Module 4: DCF (Discounted Cash Flow) Modeling

Welcome to the Heart of Investment Banking! 🎯

What is a DCF Model?

Imagine you're buying a rental property. You wouldn't just look at the asking price - you'd calculate how much rent you'll collect every year, subtract costs, and figure out what those future cash flows are worth **today**. That's exactly what a DCF model does for companies!

DCF = Discounted Cash Flow

It answers one question: *"What is this business worth based on the cash it will generate in the future?"*

This is **THE** valuation method used by:

- Investment bankers advising on M&A deals
- Private equity firms deciding what to pay for companies
- Hedge funds determining if a stock is undervalued
- Corporate finance teams making investment decisions

Why learn DCF in Python instead of Excel?

- ✅ **Faster**: Change one assumption, instantly see results
- ✅ **More powerful**: Run 1,000 scenarios in seconds
- ✅ **Professional**: This is what top-tier firms are moving towards
- ✅ **Transparent**: All calculations are clearly documented in code
- ✅ **Repeatable**: Build once, use for every deal

What You'll Build in This Module

By the end of this module, you'll create a **production-quality DCF model** that includes:

📊 The Complete Package:

1. **Historical Analysis** - Understanding the company's track record
2. **Revenue Projections** - Forecasting future sales with declining growth
3. **Operating Model** - Projecting EBITDA, D&A, EBIT
4. **Tax Calculations** - NOPAT (Net Operating Profit After Tax)
5. **Working Capital** - Modeling AR, Inventory, AP changes
6. **CapEx Forecasting** - Growth vs. Maintenance capital expenditures
7. **Free Cash Flow** - The cash available to all investors
8. **WACC Calculation** - Your discount rate (cost of capital)
9. **Terminal Value** - What the business is worth beyond your projection period
10. **Enterprise & Equity Value** - The final valuation
11. **Sensitivity Analysis** - How changes in assumptions affect value
12. **Excel Export** - Professional output for presentations

Real-world context: This is the EXACT model structure used in pitch books at Goldman Sachs, JP Morgan, and top PE firms!

The DCF Framework - Core Concepts

Let's build your understanding step by step. Don't worry if this seems complex - we'll break it down!

Key Concepts

Key Concepts

1. Free Cash Flow (FCF) - The Cash the Business Generates

Think of it this way: FCF is the cash left over after running the business and investing in its future.

```
Free Cash Flow = NOPAT + D&A − CapEx − Δ NWC

Where:
📌 NOPAT = Net Operating Profit After Tax = EBIT × (1 − Tax Rate)
📌 D&A = Depreciation & Amortization (non-cash, so we add it back)
📌 CapEx = Capital Expenditures (cash spent on equipment, buildings)
📌 Δ NWC = Change in Net Working Capital (cash tied up in operations)
```

Example:

- EBIT: $100M, Tax Rate: 25% → NOPAT = $75M
- Add back D&A: +$10M (didn't actually leave as cash)
- Less CapEx: -$15M (building new factory)
- Less Δ NWC: -$5M (more inventory needed)
- **FCF = $65M** ← This is what's available to debt and equity holders!

---

2. WACC (Weighted Average Cost of Capital) - Your Discount Rate

Translation: WACC is the "hurdle rate" - the minimum return investors expect.

Why discount? $100 today is worth MORE than $100 in 5 years because:

- You could invest it and earn returns
- There's risk the future cash might not materialize
- Inflation erodes purchasing power

```
WACC = (E/V × Re) + (D/V × Rd × (1−Tc))

Where:
📌 E = Market value of equity (market cap)
📌 D = Market value of debt
📌 V = Total value (E + D)
```

```
📌 Re = Cost of equity (what equity investors expect)
📌 Rd = Cost of debt (interest rate on debt)
📌 Tc = Corporate tax rate (debt is tax-deductible!)

Cost of Equity (using CAPM):
Re = Rf + β × (Rm - Rf)

📌 Rf = Risk-free rate (10-year Treasury: ~4%)
📌 β = Beta (stock volatility vs market: typical ~1.0-1.5)
📌 Rm - Rf = Equity risk premium (~6%)
```

Example:

- Market Cap (E): $5,000M
- Debt (D): $1,000M
- Total Value (V): $6,000M
- Cost of Equity (Re): 4% + 1.2 × 6% = 11.2%
- Cost of Debt (Rd): 5%
- Tax Rate: 25%

WACC = (5000/6000 × 11.2%) + (1000/6000 × 5% × 0.75) **WACC = 9.95%** ← This is your discount rate!

---

3. Terminal Value - The Value Beyond Your Projections

You can't forecast cash flows forever, so after Year 5 (or 10), you need to estimate what the business is worth then.

Two methods:

Method 1: Perpetuity Growth (assume company grows forever at modest rate)

```
Terminal Value = FCF(Year 6) / (WACC - g)

Where g = perpetual growth rate (usually 2-3%, never > GDP growth)
```

Example:

- FCF in Year 5: $100M
- Growth rate (g): 3%
- WACC: 10%
- FCF in Year 6 = $100M × 1.03 = $103M
- **Terminal Value = $103M / (10% - 3%) = $1,471M**

Method 2: Exit Multiple (assume you sell at a multiple)

```
Terminal Value = EBITDA(Year 5) × Exit Multiple
```

Example:

- EBITDA in Year 5: $150M
- Exit Multiple: 12.0x (based on comparable companies)
- **Terminal Value = $150M × 12 = $1,800M**

PE insight: Most DCF value (60-80%) comes from terminal value, so this assumption is CRITICAL!

---

4. Enterprise Value to Equity Value - The Bridge

Enterprise Value (EV) = Value to ALL investors (debt + equity) **Equity Value** = Value to SHAREHOLDERS only

```
Enterprise Value = PV(Projected FCFs) + PV(Terminal Value)

Then bridge to equity:
Equity Value = EV + Cash − Debt − Minority Interest − Preferred Stock
```

Example:

- PV of Years 1-5 FCFs: $350M
- PV of Terminal Value: $900M
- **Enterprise Value = $1,250M**
- Add: Cash = +$200M
- Less: Debt = -$500M
- Less: Minority Interest = -$50M
- **Equity Value = $900M**
- Shares Outstanding: 100M
- **Price per Share = $9.00**

Step-by-Step: Building Your First DCF

We'll build the model in **7 progressive files**, each teaching one concept:

File Structure:

1. `01_historical_analysis.py` - Analyzing the company's track record
2. `02_revenue_projections.py` - Forecasting future revenue
3. `03_operating_model.py` - Projecting EBITDA, D&A, EBIT
4. `04_working_capital.py` - Modeling NWC changes
5. `05_free_cash_flow.py` - Calculating FCF
6. `06_wacc_calculation.py` - Computing the discount rate
7. `07_valuation_complete.py` - Tying it all together + Terminal Value

Each file builds on the previous one, so you learn incrementally!

Let's Build: Step 1 - Historical Analysis

Create a new file: `01_historical_analysis.py`

```
"""
Step 1: Historical Financial Analysis

Before projecting the future, understand the past!
This is what analysts spend WEEKS doing in Excel.
"""

import pandas as pd
import numpy as np

print("STEP 1: HISTORICAL ANALYSIS")
print("=" * 70)

# Company: TechCo Inc. - Last 3 years of financials
historical = pd.DataFrame({
    'Year': [2022, 2023, 2024],
    'Revenue': [800, 920, 1050],
    'COGS': [480, 540, 610],
    'SG&A': [160, 180, 190],
    'EBITDA': [160, 200, 250],
    'D&A': [24, 28, 32],
    'CapEx': [32, 37, 42],
})

# Calculate margins and growth
historical['Gross_Profit'] = historical['Revenue'] - historical['COGS']
historical['Gross_Margin_%'] = (historical['Gross_Profit'] /
historical['Revenue']) * 100
historical['EBITDA_Margin_%'] = (historical['EBITDA'] /
historical['Revenue']) * 100

# Calculate growth rates (YoY)
historical['Revenue_Growth_%'] = historical['Revenue'].pct_change() * 100
historical['EBITDA_Growth_%'] = historical['EBITDA'].pct_change() * 100

# Calculate ratios
historical['CapEx_%_Revenue'] = (historical['CapEx'] /
historical['Revenue']) * 100
historical['D&A_%_Revenue'] = (historical['D&A'] / historical['Revenue'])
* 100

print("\\nHistorical Financials ($M):")
print(historical)

# Summary statistics
print("\\n" + "-" * 70)
print("KEY INSIGHTS:")
print("-" * 70)
```

```
print(f"Average Revenue Growth:
{historical['Revenue_Growth_%'].mean():.1f}%")
print(f"Average EBITDA Margin:
{historical['EBITDA_Margin_%'].mean():.1f}%")
print(f"Average Gross Margin:
{historical['Gross_Margin_%'].mean():.1f}%")
print(f"Average CapEx % Revenue:
{historical['CapEx_%_Revenue'].mean():.1f}%")

print("\\n✅ Historical analysis complete!")
print("💡 These trends inform our future assumptions")
```

Run this! It shows you how to analyze a company's track record.

Try this:

- Look up a real company's financials (Apple, Microsoft)
- Calculate their historical margins
- See if margins are improving or declining

Let's Build: Step 2 - Revenue Projections

Create a new file: `02_revenue_projections.py`

```
"""
Step 2: Revenue Projections

Project future revenue using declining growth rates.
This is realistic - high growth slows over time!
"""

import pandas as pd

print("STEP 2: REVENUE PROJECTIONS")
print("=" * 70)

# Last year actual revenue (from historical analysis)
base_revenue = 1050  # 2024 actual

# Projection assumptions (declining growth - realistic!)
# Year 1: 18%, Year 2: 15%, Year 3: 12%, Year 4: 10%, Year 5: 8%
growth_rates = [0.18, 0.15, 0.12, 0.10, 0.08]
years = list(range(2025, 2030))

# Build revenue projections
revenues = []
current_revenue = base_revenue

for growth_rate in growth_rates:
    current_revenue = current_revenue * (1 + growth_rate)
```

```
    revenues.append(current_revenue)

# Create DataFrame
projections = pd.DataFrame({
    'Year': years,
    'Revenue': revenues,
    'Growth_Rate_%': [g * 100 for g in growth_rates]
})

print("\\nRevenue Projections ($M):")
print(projections)

# Calculate 5-year CAGR
cagr = ((revenues[-1] / base_revenue) ** (1/5) - 1) * 100

print("\\n" + "-" * 70)
print(f"2024 Base Revenue:  ${base_revenue:,.0f}M")
print(f"2029 Projected:     ${revenues[-1]:,.0f}M")
print(f"5-Year CAGR:        {cagr:.1f}%")
print("-" * 70)

print("\\n✅ Revenue projections complete!")
print("💡 Declining growth rates are more realistic than constant growth")
```

This is critical! Revenue drives everything in the model.

Let's Build: Step 3 - Operating Model (EBITDA)

Create a new file: `03_operating_model.py`

```
"""
Step 3: Operating Model - Projecting EBITDA, D&A, EBIT

This is the profitability engine of your model!
"""

import pandas as pd

print("STEP 3: OPERATING MODEL")
print("=" * 70)

# Revenue from Step 2
years = list(range(2025, 2030))
revenues = [1239, 1425, 1596, 1756, 1896]

# Operating assumptions (based on historical analysis)
ebitda_margin = 0.238  # 23.8% (stable assumption)
da_pct_revenue = 0.030  # 3.0% of revenue
tax_rate = 0.25  # 25% corporate tax rate
```

```
# Build operating model
operating = pd.DataFrame({
    'Year': years,
    'Revenue': revenues
})

# Calculate EBITDA
operating['EBITDA'] = operating['Revenue'] * ebitda_margin
operating['EBITDA_Margin_%'] = ebitda_margin * 100

# Calculate D&A
operating['D&A'] = operating['Revenue'] * da_pct_revenue

# Calculate EBIT (Earnings Before Interest & Tax)
operating['EBIT'] = operating['EBITDA'] - operating['D&A']
operating['EBIT_Margin_%'] = (operating['EBIT'] / operating['Revenue']) *
100

# Calculate Taxes (on EBIT, not EBITDA!)
operating['Taxes'] = operating['EBIT'] * tax_rate

# Calculate NOPAT (Net Operating Profit After Tax)
operating['NOPAT'] = operating['EBIT'] - operating['Taxes']

print("\\nOperating Model ($M):")
print(operating.round(1))

print("\\n" + "-" * 70)
print("KEY METRICS:")
print("-" * 70)
print(f"EBITDA Margin:  {ebitda_margin * 100:.1f}%")
print(f"EBIT Margin:    {operating['EBIT_Margin_%'].mean():.1f}%")
print(f"Tax Rate:       {tax_rate * 100:.1f}%")
print(f"D&A % Revenue:  {da_pct_revenue * 100:.1f}%")
print("-" * 70)

print("\\n✅ Operating model complete!")
print("💡 NOPAT is the profit available to all investors (debt + equity)")
```

This is where the magic happens! From revenue to profit.

Let's Build: Step 4 - Working Capital

Create a new file: `04_working_capital.py`

```
"""
Step 4: Working Capital Changes

Working capital ties up cash! Model how much.
NWC = (Accounts Receivable + Inventory) - (Accounts Payable)
```

```
"""

import pandas as pd

print("STEP 4: WORKING CAPITAL")
print("=" * 70)

# Revenue projections
years = list(range(2025, 2030))
revenues = [1239, 1425, 1596, 1756, 1896]

# Working capital assumption
nwc_pct_revenue = 0.10  # 10% of revenue (industry dependent!)

# 2024 base year NWC
base_revenue_2024 = 1050
base_nwc = base_revenue_2024 * nwc_pct_revenue

print(f"2024 Base NWC: ${base_nwc:.1f}M ({nwc_pct_revenue*100:.0f}% of
revenue)\\n")

# Calculate NWC for each projection year
nwc_df = pd.DataFrame({
    'Year': years,
    'Revenue': revenues
})

nwc_df['NWC'] = nwc_df['Revenue'] * nwc_pct_revenue
nwc_df['Change_in_NWC'] = nwc_df['NWC'].diff()

# First year change is vs. 2024
nwc_df.loc[0, 'Change_in_NWC'] = nwc_df.loc[0, 'NWC'] - base_nwc

print("Working Capital Analysis ($M):")
print(nwc_df.round(1))

print("\\n" + "-" * 70)
print("INTERPRETATION:")
print("-" * 70)
print("Positive Change in NWC = Cash OUTFLOW (bad for FCF)")
print("  → More cash tied up in receivables, inventory")
print("Negative Change in NWC = Cash INFLOW (good for FCF)")
print("  → Less cash tied up, or better payment terms")
print("-" * 70)

print(f"\\nTotal NWC Investment (5 years):
${nwc_df['Change_in_NWC'].sum():.1f}M")

print("\\n✅ Working capital model complete!")
print("💡 Growing companies need MORE working capital (cash outflow)")
```

This is often overlooked! Growth requires working capital investment.

Let's Build: Step 5 - Free Cash Flow

Create a new file: `05_free_cash_flow.py`

```
"""
Step 5: Free Cash Flow Calculation

THIS IS IT! The cash available to all investors.
FCF = NOPAT + D&A - CapEx - Change in NWC
"""

import pandas as pd

print("STEP 5: FREE CASH FLOW")
print("=" * 70)

# Build complete projection from previous steps
years = list(range(2025, 2030))

fcf_model = pd.DataFrame({
    'Year': years,
    'Revenue': [1239, 1425, 1596, 1756, 1896],
    'EBITDA': [295, 339, 380, 418, 451],
    'D&A': [37, 43, 48, 53, 57],
    'EBIT': [258, 296, 332, 365, 394],
    'Taxes': [64, 74, 83, 91, 98],
    'NOPAT': [193, 222, 249, 274, 295]
})

# Add back D&A (non-cash expense)
fcf_model['Add_D&A'] = fcf_model['D&A']

# CapEx (4% of revenue assumption)
capex_pct = 0.04
fcf_model['Less_CapEx'] = fcf_model['Revenue'] * capex_pct

# Change in NWC (from Step 4)
fcf_model['Less_Change_NWC'] = [18.9, 18.6, 17.1, 16.0, 14.0]

# CALCULATE FREE CASH FLOW
fcf_model['FREE_CASH_FLOW'] = (
    fcf_model['NOPAT'] +
    fcf_model['Add_D&A'] -
    fcf_model['Less_CapEx'] -
    fcf_model['Less_Change_NWC']
)

print("\\nFree Cash Flow Build ($M):")
print(fcf_model.round(1))

# Summary
print("\\n" + "=" * 70)
```

```
print("FREE CASH FLOW SUMMARY:")
print("=" * 70)
for i, year in enumerate(years):
    print(f"Year {i+1} ({year}): ${fcf_model.loc[i,
'FREE_CASH_FLOW']:.1f}M")

print(f"\\nTotal 5-Year FCF: ${fcf_model['FREE_CASH_FLOW'].sum():.1f}M")
print("=" * 70)

print("\\n✅ Free cash flow calculation complete!")
print("💡 FCF is THE number that drives valuation!")
```

This is the heart of DCF! Free cash flow is what we discount.

Let's Build: Step 6 - WACC Calculation

Create a new file: `06_wacc_calculation.py`

```
"""
Step 6: WACC (Weighted Average Cost of Capital)

Your discount rate - the return investors expect.
This is what we use to bring future cash flows to present value!
"""

import pandas as pd

print("STEP 6: WACC CALCULATION")
print("=" * 70)

# CAPM Inputs (Cost of Equity)
risk_free_rate = 0.040  # 10-year Treasury: 4.0%
beta = 1.2              # Stock volatility vs market
equity_risk_premium = 0.060  # Historical equity premium: 6.0%

# Calculate Cost of Equity
cost_of_equity = risk_free_rate + (beta * equity_risk_premium)

print("\\nCOST OF EQUITY (CAPM):")
print("-" * 70)
print(f"Risk-Free Rate (Rf):        {risk_free_rate * 100:.1f}%")
print(f"Beta (β):                   {beta:.2f}")
print(f"Equity Risk Premium (Rm-Rf): {equity_risk_premium * 100:.1f}%")
print(f"\\nCost of Equity (Re):         {cost_of_equity * 100:.2f}%")
print(f"  Formula: {risk_free_rate*100:.1f}% + {beta:.1f} ×
{equity_risk_premium*100:.1f}%")

# Debt Inputs
cost_of_debt_pretax = 0.050  # 5.0% interest rate
tax_rate = 0.25              # 25% tax rate
```

```
cost_of_debt_aftertax = cost_of_debt_pretax * (1 - tax_rate)

print("\\n" + "-" * 70)
print("COST OF DEBT:")
print("-" * 70)
print(f"Pre-tax Cost of Debt:       {cost_of_debt_pretax * 100:.1f}%")
print(f"Tax Rate:                   {tax_rate * 100:.1f}%")
print(f"After-tax Cost of Debt:     {cost_of_debt_aftertax * 100:.2f}%")
print(f"  (Debt is tax-deductible!)")

# Capital Structure
market_value_equity = 5000  # $5,000M market cap
market_value_debt = 1000    # $1,000M debt
total_capital = market_value_equity + market_value_debt

weight_equity = market_value_equity / total_capital
weight_debt = market_value_debt / total_capital

print("\\n" + "-" * 70)
print("CAPITAL STRUCTURE:")
print("-" * 70)
print(f"Market Value of Equity:     ${market_value_equity:,.0f}M")
print(f"Market Value of Debt:       ${market_value_debt:,.0f}M")
print(f"Total Capital:              ${total_capital:,.0f}M")
print(f"\\nWeight of Equity (E/V):      {weight_equity * 100:.1f}%")
print(f"Weight of Debt (D/V):        {weight_debt * 100:.1f}%")

# CALCULATE WACC
wacc = (weight_equity * cost_of_equity) + (weight_debt *
cost_of_debt_aftertax)

print("\\n" + "=" * 70)
print("WACC CALCULATION:")
print("=" * 70)
print(f"WACC = (E/V × Re) + (D/V × Rd × (1-T))")
print(f"\\nWACC = ({weight_equity:.1%} × {cost_of_equity:.2%}) +
({weight_debt:.1%} × {cost_of_debt_aftertax:.2%})")
print(f"\\nWACC = {wacc * 100:.2f}%")
print("=" * 70)

print("\\n✅ WACC calculation complete!")
print("💡 This is your discount rate for the DCF!")
```

WACC is critical! Small changes have BIG impact on valuation.

Let's Build: Step 7 - Complete Valuation

Create a new file: `07_valuation_complete.py`

```
"""
Step 7: Complete DCF Valuation

Bring it ALL together:
1. Discount projected FCFs to present value
2. Calculate terminal value
3. Calculate enterprise value
4. Bridge to equity value and price per share
"""

import pandas as pd
import numpy as np

print("STEP 7: COMPLETE DCF VALUATION")
print("=" * 70)

# Free Cash Flows from Step 5
years = list(range(2025, 2030))
fcfs = [194.3, 211.6, 227.5, 243.1, 258.0]

# WACC from Step 6
wacc = 0.0995  # 9.95%

print("\\n1. DISCOUNT PROJECTED FCFs TO PRESENT VALUE:")
print("-" * 70)

# Calculate present value of each FCF
pv_fcfs = []
for i, fcf in enumerate(fcfs, 1):
    pv = fcf / ((1 + wacc) ** i)
    pv_fcfs.append(pv)
    print(f"Year {i} ({years[i-1]}): FCF ${fcf:.1f}M / (1.0995^{i}) = PV
${pv:.1f}M")

pv_projection_period = sum(pv_fcfs)

print(f"\\nPV of Projection Period (Years 1-5):
${pv_projection_period:.1f}M")

# 2. TERMINAL VALUE
print("\\n" + "=" * 70)
print("2. TERMINAL VALUE CALCULATION:")
print("=" * 70)

# Assumptions
terminal_growth_rate = 0.03  # 3% perpetual growth
terminal_fcf_year_6 = fcfs[-1] * (1 + terminal_growth_rate)

# Perpetuity formula: TV = FCF(n+1) / (WACC - g)
terminal_value = terminal_fcf_year_6 / (wacc - terminal_growth_rate)

print(f"\\nYear 6 FCF (Year 5 × 1.03): ${terminal_fcf_year_6:.1f}M")
print(f"Terminal Growth Rate:        {terminal_growth_rate * 100:.1f}%")
```

```
print(f"WACC:                          {wacc * 100:.2f}%")
print(f"\\nTerminal Value = ${terminal_fcf_year_6:.1f}M / ({wacc:.4f} -
{terminal_growth_rate:.2f})")
print(f"Terminal Value = ${terminal_value:.1f}M")

# PV of Terminal Value (discount back 5 years)
pv_terminal_value = terminal_value / ((1 + wacc) ** 5)

print(f"\\nPV of Terminal Value: ${pv_terminal_value:.1f}M")

# 3. ENTERPRISE VALUE
print("\\n" + "=" * 70)
print("3. ENTERPRISE VALUE:")
print("=" * 70)

enterprise_value = pv_projection_period + pv_terminal_value

print(f"PV of Projection Period:  ${pv_projection_period:.1f}M")
print(f"PV of Terminal Value:     ${pv_terminal_value:.1f}M")
print(f"\\nENTERPRISE VALUE:          ${enterprise_value:.1f}M")

# Terminal value as % of total (sanity check: should be 60-80%)
tv_pct = (pv_terminal_value / enterprise_value) * 100
print(f"\\n💡 Terminal Value is {tv_pct:.1f}% of total value")

# 4. EQUITY VALUE
print("\\n" + "=" * 70)
print("4. BRIDGE TO EQUITY VALUE:")
print("=" * 70)

# Balance sheet items
cash = 200
debt = 1000
minority_interest = 0
preferred_stock = 0

equity_value = enterprise_value + cash - debt - minority_interest -
preferred_stock

print(f"Enterprise Value:          ${enterprise_value:,.1f}M")
print(f"  + Cash:                  ${cash:,.1f}M")
print(f"  - Debt:                  ${debt:,.1f}M")
print(f"  - Minority Interest:     ${minority_interest:,.1f}M")
print(f"  - Preferred Stock:       ${preferred_stock:,.1f}M")
print(f"\\nEQUITY VALUE:              ${equity_value:,.1f}M")

# Price per share
shares_outstanding = 100  # 100M shares
price_per_share = equity_value / shares_outstanding

print(f"\\nShares Outstanding:        {shares_outstanding:.0f}M")
print(f"\\nPRICE PER SHARE:           ${price_per_share:.2f}")

# SUMMARY TABLE
```

```
print("\\n" + "=" * 70)
print("DCF VALUATION SUMMARY:")
print("=" * 70)

summary = pd.DataFrame({
    'Metric': [
        'WACC',
        'Terminal Growth Rate',
        'PV of Projection Period',
        'PV of Terminal Value',
        'Enterprise Value',
        'Equity Value',
        'Price per Share'
    ],
    'Value': [
        f'{wacc * 100:.2f}%',
        f'{terminal_growth_rate * 100:.1f}%',
        f'${pv_projection_period:.1f}M',
        f'${pv_terminal_value:.1f}M',
        f'${enterprise_value:.1f}M',
        f'${equity_value:.1f}M',
        f'${price_per_share:.2f}'
    ]
})

print(summary.to_string(index=False))
print("=" * 70)

print("\\n🎉 COMPLETE DCF VALUATION DONE!")
print("💡 You just valued a company like a professional investment
banker!")
```

CONGRATULATIONS! You've built a complete DCF model from scratch!

The Complete DCF Class - `dcf_model.py`

The file `dcf_model.py` in this directory contains a **production-quality DCF class** that encapsulates everything you've learned.

Key features:

- ✅ Object-oriented design (professional!)
- ✅ All calculations in one reusable class
- ✅ Sensitivity analysis built-in
- ✅ Clear documentation with docstrings
- ✅ Excel export capability

How to use it:

```
from dcf_model import DCFModel

# Initialize
model = DCFModel("TechCo Inc.", "TECH")

# Set assumptions
model.set_historical_data(years=[2022, 2023, 2024],
                          revenue=[800, 920, 1050],
                          ebitda=[160, 200, 250])

model.set_revenue_assumptions([0.18, 0.15, 0.12, 0.10, 0.08])

model.set_operating_assumptions(ebitda_margin=0.25, tax_rate=0.25,
                                da_pct_revenue=0.03,
capex_pct_revenue=0.04,
                                nwc_pct_revenue=0.10)

model.set_wacc_assumptions(risk_free_rate=0.04, equity_risk_premium=0.06,
                           beta=1.2, cost_of_debt=0.05,
                           mv_equity=5000, mv_debt=1000)

model.set_terminal_assumptions(growth_rate=0.03, ebitda_multiple=12.0)

# Build and value
projections = model.build_projections()
dcf_results = model.calculate_dcf()
equity_value = model.calculate_equity_value(cash=200, debt=1000,
shares_outstanding=100)

# Display results
model.display_summary()

# Run sensitivity
sensitivity = model.sensitivity_analysis(
    wacc_range=[0.08, 0.09, 0.10, 0.11, 0.12],
    tg_range=[0.02, 0.025, 0.03, 0.035, 0.04]
)
print(sensitivity)
```

Study the `dcf_model.py` file to see professional Python code organization!

DCF Best Practices (What IB Analysts Do)

1. Model Structure

- **Separate inputs, calculations, and outputs**
  - Never hardcode assumptions in formulas
  - Make assumptions easy to change
  - Document everything

2. Assumption Quality

Revenue Growth:

- ✅ Use industry research reports (Gartner, IDC)
- ✅ Look at management guidance
- ✅ Analyze historical trends
- ✅ Compare to competitors
- ❌ Don't just guess or use constant growth

Margins:

- ✅ Historical average (3-5 years)
- ✅ Compare to peer companies
- ✅ Consider scale effects (margins improve as company grows)
- ❌ Don't assume margins improve forever

WACC:

- ✅ Use current 10-year Treasury for risk-free rate
- ✅ Use market-based capital structure (not book value!)
- ✅ Beta from Bloomberg or CapitalIQ
- ✅ Equity risk premium: 5-7% (historical average)
- ❌ Don't use target capital structure (use current)

Terminal Growth:

- ✅ Usually 2-3% (long-term GDP growth)
- ✅ NEVER higher than GDP growth (impossible long-term)
- ✅ Lower for mature industries (1-2%)
- ❌ Don't use company's current growth rate

3. Sanity Checks (CRITICAL!)

After building your DCF, check:

Terminal Value:

- Should be 60-80% of enterprise value
- If >90%: Your projections are too short or growth too low
- If <50%: Your terminal assumptions might be too conservative

Implied Multiples:

```
implied_ev_revenue = enterprise_value / revenue_ltm
implied_ev_ebitda = enterprise_value / ebitda_ltm
```

- Compare to trading comps - should be in similar range
- If wildly different: Your assumptions are likely off

ROIC vs WACC:

- If ROIC > WACC: Company creates value (good!)
- If ROIC < WACC: Company destroys value (problem!)
- Growing companies should have ROIC >> WACC

4. Sensitivity Analysis (ALWAYS DO THIS!)

Test your key assumptions:

- WACC ± 1-2% (small changes = big impact!)
- Terminal growth ± 0.5%
- Revenue growth ± 2-3%
- EBITDA margin ± 1-2%

Create a tornado chart showing which assumptions drive value most.

5. Comparable Company Check

Your DCF should be **in the ballpark** of trading comps:

- If DCF value > trading comps: You might be too optimistic
- If DCF value < trading comps: You might be too conservative
- If massively different: Re-check your assumptions!

Investment banking rule: Never show a DCF in isolation - always compare to comps and precedents!

Practice Exercises

Apply everything you've learned! These exercises simulate real IB work.

Exercise 1: Build a Complete DCF from Scratch

Pick a public company and value it:

1. Download 3-5 years of historical financials
2. Calculate historical growth rates and margins
3. Set reasonable forward assumptions:
    - Revenue growth (declining over time)
    - EBITDA margin (based on historicals)
    - Tax rate, CapEx %, NWC %
4. Calculate WACC (use real market data)
5. Build 5-year projections
6. Calculate enterprise and equity value
7. Compare to actual market cap - are they close?

Bonus: Export your model to Excel with all assumptions clearly labeled

Exercise 2: Sensitivity Analysis

Test how assumptions affect valuation:

1. Create a 2-way sensitivity table:
    - X-axis: Terminal growth (2.0%, 2.5%, 3.0%, 3.5%, 4.0%)
    - Y-axis: WACC (8%, 9%, 10%, 11%, 12%)
    - Values: Price per share
2. Create a tornado chart:
    - Test each key assumption (revenue growth, margin, WACC, etc.)
    - Show which assumptions have the biggest impact
    - Rank from highest to lowest impact
3. Determine: Which assumption should you research most carefully?

Exercise 3: Three-Scenario Analysis

Build Base / Bull / Bear cases:

Base Case:

- Most likely assumptions
- Use historical averages
- Conservative terminal growth (2.5%)

Bull Case:

- Optimistic revenue growth (+2-3% vs base)
- Margin expansion (+1-2%)
- Higher terminal growth (3.5%)

Bear Case:

- Conservative revenue growth (-2-3% vs base)
- Margin compression (-1-2%)
- Lower terminal growth (2.0%)

Calculate the equity value range. This gives you a **valuation range**, not a point estimate!

Exercise 4: Real-World Application

Recreate an actual M&A deal:

1. Pick a recent acquisition (last 2-3 years)
2. Find the purchase price (public information)
3. Build a DCF model for the target company
4. Try to match the purchase price with your assumptions
5. What assumptions would justify the price paid?

Suggested deals:

- Microsoft / Activision Blizzard
- Broadcom / VMware
- Salesforce / Slack

This is EXACTLY what IB analysts do in M&A fairness opinions!

Solutions

Complete solutions to all exercises are in `solutions.py`.

Before checking solutions, try each exercise yourself! The learning comes from struggling through it.

Key concepts you're practicing:

- Complete DCF build from raw data
- Sensitivity analysis (finding what matters most)
- Scenario analysis (dealing with uncertainty)
- Reverse-engineering valuations (thinking like a buyer)

Special Celebration! 🎉

When you complete this module, run this special script your father created for you:

```
python dcf_celebration.py
```

There's a very special message waiting for you! 💝

Key Takeaways

✅ **DCF = Present value of future cash flows** ✅ **FCF is the cash available to all investors** (NOPAT + D&A - CapEx - Δ NWC) ✅ **WACC is your discount rate** (weighted cost of debt + equity) ✅ **Terminal value is 60-80% of total value** (be careful with this!) ✅ **Always do sensitivity analysis** (small changes = big impact) ✅ **Sanity check vs comparable companies** (is your value reasonable?) ✅ **Build scenarios, not point estimates** (Base / Bull / Bear)

The DCF is the most powerful valuation tool in finance!

Next Steps

Continue to: `Module_05_LBO_Modeling/01_LBO_Overview.md`

You'll learn how private equity firms use leveraged buyouts to generate returns!

Estimated Time: 5-6 hours **Difficulty:** ⭐ ⭐ ⭐ Intermediate-Advanced **Prerequisites:** Modules 1-3 (Python, Pandas fundamentals)

Real-world impact: This is EXACTLY what you'll do at:

- Goldman Sachs, JP Morgan (M&A Advisory)
- Blackstone, KKR (PE Investing)
- Hedge funds (Public equity research)

Master the DCF, and you're ready for Wall Street! 🚀