## 0. Imports and Setting up Anthropic API Client

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install python-dotenv

import os
import dotenv

dotenv.load_dotenv('/content/drive/MyDrive/.env')
```

```
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
True
```

```
# Load Prompts and Problem Description
prompt1_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt1_MathematicalModel.txt'
prompt2_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt2_PyomoCode.txt'
problem_desc_path = '/content/drive/MyDrive/Thesis/ProblemDescriptions/NL/NL4.txt'

prompt1_file = open(prompt1_path, "r")
prompt2_file = open(prompt2_path, "r")
problem_desc_file = open(problem_desc_path, "r")

prompt1 = prompt1_file.read()
print("Prompt 1:\n", prompt1)

prompt2 = prompt2_file.read()
print("Prompt 2:\n", prompt2)

problem_desc = problem_desc_file.read()
print("Problem Description:\n", problem_desc)
```

```
Prompt 1:
 Please write a mathematical optimization model for this problem. Include parameters, decision variables, the objective
Prompt 2:
 Please write a python pyomo code for this optimization problem.
Use sample data where needed.
Indicate where you use sample data.
Problem Description:
 We are looking at an alkylation process which will include the following 10 variables: olefin feed (barrels per day), i

We want to maximize the daily profit of this alkylation process.
The profit is defined as the revenue generated from the alkylate yield multiplied with the motor octane number, minus th

Relationships in terms of other variables for alkylate yield, motor octane number, acid dilution factor, and F-4 perform
This regression estimate can deviate in both directions from true value of these variables by 2, 1, 5 and 10 percent, re
Alkylate yield is a function of olefin feed and external isobutane-to-olefine yield. Alkalyte yield equals the amount of
The motor octane number is derived from the external isobutane-to-olefin ratio and the acid strength. The motor octane n
The acid dilution factor is calculated based on the F-4 performance number. The acid dillution factor is expressed as 35
Lastly, the F-4 performance number depends on the motor octane number. F-4 performance number is calculated as -133 plus

There are some additional constraints imposed by the nature of the chemical process.
Each variable has a lower and an upper bound.
The external isobutane-to-olefin ratio needs to equal the ratio of isobutane recycle plus isobutane makeup to olefin fee
The acid strength needs to equal the ratio of 93000 times acid addition rate to acid addition rate multiplied by acid di
Lastly, 1.22 alkylate yield needs to be equal to the combined olefin feed and isobutane makeup.
```

```
!pip install anthropic
```

Show hidden output

```
# Importing Anthropic & Setting Headers
import anthropic

client = anthropic.Anthropic(
    # defaults to os.environ.get("ANTHROPIC_API_KEY")
    api_key=os.environ.get("ANTHROPIC_API_KEY"),
)
```

## 1. Prompt 1: Create Mathematical Model

```
message = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt1,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": problem_desc}
    ]
)

response_p1 = message.content[0].text


# Print response
print(response_p1)
```

```
Let's formulate the optimization model for the alkylation process:

Parameters:
- p_alk: Price of alkylate per barrel
- c_ole: Cost of olefin feed per barrel
- c_iso: Cost of isobutane recycle per barrel
- c_acid: Cost of acid addition per thousand pounds
- c_makeup: Cost of isobutane makeup per barrel

Decision variables:
- OF: Olefin feed (barrels per day)
- IR: Isobutane recycle (barrels per day)
- AAR: Acid addition rate (thousands of pounds per day)
- IM: Isobutane makeup (barrels per day)
- AY: Alkylate yield (barrels per day)
- AS: Acid strength (weight per cent)
- MON: Motor octane number
- IOR: External isobutane-to-olefin ratio
- ADF: Acid dilution factor
- F4: F-4 performance number

Objective function:
Maximize profit = p_alk * AY * MON - (c_ole * OF + c_iso * IR + c_acid * AAR + c_makeup * IM)

Constraints:
1. Alkylate yield: 0.98 * (1.12 * OF + 0.13167 * IOR * OF - 0.00667 * IOR^2 * OF) ≤ AY ≤ 1.02 * (1.12 * OF + 0.13167 * I
2. Motor octane number: 0.99 * (86.35 + 1.098 * IOR - 0.038 * IOR^2 + 0.325 * (AS - 89)) ≤ MON ≤ 1.01 * (86.35 + 1.098 *
3. Acid dilution factor: 0.95 * (35.82 - 0.222 * F4) ≤ ADF ≤ 1.05 * (35.82 - 0.222 * F4)
4. F-4 performance number: 0.90 * (-133 + 3 * MON) ≤ F4 ≤ 1.10 * (-133 + 3 * MON)
5. External isobutane-to-olefin ratio: IOR = (IR + IM) / OF
6. Acid strength: AS = (93000 * AAR) / (AAR * ADF + 1000 * AAR)
7. Material balance: 1.22 * AY = OF + IM
8. Variable bounds:
   - OF_min ≤ OF ≤ OF_max
   - IR_min ≤ IR ≤ IR_max
   - AAR_min ≤ AAR ≤ AAR_max
   - IM_min ≤ IM ≤ IM_max
   - AY_min ≤ AY ≤ AY_max
   - AS_min ≤ AS ≤ AS_max
   - MON_min ≤ MON ≤ MON_max
   - IOR_min ≤ IOR ≤ IOR_max
   - ADF_min ≤ ADF ≤ ADF_max
   - F4_min ≤ F4 ≤ F4_max

Note: The subscripts _min and _max denote the lower and upper bounds for each variable, respectively.
```

## 2. Prompt 2: Write Code based on Mathematical Model

```
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4069,
    messages=[
        {"role": "user", "content": response_p1}
    ]
)

response_p2 = message2.content[0].text


# Print response
print(response_p2)
```

```
model.c_ole = pyo.Param(initialize=50)   # Cost of olefin feed per barrel (sample data)
model.c_iso = pyo.Param(initialize=30)   # Cost of isobutane recycle per barrel (sample data)
model.c_acid = pyo.Param(initialize=10)  # Cost of acid addition per thousand pounds (sample data)
model.c_makeup = pyo.Param(initialize=40)  # Cost of isobutane makeup per barrel (sample data)

# Define decision variables
model.OF = pyo.Var(bounds=(5000, 10000))  # Olefin feed (barrels per day)
model.IR = pyo.Var(bounds=(1000, 5000))   # Isobutane recycle (barrels per day)
model.AAR = pyo.Var(bounds=(1, 5))        # Acid addition rate (thousands of pounds per day)
model.IM = pyo.Var(bounds=(500, 2000))    # Isobutane makeup (barrels per day)
model.AY = pyo.Var()                      # Alkylate yield (barrels per day)
model.AS = pyo.Var()                      # Acid strength (weight per cent)
model.MON = pyo.Var()                     # Motor octane number
model.IOR = pyo.Var()                     # External isobutane-to-olefin ratio
model.ADF = pyo.Var()                     # Acid dilution factor
model.F4 = pyo.Var()                      # F-4 performance number

# Define objective function
model.profit = pyo.Objective(expr=model.p_alk * model.AY * model.MON - (model.c_ole * model.OF +
                                                                       model.c_iso * model.IR +
                                                                       model.c_acid * model.AAR +
                                                                       model.c_makeup * model.IM),
                             sense=pyo.maximize)

# Define constraints
model.alkylate_yield_lower = pyo.Constraint(expr=0.98 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF - 0.00667 *
model.alkylate_yield_upper = pyo.Constraint(expr=model.AY <= 1.02 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF
model.octane_number_lower = pyo.Constraint(expr=0.99 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2 + 0.325 * (mo
model.octane_number_upper = pyo.Constraint(expr=model.MON <= 1.01 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2
model.acid_dilution_lower = pyo.Constraint(expr=0.95 * (35.82 - 0.222 * model.F4) <= model.ADF)
model.acid_dilution_upper = pyo.Constraint(expr=model.ADF <= 1.05 * (35.82 - 0.222 * model.F4))
model.f4_performance_lower = pyo.Constraint(expr=0.90 * (-133 + 3 * model.MON) <= model.F4)
model.f4_performance_upper = pyo.Constraint(expr=model.F4 <= 1.10 * (-133 + 3 * model.MON))
model.external_ratio = pyo.Constraint(expr=model.IOR == (model.IR + model.IM) / model.OF)
model.acid_strength = pyo.Constraint(expr=model.AS == (93000 * model.AAR) / (model.AAR * model.ADF + 1000 * model.AAR)
model.material_balance = pyo.Constraint(expr=1.22 * model.AY == model.OF + model.IM)

# Solve the optimization problem
solver = pyo.SolverFactory('ipopt')
results = solver.solve(model)

# Print the results
print("Optimization Results:")
print("Objective Function (Profit): ", pyo.value(model.profit))
print("Olefin Feed (OF): ", pyo.value(model.OF))
print("Isobutane Recycle (IR): ", pyo.value(model.IR))
print("Acid Addition Rate (AAR): ", pyo.value(model.AAR))
print("Isobutane Makeup (IM): ", pyo.value(model.IM))
print("Alkylate Yield (AY): ", pyo.value(model.AY))
print("Acid Strength (AS): ", pyo.value(model.AS))
print("Motor Octane Number (MON): ", pyo.value(model.MON))
print("External Isobutane-to-Olefin Ratio (IOR): ", pyo.value(model.IOR))
print("Acid Dilution Factor (ADF): ", pyo.value(model.ADF))
print("F-4 Performance Number (F4): ", pyo.value(model.F4))
```

Note: I have used sample data for the price and cost parameters in the code. These values need to be replaced with the

## 4. Input Problem Data and Test Model Code

```
# Installing pyomo nd solver
!pip install -q pyomo
!wget -N -q "https://matematica.unipv.it/gualandi/solvers/ipopt-linux64.zip"
!unzip -o -q ipopt-linux64
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 23.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 3.0 MB/s eta 0:00:00
```

```
import pyomo.environ as pyo

# Create a concrete model
model = pyo.ConcreteModel()

# Define parameters
model.p_alk = pyo.Param(initialize=0.63)  # Price of alkylate per barrel (sample data)
model.c_ole = pyo.Param(initialize=5.04)   # Cost of olefin feed per barrel (sample data)
model.c_iso = pyo.Param(initialize=0.035)   # Cost of isobutane recycle per barrel (sample data)
model.c_acid = pyo.Param(initialize=10)  # Cost of acid addition per thousand pounds (sample data)
model.c_makeup = pyo.Param(initialize=3.36)  # Cost of isobutane makeup per barrel (sample data)

# Define decision variables
model.OF = pyo.Var(bounds=(0, 2000))  # Olefin feed (barrels per day)
model.IR = pyo.Var(bounds=(0, 16000))   # Isobutane recycle (barrels per day)
model.AAR = pyo.Var(bounds=(0, 120))        # Acid addition rate (thousands of pounds per day)
model.IM = pyo.Var(bounds=(0, 2000))    # Isobutane makeup (barrels per day)
model.AY = pyo.Var()                      # Alkylate yield (barrels per day)
model.AS = pyo.Var()                      # Acid strength (weight per cent)
model.MON = pyo.Var()                     # Motor octane number
model.IOR = pyo.Var()                     # External isobutane-to-olefin ratio
model.ADF = pyo.Var()                     # Acid dilution factor
model.F4 = pyo.Var()                      # F-4 performance number

# Define objective function
model.profit = pyo.Objective(expr=model.p_alk * model.AY * model.MON - (model.c_ole * model.OF +
                                                                        model.c_iso * model.IR +
                                                                        model.c_acid * model.AAR +
                                                                        model.c_makeup * model.IM),
                             sense=pyo.maximize)

# Define constraints
model.alkylate_yield_lower = pyo.Constraint(expr=0.98 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF - 0.00667 * model.
model.alkylate_yield_upper = pyo.Constraint(expr=model.AY <= 1.02 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF - 0.00
model.octane_number_lower = pyo.Constraint(expr=0.99 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2 + 0.325 * (model.AS
model.octane_number_upper = pyo.Constraint(expr=model.MON <= 1.01 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2 + 0.32
model.acid_dilution_lower = pyo.Constraint(expr=0.95 * (35.82 - 0.222 * model.F4) <= model.ADF)
model.acid_dilution_upper = pyo.Constraint(expr=model.ADF <= 1.05 * (35.82 - 0.222 * model.F4))
model.f4_performance_lower = pyo.Constraint(expr=0.90 * (-133 + 3 * model.MON) <= model.F4)
model.f4_performance_upper = pyo.Constraint(expr=model.F4 <= 1.10 * (-133 + 3 * model.MON))
model.external_ratio = pyo.Constraint(expr=model.IOR == (model.IR + model.IM) / model.OF)
model.acid_strength = pyo.Constraint(expr=model.AS == (93000 * model.AAR) / (model.AAR * model.ADF + 1000 * model.AAR))
model.material_balance = pyo.Constraint(expr=1.22 * model.AY == model.OF + model.IM)

# Solve the optimization problem
solver = pyo.SolverFactory('ipopt')
results = solver.solve(model)

# Print the results
print("Optimization Results:")
print("Objective Function (Profit): ", pyo.value(model.profit))
print("Olefin Feed (OF): ", pyo.value(model.OF))
print("Isobutane Recycle (IR): ", pyo.value(model.IR))
print("Acid Addition Rate (AAR): ", pyo.value(model.AAR))
print("Isobutane Makeup (IM): ", pyo.value(model.IM))
print("Alkylate Yield (AY): ", pyo.value(model.AY))
print("Acid Strength (AS): ", pyo.value(model.AS))
print("Motor Octane Number (MON): ", pyo.value(model.MON))
print("External Isobutane-to-Olefin Ratio (IOR): ", pyo.value(model.IOR))
print("Acid Dilution Factor (ADF): ", pyo.value(model.ADF))
print("F-4 Performance Number (F4): ", pyo.value(model.F4))
```

```
Optimization Results:
Objective Function (Profit):  176669.7143562548
Olefin Feed (OF):  2000.0
Isobutane Recycle (IR):  10113.175625405212
Acid Addition Rate (AAR):  0.0
Isobutane Makeup (IM):  2000.0
Alkylate Yield (AY):  3278.6885573769496
Acid Strength (AS):  93.000000664579
Motor Octane Number (MON):  93.83527053539444
External Isobutane-to-Olefin Ratio (IOR):  6.056587762136937
Acid Dilution Factor (ADF):  -7.146010636217708e-06
F-4 Performance Number (F4):  161.3513836212632
```

## 5. Correct The Model Code to Test Mathematical Model (if applicable)

```
import pyomo.environ as pyo

# Create a concrete model
model = pyo.ConcreteModel()
```

```
# Define parameters
model.p_alk = pyo.Param(initialize=0.63)  # Price of alkylate per barrel (sample data)
model.c_ole = pyo.Param(initialize=5.04)   # Cost of olefin feed per barrel (sample data)
model.c_iso = pyo.Param(initialize=0.035)   # Cost of isobutane recycle per barrel (sample data)
model.c_acid = pyo.Param(initialize=10)  # Cost of acid addition per thousand pounds (sample data)
model.c_makeup = pyo.Param(initialize=3.36)  # Cost of isobutane makeup per barrel (sample data)

# Define decision variables
model.OF = pyo.Var(bounds=(0, 2000))  # Olefin feed (barrels per day)
model.IR = pyo.Var(bounds=(0, 16000))   # Isobutane recycle (barrels per day)
model.AAR = pyo.Var(bounds=(0, 120))        # Acid addition rate (thousands of pounds per day)
model.IM = pyo.Var(bounds=(0, 2000))    # Isobutane makeup (barrels per day)
model.AY = pyo.Var(bounds=(0, 5000))                     # Alkylate yield (barrels per day)
model.AS = pyo.Var(bounds=(85, 93))                     # Acid strength (weight per cent)
model.MON = pyo.Var(bounds=(90, 95))                    # Motor octane number
model.IOR = pyo.Var(bounds=(3, 12))                    # External isobutane-to-olefin ratio
model.ADF = pyo.Var(bounds=(1.2, 4))                    # Acid dilution factor
model.F4 = pyo.Var(bounds=(145, 162))                     # F-4 performance number

# Define objective function
model.profit = pyo.Objective(expr=model.p_alk * model.AY * model.MON - (model.c_ole * model.OF +
                                                                        model.c_iso * model.IR +
                                                                        model.c_acid * model.AAR +
                                                                        model.c_makeup * model.IM),
                             sense=pyo.maximize)

# Define constraints
model.alkylate_yield_lower = pyo.Constraint(expr=0.98 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF - 0.00667 * model.I
model.alkylate_yield_upper = pyo.Constraint(expr=model.AY <= 1.02 * (1.12 * model.OF + 0.13167 * model.IOR * model.OF - 0.006
model.octane_number_lower = pyo.Constraint(expr=0.99 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2 + 0.325 * (model.AS
model.octane_number_upper = pyo.Constraint(expr=model.MON <= 1.01 * (86.35 + 1.098 * model.IOR - 0.038 * model.IOR**2 + 0.325
model.acid_dilution_lower = pyo.Constraint(expr=0.95 * (35.82 - 0.222 * model.F4) <= model.ADF)
model.acid_dilution_upper = pyo.Constraint(expr=model.ADF <= 1.05 * (35.82 - 0.222 * model.F4))
model.f4_performance_lower = pyo.Constraint(expr=0.90 * (-133 + 3 * model.MON) <= model.F4)
model.f4_performance_upper = pyo.Constraint(expr=model.F4 <= 1.10 * (-133 + 3 * model.MON))
model.external_ratio = pyo.Constraint(expr=model.IOR == (model.IR + model.IM) / model.OF)
model.acid_strength = pyo.Constraint(expr=model.AS == (93000 * model.AAR) / (model.AAR * model.ADF + 1000 * model.AAR))
model.material_balance = pyo.Constraint(expr=1.22 * model.AY == model.OF + model.IM)

# Solve the optimization problem
solver = pyo.SolverFactory('ipopt')
results = solver.solve(model)

# Print the results
print("Optimization Results:")
print("Objective Function (Profit): ", pyo.value(model.profit))
print("Olefin Feed (OF): ", pyo.value(model.OF))
print("Isobutane Recycle (IR): ", pyo.value(model.IR))
print("Acid Addition Rate (AAR): ", pyo.value(model.AAR))
print("Isobutane Makeup (IM): ", pyo.value(model.IM))
print("Alkylate Yield (AY): ", pyo.value(model.AY))
print("Acid Strength (AS): ", pyo.value(model.AS))
print("Motor Octane Number (MON): ", pyo.value(model.MON))
print("External Isobutane-to-Olefin Ratio (IOR): ", pyo.value(model.IOR))
print("Acid Dilution Factor (ADF): ", pyo.value(model.ADF))
print("F-4 Performance Number (F4): ", pyo.value(model.F4))
```

```
Optimization Results:
Objective Function (Profit):  176594.13705361064
Olefin Feed (OF):  2000.0
Isobutane Recycle (IR):  10113.175625405214
Acid Addition Rate (AAR):  0.0
Isobutane Makeup (IM):  2000.0
Alkylate Yield (AY):  3278.688557376949
Acid Strength (AS):  92.88853376059825
Motor Octane Number (MON):  93.79868152416277
External Isobutane-to-Olefin Ratio (IOR):  6.056587762136939
Acid Dilution Factor (ADF):  1.2
F-4 Performance Number (F4):  155.9263365300636
```