



# PRESENTATION ON

## EDA ON PETROLEUM DATASET

PRESENTED BY  
B PRANAVI  
G RAJU  
B VARSHITH

# INTRODUCTION

- Exploratory Data Analysis (EDA) is used to understand refinery and petroleum production data.
- It helps analyze crude processing, product output, efficiency, and operational trends.
- Statistical summaries and visualizations reveal patterns, variations, and anomalies in refinery operations.
- EDA supports better decision-making by improving data quality and operational insights

# OBJECTIVE

- Identify key patterns and trends.
- Understand relationships between variables.
- Detect anomalies or data quality issues.
- Discover meaningful insights that highlight biases, imbalances, and underlying patterns within the dataset.

# DATA SET

	Work shift	Type	Refinery	Origin Departure Date	Origin Net Weight	Destination Arrival Date	Destination Net Weight
0	Day shift	VB	Tehran	1398/12/28	24040	1399/01/01	24030
1	Day shift	VB	Tehran	1398/12/28	23720	1399/01/01	23700
2	Day shift	VB	Tehran	1398/12/27	26080	1399/01/01	25980
3	Day shift	VB	Tehran	1398/12/28	23260	1399/01/01	23700
4	Day shift	VB	Tehran	1398/12/29	26380	1399/01/02	26300
...	...	...	...	...	...	...	...
260	Evening shift	Condensate	South Pars	1399/03/19	22696	1399/03/20	22670
261	Evening shift	Condensate	South Pars	1399/03/19	22692	1399/03/20	22690
262	Evening shift	Condensate	South Pars	1399/03/19	22748	1399/03/20	22670
263	Evening shift	Condensate	South Pars	1399/03/19	20574	1399/03/20	20430
264	Evening shift	Condensate	South Pars	1399/03/20	22665	1399/03/20	22550

265 rows × 7 columns

# KEY COLUMNS OVERVIEW

- **Type** – The type of petroleum product or material (e.g., VB, Condensate).
- **Refinery** – The refinery location where the petroleum product was processed or dispatched.
- **Origin Departure Date** – The date on which the petroleum product was dispatched from the origin refinery.
- **Work Shift** – The operational shift during which the refinery activity occurred (Day / Evening).
- **Origin Net Weight** – The net weight of the petroleum product at the origin point.
- **Destination Arrival Date** – The date on which the petroleum product arrived at the destination.
- **Destination Net Weight** – The net weight of the petroleum product recorded at the destination.

# DATA UNDERSTANDING

```
df.shape
```

```
(265, 7)
```

→ df.shape gives the total number of rows and columns in the dataset, helping us understand its overall size.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265 entries, 0 to 264
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Work shift       265 non-null    object  
 1   Type             265 non-null    object  
 2   Refinery         265 non-null    object  
 3   Origin Departure Date  265 non-null  object  
 4   Origin Net Weight 265 non-null    int64  
 5   Destination Arrival Date 265 non-null  object  
 6   Destination Net Weight 265 non-null    int64  
dtypes: int64(2), object(5)
memory usage: 14.6+ KB
```

→ df.info() provides the structure of the dataset, including data types and missing values, which is essential for further cleaning.

```
df.describe()
```

	Origin Departure Date	Origin Net Weight	Destination Arrival Date	Destination Net Weight
<b>count</b>	265	265.000000	265	265.000000
<b>mean</b>	2020-05-02 04:26:15.849056512	24305.852830	2020-05-03 10:03:10.188679168	24296.981132
<b>min</b>	2020-03-17 00:00:00	15050.000000	2020-03-20 00:00:00	15110.000000
<b>25%</b>	2020-04-13 00:00:00	23300.000000	2020-04-14 00:00:00	23410.000000
<b>50%</b>	2020-04-25 00:00:00	24380.000000	2020-04-26 00:00:00	24340.000000
<b>75%</b>	2020-05-30 00:00:00	25250.000000	2020-05-31 00:00:00	25230.000000
<b>max</b>	2020-06-09 00:00:00	28770.000000	2020-06-09 00:00:00	28720.000000
<b>std</b>	Nan	1532.788422	Nan	1533.901721

```
df.describe(include='all')
```

	Work shift	Type	Refinery	Origin Departure Date	Origin Net Weight	Destination Arrival Date	Destination Net Weight
<b>count</b>	265	265	265	265	265.000000	265	265.000000
<b>unique</b>	3	4	4	Nan	Nan	Nan	Nan
<b>top</b>	Day shift	VB	Shiraz	Nan	Nan	Nan	Nan
<b>freq</b>	130	206	123	Nan	Nan	Nan	Nan
<b>mean</b>	Nan	Nan	Nan	2020-05-02 04:26:15.849056512	24305.852830	2020-05-03 10:03:10.188679168	24296.981132
<b>min</b>	Nan	Nan	Nan	2020-03-17 00:00:00	15050.000000	2020-03-20 00:00:00	15110.000000
<b>25%</b>	Nan	Nan	Nan	2020-04-13 00:00:00	23300.000000	2020-04-14 00:00:00	23410.000000
<b>50%</b>	Nan	Nan	Nan	2020-04-25 00:00:00	24380.000000	2020-04-26 00:00:00	24340.000000
<b>75%</b>	Nan	Nan	Nan	2020-05-30 00:00:00	25250.000000	2020-05-31 00:00:00	25230.000000
<b>max</b>	Nan	Nan	Nan	2020-06-09 00:00:00	28770.000000	2020-06-09 00:00:00	28720.000000
<b>std</b>	Nan	Nan	Nan	Nan	1532.788422	Nan	1533.901721

This function provides summary statistics for all numerical columns, such as mean, minimum, maximum, and standard deviation.

This command generates summary statistics for categorical columns, showing the number of unique values, the most frequent category, and its frequency.

# DATA CLEANING

```
df.isnull().sum()
```

```
Work shift          0  
Type               0  
Refinery           0  
Origin Departure Date 0  
Origin Net Weight  0  
Destination Arrival Date 0  
Destination Net Weight 0  
dtype: int64
```

→ The dataset contains no missing values across any of the columns, indicating high data completeness and quality. As a result, no null-value treatment is required before performing further analysis or modeling.

```
df.duplicated().sum()
```

```
np.int64(1)
```

→ The dataset contains one duplicate record, indicating a small level of data redundancy. This duplicate should be reviewed and removed if necessary to ensure accurate analysis and reliable results.

# HANDLING DUPLICATE VALUES

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

- The duplicate record has been successfully removed from the dataset. Now, there are no duplicate rows, and the data is clean for further analysis.

# DATA TYPE CORRECTION

```
import jdatetime
import pandas as pd

def jalali_to_gregorian(date_str):
    date_str = str(date_str).strip()
    y, m, d = map(int, date_str.split('/'))
    g = jdatetime.date(y, m, d).togregorian()
    return pd.to_datetime(g)

df['Origin Departure Date'] = df['Origin Departure Date'].apply(jalali_to_gregorian)

df['Destination Arrival Date'] = df['Destination Arrival Date'].apply(jalali_to_gregorian)
```

- The dates were changed from Jalali format to normal Gregorian dates(**the normal calendar dates we use every day**, like **YYYY-MM-DD** (for example, 2024-03-15)). This helps the data work properly for date calculations and analysis.

# UNIVARIATE ANALYSIS

(non-visualization)

```
df['Work shift'].value_counts()
```

[17]:

```
Work shift
Day shift      130
Night shift    95
Evening shift  39
Name: count, dtype: int64
```

Most records are from the Day shift (130), followed by the Night shift (95), while the Evening shift (39) has the least number of entries.

```
df['Type'].unique()
```

```
array(['VB', 'C5+', 'Condensate', 'Iso-Recycle'], dtype=object)
```

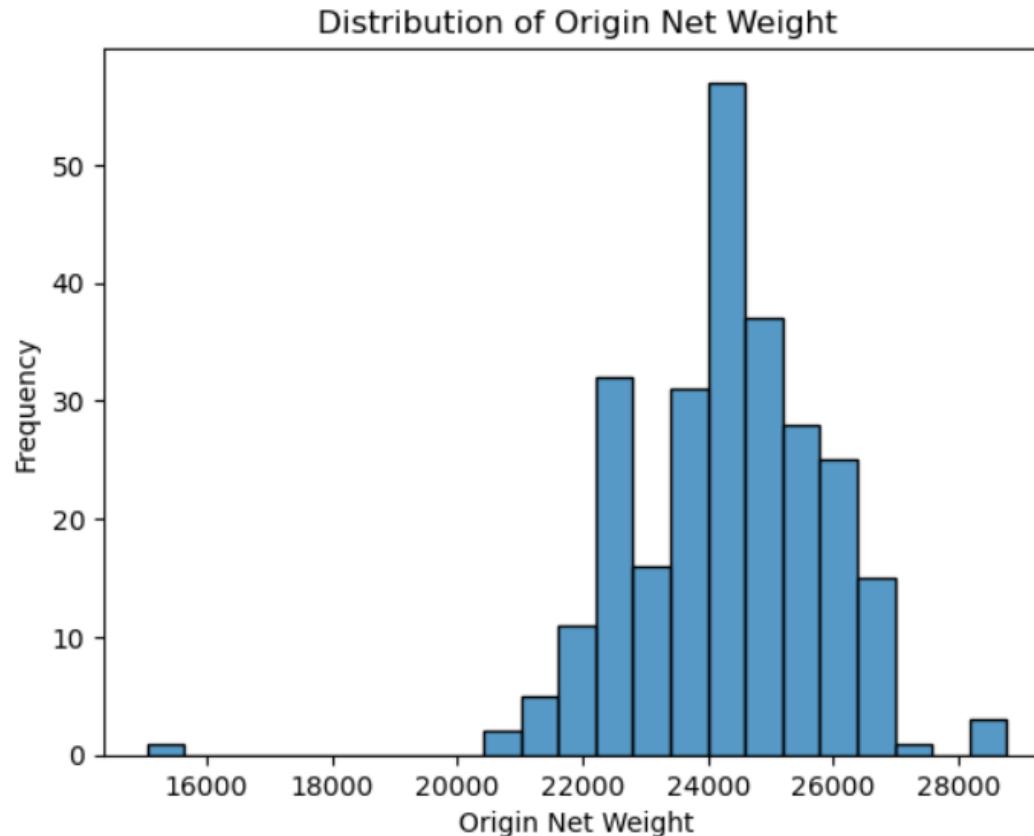
The Type column contains four unique categories: VB, C5+, Condensate, and Iso-Recycle. This shows the dataset includes multiple product or material types.

# UNIVARIATE ANALYSIS

(visualization)

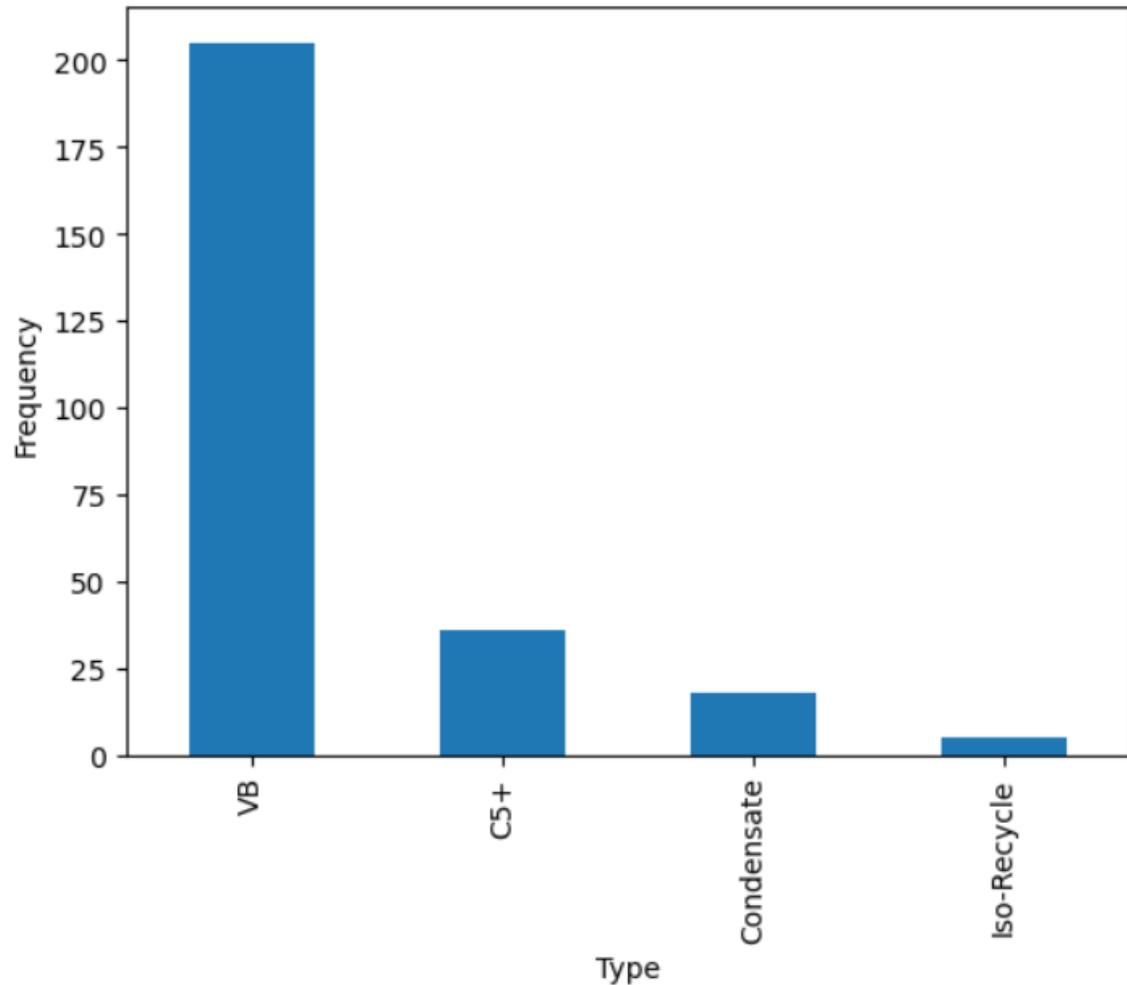
```
#numerical distribution
sns.histplot(df['Origin Net Weight'])
plt.xlabel('Origin Net Weight')
plt.ylabel('Frequency')
plt.title('Distribution of Origin Net Weight')
plt.show()
```

The Origin Net Weight values are mostly clustered between 22,000 and 26,000, showing a common weight range. A few very low and very high values appear, which may be outliers.



```
# categorical distribution
df['Type'].value_counts().plot(kind='bar')
plt.xlabel('Type') # Title for the x-axis
plt.ylabel('Frequency')
plt.show()
```

The dataset is highly dominated by the VB category, with significantly higher frequency than all other types. C5+, Condensate, and Iso-Recycle appear in much smaller proportions, indicating a strongly imbalanced categorical distribution..



# BIVARIATE ANALYSIS

(non-visualization)

```
#numerical-numerical
```

```
df[['Origin Net Weight', 'Destination Net Weight']].corr()
```

	Origin Net Weight	Destination Net Weight
Origin Net Weight	1.0000	0.9984
Destination Net Weight	0.9984	1.0000

```
#categorical-categorical
```

```
pd.crosstab(df['Refinery'], df['Work shift'])
```

	Work shift	Day shift	Evening shift	Night shift
Refinery				
Pars Petrochemical	12	12	12	
Shiraz	59	4	59	
South Pars	2	9	7	
Tehran	57	14	17	

Origin Net Weight and Destination Net Weight show a very strong positive correlation ( $\approx 0.998$ ), indicating that destination weight closely matches the origin weight with minimal variation.

The work shift distribution varies by refinery, with Shiraz and Tehran showing higher activity, while South Pars has comparatively fewer operations across shifts.

```
#numerical-categorical  
df.groupby('Refinery')['Destination Net Weight'].mean()
```

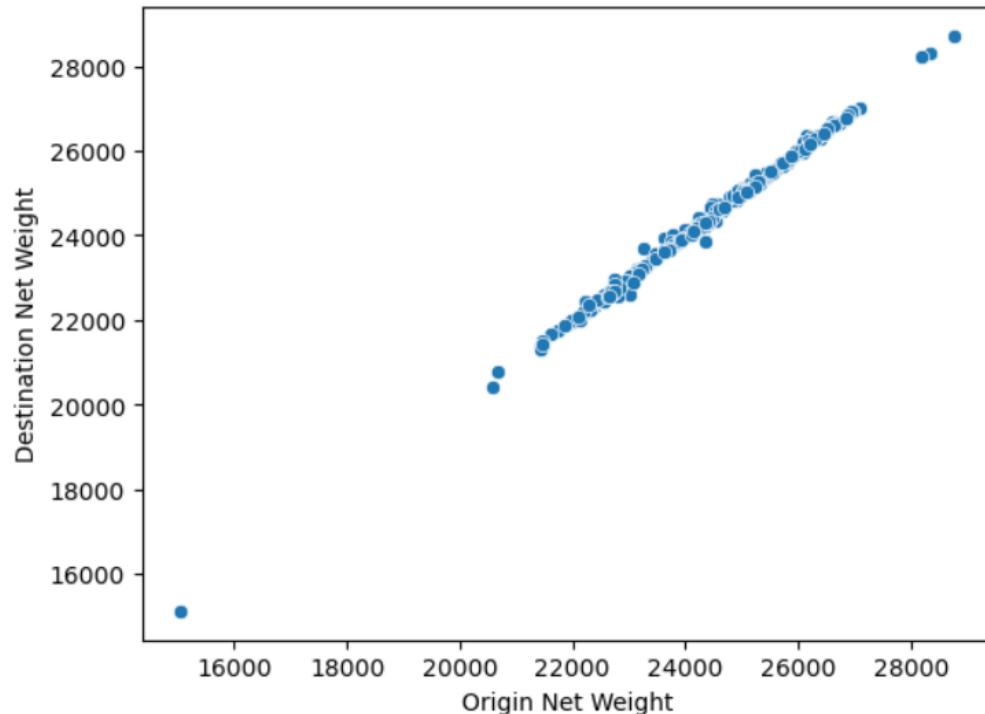
```
Refinery  
Pars Petrochemical    23202.222222  
Shiraz                24420.819672  
South Pars             22760.000000  
Tehran                24885.795455  
Name: Destination Net Weight, dtype: float64
```

The average Destination Net Weight is highest for the Tehran refinery, followed closely by Shiraz. South Pars and Pars Petrochemical have comparatively lower average weights, indicating smaller shipment volumes on average.

# BIVARIATE ANALYSIS

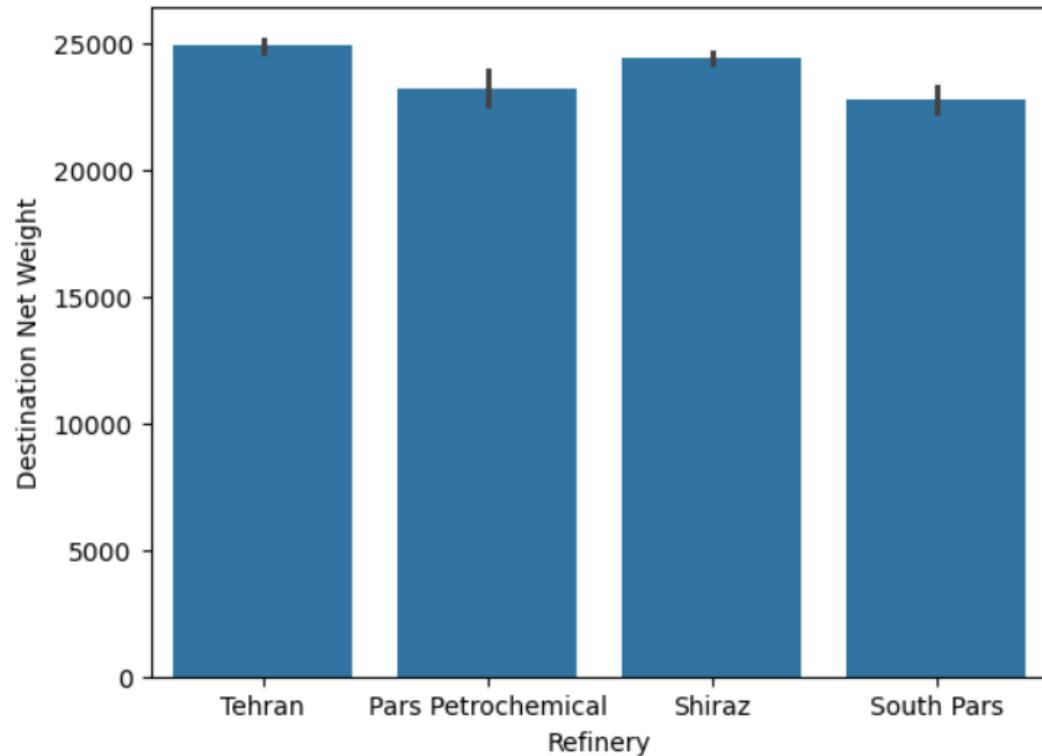
(visualization)

```
#numerical-numerical  
sns.scatterplot(x='Origin Net Weight', y='Destination Net Weight', data=df)  
plt.show()
```



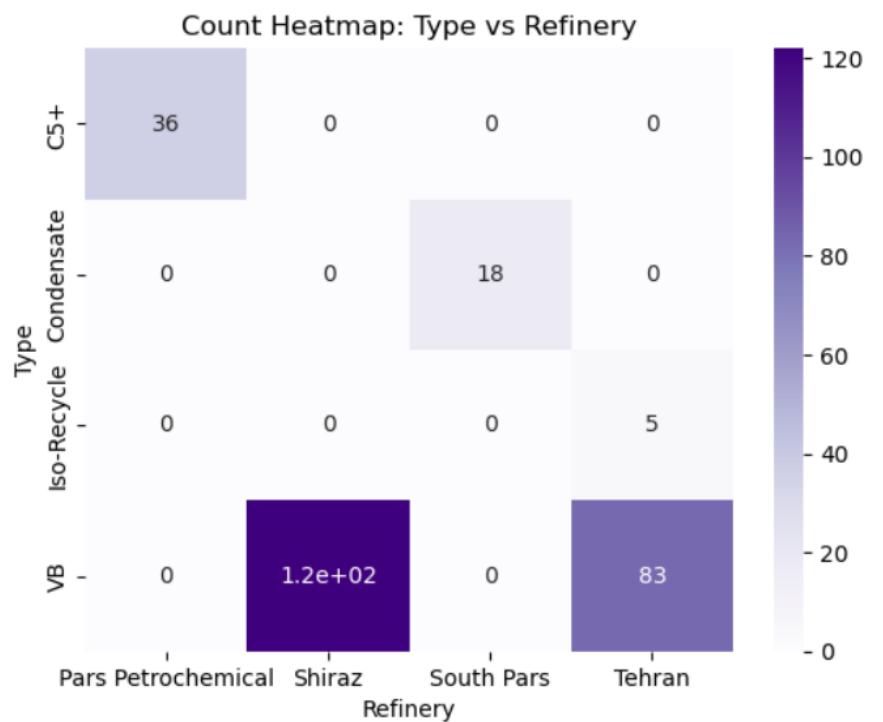
- There is a strong positive linear relationship between Origin Net Weight and Destination Net Weight, showing that destination weight increases almost proportionally with origin weight.
- The points are tightly clustered along a straight line, indicating high consistency with very few outliers.

```
#numerical-categorical  
sns.barplot(x='Refinery', y='Destination Net Weight', data=df)  
plt.show()
```



- The Tehran refinery has the highest average Destination Net Weight, followed closely by Shiraz, indicating larger shipment volumes.
- Pars Petrochemical and South Pars show lower average weights, suggesting comparatively smaller shipments across these refineries.

```
#categorical-categorical
heat = pd.crosstab(df['Type'], df['Refinery'])
sns.heatmap(heat, annot=True, cmap='Purples')
plt.title('Count Heatmap: Type vs Refinery')
plt.xlabel('Refinery')
plt.ylabel('Type')
plt.show()
```



- VB is predominantly processed at Shiraz and Tehran refineries, with Shiraz having the highest count.
- C5+ is handled mainly by Pars Petrochemical, while Condensate is concentrated at South Pars and Iso-Recycle appears only at Tehran, indicating strong specialization by refinery.

# OVERALL INSIGHTS

- The dataset shows a strong consistency between Origin and Destination Net Weights, indicating reliable transfer processes with minimal loss.
- VB is the most dominant product type, mainly processed at Shiraz and Tehran, while other types are refinery-specific, showing operational specialization.
- Tehran and Shiraz refineries handle higher shipment volumes, whereas South Pars and Pars Petrochemical operate at comparatively lower scales.
- The distribution of product types and refinery operations is highly imbalanced, with certain combinations occurring much more frequently.
- Shipment weights are generally stable and predictable, as indicated by tight clustering and limited variability in most refineries.

# CHALLENGES

- Dates were in Jalali format (like 1390) and had to be converted to Gregorian format (like 2020) to perform correct analysis. This required extra preprocessing effort.
- The dataset is dominated by the VB product type, which made comparisons with other product types less balanced and limited insights.
- Origin Net Weight and Destination Net Weight are very highly correlated, which caused redundancy and reduced the amount of new information obtained.
- Some product and refinery combinations have very few records, which restricted deeper and reliable bivariate analysis.

# CONCLUSION

- Overall, the Exploratory Data Analysis helped us clearly understand the structure, quality, and behavior of the petroleum dataset.
- The data was mostly clean, with no missing values and only one duplicate, which was successfully handled, making the dataset reliable for analysis.
- Univariate and bivariate analysis revealed key patterns such as dominant product types, refinery specialization, and consistent shipment weights.
- Despite challenges like date format conversion and category imbalance, the analysis provided meaningful insights that can support better operational and business decisions.

# THANK YOU

