Overview Of PythonFormulas

Used To Explore All The Syntex Using Sales_Date

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Sample Data

```
df = pd.DataFrame(data)
or
df=pd.read_csv("sales_data.csv")
```

BASIC DATA CHECKS

```
df.head()          --first 5 rows
df.tail()          --last 5 rows
df.info()          -- info about dataframe
df.describe()      -- summary stats
df.shape           -- rows, cols
df.columns         -- column names
df.dtypes          -- datatypes
df.isnull().sum()  -- missing values
df.duplicated().sum()  -- check duplicates
df.nunique()       --unique counts per column
```

DATA CLEANING

```
df = df.drop_duplicates()   --remove duplicates
df = df.fillna(0)           --fill missing values with 0
df["Sales_Amount"] = df["Sales_Amount"].astype(float)   -- type conversion
df["Month"] = df["Month"].astype(int)                   --ensure integer
df["Category"] = df["Category"].str.upper()             -- string formatting
df = df.rename(columns={"Sales_Amount":"Sales"})        --rename column
```

EXPLORATORY ANALYSIS

```
df["Sales"].mean()         --average sales
df["Sales"].median()       --median sales
df["Sales"].mode()[0]      --mode
df["Sales"].std()          -- standard deviation
df["Sales"].var()          --variance
df["Sales"].min(), df["Sales"].max()    --min/max
df["Sales"].quantile([0.25,0.5,0.75]) -- quartiles
df.corr()                  -- correlation matrix
```

GROUPING & AGGREGATION

```
df.groupby("Year")["Sales"].sum()        -- yearly sales
df.groupby("Region")["Sales"].mean()     -- avg sales by region
df.groupby("Category")["Sales"].count()   --count by category
df.groupby(["Year","Month"])["Sales"].sum().reset_index()      -- year+month grouping
df.pivot_table(values="Sales", index="Region", columns="Year", aggfunc="sum")    -- pivot
df.groupby("Region").agg({"Sales":["mean","max","min","sum"]})   -- multiple aggregations
```

SORTING

```
df.sort_values("Sales", ascending=False)     -- sort by sales (desc)
df.sort_values(["Year","Month"], ascending=[True,True])   --sort by multiple columns
df.sort_index()                              -- sort by index
df.nlargest(5, "Sales")                      -- top 5 sales
df.nsmallest(5, "Sales")                     -- lowest 5 sales
```

TIME SERIES

```
df["Cumulative_Sales"] = df["Sales"].cumsum()        -- cumulative sum
df["Rolling_MA3"] = df["Sales"].rolling(window=3).mean()     -- 3-month moving average
df["Rolling_MA6"] = df["Sales"].rolling(window=6).mean()    -- 6-month moving average
df["EWMA"] = df["Sales"].ewm(span=3, adjust=False).mean()   -- exponential weighted MA
df["Prev_Sales"] = df["Sales"].shift(1)              -- previous month sales
df["Growth_Rate"] = ((df["Sales"]-df["Prev_Sales"])/df["Prev_Sales"])*100  -- growth rate
```

SEGMENTATION

```
df["Quartile"] = pd.qcut(df["Sales"], 4, labels=["Q1","Q2","Q3","Q4"])  -- quartile segment
df["Monthly_Percent"] = (df["Sales"]/df["Sales"].sum())*100          -- % contribution
df["High_Sales"] = np.where(df["Sales"]>df["Sales"].mean(),"Yes","No")   -- flag high sales
df["Sales_Category"] = pd.cut(df["Sales"], bins=[0,150,250,350], labels=["Low","Medium","High"])  -- bins
```

CUSTOMER ANALYTICS (RFM Example)

```
df_rfm = df.groupby("Region").agg({
   "Month": lambda x: (df["Month"].max() - x.max()),  -- recency
   "Sales": "count",                            -- frequency
   "Sales": "sum"                               -- monetary
})
```

PREDICTIONS

```
df["Predicted_Sales"] = df["Sales"].rolling(window=3).mean().shift(1) -- rolling forecast
df["Sales_Trend"] = df["Sales"].diff()                          -- sales change
df["Sales_Volatility"] = df["Sales"].pct_change()                 --% change
df["Forecast_EWMA"] = df["Sales"].ewm(span=2).mean()             -- exponential forecast
df["CLV"] = df["Sales"].mean() * df["Month"].nunique() * 2          -- simple CLV
```

RECOMMENDATIONS

```
df["Growth_Recommendation"] = df["Growth_Rate"].apply(lambda x: "Expand" if x>10 else "Monitor")--
expand/monitor
df["Risk_Flag"] = df["Sales"].apply(lambda x: "Risk" if x<150 else "Safe")                    -- risk indicator
df["Performance_Tag"] = np.where(df["Sales"]>200,"Excellent","Average")                       -- performance
tag
df["Strategy"] = np.where((df["Growth_Rate"]>5)&(df["Sales"]>200),"Boost Marketing","Maintain") --
strategy suggestion
```

VISUALIZATION (Matplotlib)

```
plt.plot(df["Month"], df["Sales"])                  --line chart
plt.bar(df["Month"], df["Sales"])                    -- bar chart
plt.hist(df["Sales"], bins=5)                        -- histogram
plt.scatter(df["Month"], df["Sales"])                -- scatter plot
plt.pie(df.groupby("Region")["Sales"].sum(), labels=df["Region"].unique()) -- pie chart
plt.show()
```

VISUALIZATION (Seaborn)

```
sns.lineplot(x="Month", y="Sales", data=df)          -- line plot
sns.barplot(x="Region", y="Sales", data=df)          -- bar plot
sns.boxplot(x="Category", y="Sales", data=df)         -- box plot
sns.scatterplot(x="Month", y="Sales", hue="Region", data=df) -- scatter with categories
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")     -- heatmap
sns.countplot(x="Category", data=df)                 -- count of categories
sns.violinplot(x="Category", y="Sales", data=df)       -- violin plot
sns.pairplot(df, hue="Region")                       --pairplot (multi-variable relationships)
```