Optimizing Staff Scheduling at Iowa City Papa John's

Grace Trane, Arun Ganapathy, Michael Neuhaus and Carrie Cox

Applied Optimization Fall 2023

Introduction

This paper addresses staff scheduling optimization at Papa John's in Iowa City, aiming to meet customer demand, enhance operational efficiency, and improve the work environment. Leveraging real data from Papa John's management, we developed an integer linear programming modacel that integrates shift length, labor hours, and store hours. The project's primary focus is to optimize the scheduling of part-time and full-time staff to ensure a positive customer experience. Drawing from techniques explored in our previous research paper on Airport Staff Scheduling, our project aimed to provide Iowa City Papa John's management with a tool that can be utilized to schedule the most optimal number of workers and have them start their shifts at the right time. The paper discusses both practical and technical challenges encountered during the model development and provides insights into the application of the proposed optimization solution for more effective workforce management at Papa John's.

Problem Description

Papa John's currently faces the challenge of scheduling an inefficient number of employees throughout the week, leading to suboptimal operational efficiencies or high labor costs. The store's management relies on a manual scheduling process, a task claimed by the manager to have always been done by hand. However, this appears problematic when the store faces such high variation in customer demand based on the time of day, day of the week, and season of the year. The process of manually scheduling staff can be tough to always cover the dynamic nature of the industry adequately. This results in potential overstaffing or understaffing during busy periods, employee attrition rates notwithstanding. Papa John's current method not only consumes a decent amount of time and effort by management, but also impacts the overall operational performance, employee satisfaction and customer experience. Finding a way to automate this area as much as possible could positively impact those metrics and free up management's time.

Data Source

Management at Papa John's provided us with a comprehensive dataset encompassing shift schedules, sales, and orders for the week, coupled with general scheduling guidelines. This data pertains to the week of September 3-9 in 2018, with a notable spike in demand on Saturday due to a home Iowa vs Iowa State football game. It's crucial to note that this data predates the

COVID-19 pandemic, as the stores' demand dynamics underwent significant changes afterward. But this still reflects one of several typical patterns of weekly sales that can be seen throughout the year. Refer to *Table 1* in the appendix for this week's net sales. The difference in sales for each day can be shown by noticing that Saturday's sales alone make up 40% of the week's sales. *Table 2* in the appendix shows the original shifts that were provided to us by management. However, due to varying shift lengths, we opted to streamline them for model efficiency, resulting in two-hour shifts as detailed in *Table 3*.

To determine the required staffing levels, we leveraged the week's product order count. *Table 4* delineates the count of products ordered for deliveries ("Del") and inside the restaurant ("IN"). Applying Papa John's rule of thumb—1 inside worker per 15 products per hour and 1 delivery driver per 4 deliveries per hour—we calculated staff requirements. After dividing inside counts by 15 and delivery counts by 4, we aggregated these figures for each new shift, as illustrated in *Table 5*. Given the fluctuating demand, some shifts require only one employee, while others may require up to 50. Management emphasized the need for a minimum of 3 employees at all times, without exceeding an upper limit. *Table 6* outlines a hard minimum of 3 employees for every shift, with specific shift maximums. These caps were incorporated into our shift demand table, culminating in the final employee requirements showcased in *Table 7*. Bolded numbers indicate instances where the minimum and maximum employee caps were enforced. It's worth noting that the 'After' shift from Sunday to Wednesday does not require any employees, as the restaurant remains closed during these periods. This refined table served as the input for our demand model.

Optimization Formulation

The technique employed in this scheduling model is a linear programming optimization model. The model has ties to the nurse scheduling linear program seen in Homework 2 of the course and to the Auckland Airport research paper mentioned above, with modified decision variables and constraints. The model optimizes shift schedules for a specified day. It produces integer optimal solutions without explicitly imposing integrality constraints on the decision variables because it uses integer parameters as input. Thus, the model is solved as a linear program. The full algebraic formulation and Pyomo code are detailed in *Appendix B* and *C,* respectively.

Decision Variables

As seen in *Appendix B, '01. Mathematical Formulation',* there are four decision variables. The decision variables represent the number of staff starting at shift *i,* within the set *n* of shifts 1 – 9. As previously explained in *Table 3,* each shift is two hours in length.

Let ft_i be the number of full-time shift employees starting at shift *i;* these employees work four to five consecutive shifts. Let $pt1_i$ be a part-time employee working one shift, let $pt2_i$ work two consecutive shifts, and let $pt3_i$work three consecutive shifts.

Objective Function

As seen in *Appendix B, '01. Mathematical Formulation',* the objective is to minimize the total number of staff required daily.

Constraints

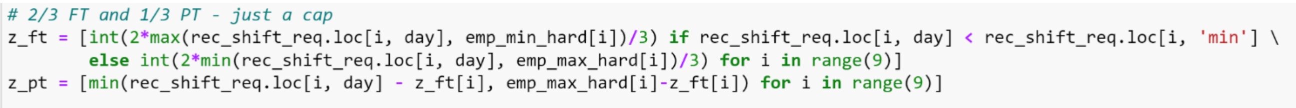
As seen in *Appendix B, '01. Mathematical Formulation',* there are five constraints for the model. Constraint (1) restricts any employee from starting their shift *after* the 5th shift. This is a practical constraint that Papa John's management has. The effect of this constraint means full-time employees starting at the 5th shift must work *five* consecutive shifts, or 10 hours, to properly staff the Papa Johns in the 9th ('After', 2 am - close) shift. This 9th shift only occurs on certain days of the week with high late-night demand, as discussed in our 'Data Source' section. All other full-time shifts are *four* consecutive shifts, or eight hours.

Constraints (2) and (3) set minimum requirements for the total number of full-time and part-time employees at any given shift. We use z_{fti} and z_{pti} as the minimum requirements (demand) for a shift *i* in *n,* for full-time and part-time employees, respectively. These variables are outputted in *Appendix B '02. User Input',* for viewing.

Constraints (4) and (5) use a similar concept for setting a maximum cap for the number of part-time employees currently working a shift *i.* These constraints essentially ensure that there is variability in the model to optimize part-time employee shifts. Of note, by dividing the maximum cap by four for $pt1_i$, the limit is 75% smaller than the other part-time shifts; this places more emphasis on the other part-time employees ($pt2_i$ and $pt3_i$) as we want to minimize an employee only working a single 2-hour shift.

Pyomo Implementation

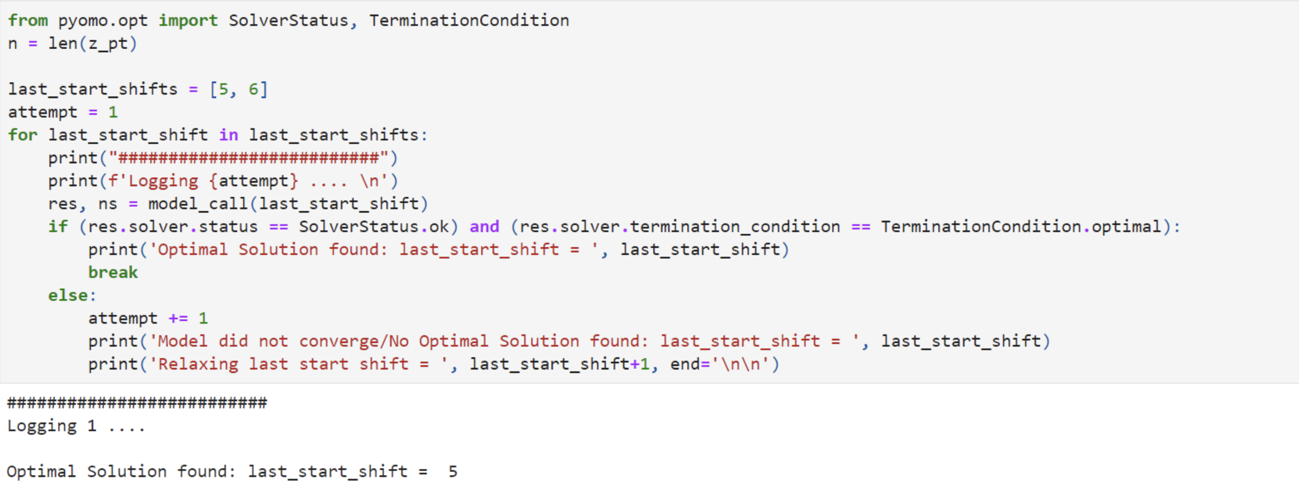
As detailed in *Appendix C,* following a user input of the day of interest, the Pyomo model was executed. A Pyomo command particularly important in our project is seen below in *Figure 1 (Appendix C '02. User Input').* This block of code details the anticipated distribution of full-time and part-time employees as 2/3rds full-time and 1/3rd part-time. Our group wanted to ensure that the model would prioritize the full-time employee. That being said, this code can be flexibly modified to change the emphasis on the different types of employees.



```
# 2/3 FT and 1/3 PT - just a cap
z_ft = [int(2*max(rec_shift_req.loc[i, day], emp_min_hard[i])/3) if rec_shift_req.loc[i, day] < rec_shift_req.loc[i, 'min'] \
        else int(2*min(rec_shift_req.loc[i, day], emp_max_hard[i])/3) for i in range(9)]
z_pt = [min(rec_shift_req.loc[i, day] - z_ft[i], emp_max_hard[i]-z_ft[i]) for i in range(9)]
```

Figure 1. Block of code prioritizing the full-time employees in the model.

An additional block of code important in our project is depicted in *Figure 2* (*Appendix C '05. Run Model'*). For a full-time employee, if the model cannot find a feasible solution to staff the Papa John's in its later hours due to constraint (1), which restricts employees from starting after shift *i*=*5*, the model flexibly relaxes this constraint an extra shift. Essentially, this allows full-time employees to start in shift *i*= 6 and work four consecutive shifts to work through close. Again, shift *i*=9 only occurs Thursdays - Saturdays.



```
from pyomo.opt import SolverStatus, TerminationCondition
n = len(z_pt)

last_start_shifts = [5, 6]
attempt = 1
for last_start_shift in last_start_shifts:
    print("##########################")
    print(f'Logging {attempt} .... \n')
    res, ns = model_call(last_start_shift)
    if (res.solver.status == SolverStatus.ok) and (res.solver.termination_condition == TerminationCondition.optimal):
        print('Optimal Solution found: last_start_shift = ', last_start_shift)
        break
    else:
        attempt += 1
        print('Model did not converge/No Optimal Solution found: last_start_shift = ', last_start_shift)
        print('Relaxing last start shift = ', last_start_shift+1, end='\n\n')
```

```
##########################
Logging 1 ....

Optimal Solution found: last_start_shift =  5
```

Figure 2. Block of code relaxing constraint (1) to shift i=6 upon initial infeasibility.

Results

Our raw results for the number of each employee starting at a shift *i* in *n* are depicted in *Appendix C*. The objective function was calculated at 36 employees in total. The results provided are for Saturday, however, any day can be set as input. *Appendix D* details the code used to create visuals from these results. Visuals include the cumulative counts of all employees on Saturday *(Figure 3),* whether we met, did not meet, or exceeded demand in each shift *(Figure 4),* a breakdown of the different types of employee shifts (*Figure 5)*, and a breakdown of inside employee vs delivery driver counts (*Figure 6)*. Our results collectively show that demand was met throughout the entire day, while minimizing the total number of employees.

As depicted below in *Figure 3*, demand varied throughout the day, resulting in a bell-curve-like distribution on Saturday, where the cumulative counts of employees are those solved from the model. Demand was met seven out of the nine shifts, and exceeded twice, depicted in *Figure 4*.

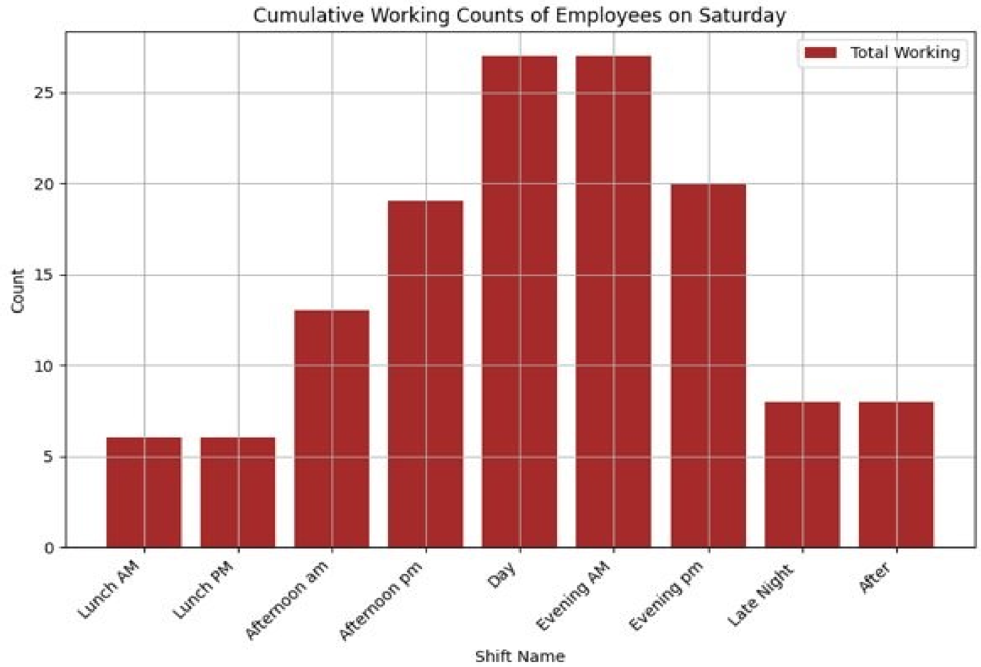


Figure 3. Optimization results – Cumulative Worker Counts (Appendix D, 'Total Staff Working Shift i')

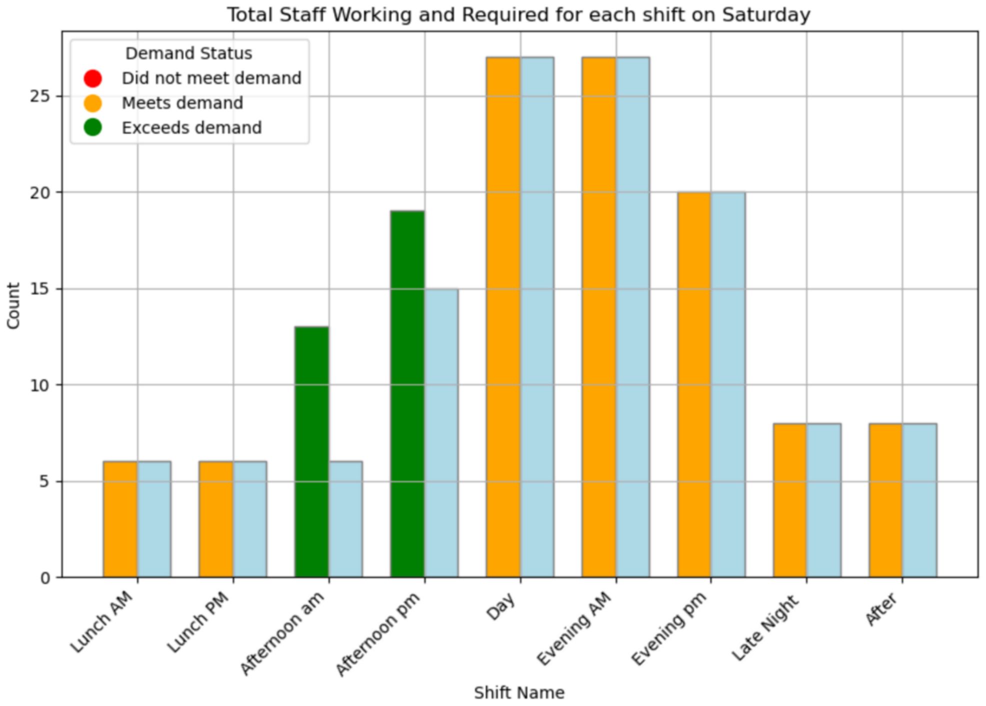


Figure 4. Cumulative demand requirements (Appendix D, 'No of Staff working in Shift i vs Demand')

Examining *Figure 5,* it is apparent that the full-time employee was preferred in the model as previously described in *Figure 1*. To calculate the breakdown of inside employees and delivery drivers in *Figure 6,* we used the original raw data from *Table 4;* we used this data instead of the capped data in *Table 7* to get the most specific ratio between the demand for inside employees and delivery drivers. The results in *Figure 6* show a near consistent split, ensuring a well-balanced store.

Overall, our model created a schedule for all seven days of the week that was optimized to minimize the total number of workers. *Table 8* below details the number of times that demand was either met, not met, or exceeded for each shift throughout the week. Demand was always met, and demand was exactly met over half of the time. 47% of the time, demand was exceeded in order to properly staff later shifts of that particular day. All in all, our team was able to systematically schedule Papa John's staff.

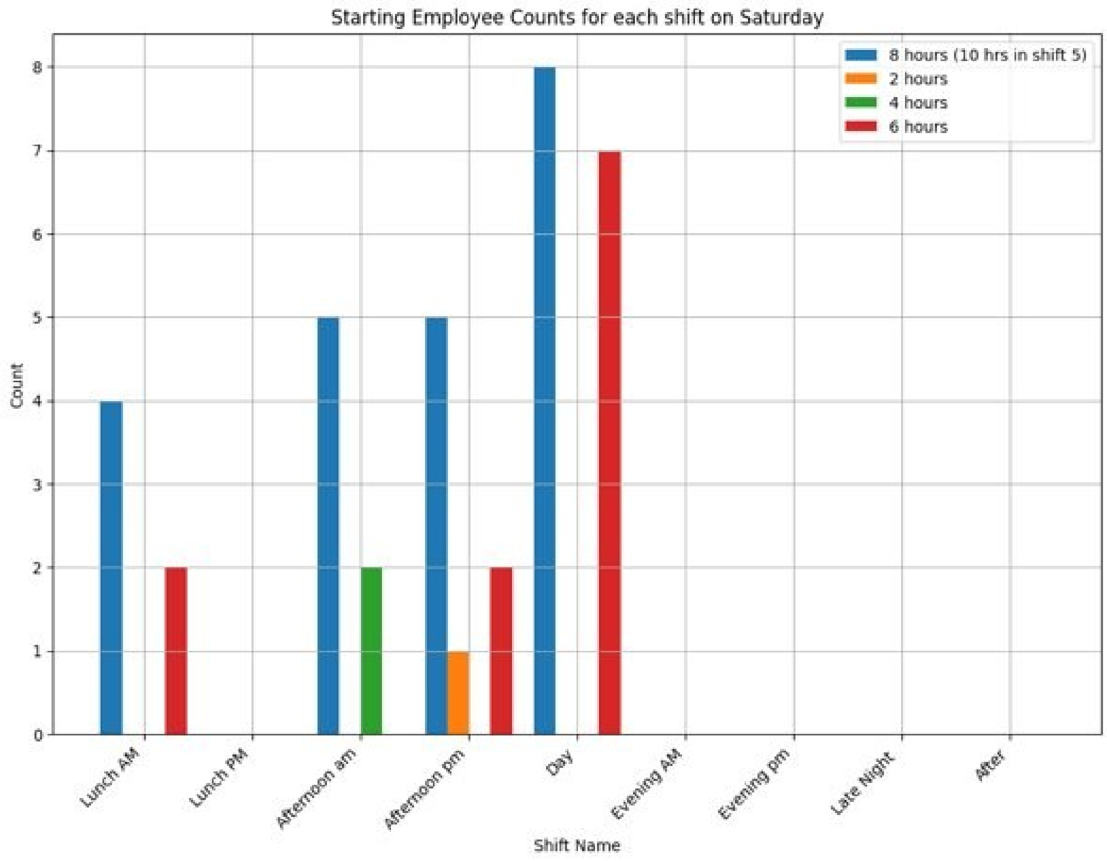


Figure 5. Results separated by shift type (Appendix D, 'No of Staff starting Shift i')

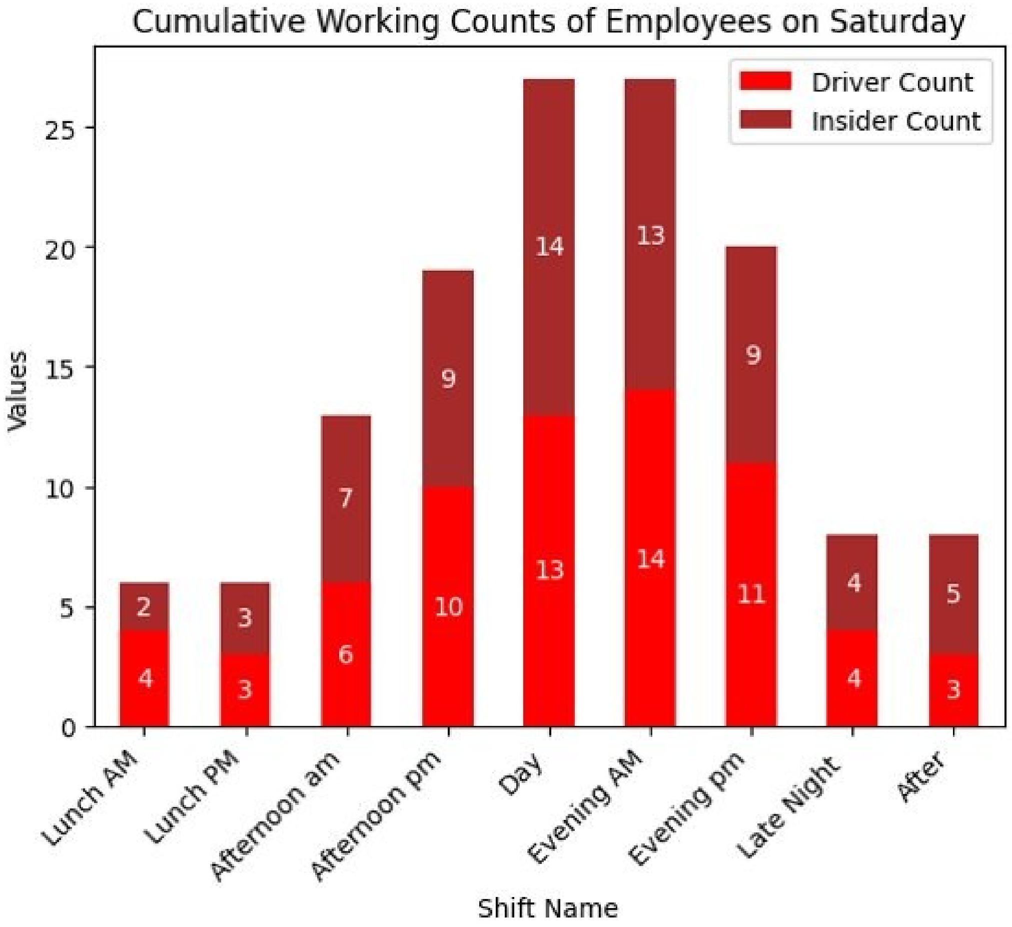
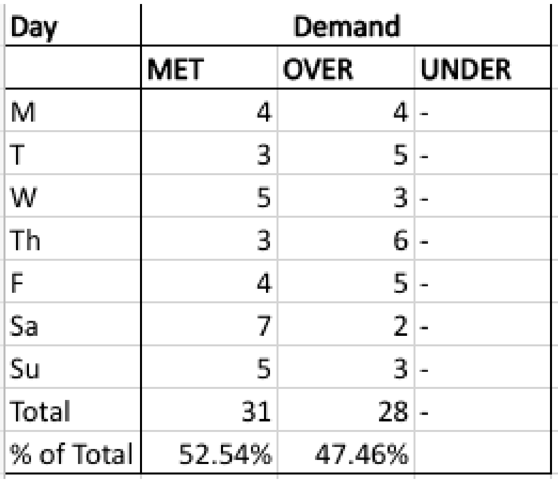


Figure 6. Ratio of Drivers and Inside Employees (Appendix D, 'Ratio visual')

Table 8. Optimization Model Results for the Week



Day	Demand		
	MET	OVER	UNDER
M	4	4	-
T	3	5	-
W	5	3	-
Th	3	6	-
F	4	5	-
Sa	7	2	-
Su	5	3	-
Total	31	28	-
% of Total	52.54%	47.46%	

Challenges

There were several issues that we had to take into consideration when constructing our model. The first set of issues were practical. There are many ‘soft’ constraints that management typically considers in fast-food restaurants. They also need to remain flexible enough to retain employees while also keeping labor costs low. Although we broke down the sales periods into two-hour segments, no employee is scheduled to work that amount of time. Overscheduling during a shift may necessitate that, but priority is given to those who have been clocked in longer out of fairness. Experience of that environment informed are model that at minimum, employees are scheduled four hours.

Another practical issue was the starting shift for employees. General practice is that employees don’t start a shift after a certain time of day. This is in line with the previous issue, but also that a driver employee has less opportunity to make money. This is a morale issue that management always contends with, and this helps guide their scheduling practices. These are the main reasons that most employees are not scheduled after shift 5 in our model (refer to *Table 3*).

The other set of challenges are technical. The more we delved into the data and all of the hard and soft constraints that management utilizes, the more complex our model needed to be. To model all possible constraints accurately and effectively would have taken us a great deal of time, and some implementations were giving us infeasible results. This is why we did not account for further constraints in the model.

A specific challenge with this model is that it was constructed over one specific week of historical, not simulated, data. Going into that week, the store had been in a downtrend of sales, and there was a big rivalry home football game that weekend. Simulated data based off sales projections would have underperformed in this case. This is also a not uncommon scenario throughout the year. These types of swings in demand are difficult to implement in a model.

Our model also had broken down all sales periods into two-hour segments. This is not how management views their scheduling decisions, as it is guided by covering sales periods generally. This would be an obstacle in implementing the model for the business. Also, we only had the number of employees per shift and did not put the constraint to prevent overtime. Management is very strict in not allowing overtime, so they would still need to account for that outside of the model. This industry, big-chain pizza delivery, also operates such that the difference between full-time and part-time employees is a grey area unless you are a managerial-level employee. Overall, we faced many obstacles and ultimately decided on constraints that we could implement successfully given the scope of the project and class material.

Conclusion

Overall, the goal of this project was to solve the issue of staff scheduling at Iowa City Papa John's. We aimed to do this by utilizing an integer linear programming model instead of the manual staffing approach that is currently being used. Having Papa John's management manual create their staff schedules can lead to inadequate staff levels for the high variability in customer demand. We utilized optimization techniques from our research paper on Auckland's airport staffing as well as from our course's homework assignment to build a model. The input data for our model came from 2018 and was used to calculate the optimal staff number for different times and days in a week. Our model results show that it can be used to meet demand while minimizing the total number of employees efficiently.

In the future, Papa John's management can use this model to dynamically change the Excel demand inputs to reflect their forecasted sales for an upcoming week, rather than being based on previous sales. This model also can be modified to prioritize different shift types that are preferred by management. This allows for there to be flexibility in prioritizing shift length in each week's schedule. In conclusion, our model can be used by Iowa City Papa John's in order to increase their operational efficiency and improve experiences for customers and employees.

Appendix A

Data Sources

Table 1: Net Sales from 9/3/18 - 9/9/18

Day	Net Sales	% of Total Weekly Sales
Monday	$2,705.80	6.68%
Tuesday	$3,270.30	8.07%
Wednesday	$3,193.83	7.88%
Thursday	$3,973.62	9.81%
Friday	$7,097.74	17.51%
Saturday	$16,509.44	40.74%
Sunday	$3,774.80	9.31%

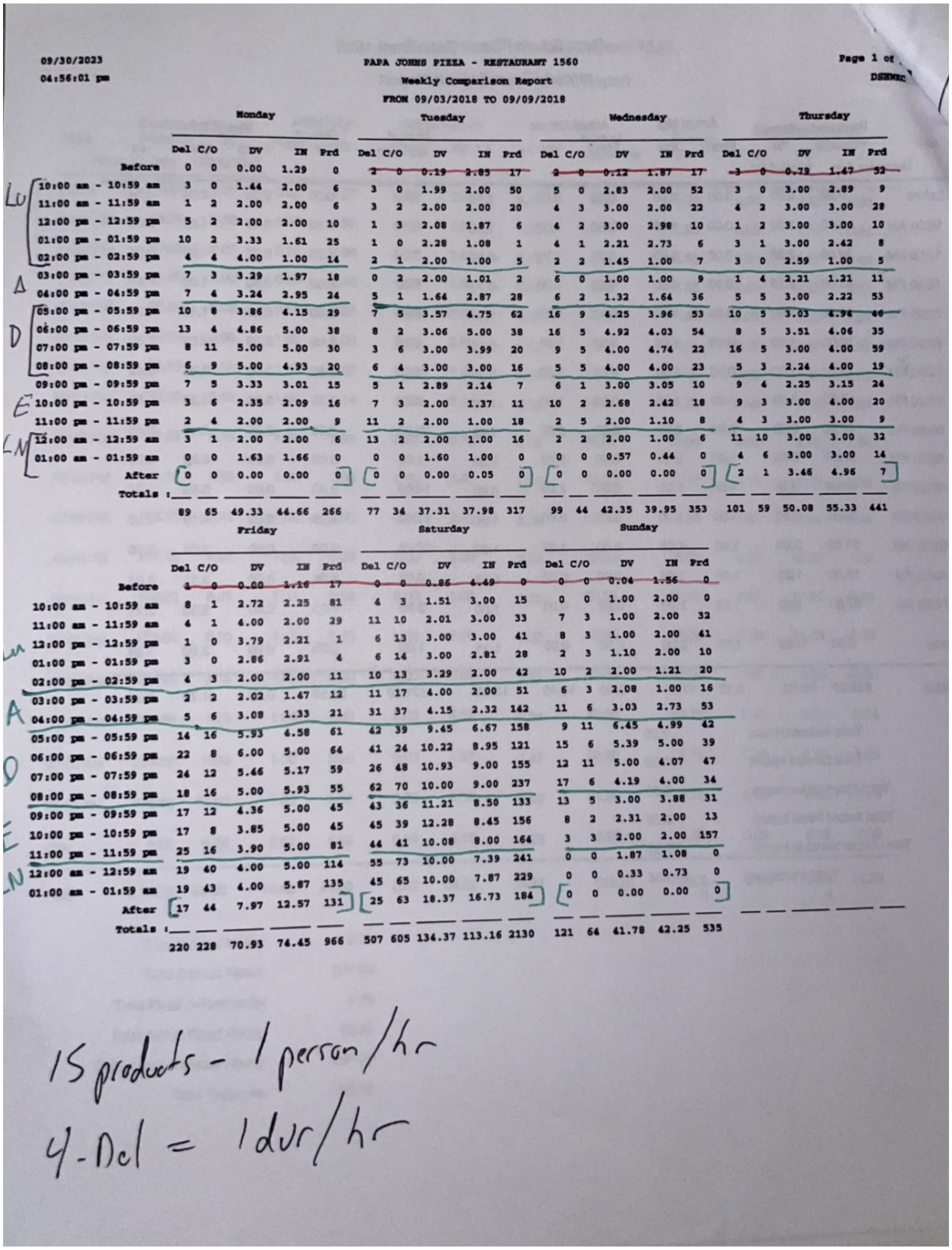
Table 2: Papa John's Original Shifts

Shift Name	Shift Time
Lunch	10:00 am - 2:59 pm
Afternoon	3:00 pm - 4:59 pm
Day	5:00 pm - 8:59 pm
Evening	9:00 pm - 11:59 pm
Late Night	12:00 am - 1:59am
After	2:00 am - close

Table 3: Updated Shifts for Modeling

Shift No.	Shift Name	Shift Time
1	Lunch AM	10:00 am - 11:59 am
2	Lunch PM	12:00 pm - 1:59 pm
3	Afternoon AM	2:00 pm - 3:59 pm
4	Afternoon PM	4:00 pm - 5:59 pm
5	Day	6:00 pm - 7:59 pm
6	Evening AM	8:00 pm - 9:59 pm
7	Evening PM	10:00 pm - 11:59 pm
8	Late Night	12:00 am - 1:59 am
9	After	2:00 am - close

Table 4: Driver and Insider Worker Raw Counts



09/30/2023
04:56:01 pm

PAPA JOHNS PIZZA - RESTAURANT 1560
Weekly Comparison Report
FROM 09/03/2018 TO 09/09/2018

Page 1 of
DSHWKC

		Monday					Tuesday					Wednesday					Thursday				
		Del	C/O	DV	IN	Prd	Del	C/O	DV	IN	Prd	Del	C/O	DV	IN	Prd	Del	C/O	DV	IN	Prd
	Before	0	0	0.00	1.29	0	~~2~~	~~0~~	~~0.19~~	~~2.85~~	~~17~~	~~2~~	~~0~~	~~0.12~~	~~1.87~~	~~17~~	~~3~~	~~0~~	~~0.79~~	~~1.47~~	~~52~~
Lu	10:00 am - 10:59 am	3	0	1.44	2.00	6	3	0	1.99	2.00	50	5	0	2.83	2.00	52	3	0	3.00	2.89	9
	11:00 am - 11:59 am	1	2	2.00	2.00	6	3	2	2.00	2.00	20	4	3	3.00	2.00	22	4	2	3.00	3.00	28
	12:00 pm - 12:59 pm	5	2	2.00	2.00	10	1	3	2.08	1.87	6	4	2	3.00	2.98	10	1	2	3.00	3.00	10
	01:00 pm - 01:59 pm	10	2	3.33	1.61	25	1	0	2.28	1.08	1	4	1	2.21	2.73	6	3	1	3.00	2.42	8
	02:00 pm - 02:59 pm	4	4	4.00	1.00	14	2	1	2.00	1.00	5	2	2	1.45	1.00	7	3	0	2.59	1.00	5
A	03:00 pm - 03:59 pm	7	3	3.29	1.97	18	0	2	2.00	1.01	2	6	0	1.00	1.00	9	1	4	2.21	1.21	11
	04:00 pm - 04:59 pm	7	4	3.24	2.95	24	5	1	1.64	2.87	28	6	2	1.32	1.64	36	5	5	3.00	2.22	53
	05:00 pm - 05:59 pm	10	8	3.86	4.15	29	7	5	3.57	4.75	62	16	9	4.25	3.96	53	10	5	3.03	4.94	46
D	06:00 pm - 06:59 pm	13	4	4.86	5.00	38	8	2	3.06	5.00	38	16	5	4.92	4.03	54	8	5	3.51	4.06	35
	07:00 pm - 07:59 pm	8	11	5.00	5.00	30	3	6	3.00	3.99	20	9	5	4.00	4.74	22	16	5	3.00	4.00	59
	08:00 pm - 08:59 pm	6	9	5.00	4.93	20	6	4	3.00	3.00	16	5	5	4.00	4.00	23	5	3	2.24	4.00	19
	09:00 pm - 09:59 pm	7	5	3.33	3.01	15	5	1	2.89	2.14	7	6	1	3.00	3.05	10	9	4	2.25	3.15	24
E	10:00 pm - 10:59 pm	3	6	2.35	2.09	16	7	3	2.00	1.37	11	10	2	2.68	2.42	18	9	3	3.00	4.00	20
	11:00 pm - 11:59 pm	2	4	2.00	2.00	9	11	2	2.00	1.00	18	2	5	2.00	1.10	8	4	3	3.00	3.00	9
LN	12:00 am - 12:59 am	3	1	2.00	2.00	6	13	2	2.00	1.00	16	2	2	2.00	1.00	6	11	10	3.00	3.00	32
	01:00 am - 01:59 am	0	0	1.63	1.66	0	0	0	1.60	1.00	0	0	0	0.57	0.44	0	4	6	3.00	3.00	14
	After	0	0	0.00	0.00	0	0	0	0.00	0.05	0	0	0	0.00	0.00	0	2	1	3.46	4.96	7
	Totals :	89	65	49.33	44.66	266	77	34	37.31	37.98	317	99	44	42.35	39.95	353	101	59	50.08	55.33	441

		Friday					Saturday					Sunday				
		Del	C/O	DV	IN	Prd	Del	C/O	DV	IN	Prd	Del	C/O	DV	IN	Prd
	Before	~~1~~	~~0~~	~~0.00~~	~~1.26~~	~~17~~	~~0~~	~~0~~	~~0.86~~	~~4.49~~	~~0~~	~~0~~	~~0~~	~~0.04~~	~~1.56~~	~~0~~
	10:00 am - 10:59 am	8	1	2.72	2.25	81	4	3	1.51	3.00	15	0	0	1.00	2.00	0
	11:00 am - 11:59 am	4	1	4.00	2.00	29	11	10	2.01	3.00	33	7	3	1.00	2.00	32
Lu	12:00 pm - 12:59 pm	1	0	3.79	2.21	1	6	13	3.00	3.00	41	8	3	1.00	2.00	41
	01:00 pm - 01:59 pm	3	0	2.86	2.91	4	6	14	3.00	2.81	28	2	3	1.10	2.00	10
	02:00 pm - 02:59 pm	4	3	2.00	2.00	11	10	13	3.29	2.00	42	10	2	2.00	1.21	20
	03:00 pm - 03:59 pm	2	2	2.02	1.47	12	11	17	4.00	2.00	51	6	3	2.08	1.00	16
A	04:00 pm - 04:59 pm	5	6	3.08	1.33	21	31	37	4.15	2.32	142	11	6	3.03	2.73	53
	05:00 pm - 05:59 pm	14	16	5.93	4.58	61	42	39	9.45	6.67	158	9	11	6.45	4.99	42
	06:00 pm - 06:59 pm	22	8	6.00	5.00	64	41	24	10.22	8.95	121	15	6	5.39	5.00	39
D	07:00 pm - 07:59 pm	24	12	5.46	5.17	59	26	48	10.93	9.00	155	12	11	5.00	4.07	47
	08:00 pm - 08:59 pm	18	16	5.00	5.93	55	62	70	10.00	9.00	237	17	6	4.19	4.00	34
	09:00 pm - 09:59 pm	17	12	4.36	5.00	45	43	36	11.21	8.50	133	13	5	3.00	3.88	31
E	10:00 pm - 10:59 pm	17	8	3.85	5.00	45	45	39	12.28	8.45	156	8	2	2.31	2.00	13
	11:00 pm - 11:59 pm	25	16	3.90	5.00	81	44	41	10.08	8.00	164	3	3	2.00	2.00	157
LN	12:00 am - 12:59 am	19	40	4.00	5.00	114	55	73	10.00	7.39	241	0	0	1.87	1.08	0
	01:00 am - 01:59 am	19	43	4.00	5.87	135	45	65	10.00	7.87	229	0	0	0.33	0.73	0
	After	17	44	7.97	12.57	131	25	63	18.37	16.73	184	0	0	0.00	0.00	0
	Totals :	220	228	70.93	74.45	966	507	605	134.37	113.16	2130	121	64	41.78	42.25	535

15 products - 1 person/hr

4-Del = 1 dvr/hr

Table 5: Calculated Demand of Employees (Summarized Raw Counts from Table 4)

Shift No	Shift Name	Shift Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	Lunch AM	10:00 am - 11:59 am	2	8	9	9	12	7	4
2	Lunch PM	12:00 pm - 1:59 pm	7	1	4	3	2	8	6
3	Afternoon AM	2:00 pm - 3:59 pm	5	1	4	3	4	12	7
4	Afternoon PM	4:00 pm - 5:59 pm	8	9	12	11	11	39	12
5	Day	6:00 pm - 7:59 pm	10	7	12	13	20	36	13
6	Evening AM	8:00 pm - 9:59 pm	6	5	5	7	16	51	12
7	Evening PM	10:00 pm - 11:59 pm	3	7	5	6	19	44	15
8	Late Night	12:00 am - 1:59 am	2	5	1	7	27	57	3
9	After	2:00 am - 3:59 am	-	-	-	1	13	19	-

A *-* means store was closed at this time

Table 6: Minimum and Maximum Caps

Shift No	Minimum	Maximum
1	3	6
2	3	6
3	3	6
4	3	15
5	3	27
6	3	27
7	3	20
8	3	8
9	3	8

Table 7: Finalized Demand of Employees

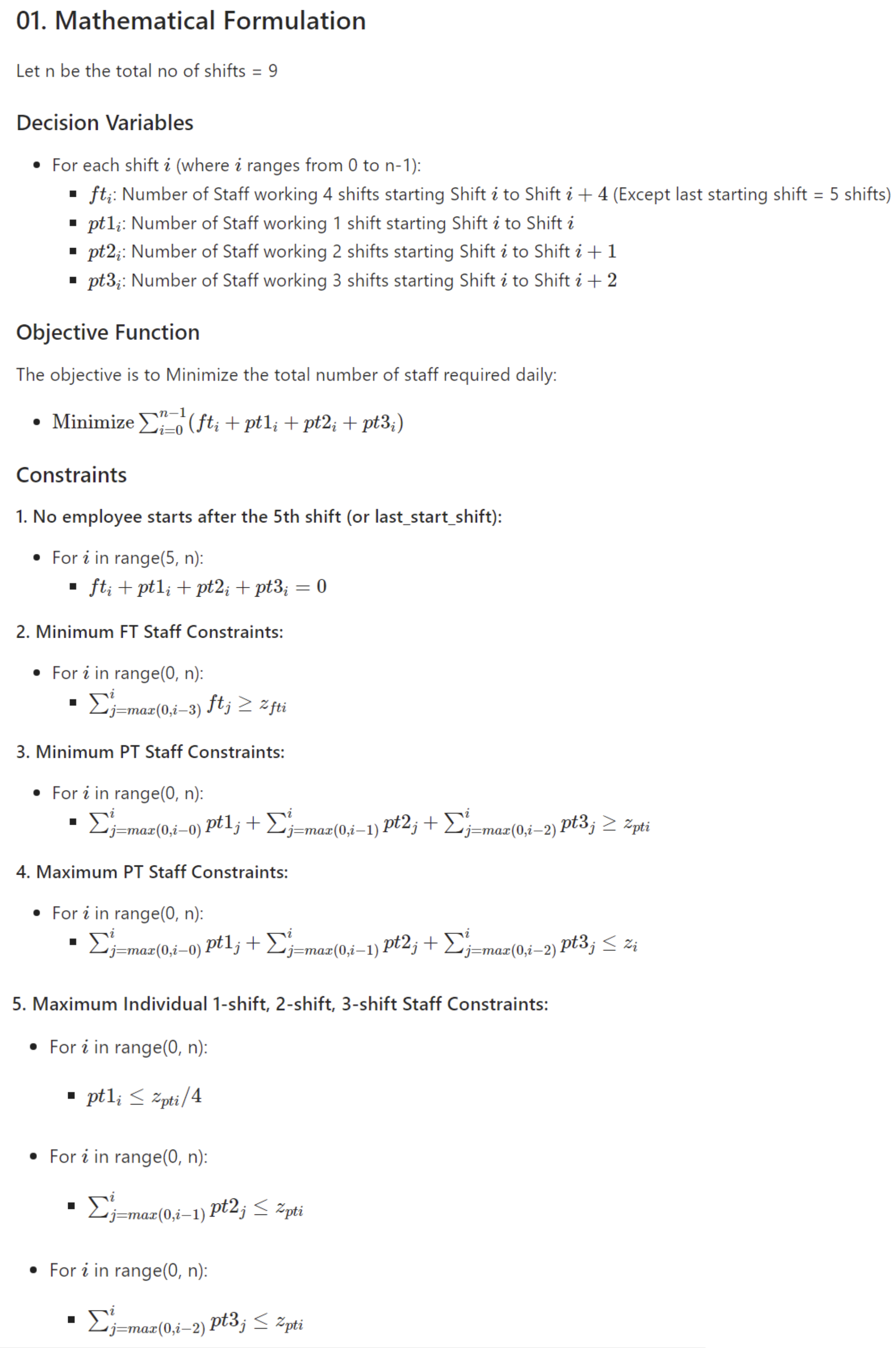
Shift No	Shift Name	Shift Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	Lunch AM	10:00 am - 11:59 am	**3**	**6**	**6**	**6**	**6**	**6**	4
2	Lunch PM	12:00 pm - 1:59 pm	**6**	3	4	3	**3**	**6**	6
3	Afternoon AM	2:00 pm - 3:59 pm	5	**3**	4	3	4	**6**	**6**

4	Afternoon PM	4:00 pm - 5:59 pm	8	9	12	11	11	**15**	12
5	Day	6:00 pm - 7:59 pm	10	7	12	13	20	**27**	13
6	Evening AM	8:00 pm - 9:59 pm	6	5	5	7	16	**27**	12
7	Evening PM	10:00 pm - 11:59 pm	3	7	5	6	19	**20**	15
8	Late Night	12:00 am - 1:59 am	**3**	**5**	**3**	7	**8**	**8**	**3**
9	After	2:00 am - 3:59 am	-	-	-	**3**	**8**	**8**	**-**

A *-* means store was closed at this time

Appendix B

Algebraic Formulation



01. Mathematical Formulation

Let n be the total no of shifts = 9

Decision Variables

- For each shift i (where i ranges from 0 to n-1):
  - ft_i: Number of Staff working 4 shifts starting Shift i to Shift $i+4$ (Except last starting shift = 5 shifts)
  - $pt1_i$: Number of Staff working 1 shift starting Shift i to Shift i
  - $pt2_i$: Number of Staff working 2 shifts starting Shift i to Shift $i+1$
  - $pt3_i$: Number of Staff working 3 shifts starting Shift i to Shift $i+2$

Objective Function

The objective is to Minimize the total number of staff required daily:

- $\text{Minimize} \sum_{i=0}^{n-1}(ft_i + pt1_i + pt2_i + pt3_i)$

Constraints

1. No employee starts after the 5th shift (or last_start_shift):

- For i in range(5, n):
  - $ft_i + pt1_i + pt2_i + pt3_i = 0$

2. Minimum FT Staff Constraints:

- For i in range(0, n):
  - $\sum_{j=max(0,i-3)}^{i} ft_j \geq z_{fti}$

3. Minimum PT Staff Constraints:

- For i in range(0, n):
  - $\sum_{j=max(0,i-0)}^{i} pt1_j + \sum_{j=max(0,i-1)}^{i} pt2_j + \sum_{j=max(0,i-2)}^{i} pt3_j \geq z_{pti}$

4. Maximum PT Staff Constraints:

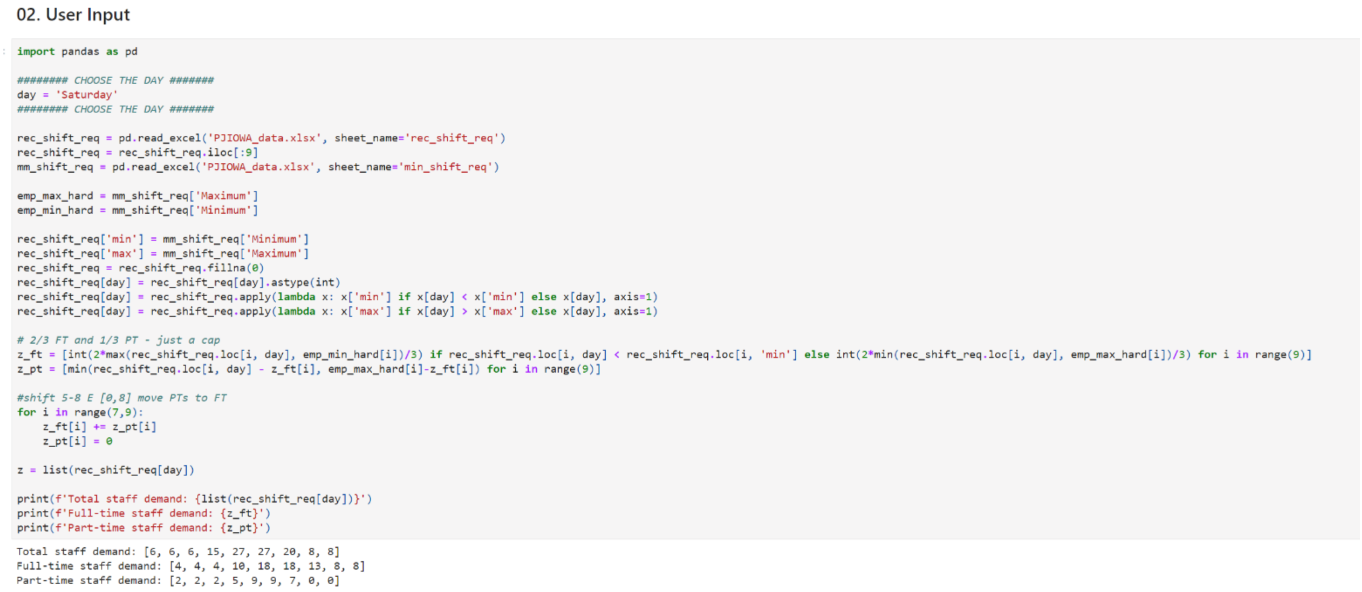
- For i in range(0, n):
  - $\sum_{j=max(0,i-0)}^{i} pt1_j + \sum_{j=max(0,i-1)}^{i} pt2_j + \sum_{j=max(0,i-2)}^{i} pt3_j \leq z_i$

5. Maximum Individual 1-shift, 2-shift, 3-shift Staff Constraints:

- For i in range(0, n):
  - $pt1_i \leq z_{pti}/4$
- For i in range(0, n):
  - $\sum_{j=max(0,i-1)}^{i} pt2_j \leq z_{pti}$
- For i in range(0, n):
  - $\sum_{j=max(0,i-2)}^{i} pt3_j \leq z_{pti}$

Appendix C

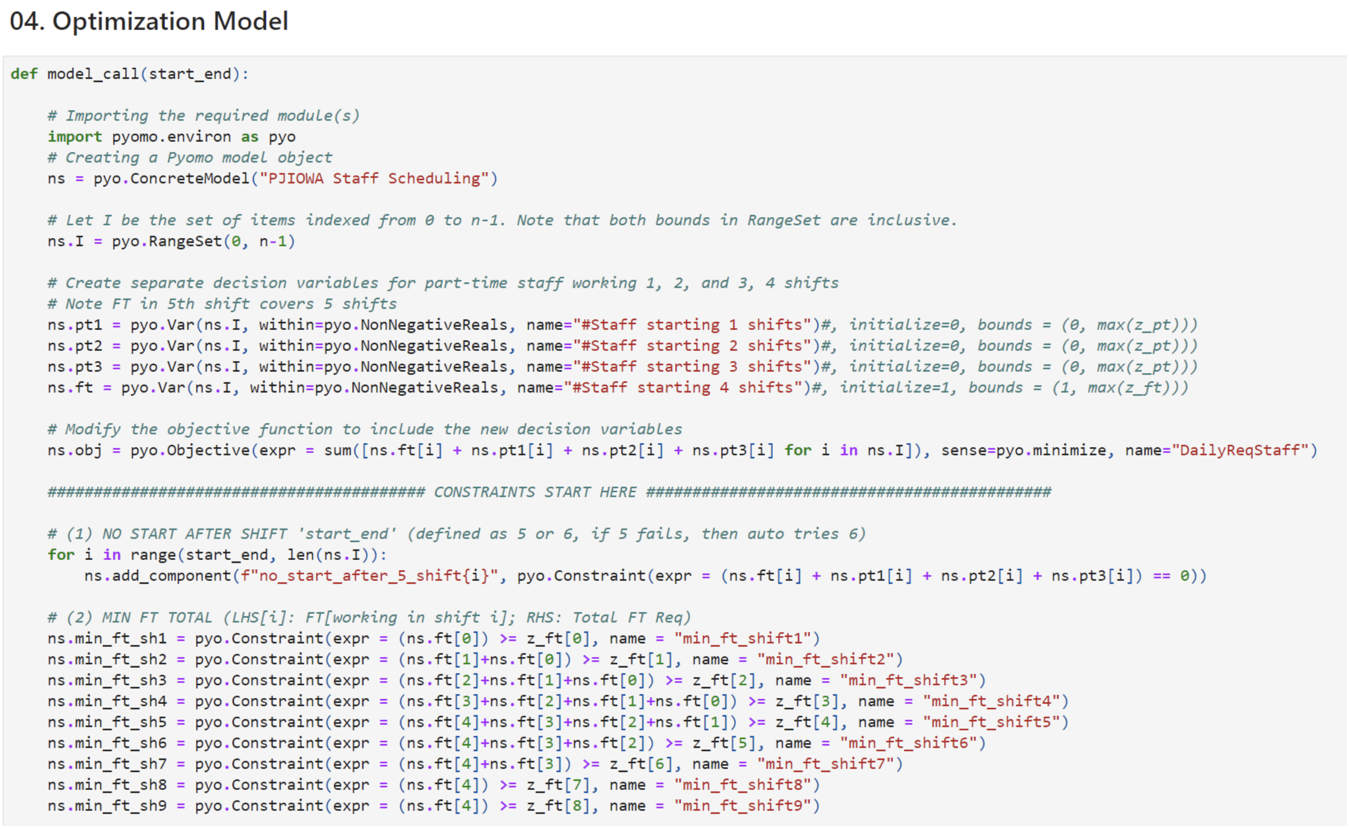
Pyomo implementation



02. User Input

```
import pandas as pd

######## CHOOSE THE DAY #######
day = 'Saturday'
######## CHOOSE THE DAY #######

rec_shift_req = pd.read_excel('PJIOWA_data.xlsx', sheet_name='rec_shift_req')
rec_shift_req = rec_shift_req.iloc[:9]
mm_shift_req = pd.read_excel('PJIOWA_data.xlsx', sheet_name='min_shift_req')

emp_max_hard = mm_shift_req['Maximum']
emp_min_hard = mm_shift_req['Minimum']

rec_shift_req['min'] = mm_shift_req['Minimum']
rec_shift_req['max'] = mm_shift_req['Maximum']
rec_shift_req = rec_shift_req.fillna(0)
rec_shift_req[day] = rec_shift_req[day].astype(int)
rec_shift_req[day] = rec_shift_req.apply(lambda x: x['min'] if x[day] < x['min'] else x[day], axis=1)
rec_shift_req[day] = rec_shift_req.apply(lambda x: x['max'] if x[day] > x['max'] else x[day], axis=1)

# 2/3 FT and 1/3 PT - just a cap
z_ft = [int(2*max(rec_shift_req.loc[i, day], emp_min_hard[i])/3) if rec_shift_req.loc[i, day] < rec_shift_req.loc[i, 'min'] else int(2*min(rec_shift_req.loc[i, day], emp_max_hard[i])/3) for i in range(9)]
z_pt = [min(rec_shift_req.loc[i, day] - z_ft[i], emp_max_hard[i]-z_ft[i]) for i in range(9)]

#shift 5-8 E [0,8] move PTs to FT
for i in range(7,9):
    z_ft[i] += z_pt[i]
    z_pt[i] = 0

z = list(rec_shift_req[day])

print(f'Total staff demand: {list(rec_shift_req[day])}')
print(f'Full-time staff demand: {z_ft}')
print(f'Part-time staff demand: {z_pt}')
```

```
Total staff demand: [6, 6, 6, 15, 27, 27, 20, 8, 8]
Full-time staff demand: [4, 4, 4, 10, 18, 18, 13, 8, 8]
Part-time staff demand: [2, 2, 2, 5, 9, 9, 7, 0, 0]
```



04. Optimization Model

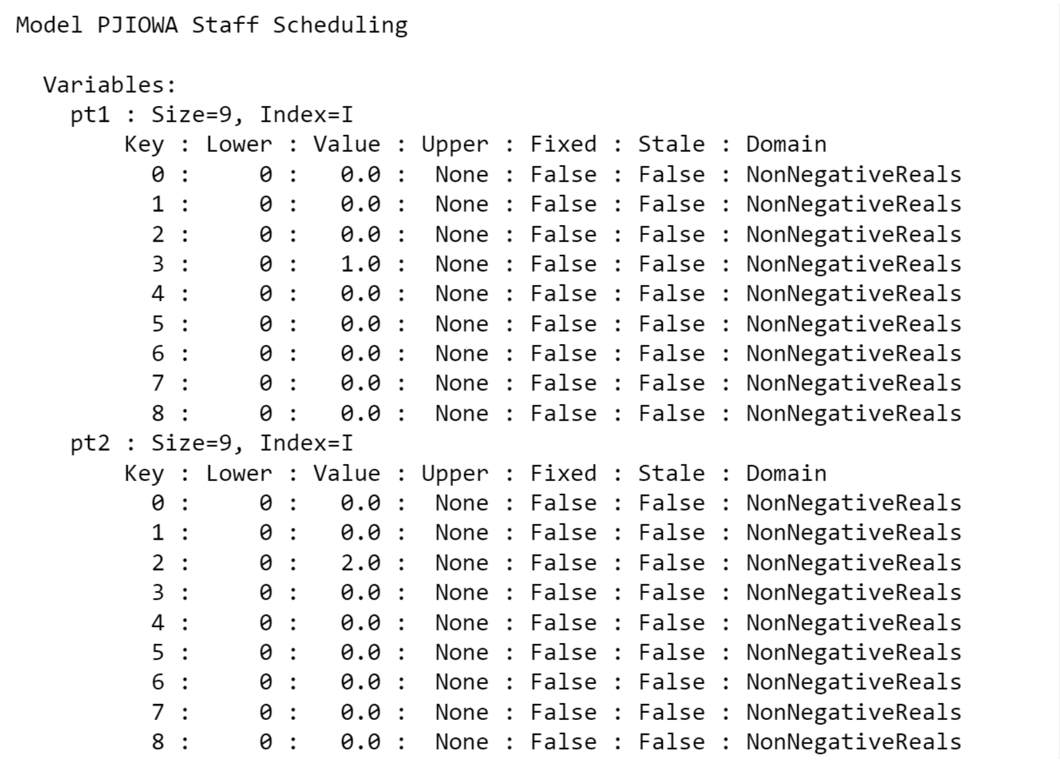
```
def model_call(start_end):

    # Importing the required module(s)
    import pyomo.environ as pyo
    # Creating a Pyomo model object
    ns = pyo.ConcreteModel("PJIOWA Staff Scheduling")

    # Let I be the set of items indexed from 0 to n-1. Note that both bounds in RangeSet are inclusive.
    ns.I = pyo.RangeSet(0, n-1)

    # Create separate decision variables for part-time staff working 1, 2, and 3, 4 shifts
    # Note FT in 5th shift covers 5 shifts
    ns.pt1 = pyo.Var(ns.I, within=pyo.NonNegativeReals, name="#Staff starting 1 shifts")#, initialize=0, bounds = (0, max(z_pt)))
    ns.pt2 = pyo.Var(ns.I, within=pyo.NonNegativeReals, name="#Staff starting 2 shifts")#, initialize=0, bounds = (0, max(z_pt)))
    ns.pt3 = pyo.Var(ns.I, within=pyo.NonNegativeReals, name="#Staff starting 3 shifts")#, initialize=0, bounds = (0, max(z_pt)))
    ns.ft = pyo.Var(ns.I, within=pyo.NonNegativeReals, name="#Staff starting 4 shifts")#, initialize=1, bounds = (1, max(z_ft)))

    # Modify the objective function to include the new decision variables
    ns.obj = pyo.Objective(expr = sum([ns.ft[i] + ns.pt1[i] + ns.pt2[i] + ns.pt3[i] for i in ns.I]), sense=pyo.minimize, name="DailyReqStaff")

    ############################################ CONSTRAINTS START HERE ############################################

    # (1) NO START AFTER SHIFT 'start_end' (defined as 5 or 6, if 5 fails, then auto tries 6)
    for i in range(start_end, len(ns.I)):
        ns.add_component(f"no_start_after_5_shift{i}", pyo.Constraint(expr = (ns.ft[i] + ns.pt1[i] + ns.pt2[i] + ns.pt3[i]) == 0))

    # (2) MIN FT TOTAL (LHS[i]: FT[working in shift i]; RHS: Total FT Req)
    ns.min_ft_sh1 = pyo.Constraint(expr = (ns.ft[0]) >= z_ft[0], name = "min_ft_shift1")
    ns.min_ft_sh2 = pyo.Constraint(expr = (ns.ft[1]+ns.ft[0]) >= z_ft[1], name = "min_ft_shift2")
    ns.min_ft_sh3 = pyo.Constraint(expr = (ns.ft[2]+ns.ft[1]+ns.ft[0]) >= z_ft[2], name = "min_ft_shift3")
    ns.min_ft_sh4 = pyo.Constraint(expr = (ns.ft[3]+ns.ft[2]+ns.ft[1]+ns.ft[0]) >= z_ft[3], name = "min_ft_shift4")
    ns.min_ft_sh5 = pyo.Constraint(expr = (ns.ft[4]+ns.ft[3]+ns.ft[2]+ns.ft[1]) >= z_ft[4], name = "min_ft_shift5")
    ns.min_ft_sh6 = pyo.Constraint(expr = (ns.ft[4]+ns.ft[3]+ns.ft[2]) >= z_ft[5], name = "min_ft_shift6")
    ns.min_ft_sh7 = pyo.Constraint(expr = (ns.ft[4]+ns.ft[3]) >= z_ft[6], name = "min_ft_shift7")
    ns.min_ft_sh8 = pyo.Constraint(expr = (ns.ft[4]) >= z_ft[7], name = "min_ft_shift8")
    ns.min_ft_sh9 = pyo.Constraint(expr = (ns.ft[4]) >= z_ft[8], name = "min_ft_shift9")
    # (3) MIN PT TOTAL (LHS[i]: PT1+PT2+PT3; RHS: Total PT Req)
    for i in ns.I:
        ns.add_component(f"min_pt_shift{i}", pyo.Constraint(expr = (sum(ns.pt1[j] for j in range(max(0, i-0), i+1)) + \
                                                                    sum(ns.pt2[j] for j in range(max(0, i-1), i+1)) + \
                                                                    sum(ns.pt3[j] for j in range(max(0, i-2), i+1))) >= z_pt[i]))

    # (4) MAX PT TOTAL (LHS[i]: PT1+PT2+PT3; RHS: Total Req)
    for i in ns.I:
        ns.add_component(f"max_pt_shift{i}", pyo.Constraint(expr = (sum(ns.pt1[j] for j in range(max(0, i-0), i+1)) + \
                                                                    sum(ns.pt2[j] for j in range(max(0, i-1), i+1)) + \
                                                                    sum(ns.pt3[j] for j in range(max(0, i-2), i+1))) <= z[i]))
```

```
    # (5) THIS PRIORITIZES 3 and 2 shift workers over 1 shift
    # Add MAX constraints for staff working 1 shift
    for i in ns.I:
        ns.add_component(f"max_pt1_shift{i+1}", pyo.Constraint(expr = (ns.pt1[i]) <= int(z_pt[i]/4)))

    # Add MAX constraints for staff working 2 shifts
    for i in ns.I:
        ns.add_component(f"max_pt2_shift{i+2}", pyo.Constraint(expr = (ns.pt2[i] + ns.pt2[i-1] if i > 0 else ns.pt2[i]) <= z_pt[i]))

    # Add MAX constraints for staff working 3 shifts
    for i in ns.I:
        ns.add_component(f"max_pt3_shift{i+3}", pyo.Constraint(expr = sum(ns.pt3[j] for j in range(max(0, i-2), i+1)) <= z_pt[i]))

    ########################################### CONSTRAINTS END HERE ##############################################

    # Solving using pyomo - glpk
    opt = pyo.SolverFactory('glpk')
    res = opt.solve(ns)
    # ns.display()

    return res, ns
```

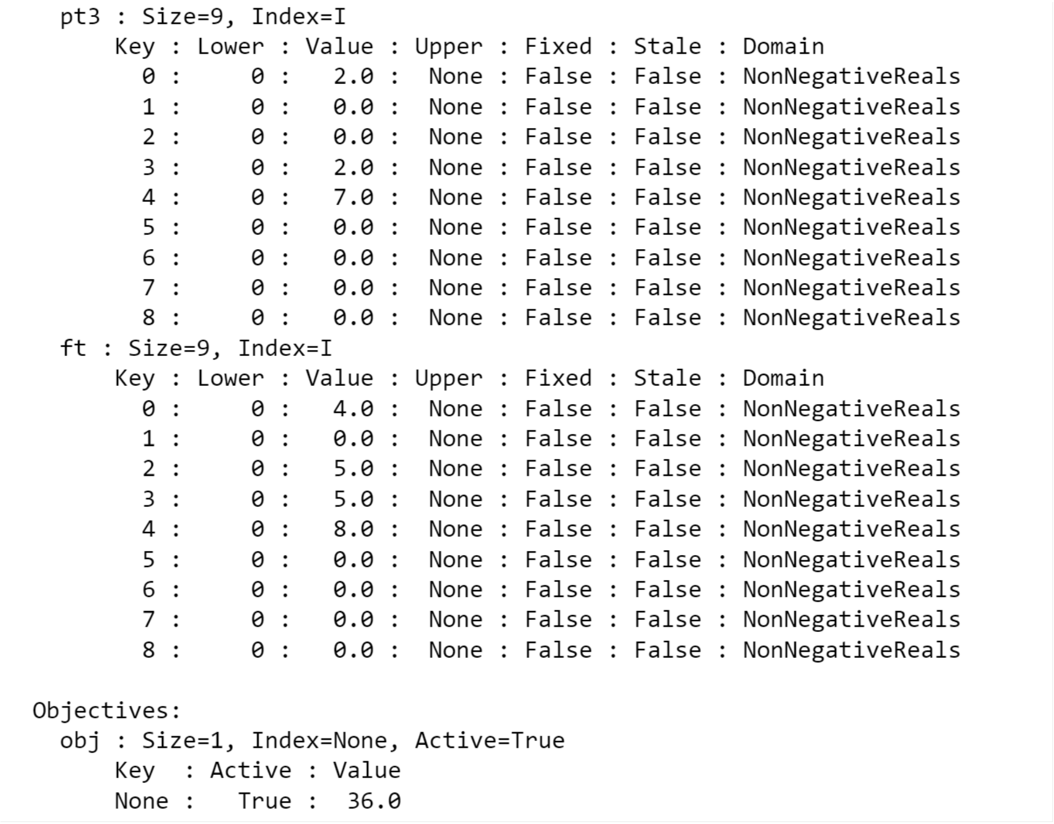
05. Run Model

```
from pyomo.opt import SolverStatus, TerminationCondition
n = len(z_pt)

last_start_shifts = [5, 6]
attempt = 1
for last_start_shift in last_start_shifts:
    print("###########################")
    print(f'Logging {attempt} .... \n')
    res, ns = model_call(last_start_shift)
    if (res.solver.status == SolverStatus.ok) and (res.solver.termination_condition == TerminationCondition.optimal):
        print('Optimal Solution found: last_start_shift = ', last_start_shift)
        break
    else:
        attempt += 1
        print('Model did not converge/No Optimal Solution found: last_start_shift = ', last_start_shift)
        print('Relaxing last start shift = ', last_start_shift+1, end='\n\n')
```

```
###########################
Logging 1 ....

Optimal Solution found: last_start_shift =  5
```



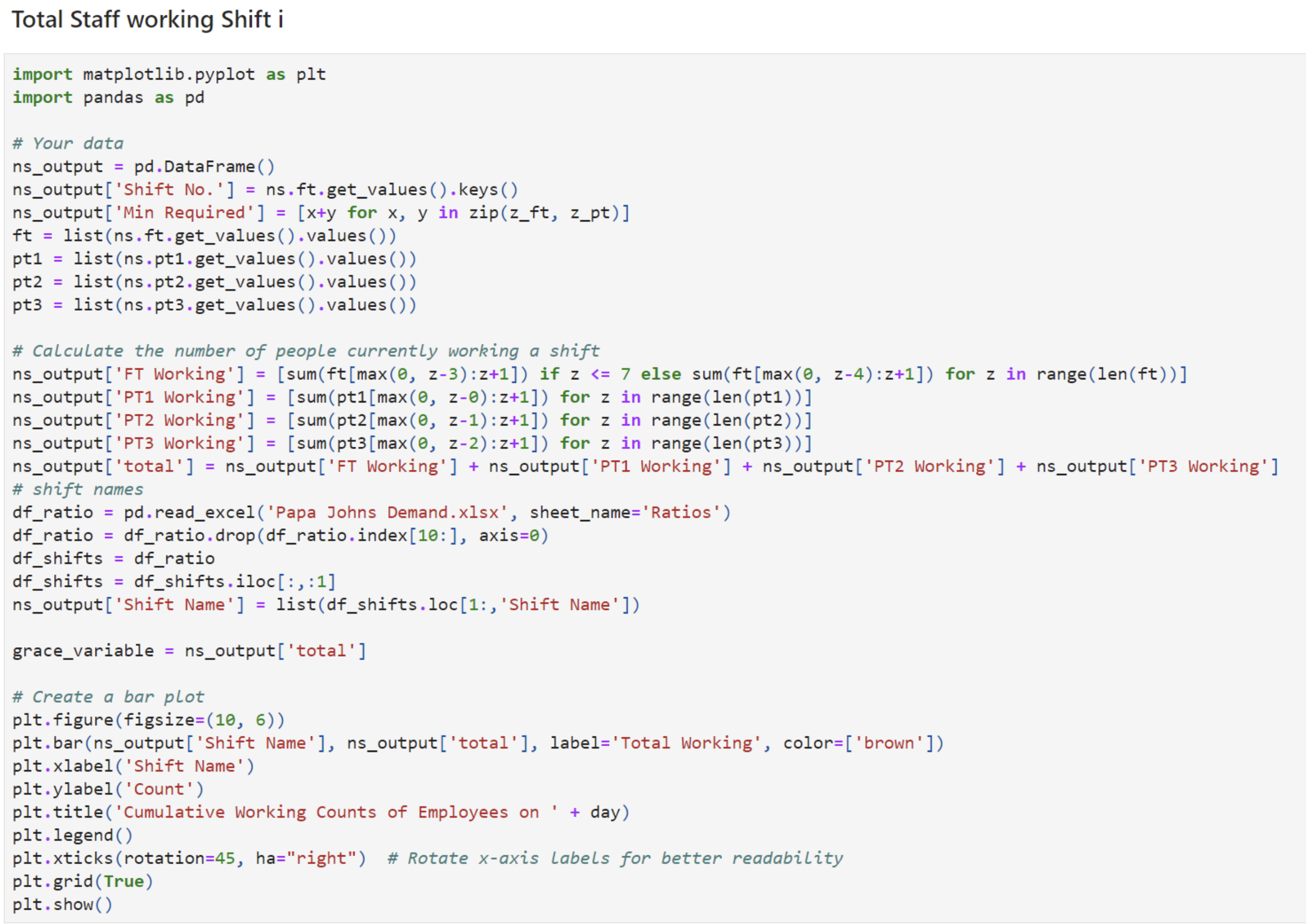
```
Model PJIOWA Staff Scheduling

  Variables:
    pt1 : Size=9, Index=I
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :   0.0 :  None : False : False : NonNegativeReals
          1 :     0 :   0.0 :  None : False : False : NonNegativeReals
          2 :     0 :   0.0 :  None : False : False : NonNegativeReals
          3 :     0 :   1.0 :  None : False : False : NonNegativeReals
          4 :     0 :   0.0 :  None : False : False : NonNegativeReals
          5 :     0 :   0.0 :  None : False : False : NonNegativeReals
          6 :     0 :   0.0 :  None : False : False : NonNegativeReals
          7 :     0 :   0.0 :  None : False : False : NonNegativeReals
          8 :     0 :   0.0 :  None : False : False : NonNegativeReals
    pt2 : Size=9, Index=I
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :   0.0 :  None : False : False : NonNegativeReals
          1 :     0 :   0.0 :  None : False : False : NonNegativeReals
          2 :     0 :   2.0 :  None : False : False : NonNegativeReals
          3 :     0 :   0.0 :  None : False : False : NonNegativeReals
          4 :     0 :   0.0 :  None : False : False : NonNegativeReals
          5 :     0 :   0.0 :  None : False : False : NonNegativeReals
          6 :     0 :   0.0 :  None : False : False : NonNegativeReals
          7 :     0 :   0.0 :  None : False : False : NonNegativeReals
          8 :     0 :   0.0 :  None : False : False : NonNegativeReals
```



```
    pt3 : Size=9, Index=I
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :   2.0 :  None : False : False : NonNegativeReals
          1 :     0 :   0.0 :  None : False : False : NonNegativeReals
          2 :     0 :   0.0 :  None : False : False : NonNegativeReals
          3 :     0 :   2.0 :  None : False : False : NonNegativeReals
          4 :     0 :   7.0 :  None : False : False : NonNegativeReals
          5 :     0 :   0.0 :  None : False : False : NonNegativeReals
          6 :     0 :   0.0 :  None : False : False : NonNegativeReals
          7 :     0 :   0.0 :  None : False : False : NonNegativeReals
          8 :     0 :   0.0 :  None : False : False : NonNegativeReals
    ft : Size=9, Index=I
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :   4.0 :  None : False : False : NonNegativeReals
          1 :     0 :   0.0 :  None : False : False : NonNegativeReals
          2 :     0 :   5.0 :  None : False : False : NonNegativeReals
          3 :     0 :   5.0 :  None : False : False : NonNegativeReals
          4 :     0 :   8.0 :  None : False : False : NonNegativeReals
          5 :     0 :   0.0 :  None : False : False : NonNegativeReals
          6 :     0 :   0.0 :  None : False : False : NonNegativeReals
          7 :     0 :   0.0 :  None : False : False : NonNegativeReals
          8 :     0 :   0.0 :  None : False : False : NonNegativeReals

  Objectives:
    obj : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True :  36.0
```

Appendix D

Code to Visualize Results



Total Staff working Shift i

```
import matplotlib.pyplot as plt
import pandas as pd

# Your data
ns_output = pd.DataFrame()
ns_output['Shift No.'] = ns.ft.get_values().keys()
ns_output['Min Required'] = [x+y for x, y in zip(z_ft, z_pt)]
ft = list(ns.ft.get_values().values())
pt1 = list(ns.pt1.get_values().values())
pt2 = list(ns.pt2.get_values().values())
pt3 = list(ns.pt3.get_values().values())

# Calculate the number of people currently working a shift
ns_output['FT Working'] = [sum(ft[max(0, z-3):z+1]) if z <= 7 else sum(ft[max(0, z-4):z+1]) for z in range(len(ft))]
ns_output['PT1 Working'] = [sum(pt1[max(0, z-0):z+1]) for z in range(len(pt1))]
ns_output['PT2 Working'] = [sum(pt2[max(0, z-1):z+1]) for z in range(len(pt2))]
ns_output['PT3 Working'] = [sum(pt3[max(0, z-2):z+1]) for z in range(len(pt3))]
ns_output['total'] = ns_output['FT Working'] + ns_output['PT1 Working'] + ns_output['PT2 Working'] + ns_output['PT3 Working']
# shift names
df_ratio = pd.read_excel('Papa Johns Demand.xlsx', sheet_name='Ratios')
df_ratio = df_ratio.drop(df_ratio.index[10:], axis=0)
df_shifts = df_ratio
df_shifts = df_shifts.iloc[:,:1]
ns_output['Shift Name'] = list(df_shifts.loc[1:,'Shift Name'])

grace_variable = ns_output['total']

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(ns_output['Shift Name'], ns_output['total'], label='Total Working', color=['brown'])
plt.xlabel('Shift Name')
plt.ylabel('Count')
plt.title('Cumulative Working Counts of Employees on ' + day)
plt.legend()
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```

No of Staff working in Shift i vs Demand

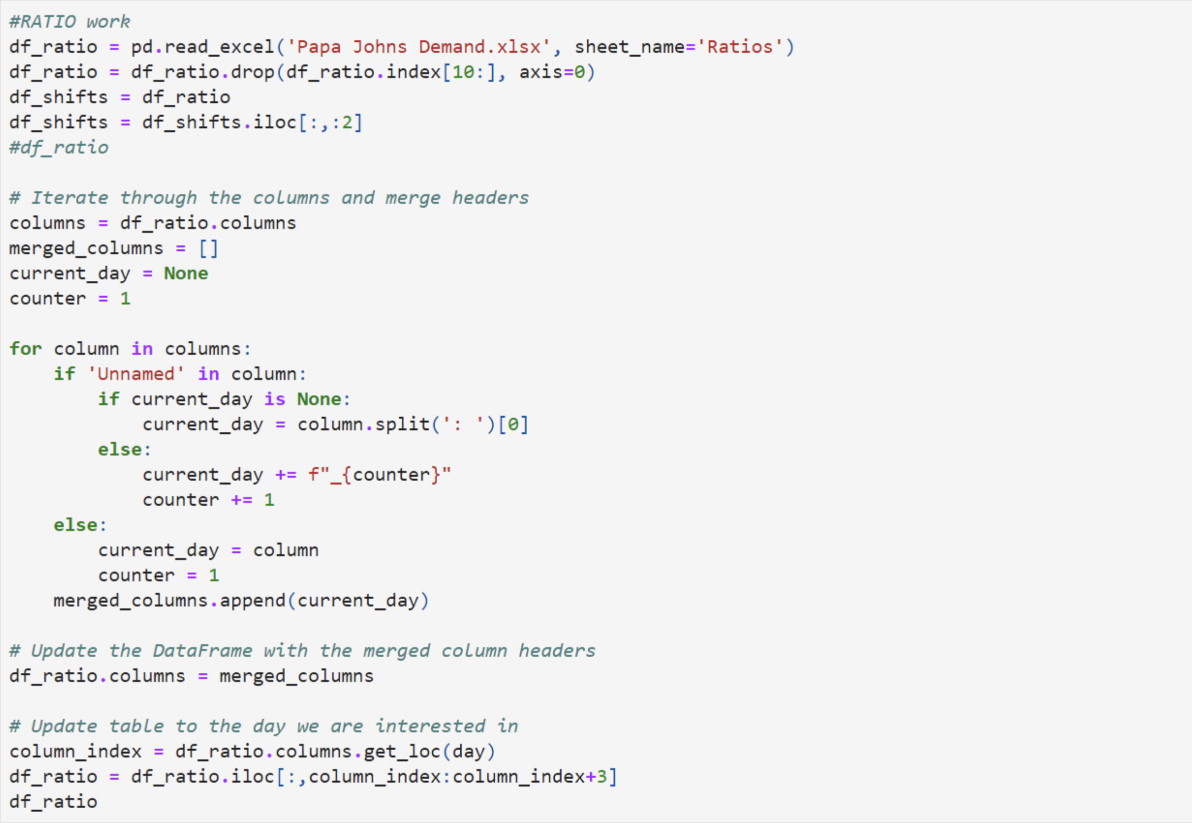
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Your data
# Assuming ns_output, z_ft, z_pt, ft, pt1, pt2, pt3, df_ratio, df_shifts, day, and rec_shift_req are defined

ns_output['Min Required'] = [x + y for x, y in zip(z_ft, z_pt)]
ns_output['FT Working'] = [sum(ft[max(0, z-3):z+1]) if z <= 7 else sum(ft[max(0, z-4):z+1]) for z in range(len(ft))]
ns_output['PT1 Working'] = [sum(pt1[max(0, z-0):z+1]) for z in range(len(pt1))]
ns_output['PT2 Working'] = [sum(pt2[max(0, z-1):z+1]) for z in range(len(pt2))]
ns_output['PT3 Working'] = [sum(pt3[max(0, z-2):z+1]) for z in range(len(pt3))]
ns_output['total'] = ns_output['FT Working'] + ns_output['PT1 Working'] + ns_output['PT2 Working'] + ns_output['PT3 Working']
ns_output['total_req'] = rec_shift_req[day]
ns_output['Shift Name'] = list(df_shifts.loc[1:, 'Shift Name'])

# Create numerical representation for x-axis ticks
shift_values = np.arange(len(ns_output['Shift Name']))

plt.figure(figsize=(10, 6))
width = 0.35

# Determine the colors and labels based on the conditions
colors = ['green' if total_val > total_req_val else ('orange' if total_val == total_req_val else 'red')
          for total_val, total_req_val in zip(ns_output['total'], ns_output['total_req'])]

colors1 = ['red' if total_val < total_req_val else '#ADD8E6'
          for total_val, total_req_val in zip(ns_output['total'], ns_output['total_req'])]

legend_labels = {'red': 'Did not meet demand', 'orange': 'Meets demand', 'green': 'Exceeds demand'}

# Plot the bars with the determined colors
bar1 = plt.bar(shift_values - width/2, ns_output['total'], width, label='Total Working', color=colors, edgecolor='grey')
bar2 = plt.bar(shift_values + width/2, ns_output['total_req'], width, label='Total Required', color=colors1, edgecolor='grey')

# Create a legend for the color map
legend_handles = [plt.Line2D([0], [0], color=color, marker='o', linestyle='', markersize=10, label=label)
                  for color, label in legend_labels.items()]

plt.legend(handles=legend_handles, title='Demand Status', loc='upper left')

plt.xlabel('Shift Name')
plt.ylabel('Count')
plt.title('Total Staff Working and Required for each shift')
plt.xticks(shift_values, ns_output['Shift Name'], rotation=45, ha="right")  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```

No of Staff starting Shift i

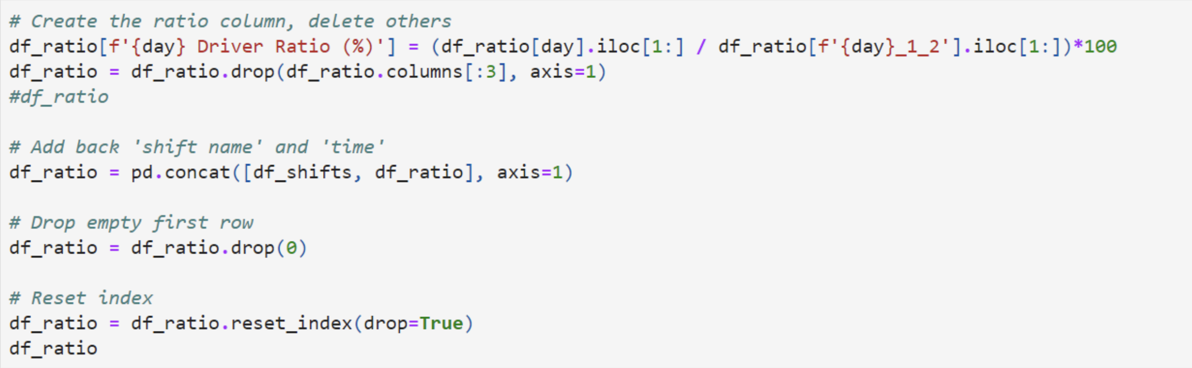
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Your data
# Assuming ns_output, z_ft, z_pt, ft, pt1, pt2, pt3, df_ratio, df_shifts, and day are defined

ns_output['Min Required'] = [x + y for x, y in zip(z_ft, z_pt)]
ns_output['FT Starting'] = [int(ft[z]) for z in range(len(ft))]
ns_output['PT1 Starting'] = [int(pt1[z]) for z in range(len(pt1))]
ns_output['PT2 Starting'] = [int(pt2[z]) for z in range(len(pt2))]
ns_output['PT3 Starting'] = [int(pt3[z]) for z in range(len(pt3))]

# Create a bar chart
plt.figure(figsize=(10,6))

bar_width = 0.2  # Adjust the width of the bars

# Plotting the bars
plt.bar(shift_values - 1.5 * bar_width, ns_output['FT Starting'], width=bar_width, label='8 hours (10 hrs in shift 5)')
plt.bar(shift_values - 0.5 * bar_width, ns_output['PT1 Starting'], width=bar_width, label='2 hours')
plt.bar(shift_values + 0.5 * bar_width, ns_output['PT2 Starting'], width=bar_width, label='4 hours')
plt.bar(shift_values + 1.5 * bar_width, ns_output['PT3 Starting'], width=bar_width, label='6 hours')

plt.xlabel('Shift Name')
plt.ylabel('Count')
plt.title('Starting Employee Counts for each shift on ' + day)
plt.legend()
plt.xticks(shift_values, shift_names, rotation=45, ha="right")  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```

06. Ratio Split - Insiders/Driver

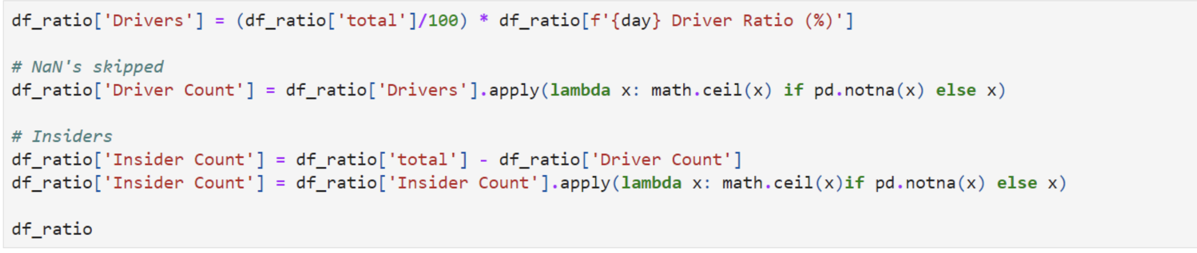
Ratio Code



```
#RATIO work
df_ratio = pd.read_excel('Papa Johns Demand.xlsx', sheet_name='Ratios')
df_ratio = df_ratio.drop(df_ratio.index[10:], axis=0)
df_shifts = df_ratio
df_shifts = df_shifts.iloc[:,:2]
#df_ratio

# Iterate through the columns and merge headers
columns = df_ratio.columns
merged_columns = []
current_day = None
counter = 1

for column in columns:
    if 'Unnamed' in column:
        if current_day is None:
            current_day = column.split(': ')[0]
        else:
            current_day += f"_{counter}"
            counter += 1
    else:
        current_day = column
        counter = 1
    merged_columns.append(current_day)

# Update the DataFrame with the merged column headers
df_ratio.columns = merged_columns

# Update table to the day we are interested in
column_index = df_ratio.columns.get_loc(day)
df_ratio = df_ratio.iloc[:,column_index:column_index+3]
df_ratio
```



```
# Create the ratio column, delete others
df_ratio[f'{day} Driver Ratio (%)'] = (df_ratio[day].iloc[1:] / df_ratio[f'{day}_1_2'].iloc[1:])*100
df_ratio = df_ratio.drop(df_ratio.columns[:3], axis=1)
#df_ratio

# Add back 'shift name' and 'time'
df_ratio = pd.concat([df_shifts, df_ratio], axis=1)

# Drop empty first row
df_ratio = df_ratio.drop(0)

# Reset index
df_ratio = df_ratio.reset_index(drop=True)
df_ratio
```



```
df_ratio['Drivers'] = (df_ratio['total']/100) * df_ratio[f'{day} Driver Ratio (%)']

# NaN's skipped
df_ratio['Driver Count'] = df_ratio['Drivers'].apply(lambda x: math.ceil(x) if pd.notna(x) else x)

# Insiders
df_ratio['Insider Count'] = df_ratio['total'] - df_ratio['Driver Count']
df_ratio['Insider Count'] = df_ratio['Insider Count'].apply(lambda x: math.ceil(x)if pd.notna(x) else x)

df_ratio
```

Ratio Visual

```
#Visualize
import pandas as pd
import matplotlib.pyplot as plt

# Selecting only the relevant columns for the bar graph
df_plot = df_ratio[['Shift Name', 'Driver Count', 'Insider Count']]

# Setting 'Shift Name' as the index for plotting
df_plot.set_index('Shift Name', inplace=True)
plt.figure(figsize=(10, 6))
# Plotting
ax = df_plot.plot(kind='bar', stacked=True, color=['red', 'brown'])

# Plot 'Drivers' bars with counts
ax.bar(9,df_plot['Driver Count'], width=0.2)
for i, driver in enumerate(df_plot['Driver Count']):
    ax.text(i, driver / 2, str(driver), ha='center', va='center', color='white')

# Plot 'Insiders' bars with counts on top of 'Drivers'
ax.bar(9,df_plot['Insider Count'], width=0.2, bottom=df_plot['Driver Count'])
for i, insider in enumerate(df_plot['Insider Count']):
    ax.text(i, df_plot['Driver Count'][i] + insider / 2, str(insider), ha='center', va='center', color='white')

# Setting labels and title
plt.xlabel('Shift Name')
plt.ylabel('Values')
plt.title('Cumulative Working Counts of Employees on ' + day)

# Display the legend
plt.legend()
# plt.grid(True)
# Show the plot
plt.xticks(rotation=45, ha="right")
plt.show()
```