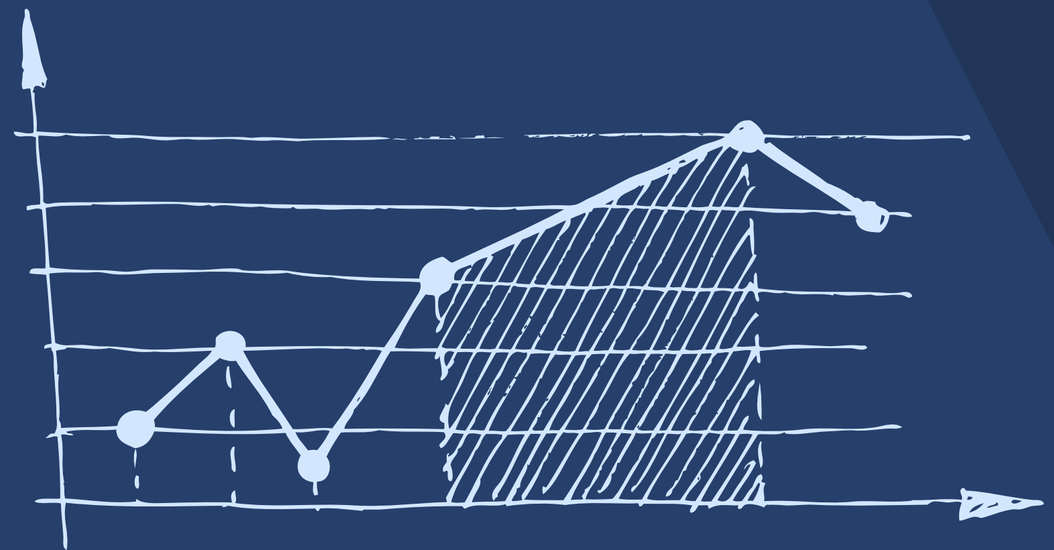
SALESENSE

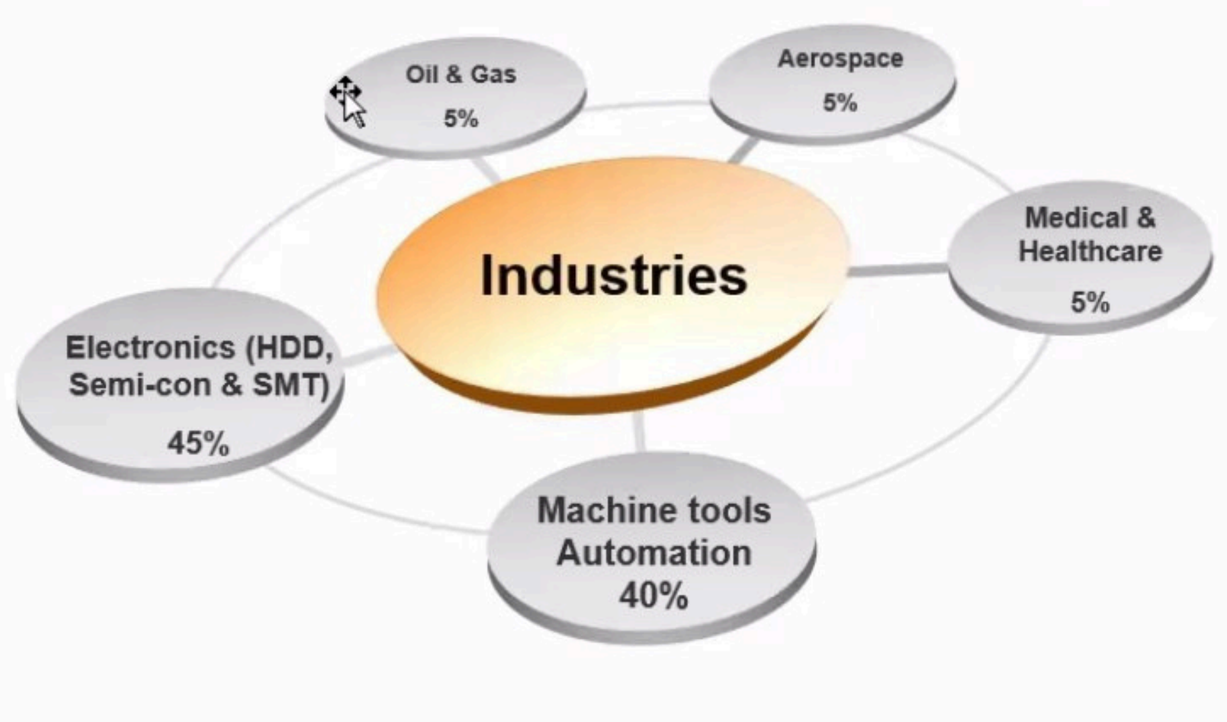
TEAM 4



SCOPE OF PRESENTATION

1. Background Information
2. The Problem
3. Methodology
4. Our Solution
5. Final Product

TSH GROUP



- Established in 2006
- 4 manufacturing sites in SEA
- Front end based in Singapore
- Back end is sited in Malaysia

TSH CORE BUSINESS

1. Precision Machining
2. Frame Fabrication
3. Sheet Metal Fabrication
4. Wire Harnessing
5. Module Assembly

PROBLEM STATEMENT

How might we develop a **sales visualisation dashboard** that includes a **forecasting model**, **highlighting key metrics** and **drill-down function** for detailed data analysis?

METHODOLOGY

Data Processing

Import and clean data using python libraries

Data Analysis

Analyse data, test forecasting models

Data Visualisation

Development of dashboard to display graphs and key metrics

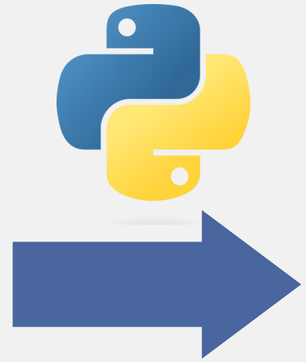
Final Presentation

Present final results, deliverables and poster

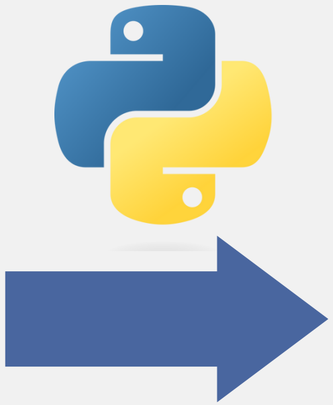
DATA PROCESSING

Raw data

#	Status	Catego	INV No	Custom	Curren	Delivery Dat	Cust.PO	SO No.	Inhous	Item	Project	Descrip	HS Code	LMW D	UOM	Quantit	ShipQT	Unit Pri	Total A	Exchan	Post Amount (MYR)
1	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22101	SOW22110004	-	18VS525A-	********	OP PANEL	8536509900	ELECTRICA	Set	10	10	26.6	266	3.2363	860.86
2	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22121	SOW22120441	-	ZR8-5NS	********	RING LUG	8466939000	MACHINE	EA	50	50	0.273	13.65	3.2363	44.18
3	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010462	-	E21VS431A	********	CABLE AW	8544429600	CABLE SET	EA	20	20	4.081	81.62	3.2363	264.15
4	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010460	-	E21VS410A	********	CABLE ATL	8544429600	CABLE SET	EA	10	10	4.081	40.81	3.2363	132.07
5	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23020030	-	E21VS415A	********	CABLE ATL	8544429600	CABLE SET	EA	10	10	3.983	39.83	3.2363	128.90
6	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010461	-	E21VS415A	********	CABLE ATL	8544429600	CABLE SET	EA	10	10	3.983	39.83	3.2363	128.90
7	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010405	-	E18VS692C	********	CABLE, LEA	8544429600	CABLE SET	EA	50	50	2.331	116.55	3.2363	377.19
8	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010406	-	E21VS701A	********	CABLE IOM	8544429600	CABLE SET	EA	1	1	31.339	31.339	3.2363	101.42
9	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23021	SOW23020015	-	18VS110A4	********	LEGEND, O	8544429600	CABLE SET	EA	20	20	0.21	4.2	3.2363	13.59
10	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23020008	-	E18VS145C	********	CABLE, TB1	8544429600	CABLE SET	EA	1	1	3.955	3.955	3.2363	12.80
11	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23020010	-	E18VS145C	********	CABLE TB1	8544429600	CABLE SET	EA	1	1	27.328	27.328	3.2363	88.44
12	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010243	-	E18VS692L	********	CABLE,RLY	8544429600	CABLE SET	EA	1	1	32.578	32.578	3.2363	105.43
13	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010354	-	E18VS692C	********	CABLE, LEA	8544429600	CABLE SET	EA	50	50	2.331	116.55	3.2363	377.19
14	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010408	-	E21VS701A	********	CABLE, TB	8544429600	CABLE SET	EA	1	1	6.006	6.006	3.2363	19.44
15	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22121	SOW22120177	-	E28VS300A	********	CABLE MT(	8544429600	CABLE SET	EA	2	2	19.845	39.69	3.2363	128.45
16	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO21091	SOB22050419	CX19-MTC-	26VSMTC-I	********	MTC VD_G	8537109900	MACHINE	Set	1	1	6298.824	6298.824	3.2363	20,384.88
17	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010194	-	E18VS694L	********	CABLE CBC	8544429600	CABLE SET	EA	1	1	23.296	23.296	3.2363	75.39
18	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22091	SOW22090195	-	E1HS692A(	********	CABLE, MT	8544429600	CABLE SET	EA	2	2	20.286	40.572	3.2363	131.30
19	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22081	SOW22080104	-	1CD014A-C	********	CB0 - CE M	8544429600	CABLE SET	Set	10	10	46.97	469.7	3.2363	1,520.09
20	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010415	-	Z18VS1008	********	MTC (MEC	8537109900	MACHINE	Set	1	1	1305.08	1305.08	3.2363	4,223.63
21	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010416	-	Z18VS1008	********	OP PANEL	8537109900	MACHINE	Set	1	1	181.72	181.72	3.2363	588.10
22	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22031	SOW22030627	-	1CD014A-1	********	NO JOG FE	8536509900	ELECTRICA	Set	1	1	31.269	31.269	3.2363	101.20
23	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010296	-	1CD014A-1	********	NON-EMI (	8536509900	ELECTRICA	Set	2	2	37.163	74.326	3.2363	240.54
24	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22071	SOW22070338	-	1CD024A-1	********	RH DOOR I	8536509900	ELECTRICA	EA	1	1	106.47	106.47	3.2363	344.57
25	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO21041	SOW21120860	-	26VS692A-	********	MTC VER.B	8536509900	ELECTRICA	EA	2	2	89.208	178.416	3.2363	577.41
26	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22031	SOW22030629	-	1CD014A-1	********	NO JOG FE	8536509900	ELECTRICA	Set	1	1	31.269	31.269	3.2363	101.20
27	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO22071	SOW22070339	-	1CD024A-1	********	RH DOOR I	8536509900	ELECTRICA	EA	2	2	106.47	212.94	3.2363	689.14
28	Open	Normal	TCM-0223(	TSH SYNER	SGD	10/2/2023	TPO23011	SOW23010350	-	1CD014A-1	********	NON-EMI (	8536509900	ELECTRICA	Set	1	1	37.163	37.163	3.2363	120.27



DATA PROCESSING



Processed Data

	SO Date	Revised Request Date	SO No.	Project	Quantity	Unit Price	Exchange	Non-ShipQTY	SONo.	UnitPrice	ExchangeRate	OrdersAmount	Outstanding	ReqWeek	ReqMonth	F
0	4/10/2021	4/10/2021	*******	*******	9		3.077	0	*******		3.077		0	40	Oct	
1	6/10/2021	6/10/2021	*******	*******	3		3.077	0	*******		3.077		0	41	Oct	
2	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
3	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
4	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
5	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
6	6/10/2021	6/10/2021	*******	*******	15		3.077	0	*******		3.077		0	41	Oct	
7	6/10/2021	6/10/2021	*******	*******	4		3.077	0	*******		3.077		0	41	Oct	
8	6/10/2021	6/10/2021	*******	*******	2		3.077	0	*******		3.077		0	41	Oct	
9	6/10/2021	6/10/2021	*******	*******	5		3.077	0	*******		3.077		0	41	Oct	
10	6/10/2021	6/10/2021	*******	*******	3		3.077	1	*******		3.077		23967.98	41	Oct	
11	6/10/2021	31/5/2022	*******	*******	3		3.077	0	*******		3.077		0	22	May	
12	6/10/2021	6/10/2021	*******	*******	5		3.077	0	*******		3.077		0	41	Oct	
13	6/10/2021	6/10/2021	*******	*******	4		3.077	0	*******		3.077		0	41	Oct	
14	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
15	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
16	6/10/2021	6/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
17	6/10/2021	6/10/2021	*******	*******	15		3.077	1	*******		3.077		15723.47	41	Oct	
18	6/10/2021	6/10/2021	*******	*******	15		3.077	0	*******		3.077		0	41	Oct	
19	7/10/2021	7/10/2021	*******	*******	2		3.077	0	*******		3.077		0	41	Oct	
20	7/10/2021	7/10/2021	*******	*******	1		3.077	0	*******		3.077		0	41	Oct	
21	7/10/2021	7/10/2021	*******	*******	3		3.077	0	*******		3.077		0	41	Oct	

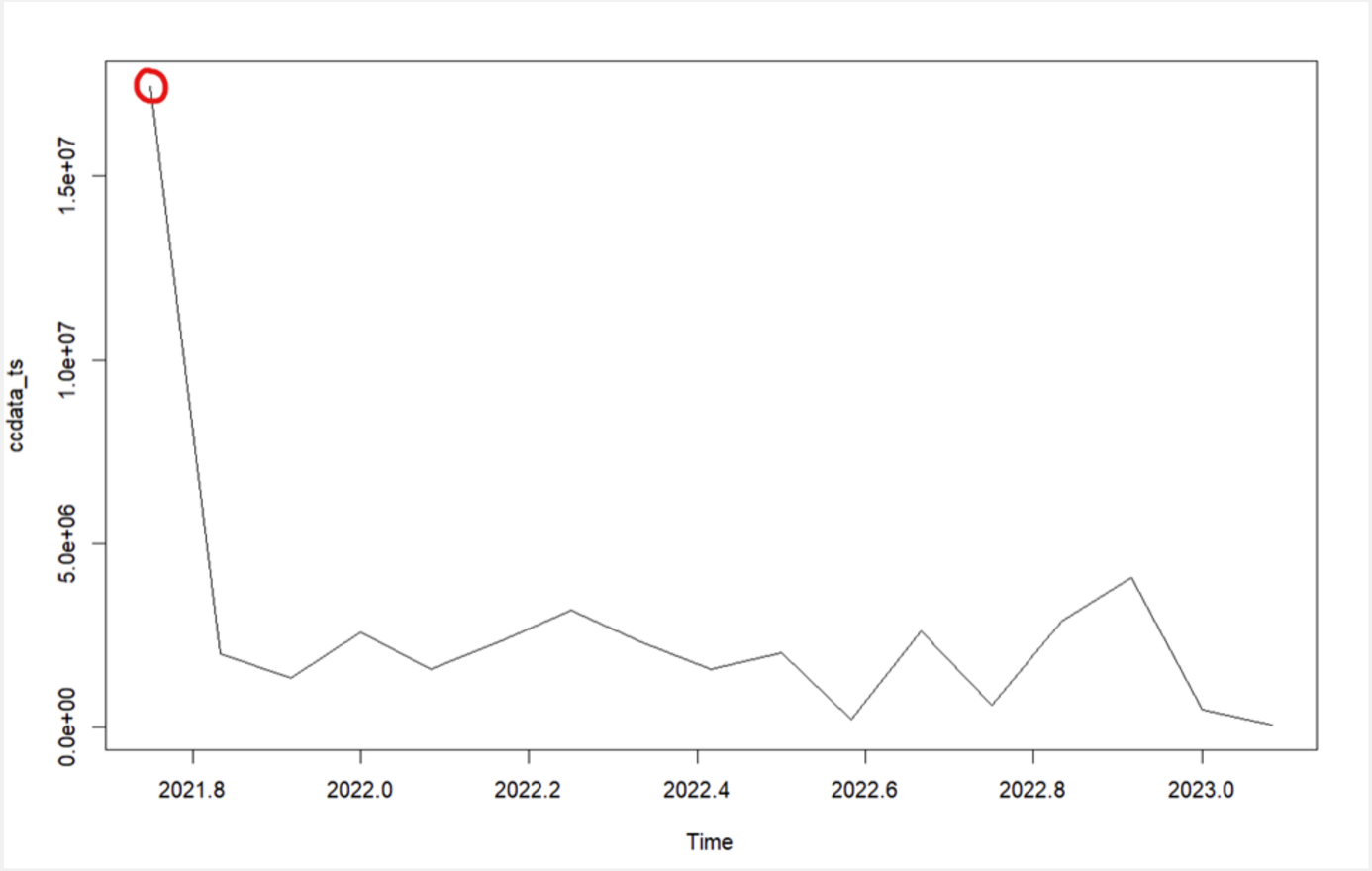
DATA PROCESSING

Python Code

```
    # Calculate the day of the week (0-6, where 0 is Sunday)
    leap_year_offset = int((year - 1900 - 1) / 4) + 1
    days_since_1900 = (year - 1900) * 365 + leap_year_offset + day_of_year - 1
    day_of_week = (days_since_1900 + 1) % 7  # 1/1/1900 was a Monday

    # Calculate the week number
    week_num = (day_of_year + day_of_week + 5) // 7

    return week_num

def date_to_month(date_str):
    date_obj = datetime.strptime(date_str, '%d/%m/%Y')
    month_str = date_obj.strftime('%b')
    return month_str

def date_to_year(date_str):
    date_obj = datetime.strptime(date_str, '%d/%m/%Y')
    year = date_obj.year
    return year

#Start processing
df2 = pd.read_csv(inputfilename)
#filter out the "Stop" status
df2 = df2[df2['Status'].isin(["Open", "Close" ,"Partially Close"])]
#select columns
col = ['SO Date', 'Revised Request Date', 'SO No.', 'Project', 'Quantity', 'Unit Price', 'Exchange Rate', 'Non-ShipQTY']
df_so = df2[col]
#data processing and remove the spacing between names of columns
df_so['SONo.'] = df_so['SO No.'].apply(lambda x: x[:3].upper())
df_so['Quantity'] = df_so['Quantity'].apply(str_to_float)
df_so['Non-ShipQTY'] = df_so['Non-ShipQTY'].apply(str_to_float)
df_so['UnitPrice'] = df_so['Unit Price'].apply(str_to_float)
df_so['ExchangeRate'] = df_so['Exchange Rate'].apply(str_to_float)
df_so['OrdersAmount'] = round(df_so['Quantity'] * df_so['UnitPrice'] * df_so['ExchangeRate'], 2)
df_so['Outstanding'] = round(df_so['Non-ShipQTY'] * df_so['UnitPrice'] * df_so['ExchangeRate'], 2)
#for wire harnessing
df_so_wh = df_so[df_so['SONo.'].isin(["SOW", "SOZ" ,"SOB", "SOK"])]
#add columns for weeknum, month, year
df_so_wh['ReqWeek'] = df_so_wh['Revised Request Date'].apply(weeknum)
df_so_wh['ReqMonth'] = df_so_wh['Revised Request Date'].apply(date_to_month)
df_so_wh['ReqYear'] = df_so_wh['Revised Request Date'].apply(date_to_year)

df_so_wh['OrderWeek'] = df_so_wh['SO Date'].apply(weeknum)
df_so_wh['OrderMonth'] = df_so_wh['SO Date'].apply(date_to_month)
df_so_wh['OrderYear'] = df_so_wh['SO Date'].apply(date_to_year)
#output a csv file
df_so_wh.to_csv('static/processed/Data SO_WH.csv')
```

DATA PROCESSING

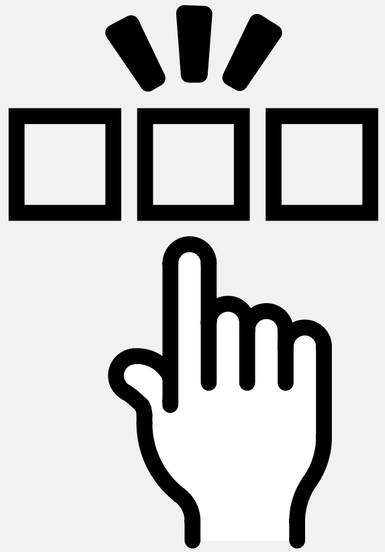
Outlier




```
# Convert to time series
ccdata_ts <- ts(revenue, start = c(2021, 11), frequency = 12)
```

DATA ANALYSIS

- Time series forecasting
- Insufficient data points to determine seasonality (2 years)
    - `time series has no or less than 2 periods`



1. ARIMA

- Linear Regression Model
- Exhibits autocorrelation and stationarity

2. Holt-Winters

- Exponential smoothing method
- Able to account for level, trend and seasonality of the data

FORECAST

1. ARIMA

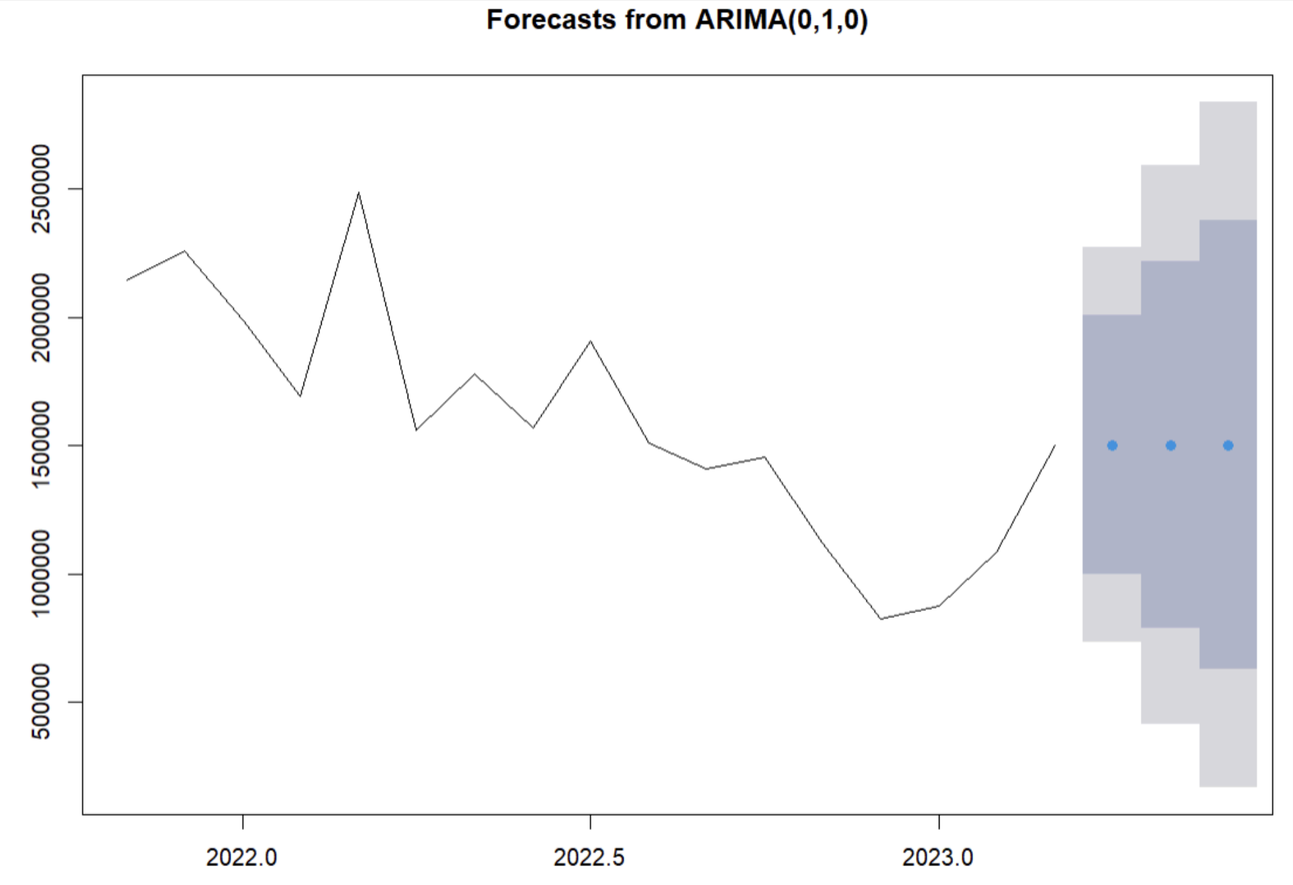
```
# Fit an ARIMA model to the current window
arima_model <- auto.arima(ts_window)

# Generate a forecast for the next 3 months with prediction intervals
forecast_next <- forecast(arima_model, h = 3, interval = "prediction")
```

2. Holt-Winters

```
# Holt-Winters ##################################
# Estimate value of alpha parameter
forecast <- HoltWinters(data_ts, gamma=FALSE)
# Display the results
forecast
```

```
# Prediction interval range at 95%
forecasts2 <- forecast:::forecast.HoltWinters(ccdata_forecast, h=3)
```

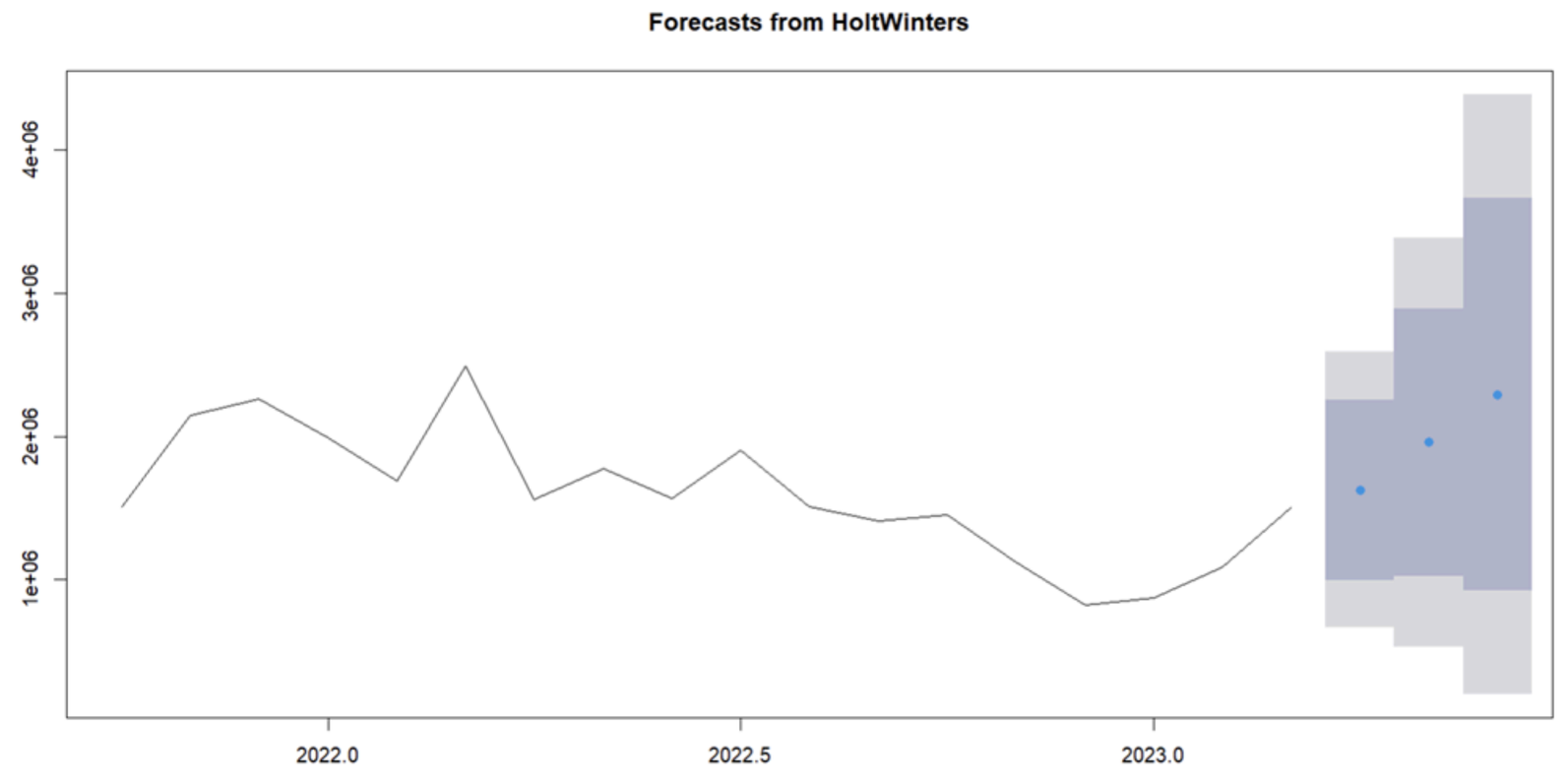
1. ARIMA



Unsatisfactory Results

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Apr 2023	1503714	999849.9	2007579	733120.1	2274309
May 2023	1503714	791142.4	2216286	413929.5	2593499
Jun 2023	1503714	630995.5	2376433	169006.0	2838423

2. Holt-Winters



1. SO
2. Revenue

1. Monthly
2. Individual Customer

```
> WAPE <- relativeabsoluteerror*100
> WAPE
[1] 21.71447
```

2. Holt-Winters

1. SO
2. Revenue

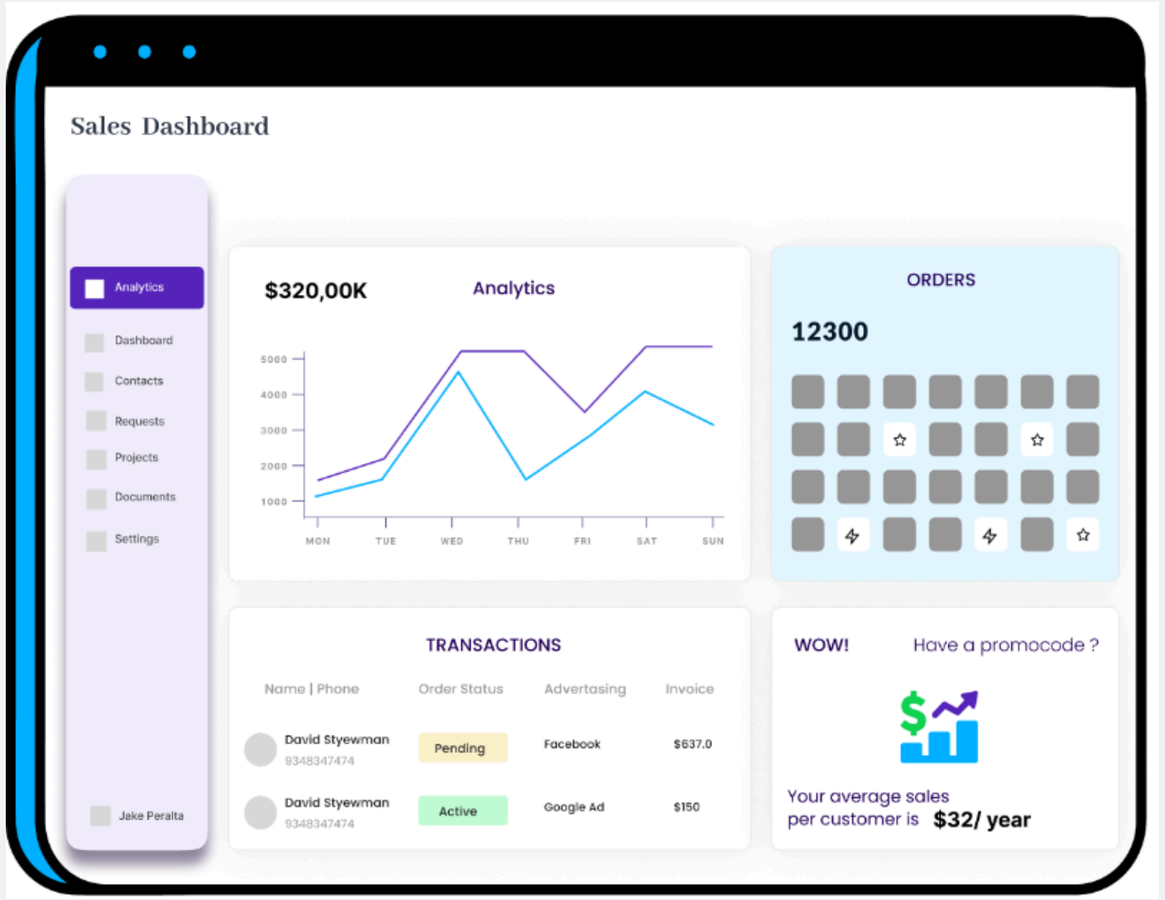
1. Monthly
2. Individual Customer

```
"Date","Forecast","Lo_95","Hi_95"
2023.25,1626806.17408082,662533.006316403,2591079.34184523
2023.33333333333,1960542.66536194,531712.001850309,3389373.32887356
2023.41666666667,2294279.15664306,202303.839868991,4386254.47341712
```

DASHBOARD

- Manually keying
- Copy Pasting
- Manually calculation

- Additional forecasting model
- One stop solution

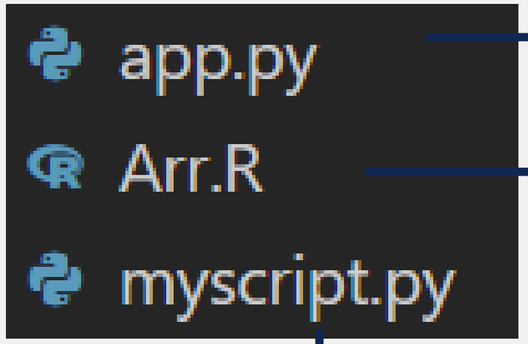


DATA VISUALIZATION

Our Stack (Web app)

- Highcharts Library
- HTML, CSS, Javascript
- Python for Backend Logic
  - Pandas module
  - Flask app framework

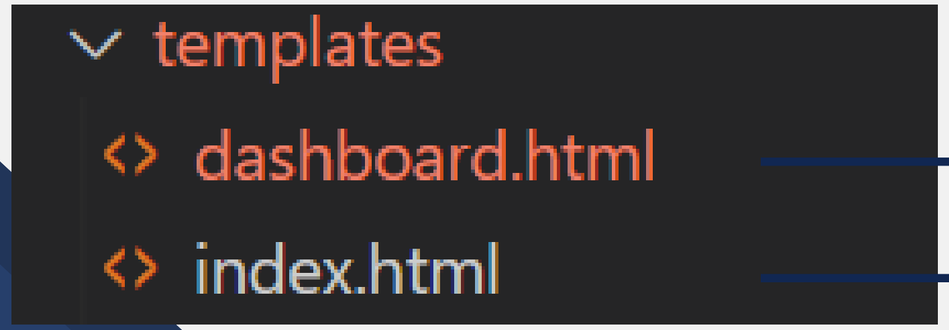
WEB APP FILE AND FOLDER STRUCTURE



app.py is the main entry point and will contain our server-side code eg. routes that run the Forecasting R script

Arr.R contains the forecasting code

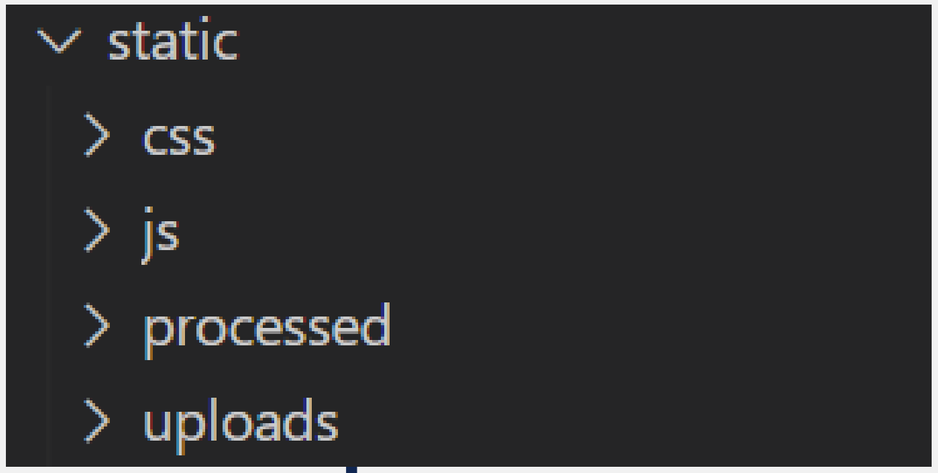
myscript.py data processing and cleaning code



dashboard.html (dashboard webpage)

index.html (the initial webpage for client to upload the raw data

WEB APP FILE AND FOLDER STRUCTURE

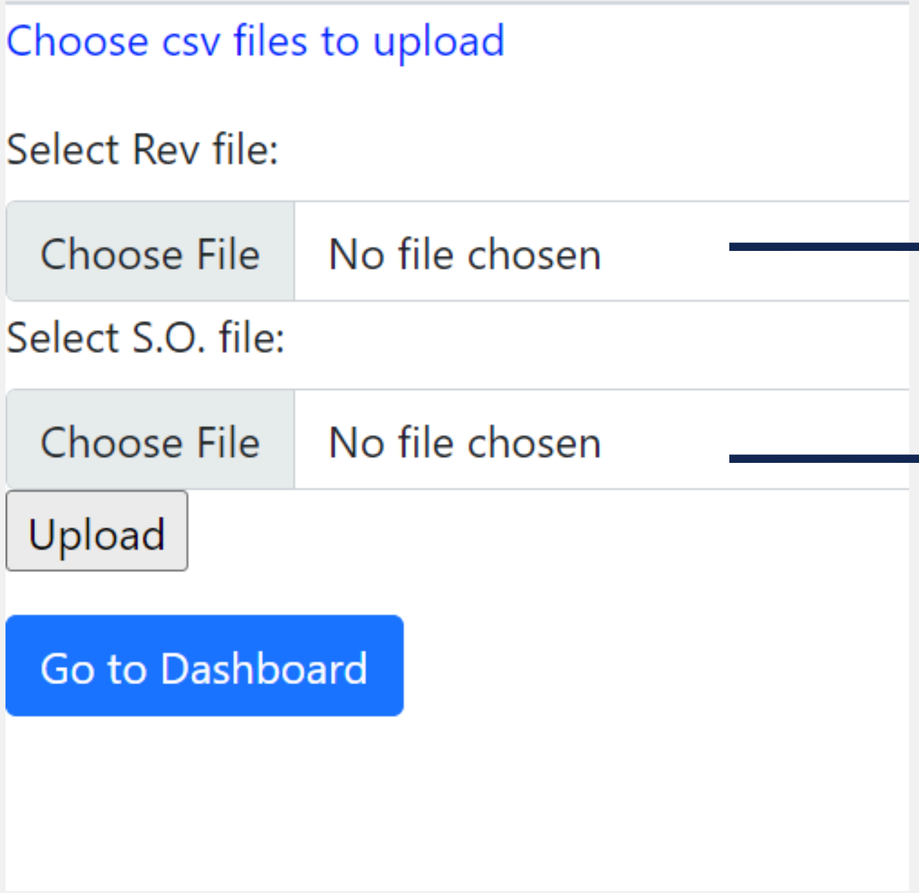


Flask
web development,
one drop at a time

css (contains the styling code files)
js (contains the .js file for Highcharts charting)
processed (where the .csv processed files go)
uploads (where your uploaded csv files go)

FINAL PRODUCT

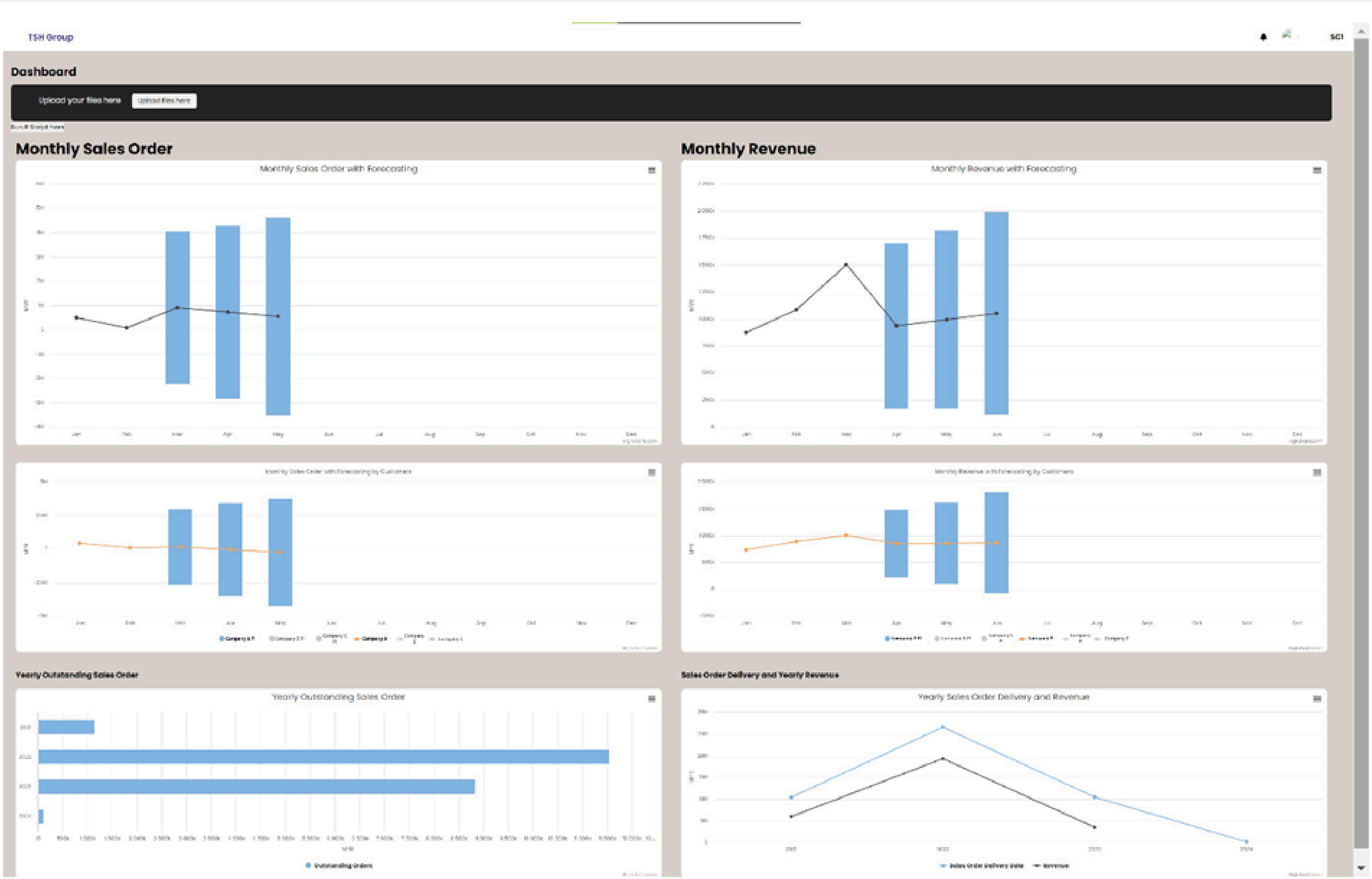
DASHBOARD INTERFACE



Upload Rev file here

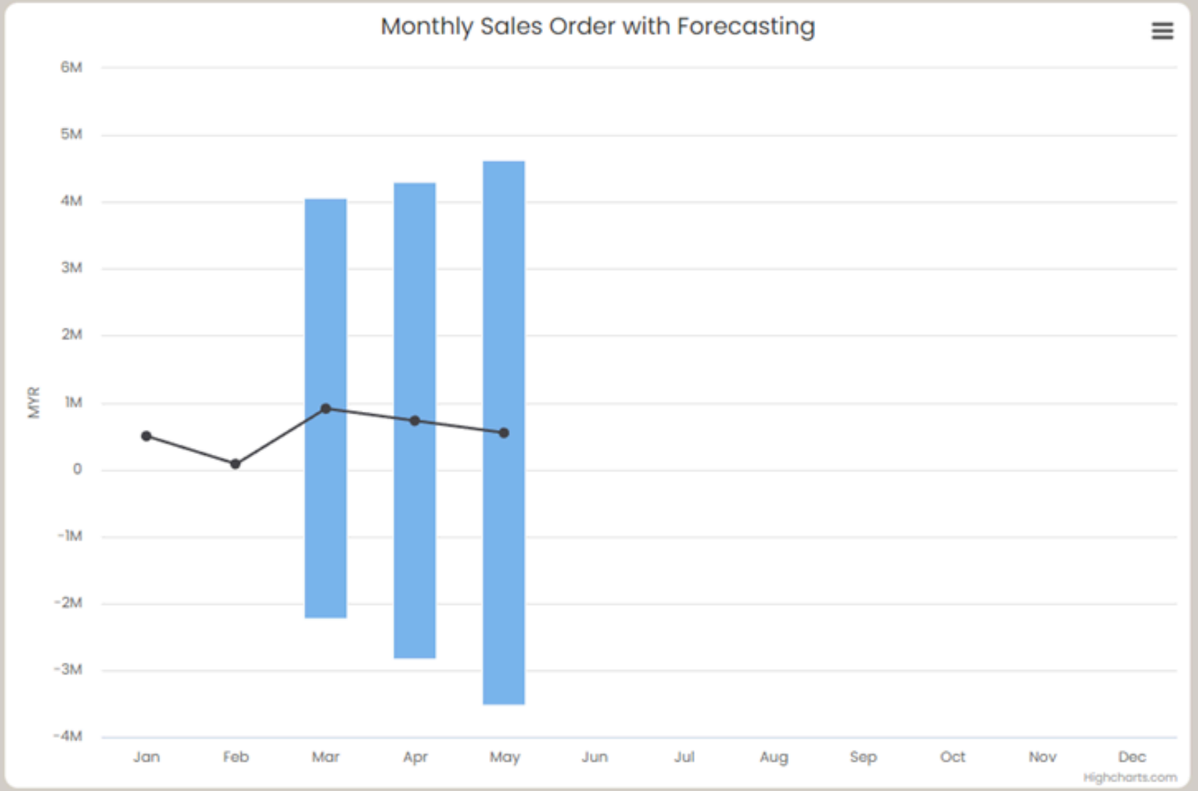
Upload S.O. file here

DASHBOARD INTERFACE

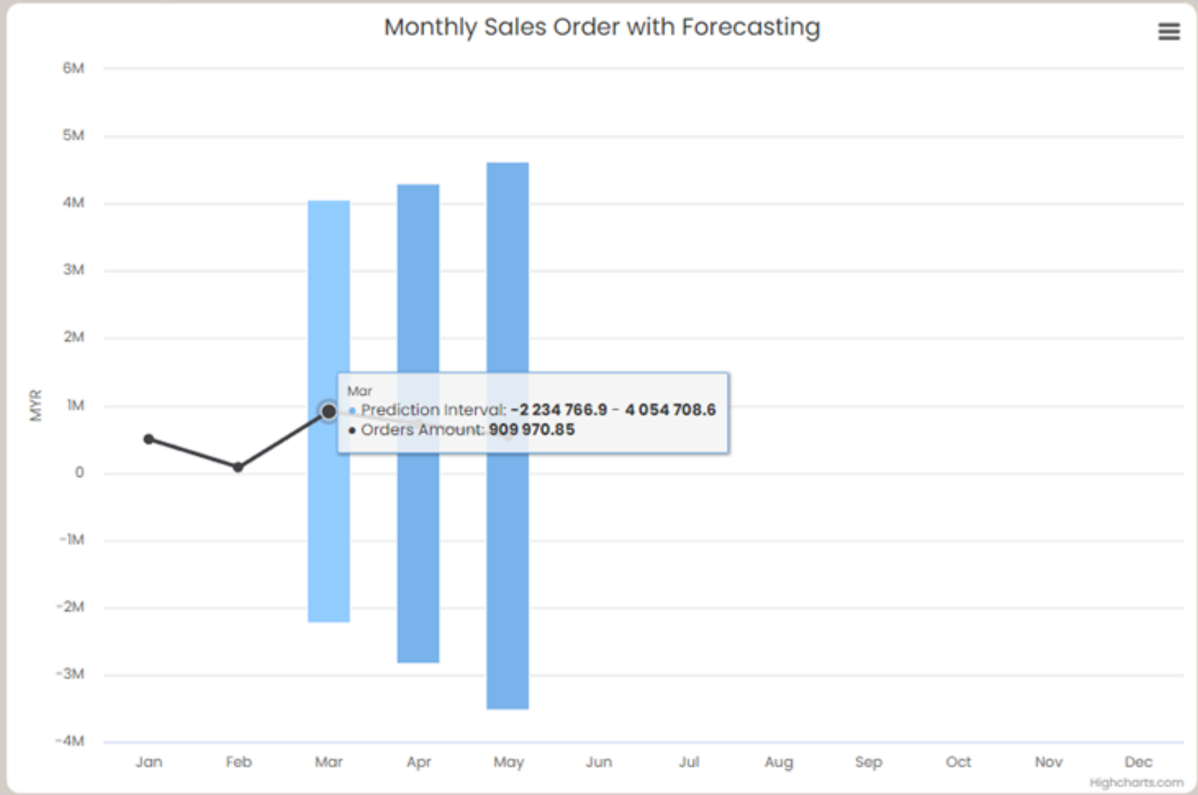


DASHBOARD INTERFACE

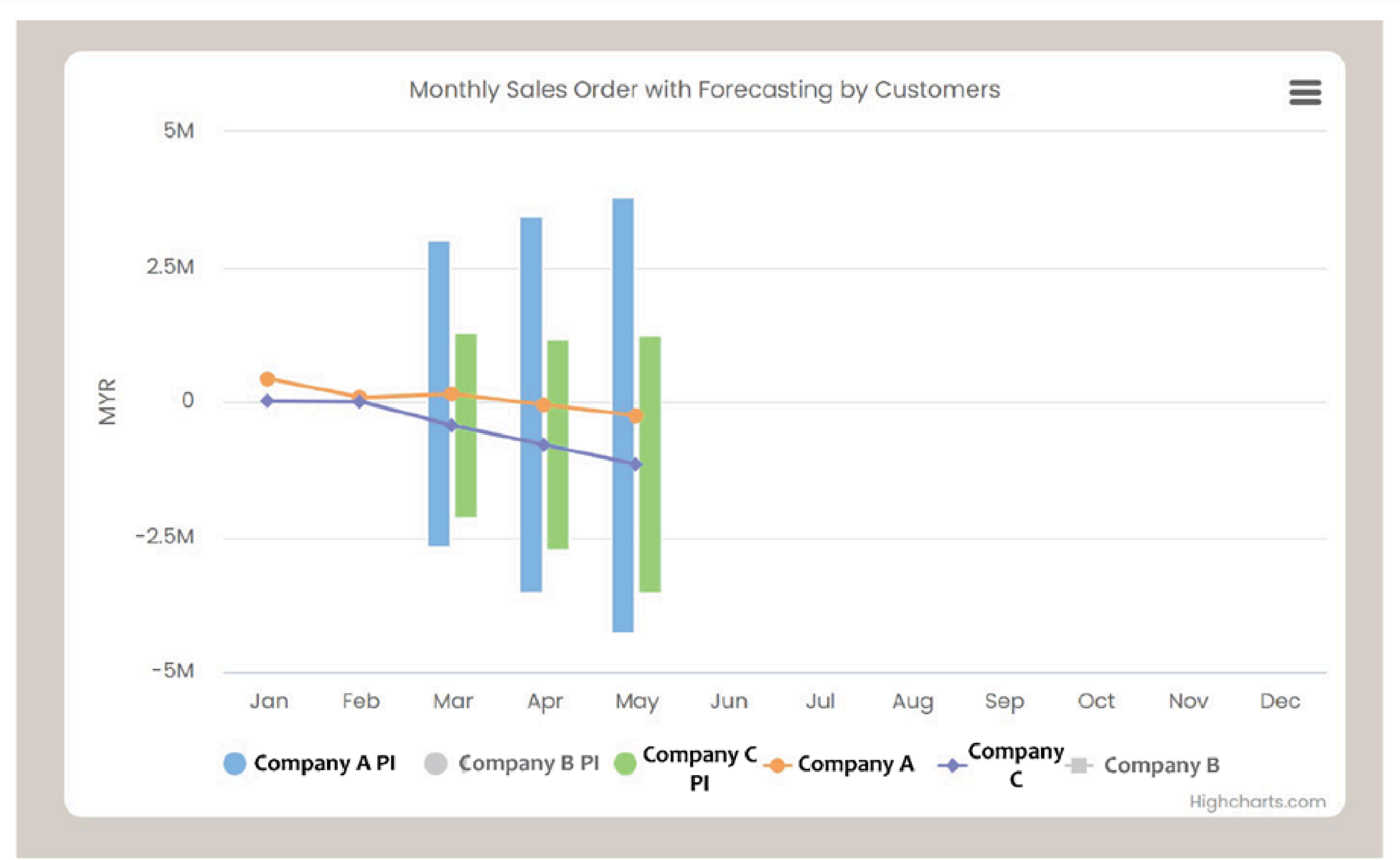
Monthly Sales Order



Monthly Sales Order

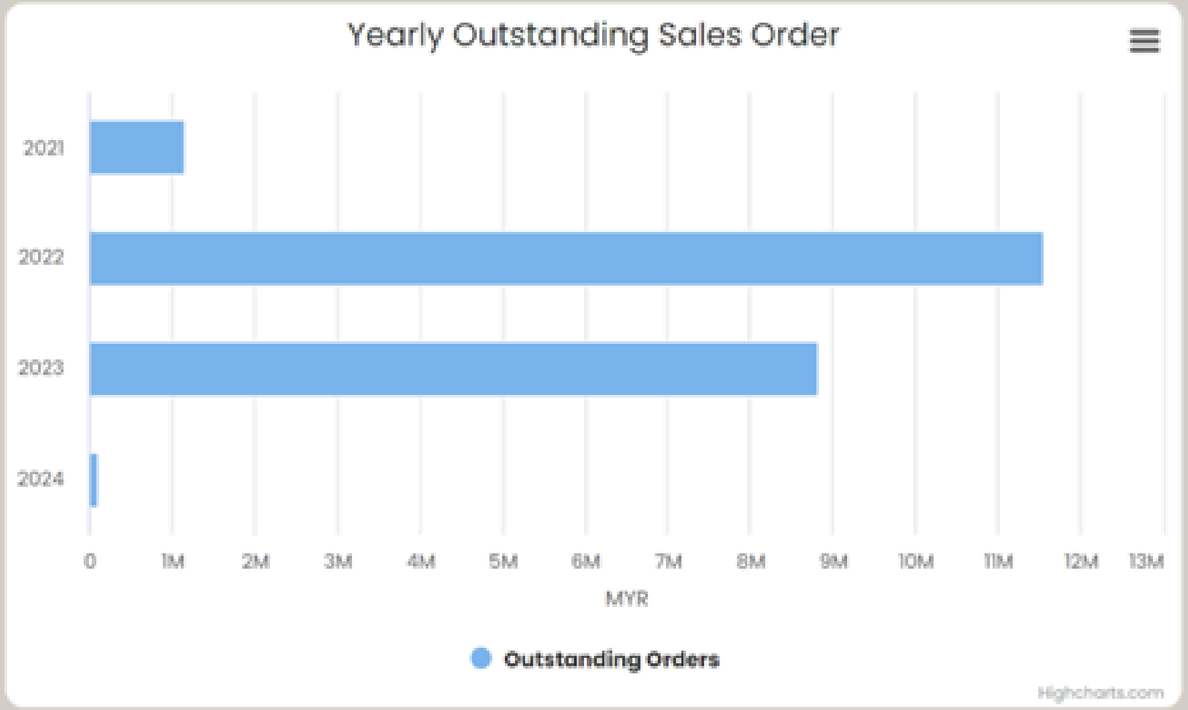


DASHBOARD INTERFACE

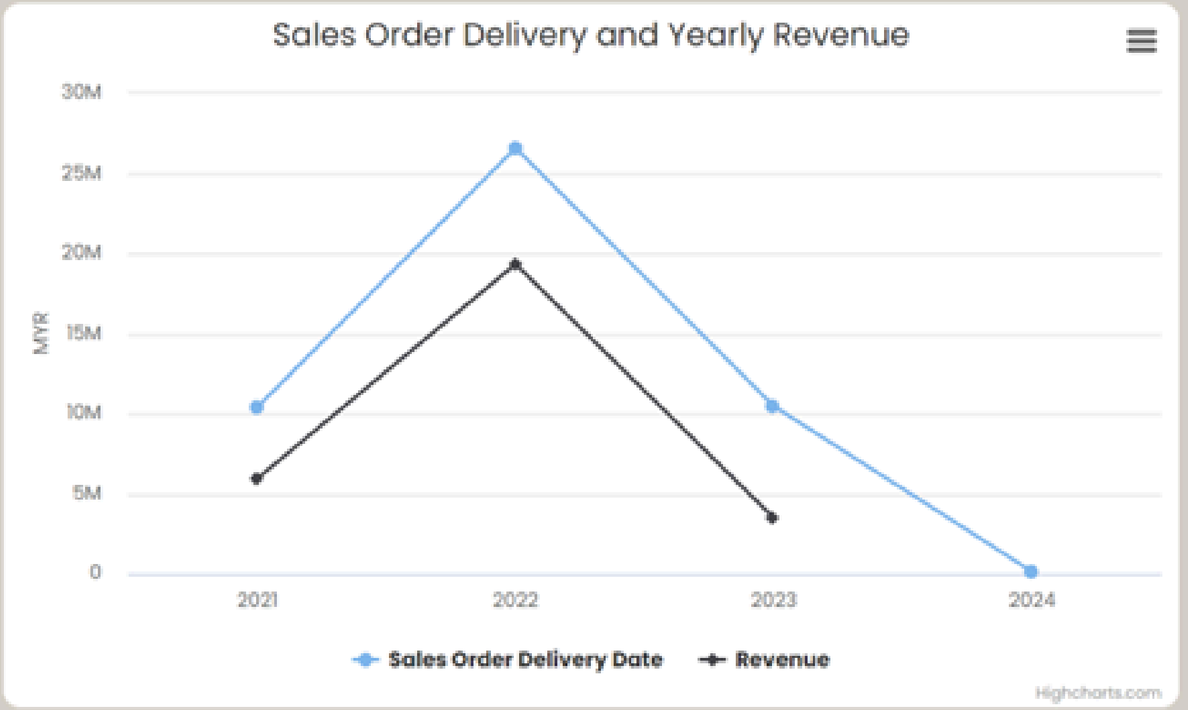


DASHBOARD INTERFACE

Yearly Outstanding Sales Order



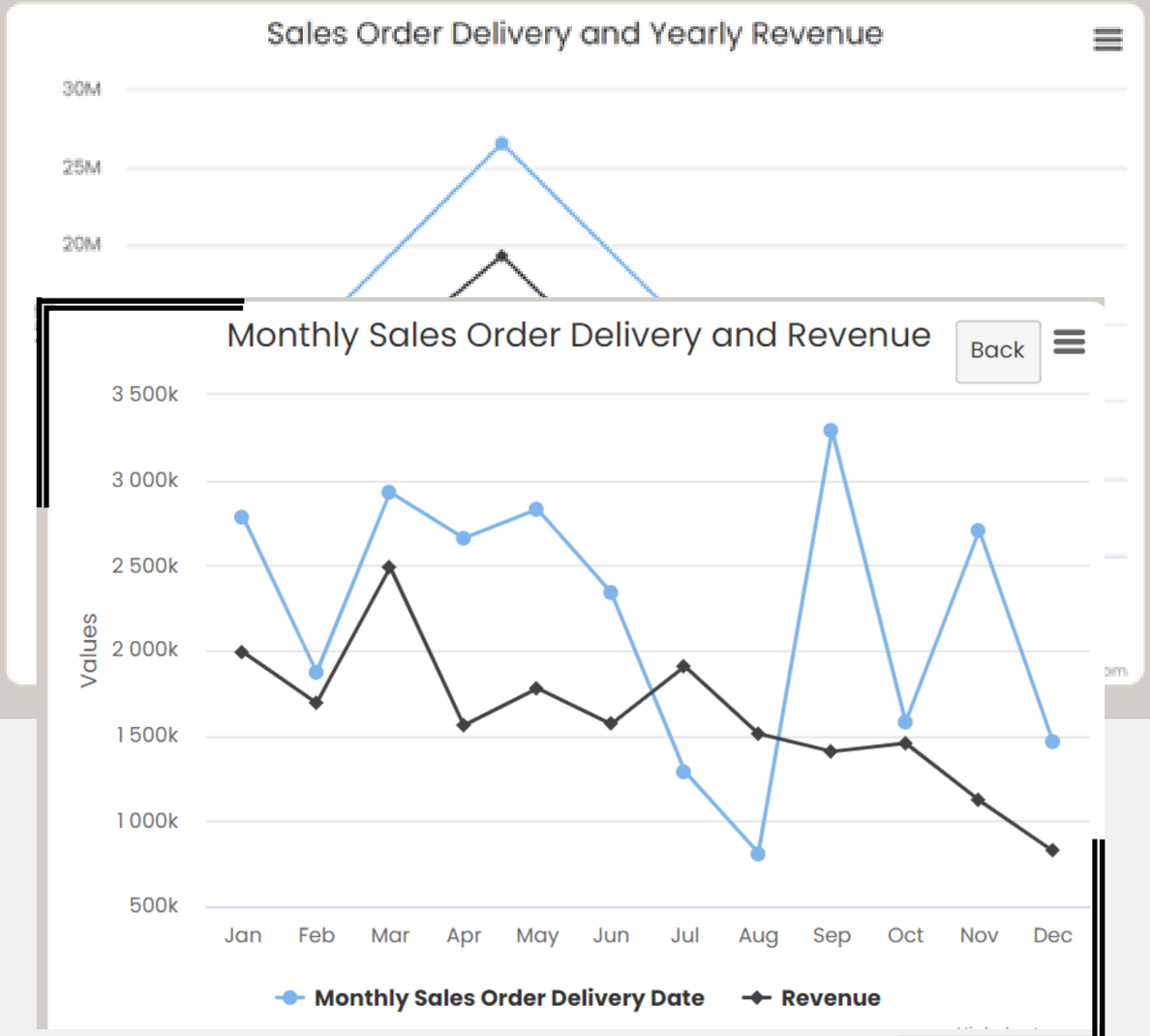
Sales Order Delivery and Yearly Revenue



DASHBOARD INTERFACE

Yearly Outstanding Sales Order

Sales Order Delivery and Yearly Revenue



DOCUMENTATION GUIDE

Dashboard

Using Flask Web Development in Python and High-charts

Run Flask

Open the command prompt
Type the following into the command prompt:
cd "C:\Users\your\path\to\folder\here"
venv\scripts\activate
python app.py

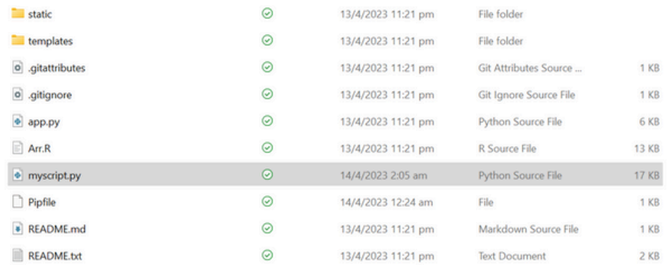
On the upload page, upload your csv file for INV and SO which has data for all available dates. This is so if past data have been updated, the dashboard can capture this change.

Click the button to go to dashboard and click the button to Run R Script. Once this is done, forecasted data will be generated and graphs can be seen.

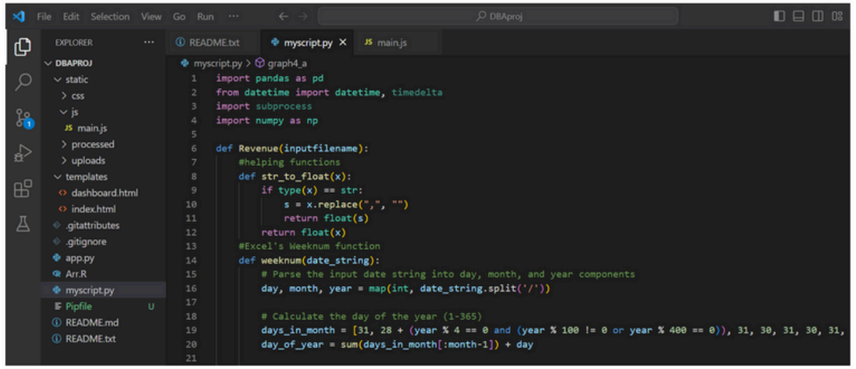
Display Charts (Any changes that can be made)

1. Changing or editing the drilldown customers graph, for instance, adding another customer to the graph.

Locate myscript.py file in the DBAProj Folder



Open myscript.py file using visual studio code, the opened file should look like below



Opening the file, you can then locate the function called graph1_a and graph2_a as seen below

- *myscript.py (the highlighted are codes that can be edited)*

DOCUMENTATION GUIDE

Forecast method used:

- Double exponential smoothing (Holt-Winters' seasonal method)
  - Time series forecasting

```
# Holt-Winters ######################################
# Estimate value of alpha parameter
ccdata_forecast <- HoltWinters(ccdata_ts, gamma=FALSE)
```

  - Uses **alpha** and **beta** smoothing factors

```
Smoothing parameters:
 alpha: 0.794479
 beta : 0.2635335
 gamma: FALSE
```

  - Considers **trend** and **seasonality** of the data

Advantages of the forecast

1. Able to smooth out any unexpected fluctuations in dataset
2. Forecast **accuracy will increase** as newer data becomes available
3. Model gives **priority to most recent points** of dataset to be accurately predict future values
4. Model can be **easily adjusted** to incorporate other factors in the future (eg. Recession etc)
5. Ensure that there are no gaps in the time series data and forecast will run with some missing data in between

a.
```
# Assign 0 to missing days
ccdata <- ccdata %>% complete(SumDate = seq.Date(min(SumDate), max(SumDate), by = "month"), fill = list(Revenue = 0))
```

Events the forecast will fail

1. Forecast will not be able to capture sudden shocks (e.g. Natural disaster, pandemic)
2. In the event of the column name being entered wrongly (capitalisation/spacing), the forecast will fail as the program will not be able to recognise the column name and retrieve the required data.
3. In the event the customer does not order in an upcoming month, this means there will be no data indicating this information. Thus, causing an inaccuracy in the forecast. (e.g. customer A made zero orders in May, the forecast will not be able to detect the absence of revenue in May and will still make a forecast for revenue in the next three months (May, June, and July) based on incomplete data. Hence, in this scenario, the model will fail to accurately predict the future revenue.

Code Guide

Installation Guide

1. Download R from the official website accordingly to your operating system
   a. Windows: https://cran.r-project.org/bin/windows/base/
   b. macOS: https://cran.r-project.org/bin/macosx/
2. Download RStudio: https://posit.co/download/rstudio-desktop/
3. Run the installer file and follow the instruction accordingly
4. Open the R file via RStudio

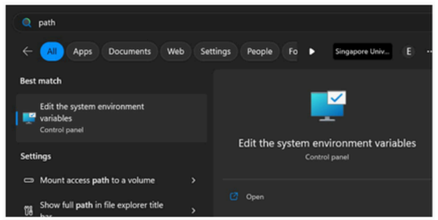
Annotation of Codes

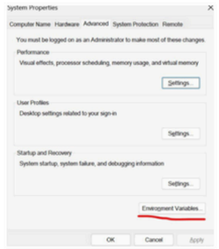
Every line of R Code is annotated line using the "#"
Codes listed below can be edited to adjust the model accordingly to attain different datasets.

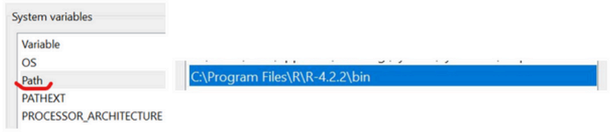
DOCUMENTATION GUIDE

Initial Install Guide

1. **(Python)** Download from official python website https://www.python.org/downloads/ Recommended latest version.
2. **(R)** Download from this website https://cran.r-project.org/. Select "*Download R for Windows*" if you are using a Windows computer. After installation, Ensure that the R\bin folder is included in your system environment PATH

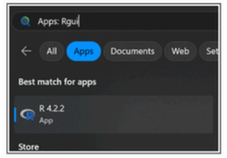




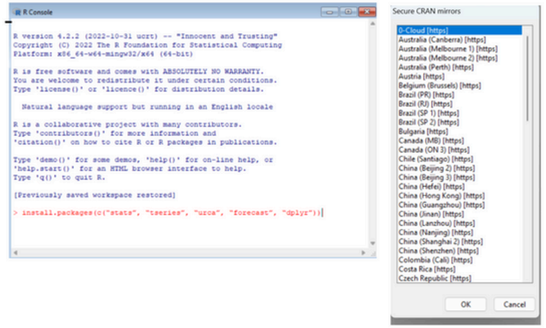


How To Manually Install R packages(if there are permission issues when running the R script)

- Open the R app as shown

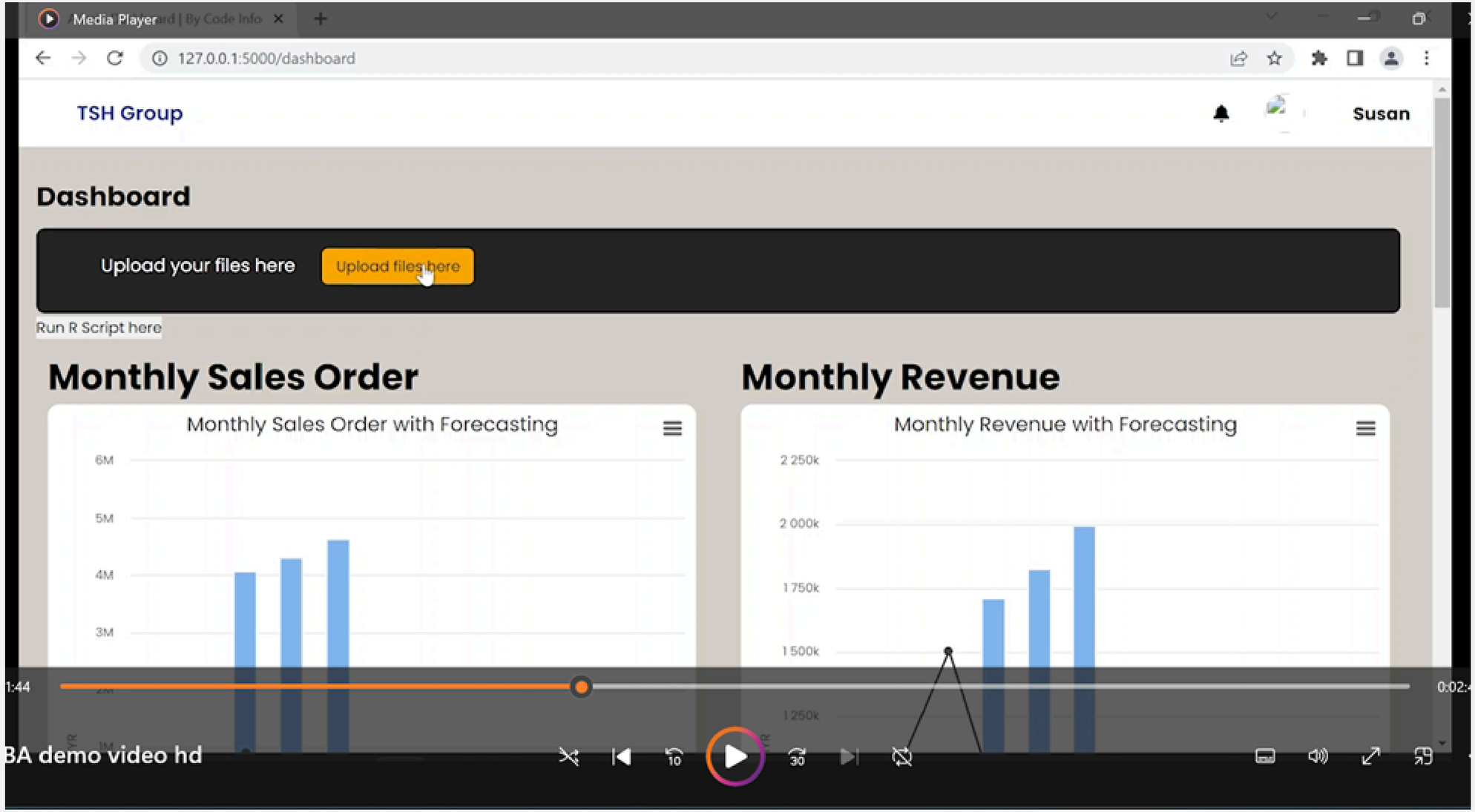


- type install.packages(c("xts", "stats", "tseries", "urca", "forecast", "dplyr", "tidyr")) , press "Enter key" and then click "ok"



3. Open the command prompt from your computer

DOCUMENTATION GUIDE



ASSUMPTIONS

	SO Date	Revised Request Date	SO No.	Project	Quantity	Unit Price	Exchange
0	4/10/2021	4/10/2021	*******	*******	9		3.077
1	6/10/2021	6/10/2021	*******	*******	3		3.077
2	6/10/2021	6/10/2021	*******	*******	1		3.077
3	6/10/2021	6/10/2021	*******	*******	1		3.077
4	6/10/2021	6/10/2021	*******	*******	1		3.077
5	6/10/2021	6/10/2021	*******	*******	1		3.077
6	6/10/2021	6/10/2021	*******	*******	15		3.077
7	6/10/2021	6/10/2021	*******	*******	4		3.077
8	6/10/2021	6/10/2021	*******	*******	2		3.077
9	6/10/2021	6/10/2021	*******	*******	5		3.077
10	6/10/2021	6/10/2021	*******	*******	3		3.077
11	6/10/2021	31/5/2022	*******	*******	3		3.077
12	6/10/2021	6/10/2021	*******	*******	5		3.077
13	6/10/2021	6/10/2021	*******	*******	4		3.077
14	6/10/2021	6/10/2021	*******	*******	1		3.077
15	6/10/2021	6/10/2021	*******	*******	1		3.077
16	6/10/2021	6/10/2021	*******	*******	1		3.077
17	6/10/2021	6/10/2021	*******	*******	15		3.077
18	6/10/2021	6/10/2021	*******	*******	15		3.077
19	7/10/2021	7/10/2021	*******	*******	2		3.077
20	7/10/2021	7/10/2021	*******	*******	1		3.077
21	7/10/2021	7/10/2021	*******	*******	3		3.077

ASSUMPTIONS

	SO Date	Revised Request Date	SO No.	Project	Quantity	Unit Price	Exchange
0	4/10/2021	4/10/2021	*******	*******	9		3.077
1	6/10/2021	6/10/2021	*******	*******	3		3.077
2	6/10/2021	6/10/2021	*******	*******	1		3.077
3	6/10/2021	6/10/2021	*******	*******	1		3.077
4	6/10/2021	6/10/2021	*******	*******	1		3.077
5	6/10/2021	6/10/2021	*******	*******	1		3.077
6	6/10/2021	6/10/2021	*******	*******	15		3.077
7	6/10/2021	6/10/2021	*******	*******	4		3.077
8	6/10/2021	6/10/2021	*******	*******	2		3.077
9	6/10/2021	6/10/2021	*******	*******	5		3.077
10	6/10/2021	6/10/2021	*******	*******	3		3.077
14	6/10/2021	6/10/2021	*******	*******	1		3.077
15	6/10/2021	6/10/2021	*******	*******	1		3.077
16	6/10/2021	6/10/2021	*******	*******	1		3.077
17	6/10/2021	6/10/2021	*******	*******	15		3.077
18	6/10/2021	6/10/2021	*******	*******	15		3.077
19	7/10/2021	7/10/2021	*******	*******	2		3.077
20	7/10/2021	7/10/2021	*******	*******	1		3.077
21	7/10/2021	7/10/2021	*******	*******	3		3.077

```
#select columns
col = ['SO Date', 'Revised Request Date', 'SO No.', 'Project', 'Quantity', 'Unit Price', 'Exchange Rate', 'Non-ShipQTY']
df_so = df2[col]
```

LIMITATIONS

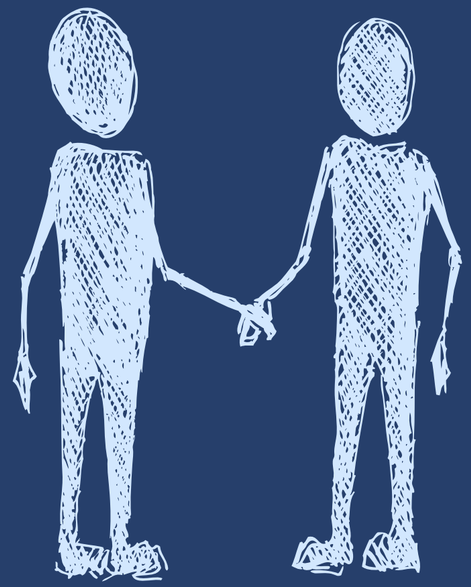
- 13 aggregated data points for forecasting
- Unable to predict sudden shocks or events (eg: worldwide pandemic)
- Inaccuracy in forecasted values if customers have not ordered in recent months

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov
2021											2144414.8
2022	1990914.4	1691708.1	2489592.8	1561086.1	1778049.4	1568544.0	1907662.9	1511695.1	1407834.4	1456025.4	1120936.1
2023	875168.5	1086235.5	1503714.4								

	Dec
2021	2261812.8
2022	823897.8

FUTURE RECOMMENDATIONS

- Improve forecasting model
- Weekly drill-down for more detailed data
- More aesthetically appealling dashboard interface



THANK YOU