Transaction Sales Data.

1. **Data preprocessing steps which I followed.**

First of all find the null values which is present in the datasets.

```
df.isna().sum()
```

```
TransactionID       0
CustomerID          5
TransactionDate     1
Trasaction Time     1
ProductID           0
ProductCategory     0
Quantity            0
PricePerUnit       14
TotalAmount        14
TrustPointsUsed     0
PaymentMethod      10
DiscountApplied     5
dtype: int64
```

Remove the all the rows which has maximum number of the null values as we have found that price Per Unit and Total Amount.

```
### As we have checked that Total Amount and price per unit price is both null there nothing we can do so we have scrap
df[df['TotalAmount'].isna()==True]
```

	TransactionID	CustomerID	TransactionDate	Trasaction Time	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod
4	5	1001.0	09-08-2024	09:00:00	2008	Grocery	1	NaN	NaN	20	Trust Points
5	6	1001.0	NaN	NaN	2007	Home Decor	1	NaN	NaN	20	Credit Card
8	9	1004.0	02-08-2024	23:00:00	2008	Fashion	1	NaN	NaN	-10	NaN
10	11	1001.0	09-08-2024	07:00:00	2003	Grocery	5	NaN	NaN	-10	Credit Card
12	13	1005.0	09-08-2024	22:00:00	2008	Grocery	3	NaN	NaN	100	Trust Points
16	17	1002.0	09-08-2024	15:00:00	2001	Toys	1	NaN	NaN	20	NaN
19	20	1002.0	03-08-2024	04:00:00	2007	Fashion	2	NaN	NaN	-10	Cash
23	24	1002.0	08-08-2024	19:00:00	2005	Electronics	1	NaN	NaN	100	Trust Points
30	31	1004.0	04-08-2024	16:00:00	2001	Home Decor	3	NaN	NaN	0	Credit Card
36	37	1002.0	09-08-2024	23:00:00	2005	Fashion	1	NaN	NaN	100	NaN
45	46	1004.0	01-08-2024	04:00:00	2004	Toys	1	NaN	NaN	100	NaN
47	48	1003.0	02-08-2024	03:00:00	2005	Home Decor	0	NaN	NaN	50	NaN
48	49	1003.0	06-08-2024	14:00:00	2007	Electronics	-1	NaN	NaN	0	Cash
49	50	1001.0	09-08-2024	08:00:00	2007	Grocery	3	NaN	NaN	50	Trust Points

As we see some quantity is -ve so we assume that -ve value shows that return the product

```
## fixing payment methord
df1['PaymentMethod'] = np.where(df1['Quantity'] <= -1, 'Return', df1['PaymentMethod'])
df1[df1['Quantity']<=-1]
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_15316\352294960.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PaymentMethod'] = np.where(df1['Quantity'] <= -1, 'Return', df1['PaymentMethod'])
```

	TransactionID	CustomerID	TransactionDate	Trasaction Time	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod
6	7	1001.0	2024-08-01	13:00:00	2007	Home Decor	-1	30.0	-30.0	-10	Return
34	35	1001.0	2024-08-06	08:00:00	2002	Electronics	-1	500.0	-500.0	100	Return
38	39	1004.0	2024-08-06	18:00:00	2007	Home Decor	-1	10.0	-10.0	20	Return

We also found that CustomerID has some null values to clear those null values. So we look who is most frequent buyer for product from each category.

```
df1['CustomerID'].fillna(df1['CustomerID'].mode()[0], inplace=True)
df1['CustomerID'] = np.where(df1['ProductCategory'] == 'Electronics',
                             df1['CustomerID'],
                             df1['CustomerID'])
df1[df1['ProductCategory'] == 'Home Decor']
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_15316\1855841639.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['CustomerID'].fillna(df1['CustomerID'].mode()[0], inplace=True)
C:\Users\abhis\AppData\Local\Temp\ipykernel_15316\1855841639.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['CustomerID'] = np.where(df1['ProductCategory'] == 'Electronics',
```

)	CustomerID	TransactionDate	Trasaction Time	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod	DiscountApplied
2	1001.0	2024-08-07	01:00:00	2004	Home Decor	1	10.0	10.0	0	Credit Card	20.0
7	1001.0	2024-08-01	13:00:00	2007	Home Decor	-1	30.0	-30.0	-10	Return	NaN
9	1003.0	2024-08-05	14:00:00	2005	Home Decor	3	500.0	1500.0	100	Cash	50.0
3	1004.0	2024-08-02	16:00:00	2007	Home Decor	2	10.0	20.0	100	Trust Points	30.0

There is some transaction the pay method is not define to figure out this we replaces null values with most frequent payment method is used.

```
df1['PaymentMethod'].fillna(df1['PaymentMethod'].mode()[0], inplace=True)
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_15316\2453949572.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['PaymentMethod'].fillna(df1['PaymentMethod'].mode()[0], inplace=True)
```

```
df1['PaymentMethod'].value_counts()
```

```
PaymentMethod
Cash            18
Trust Points     9
Credit Card      6
Return           3
Name: count, dtype: int64
```

We also found that discount value is also null so as we assume that in such cases no discount is provided.

```
df1['DiscountApplied'].fillna(0,inplace=True)
df1.head()
```

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_15316\3049835226.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['DiscountApplied'].fillna(0,inplace=True)
```

)	CustomerID	TransactionDate	Trasaction Time	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod	DiscountApplied
1	1002.0	2024-08-08	22:00:00	2008	Grocery	1	10.0	10.0	20	Trust Points	5.0
2	1001.0	2024-08-07	01:00:00	2004	Home Decor	1	10.0	10.0	0	Credit Card	20.0
3	1004.0	2024-08-02	19:00:00	2002	Grocery	3	30.0	90.0	0	Credit Card	25.0
2	1003.0	2024-08-07	17:00:00	2001	Toys	2	30.0	60.0	50	Cash	20.0
7	1001.0	2024-08-01	13:00:00	2007	Home Decor	-1	30.0	-30.0	-10	Return	0.0

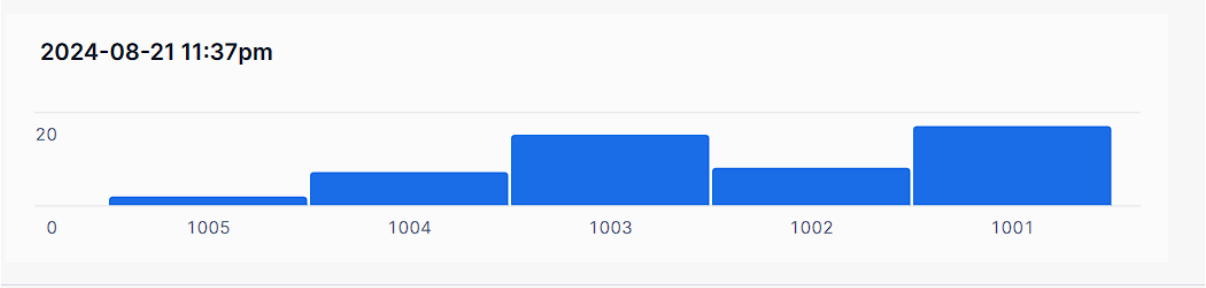
2. **Aggregation we used like sum and count is most used full in Quantity and Total amount**

- **Which customer order most quantity product**

```
//The customer which order most of thing
select "CustomerID",sum("Quantity")as orde_qty from trans_table
group by "CustomerID"
order by orde_qty desc;
```

esults Chart

CustomerID	ORDE_QTY
1001	17
1003	15
1002	8
1004	7
1005	2



- **Which Payment Method is most frequently used**

```
select "PaymentMethod",count("CustomerID") from trans_table
group by "PaymentMethod";
```

Results Chart

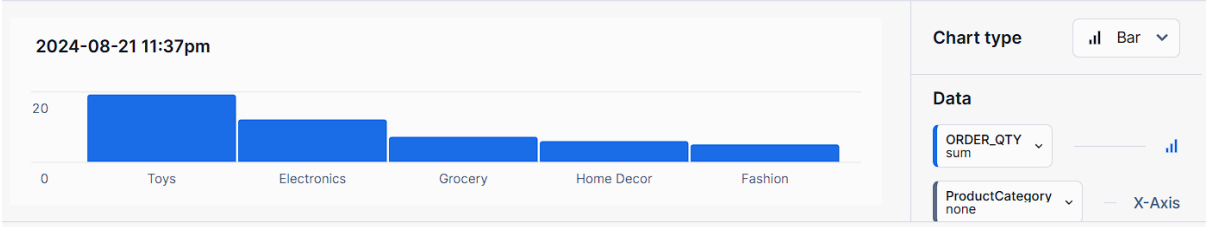
PaymentMethod	COUNT("CUSTOMERID")
Trust Points	9
Credit Card	6
Cash	18
Return	3

- **Which product category is most order by the customer.**

```
// most order product category
select "ProductCategory",sum("Quantity") as order_qty from trans_table
group by "ProductCategory"
order by order_qty  desc;
```

Results | Chart

ProductCategory	ORDER_QTY
Toys	19
Electronics	12
Grocery	7
Home Decor	6
Fashion	5

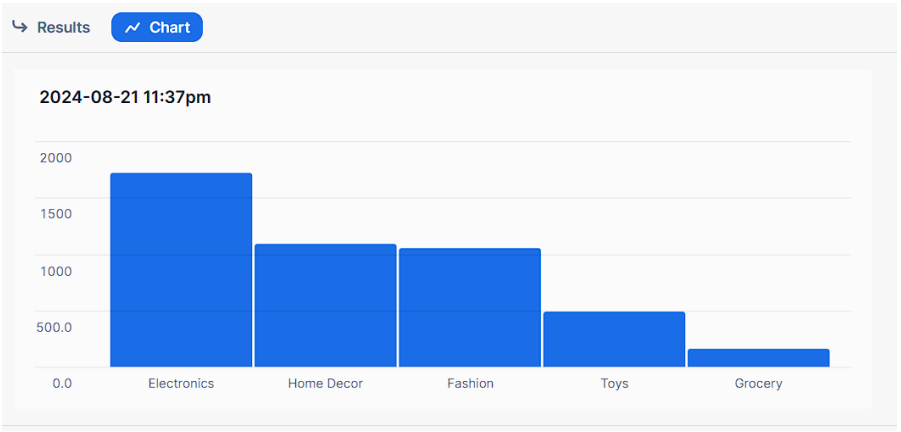


- **Which most is expensive product category.**

```
33
34  select "ProductCategory",sum("PricePerUnit") as Total_Amt from trans_table
35  group by "ProductCategory"
36  order by Total_Amt  desc;
37
```

Results | Chart

ProductCategory	TOTAL_AMT
Electronics	1720
Home Decor	1090
Fashion	1050
Toys	490
Grocery	160

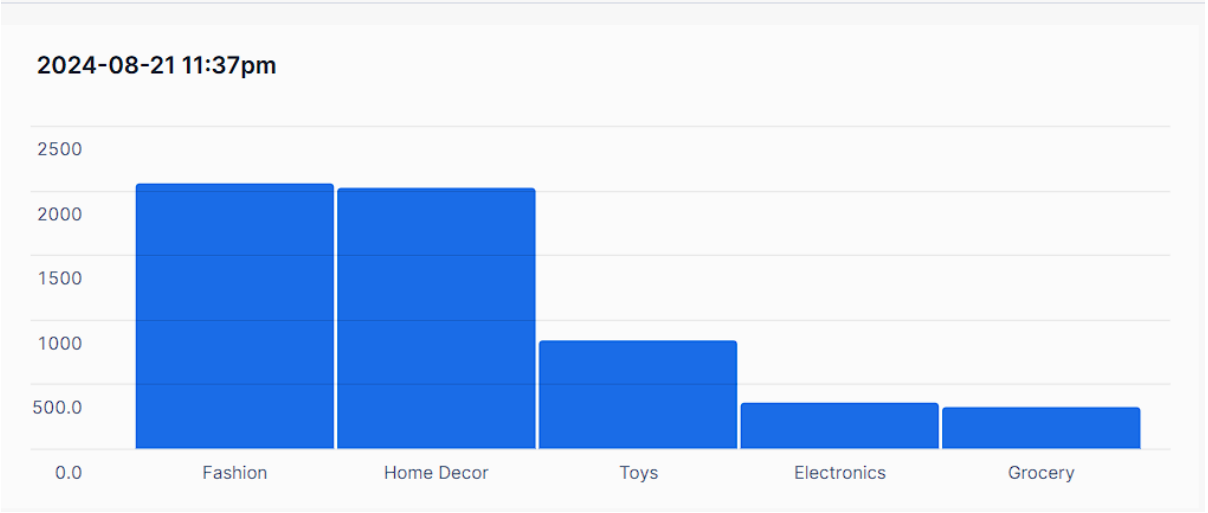


- **Which is most product category generates maximum revenue**

```
33
34  select "ProductCategory",sum("TotalAmount") as Total_Amt from trans_table
35  group by "ProductCategory"
36  order by Total_Amt  desc;
37
```

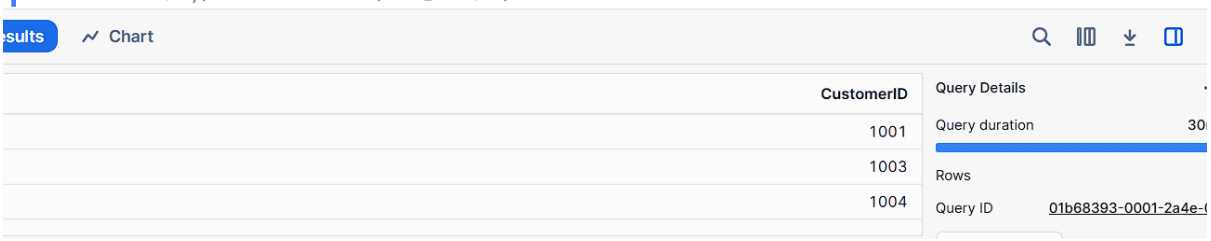
Results | Chart

ProductCategory	TOTAL_AMT
Fashion	2050
Home Decor	2020
Toys	840
Electronics	360
Grocery	320



3 Using Window function to found that who is most consecutive buyer 2 days.

```
select distinct "CustomerID" from(select "CustomerID","TransactionDate",lead("TransactionDate")over(PARTITION BY
"CustomerID" order by "TransactionDate" ) as next_date from trans_table
)
where datediff(day,"TransactionDate",next_date)=1;
```



All SQL CODE:

```
create database data_yuma_Assignment;
use data_yuma_Assignment;

CREATE or replace TABLE Trans_table (
    "Row_Number" FLOAT NOT NULL,
    "TransactionID" FLOAT NOT NULL,
    "CustomerID" FLOAT NOT NULL,
    "TransactionDate" DATE NOT NULL,
    "Trasaction Time" varchar NOT NULL,
    "ProductID" FLOAT NOT NULL,
    "ProductCategory" VARCHAR NOT NULL,
    "Quantity" FLOAT NOT NULL,
    "PricePerUnit" FLOAT NOT NULL,
    "TotalAmount" FLOAT NOT NULL,
    "TrustPointsUsed" FLOAT NOT NULL,
    "PaymentMethod" VARCHAR NOT NULL,
    "DiscountApplied" FLOAT
);

//The customer which order most of thing
select "CustomerID",sum("Quantity")as orde_qty from trans_table
group by "CustomerID"
order by orde_qty desc;
//The PaymentMethod used to most orders
select "PaymentMethod",count("CustomerID") from trans_table
group by "PaymentMethod";

// Most order product categoty
select "ProductCategory",sum("Quantity") as order_qty from trans_table
group by "ProductCategory"
order by order_qty  desc;

select "ProductCategory",sum("TotalAmount") as Total_Amt from trans_table
group by "ProductCategory"
order by Total_Amt  desc;
```

```
select dayname("TransactionDate") as Day_name,sum("TotalAmount")as Total_Amt from trans_table

group by Day_name

order by Total_Amt desc;

// The user who purchase in 2 consecutive days

select distinct "CustomerID" from(select "CustomerID","TransactionDate",lead("TransactionDate")over(PARTITION BY "CustomerID" order by "TransactionDate" ) as next_date from trans_table

)

where datediff(day,"TransactionDate",next_date)=1;
```