BAR DOUGH
HELL'S KITCHEN NYC

BAR DOUGH'S DATA-DRIVEN INSIGHTS

Adrian Valentine, Liam Salmon, and Sigfús Árnason

TAKEAWAYS FROM INITIAL MEETING

1

HAPPY HOUR MENU ALTERATION ANALYSIS:

- Time-Series Analysis: Examines the influence of happy hour on overall sales.
- Discount Impact Analysis: Evaluates the effectiveness and true value of happy hour promotions.

2

OPERATIONAL HOURS RECONSTRUCTION

- Late Night Sales Analysis: Sales trends during standard dinner hours versus late-night operations.
- Cost-Benefit Analysis: Assesses the profitability of operating the kitchen during late hours.
- Labor Optimization Strategy: Examines workforce allocation to maximize efficiency in relation to sales trends.

MANAGING OUT DATASET: TOAST

WHAT WE GOT

Full access to Toast

THE PROBLEM

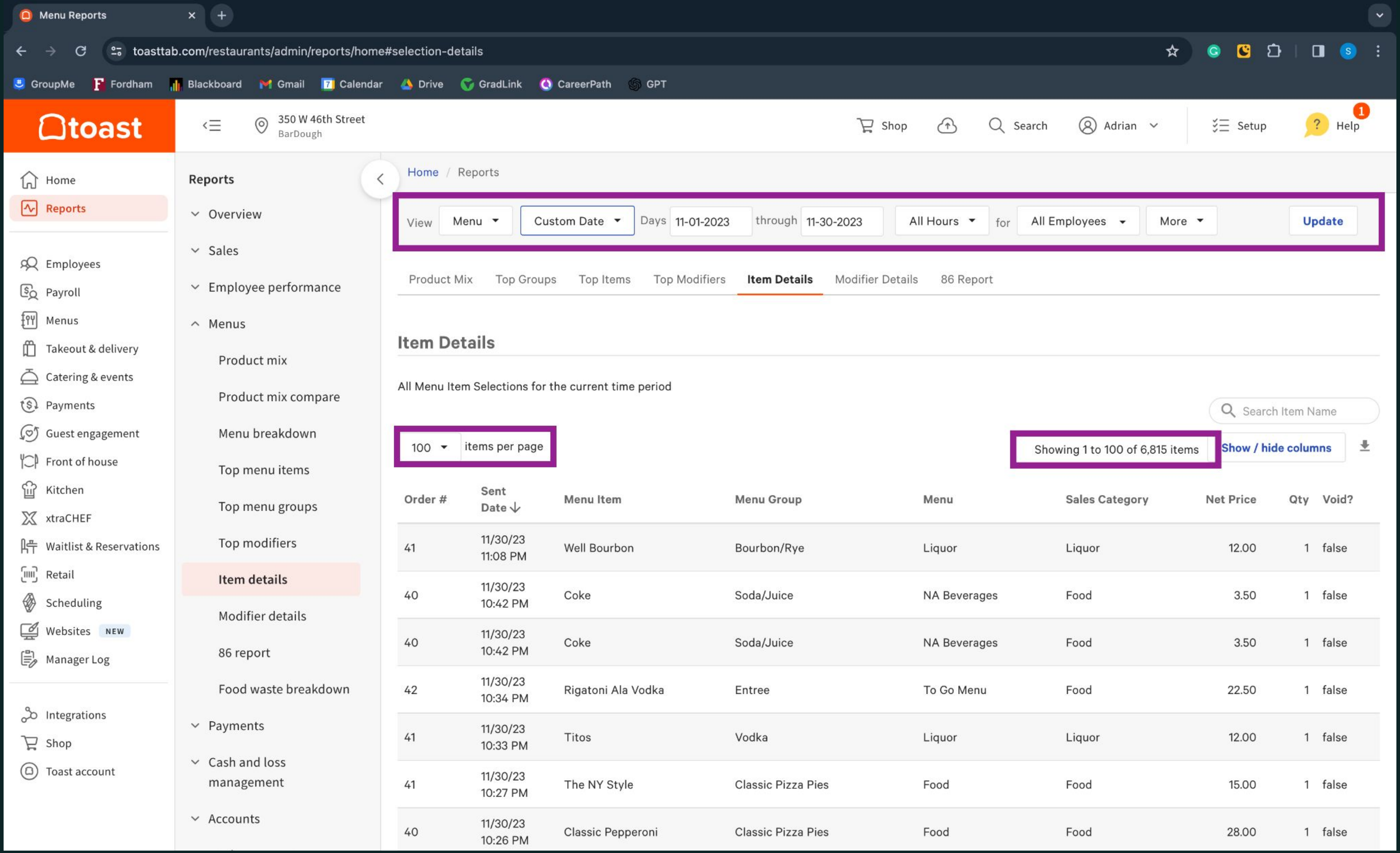
Restriction of Data Extraction

THE SOLUTION

Selenium

OUTPUT

Two Extensive Data Sets
- 68,000 & 2,300 entries



View: Menu | Custom Date | Days 11-01-2023 through 11-30-2023 | All Hours | for All Employees | More

Item Details

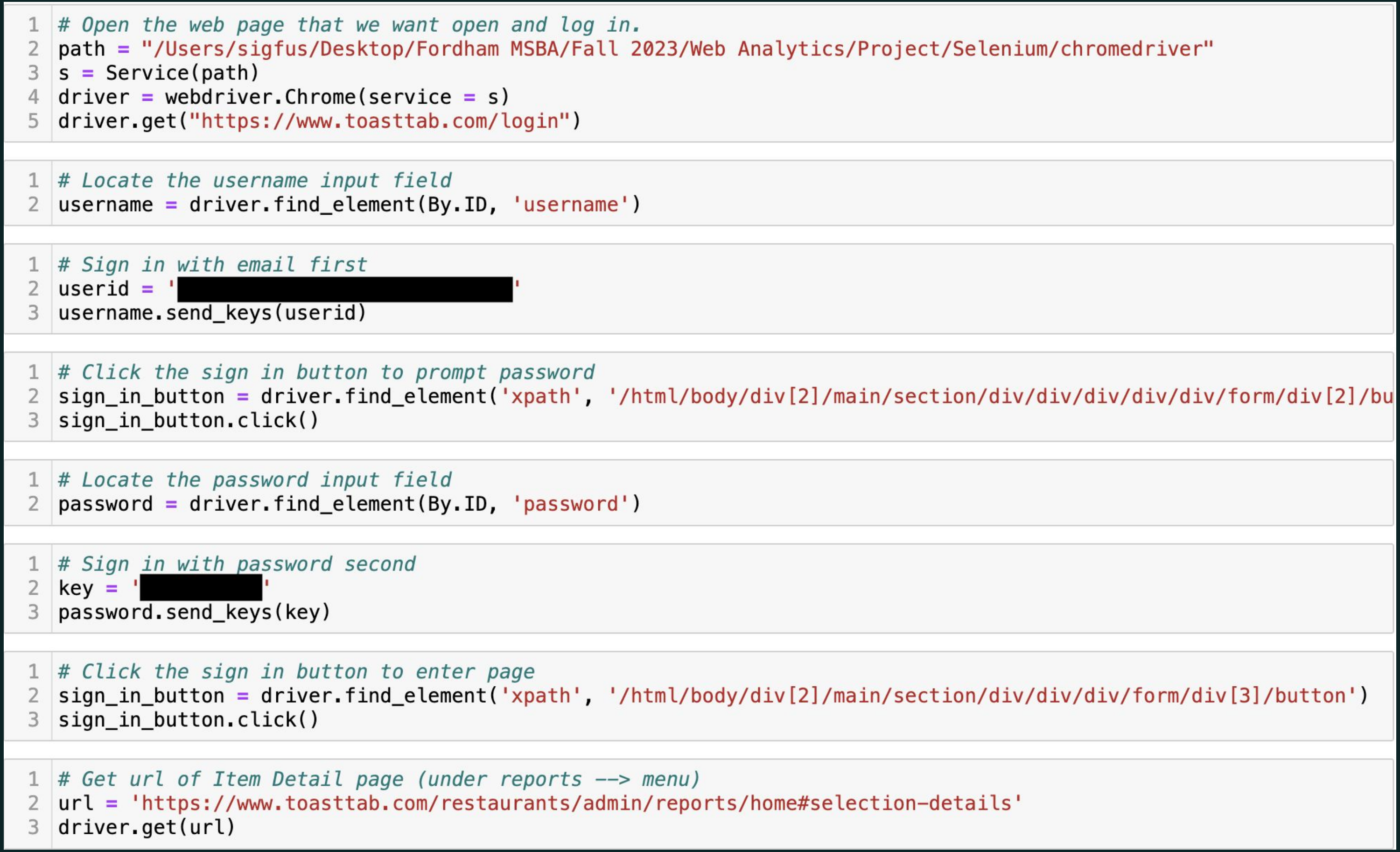
All Menu Item Selections for the current time period

100 items per page

Showing 1 to 100 of 6,815 items

Order #	Sent Date	Menu Item	Menu Group	Menu	Sales Category	Net Price	Qty	Void?
41	11/30/23 11:08 PM	Well Bourbon	Bourbon/Rye	Liquor	Liquor	12.00	1	false
40	11/30/23 10:42 PM	Coke	Soda/Juice	NA Beverages	Food	3.50	1	false
40	11/30/23 10:42 PM	Coke	Soda/Juice	NA Beverages	Food	3.50	1	false
42	11/30/23 10:34 PM	Rigatoni Ala Vodka	Entree	To Go Menu	Food	22.50	1	false
41	11/30/23 10:33 PM	Titos	Vodka	Liquor	Liquor	12.00	1	false
41	11/30/23 10:27 PM	The NY Style	Classic Pizza Pies	Food	Food	15.00	1	false
40	11/30/23 10:26 PM	Classic Pepperoni	Classic Pizza Pies	Food	Food	28.00	1	false

SELENIUM STEP 1: LOGGING INTO ACCOUNT



```
# Open the web page that we want open and log in.
path = "/Users/sigfus/Desktop/Fordham MSBA/Fall 2023/Web Analytics/Project/Selenium/chromedriver"
s = Service(path)
driver = webdriver.Chrome(service = s)
driver.get("https://www.toasttab.com/login")
```

```
# Locate the username input field
username = driver.find_element(By.ID, 'username')
```

```
# Sign in with email first
userid = '[redacted]'
username.send_keys(userid)
```

```
# Click the sign in button to prompt password
sign_in_button = driver.find_element('xpath', '/html/body/div[2]/main/section/div/div/div/div/div/form/div[2]/bu
sign_in_button.click()
```

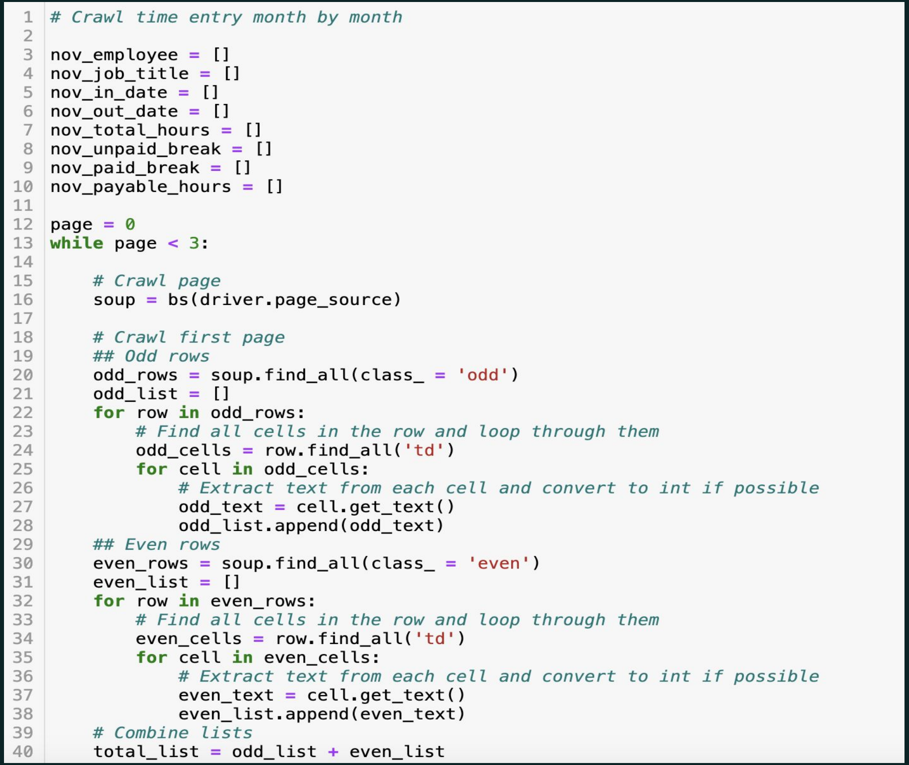
```
# Locate the password input field
password = driver.find_element(By.ID, 'password')
```

```
# Sign in with password second
key = '[redacted]'
password.send_keys(key)
```

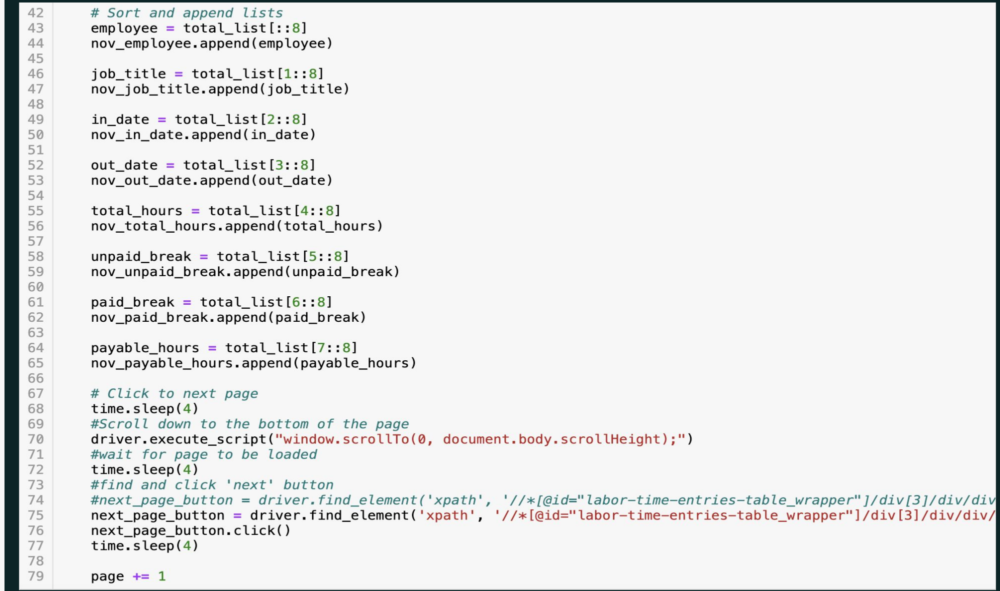
```
# Click the sign in button to enter page
sign_in_button = driver.find_element('xpath', '/html/body/div[2]/main/section/div/div/div/form/div[3]/button')
sign_in_button.click()
```

```
# Get url of Item Detail page (under reports --> menu)
url = 'https://www.toasttab.com/restaurants/admin/reports/home#selection-details'
driver.get(url)
```

SELENIUM STEP 2: RUNNING AUTOMATED CRAWLING FUNCTION MONTH BY MONTH

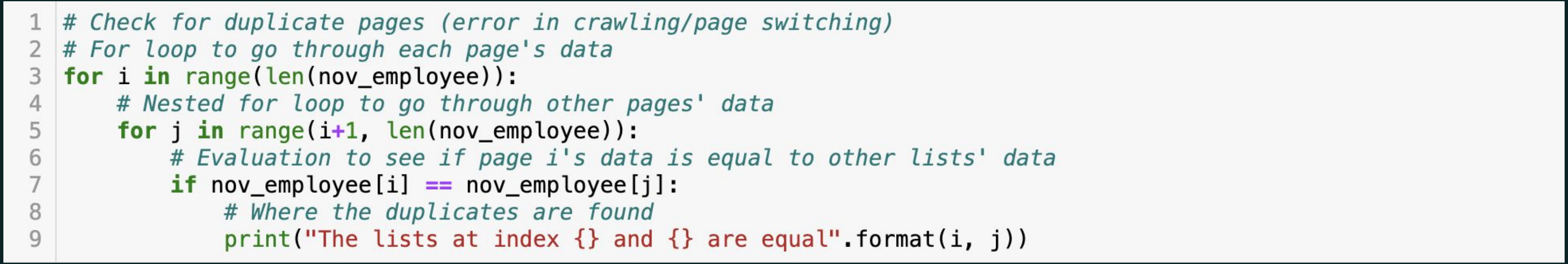


```python
# Crawl time entry month by month

nov_employee = []
nov_job_title = []
nov_in_date = []
nov_out_date = []
nov_total_hours = []
nov_unpaid_break = []
nov_paid_break = []
nov_payable_hours = []

page = 0
while page < 3:

    # Crawl page
    soup = bs(driver.page_source)

    # Crawl first page
    ## Odd rows
    odd_rows = soup.find_all(class_ = 'odd')
    odd_list = []
    for row in odd_rows:
        # Find all cells in the row and loop through them
        odd_cells = row.find_all('td')
        for cell in odd_cells:
            # Extract text from each cell and convert to int if possible
            odd_text = cell.get_text()
            odd_list.append(odd_text)
    ## Even rows
    even_rows = soup.find_all(class_ = 'even')
    even_list = []
    for row in even_rows:
        # Find all cells in the row and loop through them
        even_cells = row.find_all('td')
        for cell in even_cells:
            # Extract text from each cell and convert to int if possible
            even_text = cell.get_text()
            even_list.append(even_text)
    # Combine lists
    total_list = odd_list + even_list
```



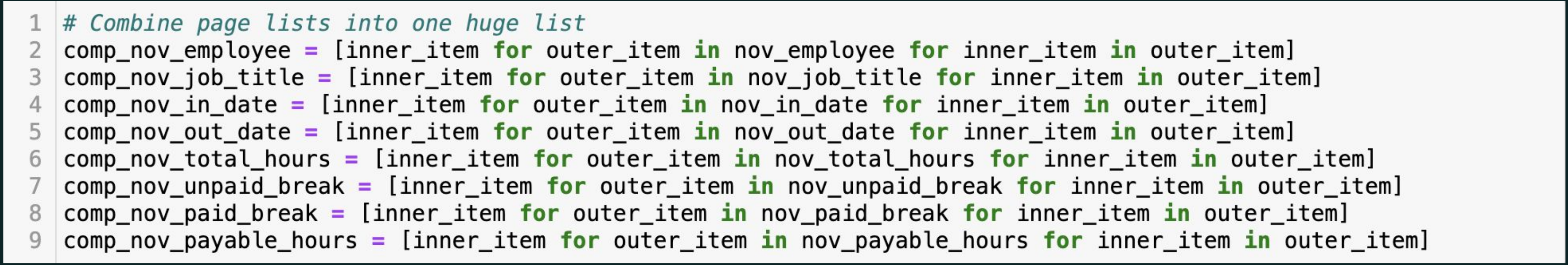
```python
    # Sort and append lists
    employee = total_list[::8]
    nov_employee.append(employee)

    job_title = total_list[1::8]
    nov_job_title.append(job_title)

    in_date = total_list[2::8]
    nov_in_date.append(in_date)

    out_date = total_list[3::8]
    nov_out_date.append(out_date)

    total_hours = total_list[4::8]
    nov_total_hours.append(total_hours)

    unpaid_break = total_list[5::8]
    nov_unpaid_break.append(unpaid_break)

    paid_break = total_list[6::8]
    nov_paid_break.append(paid_break)

    payable_hours = total_list[7::8]
    nov_payable_hours.append(payable_hours)

    # Click to next page
    time.sleep(4)
    #Scroll down to the bottom of the page
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    #wait for page to be loaded
    time.sleep(4)
    #find and click 'next' button
    #next_page_button = driver.find_element('xpath', '//*[@id="labor-time-entries-table_wrapper"]/div[3]/div/div
    next_page_button = driver.find_element('xpath', '//*[@id="labor-time-entries-table_wrapper"]/div[3]/div/div/
    next_page_button.click()
    time.sleep(4)

    page += 1
```

SELENIUM STEP 3: CHECKING FOR DUPLICATE LISTS (RE-RUN IF DUPLICATE)



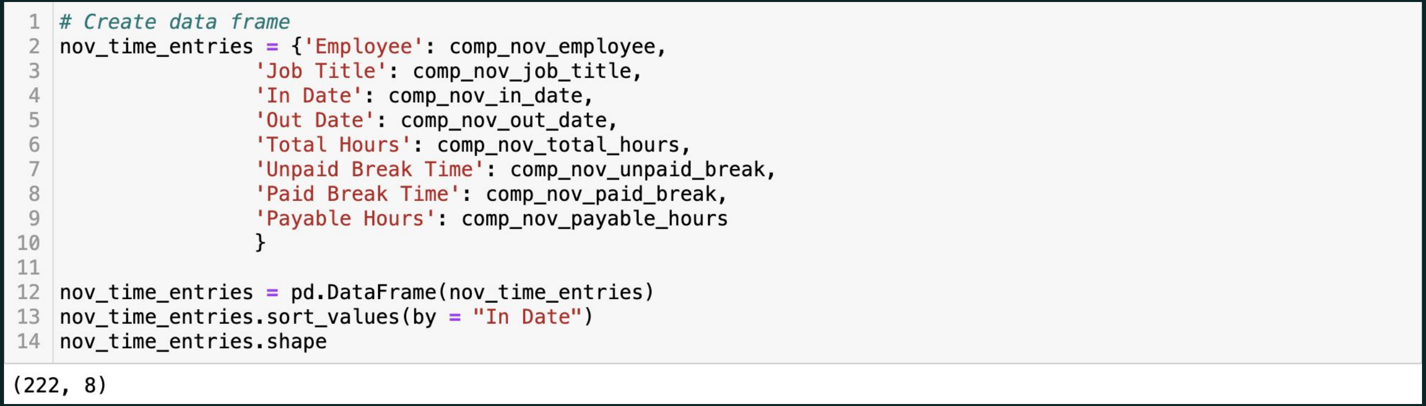
```python
# Check for duplicate pages (error in crawling/page switching)
# For loop to go through each page's data
for i in range(len(nov_employee)):
    # Nested for loop to go through other pages' data
    for j in range(i+1, len(nov_employee)):
        # Evaluation to see if page i's data is equal to other lists' data
        if nov_employee[i] == nov_employee[j]:
            # Where the duplicates are found
            print("The lists at index {} and {} are equal".format(i, j))
```

SELENIUM STEP 4: EXPANDING PAGE LISTS INTO MONTHLY LIST



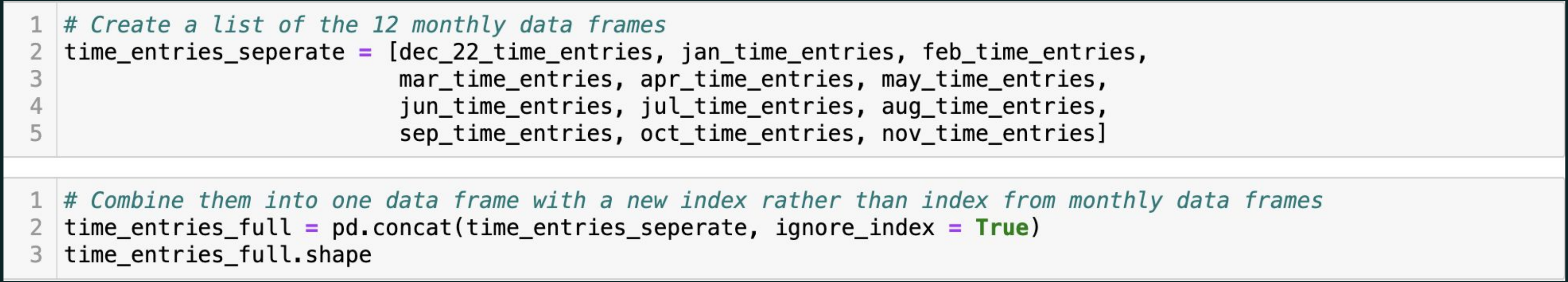
```python
# Combine page lists into one huge list
comp_nov_employee = [inner_item for outer_item in nov_employee for inner_item in outer_item]
comp_nov_job_title = [inner_item for outer_item in nov_job_title for inner_item in outer_item]
comp_nov_in_date = [inner_item for outer_item in nov_in_date for inner_item in outer_item]
comp_nov_out_date = [inner_item for outer_item in nov_out_date for inner_item in outer_item]
comp_nov_total_hours = [inner_item for outer_item in nov_total_hours for inner_item in outer_item]
comp_nov_unpaid_break = [inner_item for outer_item in nov_unpaid_break for inner_item in outer_item]
comp_nov_paid_break = [inner_item for outer_item in nov_paid_break for inner_item in outer_item]
comp_nov_payable_hours = [inner_item for outer_item in nov_payable_hours for inner_item in outer_item]
```

SELENIUM STEP 5: CREATING MONTHLY AND FULL YEAR DATA FRAMES



```
# Create data frame
nov_time_entries = {'Employee': comp_nov_employee,
                    'Job Title': comp_nov_job_title,
                    'In Date': comp_nov_in_date,
                    'Out Date': comp_nov_out_date,
                    'Total Hours': comp_nov_total_hours,
                    'Unpaid Break Time': comp_nov_unpaid_break,
                    'Paid Break Time': comp_nov_paid_break,
                    'Payable Hours': comp_nov_payable_hours
                    }

nov_time_entries = pd.DataFrame(nov_time_entries)
nov_time_entries.sort_values(by = "In Date")
nov_time_entries.shape
```

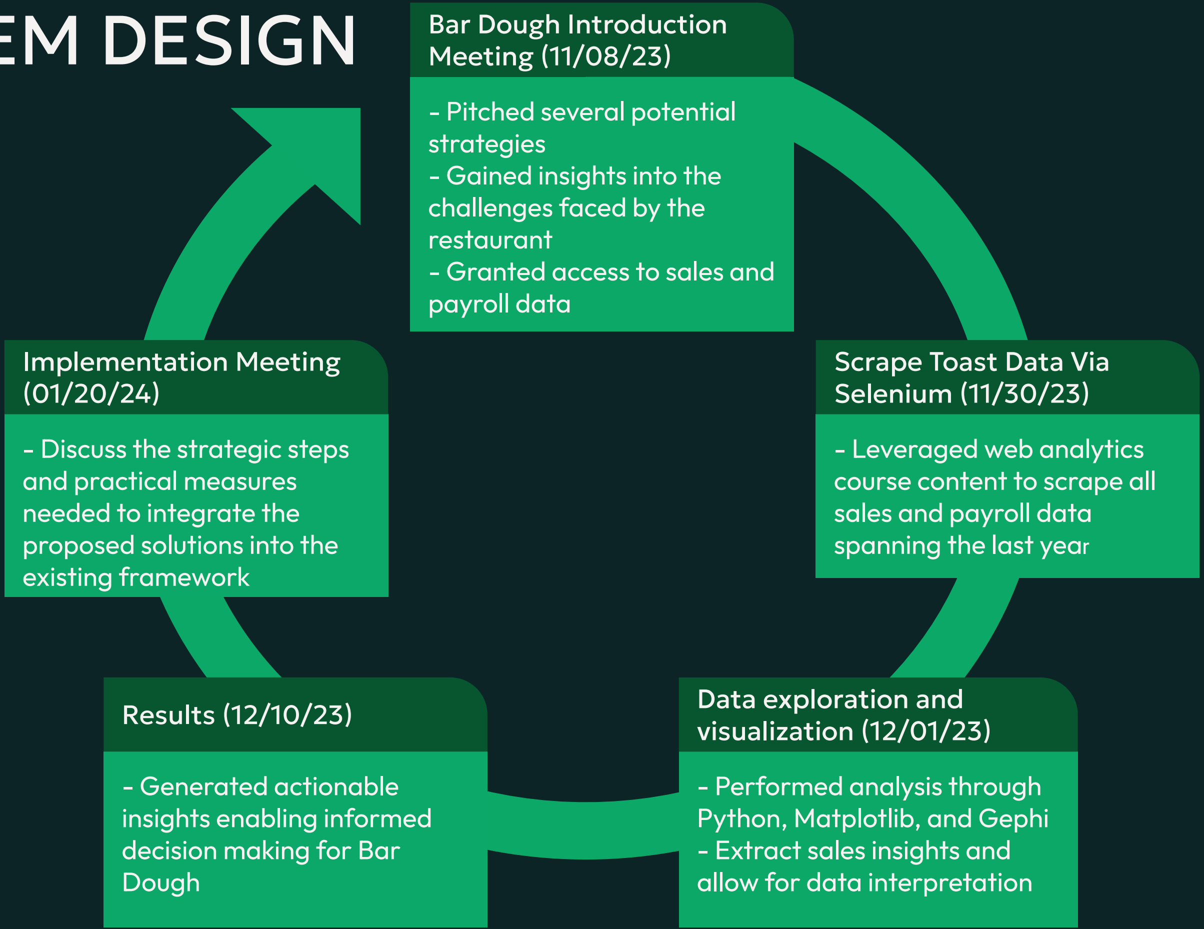
```
(222, 8)
```



```
# Create a list of the 12 monthly data frames
time_entries_seperate = [dec_22_time_entries, jan_time_entries, feb_time_entries,
                         mar_time_entries, apr_time_entries, may_time_entries,
                         jun_time_entries, jul_time_entries, aug_time_entries,
                         sep_time_entries, oct_time_entries, nov_time_entries]
```

```
# Combine them into one data frame with a new index rather than index from monthly data frames
time_entries_full = pd.concat(time_entries_seperate, ignore_index = True)
time_entries_full.shape
```

SYSTEM DESIGN



Bar Dough Introduction Meeting (11/08/23)

- Pitched several potential strategies
- Gained insights into the challenges faced by the restaurant
- Granted access to sales and payroll data

Scrape Toast Data Via Selenium (11/30/23)

- Leveraged web analytics course content to scrape all sales and payroll data spanning the last year

Data exploration and visualization (12/01/23)

- Performed analysis through Python, Matplotlib, and Gephi
- Extract sales insights and allow for data interpretation

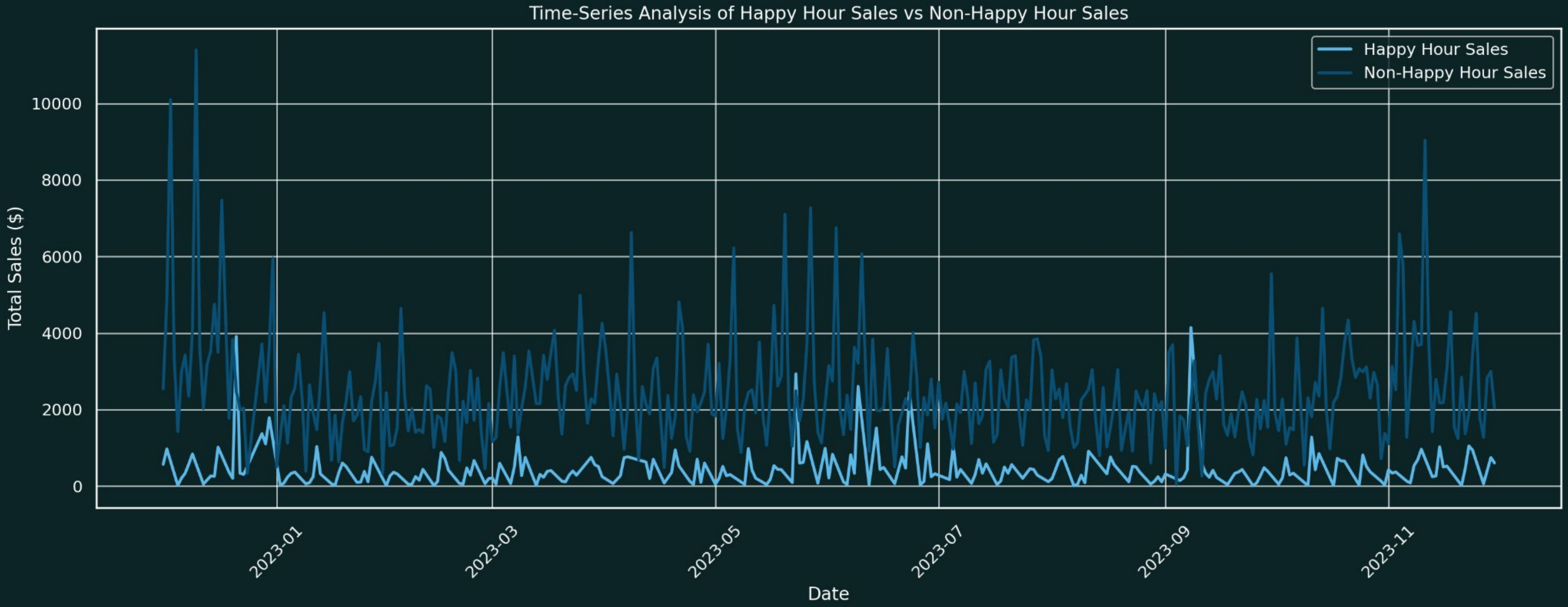
Results (12/10/23)

- Generated actionable insights enabling informed decision making for Bar Dough

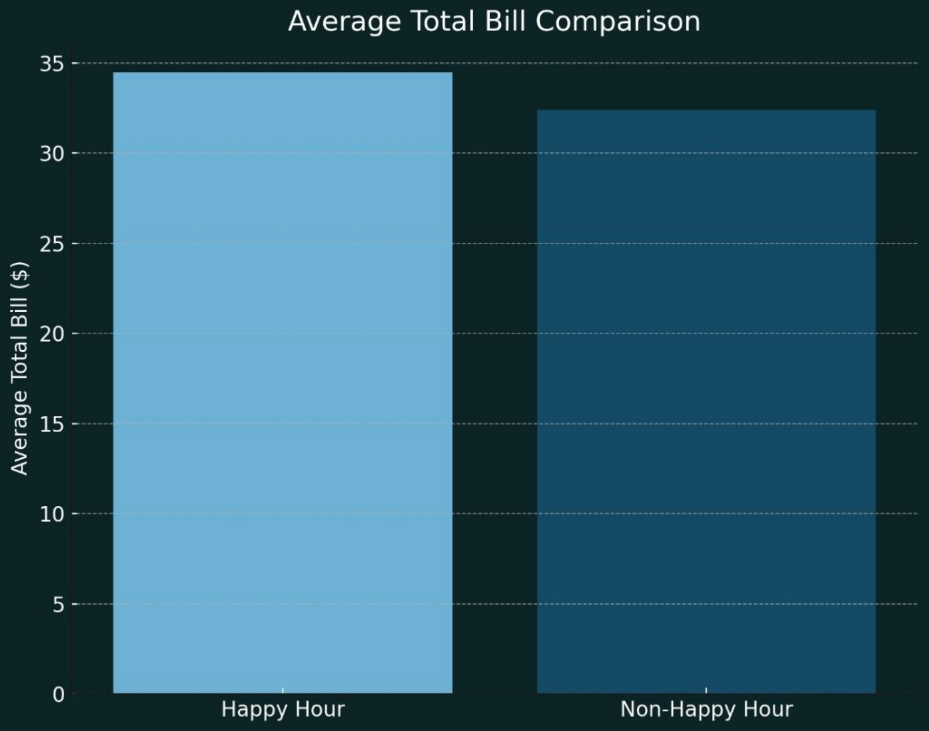
Implementation Meeting (01/20/24)

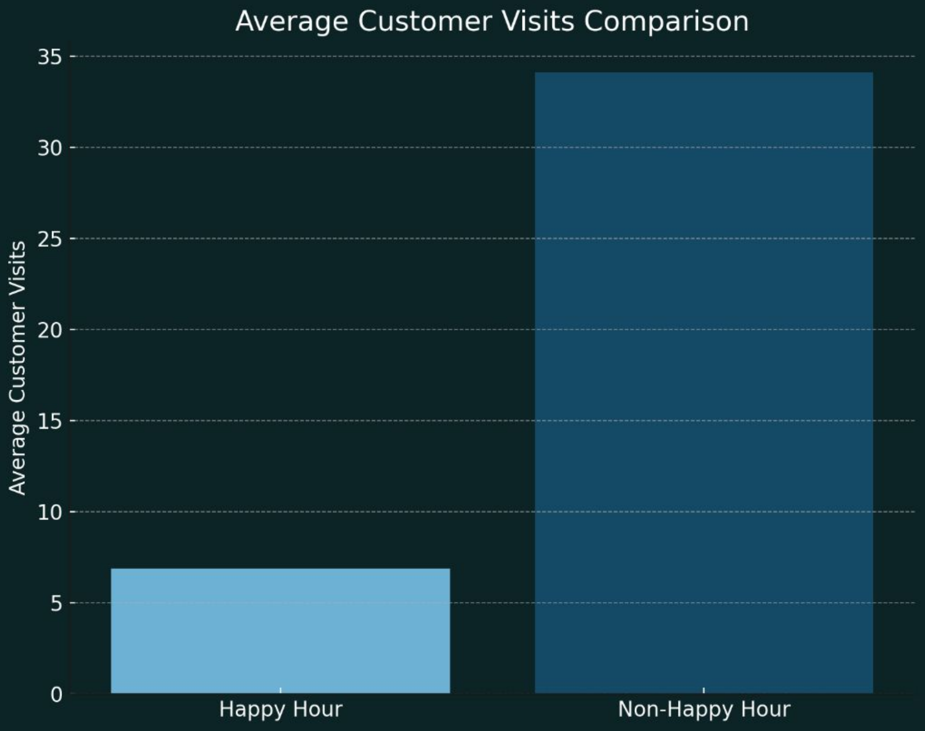
- Discuss the strategic steps and practical measures needed to integrate the proposed solutions into the existing framework

HAPPY HOUR MENU OPTIMIZATION

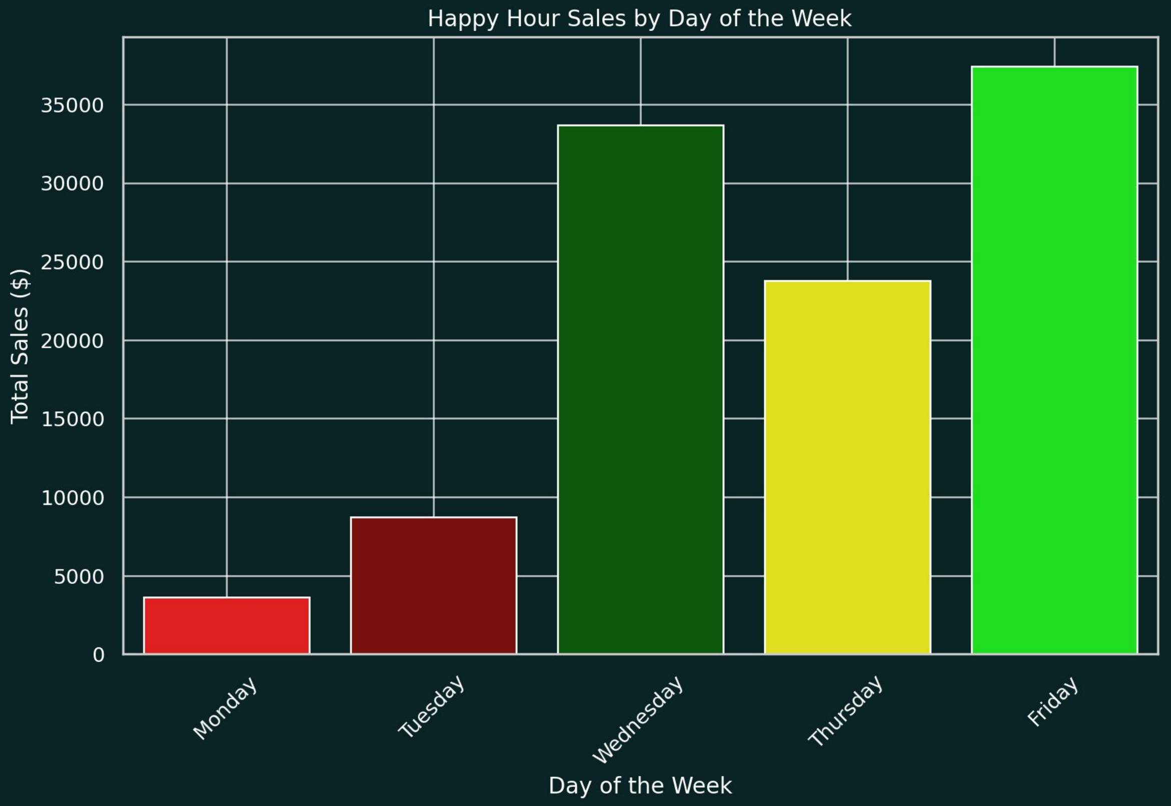


HAPPY HOUR MENU OPTIMIZATION





HAPPY HOUR MENU OPTIMIZATION



HAPPY HOUR MENU OPTIMIZATION

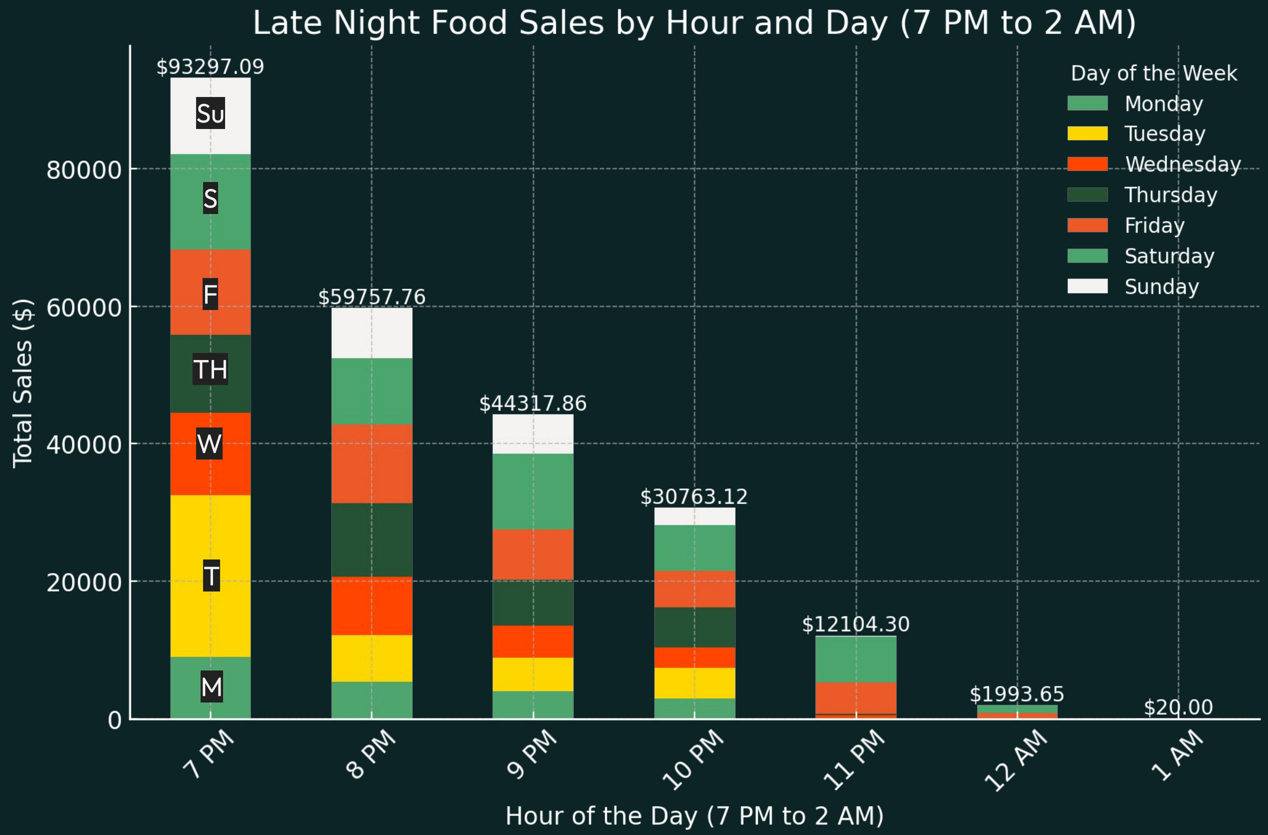
ANALYSIS

- Average Total Bill During Happy Hour: **$34.49**
- Average Total Bill Outside of Happy Hour: **$32.40**
- Average Customer Visits During Happy Hour: **6.90 visits per day**
- Average Customer Visits Outside of Happy Hour: **34.14 visits per day**
- Monday & Tuesday low HH sales.

RECOMMENDATIONS

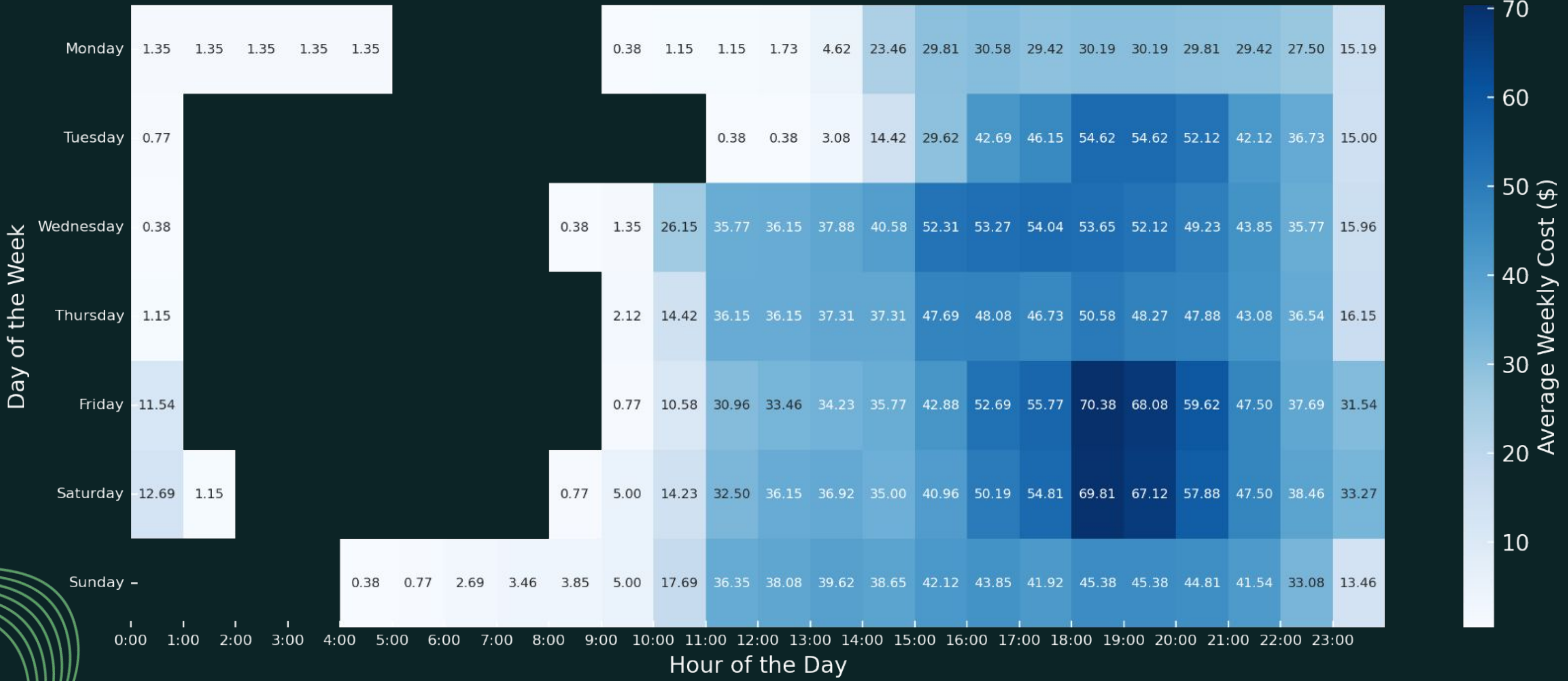
- **Enhance Promotion Visibility:** Increase awareness of Happy Hour promotions.
- **Review Discount Strategy**: Consider altering the discount structure & or HH menu.
- **Special Happy Hour Events:** Create special events or themes during Happy Hour to attract more customers, focused on Monday & Tuesday (4-9pm). Leverage NYC i.e. Sporting events, Broadway shows etc.

OPERATIONAL HOURS RECONSTRUCTION



OPERATIONAL HOURS RECONSTRUCTION

Average Weekly Staff Cost Per Hour for Bussers, Cooks, and Chefs by Day of the Week



	0:00	1:00	2:00	3:00	4:00	5:00	6:00	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00
Monday	1.35	1.35	1.35	1.35	1.35					0.38	1.15	1.15	1.73	4.62	23.46	29.81	30.58	29.42	30.19	30.19	29.81	29.42	27.50	15.19
Tuesday	0.77											0.38	0.38	3.08	14.42	29.62	42.69	46.15	54.62	54.62	52.12	42.12	36.73	15.00
Wednesday	0.38								0.38	1.35	26.15	35.77	36.15	37.88	40.58	52.31	53.27	54.04	53.65	52.12	49.23	43.85	35.77	15.96
Thursday	1.15									2.12	14.42	36.15	36.15	37.31	37.31	47.69	48.08	46.73	50.58	48.27	47.88	43.08	36.54	16.15
Friday	11.54									0.77	10.58	30.96	33.46	34.23	35.77	42.88	52.69	55.77	70.38	68.08	59.62	47.50	37.69	31.54
Saturday	12.69	1.15							0.77	5.00	14.23	32.50	36.15	36.92	35.00	40.96	50.19	54.81	69.81	67.12	57.88	47.50	38.46	33.27
Sunday					0.38	0.77	2.69	3.46	3.85	5.00	17.69	36.35	38.08	39.62	38.65	42.12	43.85	41.92	45.38	45.38	44.81	41.54	33.08	13.46

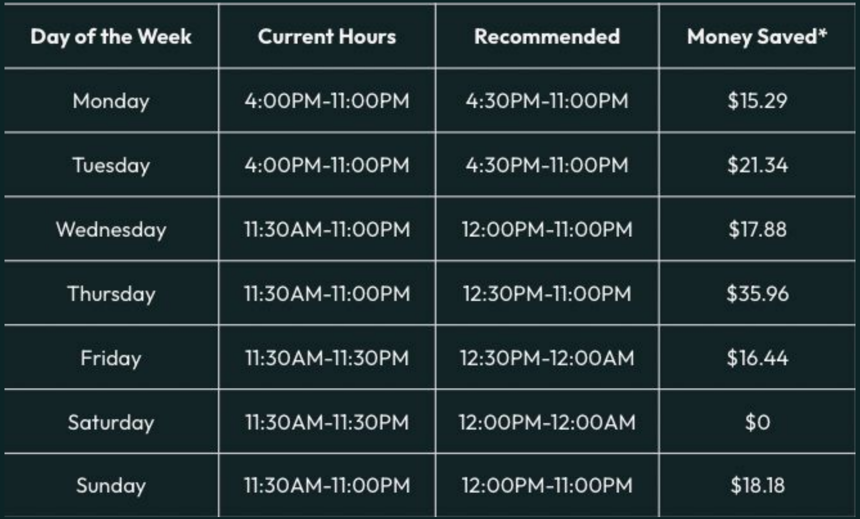
OPERATIONAL HOURS RECONSTRUCTION

ANALYSIS

- **Average First Purchase Time:**
  - Monday: 4:40PM
  - Tuesday: 4:26PM
  - Wednesday: 12:17PM
  - Thursday: 12:40PM
  - Friday: 12:27PM
  - Saturday: 12:08PM
  - Sunday: 12:15PM
- **Average Last Purchase Time:**
  - Monday: 10:39PM
  - Tuesday: 11:03PM
  - Wednesday: 10:59PM
  - Thursday: 11:12PM
  - Friday: 11:44PM
  - Saturday: 11:48PM
  - Sunday: 10:49PM

RECOMMENDATIONS

- **Reconstruct Kitchen Menu Hour:**



Day of the Week	Current Hours	Recommended	Money Saved*
Monday	4:00PM-11:00PM	4:30PM-11:00PM	$15.29
Tuesday	4:00PM-11:00PM	4:30PM-11:00PM	$21.34
Wednesday	11:30AM-11:00PM	12:00PM-11:00PM	$17.88
Thursday	11:30AM-11:00PM	12:30PM-11:00PM	$35.96
Friday	11:30AM-11:30PM	12:30PM-12:00AM	$16.44
Saturday	11:30AM-11:30PM	12:00PM-12:00AM	$0
Sunday	11:30AM-11:00PM	12:00PM-11:00PM	$18.18

*Based on the average pay for the difference in operational hours in the recommended section

- $125.1 saved per week
- $6,504.42 saved per year