650, 1400, 1250],
    '2024E': [3000, 1500, 1500]
}, index=['Total Assets', 'Total Liabilities', 'Equity'])

# Cash Flow
cash_flow = pd.DataFrame({
    '2021': [300, -150, -50],
    '2022': [380, -180, -100],
    '2023': [480, -200, -150],
    '2024E': [565, -220, -200]
}, index=['Operating CF', 'Investing CF', 'Financing CF'])

print("\\nSample Data Created:")
print("- Income Statement")
print("- Balance Sheet")
print("- Cash Flow Statement")
print()

# WRITE TO EXCEL
output_file = 'financial_model_output.xlsx'

print(f"Writing to Excel file: {output_file}")

with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
    income_statement.to_excel(writer, sheet_name='Income Statement')
    balance_sheet.to_excel(writer, sheet_name='Balance Sheet')
    cash_flow.to_excel(writer, sheet_name='Cash Flow')
```

```
    # Add a summary sheet
    summary = pd.DataFrame({
        'Metric': ['Total Revenue (2021-2024E)', 'Avg EBITDA Margin',
'Total Assets 2024E'],
        'Value': [
            income_statement.loc['Revenue'].sum(),
            (income_statement.loc['EBITDA'] /
income_statement.loc['Revenue']).mean() * 100,
            balance_sheet.loc['Total Assets', '2024E']
        ]
    })
    summary.to_excel(writer, sheet_name='Summary', index=False)

print(f"✅ Successfully created {output_file} with 4 sheets!")
print()

# READ FROM EXCEL
print(f"Reading back from Excel file: {output_file}")

# Read specific sheet
df_income = pd.read_excel(output_file, sheet_name='Income Statement',
index_col=0)
print("\\nIncome Statement (from Excel):")
print(df_income)
print()

# Read all sheets into a dictionary
all_sheets = pd.read_excel(output_file, sheet_name=None, index_col=0)

print(f"All sheets loaded:")
for sheet_name in all_sheets.keys():
    print(f"  - {sheet_name}")
print()

print("✅ Excel integration complete!")
print()
print("Note: The file 'financial_model_output.xlsx' has been created")
print("You can open it in Excel to see the results!")
```

This bridges Python and Excel! Now you can:

- Import existing Excel models into Python
- Export Python analysis to Excel for presentations
- Collaborate with Excel users seamlessly

Practice Exercises

Now it's your turn! These exercises combine everything you've learned in Module 03.

Exercise 1: Historical Revenue Analysis

Build a complete revenue analysis model:

Create a DataFrame with 5 years of quarterly revenue data (20 quarters total) for a fictional SaaS company:

- Starting quarterly revenue: $100M
- Assume 5-8% quarterly growth rate (vary it realistically)
- Calculate quarter-over-quarter (QoQ) growth %
- Calculate year-over-year (YoY) growth %
- Calculate the overall revenue CAGR
- Identify the best and worst performing quarters
- Calculate average revenue by quarter (Q1, Q2, Q3, Q4) to see seasonality

Expected Output:

```
Quarter    Revenue    QoQ_Growth_%    YoY_Growth_%
2020-Q1    100.0            NaN             NaN
2020-Q2    105.0           5.00             NaN
2020-Q3    110.3           5.00             NaN
...
```

Exercise 2: Build a Complete Tech Comps Table

Create a professional comparable company analysis:

Build a DataFrame with 6 tech companies (5 comps + 1 target):

- Include: Company Name, Revenue ($M), EBITDA ($M), Market Cap ($M), Net Debt ($M), Growth %
- Calculate: Enterprise Value, EV/Revenue, EV/EBITDA, EBITDA Margin %
- Find median and mean multiples (excluding target)
- Value the target company using both EV/Revenue and EV/EBITDA multiples
- Calculate the implied equity value
- Create a summary statistics table

Bonus:

- Add a column showing each comp's premium/discount vs median
- Sort companies by EV/EBITDA multiple

Exercise 3: Stock Returns and Risk Analysis

Download and analyze real stock data:

Pick your favorite stock (AAPL, MSFT, GOOGL, etc.):

- Download 2 years of historical price data using yfinance
- Calculate daily returns (%)
- Calculate cumulative returns (%)
- Calculate annual volatility (daily std * $\sqrt{252}$)
- Calculate the Sharpe Ratio: (Mean Daily Return * 252) / (Daily Std * $\sqrt{252}$)

    - Assume risk-free rate = 0 for simplicity
- Find the 5 best and 5 worst trading days
- Calculate monthly returns
- Find the maximum drawdown (largest peak-to-trough decline)

Bonus:

- Download S&P 500 data (^GSPC) and compare performance
- Calculate correlation between your stock and the S&P 500

Exercise 4: Excel Model Integration

Build and export a complete financial model:

Create three financial statements for a fictional company (2020-2024):

Income Statement:

- Revenue (assume 15% annual growth)
- COGS (60% of revenue)
- Gross Profit
- Operating Expenses (20% of revenue)
- EBITDA

Balance Sheet:

- Total Assets
- Total Liabilities
- Total Equity (Assets - Liabilities)

Cash Flow Statement:

- Operating Cash Flow (80% of EBITDA)
- Investing Cash Flow (negative, assume capex = 10% of revenue)
- Financing Cash Flow (balancing item)

Tasks:

1. Create all three DataFrames
2. Calculate key metrics:
    - Gross Margin %
    - EBITDA Margin %
    - ROE % (Net Income / Equity)
3. Write everything to a single Excel file with separate sheets:
    - Income Statement
    - Balance Sheet
    - Cash Flow
    - Key Metrics
4. Read the file back and verify the data

Bonus: Format the Excel file with different sheets and nice headers

---

Solutions

Complete solutions are available in `solutions.py` in this directory.

Before looking at solutions, try each exercise yourself! That's how you learn.

Key concepts to practice:

- DataFrame creation and manipulation
- Financial calculations (growth rates, margins, multiples)
- Time series analysis (QoQ, YoY, CAGR)
- Statistical functions (mean, median, std)
- Excel integration (read/write)
- Real market data (yfinance)

Challenge: Can you combine all four exercises into one comprehensive financial analysis report?

Key Takeaways

✅ **DataFrames**: The foundation of financial analysis in Python ✅ **Time Series**: Built-in support for dates and periods ✅ **Calculations**: Easy to calculate ratios, growth rates, returns ✅ **Grouping**: Analyze by categories (sector, region, etc.) ✅ **Excel Integration**: Read and write Excel files ✅ **Real Data**: Access market data via APIs (yfinance, etc.)

Next Steps

Continue to: `Module_04_DCF_Modeling/01_DCF_Overview.md`

Estimated Time: 3-4 hours **Prerequisites**: Module 2 (Python Fundamentals)