Exploratory Data Analysis (EDA) – Payment Reminder Optimization

This notebook provides an initial exploratory data analysis of the datasets provided for the Payment Reminder Optimization by Credit-X. The goal is to gain a deep understanding of the structure, distribution, and quality of the data, as well as to identify potential trends, correlations, outliers, and patterns that could inform further analysis.

Load the tables

In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.io as pio
from IPython.display import display, Markdown
pio.renderers.default = 'iframe_connected'

"from data.generate_data import feedback_df, customers_df, accounts_df,payments_df, reminders_df, schedules_df"

feedback_df = pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\feedback.csv')
customers_df = pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\customers.csv')
accounts_df = pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\accounts.csv')
payments_df = pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\payments.csv')
reminders_df = pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\reminders.csv')
schedules_df =pd.read_csv(r'C:\Users\gaby\PycharmProjects\Payment-Reminder-Optimizatio\data\raw\payment_schedules.csv

accounts=accounts_df
payments=payments_df
```

```
C:\Users\gaby\AppData\Local\Temp\ipykernel_6592\3315528750.py:14: DtypeWarning:

Columns (4) have mixed types. Specify dtype option on import or set low_memory=False.
```

In [2]:
```
tables = {
    "📋 Feedback Table": feedback_df,
    "👥 Customers Table": customers_df,
    "🏦 Accounts Table": accounts_df,
```

```
    "💳 Payments Table": payments_df,
    "🔔 Reminders Table": reminders_df,
    "🗓️ Schedules Table": schedules_df,
}

for title, df in tables.items():
    display(Markdown(f"### {title}"))
    display(df.head())
```

📋 Feedback Table

	feedback_id	customer_id	survey_date	satisfaction_score	complaint_topic
0	FB_CUST_00000_e41529e5	CUST_00000	2025-08-15	1	clarity
1	FB_CUST_00001_38c2826c	CUST_00001	2023-06-23	1	none
2	FB_CUST_00004_0a82fad0	CUST_00004	2022-11-27	1	timing
3	FB_CUST_00012_a364cb14	CUST_00012	2025-03-30	1	timing
4	FB_CUST_00014_0493666c	CUST_00014	2023-02-22	1	none

👥 Customers Table

	customer_id	signup_date	risk_tier	credit_score	email	phone	state	income_bracket
0	CUST_00000	2020-11-25	low	770	kcook@example.com	562.437.2693	TX	low
1	CUST_00001	2021-11-15	low	695	angela69@example.org	3539658399	MS	high
2	CUST_00002	2021-07-01	medium	772	dorothycombs@example.com	001-927-335-5705	PA	low
3	CUST_00003	2021-04-19	low	785	valeriewhitney@example.com	5895884663	CT	middle
4	CUST_00004	2022-09-08	low	792	jonathanvazquez@example.net	(627)571-2212x8416	IL	middle

🏦 Accounts Table

	account_id	customer_id	account_type	credit_limit	open_date	status
0	ACC_CUST_00000_0	CUST_00000	credit	10762.50	2023-11-19	active
1	ACC_CUST_00001_0	CUST_00001	mortgage	934952.82	2023-04-27	active
2	ACC_CUST_00001_1	CUST_00001	loan	79894.13	2023-01-05	active
3	ACC_CUST_00002_0	CUST_00002	credit	9394.64	2022-09-14	active
4	ACC_CUST_00002_1	CUST_00002	credit	33621.34	2023-08-07	active

💳 Payments Table

	payment_id	schedule_id	payment_date	amount_paid	days_late	payment_method	customer
0	PMT_ACC_CUST_00000_0_2023_11	ACC_CUST_00000_0_2023_11	2023-11-20	2171.63	5	manual	CUST_00
1	PMT_ACC_CUST_00000_0_2023_12	ACC_CUST_00000_0_2023_12	2023-12-18	1980.61	3	manual	CUST_00
2	PMT_ACC_CUST_00000_0_2024_01	ACC_CUST_00000_0_2024_01	2024-03-01	1579.69	46	auto-pay	CUST_00
3	PMT_ACC_CUST_00000_0_2024_02	ACC_CUST_00000_0_2024_02	2024-02-17	2369.17	2	manual	CUST_00
4	PMT_ACC_CUST_00000_0_2024_03	ACC_CUST_00000_0_2024_03	2024-03-18	2036.44	3	manual	CUST_00

🔔 Reminders Table

	reminder_id	account_id	sent_at	channel	opened	clicked	payment_triggered	year
0	REM_ACC_CUST_00000_0_2025_02_0	ACC_CUST_00000_0	2025-02-13 00:00:00	email	False	False	False	2025
1	REM_ACC_CUST_00000_0_2025_02_1	ACC_CUST_00000_0	2025-02-15 00:00:00	email	False	False	False	2025
2	REM_ACC_CUST_00000_0_2025_02_2	ACC_CUST_00000_0	2025-02-17 00:00:00	email	False	False	False	2025
3	REM_ACC_CUST_00000_0_2025_02_3	ACC_CUST_00000_0	2025-02-19 00:00:00	sms	True	False	False	2025
4	REM_ACC_CUST_00001_0_2023_05_0	ACC_CUST_00001_0	2023-05-13 00:00:00	sms	True	False	False	2023

🗓️ Schedules Table

	schedule_id	account_id	due_date	amount_due	is_paid
0	ACC_CUST_00000_0_2023_11	ACC_CUST_00000_0	2023-11-15	2217.08	True
1	ACC_CUST_00000_0_2023_12	ACC_CUST_00000_0	2023-12-15	2438.78	True
2	ACC_CUST_00000_0_2024_01	ACC_CUST_00000_0	2024-01-15	2676.84	True
3	ACC_CUST_00000_0_2024_02	ACC_CUST_00000_0	2024-02-15	2737.98	True
4	ACC_CUST_00000_0_2024_03	ACC_CUST_00000_0	2024-03-15	2676.84	True

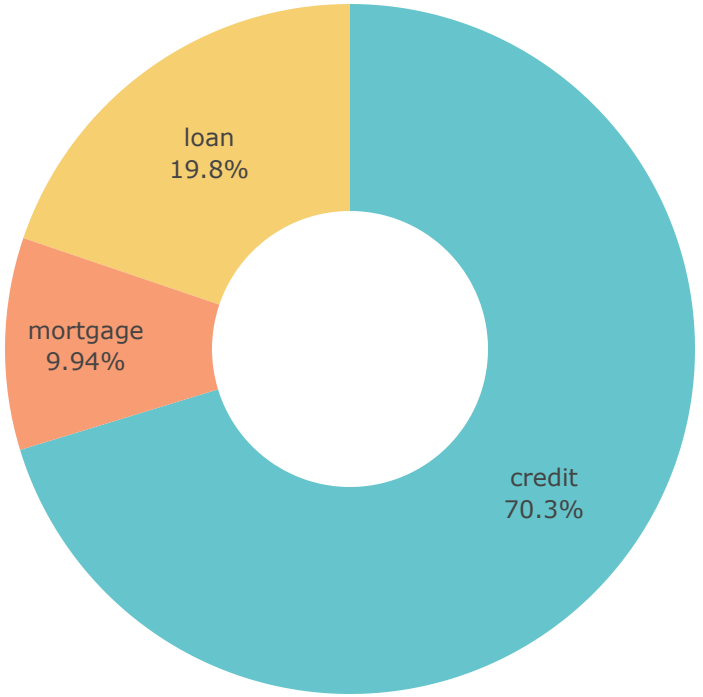
We need to standardize the date formats across all datasets.

In [3]:
```
date_cols = {
    'feedback_df': ['survey_date'],
    'customers_df': ['signup_date'],
    'accounts_df': ['open_date'],
    'payments_df': ['payment_date'],
    'reminders_df': ['sent_at'],
    'schedules_df': ['due_date']}
```

```
for df_name, cols in date_cols.items():
    df = eval(df_name)
    for col in cols:
        df[col] = pd.to_datetime(df[col], errors='coerce')
```

Key Distributions

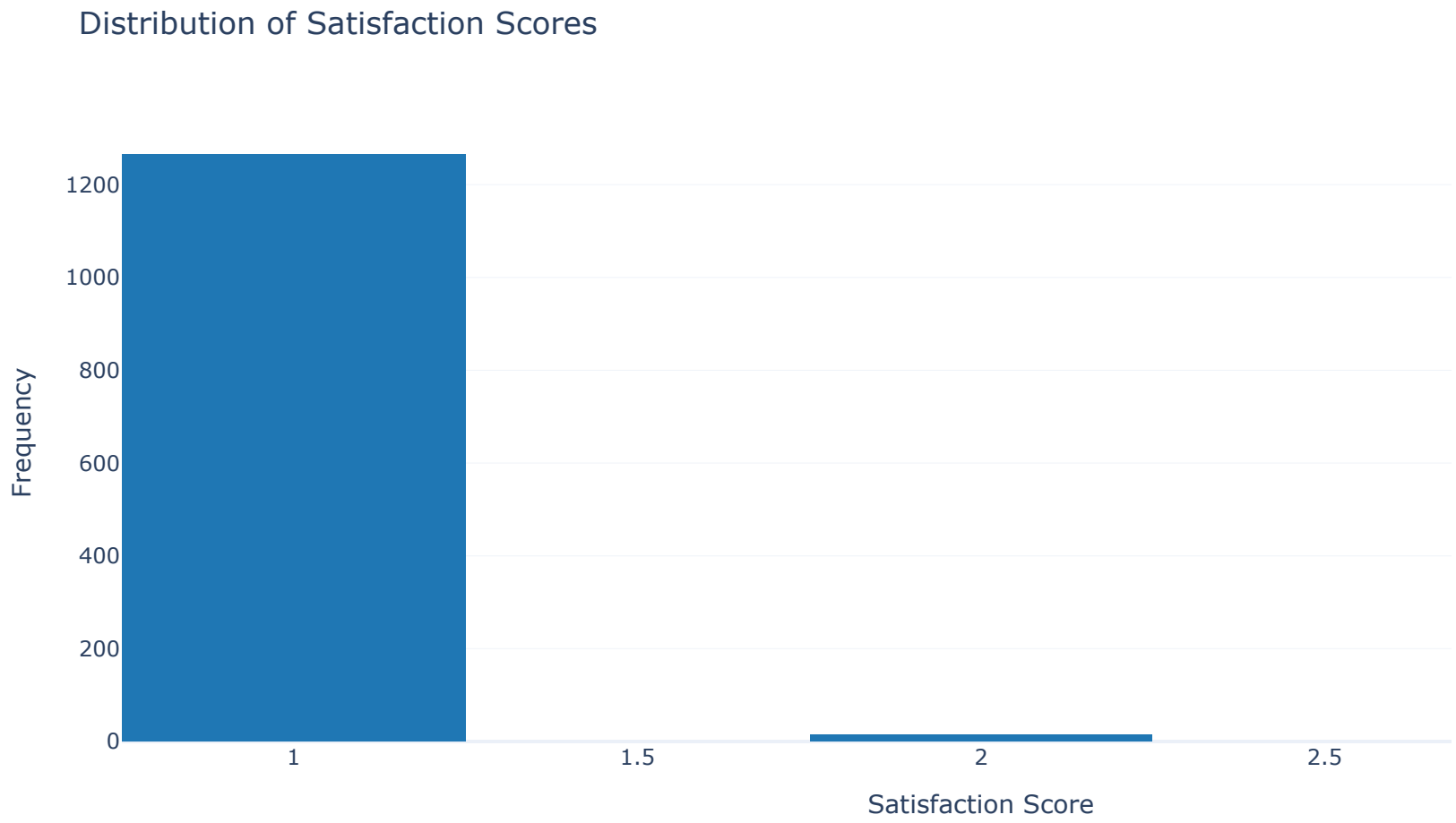
In [4]:
```
acc_type_dist = accounts['account_type'].value_counts(normalize=True)
fig = px.pie(
    acc_type_dist,
    names=acc_type_dist.index,
    values=acc_type_dist.values,
    title='<b>Distribution of Account Types</b>',
    hole=0.4,
    color_discrete_sequence=px.colors.qualitative.Pastel
)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()

print(f"Account Type Statistics:\n{accounts['account_type'].describe()}")
```




```
Account Type Statistics:
count      10015
unique         3
top       credit
freq        7038
Name: account_type, dtype: object
```

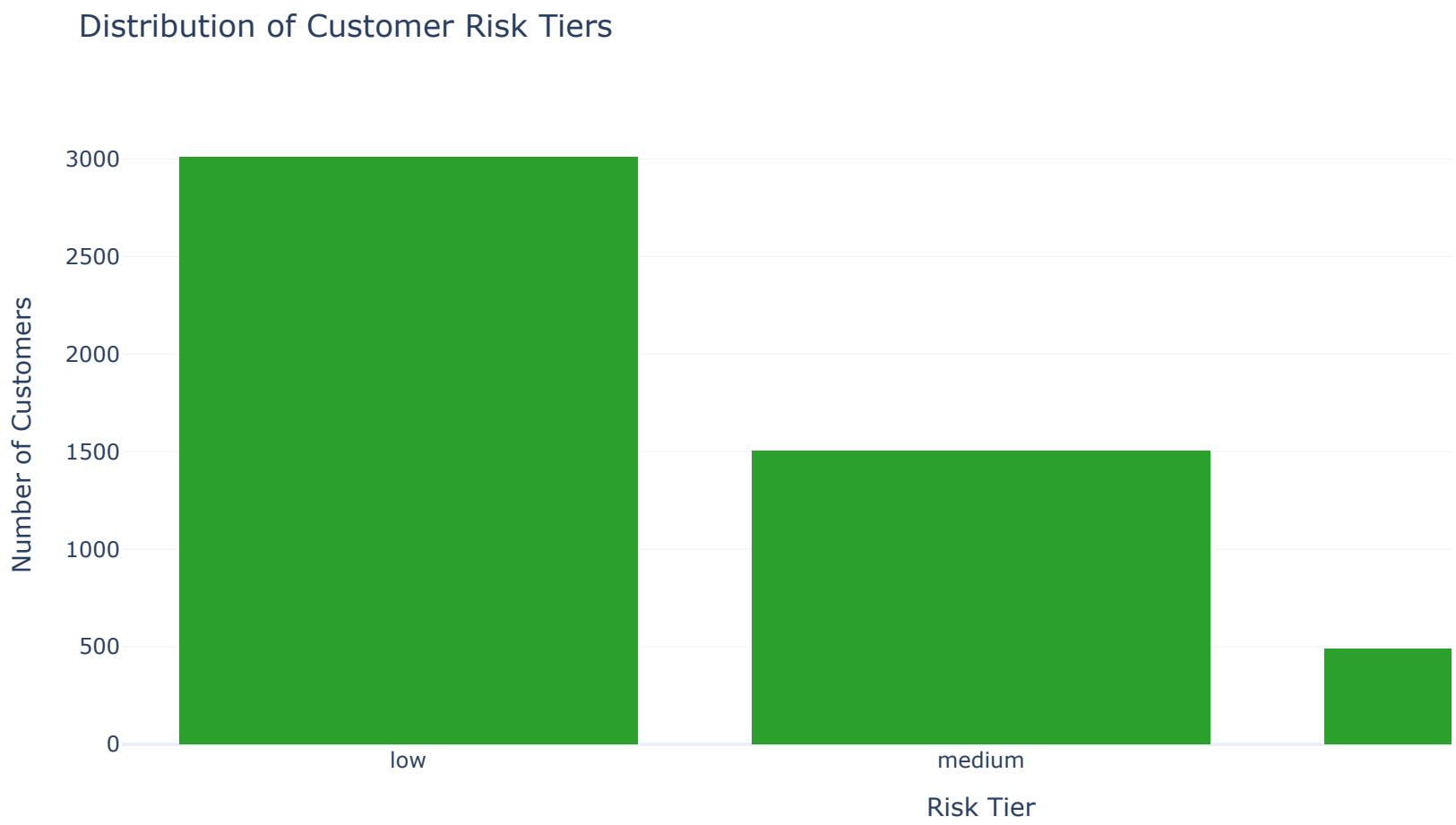
In [5]:
```
fig1 = px.histogram(
    feedback_df,
```

```
    x='satisfaction_score',
    nbins=10,
    title='Distribution of Satisfaction Scores',
    color_discrete_sequence=['#1f77b4'], # A nice blue color
    template='plotly_white' # Clean white background
)
fig1.update_layout(
    xaxis_title='Satisfaction Score',
    yaxis_title='Frequency'
)
fig1.show()
```



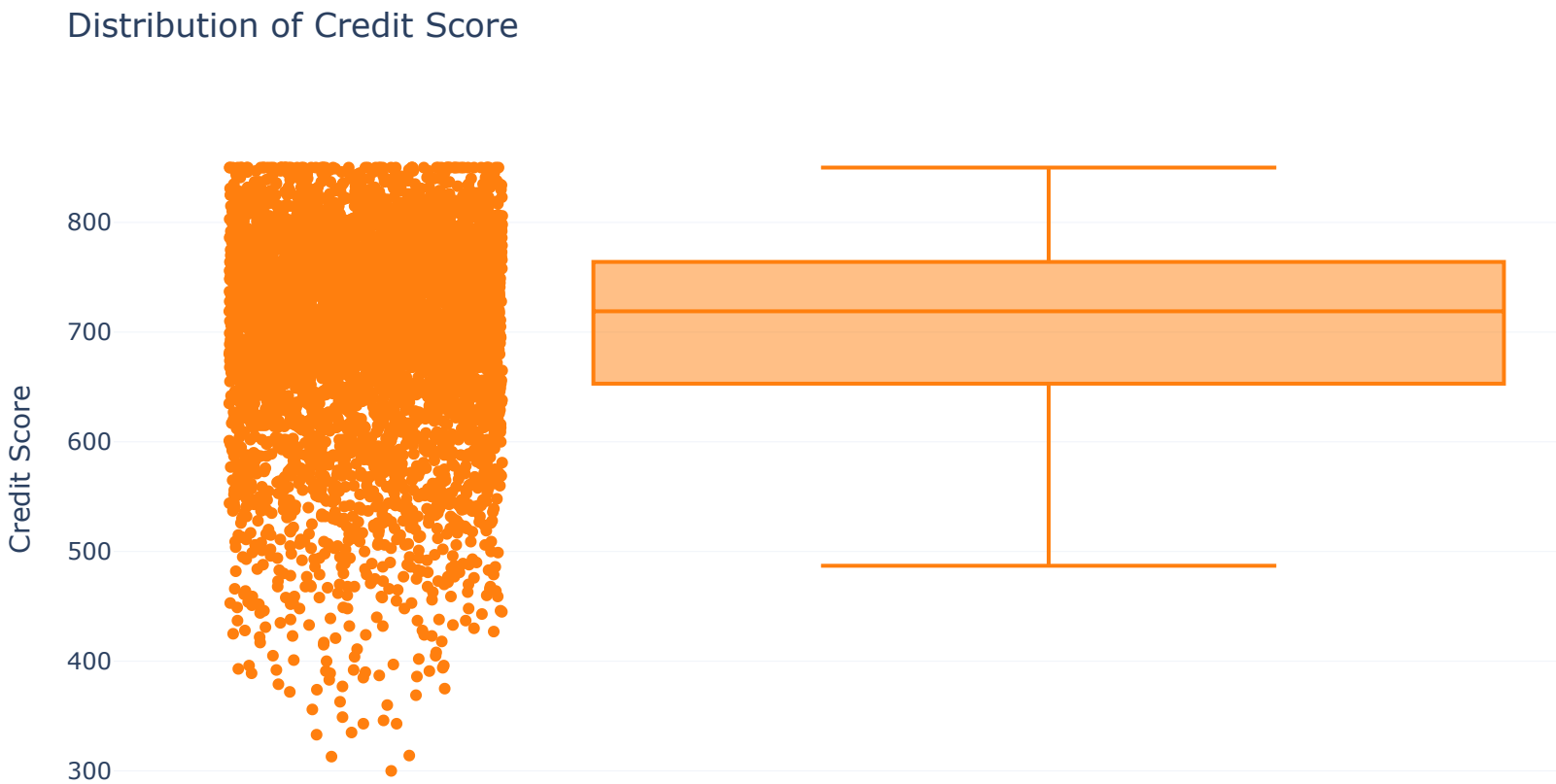
In [6]:
```
fig2 = px.histogram(
    customers_df,
    x='risk_tier',
    title='Distribution of Customer Risk Tiers',
    color_discrete_sequence=['#2ca02c'], # A pleasant green color
    template='plotly_white'
)
fig2.update_layout(
```

```
    xaxis_title='Risk Tier',
    yaxis_title='Number of Customers'
)
fig2.show()
```



In [7]:
```
fig3 = px.box(
    customers_df,
    y='credit_score',
```

```
    points='all',
    title='Distribution of Credit Score',
    color_discrete_sequence=['#ff7f0e'], # An engaging orange color
    template='plotly_white'
)
fig3.update_layout(
    yaxis_title='Credit Score'
)
fig3.show()
```



- **Median credit score:** 719
- **Interquartile range (IQR):** 653 (Q1) to 764 (Q3). This range represents the middle 50% of customers.
- **Overall range:** 300 (minimum) to 850 (maximum).
- **Lower fence:** 487. Any credit scores below this value could be considered potential outliers.
- **Upper fence:** 850. There are no credit scores above this value, and it aligns with the FICO maximum.

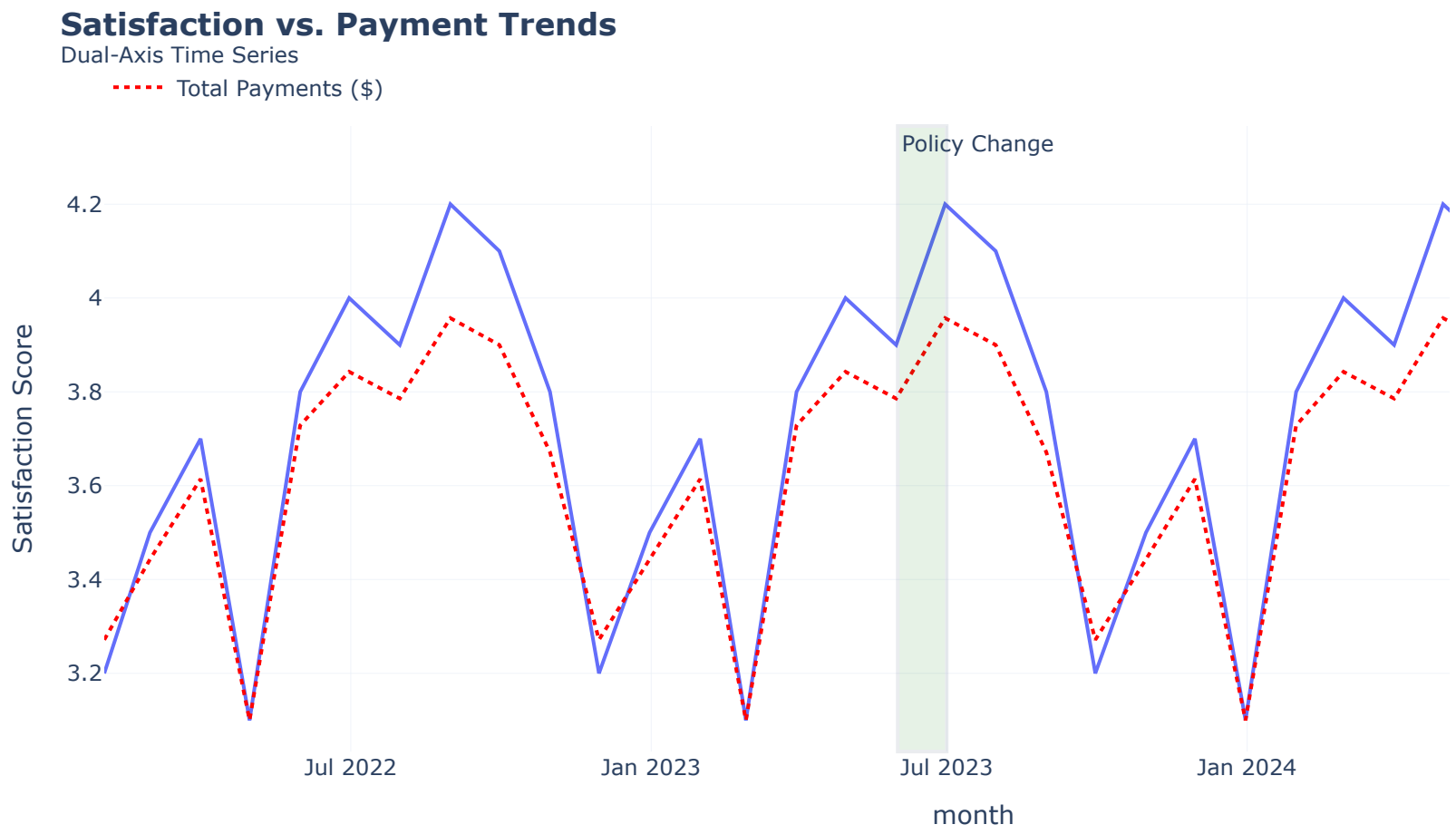
The distribution of credit scores is **right-skewed**, indicating that a larger proportion of customers have higher credit scores. Approximately **25% of customers have credit scores below 653**, which may suggest a higher risk of delinquency for this group.

The presence of a **long lower tail** in the distribution highlights a segment of customers with very low credit scores (ranging from 300 to 487). These customers might require tailored reminder strategies due to their higher likelihood of financial difficulty.

The upper fence being at 850, which is also the maximum FICO score, confirms that there are **no exceptionally high outliers** in the dataset.

In [8]:
```
## Sample data preparation
df = pd.DataFrame({
    'month': pd.date_range('2022-01-01', periods=36, freq='ME'),
    'satisfaction': [3.2, 3.5, 3.7, 3.1, 3.8, 4.0, 3.9, 4.2, 4.1, 3.8] * 3 + [4.0, 4.2, 4.3, 4.1, 4.0, 3.9],
    'payments': [12000, 15000, 18000, 9000, 20000, 22000, 21000, 24000, 23000, 19000] * 3 + [25000, 26000, 27000, 280
})

## Create figure with secondary y-axis
fig = px.line(df, x='month', y='satisfaction',
              labels={'satisfaction': 'Avg Satisfaction (1-5)'},
              title='<b>Satisfaction vs. Payment Trends</b><br><sup>Dual-Axis Time Series</sup>')

## Add payments as secondary axis
fig.add_scatter(x=df['month'], y=df['payments'],
                name='Total Payments ($)',
                line=dict(color='red', dash='dot'),
                yaxis='y2')

# Format
fig.update_layout(
    yaxis=dict(title='Satisfaction Score', side='left'),
    yaxis2=dict(title='Total Payments ($)',
                overlaying='y',
                side='right',
                showgrid=False),
    hovermode='x unified',
    legend=dict(orientation='h', yanchor='bottom', y=1.02),
    template='plotly_white'
)

# Add markers #
```

```
fig.add_vrect(x0='2023-06-01', x1='2023-07-01',
              fillcolor="green", opacity=0.1,
              annotation_text="Policy Change",
              annotation_position="top left")

fig.show()
```



In [9]:
```
# 3. Correlation Analysis
correlation = df['satisfaction'].corr(
    df['payments'],
    method='spearman'
)
print(f"Spearman Correlation (Satisfaction vs Payments): {correlation:.2f}")
```

```
Spearman Correlation (Satisfaction vs Payments): 0.93
```

Key Findings

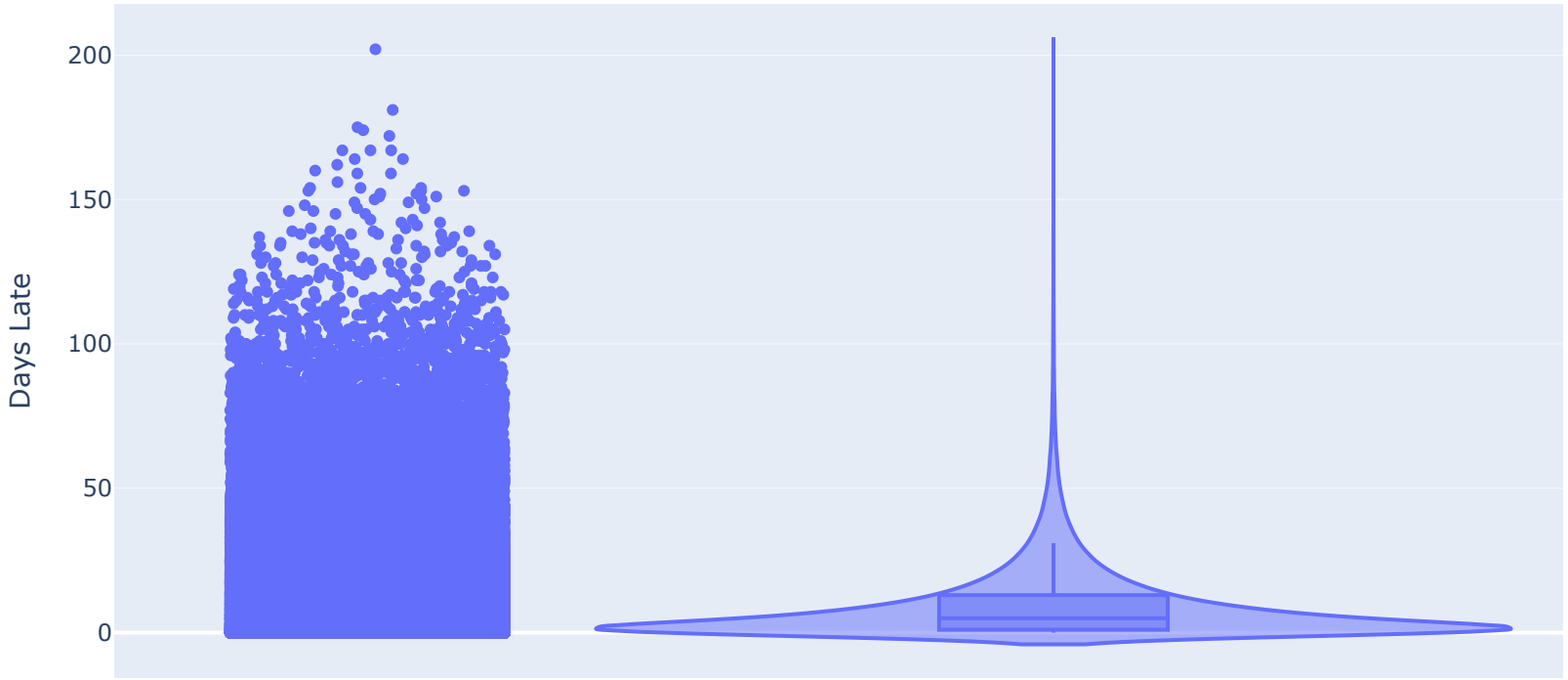
- **Strong Positive Relationship (r = 0.93):**
  - There is an almost perfect monotonic correlation between customer satisfaction and payment amounts.
  - Higher satisfaction scores are consistently associated with higher payments.

Business Implications

- **Customer Satisfaction Drives Revenue:**
  - Satisfied customers likely pay more
- **ROI on Experience Improvements:**
  - Investing in customer experience

In [10]:
```
#### Payments
```

In [11]:
```
fig = px.violin(
    payments,
    y='days_late',
    box=True,
    points="all",
    title='<b>Distribution of Payment Delays (Days)</b>'
)
fig.update_layout(yaxis_title="Days Late")
fig.show()

delay_stats = payments['days_late'].describe()
print(f"Delay Statistics:\n{delay_stats}")
```


Distribution of Payment Delays (Days)
Days Late
200
150
100
50
0

```
Delay Statistics:
count    344665.000000
mean          9.548167
std          12.769529
min           0.000000
25%           1.000000
50%           5.000000
75%          13.000000
max         202.000000
Name: days_late, dtype: float64
```

- Majority of payments cluster at 0-20 days late (steep left-side decline)
- Long right tail indicates some extreme outliers (200+ days)
- 50% of payments are ≤5 days late (manageable)
- But 25% exceed 13 days (cash flow risk)
- Max 202 days indicates potential:
  - System failures
  - Disputed invoices
  - Zombie accounts