DATA ANALYSIS PROJECT REPORT OF A RETAIL STORE

InveCareer

Invest in yourself for a brighter Career

Submitted by:-
Deepanshi Gupta
Maharishi Markandeshwar
(Deemed to be University)

Submitted to:-
InveCareer

INDEX

Acknowledgement 1

Abstract 2

Problem Statement... 3

Project Goals 4

Introduction 5

1. ASSUMPTIONS 5

2. Data Collection... 6

  2.1 Sources 6

3. Data Preprocessing 8

4. Exploratory Data Analysis (EDA) and Visualisation... 11

  Here are some steps for performing EDA: 11

- Understand the problem and the data 11
- Import and inspect the data 11
- Handle missing values.. 11
- Explore data characteristics 11
- Perform data transformation 11
- Visualize data relationships.. 11
- Handle outliers.. 11

Conclusion... 17

  Final Decision 18

Acknowledgement

I , Deepanshi Gupta, want to thank everyone who has helped and supported me throughout my project, **Retail Store Sales Analysis.**

First and foremost, a big thank you to my mentor, Mr. **Bikash Bashyal**, for giving me valuable guidance and pointing me in the right direction.

I owe a special debt of gratitude to my parents, whose constant encouragement, patience, and understanding have been the foundation of my success.

I would also like to acknowledge my friends who contributed their ideas and perspectives, which greatly enriched the project.

I appreciate each and every one of you for shaping this project and enhancing my learning experience.

Thank you all!

Abstract

This report presents a comprehensive analysis of sales data for a retail store chain, aiming to derive actionable insights to optimize inventory management and marketing strategies. The analysis focuses on understanding sales trends, identifying topselling products, and forecasting future sales. By leveraging historical sales data, the study employs various data preparation techniques, including data cleaning and transformation, to ensure the accuracy and reliability of the results.

The data analysis segment explores trends through time series analysis, evaluates product performance by categorizing topselling items, and assesses store performance by comparing sales across different locations. Various forecasting models, such as moving averages, exponential smoothing, ARIMA, and machine learning algorithms, are applied to predict future sales trends. Model evaluation is conducted to ensure high prediction accuracy.

The insights derived from the analysis provide valuable recommendations for optimizing inventory levels and formulating effective marketing strategies. The report emphasizes the importance of data-driven decision-making in enhancing operational efficiency and driving business growth. Visualizations and dashboards are developed to facilitate easy interpretation and communication of key findings.

In conclusion, the study underscores the potential of sales data analysis in transforming retail operations and highlights areas for future research to continuously improve analytical capabilities and business outcomes. The appendix section includes a detailed

data dictionary, methodology specifics, and additional charts and graphs for reference.

PROBLEM STATEMENT:

You are working as a data analyst for a retail store chain. The management wants insights into their sales data to understand trends, identify top-selling products, and forecast future sales to optimize inventory management and marketing strategies.

Project Goals

1. **Data Collection**: Obtain a dataset containing historical sales data, including information such as date of sale, product ID, quantity sold, price, etc. You can search for open datasets online or simulate your own dataset.
2. **Data Preprocessing:** Clean the data by handling missing values, removing duplicates, and converting data types if necessary. Perform any necessary data transformations, such as calculating total sales amount for each transaction.
3. **Exploratory Data Analysis (EDA):** Conduct EDA to gain insights into the sales data. Explore trends over time, seasonality in sales, correlation between different variables (e.g., sales vs. price, sales vs. product category), and identify top-selling products or categories.
4. **Visualization:** Create visualizations using Matplotlib to present your findings from the EDA phase. This could include line plots to visualize sales trends over time, bar plots to show top-selling products or categories, and scatter plots to explore relationships between variables.

Introduction

In today's competitive retail landscape, data-driven decision-making is crucial for maintaining a competitive edge. As a data analyst for our retail store chain, I have been tasked with leveraging our extensive sales data to extract meaningful insights. The primary objective is to understand sales trends, identify top-selling products, and forecast future sales. These insights will be instrumental in optimizing inventory management and refining our marketing strategies to boost profitability and customer satisfaction.

This report is structured to provide a comprehensive analysis of our sales data. We will begin with an overview of current sales trends, examining patterns over different time periods and across various store locations. This will help us identify any seasonal trends or regional variations that can inform our inventory and marketing strategies.

Next, we will delve into product-level analysis, highlighting the topselling products and categories. Understanding which products drive the most revenue and their sales cycles will enable us to make informed decisions about stock levels, promotional efforts, and product placements.

Finally, we will employ forecasting techniques to predict future sales. Accurate sales forecasts are essential for effective inventory management, ensuring that we have the right products available at the right times, minimizing stockouts and overstock situations. This will not only reduce costs but also enhance customer satisfaction by ensuring product availability.

By harnessing the power of data analytics, this report aims to provide actionable insights that will guide our retail store chain towards more efficient operations, better customer experiences, and increased profitability.

1. ASSUMPTIONS

1. The data is normally distributed.

2. The dataset is accurate and represents the true sales data.

3. The location of the store is accurate.

4. Missing values and duplicates are to be handled by removal.

5. The sales data is comprehensive and includes all necessary fields for analysis.

2. Data Collection

The process of gathering and analyzing accurate data from various sources to find answers to research problems, trends and probabilities, etc., to evaluate possible outcomes is Known as Data Collection. Knowledge is power, information is knowledge, and data is information in digitized form, at least as defined in IT. Hence, data is power. But before you can leverage that data into a successful strategy for your organization or business, you need to gather it.

2.1 Sources

Data for this project was sourced from Kaggle, which provides a variety of open datasets related to retail sales. The dataset includes:

- Date
- Customer_ID
- Transaction_ID
- SKU_Category
- SKU
- Quantity
- Sales_Amount

[2]:
```
data = pd.read_csv(r"C:\Users\admin\Downloads\scanner_data.csv\scanner_data.csv")
data.head()
```

[2]:

	Unnamed: 0	Date	Customer_ID	Transaction_ID	SKU_Category	SKU	Quantity	Sales_Amount
0	1	02/01/2016	2547	1	X52	0EM7L	1.0	3.13
1	2	02/01/2016	822	2	2ML	68BRQ	1.0	5.46
2	3	02/01/2016	3686	3	0H2	CZUZX	1.0	6.35
3	4	02/01/2016	3719	4	0H2	549KK	1.0	5.59
4	5	02/01/2016	9200	5	0H2	K8EHH	1.0	6.88

3. Data Preprocessing

Data processing involves transforming raw data into useful information. Stages of data processing include collection, filtering, sorting, and analysis. Data processing relies on various tools and techniques to ensure accurate, valuable output.

Key Steps in Data Processing:

1. Data Cleaning
   - Identifying and correcting errors or inconsistencies in the data.
   - Removing duplicates, handling missing values, and standardizing data formats to ensure quality and reliability.

2. Data Transformation
   - Converting raw data into a structured and usable format.
   - Aggregating, sorting, filtering, and encoding data to highlight important aspects and make it analyzable.

3. Data Integration
   - Combining data from different sources to provide a comprehensive dataset.
   - Ensuring that data from various systems or files is merged correctly and consistently.

4. Data Reduction
   - Reducing the volume of data while retaining its essential characteristics.
   - Techniques include data summarization, dimensionality reduction, and sampling.

5. Data Validation
   - Verifying that the processed data meets the necessary quality standards.
   - Ensuring accuracy, consistency, and completeness to maintain the integrity of the data.

```
[3]: data.dtypes
```

```
[3]: Unnamed: 0          int64
     Date               object
     Customer_ID         int64
     Transaction_ID      int64
     SKU_Category       object
     SKU                object
     Quantity          float64
     Sales_Amount      float64
     dtype: object
```

[4]:
```
data.describe(include='all').T
```

[4]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Unnamed: 0	131706.0	NaN	NaN	NaN	65853.5	38020.391614	1.0	32927.25	65853.5	98779.75	131706.0
Date	131706	363	23/09/2016	638	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Customer_ID	131706.0	NaN	NaN	NaN	12386.450367	6086.447552	1.0	7349.0	13496.0	17306.0	22625.0
Transaction_ID	131706.0	NaN	NaN	NaN	32389.604187	18709.901238	1.0	16134.0	32620.0	48548.0	64682.0
SKU_Category	131706	187	N8U	10913	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SKU	131706	5242	UNJKW	2007	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Quantity	131706.0	NaN	NaN	NaN	1.485311	3.872667	0.01	1.0	1.0	1.0	400.0
Sales_Amount	131706.0	NaN	NaN	NaN	11.981524	19.359699	0.02	4.23	6.92	12.33	707.73

Dropping unique identifiers

[5]:
```
df = data.drop(data.columns[0],axis=1)
df.head()
```

[5]:

	Date	Customer_ID	Transaction_ID	SKU_Category	SKU	Quantity	Sales_Amount
0	02/01/2016	2547	1	X52	0EM7L	1.0	3.13
1	02/01/2016	822	2	2ML	68BRQ	1.0	5.46
2	02/01/2016	3686	3	0H2	CZUZX	1.0	6.35
3	02/01/2016	3719	4	0H2	549KK	1.0	5.59
4	02/01/2016	9200	5	0H2	K8EHH	1.0	6.88

Convert to categories

[6]:
```
for col in ['SKU_Category','SKU']:
    df[col] = df[col].astype('category')

df.dtypes
```

[6]:
```
Date               object
Customer_ID         int64
Transaction_ID      int64
SKU_Category     category
SKU              category
Quantity          float64
Sales_Amount      float64
dtype: object
```

Convert Date to date time feature

[7]:
```
df.Date = pd.to_datetime(df.Date,format='%d/%m/%Y')
df.head()
```

[7]:

	Date	Customer_ID	Transaction_ID	SKU_Category	SKU	Quantity	Sales_Amount
0	2016-01-02	2547	1	X52	0EM7L	1.0	3.13
1	2016-01-02	822	2	2ML	68BRQ	1.0	5.46
2	2016-01-02	3686	3	0H2	CZUZX	1.0	6.35
3	2016-01-02	3719	4	0H2	549KK	1.0	5.59
4	2016-01-02	9200	5	0H2	K8EHH	1.0	6.88

```
[8]: # df["Year"] = df.Date.dt.year
df["Month"] = df.Date.dt.month
df["Day"] = df.Date.dt.day
df["Day_Of_Week"] = df.Date.dt.day_of_week
df["Day_Name"] = df.Date.dt.day_name().astype('category')
days = df[['Day_Name','Day_Of_Week']].drop_duplicates().reset_index()['Day_Name']
df["Day_Name"] = df["Day_Name"].cat.reorder_categories(days.to_list())
df["Month_Name"] = df.Date.dt.month_name().astype('category')
months = df[['Month_Name','Month']].drop_duplicates().reset_index()['Month_Name']
df["Month_Name"] = df["Month_Name"].cat.reorder_categories(months.to_list())

df.head()
```

[8]:

	Date	Customer_ID	Transaction_ID	SKU_Category	SKU	Quantity	Sales_Amount	Month	Day	Day_Of_Week	Day_Name	Month_Name
0	2016-01-02	2547	1	X52	0EM7L	1.0	3.13	1	2	5	Saturday	January
1	2016-01-02	822	2	2ML	68BRQ	1.0	5.46	1	2	5	Saturday	January
2	2016-01-02	3686	3	0H2	CZUZX	1.0	6.35	1	2	5	Saturday	January
3	2016-01-02	3719	4	0H2	549KK	1.0	5.59	1	2	5	Saturday	January
4	2016-01-02	9200	5	0H2	K8EHH	1.0	6.88	1	2	5	Saturday	January

```
[9]: df.describe(include='all').T
```

[9]:

	count	unique	top	freq	mean	min	25%	50%	75%	max	std
Date	131706	NaN	NaN	NaN	2016-07-04 18:00:03.608036096	2016-01-02 00:00:00	2016-04-05 00:00:00	2016-07-02 00:00:00	2016-10-07 00:00:00	2016-12-31 00:00:00	NaN
Customer_ID	131706.0	NaN	NaN	NaN	12386.450367	1.0	7349.0	13496.0	17306.0	22625.0	6086.447552
Transaction_ID	131706.0	NaN	NaN	NaN	32389.604187	1.0	16134.0	32620.0	48548.0	64682.0	18709.901238
SKU_Category	131706	187	N8U	10913	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SKU	131706	5242	UNJKW	2007	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Quantity	131706.0	NaN	NaN	NaN	1.485311	0.01	1.0	1.0	1.0	400.0	3.872667
Sales_Amount	131706.0	NaN	NaN	NaN	11.981524	0.02	4.23	6.92	12.33	707.73	19.359699
Month	131706.0	NaN	NaN	NaN	6.623791	1.0	4.0	7.0	10.0	12.0	3.472462
Day	131706.0	NaN	NaN	NaN	15.645506	1.0	8.0	16.0	23.0	31.0	8.509053
Day_Of_Week	131706.0	NaN	NaN	NaN	2.657548	0.0	1.0	3.0	4.0	6.0	1.820095
Day_Name	131706	7	Friday	23396	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Month_Name	131706	12	December	12535	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4. Exploratory Data Analysis (EDA) and Visualisation

Exploratory Data Analysis (EDA) is a data science method that helps data scientists understand raw data sets before modeling or transformation. EDA involves analyzing and summarizing data sets to identify patterns, anomalies, and relationships between variables. It can also help data scientists determine how to manipulate data sources to get the answers they need. EDA is often performed using statistical and visualization techniques, such as charts, plots, and infographics.

Here are some steps for performing EDA:

- Understand the problem and the data
- Import and inspect the data
- Handle missing values
- Explore data characteristics
- Perform data transformation
- Visualize data relationships
- Handle outliers

EDA

```
[10]: df.Customer_ID.nunique()

[10]: 22625

[37]: def barplot_with_percentage(data,x_label,y_label,figsize=(9,5)):
          plt.figure(figsize=figsize)
          g = sns.barplot(data,x=x_label,y=y_label)
          g.set_xticklabels(labels=data[x_label].to_list(), rotation=90)

          for p in g.patches:
              txt = str(p.get_height().round(2)) + '%'
              txt_x = p.get_x()
              txt_y = p.get_height() + 0.1
              g.text(txt_x,txt_y,txt)

          plt.show()
```

SKU Categories

[12]:
```
df.SKU_Category.value_counts(normalize=True).head()
```

[12]:
```
SKU_Category
N8U    0.082859
R6E    0.038715
LPF    0.038434
P42    0.036718
U5F    0.034698
Name: proportion, dtype: float64
```

[13]:
```
df.SKU_Category.value_counts().tail()
```

[13]:
```
SKU_Category
7MA    3
U3N    2
2JO    1
OTK    1
QON    1
Name: count, dtype: int64
```

[14]:
```
df.SKU.value_counts().head()
```

[14]:
```
SKU
UNJKW    2007
COWU2     791
OV1P9     737
M6J9W     698
C6TXL     689
Name: count, dtype: int64
```

[15]:
```
df.SKU.value_counts().tail()
```

[15]:
```
SKU
TQGGG    1
TQ58U    1
TPKG7    1
FR26V    1
ZZX6K    1
Name: count, dtype: int64
```

Date

[16]:
```
df.Date.value_counts()
```

[16]:
```
Date
2016-09-23    638
2016-12-15    614
2016-09-22    606
2016-05-13    602
2016-12-16    594
             ...
2016-07-31    128
2016-01-03    111
2016-08-28    107
2016-12-24    100
2016-03-28     73
Name: count, Length: 363, dtype: int64
```

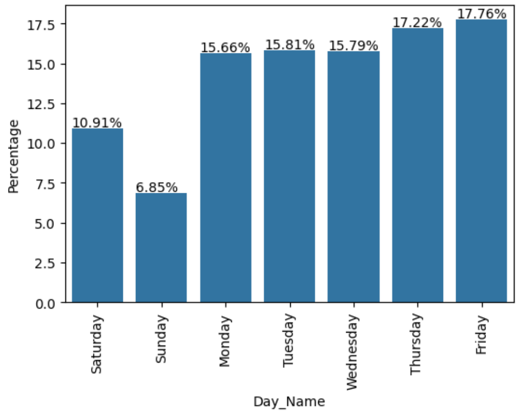
[17]:
```
by_day_of_week = (df.Day_Name.value_counts(normalize=True,sort=False) * 100).reset_index()
by_day_of_week.columns = ['Day_Name','Percentage']

by_day_of_week
```

[17]:

	Day_Name	Percentage
0	Saturday	10.907628
1	Sunday	6.851624
2	Monday	15.656842
3	Tuesday	15.810214
4	Wednesday	15.788195
5	Thursday	17.221691
6	Friday	17.763807

[18]:
```
barplot_with_percentage(by_day_of_week,'Day_Name','Percentage',figsize=(6,4))
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\2432454399.py:4: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
  g.set_xticklabels(labels=data[x_label].to_list(), rotation=90)
```



- Friday is the most popular day of the week accounting for 17.76% of all transactions followed by Thursday with 17.22% of all transactions.

- Weekends account for the lowest number of transactions with 6.85% on Sundays and 10.91% on Saturdays.

[19]:
```
totals_by_day = df.loc[:,['Day_Name','Sales_Amount']].groupby('Day_Name').sum('Sales_Amount').reset_index()
totals_by_day.columns = ['Day_Name','Total_Sales']
total = totals_by_day.Total_Sales.sum()
totals_by_day.Total_Sales = totals_by_day.Total_Sales / total * 100
totals_by_day
```

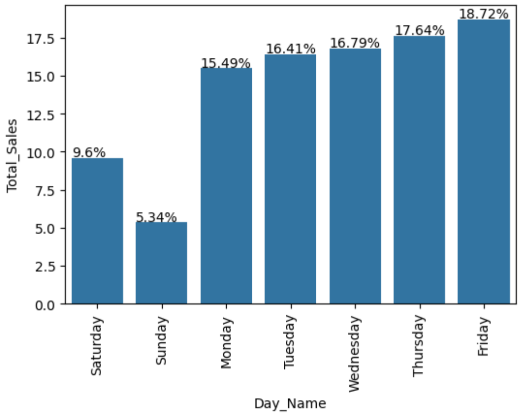
```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\89347385.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  totals_by_day = df.loc[:,['Day_Name','Sales_Amount']].groupby('Day_Name').sum('Sales_Amount').reset_index()
```

[19]:

	Day_Name	Total_Sales
0	Saturday	9.602983
1	Sunday	5.342246
2	Monday	15.492405
3	Tuesday	16.410937
4	Wednesday	16.793778
5	Thursday	17.641737
6	Friday	18.715914

```
[20]: barplot_with_percentage(totals_by_day,'Day_Name','Total_Sales',figsize=(6,4))
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\2432454399.py:4: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  g.set_xticklabels(labels=data[x_label].to_list(), rotation=90)
```



- Fridays saw the highest weekly total sales followed by Thursdays with 18.72% and 17.64% respectively.

- Sundays saw the lowest weekly totals sales followed by Saturdays with 5.34% and 9.6% respectively.

Month

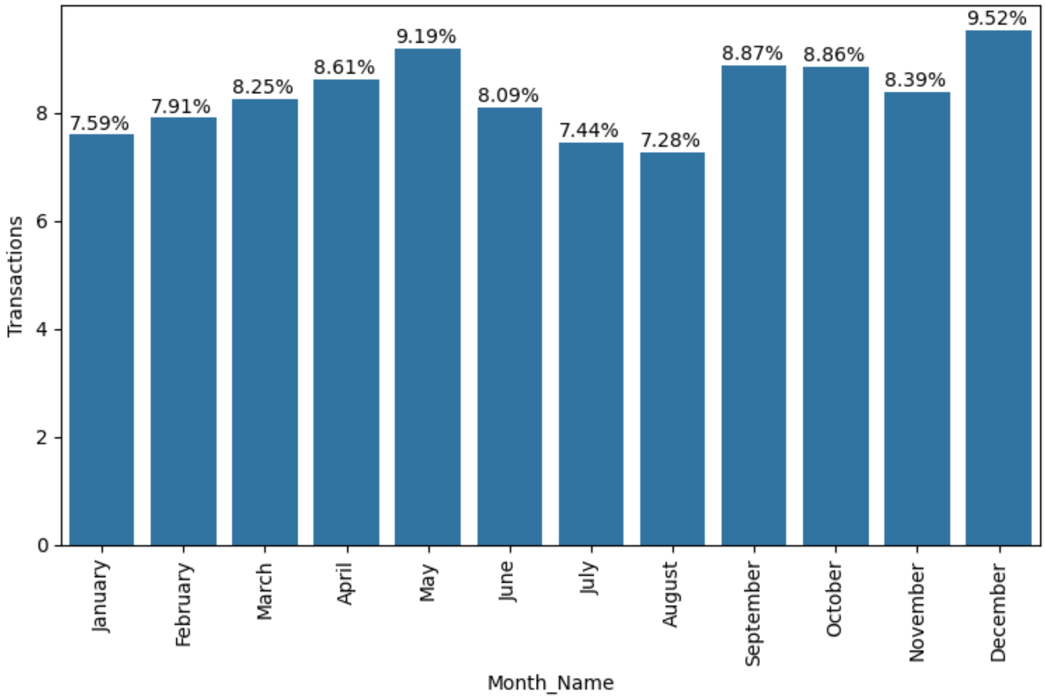
```
[21]: by_month = (df.Month_Name.value_counts(normalize=True,sort=False) * 100).reset_index()
by_month.columns = ['Month_Name','Transactions']

by_month
```

[21]:

	Month_Name	Transactions
0	January	7.590391
1	February	7.905486
2	March	8.249434
3	April	8.611605
4	May	9.189407
5	June	8.093025
6	July	7.443852
7	August	7.276054
8	September	8.872033
9	October	8.859126
10	November	8.392177
11	December	9.517410

```
[22]: barplot_with_percentage(by_month,'Month_Name','Transactions')
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\2432454399.py:4: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  g.set_xticklabels(labels=data[x_label].to_list(), rotation=90)
```



[23]:
```
totals_by_month = df.loc[:,['Month_Name','Sales_Amount']].groupby('Month_Name').sum('Sales_Amount').reset_index()
totals_by_month.columns = ['Month_Name','Total_Sales']
total = totals_by_month.Total_Sales.sum()
totals_by_month.Total_Sales = totals_by_month.Total_Sales / total * 100
totals_by_month
```

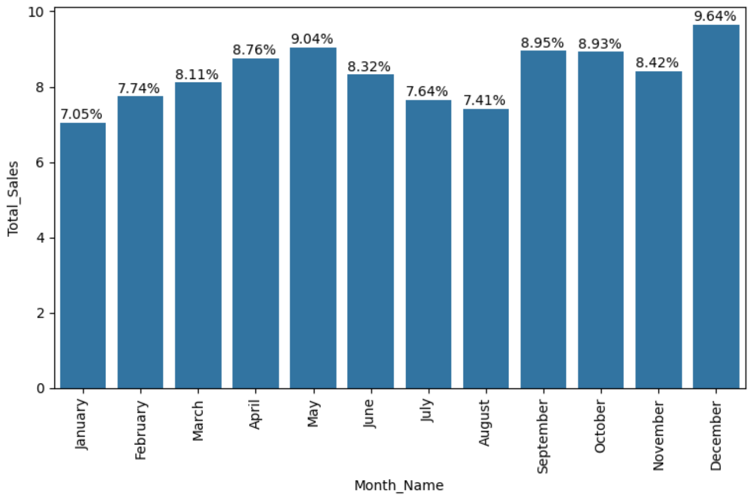
```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\1640589603.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True
in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  totals_by_month = df.loc[:,['Month_Name','Sales_Amount']].groupby('Month_Name').sum('Sales_Amount').reset_index()
```

[23]:

	Month_Name	Total_Sales
0	January	7.046740
1	February	7.738379
2	March	8.106553
3	April	8.755940
4	May	9.044130
5	June	8.320794
6	July	7.641890
7	August	7.408499
8	September	8.950344
9	October	8.925859
10	November	8.420791
11	December	9.640080

[24]:
```
barplot_with_percentage(totals_by_month,'Month_Name','Total_Sales')
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_8984\2432454399.py:4: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
  g.set_xticklabels(labels=data[x_label].to_list(), rotation=90)
```



- ➢ December saw highest in total sales followed by May accounting for 9.64% and 9.04% of total sales.

- ➢ January had the lowest in total sales followed by Auguest accounting for 7.05% and 7.41% of total sales.

On further analysis I found that,

- IW1, ORW, OTK, H4E, and QFK are among the categories with only 1 SKU.
- LPF has the most number of SKUs with 265 followed by Q4N with 123 SKUs.

Conclusion

From this analysis, I came onto the following conclusions:

- Friday is the most popular day of the week accounting for 17.76% of all transactions followed by Thursday with 17.22% of all transactions.
- Weekends account for the lowest number of transactions with 6.85% on Sundays and 10.91% on Saturdays.
- Fridays saw the highest weekly total sales followed by Thursdays with 18.72% and 17.64% respectively.
- Sundays saw the lowest weekly totals sales followed by Saturdays with 5.34% and 9.6% respectively.
- December saw highest in total sales followed by May accounting for 9.64% and 9.04% of total sales.
- January had the lowest in total sales followed by Auguest accounting for 7.05% and 7.41% of total sales.
- 11535 customers have only made 1 transaction
- Most transactions a single customer has made is 99

- IW1, ORW, OTK, H4E, and QFK are among the categories with only 1 SKU
- LPF has the most number of SKUs with 265 followed by Q4N with 123 SKUs

Final Decision

This analysis has provided valuable insights to drive strategic decisions, improve forecasting, optimize inventory, and implement targeted marketing strategies. Continuous analysis and strategic adjustments will ensure sustained growth and a competitive edge in the market.

Some suggestions that the retailer can work upon are:

- ❖ They can provide some special discounts to return customers as many customers made only 1 transaction.
- ❖ They can increase SKUs in IW1, ORW, OTK, H4E, and QFK.
- ❖ Run special offers on weekends to boost the sales.