Import libraries

In [1]:
```python
import pandas as pd
```

In [2]:
```python
trxn = pd.read_csv("Retail_Data_Transactions.csv")
response = pd.read_csv("Retail_Data_Response.csv")
```

In [3]:
```python
trxn.head(5)
```

Out[3]:

	customer_id	trans_date	tran_amount
0	CS5295	2013-02-11	35
1	CS4768	2015-03-15	39
2	CS2122	2013-02-26	52
3	CS1217	2011-11-16	99
4	CS1850	2013-11-20	78

In [4]:
```python
response.head(5)
```

Out[4]:

	customer_id	response
0	CS1112	0
1	CS1113	0
2	CS1114	1
3	CS1115	1
4	CS1116	1

Merging the tables

In [5]:
```python
df = trxn.merge(response,on="customer_id",how="left")
```

In [6]:
```python
df.head(3)
```

Out[6]:

	customer_id	trans_date	tran_amount	response
0	CS5295	2013-02-11	35	1.0
1	CS4768	2015-03-15	39	1.0
2	CS2122	2013-02-26	52	0.0

Check nulls values

In [7]:
```python
df.isnull().sum()
```

Out[7]:
```
customer_id     0
trans_date      0
tran_amount     0
response       31
dtype: int64
```

Check Data types

In [8]:
```python
df.dtypes
```

Out[8]:
```
customer_id     object
trans_date      object
tran_amount      int64
response       float64
dtype: object
```

In [9]:
```python
df["trans_date"] = pd.to_datetime(df["trans_date"])
```

In [10]:
```python
df["response"] = df["response"].astype("object")
```

In [11]:
```python
df.dtypes
```

Out[11]:
```
customer_id            object
trans_date     datetime64[ns]
tran_amount             int64
response               object
dtype: object
```

In [12]:
```python
df["response"] = df["response"].fillna(0).astype("int64")
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6472\1415641469.py:1: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will change in a future version. Call result.infer_objects(copy=False) instead. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df["response"] = df["response"].fillna(0).astype("int64")
```

In [13]:
```python
df.dtypes
```

Out[13]:
```
customer_id            object
trans_date     datetime64[ns]
tran_amount             int64
response                int64
dtype: object
```

Checking Outliers using z_score

In [14]:
```python
import numpy as np
from scipy import stats
```

In [15]:
```python
z_score = np.abs(stats.zscore(df["tran_amount"]))
threshold = 3
outliers = z_score > threshold
print(outliers)
```

```
[False False False ... False False False]
```

In [16]:
```python
z_score = np.abs(stats.zscore(df["response"]))
threshold = 3
outliers = z_score > threshold
print(outliers)
```

```
[False False False ... False False False]
```

Create new columns

In [17]:
```python
df["month"] = df["trans_date"].dt.month_name()
df["year"] = df["trans_date"].dt.year
```

In [18]:
```python
df.head(3)
```

Out[18]:

	customer_id	trans_date	tran_amount	response	month	year
0	CS5295	2013-02-11	35	1	February	2013
1	CS4768	2015-03-15	39	1	March	2015
2	CS2122	2013-02-26	52	0	February	2013

In [19]:
```python
import plotly.express as px
```

Which three months have the highest amount of transactions

In [20]:
```python
month_sales = df.groupby("month")["tran_amount"].sum().sort_values(ascending=False).reset_index()
```

In [21]:
```python
month_sales.head(3)
```

Out[21]:

	month	tran_amount
0	August	726921
1	October	725320
2	January	724107

In [22]:
```python
fig = px.bar(
    month_sales,
    x="month",
    y="tran_amount",
    title="Highest Month Transaction",
    color="tran_amount",
    color_continuous_scale="Cividis",
)
fig.update_layout(width=700,height=400)
fig.show()
```

Top 5 customer have highest co of orders

In [23]:
```python
customer_count = df["customer_id"].value_counts().reset_index()
customer_count.head(5)
```

Out[23]:

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS5109	35
4	CS2620	35

In [24]:
```python
fig = px.bar(
    customer_count.head(5),
    x="customer_id",
    y="count",
    title="Top 5 Customers by Order Count",
    color="count",
    color_continuous_scale="Cividis",
)
fig.update_layout(width=500,height=400)
fig.show()
```

Top 10 customer having highest sales

In [25]:
```python
customer_sales = df.groupby("customer_id")["tran_amount"].sum().reset_index().sort_values(by="tran_amount",ascending=False)
customer_sales.head(10)
```

Out[25]:

	customer_id	tran_amount
3312	CS4424	2933
3208	CS4320	2647
4640	CS5752	2612
3548	CS4660	2527
2687	CS3799	2513
3997	CS5109	2506
2962	CS4074	2462
2693	CS3805	2453
3496	CS4608	2449
4443	CS5555	2439

In [26]:
```python
fig = px.bar(
    customer_sales.head(10),
    x="customer_id",
    y="tran_amount",
    title="Highest Customers Sales",
    color="tran_amount",
    color_continuous_scale ="Cividis"
)
fig.update_layout(width=700,height=400)
fig.show()
```

Which year has the highest number of transactions

In [27]:
```python
yearly_sales = df.groupby("year")["tran_amount"].sum().reset_index().sort_values(by="tran_amount",ascending=False)
```

In [28]:
```python
yearly_sales
```

Out[28]:

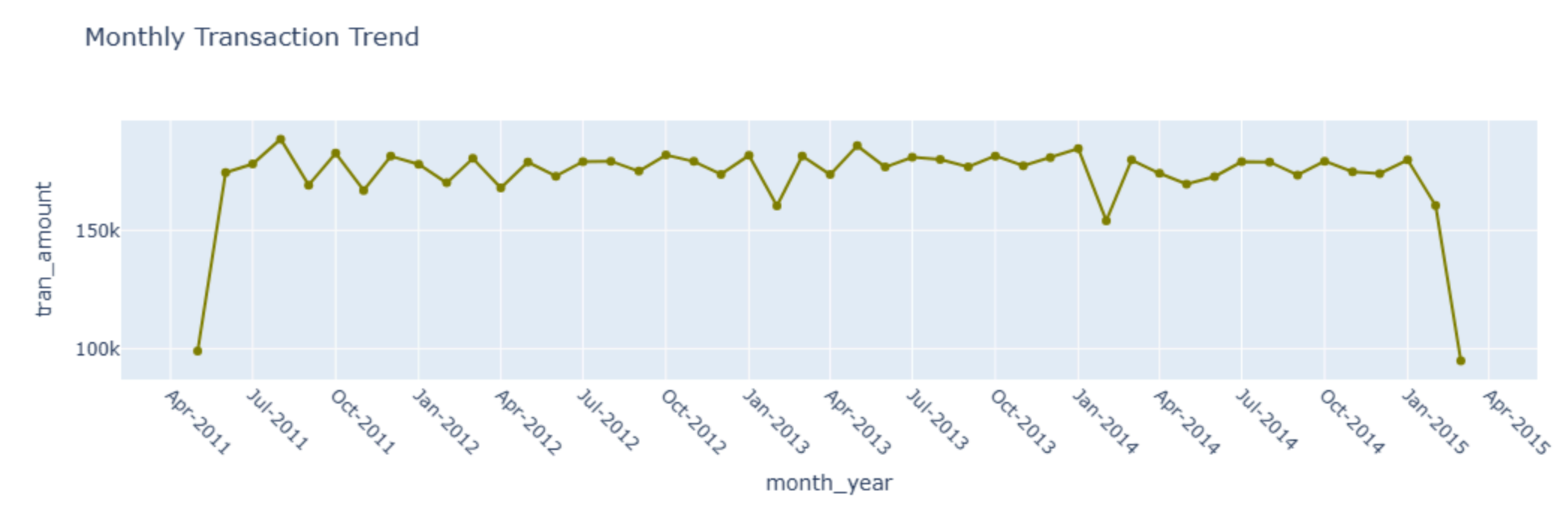
	year	tran_amount
2	2013	2137368
1	2012	2116599
3	2014	2094508
0	2011	1340339
4	2015	435175

In [29]:
```python
fig = px.bar(
    yearly_sales,
    x="year",
    y="tran_amount",
    title="Highest Year Transaction",
    color="tran_amount",
    color_continuous_scale="Cividis",
)
fig.update_layout(width=700,height=400)
fig.show()
```

Advanced Analytics

Times Series Analysis

In [30]:
```python
df["month_year"] = df["trans_date"].dt.to_period("M").dt.to_timestamp()

month_year_sales = (
    df.groupby("month_year")["tran_amount"]
    .sum()
    .reset_index()
    .sort_values("month_year")
)


fig = px.line(
    month_year_sales,
    x="month_year",
    y="tran_amount",
    title="Monthly Transaction Trend",
    color_discrete_sequence=["olive"] ,
    markers=True
)

fig.update_xaxes(
    dtick="M3",          # tick every 3 months
    tickformat="%b-%Y",
     tickangle=45
)

fig.show()
```



In [32]:
```python
df.head(3)
```

Out[32]:

	customer_id	trans_date	tran_amount	response	month	year	month_year
0	CS5295	2013-02-11	35	1	February	2013	2013-02-01
1	CS4768	2015-03-15	39	1	March	2015	2015-03-01
2	CS2122	2013-02-26	52	0	February	2013	2013-02-01

Cohort Segmentation

In [35]:
```python
# Recency - most recent orders
recency = df.groupby("customer_id")["trans_date"].max()
# frequency - count of orders
frequency = df.groupby("customer_id")["trans_date"].count()
# monetory - sum of amount
monetory = df.groupby("customer_id")["tran_amount"].sum()

# combine all into dataframe
rfm = pd.DataFrame({"recency":recency,
                  "frequency":frequency,
                  "monetory":monetory})
```

In [36]:
```python
rfm
```

Out[36]:

customer_id	recency	frequency	monetory
CS1112	2015-01-14	15	1012
CS1113	2015-02-09	20	1490
CS1114	2015-02-12	19	1432
CS1115	2015-03-05	22	1659
CS1116	2014-08-25	13	857
...	...	...	...
CS8996	2014-12-09	13	582
CS8997	2014-06-28	14	543
CS8998	2014-12-22	13	624
CS8999	2014-07-02	12	383
CS9000	2015-02-28	13	533

6889 rows × 3 columns

In [59]:
```python
def seg_customer(row):
    if row["recency"].year>=2015 and row["frequency"]>=15 and row["monetory"]>1000:
        return "PO"
    elif(row["recency"].year>=2014) and (row["recency"].year<2015) and (10<=row["frequency"]<15) and (500<=row["monetory"]<1000):
        return "P1"
    else:
        return "P2"
```

In [63]:
```python
rfm["segment"] = rfm.apply(seg_customer,axis=1)
```

In [64]:
```python
rfm
```

Out[64]:

customer_id	recency	frequency	monetory	segment
CS1112	2015-01-14	15	1012	PO
CS1113	2015-02-09	20	1490	PO
CS1114	2015-02-12	19	1432	PO
CS1115	2015-03-05	22	1659	PO
CS1116	2014-08-25	13	857	P1
...	...	...	...	...
CS8996	2014-12-09	13	582	P1
CS8997	2014-06-28	14	543	P1
CS8998	2014-12-22	13	624	P1
CS8999	2014-07-02	12	383	P2
CS9000	2015-02-28	13	533	P2

6889 rows × 4 columns

Churn Analysis

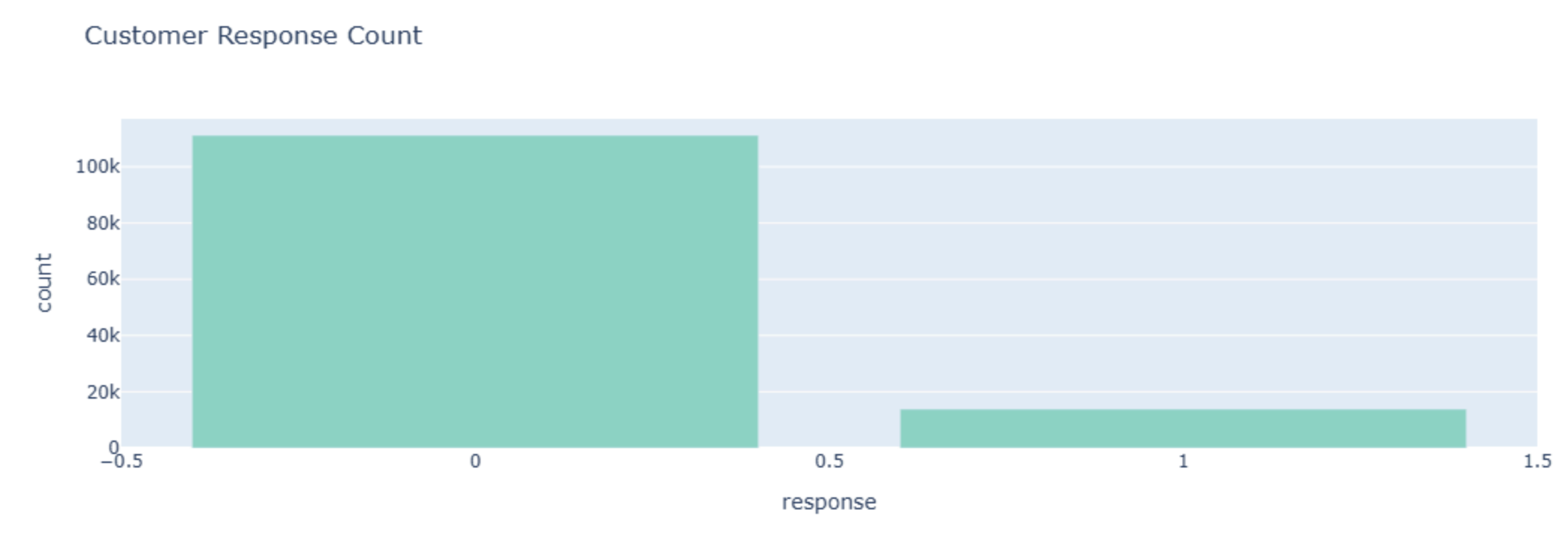
In [70]:
```python
# Count the no of churned and active customers

churn_count = df["response"].value_counts().reset_index()
```

In [71]:
```python
churn_count
```

Out[71]:

	response	count
0	0	111158
1	1	13842

In [81]:
```python
fig = px.bar(churn_count,
           x= "response",
           y="count",
           title= "Customer Response Count",
           color_discrete_sequence=px.colors.qualitative.Set3)
fig.update_layout(width = 400, height = 400)
fig.show()
```



Analyze Top Customers

In [90]:
```python
top5_customer = monetory.sort_values(ascending=False).head(5).index
top5_customer_df = df[df["customer_id"].isin(top5_customer)]

top5customer_sales = top5_customer_df.groupby(["customer_id","month_year"])["tran_amount"].sum().unstack(level=0)
```

In [91]:
```python
fig = px.line(
    top5customer_sales,
    title="Top 5 Customers Monthly Sales Trend"
)
fig.update_layout(width=900, height=450)
fig.show()
```


Top 5 Customers Monthly Sales Trend
sales
500
250
200
150
100
50
Jul 2011
Jan 2012
Jul 2012
Jan 2013
Jul 2013
Jan 2014
Jul 2014
Jan 2015
month_year
customer_id
C13799
C14320
C14424
C14660
C16752

In [ ]: