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/LP/LP4.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:
 Consider a farmer who specializes in raising wheat, corn, and sugar beets on her 500 acres of land. During the winter,

The farmer knows that at least 200 tons (T) of wheat and 240 T of corn are needed for cattle feed. These amounts can be
Any production in excess of the feeding requirement would be sold.

Over the last decade, mean selling prices have been $170 and $150 per ton of wheat and corn, respectively.
The purchase prices are 40% more than this due to the wholesaler’s margin and transportation costs.

Another profitable crop is sugar beet, which she expects to sell at $36/T. However, the European Commission imposes a qu

Based on past experience, the farmer knows that the mean yield on her land is roughly 2.5 T, 3 T, and 20 T per acre for

The farmer can decide how much of her land she will use to grow each of the three products: wheat, corn and sugar beets.
```

```
!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)
```

```
Great! Let's formulate the mathematical optimization model for the farmer's problem. We'll define the parameters, decisi

Parameters:
- W_min: minimum requirement of wheat for cattle feed (200 T)
- C_min: minimum requirement of corn for cattle feed (240 T)
- P_ws: selling price of wheat ($170/T)
- P_cs: selling price of corn ($150/T)
- P_wp: purchase price of wheat ($238/T, which is 40% more than the selling price)
- P_cp: purchase price of corn ($210/T, which is 40% more than the selling price)
- P_bs: selling price of sugar beets within the quota ($36/T)
- P_be: selling price of sugar beets exceeding the quota ($10/T)
- Q_b: quota for sugar beet production (6000 T)
- Y_w: yield of wheat per acre (2.5 T/acre)
- Y_c: yield of corn per acre (3 T/acre)
- Y_b: yield of sugar beets per acre (20 T/acre)
- C_w: cost of planting an acre of wheat ($150/acre)
- C_c: cost of planting an acre of corn ($230/acre)
- C_b: cost of planting an acre of sugar beets ($260/acre)
- L: total available land (500 acres)

Decision Variables:
- A_w: acres of land devoted to wheat
- A_c: acres of land devoted to corn
- A_b: acres of land devoted to sugar beets
- W_s: tons of wheat sold
- W_p: tons of wheat purchased
- C_s: tons of corn sold
- C_p: tons of corn purchased
- B_s: tons of sugar beets sold within the quota
- B_e: tons of sugar beets sold exceeding the quota

Objective Function:
Minimize total costs = Planting costs + Purchasing costs - Sales revenue
Min: C_w * A_w + C_c * A_c + C_b * A_b + P_wp * W_p + P_cp * C_p - P_ws * W_s - P_cs * C_s - P_bs * B_s - P_be * B_e

Constraints:
1. Land constraint: A_w + A_c + A_b <= L
2. Wheat balance: Y_w * A_w + W_p - W_s >= W_min
3. Corn balance: Y_c * A_c + C_p - C_s >= C_min
4. Sugar beet quota: B_s <= Q_b
5. Sugar beet balance: Y_b * A_b = B_s + B_e
6. Non-negativity constraints: A_w, A_c, A_b, W_s, W_p, C_s, C_p, B_s, B_e >= 0

The objective function minimizes the total costs, which include planting costs, purchasing costs, and sales revenue (as
```

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.Y_b = Param(initialize=20)  # yield of sugar beets per acre (T/acre)
    model.C_w = Param(initialize=150)  # cost of planting an acre of wheat ($/acre)
    model.C_c = Param(initialize=230)  # cost of planting an acre of corn ($/acre)
    model.C_b = Param(initialize=260)  # cost of planting an acre of sugar beets ($/acre)
    model.L = Param(initialize=500)  # total available land (acres)

    # Define decision variables
    model.A_w = Var(domain=NonNegativeReals)  # acres of land devoted to wheat
    model.A_c = Var(domain=NonNegativeReals)  # acres of land devoted to corn
    model.A_b = Var(domain=NonNegativeReals)  # acres of land devoted to sugar beets
    model.W_s = Var(domain=NonNegativeReals)  # tons of wheat sold
    model.W_p = Var(domain=NonNegativeReals)  # tons of wheat purchased
    model.C_s = Var(domain=NonNegativeReals)  # tons of corn sold
    model.C_p = Var(domain=NonNegativeReals)  # tons of corn purchased
    model.B_s = Var(domain=NonNegativeReals)  # tons of sugar beets sold within the quota
    model.B_e = Var(domain=NonNegativeReals)  # tons of sugar beets sold exceeding the quota

    # Define objective function
    def total_cost(model):
        return (model.C_w * model.A_w + model.C_c * model.A_c + model.C_b * model.A_b +
                model.P_wp * model.W_p + model.P_cp * model.C_p -
                model.P_ws * model.W_s - model.P_cs * model.C_s -
                model.P_bs * model.B_s - model.P_be * model.B_e)

    model.objective = Objective(rule=total_cost, sense=minimize)

    # Define constraints
    model.land_constraint = Constraint(expr=model.A_w + model.A_c + model.A_b <= model.L)
    model.wheat_balance = Constraint(expr=model.Y_w * model.A_w + model.W_p - model.W_s >= model.W_min)
    model.corn_balance = Constraint(expr=model.Y_c * model.A_c + model.C_p - model.C_s >= model.C_min)
    model.sugar_beet_quota = Constraint(expr=model.B_s <= model.Q_b)
    model.sugar_beet_balance = Constraint(expr=model.Y_b * model.A_b == model.B_s + model.B_e)

    # Solve the optimization problem
    solver = SolverFactory('glpk')  # You can use any available solver
    results = solver.solve(model)

    # Print the results
    print("Acres of wheat:", model.A_w.value)
    print("Acres of corn:", model.A_c.value)
    print("Acres of sugar beets:", model.A_b.value)
    print("Tons of wheat sold:", model.W_s.value)
    print("Tons of wheat purchased:", model.W_p.value)
    print("Tons of corn sold:", model.C_s.value)
    print("Tons of corn purchased:", model.C_p.value)
    print("Tons of sugar beets sold within the quota:", model.B_s.value)
    print("Tons of sugar beets sold exceeding the quota:", model.B_e.value)
    print("Total cost:", model.objective())
    ```
```

## 4. Input Problem Data and Test Model Code

```
# Installing pyomo nd solver
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
!pip install glpk
```

Show hidden output

```
from pyomo.environ import *

# Create a Concrete Model
model = ConcreteModel()

# Define parameters
model.W_min = Param(initialize=200)  # minimum requirement of wheat for cattle feed (T)
model.C_min = Param(initialize=240)  # minimum requirement of corn for cattle feed (T)
model.P_ws = Param(initialize=170)  # selling price of wheat ($/T)
model.P_cs = Param(initialize=150)  # selling price of corn ($/T)
model.P_wp = Param(initialize=238)  # purchase price of wheat ($/T)
model.P_cp = Param(initialize=210)  # purchase price of corn ($/T)
model.P_bs = Param(initialize=36)  # selling price of sugar beets within the quota ($/T)
model.P_be = Param(initialize=10)  # selling price of sugar beets exceeding the quota ($/T)
model.Q_b = Param(initialize=6000)  # quota for sugar beet production (T)
model.Y_w = Param(initialize=2.5)  # yield of wheat per acre (T/acre)
model.Y_c = Param(initialize=3)  # yield of corn per acre (T/acre)
model.Y_b = Param(initialize=20)  # yield of sugar beets per acre (T/acre)
model.C_w = Param(initialize=150)  # cost of planting an acre of wheat ($/acre)
model.C_c = Param(initialize=230)  # cost of planting an acre of corn ($/acre)
model.C_b = Param(initialize=260)  # cost of planting an acre of sugar beets ($/acre)
model.L = Param(initialize=500)  # total available land (acres)
```
```

```
# Define decision variables
model.A_w = Var(domain=NonNegativeReals)  # acres of land devoted to wheat
model.A_c = Var(domain=NonNegativeReals)  # acres of land devoted to corn
model.A_b = Var(domain=NonNegativeReals)  # acres of land devoted to sugar beets
model.W_s = Var(domain=NonNegativeReals)  # tons of wheat sold
model.W_p = Var(domain=NonNegativeReals)  # tons of wheat purchased
model.C_s = Var(domain=NonNegativeReals)  # tons of corn sold
model.C_p = Var(domain=NonNegativeReals)  # tons of corn purchased
model.B_s = Var(domain=NonNegativeReals)  # tons of sugar beets sold within the quota
model.B_e = Var(domain=NonNegativeReals)  # tons of sugar beets sold exceeding the quota

# Define objective function
def total_cost(model):
    return (model.C_w * model.A_w + model.C_c * model.A_c + model.C_b * model.A_b +
            model.P_wp * model.W_p + model.P_cp * model.C_p -
            model.P_ws * model.W_s - model.P_cs * model.C_s -
            model.P_bs * model.B_s - model.P_be * model.B_e)

model.objective = Objective(rule=total_cost, sense=minimize)

# Define constraints
model.land_constraint = Constraint(expr=model.A_w + model.A_c + model.A_b <= model.L)
model.wheat_balance = Constraint(expr=model.Y_w * model.A_w + model.W_p - model.W_s >= model.W_min)
model.corn_balance = Constraint(expr=model.Y_c * model.A_c + model.C_p - model.C_s >= model.C_min)
model.sugar_beet_quota = Constraint(expr=model.B_s <= model.Q_b)
model.sugar_beet_balance = Constraint(expr=model.Y_b * model.A_b == model.B_s + model.B_e)

# Solve the optimization problem
solver = SolverFactory('glpk')  # You can use any available solver
results = solver.solve(model)

# Print the results
print("Acres of wheat:", model.A_w.value)
print("Acres of corn:", model.A_c.value)
print("Acres of sugar beets:", model.A_b.value)
print("Tons of wheat sold:", model.W_s.value)
print("Tons of wheat purchased:", model.W_p.value)
print("Tons of corn sold:", model.C_s.value)
print("Tons of corn purchased:", model.C_p.value)
print("Tons of sugar beets sold within the quota:", model.B_s.value)
print("Tons of sugar beets sold exceeding the quota:", model.B_e.value)
print("Total cost:", model.objective())
```

```
Acres of wheat: 120.0
Acres of corn: 80.0
Acres of sugar beets: 300.0
Tons of wheat sold: 99.9999999999999
Tons of wheat purchased: 0.0
Tons of corn sold: 0.0
Tons of corn purchased: 0.0
Tons of sugar beets sold within the quota: 6000.0
Tons of sugar beets sold exceeding the quota: 0.0
Total cost: -118599.99999999999
```

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