Introduction

Many adults look forward to spending their free time touring the bars and restaurants Columbus has to offer. The Short North Arts District is a vibrant Columbus neighborhood known for its nightlife and dining experiences commonly enjoyed by those looking to unwind. As students, we understand that a night full of entertainment can be expensive, making it a daunting task. We sought to discover the least expensive way to explore Short North nightlife by collecting pricing data from drink menus and the bar's corresponding yelp rating. This minimization problem ensures different types of drinks are purchased, the average yelp rating of all 10 bars is at least 3.5, and that multiple bars are visited. These constraints preserve the concept of exploration and offer a variety of establishments and drink types so one can get the most out of a single night.

Model Formulation

Decision Variables

y_i, a binary variable that is 0 when the corresponding bar is not visited and 1 when it is, with $i = 1, \ldots, 10$

sc_i, an integer variable that is the number of signature cocktails bought at the corresponding bar, with $i = 1, \ldots, 9$

db_i, an integer variable that is the number of domestic beers bought at the corresponding bar, with $i = 1, \ldots, 4, 6, \ldots, 10$

cb_i, an integer variable that is the number of craft beers bought at the corresponding bar, with $i = 1, \ldots, 4, 6, \ldots, 10$

s_i, an integer variable that is the number of shots bought at the corresponding bar, with $i = 1, \ldots, 6, 8, 9$

w_i, an integer variable that is the number of wine bought at the corresponding bar, with $i = 1, \ldots, 4, 6, \ldots, 10$

Z is money spent in dollars

Model

$$MinZ = 10sc_1 + 5.5db_1 + 6.5cb_1 + 7s_1 + 7w_1 + 12sc_2 + 5db_2 + 7cb_2 + 8s_2 + 8w_2 + 10sc_3 + 7db_3 + 8cb_3 + 4s_3 + 8w_3 + 10sc_4 + 7db_4 + 8cb_4 + 6s_4 + 9w_4 + 9sc_5 + 8s_5 + 17sc_6 + 6db_6 + 8cb_6 + 15s_6 + 11w_6 + 12sc_7 + 7db_7 + 7cb_7 + 12w_7 + 8sc_8 + 3db_8 + 5cb_8 + 4s_8 + 6w_8 + 9sc_9 + 4.25db_9 + 5.75cb_9 + 5s_9 + 9w_9 + 5db_{10} + 6.5cb_{10} + 5w_{10}$$

S.T.

$$sc_1 + db_1 + cb_1 + s_1 + w_1 <= 2$$

$$sc_2 + db_2 + cb_2 + s_2 + w_2 <= 2$$

$$sc_3 + db_3 + cb_3 + s_3 + w_3 <= 2$$

$$sc_4 + db_4 + cb_4 + s_4 + w_4 <= 2$$

$$sc_5 + s_5 <= 2$$

$$sc_6 + db_6 + cb_6 + s_6 + w_6 <= 2$$

$$sc_7 + db_7 + cb_7 + w_7 <= 2$$

$$sc_8 + db_8 + cb_8 + s_8 + w_8 <= 2$$

$$sc_9 + db_9 + cb_9 + s_9 + w_9 <= 2$$

$$db_{10} + cb_{10} + w_{10} <= 2$$

$$db_1 + db_2 + db_3 + db_4 + db_6 + db_7 + db_8 + db_9 + db_{10} <= 2$$

$$cb_1 + cb_2 + cb_3 + cb_4 + cb_6 + cb_7 + cb_8 + cb_9 + cb_{10} <= 2$$

$$sc_1 + sc_2 + sc_3 + sc_4 + sc_5 + sc_6 + sc_7 + sc_8 + sc_9 >= 1$$

$$sc_1 + db_1 + cb_1 + s_1 + w_1 + sc_2 + db_2 + cb_2 + s_2 + w_2 + sc_3 + db_3 + cb_3 + s_3 + w_3 + sc_4 + db_4 + cb_4 + s_4 + w_4 + sc_5 + s_5 + sc_6 + db_6 + cb_6 + s_6 + w_6 + sc_7 + db_7 + cb_7 + w_7 + sc_8 + db_8 + cb_8 + s_8 + w_8 + sc_9 + db_9 + cb_9 + s_9 + w_9 + db_{10} + cb_{10} + w_{10} == 8$$

$$100(1 - y_1) + sc_1 + db_1 + cb_1 + s_1 + w_1 >= 1$$

$$100(1 - y_2) + sc_2 + db_2 + cb_2 + s_2 + w_2 >= 1$$

$$100(1 - y_3) + sc_3 + db_3 + cb_3 + s_3 + w_3 >= 1$$

$$100(1 - y_4) + sc_4 + db_4 + cb_4 + s_4 + w_4 >= 1$$

$$100(1 - y_5) + sc_5 + s_5 >= 1$$

$$100(1 - y_6) + sc_6 + db_6 + cb_6 + s_6 + w_6 >= 1$$

$$100(1 - y_7) + sc_7 + db_7 + cb_7 + w_7 >= 1$$

$$100(1 - y_8) + sc_8 + db_8 + cb_8 + s_8 + w_8 >= 1$$

$$100(1 - y_9) + sc_9 + db_9 + cb_9 + s_9 + w_9 >= 1$$

$$100(1 - y_{10}) + db_{10} + cb_{10} + w_{10} >= 1$$

$$-100y_1 + sc_1 + db_1 + cb_1 + s_1 + w_1 <= 0$$

$$-100y_2 + sc_2 + db_2 + cb_2 + s_2 + w_2 <= 0$$

$$-100y_3 + sc_3 + db_3 + cb_3 + s_3 + w_3 <= 0$$

$$-100y_4 + sc_4 + db_4 + cb_4 + s_4 + w_4 <= 0$$

$$-100y_5 + sc_5 + s_5 <= 0$$

$$-100y_6 + sc_6 + db_6 + cb_6 + s_6 + w_6 <= 0$$

$$-100y_7 + sc_7 + db_7 + cb_7 + w_7 <= 0$$

$$-100y_8 + sc_8 + db_8 + cb_8 + s_8 + w_8 <= 0$$

$$-100y_9 + sc_9 + db_9 + cb_9 + s_9 + w_9 <= 0$$

$$-100y_{10} + db_{10} + cb_{10} + w_{10} <= 0$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} >= 5$$

$$4y_1 + 4y_2 + 4y_3 + 4y_4 + 2.5y_5 + 4y_6 + 4.5y_7 + 3.5y_8 + 2y_9 + 2y_{10} >= 3.5(y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10})$$

$$sc_i >= 0, i = 1, \ldots, 9$$

$$db_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

$$cb_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

$$s_i >= 0, i = 1, \ldots, 6, 8, 9$$

$$w_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

Breakdown

OBJECTIVE FUNCTION

$$MinZ = 10sc_1 + 5.5db_1 + 6.5cb_1 + 7s_1 + 7w_1 + 12sc_2 + 5db_2 + 7cb_2 + 8s_2 + 8w_2 + 10sc_3 + 7db_3 + 8cb_3 + 4s_3 + 8w_3 + 10sc_4 + 7db_4 + 8cb_4 + 6s_4 + 9w_4 + 9sc_5 + 8s_5 + 17sc_6 + 6db_6 + 8cb_6 + 15s_6 + 11w_6 + 12sc_7 + 7db_7 + 7cb_7 + 12w_7 + 8sc_8 + 3db_8 + 5cb_8 + 4s_8 + 6w_8 + 9sc_9 + 4.25db_9 + 5.75cb_9 + 5s_9 + 9w_9 + 5db_{10} + 6.5cb_{10} + 5w_{10}$$

In [ ]:

```
obj_func= (10*sc[0] + 12*sc[1] + 10*sc[2] + 10*sc[3] + 9*sc[4] + 17*sc[5] + 12*sc[6] + 8*sc[7] + 9*sc[8]
    + 5.5*db[0] + 5*db[1] + 7*db[2] + 7*db[3] + 6*db[4] + 7*db[5] + 3*db[6] + 4.25*db[7] + 5*db[8]
    + 6.5*cb[0] + 7*cb[1] + 8*cb[2] + 8*cb[3] + 8*cb[4] + 7*cb[5] + 5*cb[6] + 5.75*cb[7] + 6.5*cb[8]
    + 7*s[0] + 8*s[1] + 4*s[2] + 6*s[3] + 8*s[4] + 15*s[5] + 4*s[6] + 5*s[7]
    + 7*w[0] + 8*w[1] + 8*w[2] + 9*w[3] + 11*w[4] + 12*w[5] + 6*w[6] + 9*w[7] + 5*w[8])

problem = cp.Problem(cp.Minimize(obj_func), constraints)
```

The Objective of our problem is to minimize the amount of money spent on the bar crawl, therefore it must contain the costs of each drink in each bar multiplied by the number of that drink taken.

BAR DRINK LIMIT

$$sc_1 + db_1 + cb_1 + s_1 + w_1 <= 2$$

$$sc_2 + db_2 + cb_2 + s_2 + w_2 <= 2$$

$$sc_3 + db_3 + cb_3 + s_3 + w_3 <= 2$$

$$sc_4 + db_4 + cb_4 + s_4 + w_4 <= 2$$

$$sc_5 + s_5 <= 2$$

$$sc_6 + db_6 + cb_6 + s_6 + w_6 <= 2$$

$$sc_7 + db_7 + cb_7 + w_7 <= 2$$

$$sc_8 + db_8 + cb_8 + s_8 + w_8 <= 2$$

$$sc_9 + db_9 + cb_9 + s_9 + w_9 <= 2$$

$$db_{10} + cb_{10} + w_{10} <= 2$$

In [ ]:

```
constraints.append(sc[0] + db[0] + cb[0] + s[0] + w[0] <= 2)
constraints.append(sc[1] + db[1] + cb[1] + s[1] + w[1] <= 2)
constraints.append(sc[2] + db[2] + cb[2] + s[2] + w[2] <= 2)
constraints.append(sc[3] + db[3] + cb[3] + s[3] + w[3] <= 2)
constraints.append(sc[4] + s[4] <= 2)
constraints.append(sc[5] + db[4] + cb[4] + s[5] + w[4] <= 2)
constraints.append(sc[6] + db[5] + cb[5] + w[5] <= 2)
constraints.append(sc[7] + db[6] + cb[6] + s[6] + w[6] <= 2)
constraints.append(sc[8] + db[7] + cb[7] + w[7] + s[7] <= 2)
constraints.append(db[8] + cb[8] + w[8] <= 2)
```

In order to increase the variety of bars that are chosen, we sum the number of drinks taken at each bar and limit them to be less than or equal to two.

BEER LIMIT

$$db_1 + db_2 + db_3 + db_4 + db_6 + db_7 + db_8 + db_9 + db_{10} <= 2$$

$$cb_1 + cb_2 + cb_3 + cb_4 + cb_6 + cb_7 + cb_8 + cb_9 + cb_{10} <= 2$$

In [ ]:

```
constraints.append(db[0] + db[1] + db[2] + db[3] + db[4] + db[5] + db[6] + db[7] + db[8] <= 2)
constraints.append(cb[0] + cb[1] + cb[2] + cb[3] + cb[4] + cb[5] + cb[6] + cb[7] + cb[8] <= 2)
```

Since beer is generally the cheapest drink option, if there was not a limit on the number of total beers purchased, they would likely dominate the drinks purchased. Therefore, limiting the total numbers of each beer type to two should increase the variety of drinks purchased.

EXPENSIVE DRINK MIN

$$sc_1 + sc_2 + sc_3 + sc_4 + sc_5 + sc_6 + sc_7 + sc_8 + sc_9 >= 1$$

In [ ]:

```
constraints.append(sc[0] + sc[1] + sc[2] + sc[3] + sc[4] + sc[5] + sc[6] + sc[7] + sc[8] >= 1)
```

Since the signature cocktails are a more expensive drink type, they will likely be avoided by the program if there is not a constraint enforcing that we include one. Thus the sum of all signature cocktails must be one.

TOTAL DRINK GOAL

$$sc_1 + db_1 + cb_1 + s_1 + w_1 + sc_2 + db_2 + cb_2 + s_2 + w_2 + sc_3 + db_3 + cb_3 + s_3 + w_3 + sc_4 + db_4 + cb_4 + s_4 + w_4 + sc_5 + s_5 + sc_6 + db_6 + cb_6 + s_6 + w_6 + sc_7 + db_7 + cb_7 + w_7 + sc_8 + db_8 + cb_8 + s_8 + w_8 + sc_9 + db_9 + cb_9 + s_9 + w_9 + db_{10} + cb_{10} + w_{10} == 8$$

In [ ]:

```
constraints.append(sc[0] + db[0] + cb[0] + s[0] + w[0]
    + sc[1] + db[1] + cb[1] + s[1] + w[1]
    + sc[2] + db[2] + cb[2] + s[2] + w[2]
    + sc[3] + db[3] + cb[3] + s[3] + w[3]
    + sc[4] + s[4] + w[4] + db[4] + cb[4]
    + sc[5] + db[5] + cb[5] + s[5] + w[5]
    + sc[6] + db[6] + cb[6] + w[6] + s[6]
    + sc[7] + db[7] + cb[7] + s[7] + w[7]
    + sc[8] + db[8] + cb[8] + w[8] == 8)
```

Since the goal of the problem is to minimize the total cost of drinks, we need to ensure that we are actually purchasing drinks (otherwise the number of drinks purchased may be 0). Additionally, an inequality does nothing since the minimum drink limit will always be chosen as long as a feasible solution exists. We chose for the sum of all drinks to be equal to eight.

IF THENS FOR BAR VISITS

Ensures y's are 1 when the bar has been visited

$$100(1 - y_1) + sc_1 + db_1 + cb_1 + s_1 + w_1 >= 1$$

$$100(1 - y_2) + sc_2 + db_2 + cb_2 + s_2 + w_2 >= 1$$

$$100(1 - y_3) + sc_3 + db_3 + cb_3 + s_3 + w_3 >= 1$$

$$100(1 - y_4) + sc_4 + db_4 + cb_4 + s_4 + w_4 >= 1$$

$$100(1 - y_5) + sc_5 + s_5 >= 1$$

$$100(1 - y_6) + sc_6 + db_6 + cb_6 + s_6 + w_6 >= 1$$

$$100(1 - y_7) + sc_7 + db_7 + cb_7 + w_7 >= 1$$

$$100(1 - y_8) + sc_8 + db_8 + cb_8 + s_8 + w_8 >= 1$$

$$100(1 - y_9) + sc_9 + db_9 + cb_9 + s_9 + w_9 >= 1$$

$$100(1 - y_{10}) + db_{10} + cb_{10} + w_{10} >= 1$$

Ensures y's are 0 when the bar has not been visited

$$-100y_1 + sc_1 + db_1 + cb_1 + s_1 + w_1 <= 0$$

$$-100y_2 + sc_2 + db_2 + cb_2 + s_2 + w_2 <= 0$$

$$-100y_3 + sc_3 + db_3 + cb_3 + s_3 + w_3 <= 0$$

$$-100y_4 + sc_4 + db_4 + cb_4 + s_4 + w_4 <= 0$$

$$-100y_5 + sc_5 + s_5 <= 0$$

$$-100y_6 + sc_6 + db_6 + cb_6 + s_6 + w_6 <= 0$$

$$-100y_7 + sc_7 + db_7 + cb_7 + w_7 <= 0$$

$$-100y_8 + sc_8 + db_8 + cb_8 + s_8 + w_8 <= 0$$

$$-100y_9 + sc_9 + db_9 + cb_9 + s_9 + w_9 <= 0$$

$$-100y_{10} + db_{10} + cb_{10} + w_{10} <= 0$$

In [ ]:

```
constraints.append(100*(1-y[0]) + sc[0] + db[0] + cb[0] + s[0] + w[0] >= 1)
constraints.append(100*(1-y[1]) + sc[1] + db[1] + cb[1] + s[1] + w[1] >= 1)
constraints.append(100*(1-y[2]) + sc[2] + db[2] + cb[2] + s[2] + w[2] >= 1)
constraints.append(100*(1-y[3]) + sc[3] + db[3] + cb[3] + s[3] + w[3] >= 1)
constraints.append(100*(1-y[4]) + sc[4] + s[4] >= 1)
constraints.append(100*(1-y[5]) + sc[5] + db[4] + cb[4] + s[5] + w[4] >= 1)
constraints.append(100*(1-y[6]) + sc[6] + db[5] + cb[5] + w[5] >= 1)
constraints.append(100*(1-y[7]) + sc[7] + db[6] + cb[6] + s[6] + w[6] >= 1)
constraints.append(100*(1-y[8]) + sc[8] + db[7] + cb[7] + w[7] + s[7] >= 1)
constraints.append(100*(1-y[9]) + db[8] + cb[8] + w[8] >= 1)

constraints.append(-100*y[0] + sc[0] + db[0] + cb[0] + s[0] + w[0] <= 0)
constraints.append(-100*y[1] + sc[1] + db[1] + cb[1] + s[1] + w[1] <= 0)
constraints.append(-100*y[2] + sc[2] + db[2] + cb[2] + s[2] + w[2] <= 0)
constraints.append(-100*y[3] + sc[3] + db[3] + cb[3] + s[3] + w[3] <= 0)
constraints.append(-100*y[4] + sc[4] + s[4] <= 0)
constraints.append(-100*y[5] + sc[5] + db[4] + cb[4] + s[5] + w[4] <= 0)
constraints.append(-100*y[6] + sc[6] + db[5] + cb[5] + w[5] <= 0)
constraints.append(-100*y[7] + sc[7] + db[6] + cb[6] + s[6] + w[6] <= 0)
constraints.append(-100*y[8] + sc[8] + db[7] + cb[7] + w[7] + s[7] <= 0)
constraints.append(-100*y[9] + db[8] + cb[8] + w[8] <= 0)
```

The goal of these constraints is to create a series of either-or constraints that ensure that, for each bar, if a drink has been purchased at the bar, the corresponding binary variable is one. Otherwise the binary variable is zero. If the value is one, the greater than or equal to constraints are enforced. If the value is zero, the less than or equal to constraints are enforced.

NUMBER OF LOCATIONS TO VISIT

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} >= 5$$

In [ ]:

```
constraints.append(y[0] + y[1] + y[2] + y[3] + y[4] + y[5] + y[6] + y[7] + y[8] + y[9] >= 5)
```

We further enforce the number of bars visited here. Since these are one when a corresponding bar is visited, this ensures that the number of bars visited will be at least 5.

AVERAGE REVIEW MIN

$$4y_1 + 4y_2 + 4y_3 + 4y_4 + 2.5y_5 + 4y_6 + 4.5y_7 + 3.5y_8 + 2y_9 + 2y_{10} >= 3.5(y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10})$$

In [ ]:

```
constraints.append(4*y[0] + 4*y[1] + 4*y[2] + 4*y[3] + 2.5*y[4] + 4*y[5] + 4.5*y[6] + 3.5*y[7] + 2*y[8] + 2*y[9] >= 3.5*(y[0] + y[1] + y[2] + y[3] + y[4] + y[5] + y[6
```

Since we wish for the bars to be of relatively high quality, we ensure the average of bars visited is greater than or equal to 3.5. We do not want these weighted by number of drinks had at each bar, which is why we use the binary variables.

NONNEGATIVITY

$$sc_i >= 0, i = 1, \ldots, 9$$

$$db_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

$$cb_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

$$s_i >= 0, i = 1, \ldots, 6, 8, 9$$

$$w_i >= 0, i = 1, \ldots, 4, 6, \ldots, 10$$

In [ ]:

```
constraints.append(sc[0] >= 0)
constraints.append(sc[1] >= 0)
constraints.append(sc[2] >= 0)
constraints.append(sc[3] >= 0)
constraints.append(sc[4] >= 0)
constraints.append(sc[5] >= 0)
constraints.append(sc[6] >= 0)
constraints.append(sc[7] >= 0)
constraints.append(sc[8] >= 0)


constraints.append(db[0] >= 0)
constraints.append(db[1] >= 0)
constraints.append(db[2] >= 0)
constraints.append(db[3] >= 0)
constraints.append(db[4] >= 0)
constraints.append(db[5] >= 0)
constraints.append(db[6] >= 0)
constraints.append(db[7] >= 0)
constraints.append(db[8] >= 0)


constraints.append(cb[0] >= 0)
constraints.append(cb[1] >= 0)
constraints.append(cb[2] >= 0)
constraints.append(cb[3] >= 0)
constraints.append(cb[4] >= 0)
constraints.append(cb[5] >= 0)
constraints.append(cb[6] >= 0)
constraints.append(cb[7] >= 0)
constraints.append(cb[8] >= 0)


constraints.append(s[0] >= 0)
constraints.append(s[1] >= 0)
constraints.append(s[2] >= 0)
constraints.append(s[3] >= 0)
constraints.append(s[4] >= 0)
constraints.append(s[5] >= 0)
constraints.append(s[6] >= 0)
constraints.append(s[7] >= 0)


constraints.append(w[0] >= 0)
constraints.append(w[1] >= 0)
constraints.append(w[2] >= 0)
constraints.append(w[3] >= 0)
constraints.append(w[4] >= 0)
constraints.append(w[5] >= 0)
constraints.append(w[6] >= 0)
constraints.append(w[7] >= 0)
constraints.append(w[8] >= 0)
```

Since we do not want any of the integer variables to be negative, we ensure that they are all greater than or equal to zero.

Results

As showed above, our objective was to minimize the amount of money spent on drinks for one day in the short north. We had a set of constraints to ensure a successful bar crawl. First we made sure to purchase 8 drinks which could be changed or modified in the future. To ensure we would have an enjoyable time, we wanted the average yelp rating of our bars to be at least 3.5 out of 5. To make sure we were purchasing a variety of drinks, we made sure to buy at least one signature cocktail, purchase a maximum of four beers total, and made sure to only buy two of the same drink at a bar. We also wanted to make sure we visited and bought drinks at no less than 5 different bars.

The optimized solution came out to spending 40 dollars in total and visiting Standard, Bodega, Pint House, Union Cafe, and Brothers. The different drink types purchased include: shots, signature cocktail, domestic beers, and wine. The particular amount and drinks and where they were purchased are illustrated in the table below.

In [ ]:

```
# Results from optimization
import pandas as pd
def highlight_max(x):
    return ['font-weight: bold' if (v == x.loc[len(x) - 1]) else ''
            for v in x]

bars_visited = ['Standard', 'Bodega', 'Pint House', 'Union Cafe', 'Brothers',
                'Total Cost']
drinks_purchased  = ['Signature Cocktail (1)', 'Shots (2)', 'Shots (1)',
                     'Domestic Beer (2)', 'Wine (2)', ' ']
cost = ['$10.00', '$8.00', '$6.00', '$6.00', '$10.00', '$40.00']

results_main = pd.DataFrame({'Bar Visited': bars_visited,
                             'Drinks Purchased': drinks_purchased,
                            'Cost (Dollars)': cost})
results_main.style.apply(highlight_max)
```

Out[ ]:

	Bar Visited	Drinks Purchased	Cost (Dollars)
0	Standard	Signature Cocktail (1)	$10.00
1	Bodega	Shots (2)	$8.00
2	Pint House	Shots (1)	$6.00
3	Union Cafe	Domestic Beer (2)	$6.00
4	Brothers	Wine (2)	$10.00
5	**Total Cost**		**$40.00**

Post-Optimality Analysis

Our optimal solution is largely based on the constraints put on the objective function. When creating our model, we were interested in creating a balanced night that includes a variety of both drink types and bars visited. We decided to run our model under a few different sets of conditions, such as increasing the average Yelp rating to 4.1, removing the average Yelp rating constraint, and removing constraints requiring multiple drink types to be purchased. We have demonstrate below that the value for the objective function does not change by a large degree when the constraints are manipulated. However, it is the case that the distribution of drink types varies widely depending on what drinks are constrained. It was also apparent that when allowing the user to purchase many drinks at a single bar, the distribution of drink types lowers significantly and only 2 bars would be preferred when minimizing. Ultimately, our model is highly sensitive to the amount of bars required and how many drinks may be purchased at each bar. When these constraints are modified, the variety bars visited and drink types purchased may vary as shown in the tables below.

In [ ]:

```
# Table 1
def highlight_max(x):
    return ['font-weight: bold' if (v == x.loc[len(x) - 1]) else ''
            for v in x]

bars_visited = ['Standard', 'Bristol', 'Bodega', 'Gaswerks', 'Union Cafe',
                'Total Cost']
drinks_purchased  = ['Domestic Beer', 'Domestic Beer', 'Shots (2)',
                     'Domestic Beer (2)', 'Domestic Beer (2)', ' ']
cost = ['$5.50', '$5.00', '$8.00', '$8.50', '$6.00', '$33.00']

table_1 = pd.DataFrame({'Bar Visited': bars_visited,
                             'Drinks Purchased': drinks_purchased,
                            'Cost (Dollars)': cost})
table_1.style.apply(highlight_max)
```

Out[ ]:

	Bar Visited	Drinks Purchased	Cost (Dollars)
0	Standard	Domestic Beer	$5.50
1	Bristol	Domestic Beer	$5.00
2	Bodega	Shots (2)	$8.00
3	Gaswerks	Domestic Beer (2)	$8.50
4	Union Cafe	Domestic Beer (2)	$6.00
5	**Total Cost**		**$33.00**

The results of our minimization when we place no constraints on the types of drinks purchased. I.e, the amount of beer or signature cocktails purchased are not limited.

In [ ]:

```
# Table 2
bars_visited = ['Bodega', 'Union Cafe', 'Union Cafe',
                'Total Cost']
drinks_purchased  = ['Shots (5)','Signature Cocktail', 'Domestic Beer (2)',
                     ' ']
cost = ['$20.00', '$8.00', '$6.00', '$34.00']

table_2 = pd.DataFrame({'Bar Visited': bars_visited,
                             'Drinks Purchased': drinks_purchased,
                            'Cost (Dollars)': cost})
table_2.style.apply(highlight_max)
```

Out[ ]:

	Bar Visited	Drinks Purchased	Cost (Dollars)
0	Bodega	Shots (5)	$20.00
1	Union Cafe	Signature Cocktail	$8.00
2	Union Cafe	Domestic Beer (2)	$6.00
3	**Total Cost**		**$34.00**

The results of our minimization when the number of drinks purchased at a single bar is not constrained. This does not require an individual to visit multiple bars.

Appendix A

The presentation video can be found at:

https://youtu.be/c0MlBTbO358

Appendix B

https://github.com/ebyrd-cs/ISE3230_Group5_AU22

Appendix C

The tasks were close to evenly split between each member, and each member attented and participated in each meeting/step of the process including project proposal, constraint formulation, coding, and report creation.

-Philip Kinnersley, Ethan Byrd, Sophia Ricard, Carli Werner