Project Goal

To analyze Balaji Fast Food's sales data in order to:

- Analyzing sales trends over time

- Understand customer purchasing behavior and preferences.

- Identify top-selling items and underperforming products.

- Assess sales performance across different time periods and payment methods.

- Evaluate staff performance based on sales handled.

- Provide data-driven recommendations to improve profitability and operational efficiency.

Some areas to carry out the analysis

- What is the total revenue generated over the entire period?

- How do sales vary month by month?

- What days of the week bring in the highest and lowest sales?

- What is the most sold product by quantity?

- What payment methods is the most commonly used?

- What time of the day sees the highest sales volume?

- Which items are most popular during differnt times of the day (Morning, Afternoon, Evening, Night, Midnight)?

- Which staff member handled the most transactions?

- Who generated the highest sales revenue?

- What is the average quantity sold ?

- Are there any gaps(missing data) or inconsistent value in the dataset?

- Fix bad data

- Correct any missing data/gaps

Here is the breakdown of the information in each column:

. Order_id: a unique identifier for each order

. Date: date of transaction

. Item_name: name of the food

. Item_type: catergory of items (Fastfood or Bevergaes).

. Item_price:price of the items for 1 quantity

. Quantity:how much the customer orders

. Transaction_amount: the total amount paid by customers

. Transaction_type:payment method (Cash, online, others)

. Received_by: gender of the person handling the transaction.

. Time_of_sales: different times of the day (Morning, Evening, Afternoon,Night,Midnight)

In [1]:
```
# To import the needed libraries
import pandas as pd
import numpy as np
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
C:\Users\HELLO\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of
'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
# To call in the file
data = pd.read_csv("Balaji_Fast_Food_Sales.csv")
```

In [3]:
```
# To show the first 10 rows of the dataset
data.head(10)
```

Out [3]:

	order_id	date	item_name	item_type	item_price	quantity	transaction_amount	transaction_type	received_by	time_of_sale
0	1	2022-03-07	Aalopuri	Fastfood	20	13	260	NaN	Mr.	Night
1	2	2022-08-23	Vadapav	Fastfood	20	15	300	Cash	Mr.	Afternoon
2	3	2022-11-20	Vadapav	Fastfood	20	1	20	Cash	Mr.	Afternoon
3	4	2023-03-02	Sugarcane juice	Beverages	25	6	150	Online	Mr.	Night
4	5	2022-02-10	Sugarcane juice	Beverages	25	8	200	Online	Mr.	Evening
5	6	2022-11-14	Vadapav	Fastfood	20	10	200	Cash	Mr.	Evening
6	7	2022-03-05	Sugarcane juice	Beverages	25	9	225	Cash	Mr.	Evening
7	8	2022-12-22	Panipuri	Fastfood	20	14	280	Online	Mr.	Night
8	9	2022-10-06	Panipuri	Fastfood	20	1	20	Cash	Mrs.	Morning
9	10	2022-09-16	Panipuri	Fastfood	20	5	100	Online	Mr.	Afternoon

DATA CLEANING

In [4]:
```
# To know the shape or size of the dataset
data.shape
```

Out [4]: (1000, 10)

In [5]:
```
# To know the dataset columns
data.columns
```

Out [5]:
```
Index(['order_id', 'date', 'item_name', 'item_type', 'item_price', 'quantity',
       'transaction_amount', 'transaction_type', 'received_by',
       'time_of_sale'],
      dtype='object')
```

In [6]:
```
#To know the data types and get familiarise with the data to know how to work with them.
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
```

In [7]:
```
# To check for misssing value
data.isnull().sum()
```

Out [7]:
```
order_id              0
date                  0
item_name             0
item_type             0
item_price            0
quantity              0
transaction_amount    0
transaction_type    107
received_by           0
```

```
time_of_sale            0
dtype: int64
```

In [8]:
```
# To handle missing values in 'transaction_type'
data['transaction_type'].fillna('Unknown',inplace=True)
```

```
C:\Users\HELLO\AppData\Local\Temp\ipykernel_4664\2576236135.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the original object.


  data['transaction_type'].fillna('Unknown',inplace=True)
```

In [9]:
```
# Checking for duplicates
data.drop_duplicates(inplace=True)
```

In [10]:
```
# To convert date column to datetime format
data['date'] = pd.to_datetime(data['date'])
```

In [11]:
```
# To convert period to timestamp format
data['month'] = data['date'].dt.to_period('M').dt.to_timestamp()
```

In [12]:
```
# To check if missing value is gone
data.isnull().sum()
```

Out [12]:
```
order_id              0
date                  0
item_name             0
item_type             0
item_price            0
quantity              0
transaction_amount    0
transaction_type      0
received_by           0
time_of_sale          0
month                 0
dtype: int64
```

EXPLORATORY DATA ANALYSIS (EDA)

In [13]:
```
#To show the number of rows and columns in the dataset
data.shape
```

Out [13]:
```
(1000, 11)
```

In [14]:
```
#To show the first 10 rows of the dataset
data.head(10)
```

Out [14]:

	order_id	date	item_name	item_type	item_price	quantity	transaction_amount	transaction_type	received_by	time_of_sale	month
0	1	2022-03-07	Aalopuri	Fastfood	20	13	260	Unknown	Mr.	Night	2022-03-01
1	2	2022-08-23	Vadapav	Fastfood	20	15	300	Cash	Mr.	Afternoon	2022-08-01
2	3	2022-11-20	Vadapav	Fastfood	20	1	20	Cash	Mr.	Afternoon	2022-11-01
3	4	2023-03-02	Sugarcane juice	Beverages	25	6	150	Online	Mr.	Night	2023-03-01
4	5	2022-02-10	Sugarcane juice	Beverages	25	8	200	Online	Mr.	Evening	2022-02-01
5	6	2022-11-14	Vadapav	Fastfood	20	10	200	Cash	Mr.	Evening	2022-11-01
6	7	2022-03-05	Sugarcane juice	Beverages	25	9	225	Cash	Mr.	Evening	2022-03-01
7	8	2022-12-22	Panipuri	Fastfood	20	14	280	Online	Mr.	Night	2022-12-01
8	9	2022-10-06	Panipuri	Fastfood	20	1	20	Cash	Mrs.	Morning	2022-10-01
9	10	2022-09-16	Panipuri	Fastfood	20	5	100	Online	Mr.	Afternoon	2022-09-01

In [15]:
```
# To get the shape of the data
data.shape
```

Out [15]:
```
(1000, 11)
```

In [16]:
```
# To check if missing value is gone
data.isnull().sum()
```

Out [16]:
```
order_id              0
date                  0
item_name             0
item_type             0
item_price            0
quantity              0
transaction_amount    0
transaction_type      0
received_by           0
time_of_sale          0
month                 0
dtype: int64
```

In [17]:
```
# To get the summary statistics of the data
data.describe()
```

Out [17]:

	order_id	date	item_price	quantity	transaction_amount	month
count	1000.000000	1000	1000.000000	1000.000000	1000.000000	1000
mean	500.500000	2022-10-02 05:15:21.600000	33.315000	8.162000	275.230000	2022-09-17 16:35:02.400000256
min	1.000000	2022-01-04 00:00:00	20.000000	1.000000	20.000000	2022-01-01 00:00:00
25%	250.750000	2022-06-17 00:00:00	20.000000	4.000000	120.000000	2022-06-01 00:00:00
50%	500.500000	2022-09-21 00:00:00	25.000000	8.000000	240.000000	2022-09-01 00:00:00
75%	750.250000	2023-01-02 00:00:00	50.000000	12.000000	360.000000	2023-01-01 00:00:00
max	1000.000000	2023-12-03 00:00:00	60.000000	15.000000	900.000000	2023-12-01 00:00:00
std	288.819436	NaN	14.921744	4.413075	204.402979	NaN

Sales trends over time

In [18]:
```
# To group sales by month
Monthly_sales = data.groupby('month')['transaction_amount'].sum().reset_index()
Monthly_sales.columns = ["Monthly_sales", "transaction_amount"]
Monthly_sales
```

Out [18]:

	Monthly_sales	transaction_amount
0	2022-01-01	5785
1	2022-02-01	8010
2	2022-03-01	4720
3	2022-04-01	17610
4	2022-05-01	21155
5	2022-06-01	16670
6	2022-07-01	17420
7	2022-08-01	21285
8	2022-09-01	21340
9	2022-10-01	19240
10	2022-11-01	21200
11	2022-12-01	21140
12	2023-01-01	19680
13	2023-02-01	17510
14	2023-03-01	20400
15	2023-04-01	3450
16	2023-05-01	5015
17	2023-06-01	2490
18	2023-07-01	2030
19	2023-08-01	1680
20	2023-09-01	2530

	Monthly_sales	transaction_amount
21	2023-10-01	1955
22	2023-11-01	1535
23	2023-12-01	1380

Sales by product categories

In [19]:
```
# To group items by quality
item_names = data.groupby("item_name")['quantity'].sum().reset_index()
item_names.columns = ["item_name", "quantity"]
item_names
```

Out [19]:

	item_name	quantity
0	Aalopuri	1044
1	Cold coffee	1361
2	Frankie	1150
3	Panipuri	1226
4	Sandwich	1097
5	Sugarcane juice	1278
6	Vadapav	1006

Sales by Transaction Type

In [20]:
```
# To group sales by Transaction type
Transaction_type = data.groupby("transaction_type")['transaction_amount'].sum().reset_index()
Transaction_type.columns = ["Transaction_type", "Transaction_amount"]
Transaction_type
```

Out [20]:

	Transaction_type	Transaction_amount
0	Cash	132840
1	Online	110595
2	Unknown	31795

Sales by Time Of Day

In [21]:
```
# To group sales by time of day
Time_of_sale = data.groupby("time_of_sale")['transaction_amount'].sum().reset_index()
Time_of_sale.columns = ["time_of_sale", "Transaction_amount"]
Time_of_sale
```

Out [21]:

	time_of_sale	Transaction_amount
0	Afternoon	56345
1	Evening	52355
2	Midnight	50725
3	Morning	53730
4	Night	62075

Sales by Staff Member

In [22]:
```
# To group sales by Staff member
Staff_Member = data.groupby("received_by")['transaction_amount'].sum().reset_index()
Staff_Member.columns = ["received_by", "Transaction_amount"]
Staff_Member
```

Out [22]:

	received_by	Transaction_amount
0	Mr.	143440
1	Mrs.	131790

METRICS

Total Revenue generated

In [23]:
```
# Total revenue
data['transaction_amount'].sum()
```

Out [23]: 275230

Total Quantity Sold

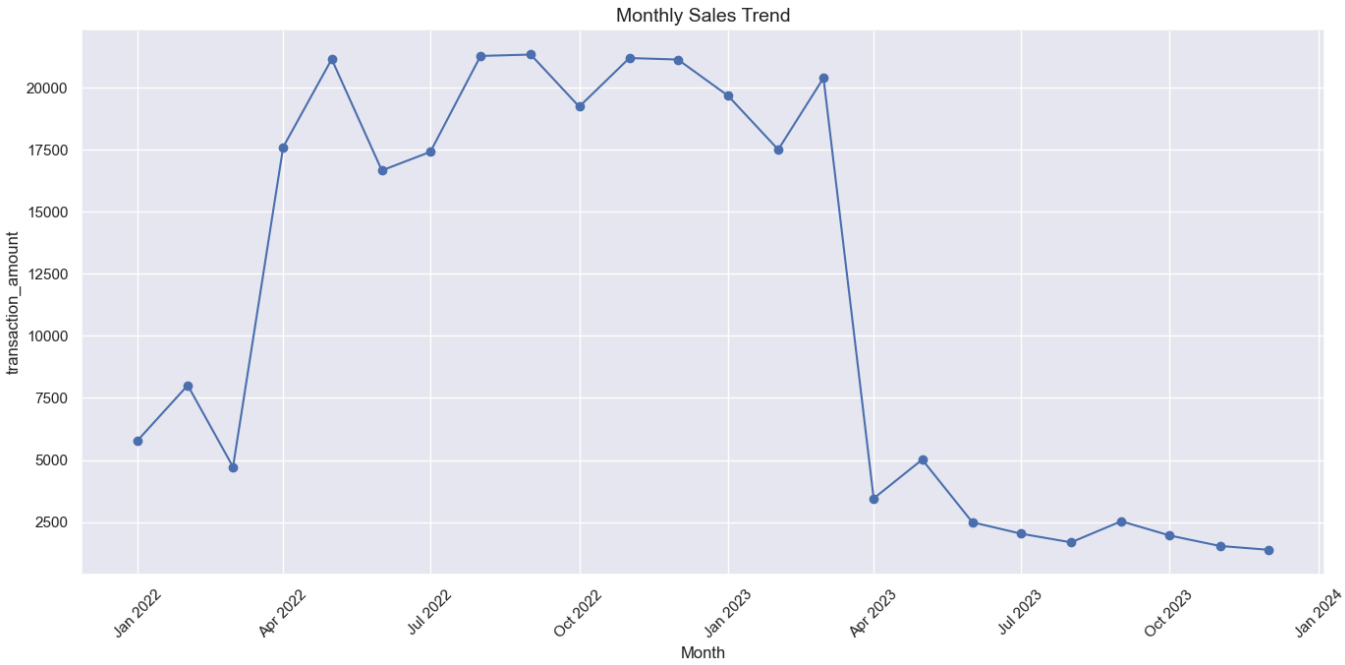
In [24]:
```
# Total Quantity
data['quantity'].sum()
```

Out [24]: 8162

DATA VISUALIZATION

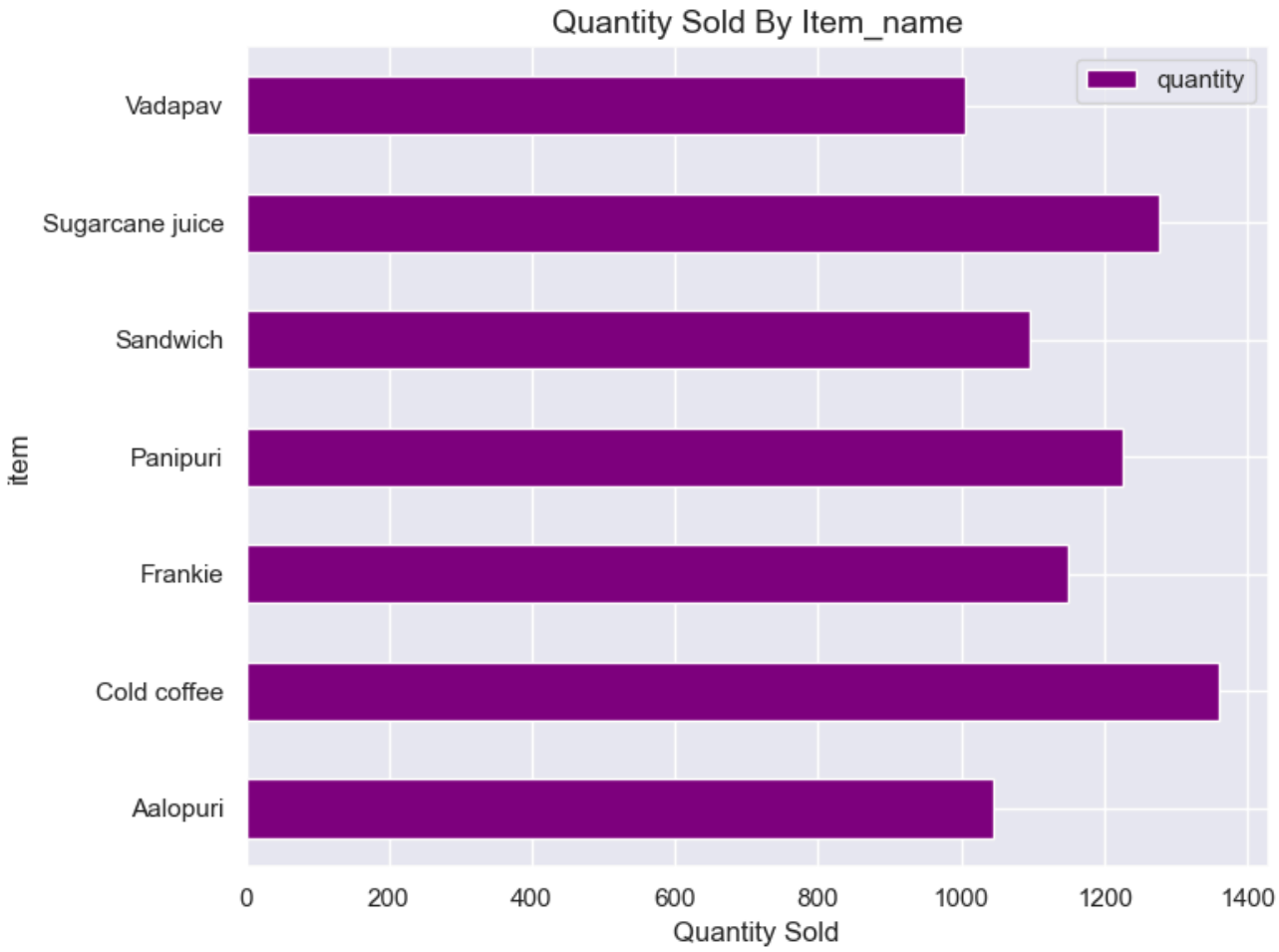
Line chart for Sales Overtime

In [25]:
```
# To plot a line chart to visualize the trend over time
plt.figure(figsize=(14,7))
plt.plot(Monthly_sales['Monthly_sales'],Monthly_sales['transaction_amount'],marker="o",linestyle='-')
plt.title("Monthly Sales Trend", fontsize=14)
plt.xlabel("Month")
plt.ylabel("transaction_amount")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



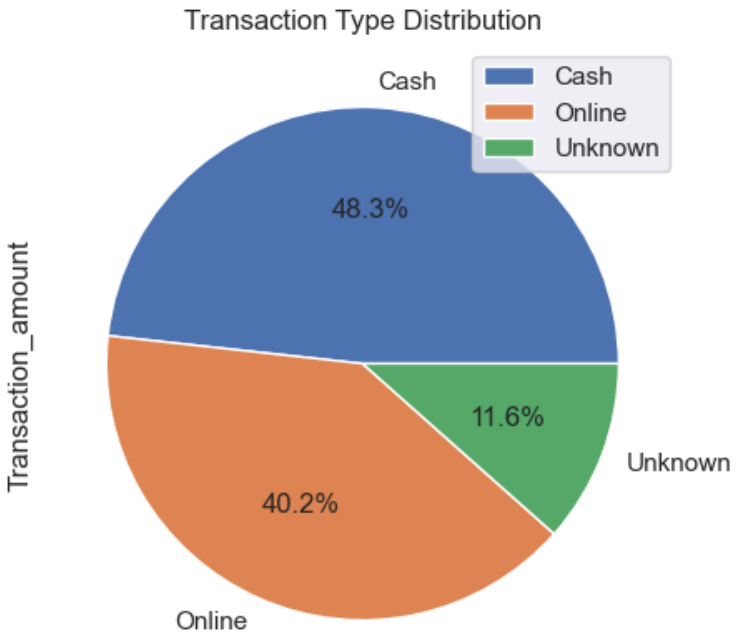
Bar Chart for Quantity by Item

In [26]:
```
# To plot a bar chart to visualize the quantity by item
item_names.plot(x='item_name', y='quantity', kind='barh',figsize=(8,6),facecolor='purple')
plt.title("Quantity Sold By Item_name", fontsize=14)
plt.ylabel("item")
plt.xlabel("Quantity Sold")
plt.tight_layout()
plt.show()
```



Pie Chart For Transaction Type distribution

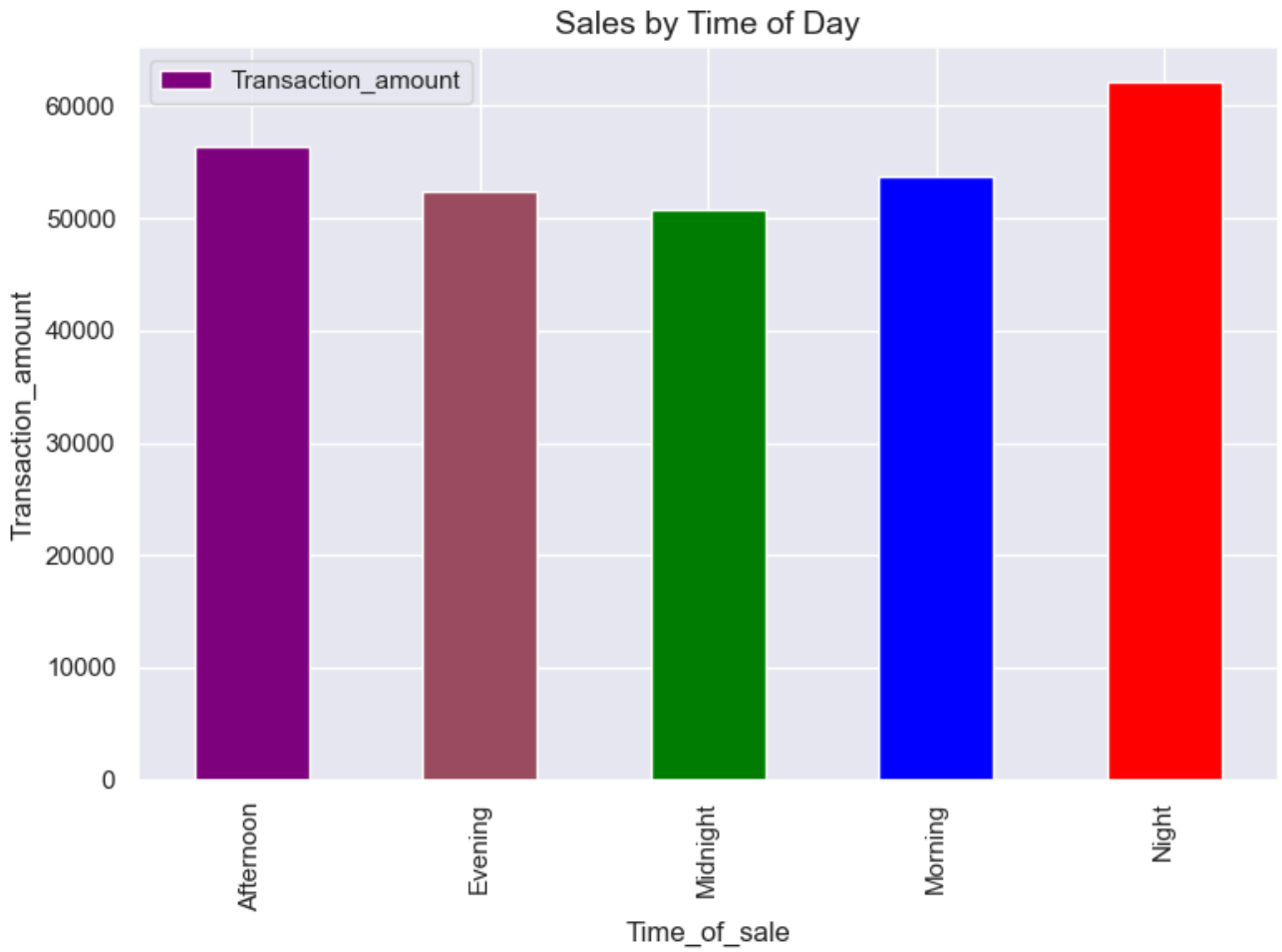
In [27]:
```
# To plot a pie chart for transaction type distribution
Transaction_type.plot(y='Transaction_amount', kind='pie', figsize=(5,5),autopct='%1.1f%%', labels=['Cash','Onl
plt.title("Transaction Type Distribution")
plt.show()
```



Bar chart for Sales by Time of the Day

In [28]:
```
# To plot a bar chart to visualize the sales by time of the day
Time_of_sale.plot(x='time_of_sale', y='Transaction_amount', kind='bar',figsize=(8,6),color={'#9b4f60','blue','
plt.title("Sales by Time of Day", fontsize=14)
plt.xlabel("Time_of_sale")
```

```
plt.ylabel("Transaction_amount")
plt.tight_layout()
plt.show()
```



Sales by Staff

In [29]:
```
# To plot a bar chart to visualize the sales by staff member
Staff_Member.plot(x='received_by', y='Transaction_amount', kind='bar',figsize=(8,5),facecolor='red')
plt.title("Sales by Staff", fontsize=14)
plt.xlabel("Staff_Member")
plt.ylabel("Transaction_amount")
plt.tight_layout()
plt.show()
```


Sales by Staff
Transaction_amount
140000
120000
100000
80000
60000
40000
20000
0
Transaction_amount
Mr.
Mrs.
Staff_Member