Kalpesh_Patil_Tasks

Objectives

- The primary objective of this project is to analyze customer renewal patterns and performance metrics for The AA Company, with a focus on improving save rates, understanding discount utilization, and identifying key factors influencing customer retention.
- This involves calculating and comparing save rates and discount rates for specific months, evaluating team and agent performance, analyzing the impact of tenure mix on renewals, and identifying month-to-month changes in customer behavior.
- The insights derived from this analysis aim to help the company optimize its strategies for retaining customers, enhancing team performance, and aligning discounts and pricing with customer preferences, ultimately driving higher renewal rates and long-term loyalty.

Importing necessary libraries

In [1]:
```python
import numpy as np
import pandas as pd
from warnings import filterwarnings
filterwarnings('ignore')

import matplotlib.pyplot as plt
import seaborn as sns
```

Extracting Data-Frame from the Excel file sheets

In [2]:
```python
file = 'TestData for Analyst Interview.xlsx'

data_sheet = pd.read_excel(file, sheet_name=None)

print("Sheet names:", data_sheet.keys())
```

```
Sheet names: dict_keys(['tasks', 'key', 'Dummy Data'])
```

In [3]:
```
data = data_sheet['Dummy Data']
data.head()
```

Out[3]:

	TRANSACTION_DATE	Transaction Month	TENURE	REPEAT_CALLER	OPERATOR_ID	PAYMENT_
0	2019-06-01 09:52:35	2019-06-01	1	N	Agent159	
1	2019-06-01 09:52:39	2019-06-01	5	Y	Agent39	
2	2019-06-01 09:53:05	2019-06-01	5	Y	Agent220	
3	2019-06-01 09:53:06	2019-06-01	1	N	Agent191	
4	2019-06-01 09:53:13	2019-06-01	5	Y	Agent190	

In [4]:
```
data.shape
```

Out[4]:
```
(198989, 18)
```

- There are 1,98,989 rows and 18 columns in this dataset.

Main data

In [5]:
```
data.head()
```

Out[5]:

	TRANSACTION_DATE	Transaction Month	TENURE	REPEAT_CALLER	OPERATOR_ID	PAYMENT_
0	2019-06-01 09:52:35	2019-06-01	1	N	Agent159	
1	2019-06-01 09:52:39	2019-06-01	5	Y	Agent39	
2	2019-06-01 09:53:05	2019-06-01	5	Y	Agent220	
3	2019-06-01 09:53:06	2019-06-01	1	N	Agent191	
4	2019-06-01 09:53:13	2019-06-01	5	Y	Agent190	

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 198989 entries, 0 to 198988
Data columns (total 18 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   TRANSACTION_DATE    198989 non-null  datetime64[ns]
 1   Transaction Month   198989 non-null  datetime64[ns]
 2   TENURE              198989 non-null  int64
 3   REPEAT_CALLER       198989 non-null  object
 4   OPERATOR_ID         198989 non-null  object
 5   PAYMENT_TYPE        198989 non-null  object
 6   TYPE                198989 non-null  object
 7   SAVE_TYPE           198989 non-null  object
 8   MEMBERSHIP_TYPE     198989 non-null  object
 9   PRODUCT_TYPE        198989 non-null  object
 10  SITE_NAME           198989 non-null  object
 11  TEAM_NAME           198989 non-null  object
 12  SAM_RENEWAL_OFFER   49387 non-null   object
 13  INVITED_PREMIUM     147929 non-null  float64
 14  TOTAL_DISCOUNT      147929 non-null  float64
 15  RENEWAL_PREMIUM     198989 non-null  float64
 16  Renewed?            198989 non-null  int64
 17  Discount Given      147905 non-null  float64
dtypes: datetime64[ns](2), float64(4), int64(2), object(10)
memory usage: 27.3+ MB
```

Checking Missing Values:

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
TRANSACTION_DATE          0
Transaction Month         0
TENURE                    0
REPEAT_CALLER             0
OPERATOR_ID               0
PAYMENT_TYPE              0
TYPE                      0
SAVE_TYPE                 0
MEMBERSHIP_TYPE           0
PRODUCT_TYPE              0
SITE_NAME                 0
TEAM_NAME                 0
SAM_RENEWAL_OFFER    149602
INVITED_PREMIUM       51060
TOTAL_DISCOUNT        51060
RENEWAL_PREMIUM           0
Renewed?                  0
Discount Given        51084
dtype: int64
```

Here we do not need to explicitly handle null values separately for this task because:

- The condition RENEWAL_PREMIUM > 0 automatically excludes rows where RENEWAL_PREMIUM is null.

- The volume of transactions (denominator) includes only valid rows that have data for the month and can potentially have a RENEWAL_PREMIUM.
- Null values in RENEWAL_PREMIUM are not considered in the save rate calculation because they do not represent valid transactions with renewal data. Similarly, for the volume of transactions, only rows with valid transaction details and month information

Task 1

What were the save rates and discount rates for June and July

In [8]:
```
data.columns
```

Out[8]:
```
Index(['TRANSACTION_DATE', 'Transaction Month', 'TENURE', 'REPEAT_CALLER',
       'OPERATOR_ID', 'PAYMENT_TYPE', 'TYPE', 'SAVE_TYPE', 'MEMBERSHIP_TYP
E',
       'PRODUCT_TYPE', 'SITE_NAME', 'TEAM_NAME', 'SAM_RENEWAL_OFFER',
       'INVITED_PREMIUM', 'TOTAL_DISCOUNT', 'RENEWAL_PREMIUM', 'Renewed?',
       'Discount Given'],
      dtype='object')
```

In [9]:
```
save_rate_full_data = data[data['RENEWAL_PREMIUM']>0]['Transaction Month'].
save_rate_full_data * 100
```

Out[9]:
```
74.30410726221048
```

- The goal of Task 1 was to calculate the save rate, which measures the percentage of transactions where customers renewed their services. Specifically, I calculated the save rate for:
- The entire dataset.
- The months of June and July, separately.
- The save rate is calculated as: (Number of Transactions Where (RENEWAL_PREMIUM > 0) / Total Volume of Transactions) * 100
- Here:

1. RENEWAL_PREMIUM > 0 indicates a successful renewal.
2. The volume of transactions is the total number of rows for the dataset or a specific month, excluding null values in RENEWAL_PREMIUM.

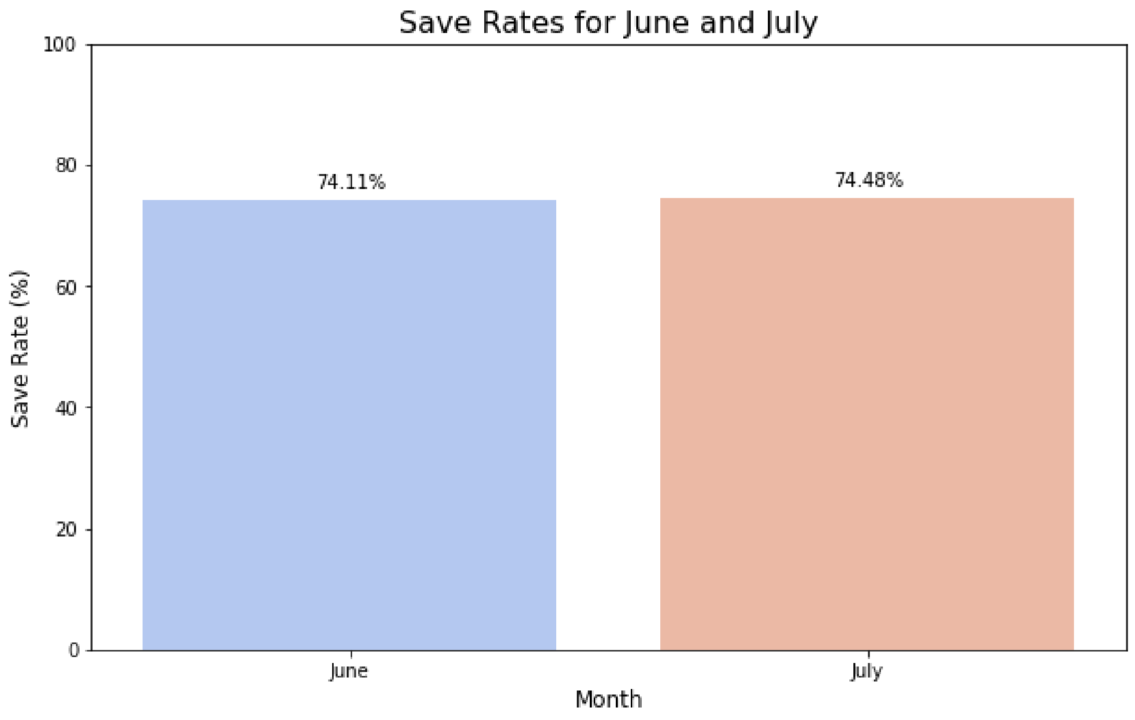
- The overall save rate of 74.30%.

In [10]:
```
temp_data = data

temp_data['Transaction Month'] = temp_data['TRANSACTION_DATE'].dt.month_nam

june_data = temp_data[temp_data['Transaction Month'] == 'June']
july_data = temp_data[temp_data['Transaction Month'] == 'July']

june_save_rate = june_data[june_data['RENEWAL_PREMIUM'] > 0].shape[0] / jun
july_save_rate = july_data[july_data['RENEWAL_PREMIUM'] > 0].shape[0] / jul


print(f"Save Rate for June: {june_save_rate:.2%}")
print(f"Save Rate for July: {july_save_rate:.2%}")
```

```
Save Rate for June: 74.11%
Save Rate for July: 74.48%
```

- The save rates were 74.11% for June and 74.48% for July.
- About three-quarters of all transactions resulted in renewals across the dataset.
- The save rate was slightly higher in July compared to June, which might indicate improved renewal performance during that month.

In [11]:
```
# Visualization of Save Rates
save_rates = pd.DataFrame({
    'Month': ['June', 'July'],
    'Save Rate (%)': [june_save_rate * 100, july_save_rate * 100]
})

plt.figure(figsize=(10, 6))
sns.barplot(x='Month', y='Save Rate (%)', data=save_rates, palette='coolwar
plt.title('Save Rates for June and July', fontsize=16)
plt.ylabel('Save Rate (%)', fontsize=12)
plt.xlabel('Month', fontsize=12)
plt.ylim(0, 100)
for index, value in enumerate(save_rates['Save Rate (%)']):
    plt.text(index, value + 2, f'{value:.2f}%', ha='center', fontsize=10)
plt.show()
```



In [12]:
```
# june_discount_rate
june_total_discount = june_data['TOTAL_DISCOUNT'].sum()
june_invited_premium = june_data['INVITED_PREMIUM'].sum()
june_discount_rate = (june_total_discount / june_invited_premium )

# Calculate Discount Rate for July
july_total_discount = july_data['TOTAL_DISCOUNT'].sum()
july_invited_premium = july_data['INVITED_PREMIUM'].sum()
july_discount_rate = (july_total_discount / july_invited_premium)

# Print results
print(f"Discount Rate for June: {june_discount_rate:.2%}")
print(f"Discount Rate for July: {july_discount_rate:.2%}")
```

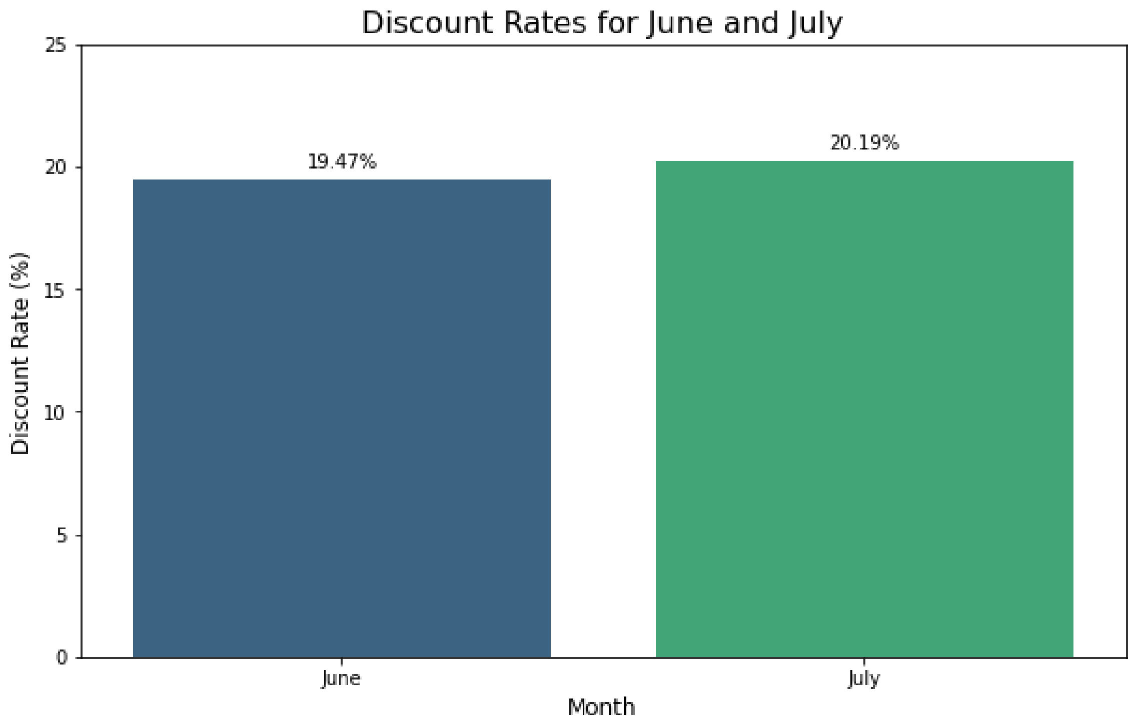
```
Discount Rate for June: 19.47%
Discount Rate for July: 20.19%
```

- I calculated the discount rate for June and July by dividing the total discounts (TOTAL_DISCOUNT) by the total invited premium (INVITED_PREMIUM).

- The discount rate tells us what percentage of the premium value was offered as discounts to customers.

- The results showed that:
  - In June, the discount rate was 19.47%.
  - In July, the discount rate increased slightly to 20.19%.
  - This increase in discounts aligns with the slightly higher save rate in July (74.48% compared to 74.11% in June), suggesting that offering higher discounts may have positively influenced customer renewals."

In [13]:
```
# Visualization of Discount Rates
discount_rates = pd.DataFrame({
    'Month': ['June', 'July'],
    'Discount Rate (%)': [june_discount_rate * 100, july_discount_rate * 10
})

plt.figure(figsize=(10, 6))
sns.barplot(x='Month', y='Discount Rate (%)', data=discount_rates, palette=
plt.title('Discount Rates for June and July', fontsize=16)
plt.ylabel('Discount Rate (%)', fontsize=12)
plt.xlabel('Month', fontsize=12)
plt.ylim(0, 25)
for index, value in enumerate(discount_rates['Discount Rate (%)']):
    plt.text(index, value + 0.5, f'{value:.2f}%', ha='center', fontsize=10)
plt.show()
```



Actionable Insights:

- July showed slightly higher save rates and discount rates, suggesting that offering more discounts might have encouraged renewals.

- The save rate for the full dataset is an average that accounts for months with both higher and lower performance, while the save rate for July reflects only one specific, better-performing period. July's performance likely benefited from factors like higher

Task 2

Identify the highest and lowest performing team and agent (on save rate) for June and July

In [14]:
```
save_rate_team_june = june_data.groupby('TEAM_NAME').apply(lambda x: x[x['R

highest_team_june = save_rate_team_june.idxmax(), save_rate_team_june.max()
lowest_team_june = save_rate_team_june.idxmin(), save_rate_team_june.min()

print(f"Highest Performing Team in June: {highest_team_june[0]} with Save R
print(f"Lowest Performing Team in June: {lowest_team_june[0]} with Save Rat
```

```
Highest Performing Team in June: Team24 with Save Rate: 76.73%
Lowest Performing Team in June: Team22 with Save Rate: 0.00%
```

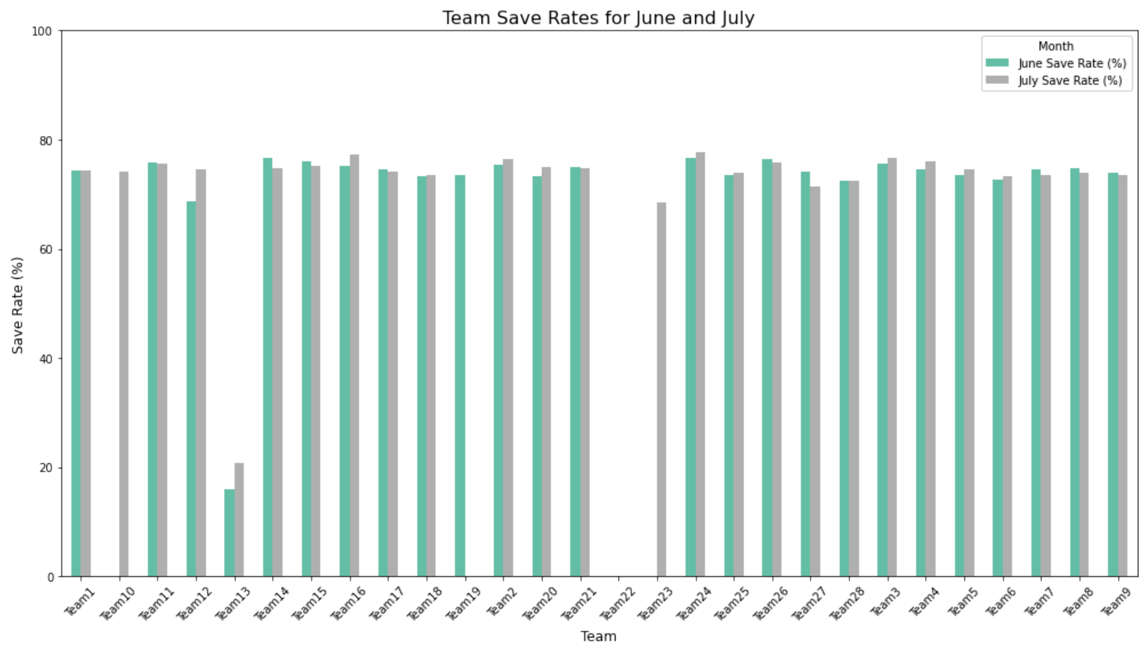
- The highest-performing team (Team24) achieved a save rate of 76.73%, meaning that 76.73% of their transactions in June led to successful renewals.
- The lowest-performing team (Team22) had a save rate of 0.00%, indicating that none of their transactions in June resulted in renewals.

In [15]:
```
save_rate_team_july = july_data.groupby('TEAM_NAME').apply(lambda x: x[x['R

highest_team_july = save_rate_team_july.idxmax(), save_rate_team_july.max()
lowest_team_july = save_rate_team_july.idxmin(), save_rate_team_july.min()

print(f"Highest Performing Team in June: {highest_team_june[0]} with Save R
print(f"Lowest Performing Team in June: {lowest_team_june[0]} with Save Rat
```

```
Highest Performing Team in June: Team24 with Save Rate: 76.73%
Lowest Performing Team in June: Team22 with Save Rate: 0.00%
```

- The team with the highest Save Rate is Team24, with a Save Rate of 76.73%. This means that 76.73% of the cases handled by Team24 resulted in successful renewals.
- The team with the lowest Save Rate is Team22, with a Save Rate of 0.00%. This means that none of the cases handled by Team22 resulted in successful renewals, which is a significant concern for performance.

In [16]:
```python
# Save Rates for Teams in June and July
team_save_rates = pd.DataFrame({
    'Team': save_rate_team_june.index.union(save_rate_team_july.index),
    'June Save Rate (%)': save_rate_team_june.reindex(save_rate_team_june.i
    'July Save Rate (%)': save_rate_team_july.reindex(save_rate_team_june.i
}).set_index('Team')

team_save_rates.plot(kind='bar', figsize=(14, 8), colormap='Set2')
plt.title('Team Save Rates for June and July', fontsize=16)
plt.ylabel('Save Rate (%)', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.legend(title='Month')
plt.ylim(0, 100)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [17]:
```python
save_rate_agent_june = june_data.groupby('OPERATOR_ID').apply(lambda x: x[x

highest_agent_june = save_rate_agent_june.idxmax(), save_rate_agent_june.ma
lowest_agent_june = save_rate_agent_june.idxmin(), save_rate_agent_june.min

print(f"Highest Performing Agent in June: {highest_agent_june[0]} with Save
print(f"Lowest Performing Agent in June: {lowest_agent_june[0]} with Save R
```

```
Highest Performing Agent in June: Agent47 with Save Rate: 85.85%
Lowest Performing Agent in June: Agent100 with Save Rate: 0.00%
```

1. Agent47 is performing exceptionally well, securing renewals in the vast majority of their cases. This could indicate they have a solid strategy or approach for handling cases, which other agents could potentially learn from.

2. On the other hand, **Agent100's Save Rate of 0% is concerning**, suggesting that none of their cases resulted in renewals. It may indicate issues such as:

- **A lack of effective strategies to retain customers.**
- **Possible challenges in communication or follow-up with clients.**

- **Training or resource needs.**

In [18]:
```
save_rate_agent_july = july_data.groupby('OPERATOR_ID').apply(lambda x: x[x

highest_agent_july = save_rate_agent_july.idxmax(), save_rate_agent_july.ma
lowest_agent_july = save_rate_agent_july.idxmin(), save_rate_agent_july.min

print(f"Highest Performing Agent in July: {highest_agent_july[0]} with Save
print(f"Lowest Performing Agent in July: {lowest_agent_july[0]} with Save R
```

```
Highest Performing Agent in July: Agent244 with Save Rate: 100.00%
Lowest Performing Agent in July: Agent100 with Save Rate: 0.00%
```

- Agent244 is excelling with a perfect Save Rate, while Agent100 is facing challenges.
- We have to identify the factors that contribute to these outcomes can help improve team performance overall.
- Factors may be:
  - Lack of effective renewal strategies.
  - Insufficient customer follow-up or engagement.
  - Potential training gaps or other obstacles that hinder their ability to close renewals.

Actionable Findings:

1. High performers can share strategies.
2. Low performers may need training or other support.

Recommondations:

- Teams like Team24 and agents like Agent47 / Agent244 can share their techniques and strategies through workshops or mentoring sessions.
- Focus on upskilling low-performing teams like Team22 and agents like Agent100 by providing specific training on customer engagement, objection handling, and renewal follow-ups.

TASK 3

What was the impact in a change in mix of tenure of callers in July compared to June ?

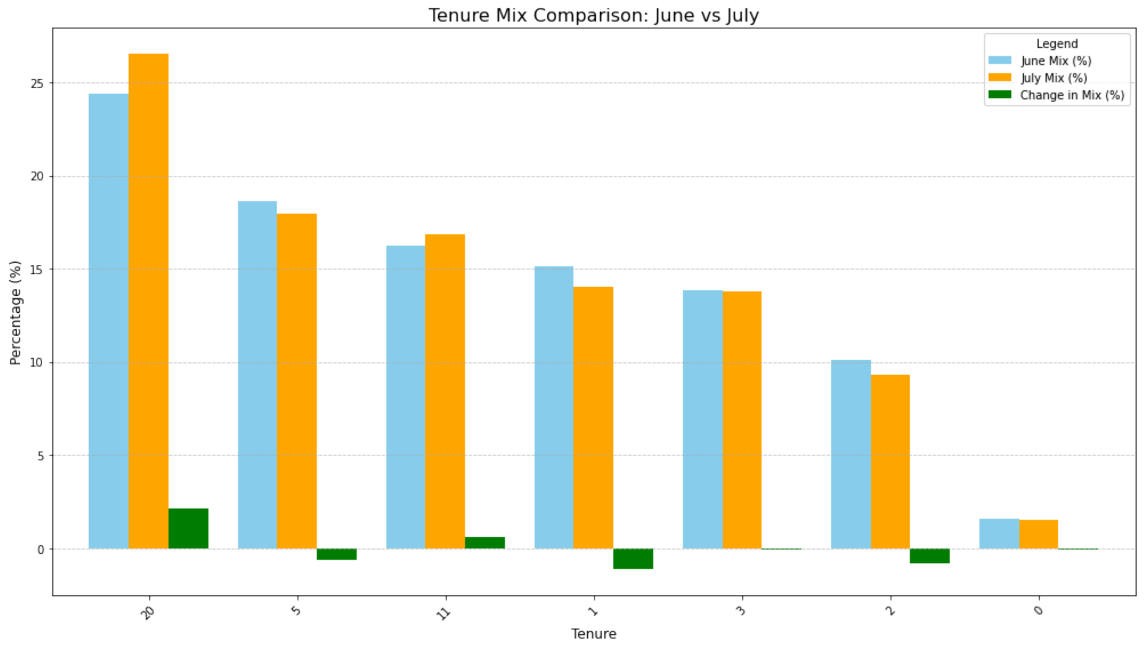
In [19]:
```
june_data.columns
```

Out[19]:
```
Index(['TRANSACTION_DATE', 'Transaction Month', 'TENURE', 'REPEAT_CALLER',
       'OPERATOR_ID', 'PAYMENT_TYPE', 'TYPE', 'SAVE_TYPE', 'MEMBERSHIP_TYP
E',
       'PRODUCT_TYPE', 'SITE_NAME', 'TEAM_NAME', 'SAM_RENEWAL_OFFER',
       'INVITED_PREMIUM', 'TOTAL_DISCOUNT', 'RENEWAL_PREMIUM', 'Renewed?',
       'Discount Given'],
      dtype='object')
```

In [20]:
```
june_tenure_mix = june_data['TENURE'].value_counts(normalize=True) * 100
july_tenure_mix = july_data['TENURE'].value_counts(normalize=True) * 100

tenure_comparison = pd.DataFrame({
    'June Mix (%)': june_tenure_mix,
    'July Mix (%)': july_tenure_mix
})

tenure_comparison['Change in Mix (%)'] = tenure_comparison['July Mix (%)']

print("Tenure Mix Comparison (June vs July):")
print(tenure_comparison)
```

```
Tenure Mix Comparison (June vs July):
    June Mix (%)  July Mix (%)  Change in Mix (%)
20     24.398412     26.541771           2.143359
5      18.610971     17.971534          -0.639438
11     16.259188     16.832499           0.573311
1      15.157124     14.027557          -1.129568
3      13.848557     13.768858          -0.079700
2      10.121869      9.311273          -0.810596
0       1.603878      1.546509          -0.057369
```

In [21]:
```
# Tenure Mix Comparison
tenure_comparison.plot(kind='bar', figsize=(14, 8), color=['skyblue', 'oran
plt.title('Tenure Mix Comparison: June vs July', fontsize=16)
plt.ylabel('Percentage (%)', fontsize=12)
plt.xlabel('Tenure', fontsize=12)
plt.legend(['June Mix (%)', 'July Mix (%)', 'Change in Mix (%)'], title='Le
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Impact:

- The Tenure Mix Comparison between June and July reveals notable shifts in the distribution of customer tenure.

- Customers with a tenure of 20 months represent the largest segment, increasing from 24.40% in June to 26.54% in July, marking a 2.14% growth, which indicates that long-tenured customers may be renewing at a higher rate or new renewals are concentrated among this group.
- Conversely, customers with a tenure of 1 month and 2 months experienced declines of 1.13% and 0.81%, respectively, suggesting potential challenges in retaining newer customers.
- The slight drop for the 5-month group (-0.64%) and the relatively stable proportions for other tenures indicate a more consistent behavior across mid-tenure customers.
- Overall, these changes highlight a potential need for tailored strategies to engage newer customers and maintain loyalty, while leveraging the positive momentum among longer-tenured customers to drive higher renewals.

Recommendations:

- Develop targeted onboarding and engagement strategies for customers with 1–2 months tenure to improve retention rates.
- Retain newer customers with better onboarding and follow-ups.
- Reward long-tenured customers to sustain their loyalty.
- Investigate slight declines in the 5-month tenure group to identify any specific issues (e.g., product dissatisfaction or lack of communication).
- Regularly analyze tenure-based trends to ensure strategies remain adaptive to shifts in customer behavior.

Task 4:

What kind of things would impact the save rates month to month ?

- Customer Behavior: New customers are more price-sensitive, while long-term ones are more loyal.
- Discounts: Higher discounts can encourage renewals, but over-dependence may reduce profitability.
- Agent Performance: Skilled agents and efficient teams drive higher save rates.
- External Factors: Economic conditions, competitor actions, and even natural events can affect customer decisions.
- Payment Type: Automatic payments lead to higher renewal rates.

General Understanding from this Task as per My Knowledge:

- Trends: Save rates improve with higher discounts and better service.
- Performance Insights: Exceptional agents and teams stand out due to their effective strategies, while poor performers need support.
- Customer Behavior: Long-tenured customers are loyal, but short-tenured ones need attention.
- Monthly Changes: External factors, pricing strategies, and customer service all play a role in month-to-month save rate fluctuations.