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)
```

```
Let's formulate this problem as a mathematical optimization model. We'll define the parameters, decision variables, obje

Parameters:
- Total land available: 500 acres
- Minimum wheat requirement for cattle feed: 200 tons
- Minimum corn requirement for cattle feed: 240 tons
- Selling price of wheat: $170/ton
- Selling price of corn: $150/ton
- Purchase price of wheat: $238/ton (40% more than selling price)
- Purchase price of corn: $210/ton (40% more than selling price)
- Selling price of sugar beets within quota: $36/ton
- Selling price of sugar beets exceeding quota: $10/ton
- Sugar beet quota: 6000 tons
- Yield of wheat: 2.5 tons/acre
- Yield of corn: 3 tons/acre
- Yield of sugar beets: 20 tons/acre
- Planting cost of wheat: $150/acre
- Planting cost of corn: $230/acre
- Planting cost of sugar beets: $260/acre

Decision Variables:
- W: Acres of land devoted to wheat
- C: Acres of land devoted to corn
- S: Acres of land devoted to sugar beets
- WB: Tons of wheat bought from wholesaler
- CB: Tons of corn bought from wholesaler
- SE: Tons of sugar beets sold exceeding quota

Objective Function:
Minimize: 150W + 230C + 260S + 238WB + 210CB - 170(2.5W - 200) - 150(3C - 240) - 36(20S) + 26SE

Constraints:
1. Land constraint: W + C + S <= 500
2. Wheat requirement: 2.5W + WB >= 200
3. Corn requirement: 3C + CB >= 240
4. Sugar beet quota: 20S - SE <= 6000
5. Non-negativity: W, C, S, WB, CB, SE >= 0

In the objective function, the positive terms represent costs (planting costs and purchase costs), while the negative te

The land constraint ensures that the total land used for the three crops does not exceed 500 acres. The wheat and corn r
```

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)
```

~~~~
# Define the decision variables
model.W = Var(domain=NonNegativeReals)  # Acres of land devoted to wheat
model.C = Var(domain=NonNegativeReals)  # Acres of land devoted to corn
model.S = Var(domain=NonNegativeReals)  # Acres of land devoted to sugar beets
model.WB = Var(domain=NonNegativeReals)  # Tons of wheat bought from wholesaler
model.CB = Var(domain=NonNegativeReals)  # Tons of corn bought from wholesaler
model.SE = Var(domain=NonNegativeReals)  # Tons of sugar beets sold exceeding quota

# Define the objective function
model.obj = Objective(expr=model.wheat_planting_cost * model.W +
                           model.corn_planting_cost * model.C +
                           model.sugar_beets_planting_cost * model.S +
                           model.wheat_purchase_price * model.WB +
                           model.corn_purchase_price * model.CB -
                           model.wheat_sell_price * (model.wheat_yield * model.W - model.min_wheat_req) -
                           model.corn_sell_price * (model.corn_yield * model.C - model.min_corn_req) -
                           model.sugar_beets_sell_price_quota * (model.sugar_beets_yield * model.S) +
                           (model.sugar_beets_sell_price_quota - model.sugar_beets_sell_price_exceed) * model.SE,
                      sense=minimize)

# Define the constraints
model.land_constraint = Constraint(expr=model.W + model.C + model.S <= model.land_available)
model.wheat_requirement = Constraint(expr=model.wheat_yield * model.W + model.WB >= model.min_wheat_req)
model.corn_requirement = Constraint(expr=model.corn_yield * model.C + model.CB >= model.min_corn_req)
model.sugar_beets_quota = Constraint(expr=model.sugar_beets_yield * model.S - model.SE <= model.sugar_beets_quota)

# Solve the optimization problem
SolverFactory('glpk').solve(model)

# Print the optimal solution
print("Optimal Solution:")
print(f"Acres of land devoted to wheat: {model.W.value:.2f}")
print(f"Acres of land devoted to corn: {model.C.value:.2f}")
print(f"Acres of land devoted to sugar beets: {model.S.value:.2f}")
print(f"Tons of wheat bought from wholesaler: {model.WB.value:.2f}")
print(f"Tons of corn bought from wholesaler: {model.CB.value:.2f}")
print(f"Tons of sugar beets sold exceeding quota: {model.SE.value:.2f}")
print(f"Optimal objective value: {model.obj.expr():.2f}")
```

This code defines the optimization problem using the Pyomo modeling framework. The parameters, decision variables, obj

The code uses the GLPK solver to solve the optimization problem. After solving, it prints the optimal values of the de

Note: Make sure you have Pyomo and GLPK installed to run this code. You can install them using pip:

```
pip install pyomo
~~~~

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 the model
model = ConcreteModel()

# Define the parameters
model.land_available = Param(initialize=500)
model.min_wheat_req = Param(initialize=200)
model.min_corn_req = Param(initialize=240)
model.wheat_sell_price = Param(initialize=170)
model.corn_sell_price = Param(initialize=150)
model.wheat_purchase_price = Param(initialize=238)
model.corn_purchase_price = Param(initialize=210)
model.sugar_beets_sell_price_quota = Param(initialize=36)
model.sugar_beets_sell_price_exceed = Param(initialize=10)
model.sugar_beets_quota = Param(initialize=6000)
model.wheat_yield = Param(initialize=2.5)
model.corn_yield = Param(initialize=3)
model.sugar_beets_yield = Param(initialize=20)
model.wheat_planting_cost = Param(initialize=150)
model.corn_planting_cost = Param(initialize=230)
model.sugar_beets_planting_cost = Param(initialize=260)

# Define the decision variables
model.W = Var(domain=NonNegativeReals)  # Acres of land devoted to wheat
model.C = Var(domain=NonNegativeReals)  # Acres of land devoted to corn
```

```
model.C = Var(domain=NonNegativeReals)  # Acres of land devoted to corn
model.S = Var(domain=NonNegativeReals)  # Acres of land devoted to sugar beets
model.WB = Var(domain=NonNegativeReals)  # Tons of wheat bought from wholesaler
model.CB = Var(domain=NonNegativeReals)  # Tons of corn bought from wholesaler
model.SE = Var(domain=NonNegativeReals)  # Tons of sugar beets sold exceeding quota

# Define the objective function
model.obj = Objective(expr=model.wheat_planting_cost * model.W +
                           model.corn_planting_cost * model.C +
                           model.sugar_beets_planting_cost * model.S +
                           model.wheat_purchase_price * model.WB +
                           model.corn_purchase_price * model.CB -
                           model.wheat_sell_price * (model.wheat_yield * model.W - model.min_wheat_req) -
                           model.corn_sell_price * (model.corn_yield * model.C - model.min_corn_req) -
                           model.sugar_beets_sell_price_quota * (model.sugar_beets_yield * model.S) +
                           (model.sugar_beets_sell_price_quota - model.sugar_beets_sell_price_exceed) * model.SE,
                      sense=minimize)

# Define the constraints
model.land_constraint = Constraint(expr=model.W + model.C + model.S <= model.land_available)
model.wheat_requirement = Constraint(expr=model.wheat_yield * model.W + model.WB >= model.min_wheat_req)
model.corn_requirement = Constraint(expr=model.corn_yield * model.C + model.CB >= model.min_corn_req)
model.sugar_beets_quota = Constraint(expr=model.sugar_beets_yield * model.S - model.SE <= model.sugar_beets_quota)

# Solve the optimization problem
SolverFactory('glpk').solve(model)

# Print the optimal solution
print("Optimal Solution:")
print(f"Acres of land devoted to wheat: {model.W.value:.2f}")
print(f"Acres of land devoted to corn: {model.C.value:.2f}")
print(f"Acres of land devoted to sugar beets: {model.S.value:.2f}")
print(f"Tons of wheat bought from wholesaler: {model.WB.value:.2f}")
print(f"Tons of corn bought from wholesaler: {model.CB.value:.2f}")
print(f"Tons of sugar beets sold exceeding quota: {model.SE.value:.2f}")
print(f"Optimal objective value: {model.obj.expr():.2f}")
```

```
WARNING:pyomo.core:Implicitly replacing the Component attribute sugar_beets_quota (type=<class 'pyomo.core.base.param.Sc
This is usually indicative of a modelling error.
To avoid this warning, use block.del_component() and block.add_component().
Optimal Solution:
Acres of land devoted to wheat: 120.00
Acres of land devoted to corn: 80.00
Acres of land devoted to sugar beets: 300.00
Tons of wheat bought from wholesaler: 0.00
Tons of corn bought from wholesaler: 0.00
Tons of sugar beets sold exceeding quota: 0.00
Optimal objective value: -118600.00
```

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