Rakamin Academy X KALBE Nutritionals

Final Task

Data Science Project-Based Virtual Internship: Rakamin Academy x Kalbe Nutritionals

Presented by
Biyan Bahtiar Ramadhan

Hi!
I’m Biyan

About Me

As a Certified Data Scientist and Analyst, I have expertise in crafting dashboards using Tableau, Power BI, and Looker Studio as well as analysis from exploratory to statistics and machine learning. With experience in multiple industry including healthcare, handling big data and my skills in using Python, SQL and Excel, I will always look for a way to obtain insights for you, either solo or in team.

Data Analyst Associate

• Provide analysis on stealth B2B SaaS company specializing in Human Resource Management System (HRIS).
• Created analytical dashboard that monitors funnel conversion rate and leads customer generations.
• Created leads segmentation to help company customize different marketing/sales strategy and performed impact analysis.
• Analyzed funnel conversion rate and gave recommendation and impact analysis to optimize conversion rate based on customer survey results and leads characteristics.

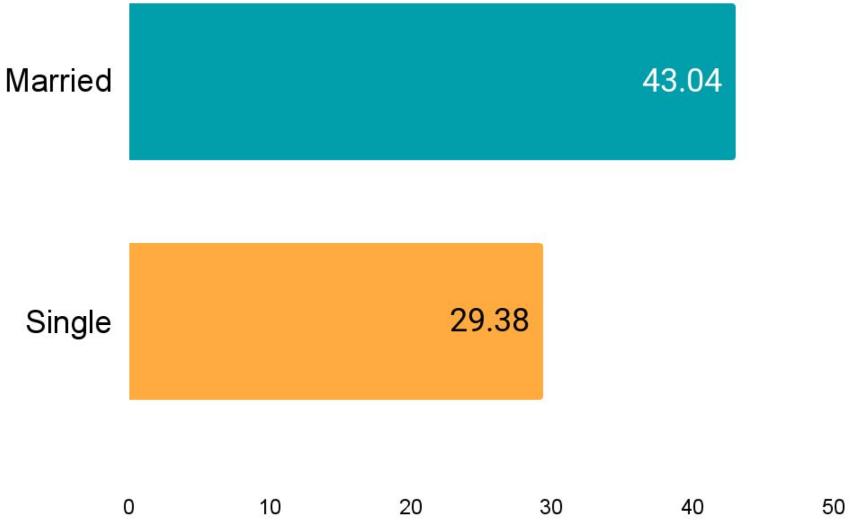
General Practitioner

• Performed thalassemia screening on over 1000 of patients yearly and make report for patients and physician.
• Input and maintained thalassemia screening database which contains more than 5000 record.
• Perform genetic counseling for patients with thalassemia and other rare disease such as Duchenne Muscular Distrophy (DMD), Disorder of Sex Development (DSD) and more.
• Did laboratory work in relation with thalassemia screening and prepared patient samples before sending to other laboratory.

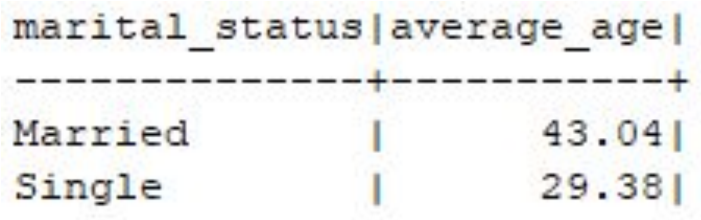
1

EDA using SQL

The Average Age Based on Marital Status

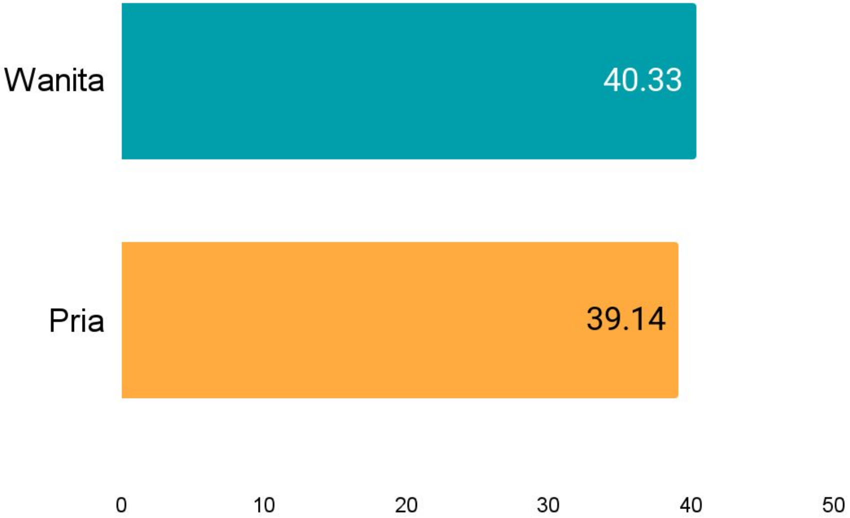


```
select
          "Marital Status" as marital_status
          , ROUND(AVG(age),2) as average_age
from public.customer
where "Marital Status" in ('Married', 'Single')
group by 1;
```

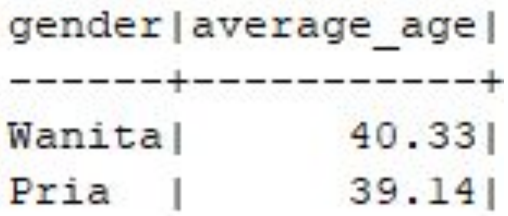


```
marital_status|average_age|
--------------+-----------+
Married       |      43.04|
Single        |      29.38|
```

The Average Age Based on Gender

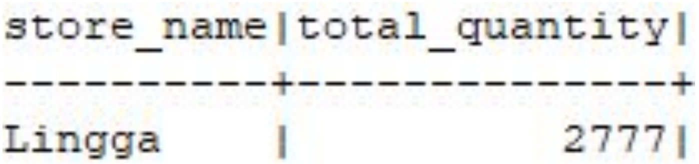


```
SELECT
        case
                when gender = 0 then 'Wanita'
                else 'Pria'
        end as gender
        , ROUND(AVG(age),2) as average_age
FROM public.customer
group by 1;
```



```
gender|average_age|
------+-----------+
Wanita|      40.33|
Pria  |      39.14|
```

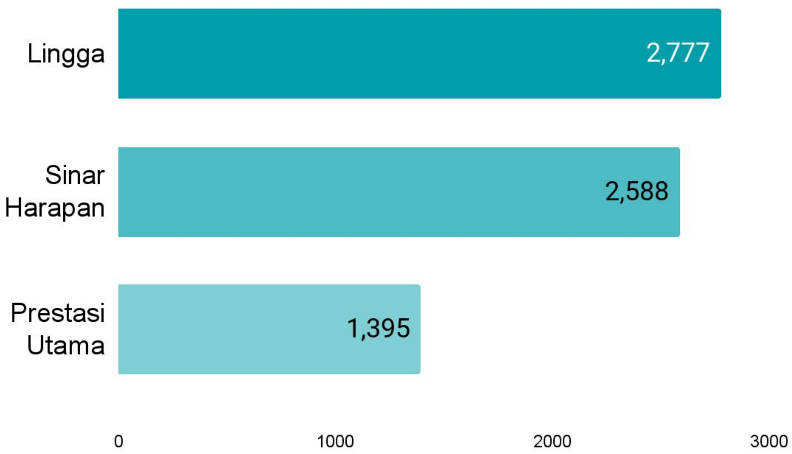
Store Name with The Most Quantity

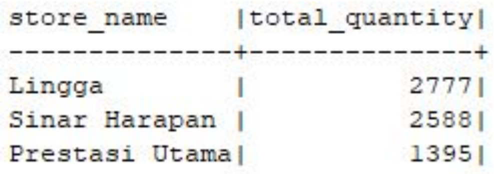


```
store_name|total_quantity|
----------+--------------+
Lingga    |          2777|
```

```
with table_1 as(
	select
		s.storename as store_name
		, sum(t.qty) as total_quantity
		, rank() over(order by sum(t.qty) desc) as
"rank"
	from store s
	inner join "transaction" t
		on s.storeid = t.storeid
	group by 1
)
select
	store_name
	, total_quantity
from table_1
where "rank" = 1;
```

Top 3 Store Name with The Most Quantity

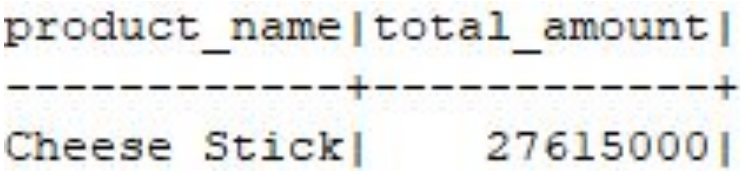




```
store_name     |total_quantity|
---------------+--------------+
Lingga         |          2777|
Sinar Harapan  |          2588|
Prestasi Utama |          1395|
```

```
with table_1 as(
	select
		s.storename as store_name
		, sum(t.qty) as total_quantity
		, rank() over(order by sum(t.qty) desc) as
"rank"
	from store s
	inner join "transaction" t
		on s.storeid = t.storeid
	group by 1
)
select
	store_name
	, total_quantity
from table_1
where "rank" <= 3;
```

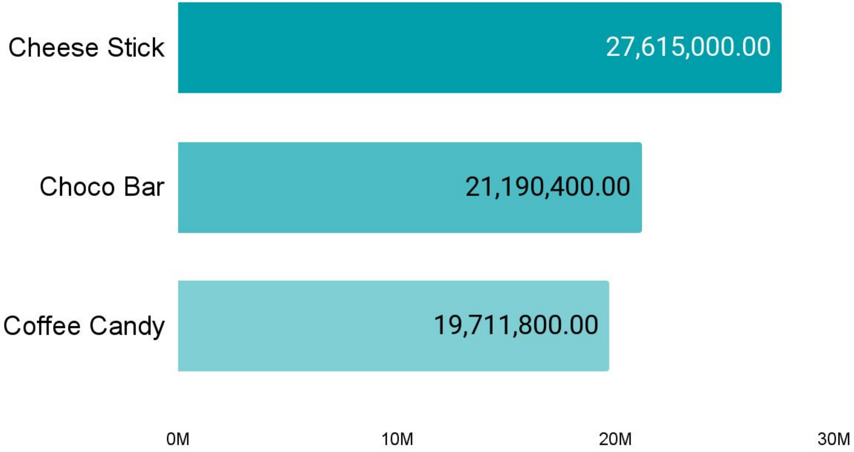
Product with The Most Total Amount

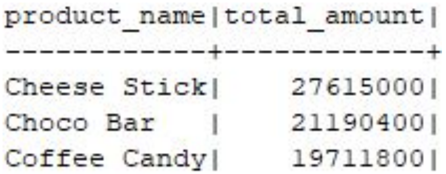


```
product_name|total_amount|
------------+------------+
Cheese Stick|    27615000|
```

```
with table_1 as(
	select
		p."Product Name" as product_name
		, sum(t.totalamount) as total_amount
		, rank() over(order by sum(t.totalamount) desc)
as "rank"
	from product p
	inner join "transaction" t
		on p.productid = t.productid
	group by 1
)
select
	product_name
	, total_amount
from table_1
where "rank" = 1;
```

Top 3 Product with The Most Total Amount



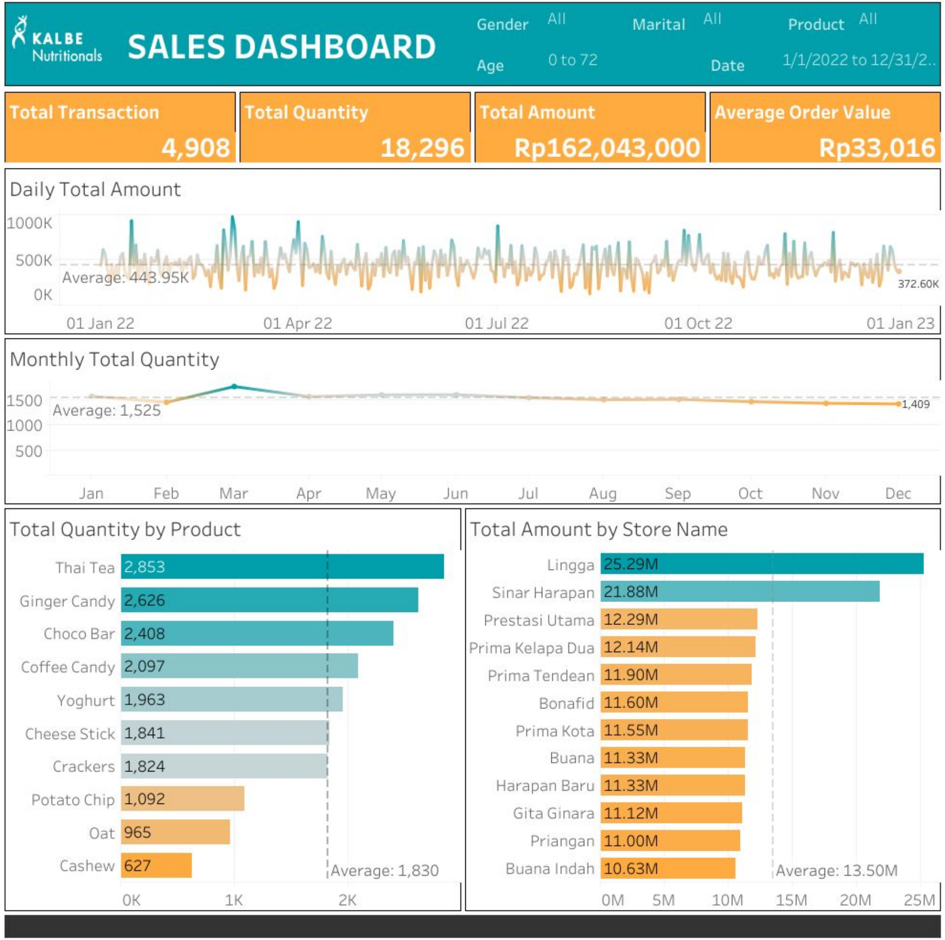


```
product_name|total_amount|
------------+------------+
Cheese Stick|    27615000|
Choco Bar   |    21190400|
Coffee Candy|    19711800|
```

```
with table_1 as(
	select
		p."Product Name" as product_name
		, sum(t.totalamount) as total_amount
		, rank() over(order by sum(t.totalamount) desc)
as "rank"
	from product p
	inner join "transaction" t
		on p.productid = t.productid
	group by 1
)
select
	product_name
	, total_amount
from table_1
where "rank" <= 3;
```

2

Dashboard in Tableau



KALBE Nutritionals

SALES DASHBOARD

Gender All

Marital All

Product All

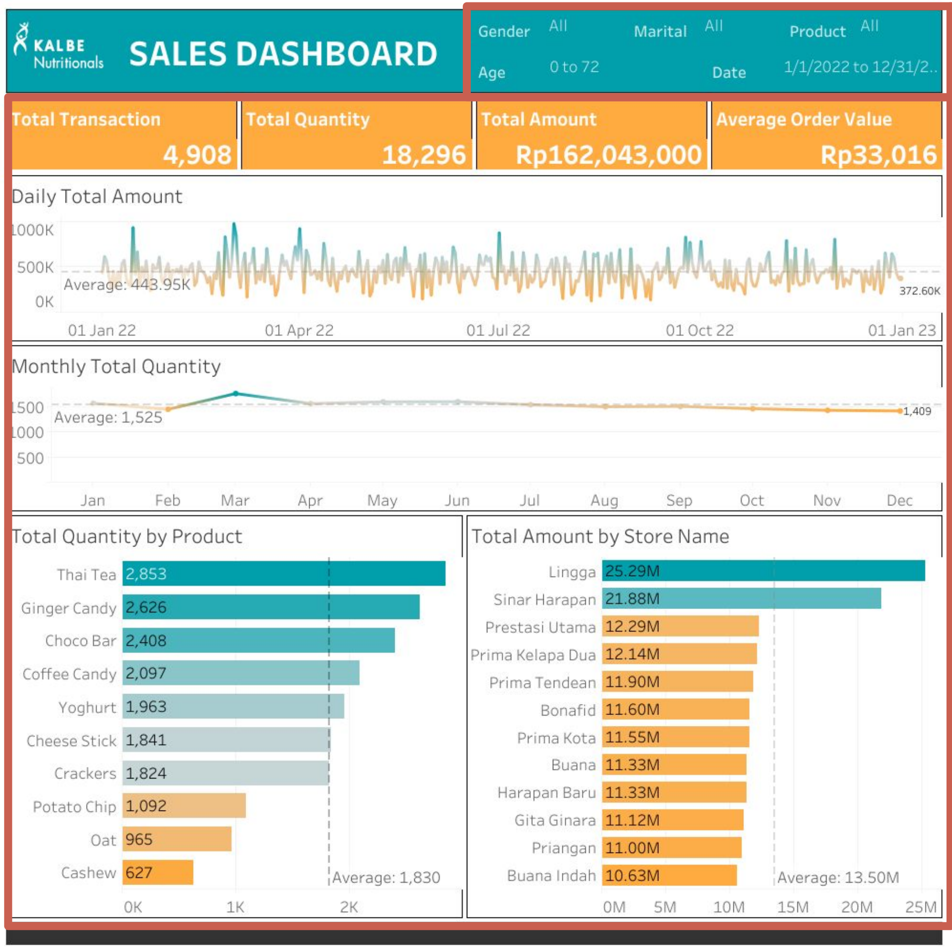
Age 0 to 72

Date 1/1/2022 to 12/31/2..

Total Transaction	Total Quantity	Total Amount	Average Order Value
4,908	18,296	Rp162,043,000	Rp33,016

Tableau Dashboard

(Link)



KALBE Nutritionals **SALES DASHBOARD**

Gender All | Marital All | Product All
Age 0 to 72 | Date 1/1/2022 to 12/31/2..

Total Transaction	Total Quantity	Total Amount	Average Order Value
4,908	18,296	Rp162,043,000	Rp33,016

Gender All | Marital All | Product All
Age 0 to 72 | Date 1/1/2022 to 12/31/2..

User can select the filter they want by clicking these dropdowns. Available filter are **gender**, **marital status**, **product**, **age** and **date.**

Available chart are **Daily Total Amount**, **Monthly Total Quantity**, **Total Quantity by Product** and **Total Amount by Store Name.**

Scorecard are **Total Transaction**, **Total Quantity, Total Amount** and **AOV.**

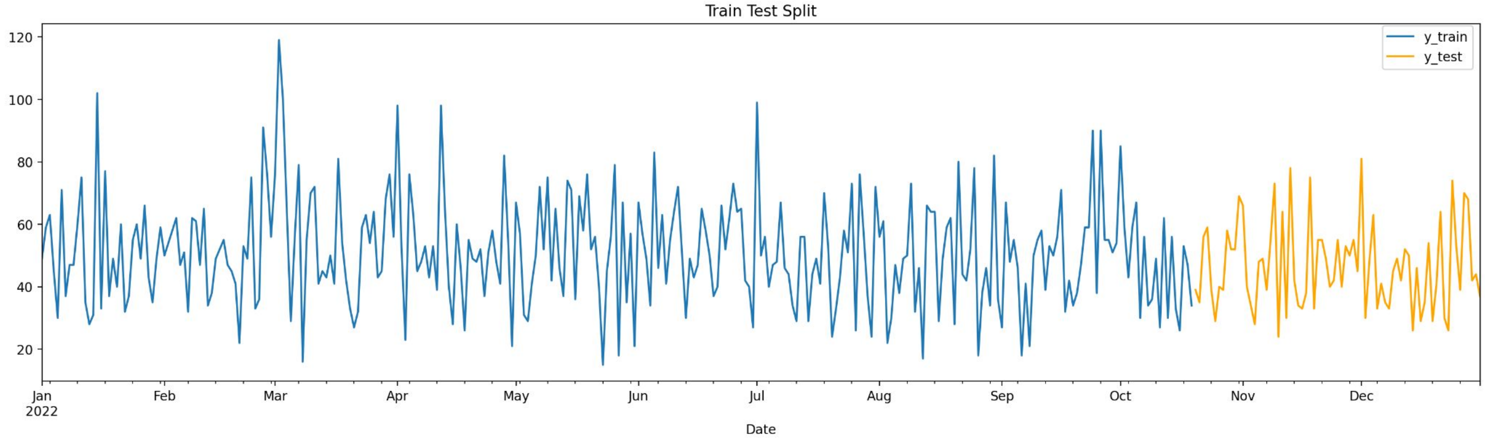
All of these are affected by filter.

3

Stock Inventory Forecast Using ARIMA

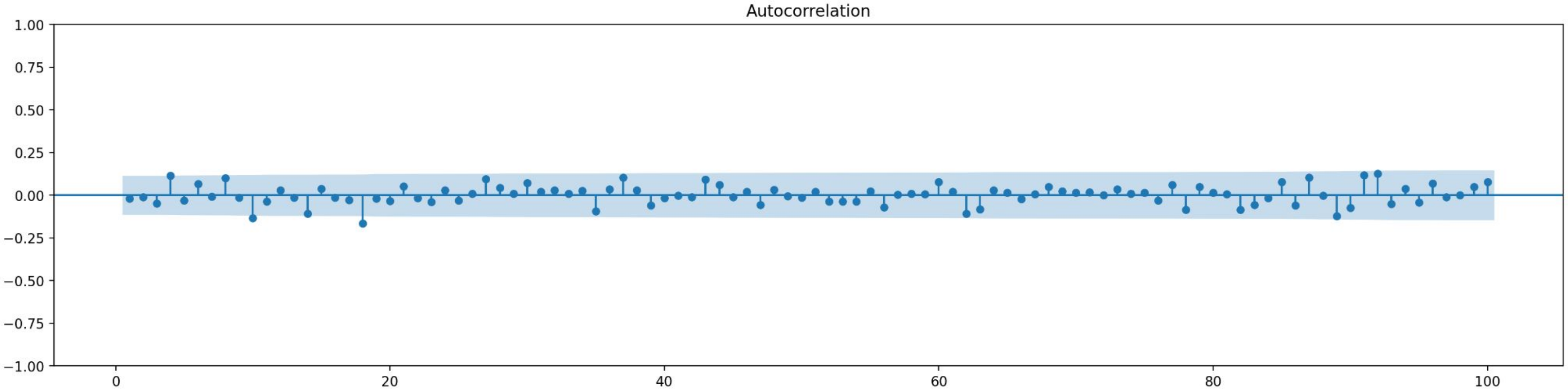
1. Prepare the Data and Train-Test Split

```
# prepare the data
arima_df = transaction_df[['Date',
'Qty']].copy().sort_values(by='Date').groupby('Date')[['Qty']].sum()
# train_test_split
arima_df_len = round(len(arima_df)*0.8)
y_train, y_test = arima_df[['Qty']].iloc[:arima_df_len], arima_df[['Qty']].iloc[arima_df_len:]
```



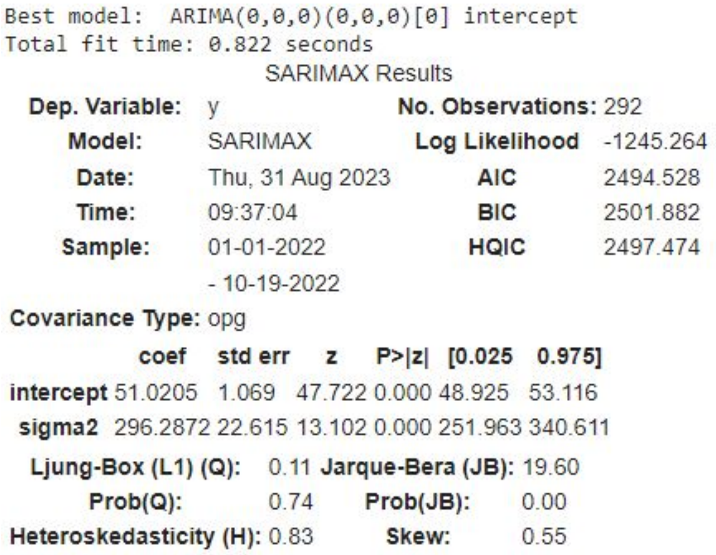
2. Determine Seasonality - No Seasonality

```
# No seasonality
acf_data = y_train.copy()
acf_data = acf_data - acf_data.rolling(9).mean()
acf_data.dropna(inplace=True)
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(20,10))
plot_acf(acf_data, lags=100, zero=False, ax=ax[0])
sns.lineplot(acf_data, ax=ax[1])
plt.show()
```



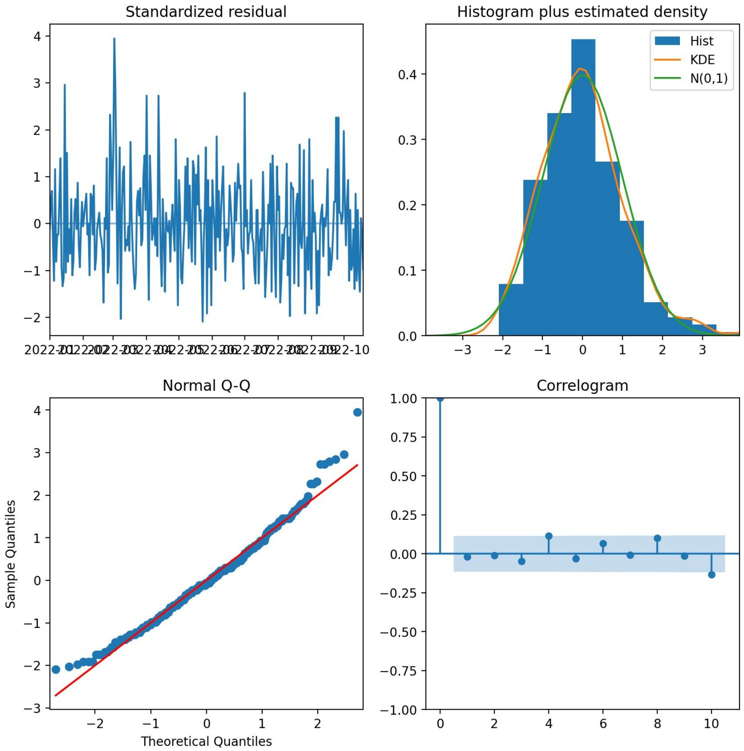
3. Determine Differencing and Create Model

```python
# decide the correct differencing and create model
model = auto_arima(y_train, trace=True, suppress_warnings=True, max_p=20, max_d=20,
                   max_q=20)
model.summary()
```



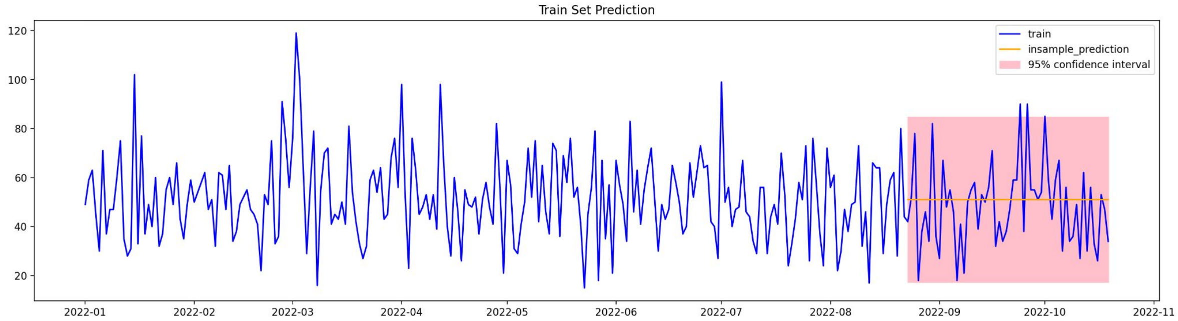
```
Best model:  ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 0.822 seconds
                    SARIMAX Results
  Dep. Variable:    y                  No. Observations: 292
         Model:     SARIMAX              Log Likelihood  -1245.264
          Date:     Thu, 31 Aug 2023          AIC        2494.528
          Time:     09:37:04                  BIC        2501.882
        Sample:     01-01-2022               HQIC        2497.474
                    - 10-19-2022
Covariance Type: opg
                coef     std err    z      P>|z|  [0.025   0.975]
intercept  51.0205   1.069   47.722 0.000 48.925   53.116
 sigma2   296.2872 22.615 13.102 0.000 251.963 340.611
  Ljung-Box (L1) (Q):   0.11 Jarque-Bera (JB): 19.60
            Prob(Q):    0.74      Prob(JB):     0.00
Heteroskedasticity (H): 0.83        Skew:       0.55
```

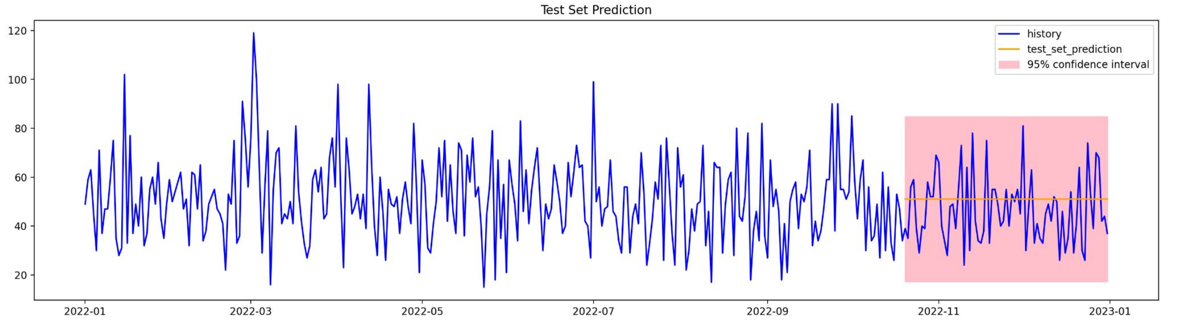
4. Plot Diagnostics



```
# plot_diagnostics
model.plot_diagnostics(figsize=(10,10))
plt.show()
```

5. Train & Test Set RMSE

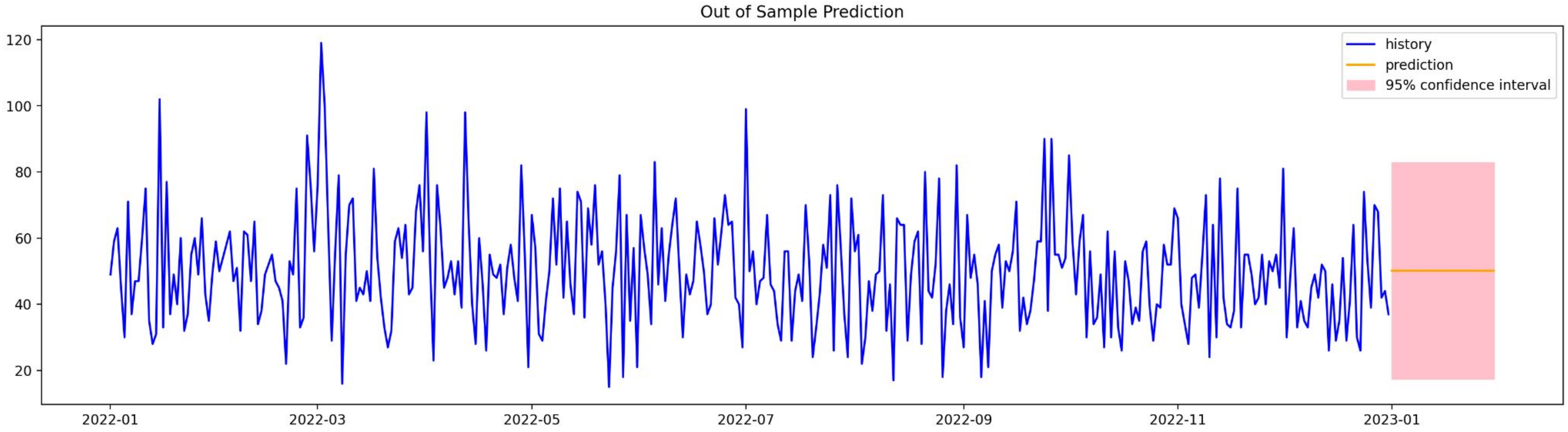




	RMSE
Train set	16.862
Test set	14.499

6. Out-of-Sample Prediction (30 days)

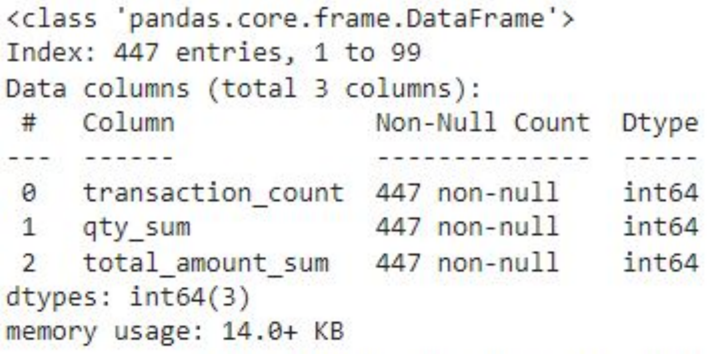
	Stock for Tomorrow	Stock for 1 Week	Stock for 30 days
Prediction	51	357	1530



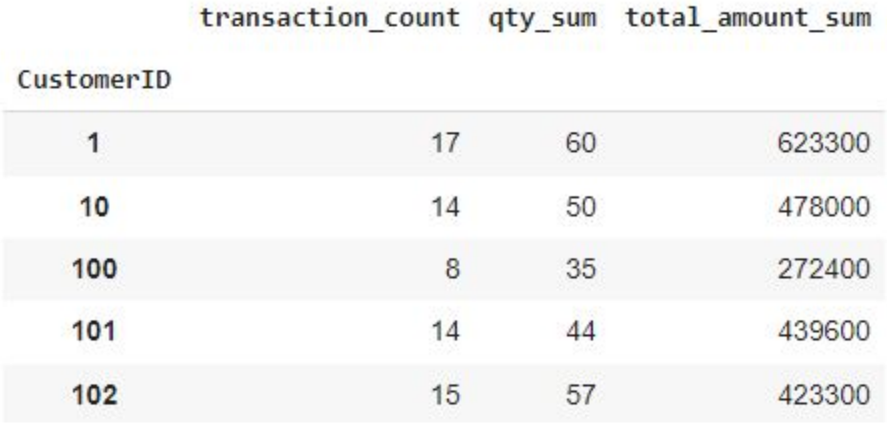
4

Customer Segmentation for Personalized Promotion

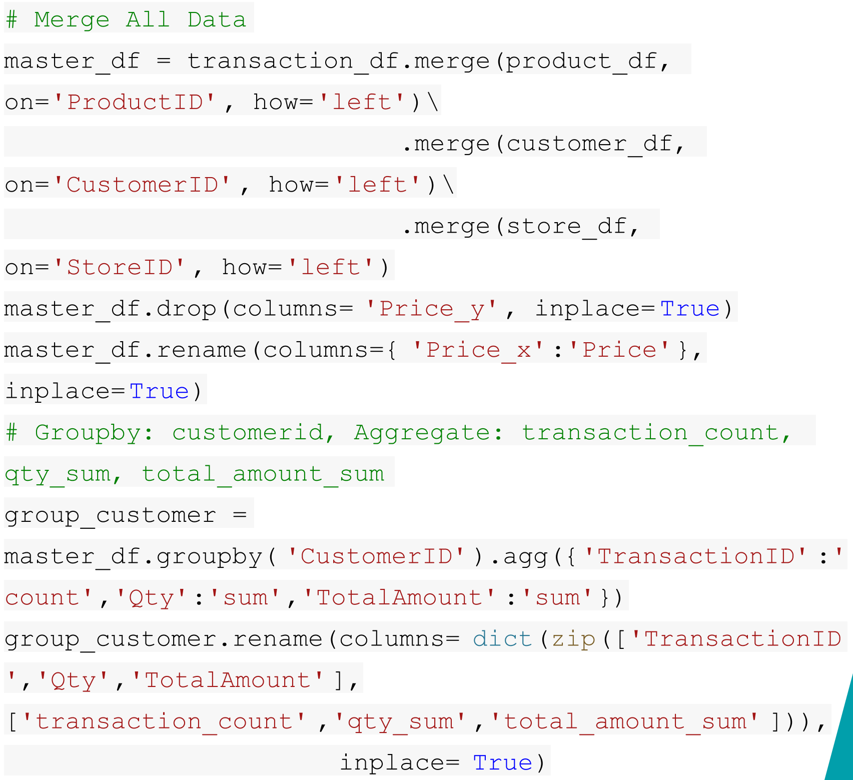
1. Preparation



```
<class 'pandas.core.frame.DataFrame'>
Index: 447 entries, 1 to 99
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   transaction_count  447 non-null    int64
 1   qty_sum            447 non-null    int64
 2   total_amount_sum   447 non-null    int64
dtypes: int64(3)
memory usage: 14.0+ KB
```

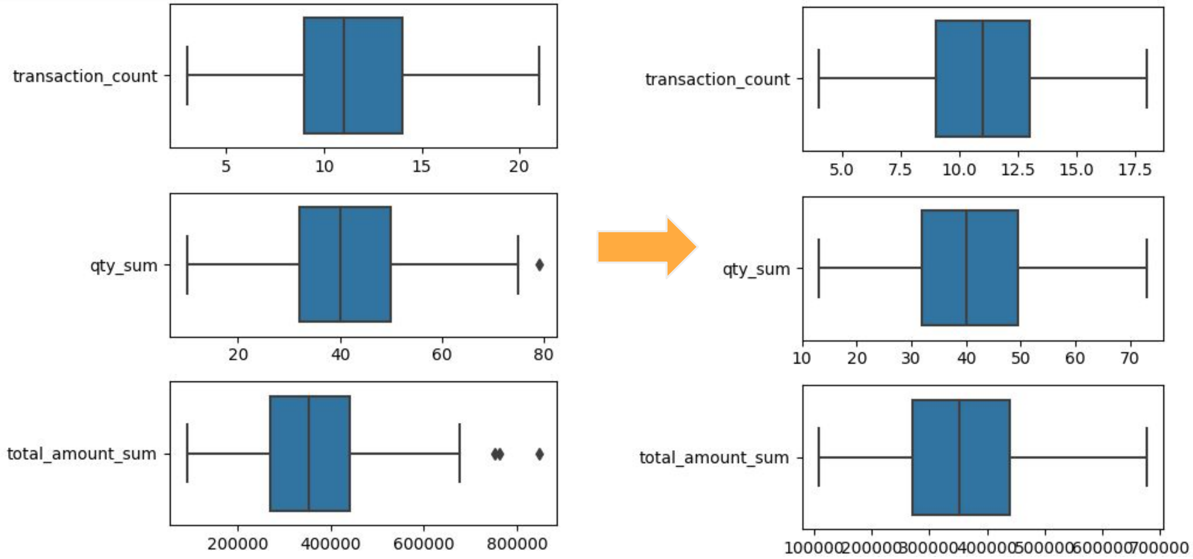


CustomerID	transaction_count	qty_sum	total_amount_sum
1	17	60	623300
10	14	50	478000
100	8	35	272400
101	14	44	439600
102	15	57	423300



```
# Merge All Data
master_df = transaction_df.merge(product_df,
on='ProductID', how='left')\
                                  .merge(customer_df,
on='CustomerID', how='left')\
                                  .merge(store_df,
on='StoreID', how='left')
master_df.drop(columns= 'Price_y', inplace=True)
master_df.rename(columns={ 'Price_x':'Price'},
inplace=True)
# Groupby: customerid, Aggregate: transaction_count,
qty_sum, total_amount_sum
group_customer =
master_df.groupby( 'CustomerID').agg({'TransactionID':'
count','Qty':'sum','TotalAmount':'sum'})
group_customer.rename(columns= dict(zip(['TransactionID
','Qty','TotalAmount'],
['transaction_count','qty_sum','total_amount_sum'])),
                             inplace= True)
```

2. Removing Outlier



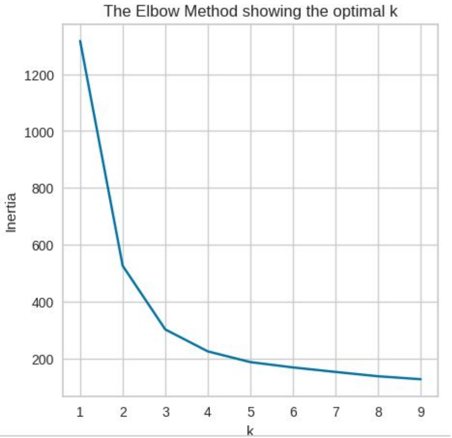
Before	After	%Data Removed
447	439	1,82%

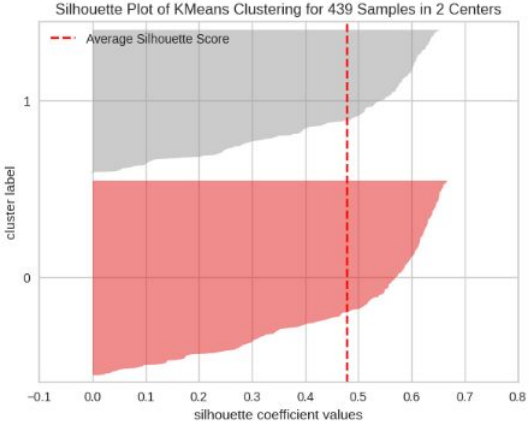
3. Preprocessing - Standardization

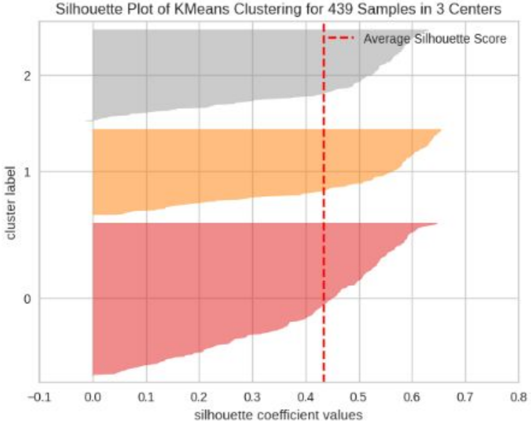
```
preprocessing = group_customer_no_outlier.copy()

# StandardScaler
ss = StandardScaler()
ss_preprocessing = ss.fit_transform(preprocessing[preprocessing.columns])
```

4. Decide Number of Cluster







Elbow method suggest 3 cluster while silhouette score suggest 2-3 cluster. I decide to pick **3 cluster**.

5. Segmentation Result

	💸High Spender	👌 Medium Spender	💔Low Spender
Total Customer	129	203	115
Female:Male	1,08:1	1,23:1	1,21:1
Married:Single	2,82:1	3,41:1	3,6:1
Avg. Age	40,4	38,7	39,7
Income	8,45	8,49	8,92
Avg. Transaction Cnt	14,9	11	7,4
Avg. Quantity Bought	56,3	39,5	26,0
Avg. Spending	511,679	348,426	220,050

Thanks!

Link to task folder

Link to presentation video

This presentation template was created by **Slidesgo**, including icons by_**Flaticon**, and infographics & images by **Freepik** and illustrations by **Stories**