+ Code + Text

Importing library

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

Importing dataset file

```
df=pd.read_csv(r"/content/Data Model - Pizza Sales-CSV.csv")
```

Printing the details in the file

```
df
```

	pizza_id	quantity	order_date	order_time	unit_price	total_price(USD)	total_price(INR)	pizza_size	pizza_category	pizz
	hawaiian_m	1	1/1/2021	11:38:36	13.25	13.25	1086.5	M	Classic	Mo
	classic_dlx_m	1	1/1/2021	11:57:40	16.00	16.00	1312.0	M	Classic	M
	five_cheese_l	1	1/1/2021	11:57:40	18.50	18.50	1517.0	L	Veggie	Moz Pro
	ital_supr_l	1	1/1/2021	11:57:40	20.75	20.75	1701.5	L	Supreme	Ca
	mexicana_m	1	1/1/2021	11:57:40	16.00	16.00	1312.0	M	Veggie	Pe Pe
	...	...	...	...	...	...	...	...	...	
	ckn_alfredo_m	1	12/31/2021	21:23:10	16.75	16.75	1373.5	M	Chicken	
	four_cheese_l	1	12/31/2021	21:23:10	17.95	17.95	1471.9	L	Veggie	Gorg
	napolitana_s	1	12/31/2021	21:23:10	12.00	12.00	984.0	S	Classic	An
	mexicana_l	1	12/31/2021	22:09:54	20.25	20.25	1660.5	L	Veggie	Pe Pe
	bbq_ckn_s	1	12/31/2021	23:02:05	12.75	12.75	1045.5	S	Chicken	Barb Red

Here (rows, columns)

```
df.shape
```

```
(48620, 15)
```

Information about the file is given like datatype, null values etc....

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48620 entries, 0 to 48619
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   S.no               48620 non-null  int64  
 1   order_id           48620 non-null  int64  
 2   pizza_id           48620 non-null  object 
 3   quantity           48620 non-null  int64  
 4   order_date         48620 non-null  object 
 5   order_time         48620 non-null  object 
 6   unit_price         48620 non-null  float64
 7   total_price(USD)   48620 non-null  float64
 8   total_price(INR)   48620 non-null  float64
 9   pizza_size         48620 non-null  object 
 10  pizza_category     48620 non-null  object 
 11  pizza_ingredients  48620 non-null  object 
 12  pizza_name         48620 non-null  object 
 13  Unnamed: 13        0 non-null      float64
 14  Unnamed: 14        1 non-null      float64
dtypes: float64(5), int64(3), object(7)
memory usage: 5.6+ MB
```

Here At What time, how many pizzas were ordered is given

```
df["order_time"] = df["order_time"].astype("string")
df[["hour","minute","second"]] = df["order_time"].str.split(":",expand=True)
df["hour"].value_counts().sort_index()
```

hour	count
10	17
11	2672
12	6543
13	6203
14	3521
15	3170
16	4185
17	5143
18	5359
19	4350
20	3487
21	2528
22	1370
23	68
9	4

Here the Total amount of sales is happened in that year

```
totalsales=df["total_price(INR)"].sum()
print(totalsales , "rupees")
```

```
67064524.1 rupees
```

Here analysing the data for each days.

```
df['order_day'] = pd.to_datetime(df['order_date'], errors='coerce')
```

```
df["order_day"] = df["order_day"].dt.day_name()
df["order_day"].value_counts().sort_index()
```

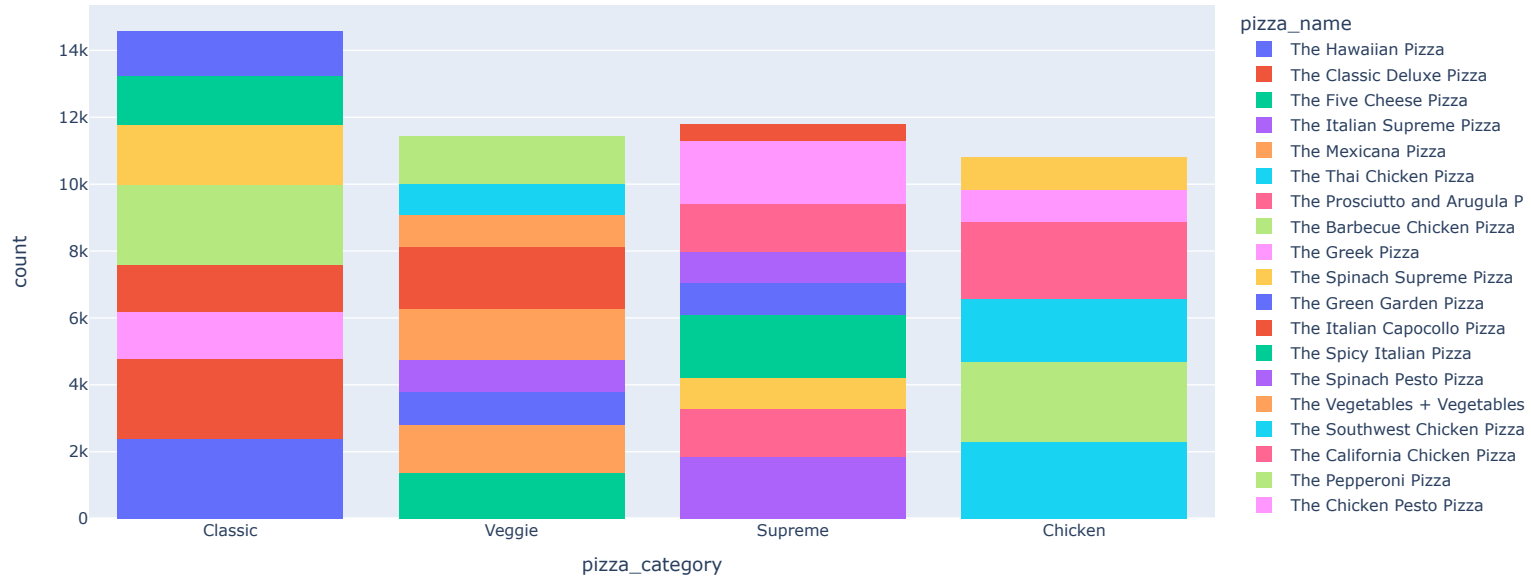
order_day	count
Friday	7323
Monday	5917
Saturday	8106
Sunday	7355
Thursday	6797
Tuesday	6369
Wednesday	6753

Here we have 4 categories of pizzas

```
px.histogram(df,x="order_day",color='pizza_category')
```

Each of the categories there were some types

```
px.histogram(df,x="pizza_category",color="pizza_name")
```



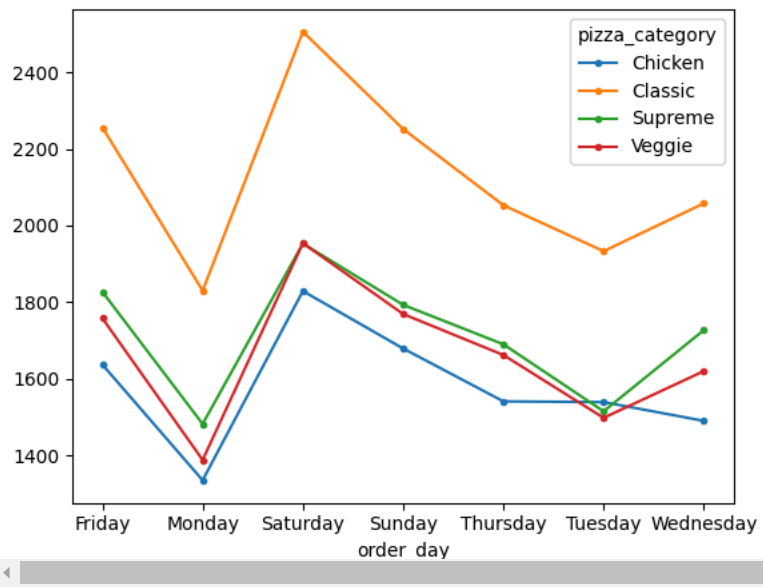
Top selling pizza's

```
top5 = df.groupby("pizza_name")["quantity"].count().sort_values(ascending=False).head(5)
top5
```

pizza_name	quantity
The Classic Deluxe Pizza	2416
The Barbecue Chicken Pizza	2372
The Hawaiian Pizza	2370
The Pepperoni Pizza	2369
The Thai Chicken Pizza	2315

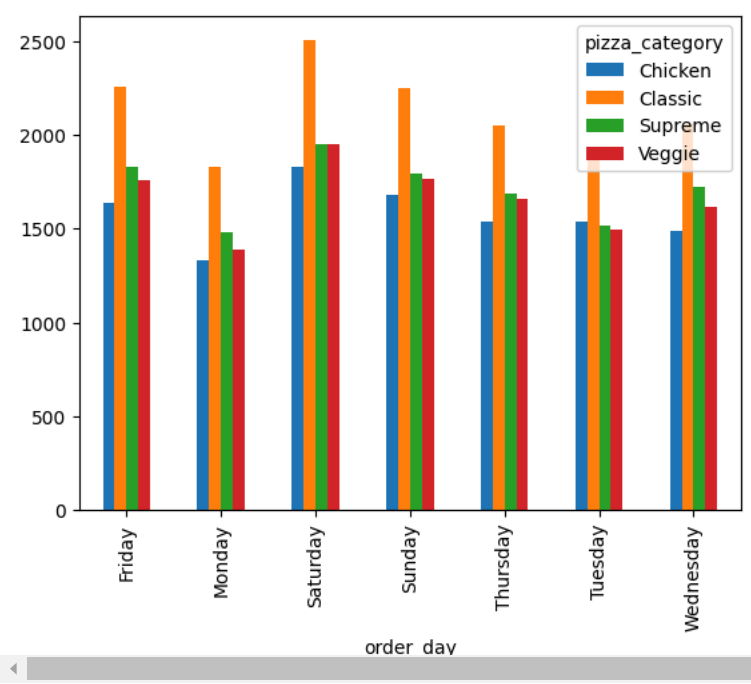
Order day VS quantity of pizza's

```
spread=df.groupby(["order_day","pizza_category"])["quantity"].sum().unstack()
spread=spread.plot(marker = ".")
```



Using bar graph

```
spread2 = df.groupby(["order_day", "pizza_category"])["quantity"].sum().unstack()
spread2 = spread2.plot.bar()
```



Least selling pizza's

```
lowest=df.loc[df['pizza_category'] == "Chicken"]
lowest=lowest.groupby("pizza_name")["quantity"].sum().sort_values(ascending=True)
lowest
```

pizza_name	quantity
The Chicken Pesto Pizza	973
The Chicken Alfredo Pizza	987
The Southwest Chicken Pizza	1917
The California Chicken Pizza	2370
The Thai Chicken Pizza	2371
The Barbecue Chicken Pizza	2432

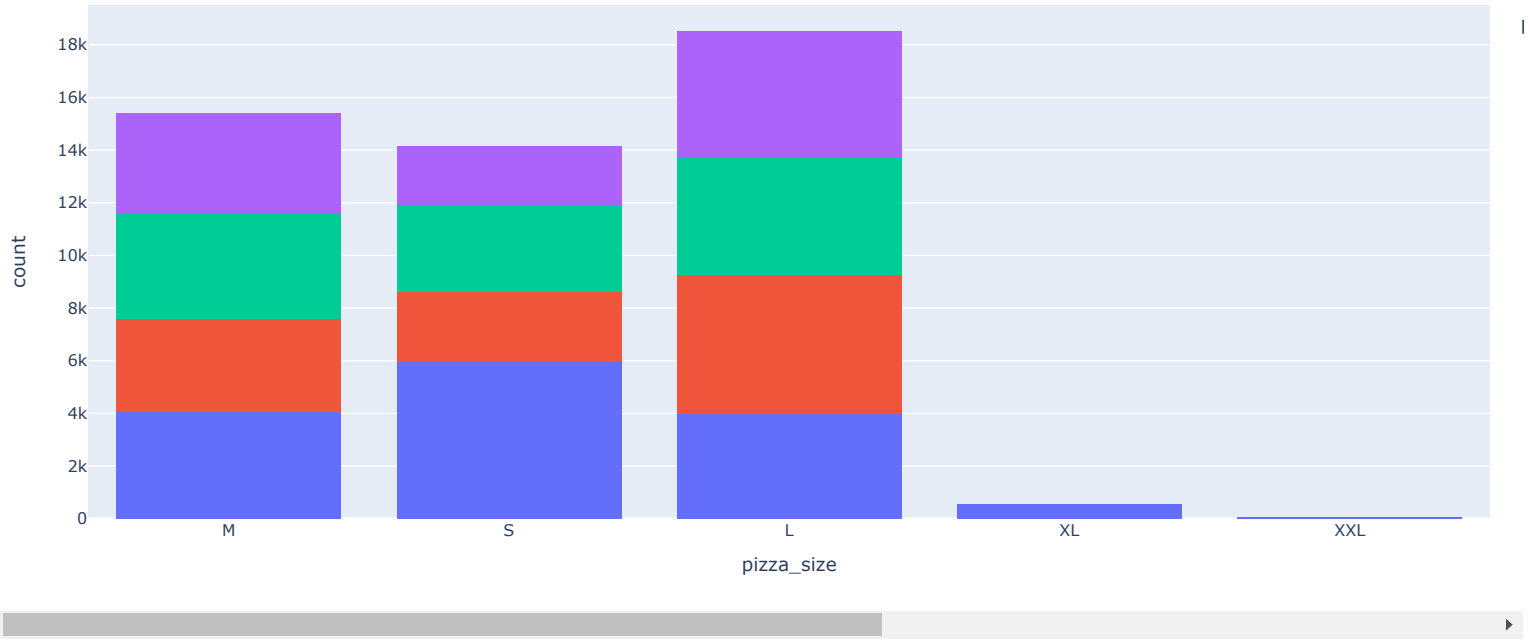
Size of the pizza

```
df.pizza_size.value_counts()
```

pizza_size	count
L	18526
M	15385
S	14137
XL	544
XXL	28

pizza size VS Category

```
px.histogram(df,x="pizza_size",color="pizza_category")
```



Packages for Machine Learning

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
df = pd.read_csv(r"/content/Data Model - Pizza Sales-CSV.csv")
```

Data preprocessing

```
df["order_day"] = pd.to_datetime(df['order_date'], errors='coerce')
df["order_day"] = df["order_day"].dt.day_name()
```

Convert categorical columns to numerical

```
label_encoder = LabelEncoder()
df['pizza_size'] = label_encoder.fit_transform(df['pizza_size'])
df['pizza_category'] = label_encoder.fit_transform(df['pizza_category'])
df['order_day'] = label_encoder.fit_transform(df['order_day'])
df['pizza_name'] = label_encoder.fit_transform(df['pizza_name'])
```

features and target

```
X = df[['pizza_size', 'pizza_category', 'order_day', 'pizza_name']]
y = df['total_price(INR)']
```

data into train and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model selection and training

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Make predictions

```
y_pred = model.predict(X_test)
```

Evaluate the model

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

MSE and R2 were printed

```
print(f"Mean Squared Error: {mse}")
print(f"R2 Score: {r2}")
```

```
Mean Squared Error: 83378.68771180959
R2 Score: 0.3803807341108836
```

Evaluation based on MSE and R2

```
# Evaluation based on MSE and R2
if mse < 5000 and r2 > 0.8:
    print("The model is performing well with a low MSE and high R² score.")
    print("Recommendation: This model is suitable for making predictions.")
elif mse < 10000 and r2 > 0.6:
    print("The model is performing decently with a moderate MSE and acceptable R² score.")
    print("Recommendation: This model is acceptable, but there is room for improvement.")
else:
    print("The model is performing poorly with a high MSE and/or low R² score.")
    print("Recommendation: Consider improving the model by trying other algorithms or tuning hyperparameters.")
```

```
The model is performing poorly with a high MSE and/or low R² score.
Recommendation: Consider improving the model by trying other algorithms or tuning hyperparameters.
```

Prediction for new data

```
new_data = [[2, 1, 3, 5]]  # Modify this based on actual values
predicted_sales = model.predict(new_data)
print(f"Predicted Sales: {predicted_sales[0]} INR")
```