```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files Restaurant sales.csv
Restaurant sales.csv(text/csv) - 62494 bytes, last modified: 11/1/2026 - 100% done
Saving Restaurant sales.csv to Restaurant sales.csv
```

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

```
restaurants_sales_data=pd.read_csv('Restaurant sales.csv')
print(restaurants_sales_data .head())
```

```
   order_id        date        item_name  item_type  item_price  quantity  \
0         1  07-03-2022         Aalopuri   Fastfood          20        13
1         2   8/23/2022          Vadapav   Fastfood          20        15
2         3  11/20/2022          Vadapav   Fastfood          20         1
3         4  02-03-2023  Sugarcane juice  Beverages          25         6
4         5  10-02-2022  Sugarcane juice  Beverages          25         8

   transaction_amount transaction_type received_by time_of_sale
0                 260              NaN         Mr.        Night
1                 300             Cash         Mr.    Afternoon
2                  20             Cash         Mr.    Afternoon
3                 150           Online         Mr.        Night
4                 200           Online         Mr.      Evening
```

```
restaurants_sales_data.shape
```

```
(1000, 10)
```

```
restaurants_sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   order_id            1000 non-null   int64
 1   date                1000 non-null   object
 2   item_name           1000 non-null   object
 3   item_type           1000 non-null   object
 4   item_price          1000 non-null   int64
 5   quantity            1000 non-null   int64
 6   transaction_amount  1000 non-null   int64
 7   transaction_type    893 non-null    object
 8   received_by         1000 non-null   object
 9   time_of_sale        1000 non-null   object
dtypes: int64(4), object(6)
memory usage: 78.3+ KB
```

```
restaurants_sales_data.describe()
```

	order_id	item_price	quantity	transaction_amount
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	33.315000	8.162000	275.230000
std	288.819436	14.921744	4.413075	204.402979
min	1.000000	20.000000	1.000000	20.000000
25%	250.750000	20.000000	4.000000	120.000000
50%	500.500000	25.000000	8.000000	240.000000
75%	750.250000	50.000000	12.000000	360.000000
max	1000.000000	60.000000	15.000000	900.000000

```
restaurants_sales_data.isnull().sum()
```

	0
order_id	0
date	0
item_name	0
item_type	0
item_price	0
quantity	0
transaction_amount	0

```
restaurants_sales_data["item_name"]=restaurants_sales_data["item_name"].fillna(restaurants_sales_data["item_name"].mode()[0]
restaurants_sales_data
```

time_of_sale order_id	0 date	item_name	item_type	item_price	quantity	transaction_amount	transaction_type	received_by	time
0	1	07-03-2022	Aalopuri	Fastfood	20	13	260	NaN	Mr.	
1	2	8/23/2022	Vadapav	Fastfood	20	15	300	Cash	Mr.	
2	3	11/20/2022	Vadapav	Fastfood	20	1	20	Cash	Mr.	
3	4	02-03-2023	Sugarcane juice	Beverages	25	6	150	Online	Mr.	
4	5	10-02-2022	Sugarcane juice	Beverages	25	8	200	Online	Mr.	
...	...	...	...	...	...	...	...	...	...	
995	996	3/19/2023	Frankie	Fastfood	50	10	500	NaN	Mrs.	
996	997	9/20/2022	Sandwich	Fastfood	60	7	420	NaN	Mr.	
997	998	1/26/2023	Sandwich	Fastfood	60	13	780	NaN	Mr.	
998	999	8/27/2022	Panipuri	Fastfood	20	5	100	NaN	Mrs.	

```
restaurants_sales_data["item_type"].str.upper()
```

	item_type
0	FASTFOOD
1	FASTFOOD
2	FASTFOOD
3	BEVERAGES
4	BEVERAGES
...	...
995	FASTFOOD
996	FASTFOOD
997	FASTFOOD
998	FASTFOOD
999	FASTFOOD

1000 rows × 1 columns

dtype: object

```
restaurants_sales_data.duplicated().sum()
```

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

```
restaurants_sales_data["transaction_type"].value_counts()
```

```
restaurants_sales_data.groupby("item_name")["quantity"].sum()
```

transaction_type

	quantity
Cash	476
Online	417

item_name

Aalopuri	1044
Cold coffee	1361
Frankie	1150
Panipuri	1226
Sandwich	1097
Sugarcane juice	1278
Vadapav	1006

dtype: int64

dtype: int64

```
unique_item_name=restaurants_sales_data["item_name"].unique()
print("unique item name:")
for item_name in unique_item_name:
  print(item_name)
```

```
unique item name:
Aalopuri
Vadapav
Sugarcane juice
Panipuri
Frankie
Sandwich
Cold coffee
```

```
total_quantity_sold=restaurants_sales_data["quantity"].sum()
print(f'total quantity of goods sold:{total_quantity_sold}')
```

```
total quantity of goods sold:8162
```

```
total_quantity_sold_by_item=restaurants_sales_data.groupby("item_name")["quantity"].sum()
print("total quantity of goods sold for each item:")
print(total_quantity_sold_by_item)
```

```
total quantity of goods sold for each item:
item_name
Aalopuri           1044
Cold coffee        1361
Frankie            1150
Panipuri           1226
Sandwich           1097
Sugarcane juice    1278
Vadapav            1006
Name: quantity, dtype: int64
```

Data Visualization

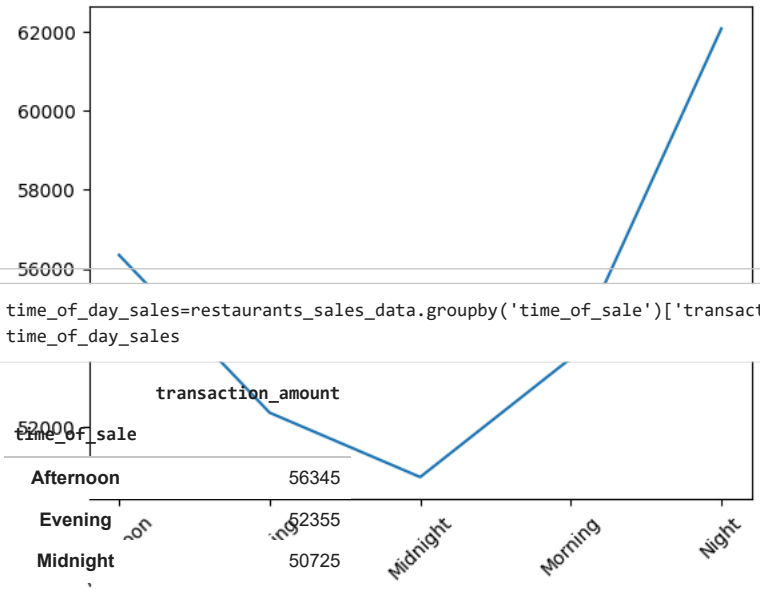
1. Line Plot

```
daily_sales=restaurants_sales_data.groupby('time_of_sale')['transaction_amount'].sum()
daily_sales.head()
```

time_of_sale	transaction_amount
Afternoon	56345
Evening	52355
Midnight	50725
Morning	53730
Night	62075

dtype: int64

```
plt.plot(daily_sales.index,daily_sales.values)
plt.xticks(rotation=45)
plt.show()
```

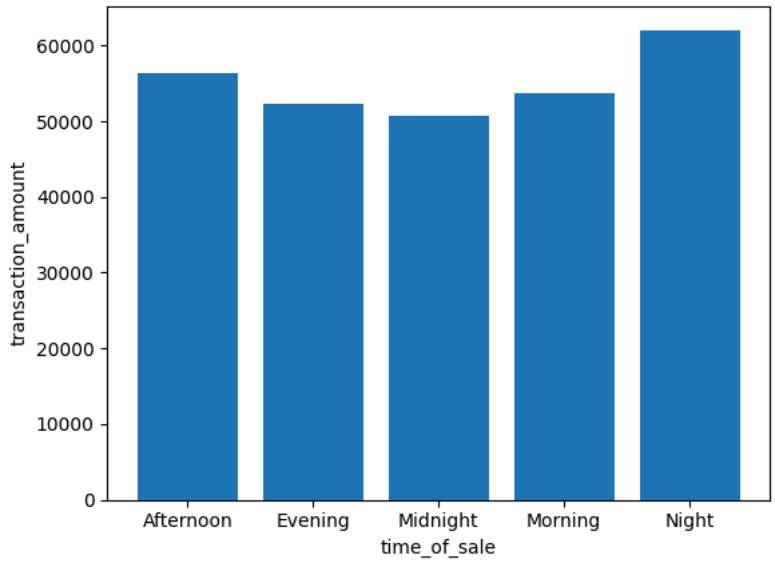



```
time_of_day_sales=restaurants_sales_data.groupby('time_of_sale')['transaction_amount'].sum()
time_of_day_sales
```

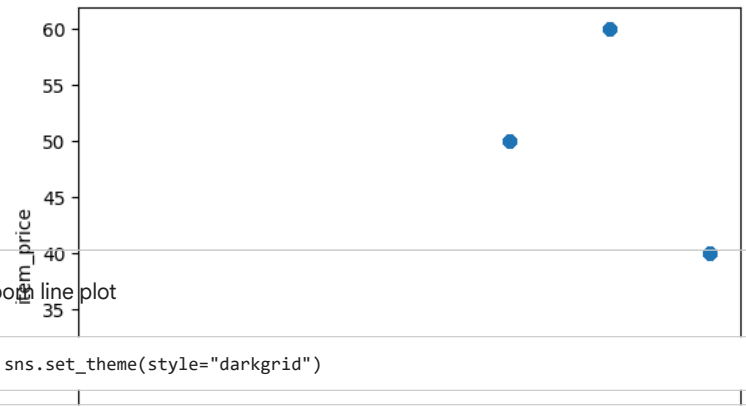
time_of_sale	transaction_amount
Afternoon	56345
Evening	52355
Midnight	50725
Morning	53730
Night	62075

dtype: int64

```
plt.bar(time_of_day_sales.index,time_of_day_sales.values)
plt.xlabel('time_of_sale')
plt.ylabel('transaction_amount')
plt.show()
```



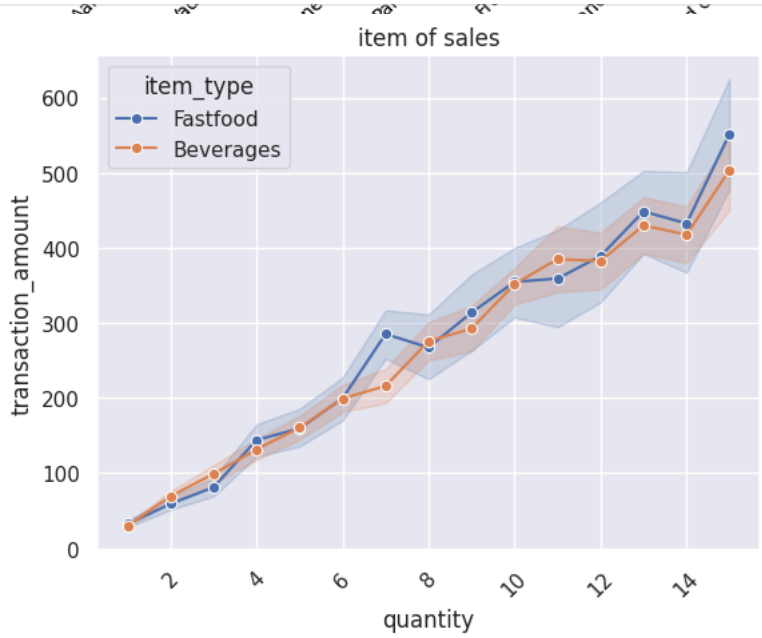

```
plt.scatter(restaurants_sales_data['item_name'],restaurants_sales_data['item_price'])
plt.xlabel('item_name')
plt.xticks(rotation=45)
plt.ylabel('item_price')
plt.show()
```



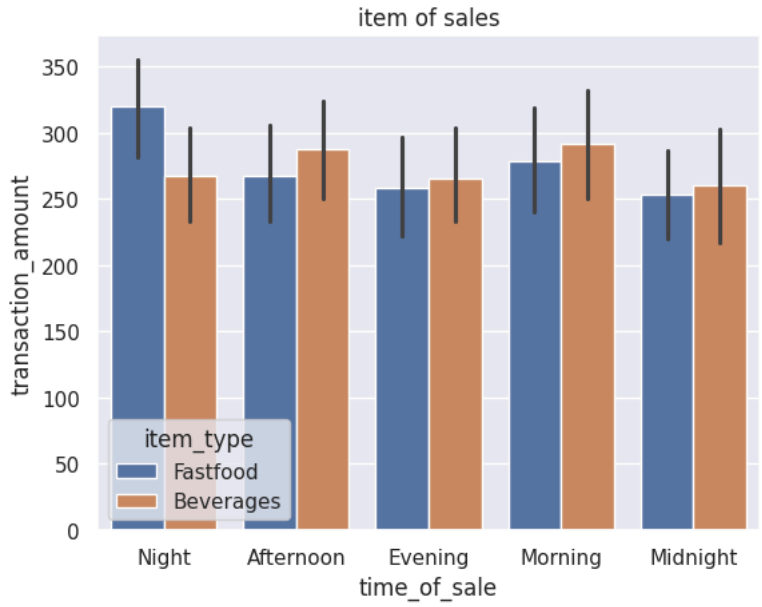
Seaborn line plot

```
sns.set_theme(style="darkgrid")
```

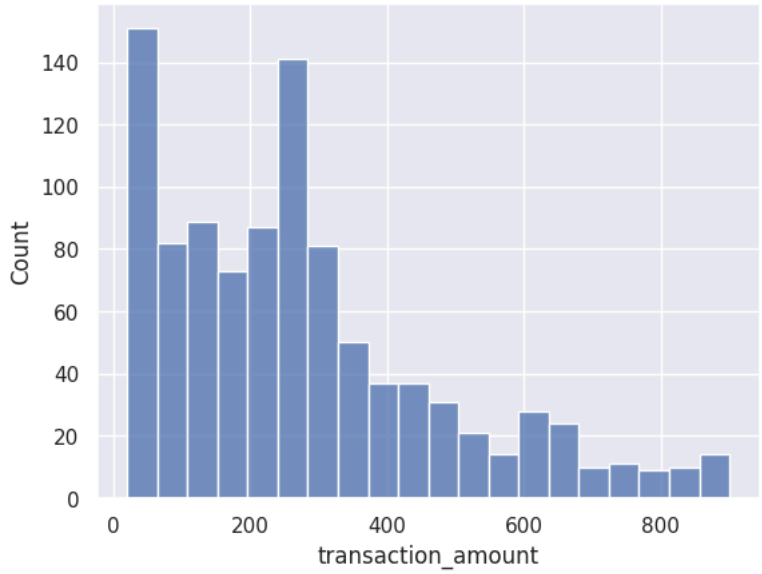
```
sns.lineplot(x='quantity',y='transaction_amount', hue='item_type',data=restaurants_sales_data,marker='o')
plt.title('item of sales')
plt.figsize=(10,6)
plt.xticks(rotation=45)
plt.show()
```



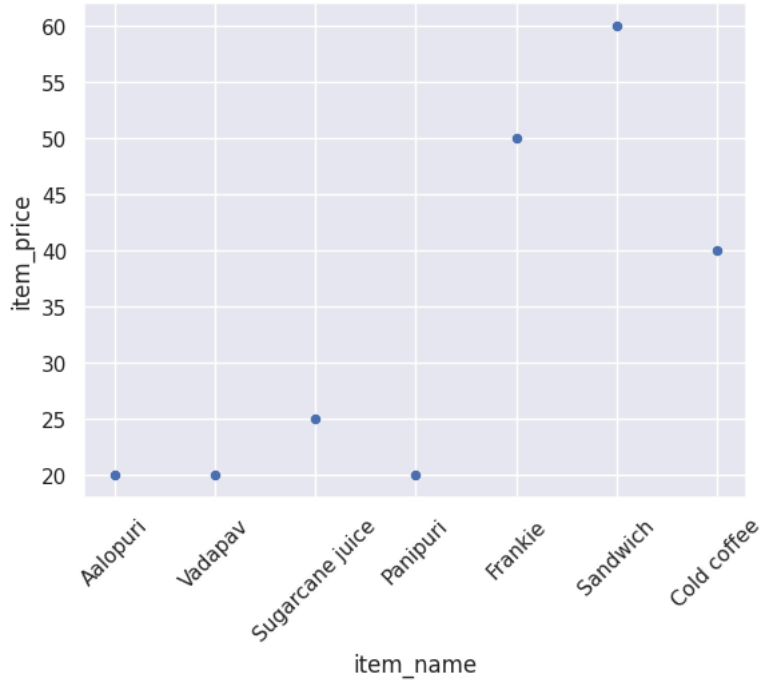

```
sns.barplot(x='time_of_sale',y='transaction_amount',hue='item_type',data=restaurants_sales_data)
plt.title('item of sales')
plt.show()
```




```
sns.histplot(restaurants_sales_data['transaction_amount'],bins=20)
plt.show()
```




```
sns.scatterplot(x='item_name',y='item_price',data=restaurants_sales_data)
plt.xticks(rotation=45)
plt.show()
```




```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
```

```
X = restaurants_sales_data.drop(columns=["transaction_amount", "order_id", "date"])
y = restaurants_sales_data["transaction_amount"]

cat_cols = X.select_dtypes(include="object").columns
num_cols = X.select_dtypes(exclude="object").columns

preprocess = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ("num", "passthrough", num_cols)
])

model = RandomForestRegressor(random_state=42)

pipe = Pipeline([
    ("preprocess", preprocess),
```

```
    ("model", model)
])

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

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
```

```
# Import evaluation metrics
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Predict on test data
y_pred = pipe.predict(X_test)

# Calculate RMSE
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("Root Mean Square Error (RMSE):", rmse)

# Calculate R2 Score
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
```

```
Root Mean Square Error (RMSE): 1.27071338231719
R2 Score: 0.999962420079722
```

```
!pip install gradio scikit-learn pandas joblib numpy
```

Show hidden output

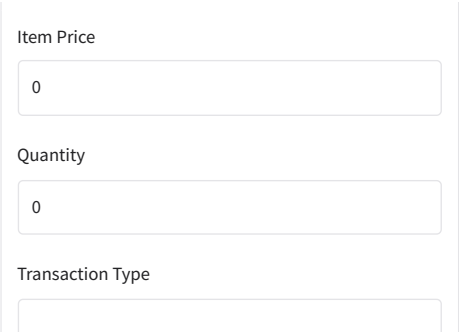
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
import joblib

# Load dataset
df = pd.read_csv("Restaurant sales.csv")

# Features & target
X = df.drop(columns=["transaction_amount", "order_id", "date"])
y = df["transaction_amount"]

cat_cols = X.select_dtypes(include="object").columns
num_cols = X.select_dtypes(exclude="object").columns

preprocess = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ("num", "passthrough", num_cols)
])

model = RandomForestRegressor(random_state=42)

pipe = Pipeline([
    ("preprocess", preprocess),
    ("model", model)
])

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

pipe.fit(X_train, y_train)

# Save model
joblib.dump(pipe, "model.pkl")

print("Model saved successfully")
```

```
Model saved successfully
```

```
import gradio as gr
import pandas as pd
import joblib

# Load trained model
```

```
model = joblib.load("model.pkl")

# Prediction function
def predict_sales(item_name, item_type, item_price, quantity,
                  transaction_type, received_by, time_of_sale):

    input_data = pd.DataFrame([{
        "item_name": item_name,
        "item_type": item_type,
        "item_price": item_price,
        "quantity": quantity,
        "transaction_type": transaction_type,
        "received_by": received_by,
        "time_of_sale": time_of_sale
    }])

    prediction = model.predict(input_data)
    return f"Predicted Transaction Amount: ₹ {round(prediction[0], 2)}"

# Gradio Interface
interface = gr.Interface(
    fn=predict_sales,
    inputs=[
        gr.Dropdown(["Vadapav", "Aalopuri", "Sugarcane juice"], label="Item Name"),
        gr.Dropdown(["Fastfood", "Beverages"], label="Item Type"),
        gr.Number(label="Item Price"),
        gr.Number(label="Quantity"),
        gr.Dropdown(["Cash", "Online"], label="Transaction Type"),
        gr.Dropdown(["Mr.", "Mrs."], label="Received By"),
        gr.Dropdown(["Morning", "Afternoon", "Evening", "Night"], label="Time of Sale")
    ],
    outputs="text",
    title="🍽️ Restaurant Sales Prediction",
    description="Predict restaurant transaction amount using Machine Learning"
)

interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://56c0705d006f4ca0c8.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the
```



Next steps: Deploy to Cloud Run