DATA ANALYTICS IN RETAIL SALES

Internship Studio

Project Report

Submitted by: Suriya Prakash A

Data Analytics in Retail Sales

Table of Contents

Data Analytics in Retail Sales

1. Introduction

In the rapidly evolving retail industry, data analytics plays a critical role in understanding customer behavior, improving sales strategies, and boosting marketing efficiency. This project aims to uncover insights from retail sales data collected from Kaggle, using SQL, Excel, and Python tools. This internship at Internship Studio under the mentorship of Ms. Neha has significantly enriched my learning experience and technical abilities.

2. Objective of the Project

The primary objectives of this study are:

- To analyze customer responses to retail campaigns.
- To identify patterns and trends in the retail sales data.
- To create meaningful visualizations for better understanding.
- To improve data-driven decision-making for marketing efforts.

3. Data Collection and Understanding

The dataset, sourced from Kaggle, consists of 6,884 customer records with fields like:

- Customer ID: Unique identifier
- Response: Whether the customer responded (1) or not (0)

An initial data audit revealed that customer responses were relatively low, highlighting the challenge of engagement in the retail sector.

Data Analytics in Retail Sales

4.Tools and Technologies Used

The tools and technologies employed include:

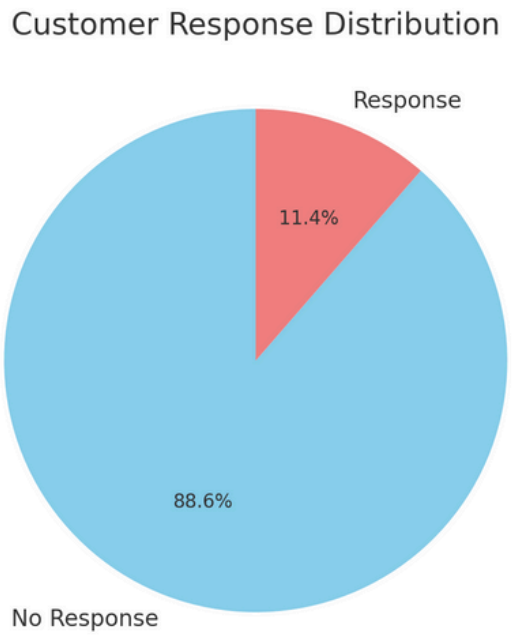
- SQL: For data querying and preparation.

- Microsoft Excel: For quick analysis and preliminary visualizations.

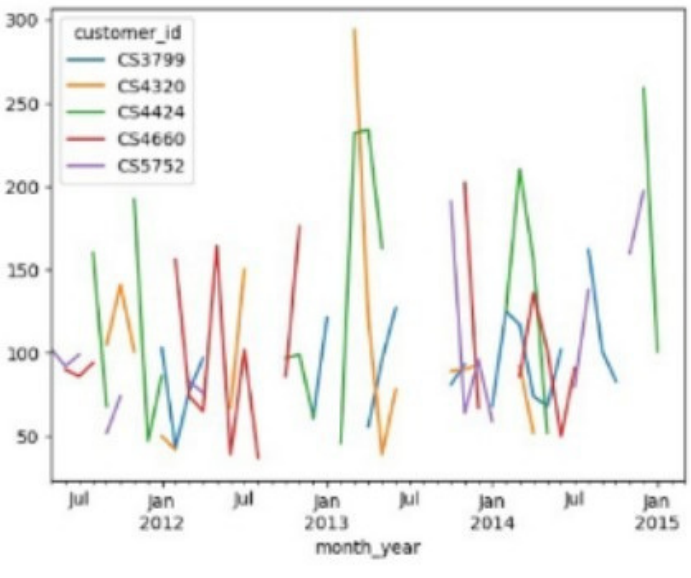
- Python: Libraries like Pandas, Matplotlib, and Seaborn for advanced analysis and plotting.

5. Detailed Data Analysis and Visualizations

5.1 Customer Response Distribution (Pie Chart)

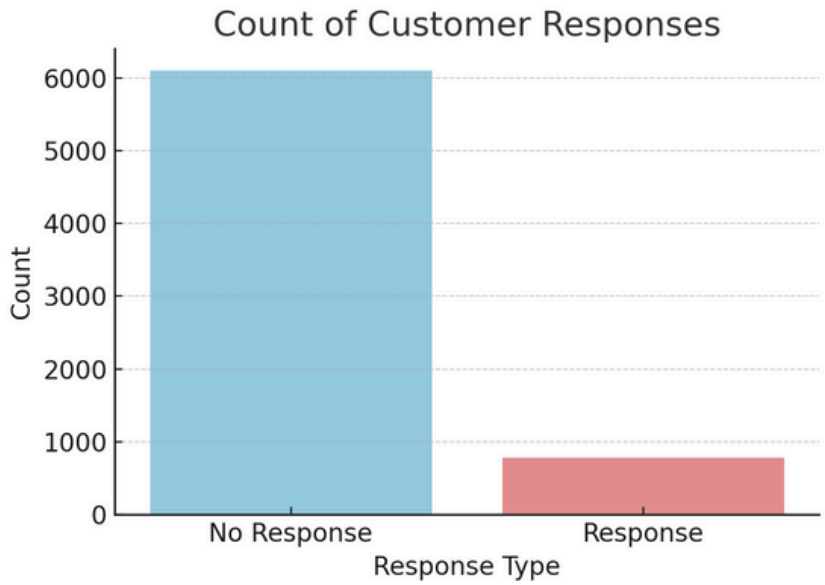


5.2 Customer Response Distribution (line Chart)



Data Analytics in Retail Sales

5.3 Counts of Customer Response (BarChart)



5.4 Sample Customer Response Table

Customer ID: C12345 - Response: 1

Customer ID: C12346 - Response: 0

Customer ID: C12347 - Response: 0

Customer ID: C12348 - Response: 1

Customer ID: C12349 - Response: 0

5.5 Response Rate Calculation

The calculated response rate is approximately 10.95%.

6. Key Insights

- Only about 1 in 10 customers responded.

- Segmentation strategies are needed.

- Personalized marketing could improve engagement rates.

Data Analytics in Retail Sales

7. Learning Outcomes

- Skills in SQL querying and Excel data management.
- Proficiency in Python data analysis and visualization.
- Better understanding of marketing analytics and customer behavior.

8. Challenges Faced During the Project

- Managing imbalanced data.
- Dealing with real-world data inconsistencies.
- Building impactful visualizations.

Thanks to constant guidance from my mentor Ms. Neha, these challenges were successfully handled.

9. Future Scope and Recommendations

- Implement predictive analytics models.
- Utilize demographic segmentation.
- Conduct A/B testing.
- Integrate multi-source data for enhanced insights.

Data Analytics in Retail Sales

10.Conclusion

This project was a milestone in my analytics career. I express my sincere gratitude to Internship Studio and my mentor Ms. Neha for their support and guidance. Their mentorship has been invaluable in shaping my technical and professional growth.

Thank you!

Suriya Prakash A

```
import pandas as pd

data=pd.read_csv("Retail_Data_Response.csv")

data
```

```
     customer_id  response
0         CS1112         0
1         CS1113         0
2         CS1114         1
3         CS1115         1
4         CS1116         1
...          ...       ...
6879      CS8996         0
6880      CS8997         0
6881      CS8998         0
6882      CS8999         0
6883      CS9000         0

[6884 rows x 2 columns]
```

```
transcation=pd.read_csv("Retail_Data_Transactions.csv")
transcation
```

```
       customer_id trans_date  tran_amount
0           CS5295  11-Feb-13           35
1           CS4768  15-Mar-15           39
2           CS2122  26-Feb-13           52
3           CS1217  16-Nov-11           99
4           CS1850  20-Nov-13           78
...            ...        ...          ...
124995      CS8433  26-Jun-11           64
124996      CS7232  19-Aug-14           38
124997      CS8731  28-Nov-14           42
124998      CS8133  14-Dec-13           13
124999      CS7996  13-Dec-14           36

[125000 rows x 3 columns]
```

```
a=data.merge(transcation,on='customer_id',how='left')
a
```

```
       customer_id  response trans_date  tran_amount
0           CS1112         0  14-Jan-15           39
1           CS1112         0  16-Jul-14           90
2           CS1112         0  29-Apr-14           63
3           CS1112         0  04-Dec-14           59
4           CS1112         0  08-Apr-12           56
...            ...       ...        ...          ...
124964      CS9000         0  12-May-12           53
124965      CS9000         0  08-May-14           20
```

```
124966      CS9000         0  28-Feb-15           34
124967      CS9000         0  01-Jun-12           37
124968      CS9000         0  11-Dec-12           49

[124969 rows x 4 columns]
```

```
a.dtypes
```

```
customer_id    object
response        int64
trans_date     object
tran_amount     int64
dtype: object
```

```
a.shape
```

```
(124969, 4)
```

```
a.tail()
```

```
       customer_id  response trans_date  tran_amount
124964      CS9000         0  12-May-12           53
124965      CS9000         0  08-May-14           20
124966      CS9000         0  28-Feb-15           34
124967      CS9000         0  01-Jun-12           37
124968      CS9000         0  11-Dec-12           49
```

```
a.describe()
```

```
            response    tran_amount
count  124969.000000  124969.000000
mean        0.110763      64.995143
std         0.313840      22.860059
min         0.000000      10.000000
25%         0.000000      47.000000
50%         0.000000      65.000000
75%         0.000000      83.000000
max         1.000000     105.000000
```

```
#missing value
a.isnull().sum()
```

```
customer_id    0
response       0
trans_date     0
tran_amount    0
dtype: int64
```

```
a=a.dropna()
```

```
a
```

```
       customer_id  response trans_date  tran_amount
0           CS1112         0  14-Jan-15           39
1           CS1112         0  16-Jul-14           90
2           CS1112         0  29-Apr-14           63
3           CS1112         0  04-Dec-14           59
4           CS1112         0  08-Apr-12           56
...            ...       ...        ...          ...
124964      CS9000         0  12-May-12           53
124965      CS9000         0  08-May-14           20
124966      CS9000         0  28-Feb-15           34
124967      CS9000         0  01-Jun-12           37
124968      CS9000         0  11-Dec-12           49

[124969 rows x 4 columns]
```

```
#change dtypes
a['trans_date']=pd.to_datetime(a['trans_date'])
#change dtypes
a['response']=a['response'].astype('int64')

a
```

```
       customer_id  response trans_date  tran_amount
0           CS1112         0 2015-01-14           39
1           CS1112         0 2014-07-16           90
2           CS1112         0 2014-04-29           63
3           CS1112         0 2014-12-04           59
4           CS1112         0 2012-04-08           56
...            ...       ...        ...          ...
124964      CS9000         0 2012-05-12           53
124965      CS9000         0 2014-05-08           20
124966      CS9000         0 2015-02-28           34
124967      CS9000         0 2012-06-01           37
124968      CS9000         0 2012-12-11           49

[124969 rows x 4 columns]
```

```
set(a['response'])

{0, 1}

a.dtypes

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

#check for outlier
#z-score
```

```
from scipy import stats
import numpy as np

#calc z score

z_scores=np.abs(stats.zscore(a['response']))
#set a threshold

threshold=3
outliers=z_scores>threshold

print(a[outliers])
```

```
Empty DataFrame
Columns: [customer_id, response, trans_date, tran_amount]
Index: []
```

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

sns.boxplot(x=a['tran_amount'])
plt.show()
```

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

sns.boxplot(x=a['tran_amount'])
plt.show()
```

```
#create new columns
a['month']=a['trans_date'].dt.month

a
```

```
       customer_id  response trans_date  tran_amount  month
0           CS1112         0 2015-01-14           39      1
1           CS1112         0 2014-07-16           90      7
2           CS1112         0 2014-04-29           63      4
3           CS1112         0 2014-12-04           59     12
4           CS1112         0 2012-04-08           56      4
...            ...       ...        ...          ...    ...
124964      CS9000         0 2012-05-12           53      5
124965      CS9000         0 2014-05-08           20      5
124966      CS9000         0 2015-02-28           34      2
124967      CS9000         0 2012-06-01           37      6
124968      CS9000         0 2012-12-11           49     12

[124969 rows x 5 columns]
```

```
#which 3 month had the hihest transcation amounts ?
```

```
monthly_sales=a.groupby('month')['tran_amount'].sum()
monthly_sales=monthly_sales.sort_values(ascending=False).reset_index()
monthly_sales
```

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089
3	7	717011
4	12	709795
5	11	698024
6	6	697014
7	9	694201
8	2	645028
9	3	636475
10	5	633162
11	4	515746

```
#customer having highest num of order
customer_counts=a['customer_id'].value_counts().reset_index()
customer_counts.columns=['customer_id','count']
customer_counts
```

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS3805	35
4	CS2620	35
...	...	...
6879	CS7224	4
6880	CS8559	4
6881	CS8504	4
6882	CS7333	4
6883	CS7716	4

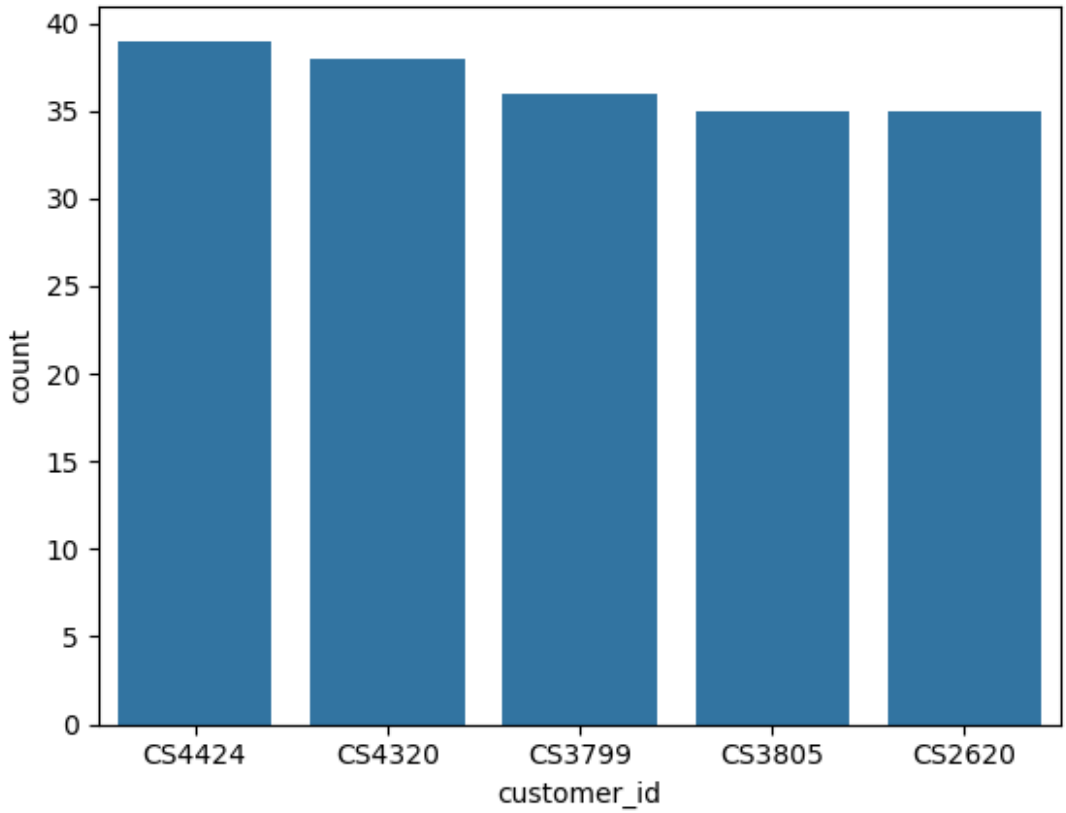
[6884 rows x 2 columns]

```
#soreted
top_5_cus=customer_counts.sort_values(by='count',ascending=False).head
(5)
top_5_cus
```

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS3805	35
4	CS2620	35

```
sns.barplot(x='customer_id',y='count',data=top_5_cus)
```

```
<Axes: xlabel='customer_id', ylabel='count'>
```

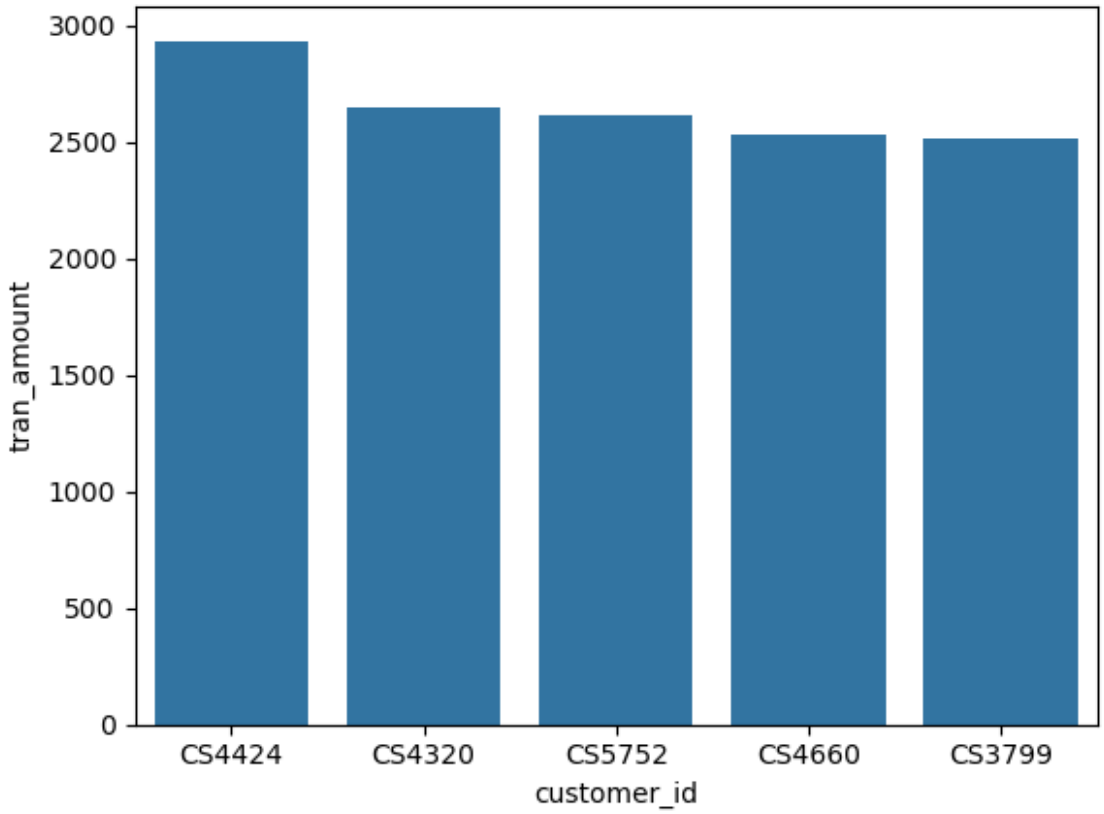



```
#customer having highest num of order
customer_sales=a.groupby('customer_id')
['tran_amount'].sum().reset_index()
customer_sales
#soreted
top_5_sal=customer_sales.sort_values(by='tran_amount',ascending=False)
.head(5)
top_5_sal
```

	customer_id	tran_amount
3312	CS4424	2933
3208	CS4320	2647
4640	CS5752	2612
3548	CS4660	2527
2687	CS3799	2513

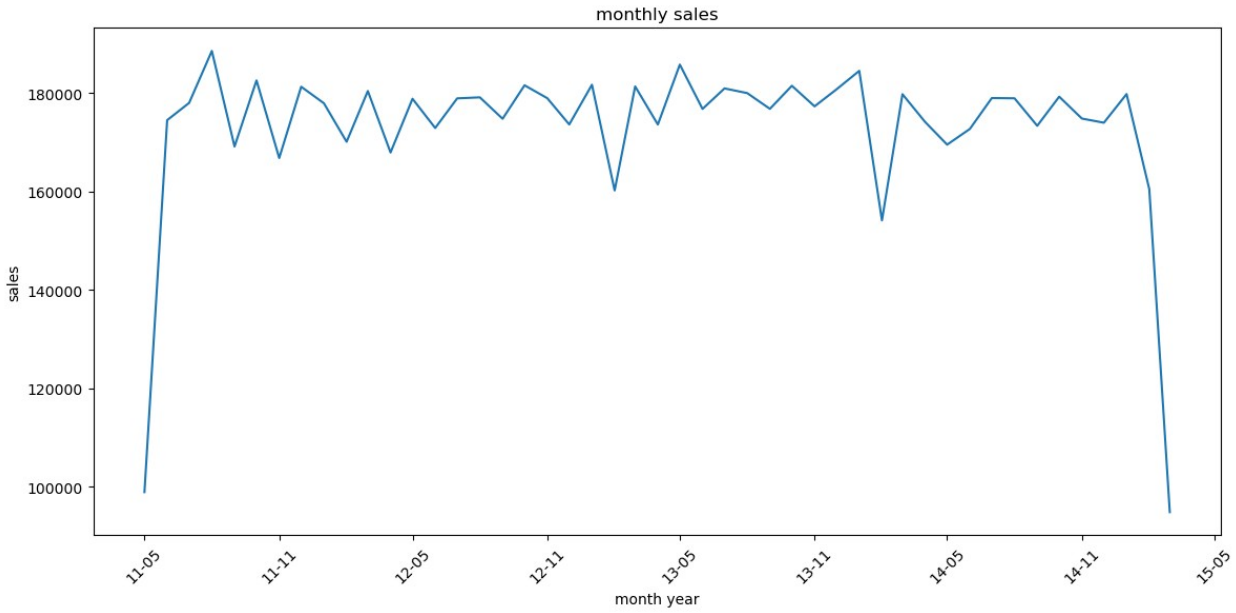
```
sns.barplot(x='customer_id',y='tran_amount',data=top_5_sal)
```

```
<Axes: xlabel='customer_id', ylabel='tran_amount'>
```




```
#advanced analytics
#time series analysis

import matplotlib.dates as mdates
a['month_year']=a['trans_date'].dt.to_period('M')
monthly_sales=a.groupby('month_year')['tran_amount'].sum()
monthly_sales.index=monthly_sales.index.to_timestamp()
plt.figure(figsize=(12,6))
plt.plot(monthly_sales.index,monthly_sales.values)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%y-%m'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=6))
plt.xlabel('month year')
plt.ylabel('sales')
plt.title('monthly sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
# cohont segmentation

recency=a.groupby('customer_id')['trans_date'].max()
#frequency
frequency=a.groupby('customer_id')['trans_date'].count()

#monetary

monetary=a.groupby('customer_id')['tran_amount'].sum()

#combine
rfm=pd.DataFrame({'recency':recency,'frequecy':frequency,'monetary':mo
netary})

rfm
```

customer_id	recency	frequecy	monetary
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

[6884 rows x 3 columns]

```
#customer segmentation

def segment_customer(row):
    if row['recency'].year>=2021 and row['frequency']>=15 and
row['monetary']>1000:
        return 'p0'
    elif(2011<=row['recency'].year>=2021 and row['frequency']>=15 and
row['monetary']>1000):
        return 'p1'
    else:
        return 'p2'
rfm['segment']=rfm.apply(segment_customer,axis=1)


rfm
```

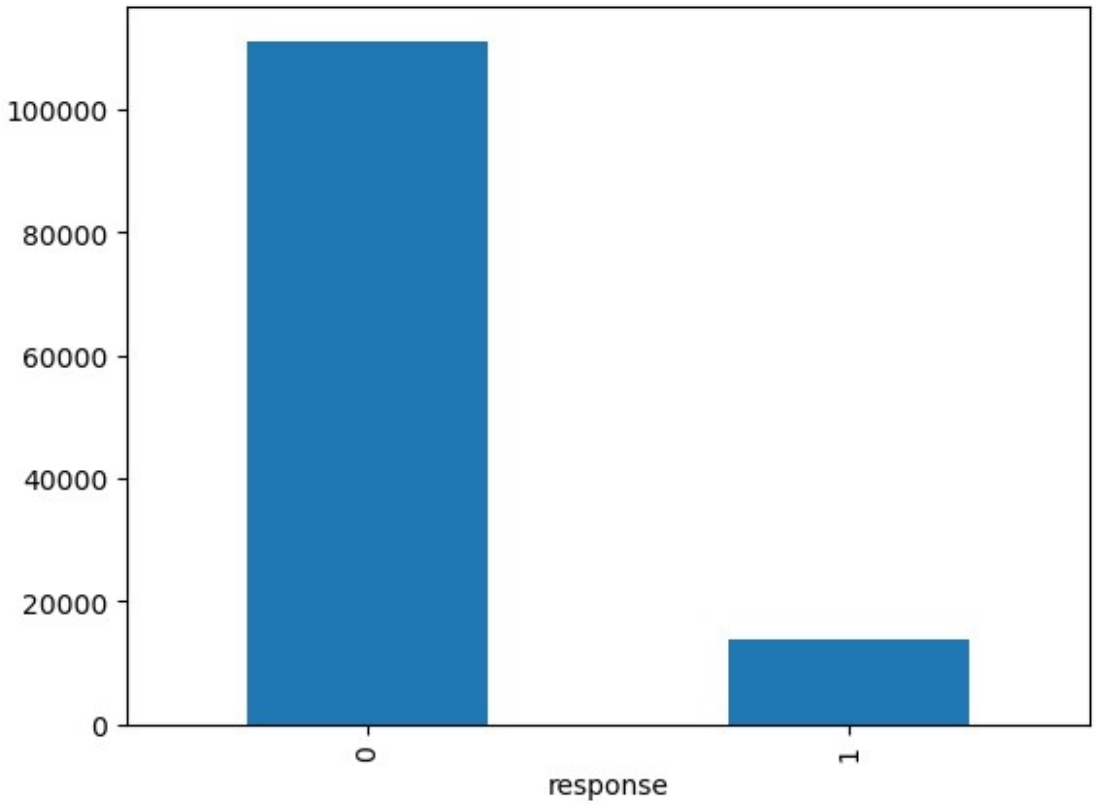
customer_id	recency	frequecy	monetary	segment
CS1112	2015-01-14	15	1012	p2
CS1113	2015-02-09	20	1490	p2
CS1114	2015-02-12	19	1432	p2
CS1115	2015-03-05	22	1659	p2
CS1116	2014-08-25	13	857	p2
...	...	...	...	...
CS8996	2014-12-09	13	582	p2
CS8997	2014-06-28	14	543	p2
CS8998	2014-12-22	13	624	p2
CS8999	2014-07-02	12	383	p2
CS9000	2015-02-28	13	533	p2

[6884 rows x 4 columns]

```
#churn analysis

churn_counts=a['response'].value_counts()
#plot
churn_counts.plot(kind='bar')
```

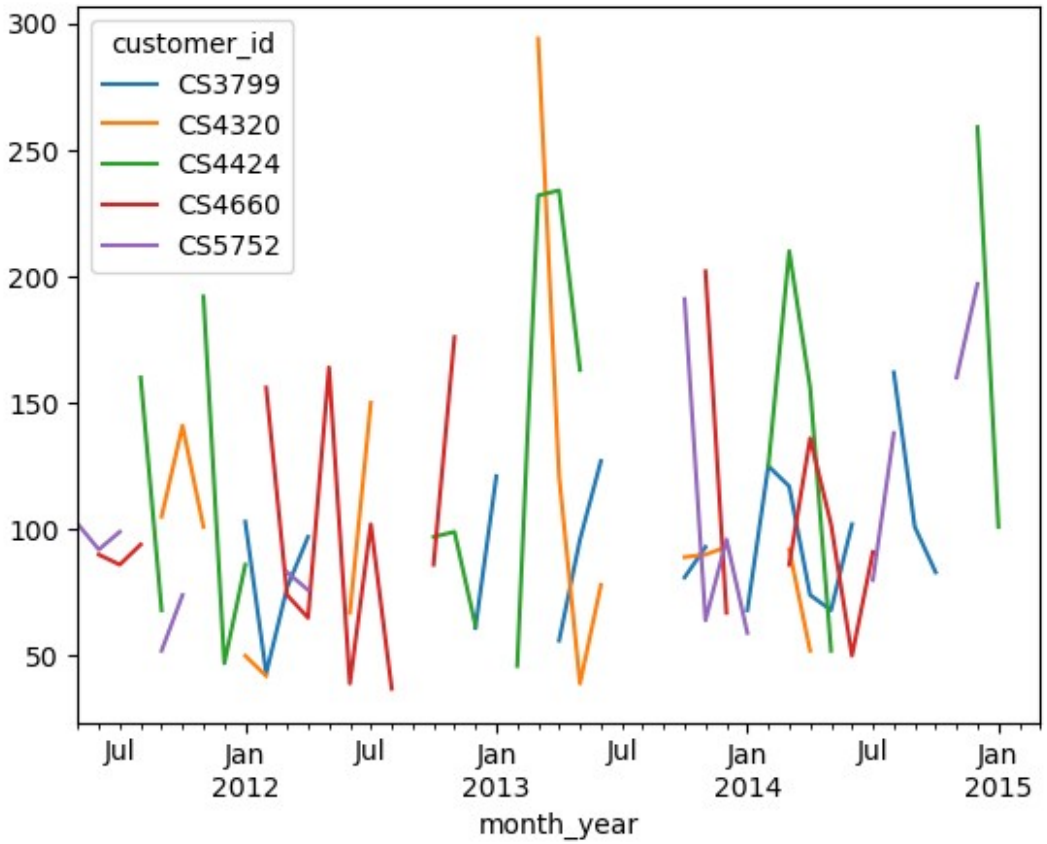
<Axes: xlabel='response'>




```
# analysing top customer
top_5_cus=monetary.sort_values(ascending=False).head(5).index
top_customers_a=a[a['customer_id'].isin(top_5_cus)]

top_customer_sales=top_customers_a.groupby(['customer_id','month_year'
])['tran_amount'].sum().unstack(level=0)
top_customer_sales.plot(kind='line')
```

```
<Axes: xlabel='month_year'>
```




```
a
```

	customer_id	response	trans_date	tran_amount	month	month_year
0	CS1112	0	2015-01-14	39	1	2015-01
1	CS1112	0	2014-07-16	90	7	2014-07
2	CS1112	0	2014-04-29	63	4	2014-04
3	CS1112	0	2014-12-04	59	12	2014-12
4	CS1112	0	2012-04-08	56	4	2012-04
...	...	...	...	...	...	...
124964	CS9000	0	2012-05-12	53	5	2012-05
124965	CS9000	0	2014-05-08	20	5	2014-05
124966	CS9000	0	2015-02-28	34	2	2015-02
124967	CS9000	0	2012-06-01	37	6	2012-06
124968	CS9000	0	2012-12-11	49	12	2012-12

```
[124969 rows x 6 columns]

a.to_csv('MainData.csv')

rfm.to_csv('Addanlys.csv')
```