Report ανάλυσης

Καθαρισμός και προετοιμασία δεδομένων

Τα δεδομένα των visitors, scheduling και receipts αφορούν τους μήνες Ιανουάριος-Νοέμβριος του 2024, ενώ τα δεδομένα για το NPS των διαφορετικών καταστημάτων περιλαμβάνουν και τον Δεκέμβρη του 2023. Αποφασίσαμε να αφαιρέσουμε τα δεδομένα από το 2023 ώστε να υπάρχει αντιστοιχία σε όλες τις ημερομηνίες από τα διαφορετικά δεδομένα.

Παρατηρήσαμε ότι υπάρχουν εγγραφές visitors και receipts εκτός του ωραρίου των καταστημάτων. Τις επιπλέον εγγραφές των visitors τις αποδώσαμε στο ότι το σύστημα καταγράφει τους υπαλλήλους που εισέρχονται και εξέρχονται στα καταστήματα κατά την έναρξη και λήξη της βάρδιάς τους. Τις επιπλέον εγγραφές receipts τις αποδώσαμε σε ηλεκτρονικές αγορές τις οποίες δηλώνουν οι πελάτες να παραλάβουν από το εκάστοτε κατάστημα. Για αυτόν τον λόγο όλες τις εγγραφές εκτός ωραρίου που αφορούν visitors και receipts τις αφαιρέσαμε.

Η αφαίρεση έγινε σύμφωνα με το ωράριο των διαφορετικών καταστημάτων (π.χ. το Mall τις καθημερινές λειτουργεί 10:00 - 20:00 ενώ τα υπόλοιπα καταστήματα 09:00 - 21:00)

```
# Add a column for the day of the week (0=Monday, 6=Sunday)
visitors_df['day_of_week'] = visitors_df['date'].dt.dayofweek

# Define conditions for removal
# For "mall" store
condition_mall_weekdays = (
    (visitors_df['store'] == 'mall') &
    (visitors_df['day_of_week'].between(0, 4)) &  # Monday to Friday
    ((visitors_df['hour'] < 10) | (visitors_df['hour'] > 20))
)

condition_mall_saturday = (
    (visitors_df['store'] == 'mall') &
    (visitors_df['day_of_week'] == 5) &  # Saturday
    ((visitors_df['hour'] < 11) | (visitors_df['hour'] > 19))
)

# For stores other than "mall"
condition_other_weekdays = (
    (visitors_df['store'] != 'mall') &
    (visitors_df['day_of_week'].between(0, 4)) &  # Monday to Friday
    ((visitors_df['hour'] < 9) | (visitors_df['hour'] > 20))
)

condition_other_saturday = (
    (visitors_df['store'] != 'mall') &
    (visitors_df['day_of_week'] == 5) &  # Saturday
    ((visitors_df['hour'] < 9) | (visitors_df['hour'] > 19))
)

# Combine all conditions
removal_condition = (
    condition_mall_weekdays |
    condition_mall_saturday |
    condition_other_weekdays |
    condition_other_saturday
)

# Remove rows that satisfy the condition
```

```
filtered_visitors_df = visitors_df[~removal_condition]

# Drop the day_of_week column (optional, if you don't want to keep it)
filtered_visitors_df = filtered_visitors_df.drop(columns=['day_of_week'])
# Add a column for the day of the week (0=Monday, 6=Sunday)
receipts_df['day_of_week'] = receipts_df['Date'].dt.dayofweek

# Define conditions for removal
# For "mall" store
condition_mall_weekdays = (
    (receipts_df['store'] == 'mall') &
    (receipts_df['day_of_week'].between(0, 4)) &  # Monday to Friday
    ((receipts_df['Hour'] < 10) | (receipts_df['Hour'] > 20))
)

condition_mall_saturday = (
    (receipts_df['store'] == 'mall') &
    (receipts_df['day_of_week'] == 5) &  # Saturday
    ((receipts_df['Hour'] < 11) | (receipts_df['Hour'] > 19))
)

# For stores other than "mall"
condition_other_weekdays = (
    (receipts_df['store'] != 'mall') &
    (receipts_df['day_of_week'].between(0, 4)) &  # Monday to Friday
    ((receipts_df['Hour'] < 9) | (receipts_df['Hour'] > 20))
)

condition_other_saturday = (
    (receipts_df['store'] != 'mall') &
    (receipts_df['day_of_week'] == 5) &  # Saturday
    ((receipts_df['Hour'] < 9) | (receipts_df['Hour'] > 19))
)

# Combine all conditions
removal_condition = (
    condition_mall_weekdays |
    condition_mall_saturday |
    condition_other_weekdays |
    condition_other_saturday
)

# Remove rows that satisfy the condition
filtered_receipts_df = receipts_df[~removal_condition]

# Drop the day_of_week column (optional, if you don't want to keep it)
filtered_receipts_df = filtered_receipts_df.drop(columns=['day_of_week'])
```

Ακόμη, προστέθηκαν επιπλέον στήλες που να υποδεικνύουν το αν η εγγραφή αναφέρεται σε weekday ή weekend, καθώς και αν η εγγραφή ανήκει σε κάποια τυχόν εκπτωτική περίοδο του 2024 ή όχι, και αν ναι σε ποια (Black Week, Winter offers, Summer offers, No Offer).

```
weekends = ['Saturday', 'Sunday']
receipts['Weekend'] = receipts['Day of the Week'].apply(lambda x: 'Weekend' if x in weekends else 'Weekday')
visitors['Weekend'] = visitors['Day of the Week'].apply(lambda x: 'Weekend' if x in weekends else 'Weekday')
```

Python

```python
def assign_offer(date):
    if pd.Timestamp("2024-01-15") <= date <= pd.Timestamp("2024-02-29"):
        return "Winter Offers"
    elif pd.Timestamp("2024-07-08") <= date <= pd.Timestamp("2024-08-31"):
        return "Summer Offers"
    elif pd.Timestamp("2024-11-25") <= date <= pd.Timestamp("2024-12-02"):
        return "Black Week"
    else:
        return "No Offer"

receipts['Offer'] = receipts['Date'].apply(assign_offer)
visitors['Offer'] = visitors['Date'].apply(assign_offer)
```

Python

Employee Workload

Η μετρική αυτή εκφράζει τους visitors που αντιστοιχούν (προς εξυπηρέτηση) σε έναν employee. Υπολογίζεται διαιρώντας τον αριθμό των visitors στο κατάστημα μία συγκεκριμένη ώρα με τον αριθμό των employees των οποίων η βάρδια καλύπτει αυτήν την ώρα.

```python
import numpy as np

# Ensure Start and End times are datetime objects
schedule_df['Start'] = pd.to_datetime(schedule_df['Start'],
format='%H:%M:%S').dt.time
schedule_df['End'] = pd.to_datetime(schedule_df['End'], format='%H:%M:%S').dt.time

# Add an Hour column to schedule_df for each hour an employee works
def expand_schedule(row):
   start_hour = row['Start'].hour
   end_hour = row['End'].hour
   hours = list(range(start_hour, end_hour))
   return [(row['Date'], hour, row['store'], row['Employee ID']) for hour in hours]

expanded_schedule = schedule_df.apply(expand_schedule, axis=1)
expanded_schedule = [item for sublist in expanded_schedule for item in sublist]
expanded_schedule_df = pd.DataFrame(expanded_schedule, columns=['Date', 'Hour',
'store', 'Employee ID'])

# Count the number of employees working at each store, date, and hour
employee_count = expanded_schedule_df.groupby(['Date', 'Hour',
'store']).size().reset_index(name='Employee Count')

# Merge filtered_visitors_df with employee_count
df_merged = pd.merge(filtered_visitors_df, employee_count, on=['Date', 'Hour',
'store'], how='left')

# Calculate visitors per employee
df_merged['Visitors per Employee'] = df_merged['visitors'] / df_merged['Employee
Count']

# Fill NaN values (if there are no employees working at a specific time, set
visitors per employee to NaN)
df_merged['Visitors per Employee'] = df_merged['Visitors per
Employee'].replace([np.inf, -np.inf], np.nan)

# Final dataframe
visitors_per_employee = df_merged[['Date', 'Hour', 'store', 'visitors', 'Employee
Count', 'Visitors per Employee']]

visitors_per_employee = visitors_per_employee.dropna()
```

Σχέση NPS - Employee Workload

Correlation analysis

Δεδομένου ότι η Public δίνει μεγάλη έμφαση στην ικανοποίηση των πελατών, σκεφτήκαμε να κάνουμε μια ανάλυση για να δούμε αν το NPS και οι employees συνδέονται μεταξύ τους. Αποφασίσαμε λοιπόν να κάνουμε ένα correlation analysis για κάθε κατάστημα ξεχωριστά. Στο **P+H Store**, που έχει και το χαμηλότερο NPS, η συσχέτιση ήταν έντονα αρνητική (-0.59681). Αυτό σημαίνει ότι όταν οι υπάλληλοι έχουν μεγάλο φόρτο εργασίας, το NPS πέφτει, πιθανώς επειδή οι πελάτες σε αυτά τα καταστήματα χρειάζονται περισσότερη βοήθεια και όταν το προσωπικό είναι πιεσμένο, η εξυπηρέτηση χειροτερεύει.

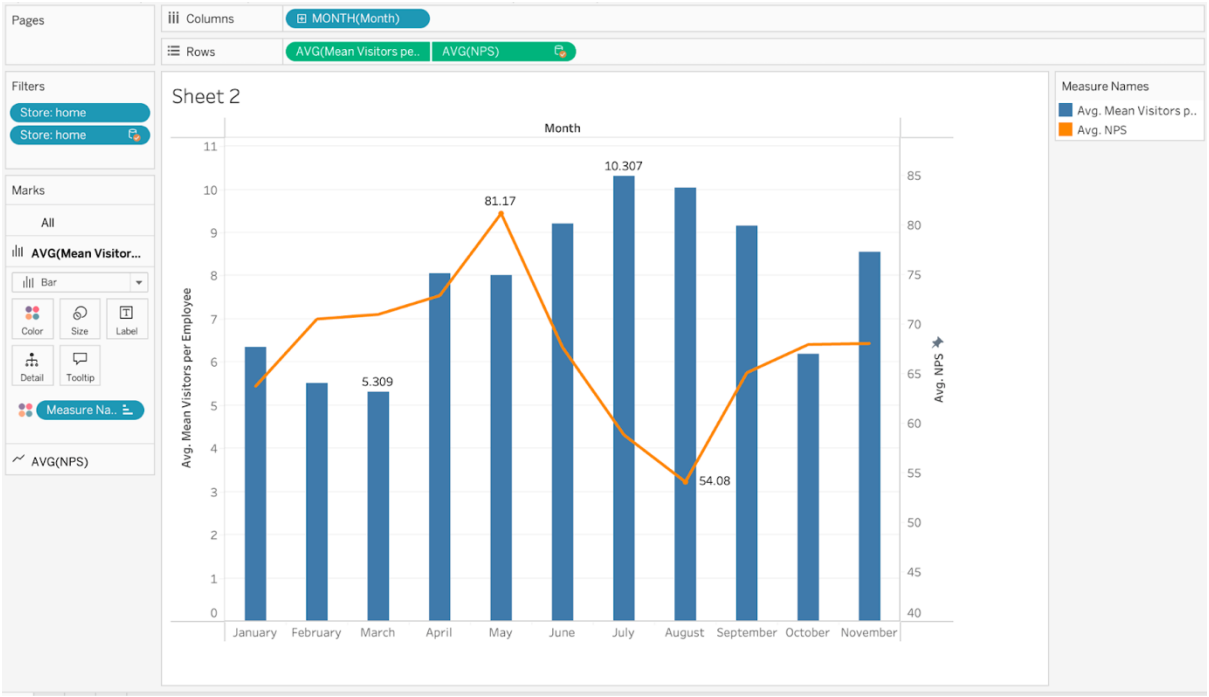
```python
store_types = final_data["store ID"].unique()
correlation_results = {}

for store in store_types:
    subset = final_data[final_data["store ID"] == store]
    corr_nps_workload = subset["NPS"].corr(subset["Mean Visitors per Employee"])
    correlation_results[store] = corr_nps_workload
    print(f"Correlation between NPS and Employee Workload for {store}: {corr_nps_workload:.5f}")
```

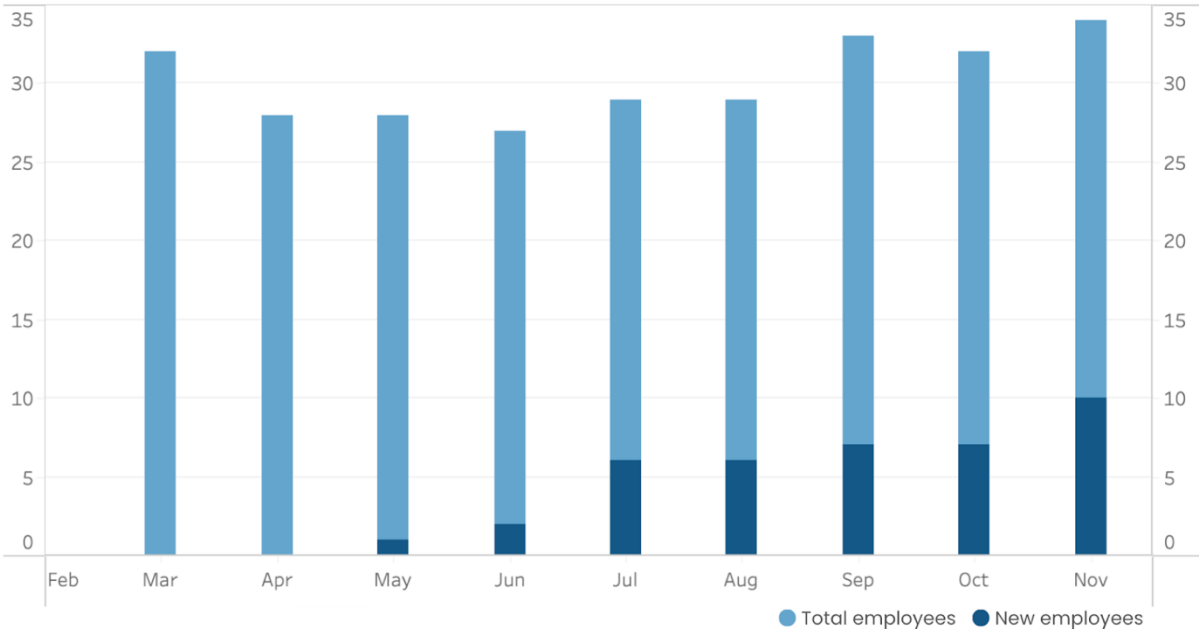
Python

Public + Home

Αποφασίσαμε να διερευνήσουμε σε μεγαλύτερο βάθος τη σχέση NPS - Employee Workload στο συγκεκριμένο κατάστημα.
Στο παρακάτω διάγραμμα οι μπλε ράβδοι δείχνουν τα επίπεδα της μετρικής Employee workload ενώ η πορτοκαλί γραμμή την πορεία του NPS μέσα στον χρόνο μελέτης ανά τους μήνες.
Οι δύο αυτές μετρικές έχουν αρνητική σχέση, επομένως όποτε το Workload είναι αυξημένο, βλέπουμε το NPS μειωμένο.
Παρατηρούμε μία μεγάλη πτώση τους θερινούς μήνες, η οποία σημειώνει το ελάχιστο NPS καταγεγραμμένο σε όλα τα καταστήματα. Η πτώση αυτή αποδίδεται στο αυξημένο Workload στους ίδιους μήνες.



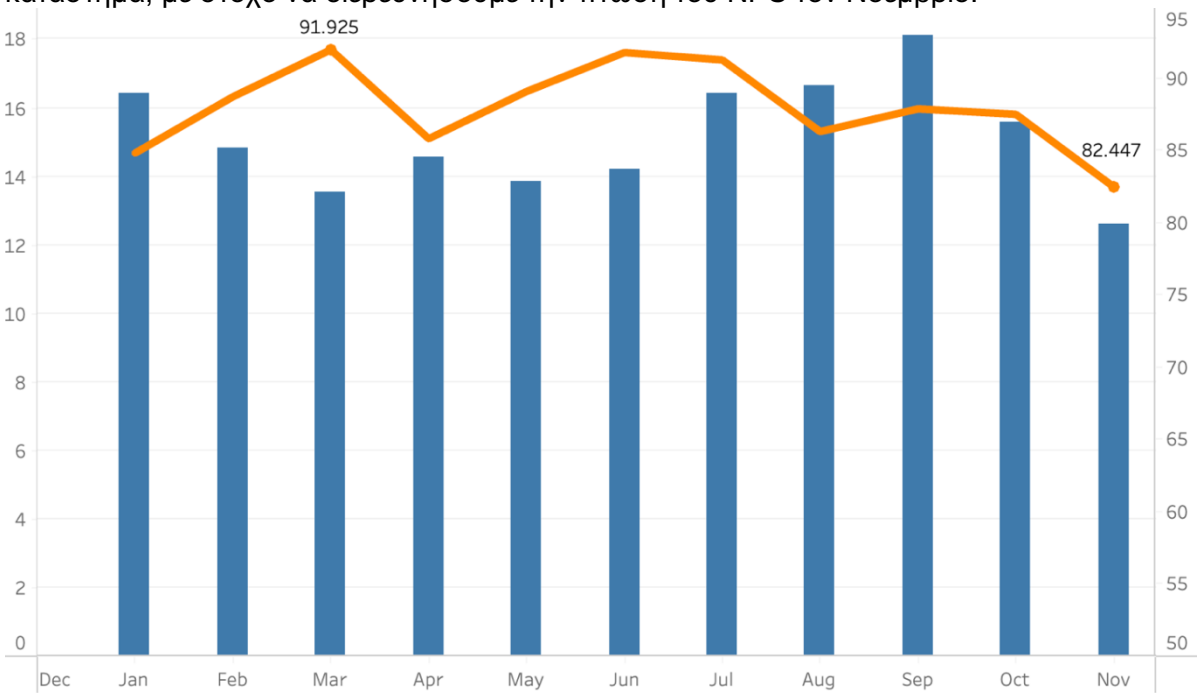
Παράλληλα, τους ίδιους μήνες παρατηρήσαμε ότι στο συγκεκριμένο κατάστημα εργάζονται αρκετοί καινούριοι υπάλληλοι, όπως φαίνεται και στο παρακάτω διάγραμμα. Το σκούρο μπλε κομμάτι των ράβδων είναι οι υπάλληλοι που εργάζονται στο κατάστημα λιγότερο από δύο μήνες και το γαλάζιο οι υπόλοιποι. Η επιλογή των δύο μηνών έγινε καθώς σε πηγές όπως η ιστοσελίδα RetailWire αναφέρεται πως ένας νεοπροσληφθέντας υπάλληλος καταστήματος λιανικής χρειάζεται κατά μέσο όρο 2 με 3 μήνες εργασίας για να ξεκινήσει να αποδίδει σε ικανοποιητικό βαθμό.



Συνεπώς, τα δεδομένα δείχνουν ότι η πτώση του NPS από τον Ιούνιο μέχρι και τον Αύγουστο οφείλεται στο γεγονός ότι το κατάστημα δεν έχει αρκετό προσωπικό σε σχέση με τον αριθμό των επισκεπτών, ενώ παράλληλα σημαντικός αριθμός των εργαζομένων είναι καινούριοι και επακολούθως δεν διαθέτουν επαρκή εμπειρία να εξυπηρετήσουν αποδοτικά τους πελάτες.

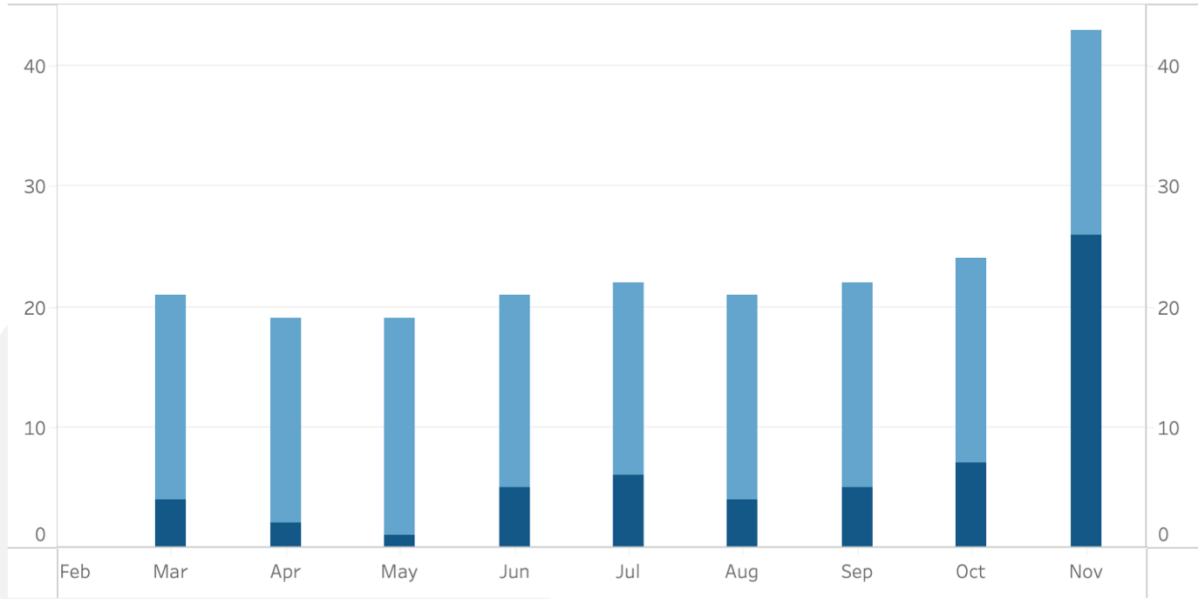
Μικρό επαρχιακό κατάστημα

Αποφασίσαμε να μελετήσουμε τη σχέση NPS - Employee Workload και στο μικρό επαρχιακό κατάστημα, με στόχο να διερευνήσουμε την πτώση του NPS τον Νοέμβριο.



Παρατηρούμε ότι τον Νοέμβριο το Workload έχει τη μικρότερη τιμή σε όλον τον χρόνο, που είναι αντιφατικό.

Ωστόσο, τον συγκεκριμένο μήνα είναι η εκπτωτική περίοδος Black Friday - Black Week. Το μέσο Employee Workload κατά τη διάρκεια του Black Week φτάνει τους 16 visitors, τιμή πολύ υψηλή για το συγκεκριμένο κατάστημα, ενώ τον υπόλοιπο μήνα τους 10 visitors.



Παράλληλα, πάνω από τους μισούς υπαλλήλους που εργάζονται τον μήνα αυτόν είναι καινούριοι (26 στους 43).

Επομένως, η πτώση του NPS είναι πιθανόν να οφείλεται κυρίως στον μεγάλο αριθμό νέων υπαλλήλων που δεν έχουν επαρκή εμπειρία στην εργασία ώστε να εξυπηρετήσουν τους πελάτες σε ικανοποιητικό βαθμό.

Για τον υπολογισμό των καινούριων εργαζομένων μετρήθηκαν για τον κάθε μήνα τα employee IDs που εμφανίζονται για πρώτη φορά τον ίδιο ή τον προηγούμενο μήνα και έπειτα αφαιρέθηκαν οι 2 πρώτοι μήνες από τα αποτελέσματα:

```
# Step 1: Ensure correct data types
schedule_df['Date'] = pd.to_datetime(schedule_df['Date'])

# Step 2: Total Employees per Store per Month
total_employees = schedule_df.groupby(['store', 'Month'])['Employee
ID'].nunique().reset_index()
total_employees.rename(columns={'Employee ID': 'Sum employees'}, inplace=True)

# Step 3: Identify New Employees
# Sort the data for consistent processing
schedule_df = schedule_df.sort_values(by=['Employee ID', 'store',
'Month']).reset_index(drop=True)

# Create a helper dataframe to track employee activity
employee_activity = schedule_df.groupby(['Employee ID',
'store'])['Month'].unique().reset_index()

# Function to find new employees
def identify_new_employees(activity_df):
   new_employees_data = []

   for _, row in activity_df.iterrows():
       emp_id = row['Employee ID']
       store = row['store']
       months = sorted(row['Month'])  # Sort months for sequential comparison

       for idx, current_month in enumerate(months):
           # For the first two months, the employee is considered new
           if idx < 2:
               is_new = True
           else:
               # Check if the employee worked in the previous two months
               prev_month_1 = current_month - 1
               prev_month_2 = current_month - 2
               is_new = not (prev_month_1 in months and prev_month_2 in months)

           new_employees_data.append({
               'store': store,
               'Month': current_month,
               'Employee ID': emp_id,
               'is_new': is_new
           })

   return pd.DataFrame(new_employees_data)

# Apply the function
new_employees_df = identify_new_employees(employee_activity)

# Filter only new employees and count them
new_employees_count =
new_employees_df[new_employees_df['is_new']].groupby(['store',
'Month']).size().reset_index(name='New employees')

# Step 4: Merge the total and new employees data
new_emp = pd.merge(total_employees, new_employees_count, on=['store', 'Month'],
how='left')
```

```
# Fill missing values for new employees
new_emp['New employees'] = new_emp['New employees'].fillna(0).astype(int)

# Remove months 1 and 2
filtered_new_emp = new_emp[~new_emp['Month'].isin([1, 2])].reset_index(drop=True)
```

Διακράτηση υπαλλήλων

Για την εύρεση του αριθμού των υπαλλήλων που έφυγαν από την εταιρία ανά μήνα, χρησιμοποιήσαμε τα employee IDs που σταματούν να εμφανίζονται τον συγκεκριμένο μήνα, ενώ ήταν παρόντα τον προηγούμενο.

```
# Ensure 'Date' column is in datetime format
schedule_df['Date'] = pd.to_datetime(schedule_df['Date'])

# Group by 'Month' and collect unique employee IDs
monthly_employees = schedule_df.groupby('Month')['Employee ID'].unique()

# Initialize dictionaries to store results
disappearing_employees_per_month = {}

previous_month_employees = set()

# Iterate through each month (from 1 to 12)
for month in range(1, 13):
    employees = set(monthly_employees.get(month, []))  # Get employees or empty set

    # Disappearing employees: those in the previous month but not in the current month
    disappearing_employees = previous_month_employees - employees
    disappearing_employees_per_month[month] = len(disappearing_employees)

    # Update previous month employees
    previous_month_employees = employees

# Create the final DataFrame
summary_df = pd.DataFrame({
    'Month': new_employees_per_month.keys(),
    'Disappearing Employees': disappearing_employees_per_month.values()
})
summary_df = summary_df[~summary_df['Month'].isin([1, 12])]
```

Δείκτης Μετατροπής Επισκέψεων σε Πωλήσεις (Conversion Rate)

Ως conversion rate ορίστηκε το πηλίκο των αποδείξεων (receipts) προς τις επισκέψεις (visits) για μία συγκεκριμένη χρονική περίοδο. Χρησιμοποιώντας κατάλληλα το PowerBI, μπορέσαμε και υπολογίσαμε το δείκτη με βάση διάφορες χρονικές διάρκειες:

- Ανά τρίμηνο (quarterly)
- Ανά μήνα (monthly)
- Ανά ημέρα (daily)

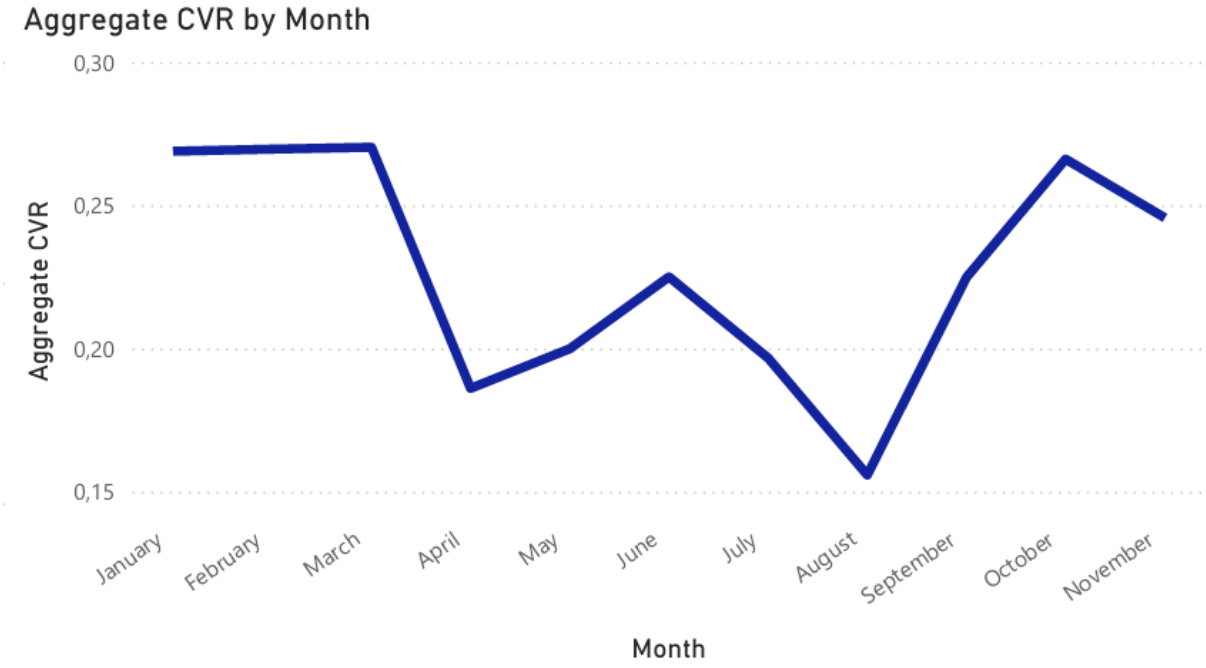
Δεδομένου ότι η ανάλυση ανά μήνα μας δίνει μία καλύτερη συνολική εικόνα (βέλτιστη ισορροπία μεταξύ συχνότητας μετρήσεων) για την εξελικτική πορεία του δείκτη Conversion Rate - ή CVR για συντομία - παρατίθενται τα διαγράμματα ανά μήνα και ανά κατάστημα σχετικά με το CVR:

Συνολική Εικόνα 4 καταστημάτων:

1. Πιο παραγωγικοί μήνες είναι οι: Μάρτιος και Οκτώβριος με CVR κοντά στο 36%. Όπως προκύπτει, οι μήνες χαμηλής επισκεψιμότητας δείχνουν να έχουν αυξημένο ποσοστό μετατροπής, υποδεικνύοντας καλύτερη εξυπηρέτηση των πελατών.
2. Σε πλήρη αντίθεση, οι μήνες υψηλής επισκεψιμότητας Απρίλιος, Μάιος και Ιούνιος (2ο τρίμηνο) έχουν τον χαμηλότερο παραγωγικό δείκτη, με μέσο CVR 29%.

Public & Home:

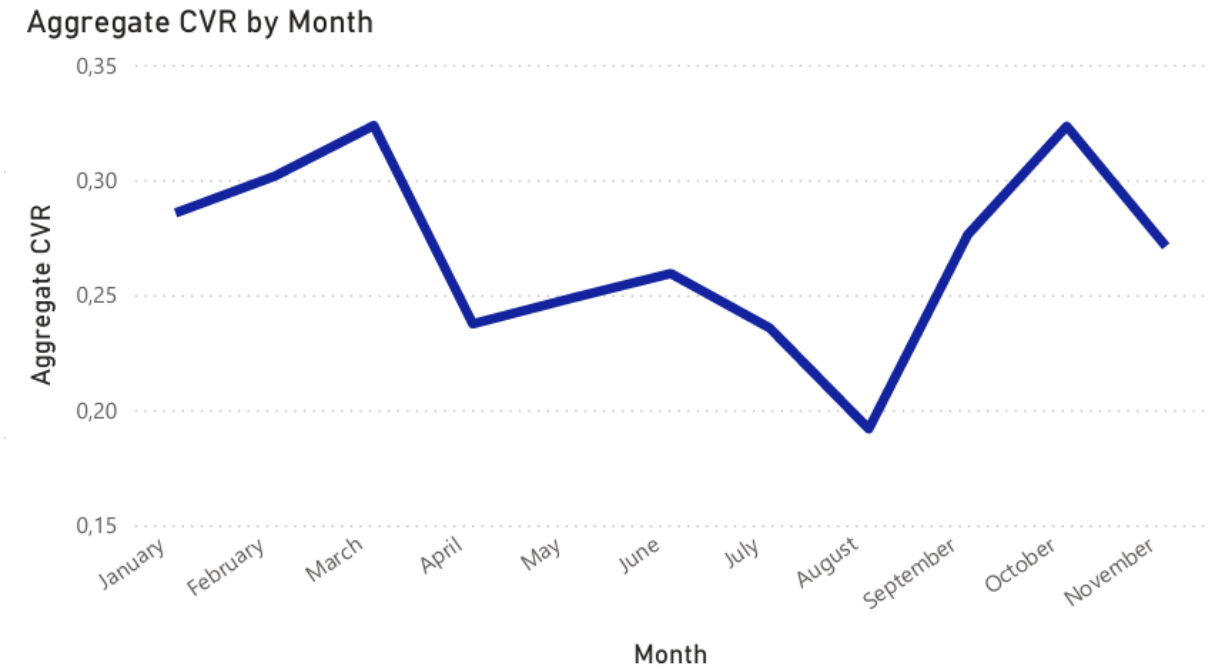
Ενδιαφέρουσες παρατηρήσεις επί του διαγράμματος είναι ότι:



- Οι πιο παραγωγικοί μήνες είναι ο Μάρτιος και ο Οκτώβριος με CVR 27%.
- Η λιγότερο παραγωγική περίοδος είναι το τρίμηνο Ιούλιος-Αύγουστος-Σεπτέμβριος με μέσο CVR μόλις 19%.

Διαπιστώνεται ότι η επίδοση του καταστήματος P&H είναι αρκετά κάτω από το Μ.Ο. των καταστημάτων, σημαίνοντας την αδυναμία του να κεφαλαιοποιήσει αποτελεσματικά τις ευκαιρίες που του δίνονται μέσω των επισκέψεων των καταναλωτών.

Μικρό κατάστημα Αθήνας:



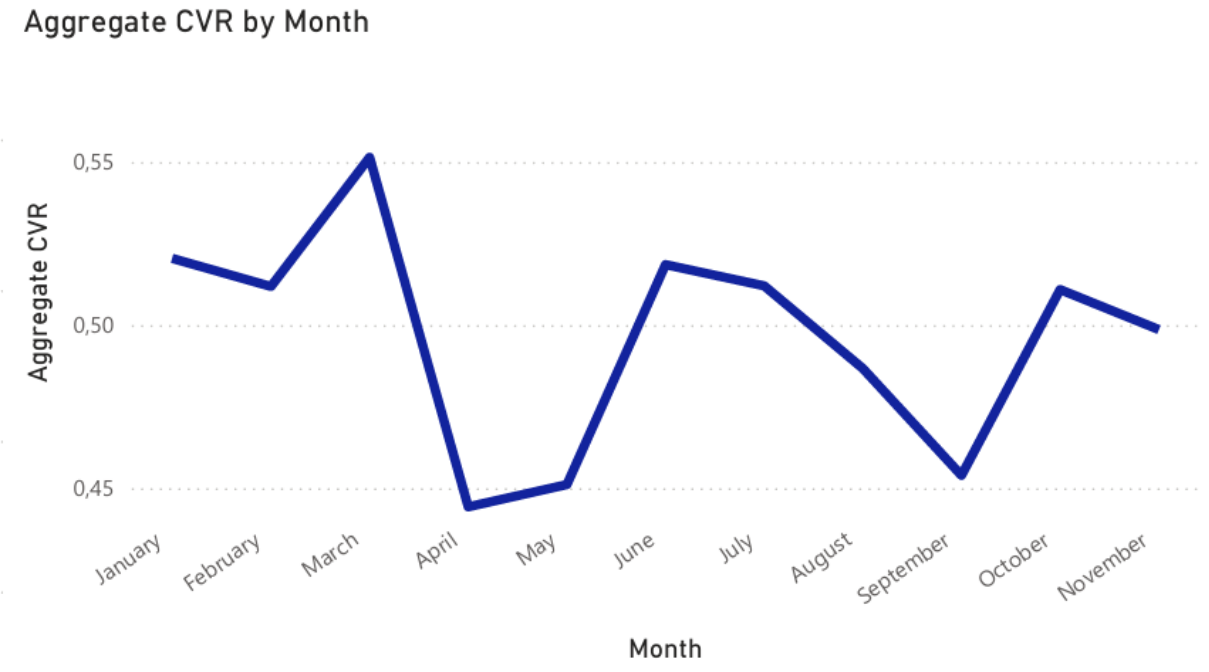
Ενδιαφέρουσες παρατηρήσεις:

- Οι πιο παραγωγικοί μήνες είναι ο Μάρτιος και ο Οκτώβριος με 32% CVR.
- Η λιγότερο παραγωγική περίοδος είναι το τρίμηνο Ιούλιος-Αύγουστος-Σεπτέμβριος με μέσο CVR μόλις 23%.

Διαφαίνονται καλύτερες επιδόσεις από αυτές του καταστήματος P&H, αλλά η αποτελεσματικότητα των πωλήσεων εξακολουθεί να βρίσκεται πολύ κάτω από τον Μ.Ο.

Κατάστημα στο Mall:

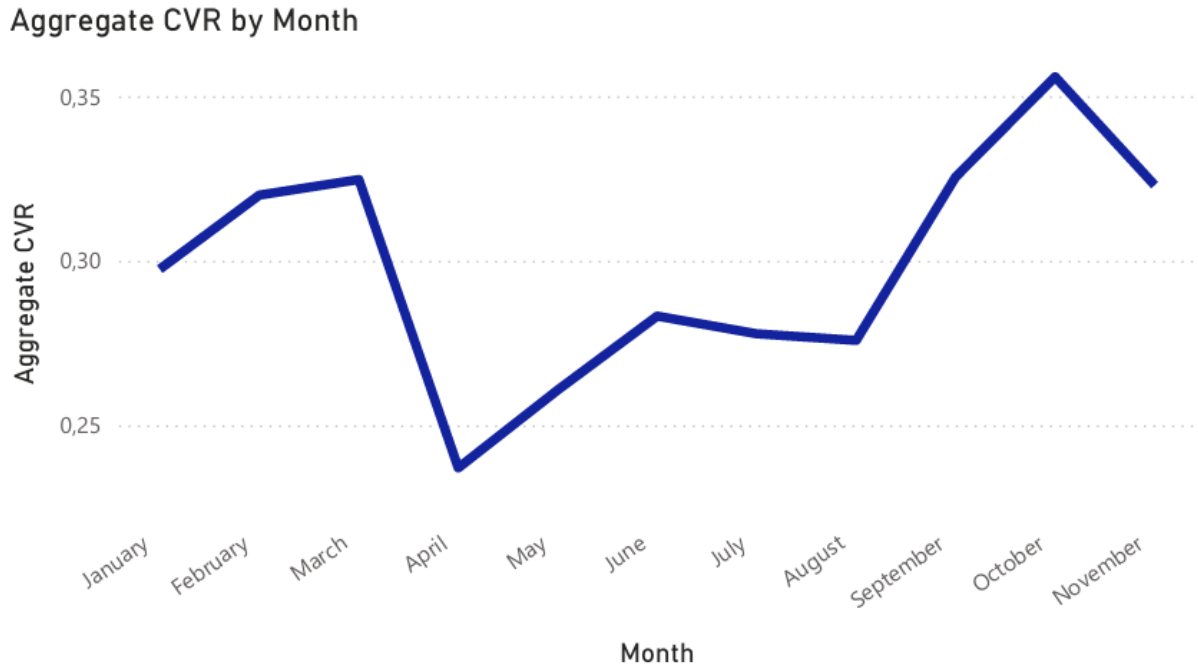
Κύρια σημεία ενδιαφέροντος:



- Ο Μάρτιος είναι πάλι ο πιο παραγωγικός μήνας με δείκτη μετατροπής 55%.
- Η λιγότερο παραγωγική περίοδος είναι το τρίμηνο Απρίλιος-Μάιος-Ιούνιος με μέσο CVR 47%.
- Ένα ακόμα insight που προέκυψε μέσω του PowerBI: Η Κυριακή είναι η πιο παραγωγική ημέρα της εβδομάδας, με 60% δείκτη μετατροπής!

Το συγκεκριμένο κατάστημα φαίνεται να έχει με διαφορά την καλύτερη ικανότητα να μετατρέπει τις επισκέψεις του σε πωλήσεις, όπου ακόμα και οι “κακές” περίοδοι του είναι πολύ πάνω από τον προσδοκώμενο Μέσο Όρο.
Οι ανοιχτές Κυριακές φαίνεται να προωθούν το κατάστημα έντονα προς αυτήν την κατεύθυνση, αφού αποτελούν πόλο έλξης για τους καταναλωτές οι οποίοι βρίσκουν την ευκαιρία να πραγματοποιήσουν αγορές τις ημέρες ξεκούρασης.

Επαρχιακό Κατάστημα:



- Καλύτερος μήνας εμφανίζεται ο Οκτώβριος, με CVR περί του 36%.
- Το δεύτερο τετράμηνο (Απρίλιος-Μάιος-Ιούνιος) έχει τη χαμηλότερη παραγωγικότητα με μέση τιμή 26%.

Γενικά επιδόσεις που κυμαίνονται κοντά στον Συνολικό Μέσο Όρο. Το κατάστημα δείχνει να έχει καλή απόδοση, κάτι που οφείλει να διατηρηθεί σε τουλάχιστον το ίδιο επίπεδο.

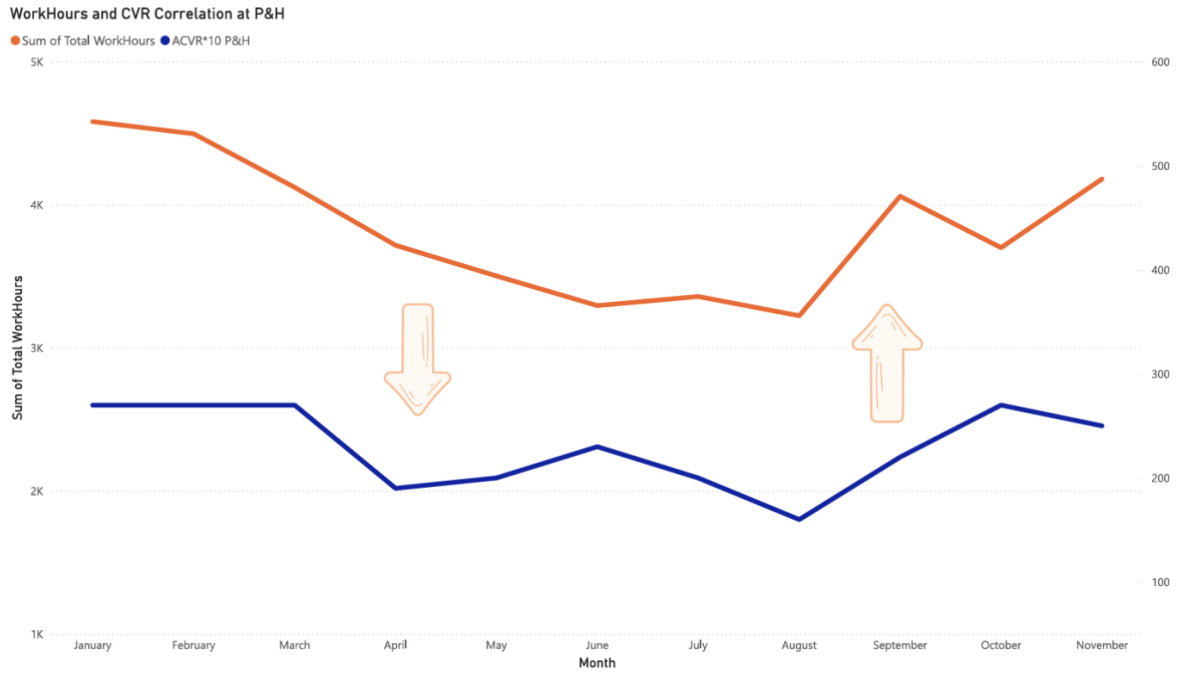
Συσχέτιση Conversion Rate με Εργατοώρες καταστημάτων

Πραγματοποιήθηκε μία συγκριτική ανάλυση μεταξύ του δείκτη μετατροπής και των συνολικών εργατοωρών που αφιερώθηκαν σε κάθε κατάστημα, για κάθε μήνα του 2024 (με εξαίρεση το Δεκέμβριο όπου δεν υπήρχαν διαθέσιμα δεδομένα). Οι εργατοώρες εξήχθησαν

από τα αρχικά δεδομένα υπολογίζοντας τη διαφορά μεταξύ της λήξης και της έναρξης της βάρδιας για κάθε εγγραφή.

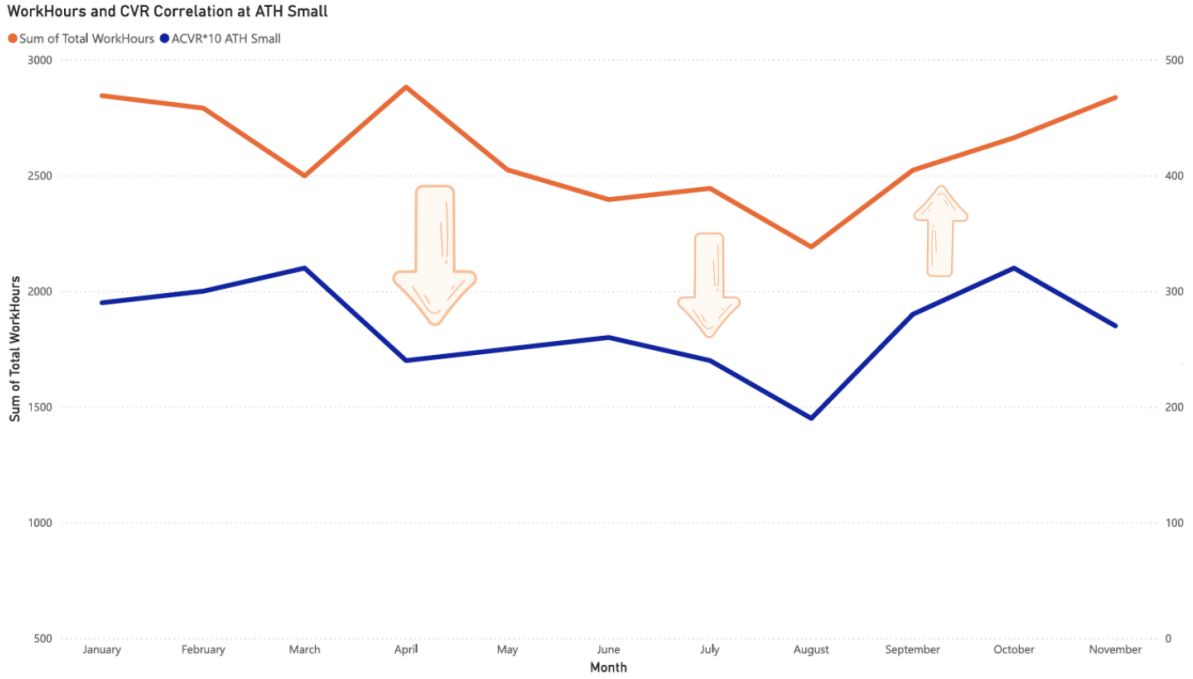
Η γενική εικόνα έδειξε ότι το σύνολο των εργατοωρών **έχει επίδραση** στην αποτελεσματικότητα των πωλήσεων. Σε άλλα καταστήματα αυτό φάνηκε ξεκάθαρα, ενώ σε άλλα η επίδραση ήταν πιο έμμεση.

Public & Home:



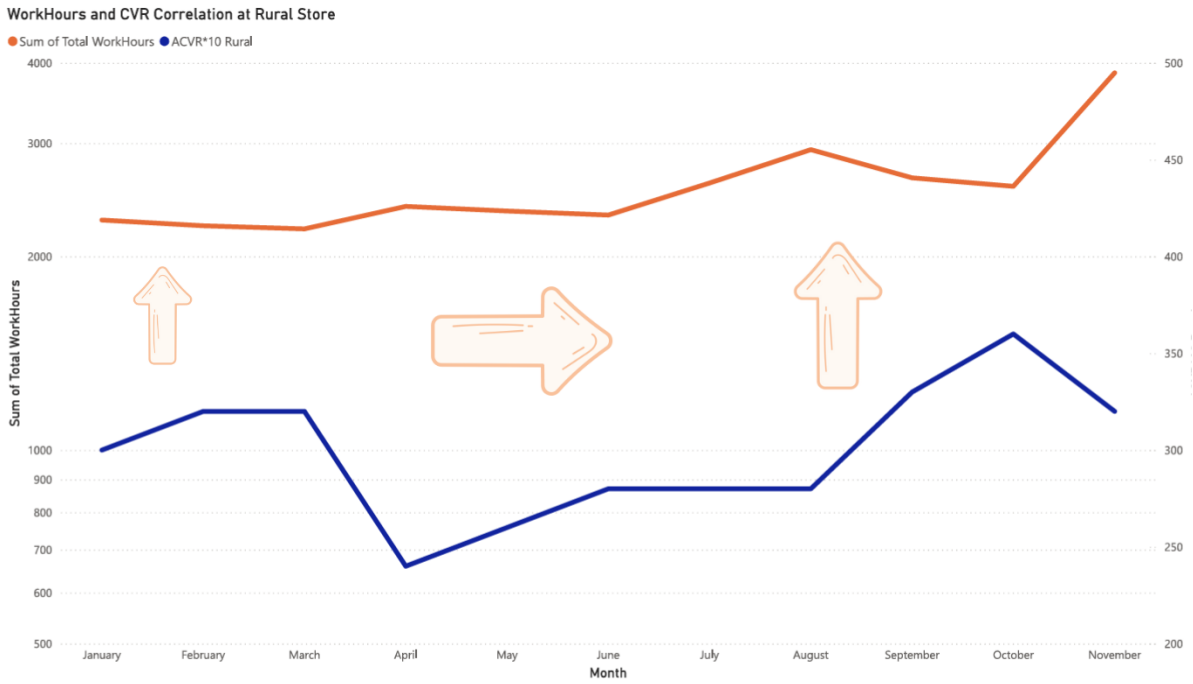
Με πορτοκαλί απεικονίζεται η εξελικτική γραμμή των συνολικών εργατοωρών ανά μήνα, ενώ με σκούρο μπλε είναι η γραμμή του Conversion Rate. Παρατηρούμε ότι όσο φθίνει ο αριθμός των εργατοωρών, τόσο μειώνεται και το CVR. Αντίστοιχα, όταν οι εργατοώρες αρχίζουν να αυξάνουν μετά τον Αύγουστο, το CVR αποκτά και αυτό ανοδική πορεία.

Μικρό κατάστημα στην Αθήνα:

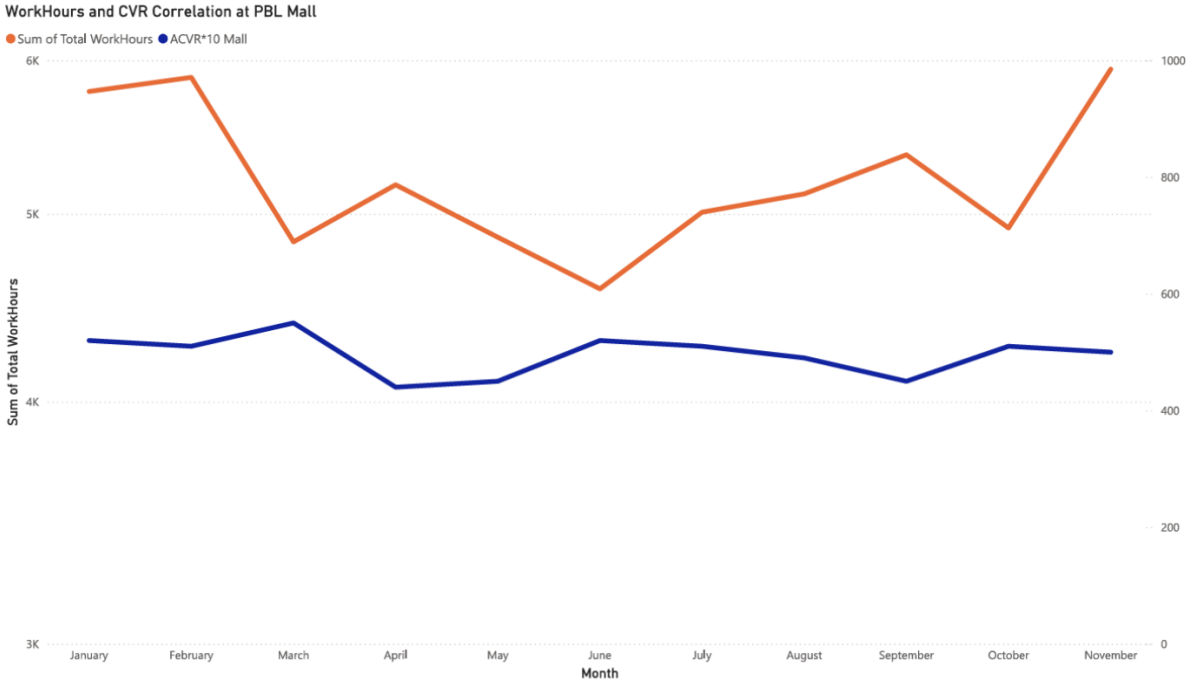


Παρόμοιες παρατηρήσεις έχουμε και για το μικρό κατάστημα στην Αθήνα.

Επαρχιακό κατάστημα:



Στο επαρχιακό κατάστημα η θετική συσχέτιση φαίνεται να διαρκεί έως τον Οκτώβριο. Τον Νοέμβριο παρότι οι εργατοώρες που αφιερώνονται στο κατάστημα ανεβαίνουν έντονα, ο δείκτης μετατροπής μειώνεται εμφανώς. Αυτό θα μπορούσε να είναι επίδραση του πολυάριθμου νέου προσωπικού, το οποίο δεν έχει την απαιτούμενη εμπειρία να εξυπηρετήσει ικανοποιητικά τους επισκέπτες.



Κατάστημα στο Mall:

Τέλος, το κατάστημα στο Mall φαίνεται να αποτελεί τη μοναδική εξαίρεση, καθώς δεν υπάρχει άμεση εμφανής συσχέτιση των εργατοωρών με τον δείκτη μετατροπής. Η βασικότερη αιτία για αυτήν την “σύγχυση” είναι η εξαιρετικά ευρεία διασπορά των εργατοωρών από μήνα σε μήνα. Σαφώς αυτή η υψηλή μεταβλητότητα των βαρδιών δεν μας βοηθά να εξάγουμε ένα ασφαλές συμπέρασμα. Η εικόνα θα ήταν πολύ διαφορετική εάν το κατάστημα είχε πιο σταθερό και πιο ισορροπημένο εργασιακό πρόγραμμα.

Συστάσεις προς τα καταστήματα Public

Συνοψίζοντας τα insights που διαπιστώθηκαν μέσω της ανάλυσης μας, συστήνουμε στην Εταιρεία Public να λάβει σοβαρά υπ’ όψιν της τις παρακάτω προτάσεις:

1. Βελτίωση της διακράτησης των υπαλλήλων μέσω προσφοράς ανταγωνιστικών μισθών και προνομίων, με στόχο την διατήρηση του προσωπικού το οποίο -με υψηλό κόστος- εκπαιδεύεται για τις θέσεις εργασίας.
2. Εξισορρόπηση του εργασιακού φόρτου στο κατάστημα Public+Home, μέσω της πρόσληψης νέων υπαλλήλων.
3. Βελτίωση των χειμερινών εκπτώσεων. Είναι ένα μεγάλο “παράθυρο” πωλήσεων, το οποίο η εταιρεία υποαξιοποιεί, με κόστος στα κέρδη της.
4. Σταθεροποίηση των εργατοωρών (ειδικά στο κατάστημα Mall). Η εξισορροπημένη κατανομή των ωρών εργασίας αυξάνει την αποδοτικότητα διασφαλίζοντας επαρκή στελέχωση στις ώρες αιχμής για βελτιωμένη εξυπηρέτηση πελατών.