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/IP/IP4.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:
 Imagine you own a small shop, and you're trying to manage your employees' working hours and sales targets efficiently.
You have three kinds of goals overall: sales goals, employee hours goals and overtime goals.

Sales Goals:
You have a target of selling 5500 units.
But sometimes, you might sell fewer (negative deviation) or more (positive deviation) than this target.
On average, your full-time employees will sell 5 records per hour and your part-time employees will sell 2 records per h

Employee Hours:
You have both full-time and part-time employees.
Your full-time employees are expected to work a total of 800 hours, while part-time employees are expected to work 320 h
Similarly, like sales, there could be deviations in these working hours.

Overtime:
Additionally, your employees might end up working overtime.
You have a target of 100 hours of overtime for full-time employees.
You are not concerned about limiting the overtime of part-time employees.
Again, you might deviate from this goal.

Each of these deviations has a different priority for you.
P1 to P4 are priority factors for each deviation.
Your most important goal is to achieve your sales goal.
Second to that is the goal to not exceed 100 overtime hours for full-time employees.
Then, you next priority is to fully utilize your full-time and part time employees.
Your last goal with the lowest priority, you want to keep general overtime hours for both kinds of employees as low as p
In addition to the general priorities of you goals, you are twice as concerned about fully utilizing the full-time emplo
Finally, overtime hours of part-time employees have a relative cost of three times that of full-time employees.

Now, the goal here is to minimize the overall impact of these deviations, considering their priorities.
This means finding the best balance between sales targets, employee hours, and overtime, while keeping in mind which dev
```

```
!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)
```

```
Here's the mathematical optimization model for this problem:

Parameters:
- ST: Sales target (5500 units)
- FT: Full-time employee hours target (800 hours)
- PT: Part-time employee hours target (320 hours)
- OT: Overtime target for full-time employees (100 hours)
- P1, P2, P3, P4: Priority factors for each deviation

Decision Variables:
- S: Actual sales (units)
- F: Actual full-time employee hours
- P: Actual part-time employee hours
- OF: Actual overtime hours for full-time employees
- OP: Actual overtime hours for part-time employees
- DS+, DS-: Positive and negative deviations from sales target
- DF+, DF-: Positive and negative deviations from full-time employee hours target
- DP+, DP-: Positive and negative deviations from part-time employee hours target
- DOF+, DOF-: Positive and negative deviations from overtime target for full-time employees

Objective Function:
Minimize: P1 * (DS+ + DS-) + P2 * (DOF+ + DOF-) + P3 * (2 * (DF+ + DF-) + (DP+ + DP-)) + P4 * (OF + 3 * OP)

Constraints:
1. Sales constraint:
   S = 5 * F + 2 * P
   S + DS- - DS+ = ST

2. Full-time employee hours constraint:
   F + DF- - DF+ = FT

3. Part-time employee hours constraint:
   P + DP- - DP+ = PT

4. Overtime constraint for full-time employees:
   OF + DOF- - DOF+ = OT

5. Non-negativity constraints:
   S, F, P, OF, OP, DS+, DS-, DF+, DF-, DP+, DP-, DOF+, DOF- ≥ 0

In this model, the objective function minimizes the weighted sum of deviations from the targets, considering their prior
```

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.P = Var(within=NonNegativeReals)    # Actual part-time employee hours
model.OF = Var(within=NonNegativeReals)   # Actual overtime hours for full-time employees
model.OP = Var(within=NonNegativeReals)   # Actual overtime hours for part-time employees
model.DSplus = Var(within=NonNegativeReals)   # Positive deviation from sales target
model.DSminus = Var(within=NonNegativeReals)  # Negative deviation from sales target
model.DFplus = Var(within=NonNegativeReals)   # Positive deviation from full-time employee hours target
model.DFminus = Var(within=NonNegativeReals)  # Negative deviation from full-time employee hours target
model.DPplus = Var(within=NonNegativeReals)   # Positive deviation from part-time employee hours target
model.DPminus = Var(within=NonNegativeReals)  # Negative deviation from part-time employee hours target
model.DOFplus = Var(within=NonNegativeReals)  # Positive deviation from overtime target for full-time employees
model.DOFminus = Var(within=NonNegativeReals) # Negative deviation from overtime target for full-time employees

# Define objective function
def obj_rule(model):
    return (model.P1 * (model.DSplus + model.DSminus) +
            model.P2 * (model.DOFplus + model.DOFminus) +
            model.P3 * (2 * (model.DFplus + model.DFminus) + (model.DPplus + model.DPminus)) +
            model.P4 * (model.OF + 3 * model.OP))
model.OBJ = Objective(rule=obj_rule, sense=minimize)

# Define constraints
def sales_constraint(model):
    return model.S == 5 * model.F + 2 * model.P
model.SalesConstraint = Constraint(rule=sales_constraint)

def sales_deviation_constraint(model):
    return model.S + model.DSminus - model.DSplus == model.ST
model.SalesDeviationConstraint = Constraint(rule=sales_deviation_constraint)

def fulltime_hours_constraint(model):
    return model.F + model.DFminus - model.DFplus == model.FT
model.FulltimeHoursConstraint = Constraint(rule=fulltime_hours_constraint)

def parttime_hours_constraint(model):
    return model.P + model.DPminus - model.DPplus == model.PT
model.ParttimeHoursConstraint = Constraint(rule=parttime_hours_constraint)

def overtime_constraint(model):
    return model.OF + model.DOFminus - model.DOFplus == model.OT
model.OvertimeConstraint = Constraint(rule=overtime_constraint)

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

# Print the results
print(f"Actual Sales: {model.S.value}")
print(f"Full-time Employee Hours: {model.F.value}")
print(f"Part-time Employee Hours: {model.P.value}")
print(f"Overtime Hours for Full-time Employees: {model.OF.value}")
print(f"Overtime Hours for Part-time Employees: {model.OP.value}")
```

In this code, sample data is used for the parameters:
- Sales target (ST) is set to 5500 units
- Full-time employee hours target (FT) is set to 800 hours
- Part-time employee hours target (PT) is set to 320 hours
- Overtime target for full-time employees (OT) is set to 100 hours
```

## 4. Input Problem Data and Test Model Code

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

Show hidden output
```

```
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

# Define parameters (using sample data)
model.ST = Param(initialize=5500)  # Sales target
model.FT = Param(initialize=800)   # Full-time employee hours target
model.PT = Param(initialize=320)   # Part-time employee hours target
model.OT = Param(initialize=100)   # Overtime target for full-time employees
model.P1 = Param(initialize=4)     # Priority factor for sales deviation
model.P2 = Param(initialize=3)     # Priority factor for overtime deviation
model.P3 = Param(initialize=2)     # Priority factor for employee hours deviation
model.P4 = Param(initialize=1)     # Priority factor for total overtime

# Define decision variables
model.S = Var(within=NonNegativeReals)    # Actual sales
model.F = Var(within=NonNegativeReals)    # Actual full-time employee hours
model.P = Var(within=NonNegativeReals)    # Actual part-time employee hours
model.OF = Var(within=NonNegativeReals)   # Actual overtime hours for full-time employees
model.OP = Var(within=NonNegativeReals)   # Actual overtime hours for part-time employees
model.DSplus = Var(within=NonNegativeReals)   # Positive deviation from sales target
model.DSminus = Var(within=NonNegativeReals)  # Negative deviation from sales target
model.DFplus = Var(within=NonNegativeReals)   # Positive deviation from full-time employee hours target
model.DFminus = Var(within=NonNegativeReals)  # Negative deviation from full-time employee hours target
model.DPplus = Var(within=NonNegativeReals)   # Positive deviation from part-time employee hours target
model.DPminus = Var(within=NonNegativeReals)  # Negative deviation from part-time employee hours target
model.DOFplus = Var(within=NonNegativeReals)  # Positive deviation from overtime target for full-time employees
model.DOFminus = Var(within=NonNegativeReals) # Negative deviation from overtime target for full-time employees

# Define objective function
def obj_rule(model):
    return (model.P1 * (model.DSplus + model.DSminus) +
            model.P2 * (model.DOFplus + model.DOFminus) +
            model.P3 * (2 * (model.DFplus + model.DFminus) + (model.DPplus + model.DPminus)) +
            model.P4 * (model.OF + 3 * model.OP))
model.OBJ = Objective(rule=obj_rule, sense=minimize)

# Define constraints
def sales_constraint(model):
    return model.S == 5 * model.F + 2 * model.P
model.SalesConstraint = Constraint(rule=sales_constraint)

def sales_deviation_constraint(model):
    return model.S + model.DSminus - model.DSplus == model.ST
model.SalesDeviationConstraint = Constraint(rule=sales_deviation_constraint)

def fulltime_hours_constraint(model):
    return model.F + model.DFminus - model.DFplus == model.FT
model.FulltimeHoursConstraint = Constraint(rule=fulltime_hours_constraint)

def parttime_hours_constraint(model):
    return model.P + model.DPminus - model.DPplus == model.PT
model.ParttimeHoursConstraint = Constraint(rule=parttime_hours_constraint)

def overtime_constraint(model):
    return model.OF + model.DOFminus - model.DOFplus == model.OT
model.OvertimeConstraint = Constraint(rule=overtime_constraint)

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

# Print the results
print(f"Actual Sales: {model.S.value}")
print(f"Full-time Employee Hours: {model.F.value}")
print(f"Part-time Employee Hours: {model.P.value}")
print(f"Overtime Hours for Full-time Employees: {model.OF.value}")
print(f"Overtime Hours for Part-time Employees: {model.OP.value}")
```

```
Actual Sales: 5500.0
Full-time Employee Hours: 972.0
Part-time Employee Hours: 320.0
Overtime Hours for Full-time Employees: 100.0
Overtime Hours for Part-time Employees: 0.0
```

```
value(model.OBJ)
```

```
788.0
```

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