Ferocia Incentive Impact Analysis

Summary:

- The $10 incentive increased customer acquisition while lowering overall customer quality.
- Sign-ups in February and onboarding and activation rates declined across every milestone which indicates an influx of short-term or bonus driven users.

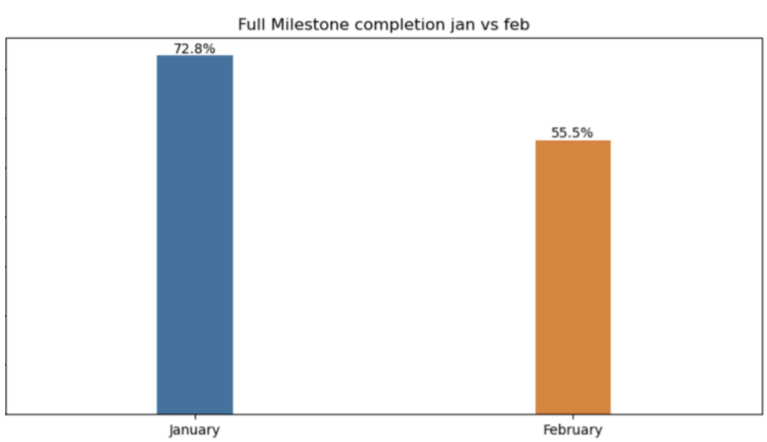
Analysis Focus:

- Purchased signups' "Volume vs. quality trade-off" (did we buy low quality signups?)
- "Funnel performance" (where do people drop off?)
- Acquisition "ROI calculation" (cost per quality customer acquired)

Analyzed the "Gen Z, Gen Y, Gen X, etc." cohorts and how they reacted to the incentive changes. Also recorded a "two-proportion z-test" to prove that the quality drop between January and February was statistically significant (it was—$p < 0.0001$). One can conclude that it was not merely a random occurrence.

Key Metrics: Onboarding Completion % by month

Metric	January	February	Change
Total New Customers	1701	4668	**+174%**
% Completed First Funding (onboarded)	89%	79%	**Up 10%**
% Fully Onboarded (All milestones)	73%	55%	**Down 18%**

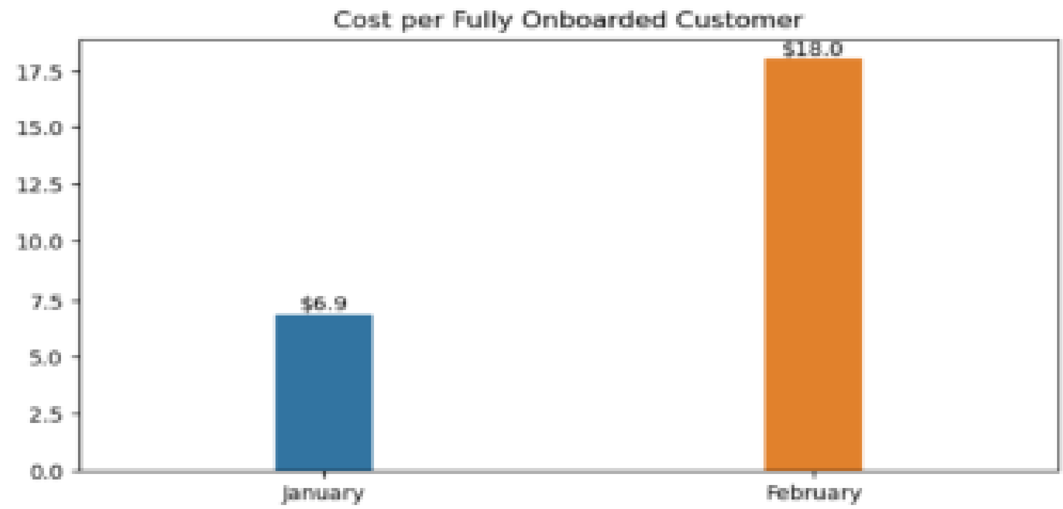


Insights:

- Doubling incentives boosted reach but lowered onboarding consistency.
- Drop in funding and purchase completion indicates weaker engagement quality.
- Customers acquired under higher incentives show slower milestone progression.

ROI & Business Implications

Month	Sign-ups	Fully Onboarded	Bonus	Total Cost	Cost/Good Customer
Jan	1,701	1,238	$5	$8,505	$6.9
Feb	4,668	2,593	$10	$46,680	$18.0



Outcomes:

- Operational expenses rose ~450%, while high-value customers increased by almost 2 times higher.
- Cost for high-value customers increased 2.6 times ($6.9 → $18).
- The probability testing ($p < 0.01$) confirms that onboarding decline is truly significant.
- February campaign delivered lower ROI despite higher volume.

Recommendations:

1. Transfer to funding or first purchase incentive.
2. **Use tiered bonuses** by generation to target real engagement.
3. A/B test to ensure the best option for incentive timing and amount.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore", category=pd.errors.SettingWithCopyWarning)
```

In [3]:
```
pwd
```

Out[3]: '/Users/ratikpant'

In [4]:
```
df = pd.read_json('/Users/ratikpant/Desktop/ferocia/ferocia-data-analyst-takehome/take-home-data_sampled-2025-0
```

data dictionary

1. “joined” is the date the customer joined Up.
2. "generation" gives the customer a label (to group cohorts) based on the customers date of birth.
3. “activated_card” means they received their plastic Up card in the mail and activated it via the app. The field is blank if they haven't yet activated their plastic card.
4. “first_purchase" logs the date they made their first purchase on Up (this can precede card activation if they use a digital wallet). This field is blank if they are yet to make a purchase.
5. “fifth_purchase" logs the date they made their fifth purchase. This needs to be no more than 30 days after their joined date in order for them to have received an additional $5 bonus. This field is blank if they are yet to make 5 purchases.
6. “first_funding” logs the date they first transferred money into Up (from an external account).
7. “fifth_funding” logs the date they made their fifth transfer into Up.
8. “last_transaction" records the date of the customers last transaction. A customer is considered
9. active” if their last transaction was within the last 30 days.
10. columns with the `inviter_` prefix contain milestone dates for the person who invited that customer.

Objective

“ Assessing the impact of increasing the referral bonus from $5 to$10 on new-customer onboarding in January vs February 2021. Approach: Cleaned and structured customer-level data, defined onboarding milestones,assumptions and compared engagement rates across months and generations.”

In [5]:
```
df
```

Out[5]:

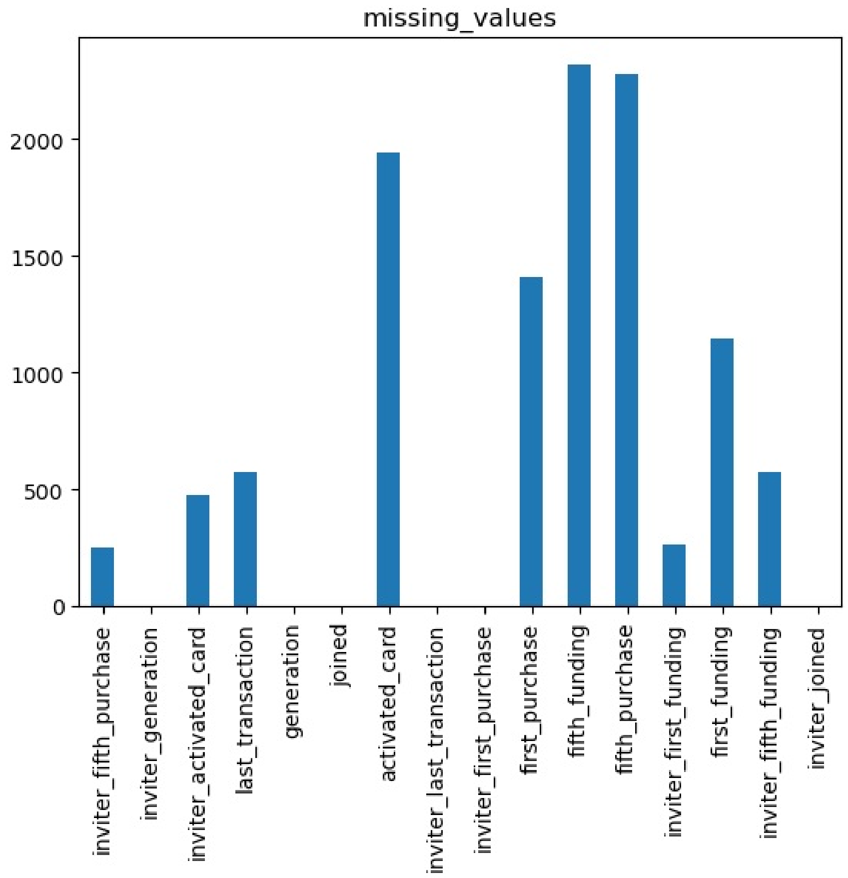
	inviter_fifth_purchase	inviter_generation	inviter_activated_card	last_transaction	generation	joined	activated_card	inviter_la
0	2020-12-29	GEN Z	2020-12-23	2021-05-14	GEN Z	2021-01-01	2021-01-17	
1	2020-02-12	GEN Y	2020-01-26	2023-02-24	GEN X	2021-01-01	2021-01-22	
2	2019-06-06	GEN Y	2019-06-05	2024-11-29	GEN Y	2021-01-01	2021-01-12	
3	2021-01-01	GEN X	2020-09-16	2024-04-24	GEN Y	2021-01-01	2021-02-06	
4	2020-07-04	GEN Z	2020-12-05	2025-03-27	GEN Z	2021-01-01	2021-01-13	
...	...	...	...	...	...	...	...	
6389	2021-03-03	GEN Z	None	None	GEN Z	2021-02-28	None	
6390	2021-02-26	GEN Y	2021-02-23	2021-02-28	GEN X	2021-02-28	None	
6391	2021-03-02	GEN Y	2021-02-18	None	GEN Y	2021-02-28	None	
6392	2020-12-24	GEN X	2020-12-21	2023-02-12	GEN Y	2021-02-28	2021-03-09	
6393	2021-03-02	GEN Z	None	2023-12-19	GEN X	2021-02-28	None	

6394 rows × 16 columns

missing values

In [6]:

```
df.isnull().sum().plot(kind = 'bar', title = 'missing_values')
plt.show()
```



there are missing values , but those missing values could mean something for the business, so we will look into it later on in the process

checking for duplicates rows

In [7]:
```
df[df.duplicated(keep=False)]
```

Out[7]:

	inviter_fifth_purchase	inviter_generation	inviter_activated_card	last_transaction	generation	joined	activated_card	inviter_la
1963	2020-02-11	GEN X	2018-12-07	None	GEN Y	2021-02-02	None	
1988	2020-02-11	GEN X	2018-12-07	None	GEN Y	2021-02-02	None	
2638	2020-02-14	BOOMER	2020-03-10	2021-02-07	GEN X	2021-02-07	None	
2651	2020-02-14	BOOMER	2020-03-10	2021-02-07	GEN X	2021-02-07	None	
2688	2021-02-03	GEN Y	2020-12-11	None	GEN Y	2021-02-07	None	
2706	2021-02-03	GEN Y	2020-12-11	None	GEN Y	2021-02-07	None	
3240	None	GEN Z	2021-02-10	2021-02-10	GEN Z	2021-02-10	None	
3242	None	GEN Z	2021-02-10	2021-02-10	GEN Z	2021-02-10	None	
3419	2021-02-12	GEN Z	2021-03-07	None	GEN Z	2021-02-12	None	
3422	2021-02-12	GEN Z	2021-03-07	None	GEN Z	2021-02-12	None	
3515	2020-12-26	GEN Y	2020-12-13	2021-02-12	BOOMER	2021-02-12	None	
3526	2020-12-26	GEN Y	2020-12-13	2021-02-12	BOOMER	2021-02-12	None	
4260	2021-01-14	GEN Y	2021-01-12	None	GEN X	2021-02-17	None	
4263	2021-01-14	GEN Y	2021-01-12	None	GEN X	2021-02-17	None	
4721	None	GEN X	None	2021-02-20	GEN Y	2021-02-20	None	
4726	None	GEN X	None	2021-02-20	GEN Y	2021-02-20	None	
4731	None	GEN X	None	2021-02-20	GEN Y	2021-02-20	None	
4741	None	GEN X	None	2021-02-20	GEN Y	2021-02-20	None	
4758	None	GEN X	None	2021-02-20	GEN Y	2021-02-20	None	
4956	2021-02-23	GEN Z	2021-03-11	None	GEN Z	2021-02-22	None	
4983	None	GEN Z	None	None	GEN Z	2021-02-22	None	
4985	None	GEN Z	None	None	GEN Z	2021-02-22	None	
5022	2021-02-23	GEN Z	2021-03-11	None	GEN Z	2021-02-22	None	
5036	2021-10-20	GEN Z	2021-02-28	None	GEN Z	2021-02-22	None	
5074	2021-10-20	GEN Z	2021-02-28	None	GEN Z	2021-02-22	None	
5211	2021-04-23	GEN Z	2021-02-15	None	GEN Z	2021-02-23	None	
5243	2021-04-23	GEN Z	2021-02-15	None	GEN Z	2021-02-23	None	
5567	2021-10-20	GEN Z	2021-05-19	None	GEN Z	2021-02-24	None	
5571	2021-10-20	GEN Z	2021-05-19	None	GEN Z	2021-02-24	None	
5698	2021-10-20	GEN Z	2021-05-19	None	GEN Z	2021-02-25	None	

5761	2021-10-20	GEN Z	2021-05-19	None	GEN Z	2021-02-25	None
5903	2021-02-25	GEN Y	None	2021-02-26	GEN Y	2021-02-26	None
6011	2021-02-25	GEN Y	None	2021-02-26	GEN Y	2021-02-26	None
6044	2021-03-04	GEN Z	None	None	GEN Z	2021-02-27	None
6068	2021-03-04	GEN Z	None	None	GEN Z	2021-02-27	None
6094	2021-02-25	GEN Y	None	2021-02-27	GEN Y	2021-02-27	None
6099	2021-02-28	GEN Z	2021-02-27	None	GEN Z	2021-02-27	None
6104	2021-02-28	GEN Z	2021-02-27	None	GEN Z	2021-02-27	None
6169	2021-02-25	GEN Y	None	2021-02-27	GEN Y	2021-02-27	None
6170	2021-02-27	GEN Z	2021-03-08	2021-02-27	GEN Z	2021-02-27	None
6176	2021-02-27	GEN Z	2021-03-08	2021-02-27	GEN Z	2021-02-27	None
6177	2021-02-25	GEN Y	None	2021-02-27	GEN Y	2021-02-27	None
6182	2021-02-25	GEN Y	None	2021-02-27	GEN Y	2021-02-27	None
6314	2021-02-27	GEN Z	None	2021-02-28	GEN X	2021-02-28	None
6318	2021-02-27	GEN Z	None	2021-02-28	GEN X	2021-02-28	None

In [8]:
```
# removing duplicates
```

In [9]:
```
df = df.drop_duplicates()
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
#converting all the date colums into datetype
```

In [12]:
```
col_to_change = ['inviter_fifth_purchase', 'inviter_activated_card', 'last_transaction', 'joined',
            'activated_card', 'inviter_last_transaction', 'inviter_first_purchase', 'first_purchase', 'fifth_fu
            'fifth_purchase', 'inviter_first_funding', 'first_funding', 'inviter_fifth_funding', 'inviter_joine
```

In [13]:
```
for col in col_to_change:
    df[col] = pd.to_datetime(df[col], errors = 'coerce')
```

In [14]:
```
df[col_to_change].dtypes
```

Out[14]:
```
inviter_fifth_purchase      datetime64[ns]
inviter_activated_card      datetime64[ns]
last_transaction            datetime64[ns]
joined                      datetime64[ns]
activated_card              datetime64[ns]
inviter_last_transaction    datetime64[ns]
inviter_first_purchase      datetime64[ns]
first_purchase              datetime64[ns]
fifth_funding               datetime64[ns]
fifth_purchase              datetime64[ns]
inviter_first_funding       datetime64[ns]
first_funding               datetime64[ns]
inviter_fifth_funding       datetime64[ns]
inviter_joined              datetime64[ns]
dtype: object
```

EDA

NEW CUSTOMERS

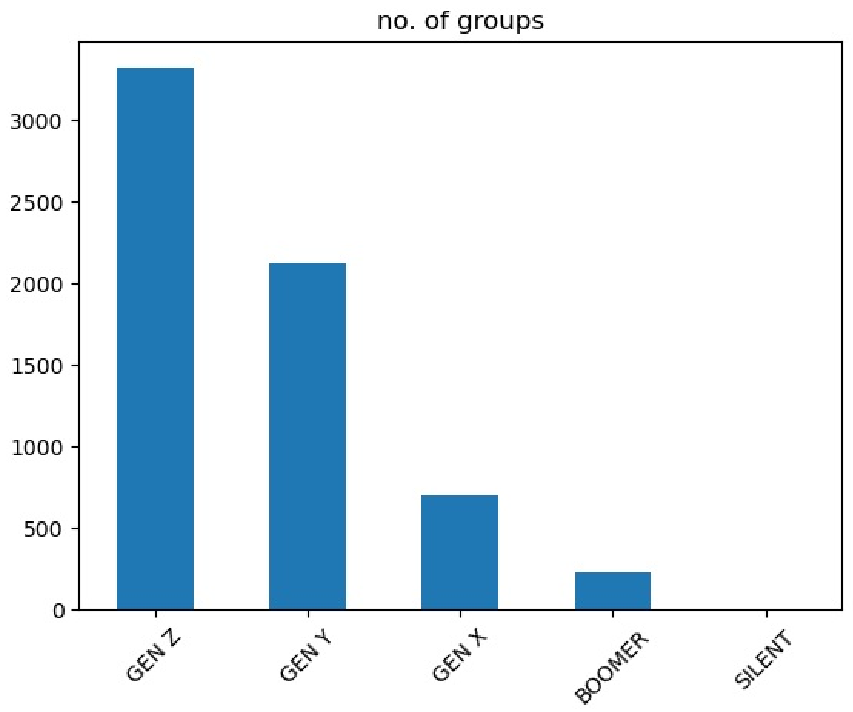
```
In [15]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6369 entries, 0 to 6393
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   inviter_fifth_purchase  6122 non-null   datetime64[ns]
 1   inviter_generation      6369 non-null   object
 2   inviter_activated_card  5904 non-null   datetime64[ns]
 3   last_transaction        5808 non-null   datetime64[ns]
 4   generation              6369 non-null   object
 5   joined                  6369 non-null   datetime64[ns]
 6   activated_card          4448 non-null   datetime64[ns]
 7   inviter_last_transaction  6369 non-null   datetime64[ns]
 8   inviter_first_purchase  6368 non-null   datetime64[ns]
 9   first_purchase          4983 non-null   datetime64[ns]
 10  fifth_funding           4076 non-null   datetime64[ns]
 11  fifth_purchase          4116 non-null   datetime64[ns]
 12  inviter_first_funding   6105 non-null   datetime64[ns]
 13  first_funding           5234 non-null   datetime64[ns]
 14  inviter_fifth_funding   5803 non-null   datetime64[ns]
 15  inviter_joined          6369 non-null   datetime64[ns]
dtypes: datetime64[ns](14), object(2)
memory usage: 845.9+ KB
```

```
In [16]: new_cx = df[['last_transaction', 'generation','joined', 'activated_card','first_funding','first_purchase','fiftl
```

No. of New customer groups

```
In [17]: new_cx['generation'].value_counts().sort_values(ascending = False).plot(kind = 'bar', title = 'no. of groups')
plt.xlabel(' ')
plt.xticks(rotation = 45)
plt.show()
```

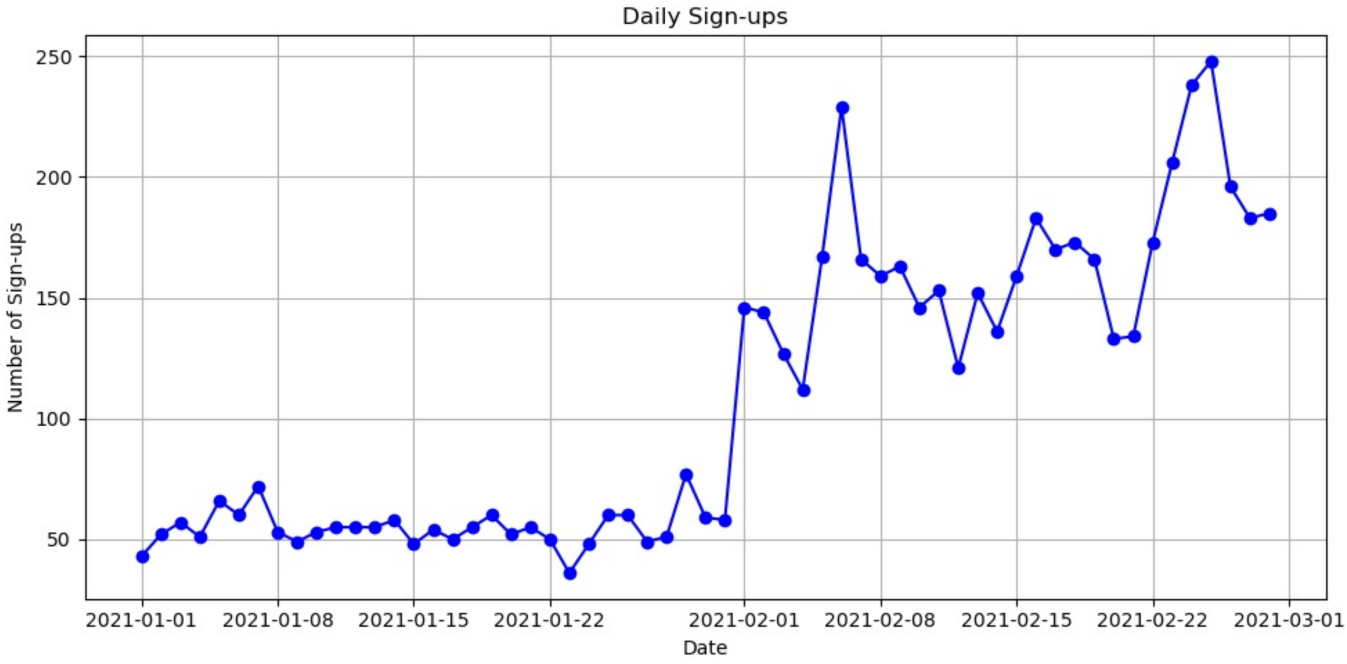


genz and geny customers signed up the most in the incentive campaign

CHECKING RAW SIGN UPS

```
In [18]: daily_signups = (
    new_cx['joined']
    .value_counts()
    .sort_index()
    .reset_index()
)
```

```python
plt.figure(figsize=(10,5))
plt.plot(daily_signups['joined'], daily_signups['count'], color='blue', marker = 'o')
plt.title('Daily Sign-ups')
plt.xlabel('Date')
plt.ylabel('Number of Sign-ups')
plt.grid(True)
plt.tight_layout()
plt.show()
```



we see there was a serious uptick in the number of raw sign ups in the month of february when the incentive were increased

In [19]:
```python
#converting date type to month name for clarity
new_cx['month_joined'] = new_cx['joined'].dt.month_name()
```

In [20]:
```python
# checking true values
new_cx['first_deposit'] = new_cx['first_funding'].notna()
new_cx['first_deposit']
```

Out[20]:
```
0        True
1        True
2        True
3        True
4        True
        ...
6389    False
6390     True
6391    False
6392     True
6393     True
Name: first_deposit, Length: 6369, dtype: bool
```

COMPLETED ALL MILESTONES:

ASSUMPTION: cx completed first purchase and funding plus fifth purchase and funding.

In [21]:
```python
#fully onboarded
new_cx['fully_onboarded'] = new_cx['first_funding'].notna()  & new_cx['first_purchase'].notna() & new_cx['fifth_
```

In [22]:
```python
full_milestones = new_cx.groupby('month_joined')['fully_onboarded'].mean().mul(100).round(2).reset_index(name =
full_milestones
```

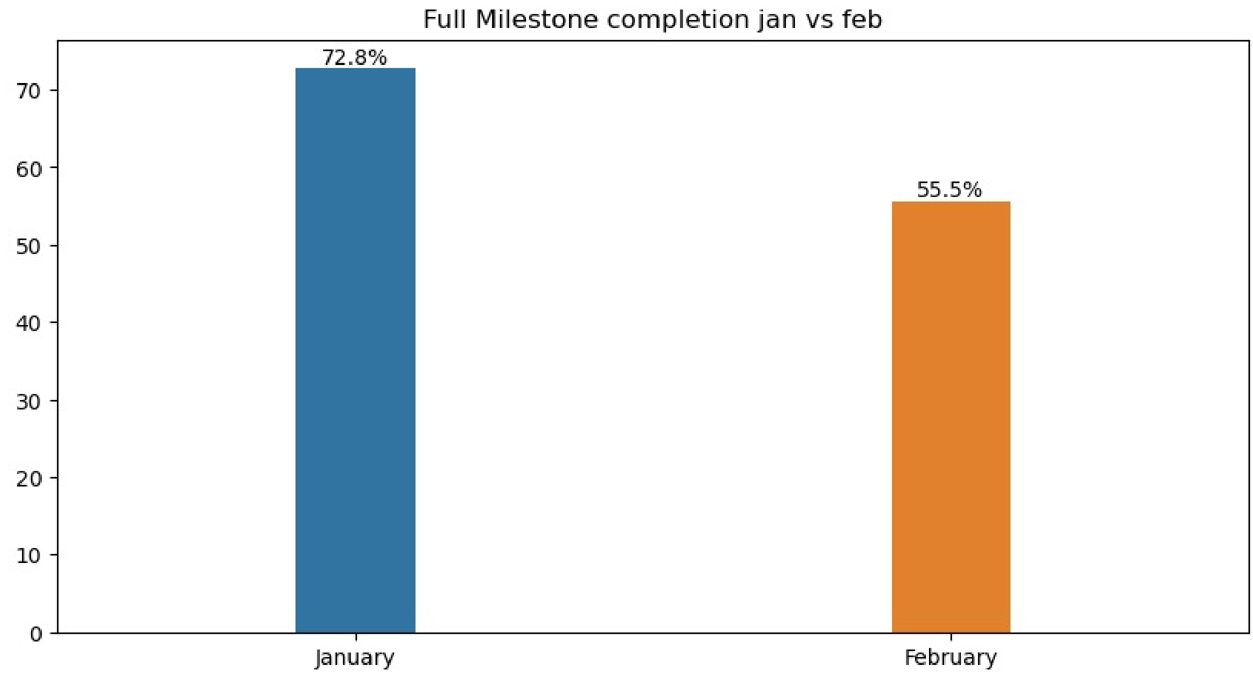
Out[22]:

	month_joined	percent
0	February	55.55
1	January	72.78

In [23]:

```
plt.figure(figsize=(10,5))
ax = sns.barplot(
    data=full_milestones,
    x='month_joined',
    y='percent', width = 0.2,order=['January', 'February']
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Full Milestone completion jan vs feb')

# Annotate bars with percentage
for p in ax.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax.annotate(f'{height:.1f}%',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



the result shows that the campaign run may have pulled ungenuine cx's.

we may have to recalibrate on strategy to offer a bonus when a new cx's completes a certain milestone and not just a signup

calculating performance of each milestone (JAN VS FEB)

In [24]:

```
new_cx['first_spent'] = new_cx['first_purchase'].notna()
new_cx['fifth_deposit'] = new_cx['fifth_funding'].notna()
new_cx['fifth_spent'] = new_cx['fifth_purchase'].notna()
new_cx.head()
```

Out[24]:

	last_transaction	generation	joined	activated_card	first_funding	first_purchase	fifth_funding	fifth_purchase	month_joined	fir
0	2021-05-14	GEN Z	2021-01-01	2021-01-17	2021-01-01	2021-01-17	2021-01-01	2021-01-18	January	
1	2023-02-24	GEN X	2021-01-01	2021-01-22	2021-01-01	2021-01-31	2021-01-14	2021-02-01	January	
2	2024-11-29	GEN Y	2021-01-01	2021-01-12	2021-01-14	2021-01-18	2021-01-23	2021-01-25	January	
3	2024-04-24	GEN Y	2021-01-01	2021-02-06	2021-05-03	2021-05-06	2021-05-08	2021-05-10	January	
4	2025-03-27	GEN Z	2021-01-01	2021-01-13	2021-01-01	2021-03-27	2021-01-29	2021-06-12	January	

In [25]:
```
milestone_summary = new_cx[['month_joined', 'first_deposit','first_spent', 'fifth_deposit', 'fifth_spent']]
```

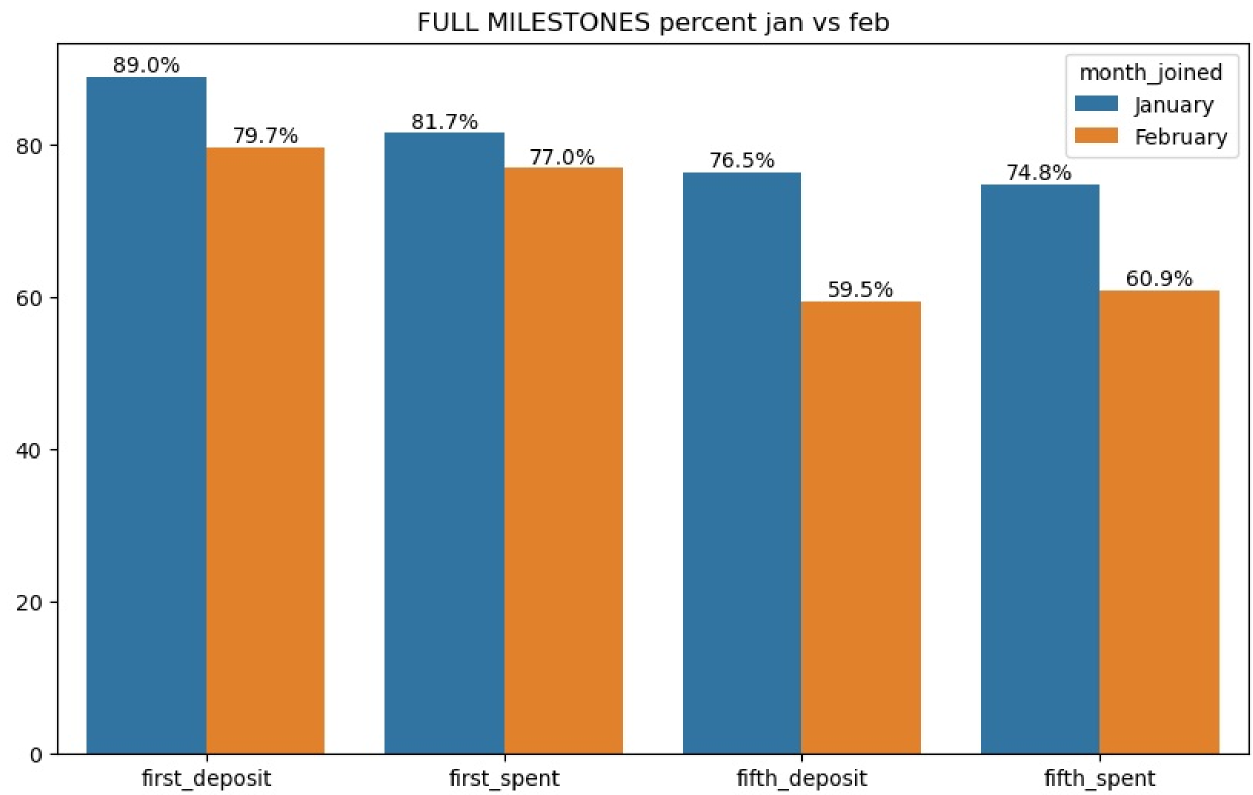
In [26]:
```
milestone_stat = milestone_summary.groupby('month_joined')[["first_deposit", "first_spent", "fifth_deposit", "f:
milestone_stat
```

Out[26]:

	month_joined	first_deposit	first_spent	fifth_deposit	fifth_spent
0	February	79.69	76.99	59.45	60.90
1	January	89.01	81.66	76.48	74.84

In [27]:
```
melted = milestone_stat.melt(
    id_vars='month_joined',
    var_name='Milestone',
    value_name='Completion (%)'
)
melted
plt.figure(figsize=(10,6))
ax7 = sns.barplot(
    data=melted,
    x='Milestone',
    y='Completion (%)', hue = 'month_joined',hue_order=['January', 'February']
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('FULL MILESTONES percent jan vs feb')

# Annotate bars with percentage
for p in ax7.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax7.annotate(f'{height:.1f}%',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



ROI CALCULATION:

WAS THE CAMPAIGN RUN WORTH the money spent?

In [28]:
```
roi_cust = new_cx[['month_joined', 'fully_onboarded']]
```

In [29]:
```
roi_cust = roi_cust.groupby('month_joined').agg(
signups = ('fully_onboarded', 'size'),
quality_customers = ('fully_onboarded', 'sum')).reset_index()
```

In [30]:
```
roi_cust
```

Out[30]:

	month_joined	signups	quality_customers
0	February	4668	2593
1	January	1701	1238

calculating total_bonus disbursed on new customers jan - 5 , feb -10

In [31]:
```
bonus = {'January': 5, 'February': 10}
roi_cust['bonus'] = roi_cust['month_joined'].map(bonus)
```

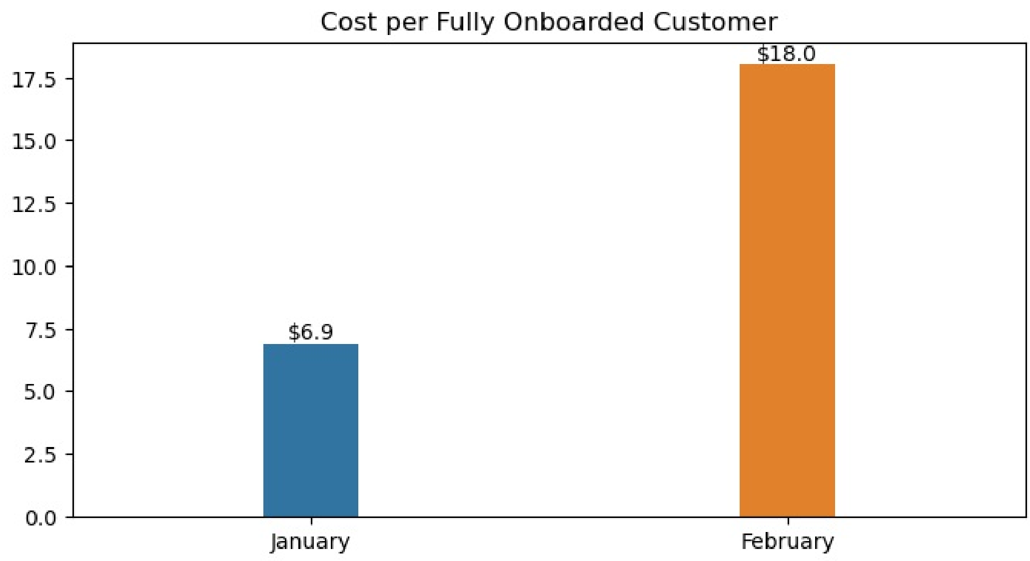
In [32]:
```
roi_cust['total_cost'] = roi_cust['signups'] * roi_cust['bonus']
roi_cust['cost_per_quality_customer'] = (roi_cust['total_cost'] / roi_cust['quality_customers']).round(2)
roi_cust
```

Out[32]:

	month_joined	signups	quality_customers	bonus	total_cost	cost_per_quality_customer
0	February	4668	2593	10	46680	18.00
1	January	1701	1238	5	8505	6.87

In [33]:
```
plt.figure(figsize=(8,4))
ax9 = sns.barplot(
    data=roi_cust,
    x='month_joined',
    y='cost_per_quality_customer', order=['January', 'February'],width = 0.2
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Cost per Fully Onboarded Customer')

# Annotate bars with percentage
for p in ax9.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax9.annotate(f'${height:.1f}',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



February’s 10 incentive attracted more customers but diluted user quality.

The campaign was not economically efficient, as engagement didn’t scale with spend.

cx_segmentation

In [34]:
```
cx_segmentation = new_cx[['generation', 'month_joined', 'fully_onboarded']]
cx_segmentation.head()
```

Out[34]:

	generation	month_joined	fully_onboarded
0	GEN Z	January	True
1	GEN X	January	True
2	GEN Y	January	True
3	GEN Y	January	True
4	GEN Z	January	True

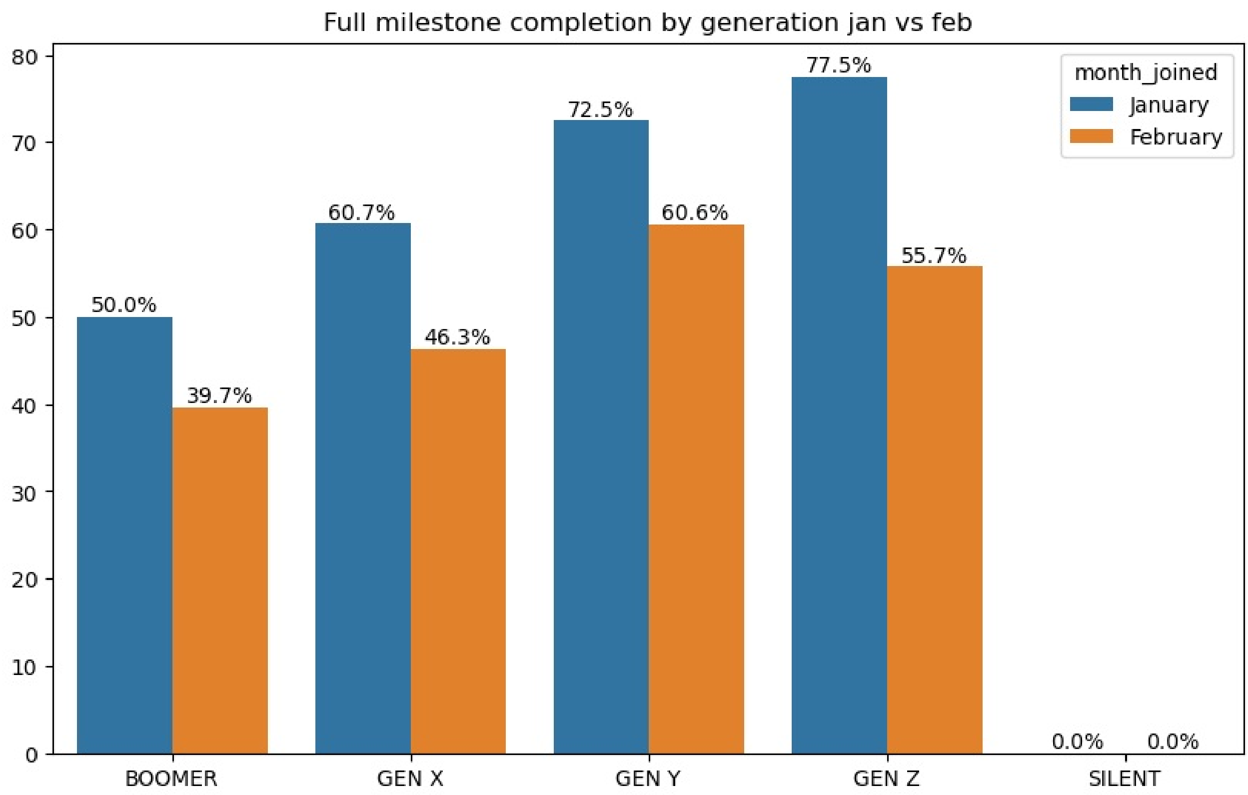
In [35]:
```
cx_seg = cx_segmentation.groupby(['generation', 'month_joined'])['fully_onboarded'].mean().mul(100).round(2).re:
cx_seg
```

Out[35]:

	generation	month_joined	percent
0	BOOMER	February	39.66
1	BOOMER	January	50.00
2	GEN X	February	46.30
3	GEN X	January	60.66
4	GEN Y	February	60.61
5	GEN Y	January	72.51
6	GEN Z	February	55.74
7	GEN Z	January	77.50
8	SILENT	February	0.00
9	SILENT	January	0.00

In [36]:
```
plt.figure(figsize=(10,6))
ax5 = sns.barplot(
    data=cx_seg,
    x='generation',
    y='percent', hue = 'month_joined',hue_order=['January', 'February']
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Full milestone completion by generation jan vs feb')

# Annotate bars with percentage
for p in ax5.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax5.annotate(f'{height:.1f}%',
                     (p.get_x() + p.get_width() / 2, height),
                     ha='center', va='bottom', fontsize=10)
plt.show()
```



“Segmenting by generation revealed that Gen Y and Gen Z users, who form the bulk of new customers, showed the sharpest decline in activation — suggesting the bonus appealed more broadly but attracted less committed users.”

onboarding completion; cx who made atleast first deposit jan vs feb

ASSUMPTION- cx first deposit qualifies as onboarding complete

In [37]:
```
# checking true values
new_cx['first_deposit'] = new_cx['first_funding'].notna()
```

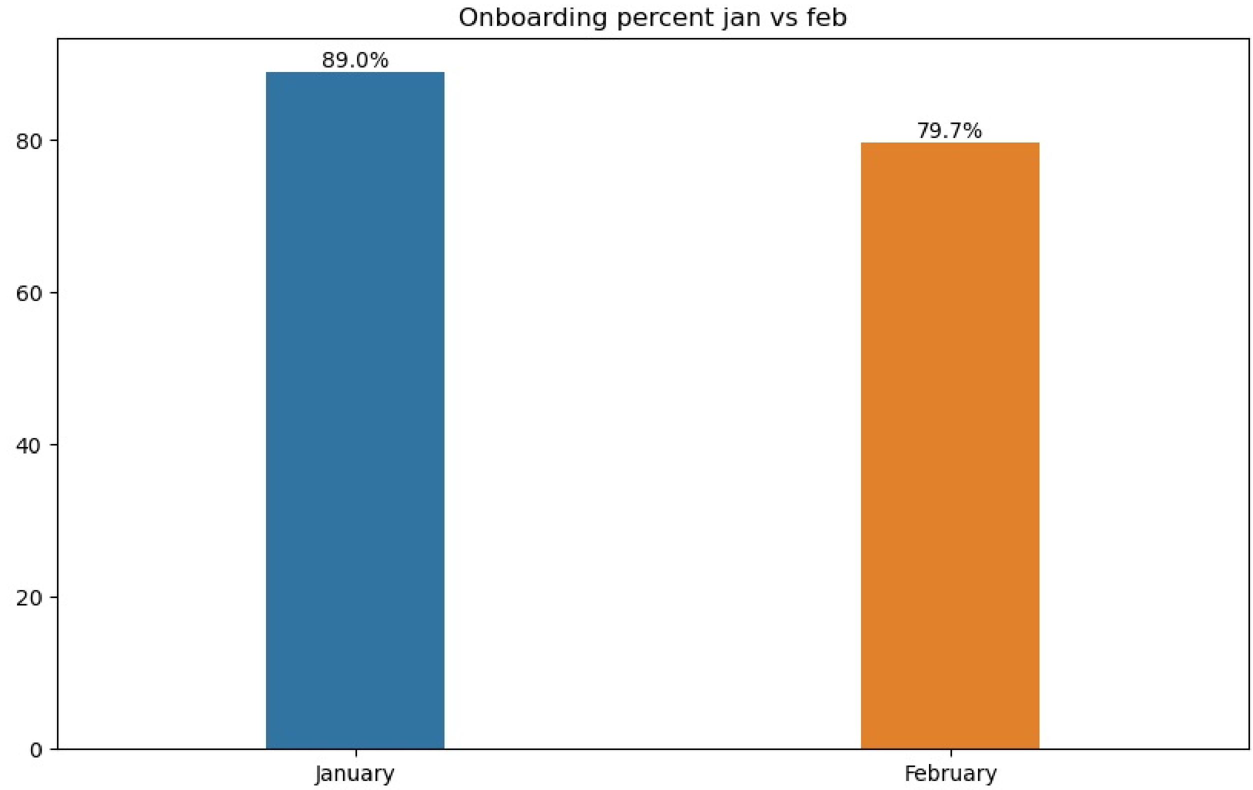
In [38]:
```
onboarding_completion = new_cx[['generation','month_joined', 'first_deposit']]
onboarding_completion = onboarding_completion.groupby('month_joined')['first_deposit'].mean().mul(100).round(2)
onboarding_completion
```

Out[38]:

	month_joined	onboarding_percent
0	February	79.69
1	January	89.01

In [39]:
```
plt.figure(figsize=(10,6))
ax1 = sns.barplot(
    data=onboarding_completion,
    x='month_joined',
    y='onboarding_percent', width = 0.3,order=['January', 'February']
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Onboarding percent jan vs feb')

# Annotate bars with percentage
for p in ax1.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax1.annotate(f'{height:.1f}%',
                     (p.get_x() + p.get_width() / 2, height),
                     ha='center', va='bottom', fontsize=10)
plt.show()
```



Despite doubling the referral incentive from 5 to 10 dollars in February, the proportion of new users who made their first deposit declined compared to January.”

In [40]:
```
#  recommendation we may have to check the quality of customers
```

card activation by month

In [41]:
```
#checking card activation true values
new_cx['card_active'] = new_cx['activated_card'].notna()
```

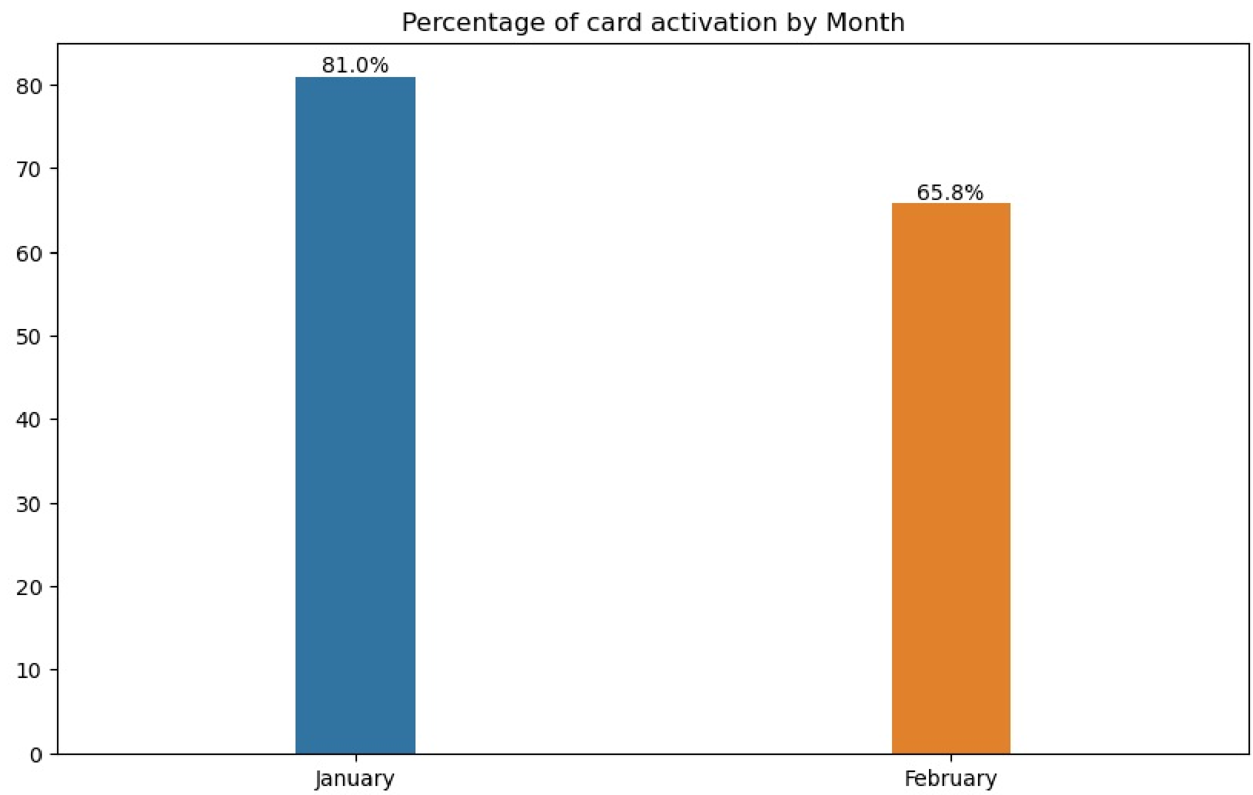
In [42]:
```
active_card = new_cx.groupby('month_joined')['card_active'].mean().mul(100).round(2).reset_index(name = 'percen
active_card
```

Out[42]:

	month_joined	percent_card_active
0	February	65.79
1	January	80.95

In [43]:
```
plt.figure(figsize=(10,6))
ax3 = sns.barplot(
    data=active_card,
    x='month_joined',
    y='percent_card_active',width = 0.2, order = ['January', 'February']
)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Percentage of card activation by Month')

# Annotate bars with percentage
for p in ax3.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax3.annotate(f'{height:.1f}%',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



card activation by generation & month

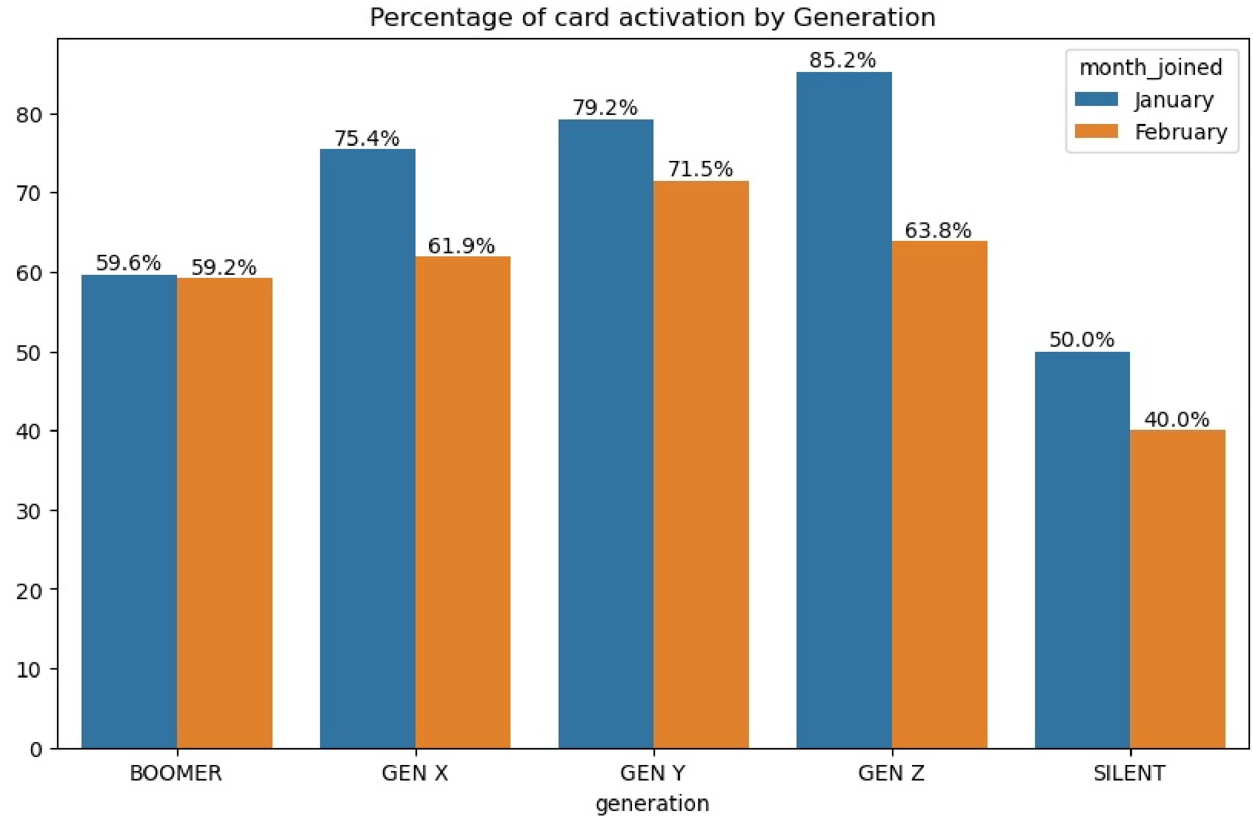
In [44]:
```
card_active = new_cx[['generation', 'month_joined', 'card_active']]
card_active = card_active.groupby(['generation', 'month_joined'])['card_active'].mean().mul(100).round(2).reset_
card_active
```

Out[44]:

	generation	month_joined	percent
0	BOOMER	February	59.20
1	BOOMER	January	59.62
2	GEN X	February	61.87
3	GEN X	January	75.41
4	GEN Y	February	71.46
5	GEN Y	January	79.20
6	GEN Z	February	63.84
7	GEN Z	January	85.21
8	SILENT	February	40.00
9	SILENT	January	50.00

In [45]:
```
plt.figure(figsize=(10,6))
ax6 = sns.barplot(
    data=card_active,
    x='generation',
    y='percent',
    hue='month_joined', hue_order = ['January', 'February']
)
plt.ylabel(' ')
plt.title('Percentage of card activation by Generation')

# Annotate bars with percentage
for p in ax6.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax6.annotate(f'{height:.1f}%',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



Card activation is popular amongst gen y and genz with most amount of customers in the group present, though genz took a beating in the month of february because of cx's only seeking incentive

In [46]:
```
#understanding [New customer's] churn rate, whether they have made a transaction in last 30 days or not
```

In [47]:
```
new_cx['active'] = new_cx['last_transaction'].notna()
```

In [48]:
```
cx_status = new_cx.groupby([ 'month_joined', 'active']).size().reset_index(name = 'count')
cx_total = new_cx.groupby('month_joined').size().reset_index(name= 'total_count')
cx_status = pd.merge(cx_status, cx_total, on = 'month_joined')
cx_status['percent'] = ((cx_status['count'] / cx_status['total_count']) * 100).round(2)
cx_status
```

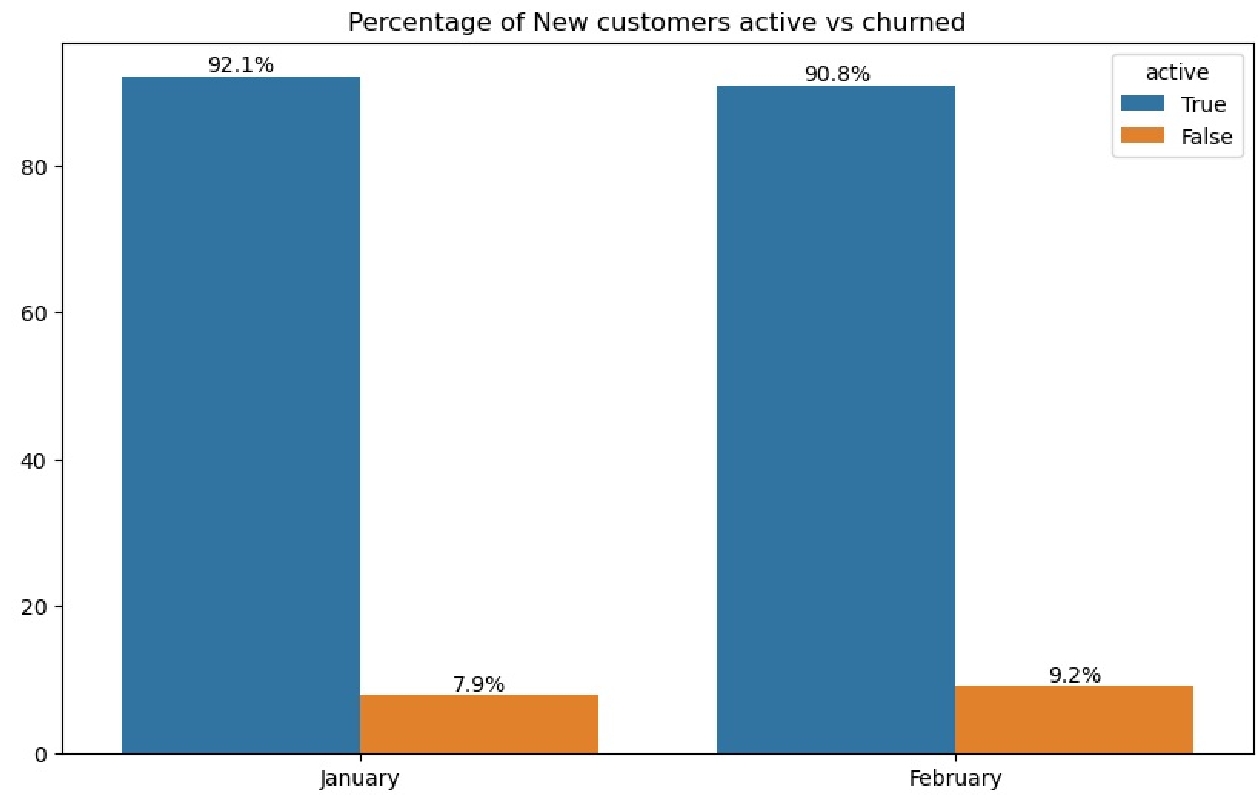
Out[48]:

	month_joined	active	count	total_count	percent
0	February	False	427	4668	9.15
1	February	True	4241	4668	90.85
2	January	False	134	1701	7.88
3	January	True	1567	1701	92.12

In [49]:
```
plt.figure(figsize=(10,6))
ax8 = sns.barplot(
    data=cx_status,
    x='month_joined',
    y='percent',
    hue = 'active',
    hue_order = [True, False],
    order = ['January', 'February']

)
plt.ylabel(' ')
plt.xlabel(' ')
plt.title('Percentage of New customers active vs churned')

# Annotate bars with percentage
for p in ax8.patches:
    height = p.get_height()
    if not pd.isna(height):
        ax8.annotate(f'{height:.1f}%',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=10)
plt.show()
```



Statistical testing

building on hypothesis:

Did the increase in incentive from 5 dollar in January to 10 dollar in February increase the number of good-quality customers?

--> Definition of “Good Quality Customer”

--> In this experiment, “good quality” means:

Customers who completed all 4 milestones:

1. first_deposit (funded once)
2. first_spent (made one purchase)
3. fifth_deposit (funded five times)
4. fifth_spent (made five purchases)

In [50]:
```
# building hypothesis
```

Null hypothesis (H_0):

The proportion of good-quality customers is the same in January and February.

Alternative hypothesis (H_1):

The proportion of good-quality customers changed (expected to decline) in February.

we select a two proportion z test to comare percentages between the groups who fully onboarded in january vs groups who fully onboarded in february

In [51]:
```
from statsmodels.stats.proportion import proportions_ztest
```

In [52]:
```
test_cx = new_cx.copy()
```

In [53]:
```
test_cx = test_cx[['month_joined', 'fully_onboarded']]
```

In [54]:
```
count = [
    test_cx.loc[test_cx['month_joined'] == 'January', 'fully_onboarded'].sum(),
    test_cx.loc[test_cx['month_joined'] == 'February', 'fully_onboarded'].sum()
]
no_of_observations = [
    test_cx.loc[test_cx['month_joined']=='January', 'fully_onboarded'].count(),
    test_cx.loc[test_cx['month_joined']=='February', 'fully_onboarded'].count()
]
```

In [56]:
```
stat, pval = proportions_ztest(count, no_of_observations)
print(stat)
if pval < 0.01:
    print('reject null hypothesis')
else:
    print('failed to reject null hypothesis')
```

```
12.42781069948263
reject null hypothesis
```

recommendation:

Move to milestone-based rewards (funding or first purchase).

Conduct A/B testing to validate optimal incentive amount and trigger.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: