Case 1: Lyft-off At Toledo Airport

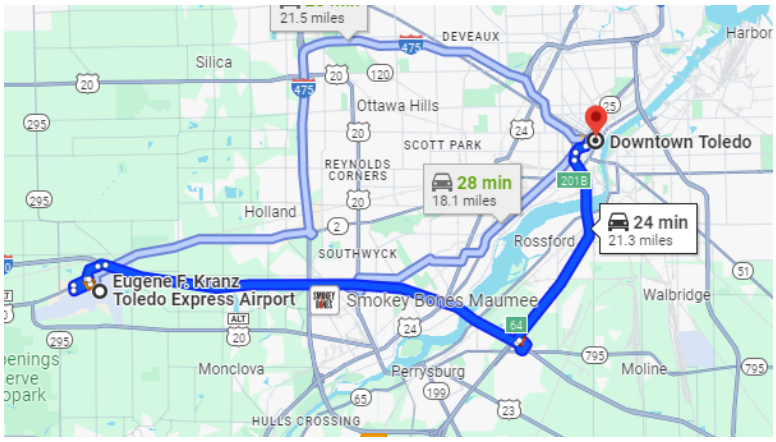
Analysis and insights provided by: Marc Torchio

3/18/2024

Contents

Executive Summary

As the stand-in pricing product manager for Lyft's ride-scheduling feature, it is the purpose of this report to present pricing recommendations for Lyft's new route between Toledo Express Airport and downtown Toledo, pictured below.



In this case, there are few assumptions, summarized below, that were pivotal throughout the analysis and modeling:

Variable**	Value
Driver CAC	$400 - $600
Driver Avg Rides/Month	100
Driver Churn per month	5%
Rider CAC	$10 - $20
Rider Avg Rides/Month	1
Match Rate When take == $6.00	60%
Match Rate When take == $3.00	93%
Rider Churn (W/O FTFD* event)	10%
Rider Churn (w/FTFD* event)	33%

**FTFD – Failed to find driver*

*** See Appendix A: 3. Programmatically calculating the input variables, for additional variables and assumptions made within this case*

Through detailed modeling and analysis, we have found the optimal take per ride to be $3.60 with a match rate of 86%, which results in a year-end net-revenue of $21,262 and a year-end profit of $9,251. Through our modeling, we have tested every possible *take* amount, ranging from $2.00 - $10.00, and have concluded that $3.60 will be the optimal price point to maximize both net-revenue and profit by year end.

Future Customer Quotes

Drivers

"While the idea of $19 per ride sounds good, the reality of waiting times and the occasional low demand makes it less appealing than I hoped. It's not as consistent as I'd like. I feel like I should get more for such a long ride, especially during rush hour."

- Jay Smith, Lyft Driver

"I thought airport rides would boost my earnings, but at only $19 dollars a ride, after factoring in the wait times and fuel costs, the compensation doesn't always seem worth it. I find myself reconsidering my strategy."

- Marty McGuire, Lyft Driver (Part-Time)

"Choosing to accept airport rides was a strategic move for me. The occasional steady demand and decent compensation make it a no-brainer. Plus, I get to meet interesting people, particularly business folks, who tip well!"

- Lynn Chamberlain, Lyft Driver

Riders

"As someone who travels frequently for work, using Lyft to get from downtown Toledo to the airport for $25 just makes sense. It's cost-effective, I can work during the ride, and I don't have to worry about parking. It's all about convenience and efficiency for me."

- Mandy Heanen, Businessperson

"I've considered using Lyft for my business trips, but the fixed $25 fare sometimes exceeds the cost of parking for quick trips. When I have back-to-back meetings across different cities, the economics of driving myself just make more sense."

- Alan Tremps, Businessperson

"I love the idea of using Lyft for ease, but the last thing I want is to miss my flight because I can't find a ride. It's happened before, and now I'm a bit hesitant, especially for early flights or during peak travel seasons."

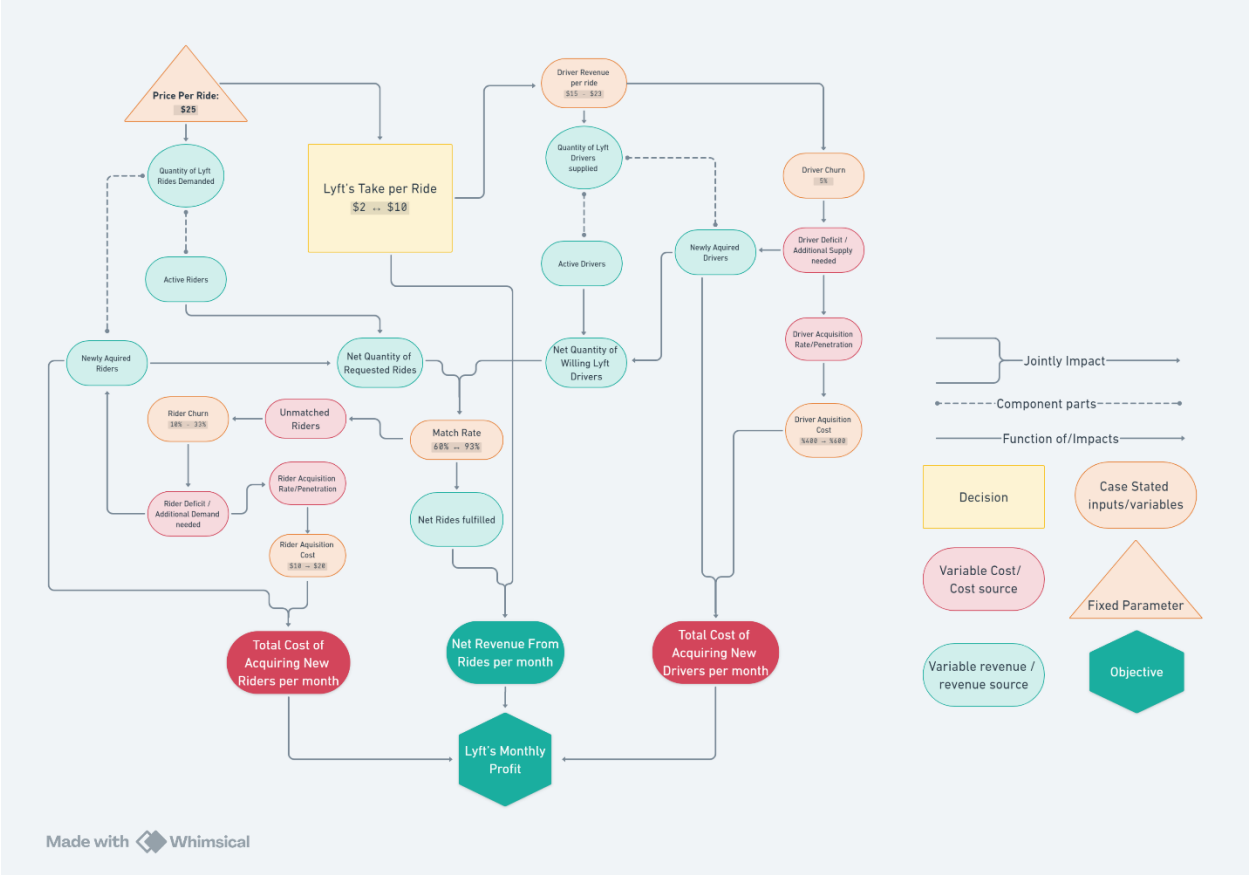
- Clark Hadly, Traveler

"Using Lyft for a $25 ride to the airport makes sense for my leisure trips. It's convenient and eliminates parking hassles. While it's just a ride, knowing the cost upfront helps with budgeting, making for a smooth start to my travels."

- Marc Torchio, Hopeful Clipboard Health recruit

Model Overview

Due to the shear amount of moving parts within this case, whether it's the effect of failed-to-find-driver events on churn, or the slight increases in CAC for each additional rider/driver acquired, it was important to thoroughly model out all the variables and their associated relationships. To do this effectively, we created a visual model pictured below:



Because this model has a lot of component parts, it was vital to showcase how each piece of this puzzle fits into the greater picture so to best gauge how changing Lyft's take will affect all the variables that go into determining Lyft's revenue, costs, and profit.

For a detailed step by step process of how this model was constructed see Appendix A. For the key takeaways from this model see below:

- The main lever we are pulling is Lyft's take which has a cascading effect both on the driver side (supply) and the rider side (demand)
- Net revenue is a function of both take and matched rides. The match rate changes based on the willingness of drivers to drive which is a function of driver revenue (Price – Lyft's take), as well as rides demanded, which is a function of the match rate and fluctuates based on the amount of failed to find driver events.
- Profit is a function of net revenue and the associated costs of acquiring new drivers and riders. For the sake of this model, we assume that Lyft works to re-acquire 100% of the last month's churned drivers, and is 75% successful at re-acquiring last month's churned riders.

Internal FAQ

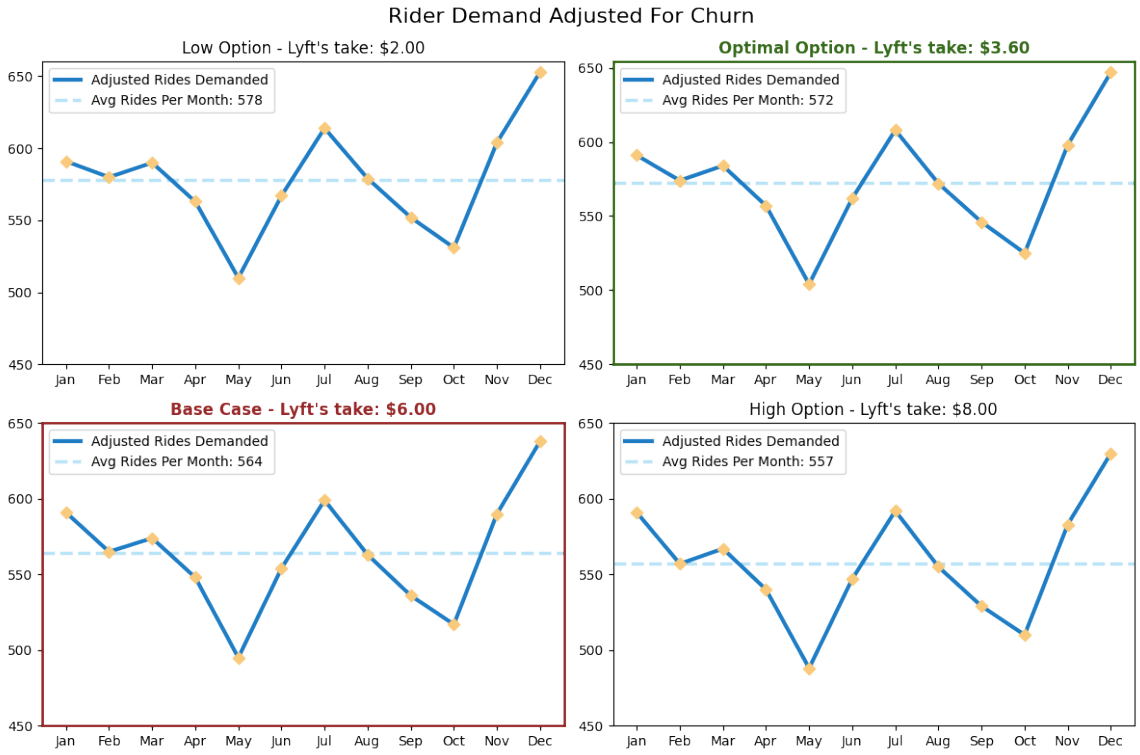
1. What problem are we solving?

Through pulling the lever that is Lyft's take we are in essence deciding the match rate. The big question is do we forego a high match rate for higher revenue per ride, or do we forego higher revenue per ride for a higher match rate. Based on our modeling, we found that a match rate of 86% and a take of $3.60 allowed us to maximize net revenue as well as profit by year's end.

However, beyond just pricing, we also must ask ourselves, "is a 86% match rate something we want to be known for?" Instead of just looking 12 months down the line, if we were to look 5 years out, would a 86% match rate end up biting us in the behind? Possibly – while the $3.60 price point does maximize net revenue and profit over the next 12 months, based on the increasing cost of acquisition, as well as the lack of good will from a mid-ranged match rate, we might expect consumers to choose a more reliable option (a match rate of >90%) in the long term. That said, for the objective in question, $3.60 remains the optimal price point over a 12 month time horizon, with a reasonable level of matched rides.

2. How does demand differ for different options?

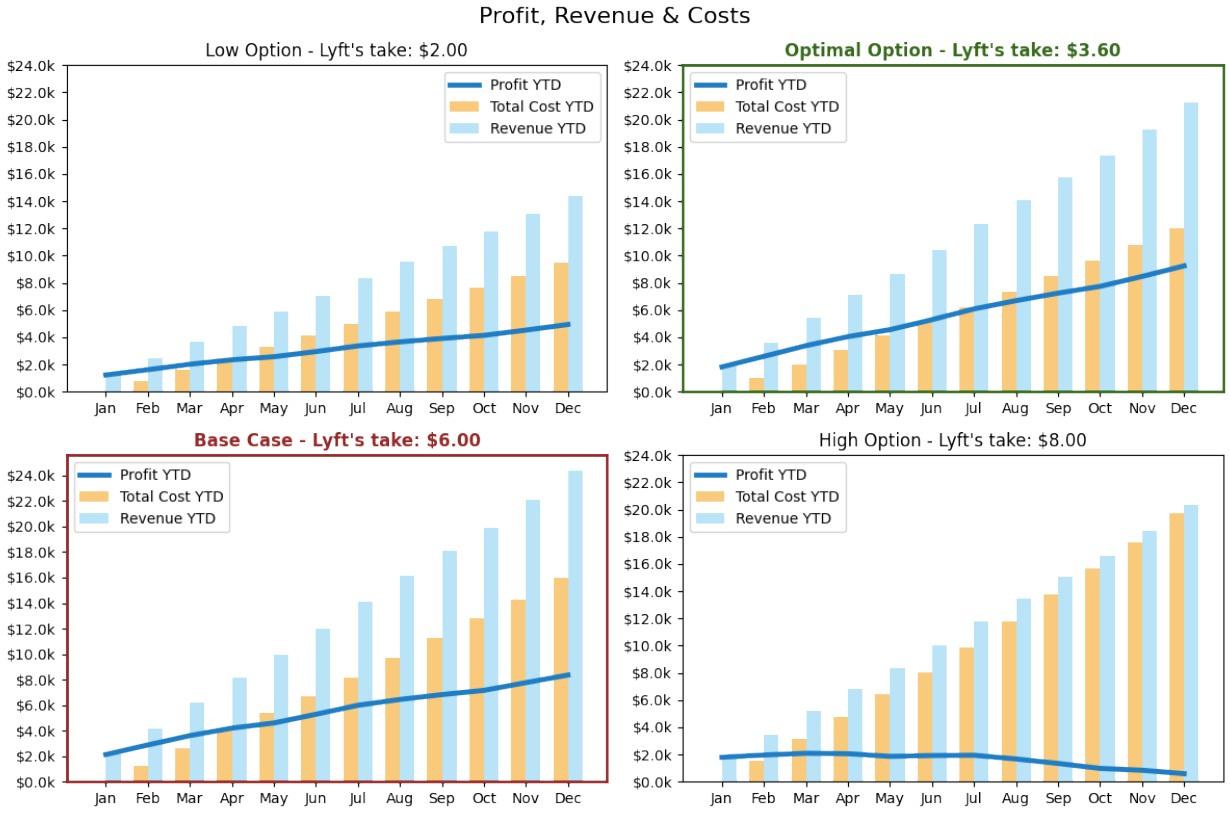
Demand for rides does indeed differ between values of Lyft's take, given the fact that a failed to find driver event (i.e. 1 – match rate) has a direct impact on the rider churn rate (33% churn vs a 10% churn). Thus, we would expect options with a higher take and lower match to have lower demand overtime, compared to a lower take and higher match rate. This can be illustrated with the plots below:



As you can see, the low option does indeed have the highest demand overtime, with the high option having the lowest demand overtime.

3. Should we just go with the low option then?

While this might look like a good option based on the quantity of rides demanded it is able to sustain, because we are working to maximize profit and net revenue over *the next 12 months*, a take of just $2.00, although it allows for a much higher match rate and fewer churn month to month, will not pay off over the 12-month timeline. To further demonstrate this, see the below option's revenue, cost, and profit plotted over time.



4. How much will the optimal option cost us in terms of CAC?

As you can see from the charts above, the base case of $6.00 and the optimal option of $3.60 have very similar net revenues. However, what allows for $3.60 to win out over time is the fact that is has a much lower cost due to less churn and smaller associated acquisition costs.

The total year end costs for the optimal option is $14,189.21.

5. Isn't this case supposed to optimize for net revenue, not profit?

Yes, indeed it is. However, we tested for both the optimal net revenue as well as profit and found that for both, the optimal price point remains $3.60. The profit optimization just adds costs to the equation which makes for a more thorough approach, as well as allowing us to account for the very real cost of losing customers overtime with a lower match rate.

6. Should we increase Price per ride for the riders?

This would be a viable strategy given the fact that there does seem to be a surplus of demand, given the failure to find driver events. Increasing price by, for example, $2.50 would indeed lower the quantity of rides demanded overall, but it would allow Lyft to maintain a healthy take, while also giving drivers a healthy per ride revenue. This dynamic would ensure the match rate remain high, allowing for Lyft to better serve the customers who are willing to pay the extra $2.50 or so dollars.

Other tactics involve surge prices – when demand is abnormally high, prices will temporarily increase, incentivizing drivers to accept more rides, thus increasing the match rate, mitigating a spike in failed to find driver events.

7. Other Recommendations?

Beyond maximizing net revenue, increasing loyalty through mitigating churn and providing a reliable rider experience should be a priority for Lyft in the long term. Beyond pulling the lever that is Lyft's take, we can also investigate other creative solutions:

- Allegiant is the sole airline operating flights to and from Toledo Express Airport. To encourage ongoing patronage, Lyft can explore a collaborative effort with Allegiant, particularly focusing on their credit card program. By forming a partnership, they could design a scheme where each dollar spent on Lyft is converted into points or miles through Allegiant's rewards system.

- Form a partnership with Toledo Express Airport to establish priority lanes for pick-ups and drop-offs, ensuring a fast and dependable estimated time of arrival for both riders and drivers.

    - Furthermore, reward frequent Lyft users by granting access to the Toledo Express Airport lounge to those who have accumulated a specified amount of Lyft miles.

Conclusion

In conclusion, the comprehensive analysis on the Lyft-off at Toledo Airport case presents a well-rounded pricing strategy for Lyft's new route between Toledo Express Airport and downtown Toledo. By meticulously evaluating various pricing points, considering both the supply and demand dynamics, and anticipating future customer perspectives, the optimal take per ride has been identified as $3.60 for the 12 month time horizon. This price point not only ensures a satisfactory match rate of 86% across 12 months, but also maximizes both net revenue and profit by the year's end, forecasting a net revenue of $21,262 and a profit of $9,251.

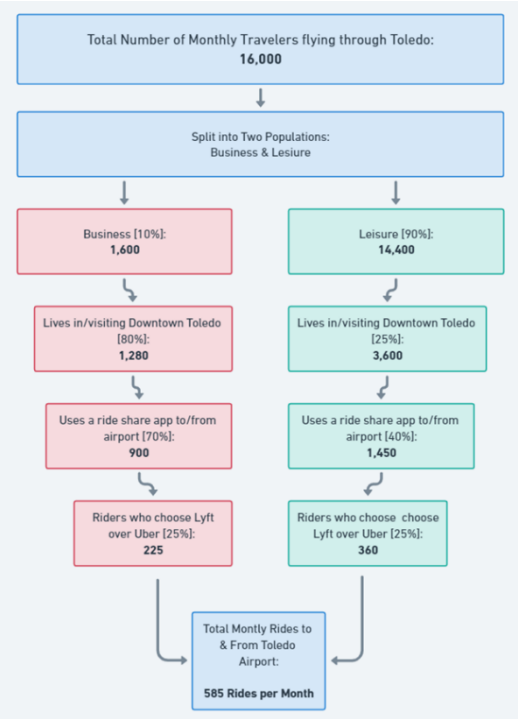
The analysis underscores the importance of balancing Lyft's take against driver and rider acquisition costs, match rates, and the potential for churn due to failed-to-find-driver events. The strategic recommendation to set the take at $3.60 is further validated by its potential to sustain a healthy driver and rider ecosystem while maintaining competitive profitability and service reliability over time. This nuanced approach considers the short-term financial goals of maximizing net revenue and profit, as well as the long-term objective of maintaining a loyal customer base through reliable match rates and satisfactory service experiences.

By addressing key concerns such as the cost of customer acquisition and the implications of varying match rates on demand, the analysis provides a holistic view of the factors influencing Lyft's pricing strategy. The executive summary, along with future customer quotes and a detailed model overview, paints a clear picture of the potential challenges and opportunities presented by the new route. This case study serves as a testament to the importance of data-driven decision-making in optimizing pricing strategies for transportation services, ensuring both profitability and customer satisfaction in a competitive market.

Appendix A: Key Steps in the modeling Process

1. Market Sizing – What is the quantity demanded for Lyft Rides

Given the fact that there is historically a surplus of demand as evident by the pricing experiment (a take of $6 is associated with a match rate of 60%; a take of $3 is associated with a match rate of 93%) we know that the supply of rides will be a function of the demand of rides. Thus, our first step in the process will be to determine the average quantity of rides demanded for the Toledo Express Airport ←→ Toledo downtown route.



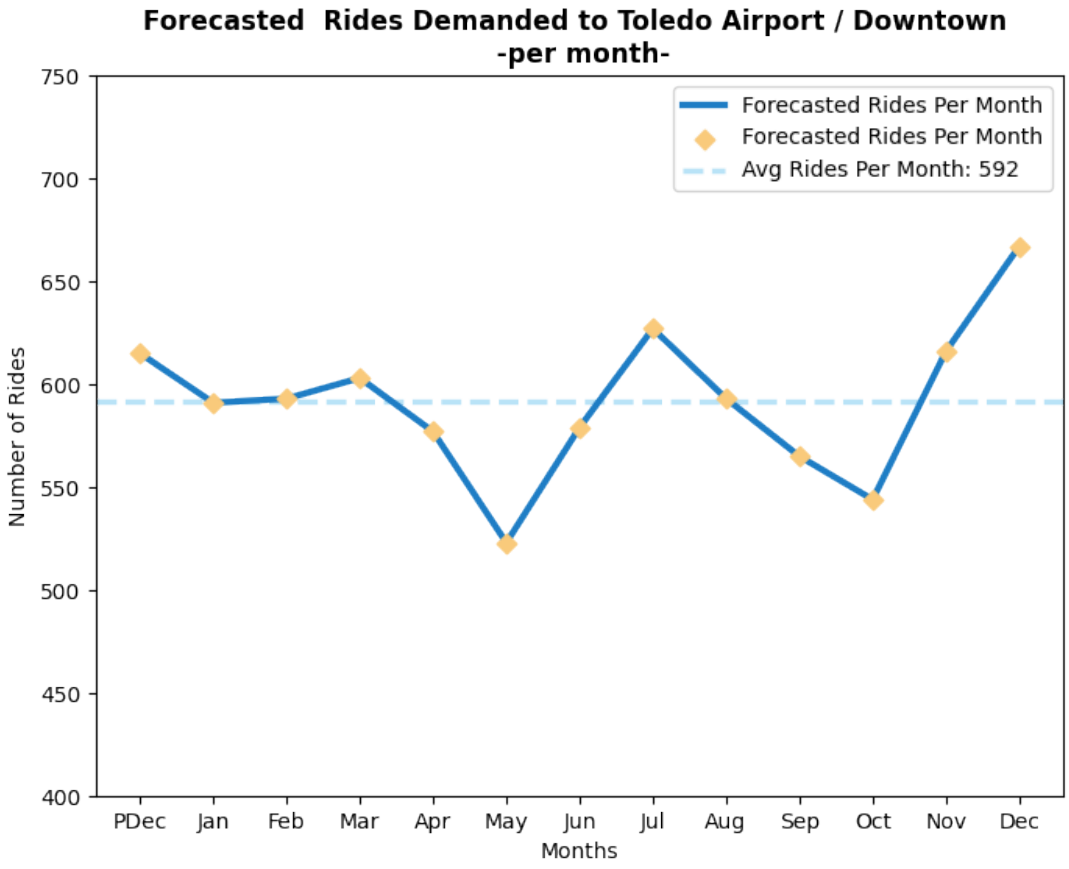
To do this, we leveraged a top-down market sizing approach using available data from Toledo Express Airport, Bloomberg, Census data and others (sources can be found in Appendix B)

The flow chart on the left demonstrates how the average monthly demand for Lyft rides to and from Toledo express airport was derived. Given the fact that Toledo Express Airport has an average of 16k travelers a month, we refined this number to a forecasted average monthly ride demand of 585. This number is in line with what we would expect given a smaller sized city like Toledo, with limited business and commercial attractions.

To replicate what demand might look like month to month, we utilized NumPy to generate random numbers along a normal distribution with a mean of 585, and a standard deviation of 10. We then incorporated seasonal factors to mimic the increases in travel associated with holidays and summer months. This code is seen below, keeping in mind that *total_rides_per_month* is equal to 585:

```
# Initalizing dataframe df with months and predicted rides requested (quantity demanded)
def rides_demanded_df():
    data = [] # List to accumulate data
    months = ['PDec','Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

    # Iteratring through all months to introduce seasonality factors and randomness into the forecasted rides demanded along a poisson distribution

    for month in months:
        num_rides = round(np.random.poisson(total_rides_per_month)) # lambda is assumed to be our predicted rides demanded per month
        if month in ['PDec','Mar', 'Dec','Nov']: # Incorpoating seasonality increase in avg travel of 8% for holidays months
            num_rides = num_rides * 1.08
        elif month in ['Jun', 'Jul', 'Aug']: # Incorpoating seasonality increase in avg travel of 5% for Summer months
            num_rides = num_rides * 1.025
        else:
            num_rides = num_rides * .95
        data.append({'Month': month, 'Number_of_rides': round(num_rides)})

    df = pd.DataFrame(data) # Construct DataFrame from accumulated data
    df.loc[1,'pm_rides'] = 0
    df.loc[2:,'pm_rides'] = df['Number_of_rides'].shift(1).fillna(0)
    return df


# Initializing base df
df = rides_demanded_df()
```

This code block resulted in the following demand for rides across the 12 months:



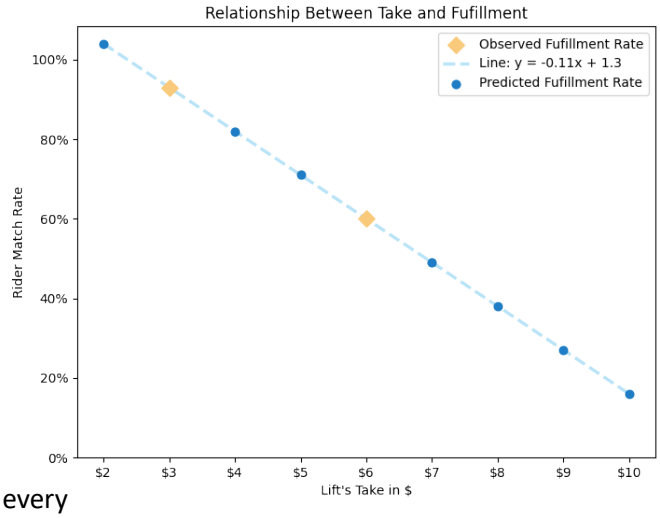
Keep in mind that this is forecasted demand assuming no churn. These levels of demand will deviate due to the fact that all of options will result in some churn to happen, both on the rider and driver side.

2. Modeling the relationship between Lyft's Take and the match rate

The next logical step in the process was to identify every possible take amount and its associated match rate. Using the data provided from the pricing experiment, we were able to model the relationship between take and match to be:

Match = -.11(take) + 1.3

We used this function to generate key value pairs (pictured below) to iterate through when testing each level of take within our Profit optimization model. Take values ranged from $2 to $10 within a .01 increase between values.



Both take and match values are the key inputs to almost every variable calculation within the profit optimization model.

```
# Creating a list of all possible values for Lyft's Take ranging from $2.00 to $10.00
lyfts_take = [i/10 for i in range(20,101, 1)]

# Function to create key value pairs between every take value and its associated match rate
def match_rate(lyfts_take):
    points = np.array([[6, .6], [3, 0.93]])  # Given points from pricing experiment
    m, b = np.polyfit(points[:, 0], points[:, 1], 1)
    # Using dictionary comprehension and vectorized operations
    return {take: round((take * m) + b, 2) for take in lyfts_take}

# Creating a list of key values pairs for lyft's take and its assoicated match rate based off the pricing experiment
take_and_match = match_rate(lyfts_take)
```

3. Programmatically calculating the input variables

The code used to calculate each input variable is pasted below. There were assumptions made at this juncture that are not local to the case prompt. Those assumptions are also summarized below:

```python
# Calculating the previous month's churned riders as a function of quantity rides demanded in the previous month and the match rate
def pm_churned_riders(demand, match_rate, norm_churn_rate=.1, FTFD_churn_rate=.33):
    FTFD_rate = 1 - match_rate
    # Logic to calculate churn based on provided formulas
    churned_base = (demand * FTFD_rate) * FTFD_churn_rate + (demand * match_rate) * norm_churn_rate
    return churned_base


# Calculating Rider Aquisition cost as a function of YTD aquistion and a starting rider CAC = $10
def adjusted_rider_CAC(row, rider_CAC = 10):
    if ((row['YTD_rider_churn'] * .0035) + rider_CAC)> 20:
        return 20
    else:
        sensitivity_adjustment = (row['YTD_rider_churn'] * .0035)
        new_CAC = sensitivity_adjustment + rider_CAC
        return new_CAC


# Calculating monthly drivers as a function of quantity rides demanded and the match rate
def monthly_drivers(Matched_rides, match_rate):
    airport_drivers = (Matched_rides / 4) / 3 # Calculating the number of unqiue Lyft drivers who on average complete 3 airport rides a week i.e. airport trips make up about ~12% of their total monthly average of completed trips (100)
    return airport_drivers


# Calculating the previous month's driver churn as a function of the driver churn rate and the previous month's airport driver count
def PM_churned_drivers(pm_monthly_drivers, lyft_take, churn_rate = .05):
    adjusted_churn_rate = churn_rate + ((lyft_take - 6) * .0065) # Variance as a function of Lyft's take is incorporated into the given driver churn rate of 5%. Function assumes Lyft's take will not exceed $10
    return round(adjusted_churn_rate * pm_monthly_drivers)


# Calculating driver aquistion cost as a function of YTD acquistion and a starting driver CAC = $400
def adjusted_driver_CAC(row, driver_cac = 400):
    if ((row['YTD_driver_churn'] * 3.5) + driver_CAC)> 600:
        return 600
    else:
        sensitivity_adjustment = (row['YTD_driver_churn'] * 3.5)# a 3.5 senstivity is an arbitrary number that insures driver CAC fluctuates in relation to the acquistion rate but increases within the range provided (400 - 600)
        new_CAC = sensitivity_adjustment + driver_CAC
        return new_CAC
```

Assumptions for Variable Calculations:

Variable	Assumption
Adjusted_rider_CAC	To incorporate the sensitivity factor on CAC given a certain rate of acquisition, we utilized - YTD_Riders_churn * .0035 - YTD_Drivers_churn * 3.5 as the sensitivity adjustments to price as these numbers allowed for reasonable variation in price given the ranges provided (Rider: $10 - $20; Driver: $400 - $600.
Pm_churned_drivers	Although we expect match rate to decrease as Lyft's take increases, we would also expect driver churn to increase with a decrease in driver revenue. To model this out we made the following assumption: - Adjusted_churn = churn_rate + ((take – 6) * .0065) This is an arbitrary function that simply allowed driver churn to fluctuate reasonably as driver revenue changes, and remain constant at the observed churn rate when price was set at $6.00
Monthly_drivers	Although we know drivers on average drive 100 rides a month, we would not assume all 100 rides are the airport rides. Thus we are assuming that around 12% of every drivers rides are a ride to/from the Toledo Airport.
Rider Acquisition success rate*	Lastly, we assumed Lyft is 75% successful in re-acquiring riders who churned last month: 1. Current rides demanded adjusted for churned drivers takes into account that 75% of churned drivers were re-acquired 2. Rider_CAC was also adjusted for the fact that we only re-acquired 75% of last months churned riders. Note – drivers were assumed to have a 100% re-acquisition rate, as price is the main driver for promotion, and there is a surplus of demand.

4. Running the profit optimization model

```python
# Running all functions to establish a comprehensive dataframe df
def create_df(df, match_rate, take):

    # Calculating adjusted riders demanded
    df['pm_churned_riders'] = round(pm_churned_riders(df['pm_rides'],match_rate))
    df['Adjusted_rides_demanded'] = round(df['Number_of_rides'] - (df['pm_churned_riders']*.25))

    # Calculating rider side variables
    df['YTD_rider_churn'] = df['pm_churned_riders'].cumsum()
    df['Adjusted_rider_CAC'] = df.apply(adjusted_rider_CAC,axis=1)
    df['Monthly_Rider_CAC'] = round((df['Adjusted_rider_CAC'] * (df['pm_churned_riders']*.75)),2)

    # Supply Variables:
    df['Matched_rides'] = round(df['Adjusted_rides_demanded'] * match_rate)
    df['Monthly_drivers'] = df['Matched_rides'].apply(monthly_drivers, args = (match_rate,))
    df['pm_monthly_drivers'] = df['Monthly_drivers'].shift(1).fillna(0)
    df['pm_churned_drivers'] = df['pm_monthly_drivers'].apply(PM_churned_drivers, args=(take,))
    df['YTD_driver_churn'] = df['pm_churned_drivers'].cumsum()
    df['Adjusted_driver_CAC'] = df.apply(adjusted_driver_CAC,axis=1)
    df['Monthly_Driver_CAC'] = round((df['Adjusted_driver_CAC'] * df['pm_churned_drivers']),2)

    # Objective variables
    df['Net_Revenue'] = round(take * df['Matched_rides'],2)
    df['Profit'] = round(df['Net_Revenue'] - (df['Monthly_Driver_CAC'] + df['Monthly_Rider_CAC']),2)

    return df[1:] # Returning the df without the previous year's december month

# Running main function to optimzie for net revenue / profit
def optimize_revenue(df):

    Net_Revenue = 0
    Profit = 0
    optimized_take = 0
    optimized_match_rate = 0

    for take, match_rate in take_and_match.items():
        initial_df = create_df(df, match_rate, take)

        if initial_df['Profit'].sum() > Profit:
            Net_Revenue = initial_df['Net_Revenue'].sum()
            Profit = initial_df['Profit'].sum()
            optimized_take = take
            optimized_match_rate = match_rate
            final_df = initial_df

        else:
            continue

    return final_df, round(Net_revenue), round(Profit), optimized_take, optimized_match_rate
```

This model builds upon every step thus far and is composed of two functions to firstly build out a data frame for each match rate and take combo, and then tests to see which combo results in the highest net revenue and profit. An example data frame output for the optimal option: $3.60 is provided below as well as included in this submission: final_df.csv.

Month	Number_of_rides	pm_rides	Adjusted_rides_demanded	pm_churned_riders	YTD_rider_churn	Adjusted_rider_CAC	Monthly_Rider_CAC	Matched_rides	Monthly_drivers	pm_monthly_drivers	pm_churned_drivers	YTD_driver_churn	Adjusted_driver_CAC	Monthly_Driver_CAC	Net_Revenue	Profit
Jan	542	0	542	0	0	10	0	417	34.75	0	0	0	400	0	1876.5	1876.5
Feb	556	542	535	83	83	10.2905	640.58	412	34.33333333	34.75	1	1	403.5	403.5	1854	809.92
Mar	639	556	618	85	168	10.588	674.98	476	39.66666667	34.33333333	1	2	407	407	2142	1060.02
Apr	558	639	534	98	266	10.931	803.43	411	34.25	39.66666667	2	4	414	828	1849.5	218.07
May	543	558	522	85	351	11.2285	715.82	402	33.5	34.25	1	5	417.5	417.5	1809	675.68
Jun	593	543	572	83	434	11.519	717.06	440	36.66666667	33.5	1	6	421	421	1980	841.94
Jul	603	593	580	91	525	11.8375	807.91	447	37.25	36.66666667	1	7	424.5	424.5	2011.5	779.09
Aug	594	603	571	92	617	12.1595	839.01	440	36.66666667	37.25	1	8	428	428	1980	712.99
Sep	549	594	526	91	708	12.478	851.62	405	33.75	36.66666667	1	9	431.5	431.5	1822.5	539.38
Oct	560	549	539	84	792	12.772	804.64	415	34.58333333	33.75	1	10	435	435	1867.5	627.86
Nov	626	560	604	86	878	13.073	843.21	465	38.75	34.58333333	1	11	438.5	438.5	2092.5	810.79
Dec	629	626	605	96	974	13.409	965.45	466	38.83333333	38.75	2	13	445.5	891	2097	240.55

Appendix B: Sources

- Toledo Express Airports Passenger Traffic Segmentation
- Population of Toledo, OH
- Percent of Travelers who Use ride-share
- Lyft Marketshare in the US