DATA SCIENCE
PROJECT REPORT

MAHARISHI MARKANDESHWAR (DEEMED TO BE UNIVERSITY)
Mullana-Ambala, Haryana
(Established under Section 3 of UGC Act, 1956)
(Accredited by NAAC with Grade 'A++')

InveCareer
Invest in yourself for a brighter Career

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

Submitted to:
InveCareer

INDEX

ACKNOWLEDGEMENT

I , **Deepanshi Gupta**, want to thank everyone who has helped and supported me throughout my tasks.

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.

At last, I would like to thank **InveCareer** who gave me this opportunity to work on these projects.

Thank you all!

ABSTRACT

This report presents a comprehensive analysis of Student Performance, E-commerce Sales, Healthcare Data, Weather Data, Employee Turnover, aiming to derive actionable insights to optimize inventory management and marketing strategies. The analysis focuses on understanding sales trends, identifying top selling 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.

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.

ASSUMPTIONS

1. The data is normally distributed.
2. The dataset is accurate and represents the true data.
4. Missing values and duplicates are to be handled by removal.
5. The sales data is comprehensive and includes all necessary fields for analysis.

TASK 1

STUDENT PERFORMANCE ANALYSIS

PROBLEM STATEMENT:

Utilize a dataset containing student exam scores, demographic information, and study habits. Analyze the distribution of exam scores and identify trends. Investigate correlations between study time, demographic factors, and exam performance. Visualize the data using bar charts, scatter plots, and histograms. Provide recommendations for improving student performance based on the analysis.

Importing Libraries

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading Dataset

[3]:
```
df = pd.read_csv(r"C:\Users\admin\Downloads\Student_performance_data _.csv")
df.head()
```

[3]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	Extracurricular	Sports	Music	Volunteering	GPA
0	1001	17	1	0	2	19.833723	7	1	2	0	0	1	0	2.929196
1	1002	18	0	0	1	15.408756	0	0	1	0	0	0	0	3.042915
2	1003	15	0	2	3	4.210570	26	0	2	0	0	0	0	0.112602
3	1004	17	1	0	3	10.028829	14	0	3	1	0	0	0	2.054218
4	1005	17	1	0	2	4.672495	17	1	3	0	0	0	0	1.288061

Data Cleaning

[4]:
```
df.dtypes
```

[4]:
```
StudentID              int64
Age                    int64
Gender                 int64
Ethnicity              int64
ParentalEducation      int64
StudyTimeWeekly      float64
Absences               int64
Tutoring               int64
ParentalSupport        int64
Extracurricular        int64
Sports                 int64
Music                  int64
Volunteering           int64
GPA                  float64
GradeClass           float64
dtype: object
```

[5]:
```
df.describe(include='all').T
```

[5]:

	count	mean	std	min	25%	50%	75%	max
StudentID	2392.0	2196.500000	690.655244	1001.000000	1598.750000	2196.500000	2794.250000	3392.000000
Age	2392.0	16.468645	1.123798	15.000000	15.000000	16.000000	17.000000	18.000000
Gender	2392.0	0.510870	0.499986	0.000000	0.000000	1.000000	1.000000	1.000000
Ethnicity	2392.0	0.877508	1.028476	0.000000	0.000000	0.000000	2.000000	3.000000
ParentalEducation	2392.0	1.746237	1.000411	0.000000	1.000000	2.000000	2.000000	4.000000
StudyTimeWeekly	2392.0	9.771992	5.652774	0.001057	5.043079	9.705363	14.408410	19.978094
Absences	2392.0	14.541388	8.467417	0.000000	7.000000	15.000000	22.000000	29.000000
Tutoring	2392.0	0.301421	0.458971	0.000000	0.000000	0.000000	1.000000	1.000000
ParentalSupport	2392.0	2.122074	1.122813	0.000000	1.000000	2.000000	3.000000	4.000000
Extracurricular	2392.0	0.383361	0.486307	0.000000	0.000000	0.000000	1.000000	1.000000
Sports	2392.0	0.303512	0.459870	0.000000	0.000000	0.000000	1.000000	1.000000
Music	2392.0	0.196906	0.397744	0.000000	0.000000	0.000000	0.000000	1.000000
Volunteering	2392.0	0.157191	0.364057	0.000000	0.000000	0.000000	0.000000	1.000000
GPA	2392.0	1.906186	0.915156	0.000000	1.174803	1.893393	2.622216	4.000000
GradeClass	2392.0	2.983696	1.233908	0.000000	2.000000	4.000000	4.000000	4.000000

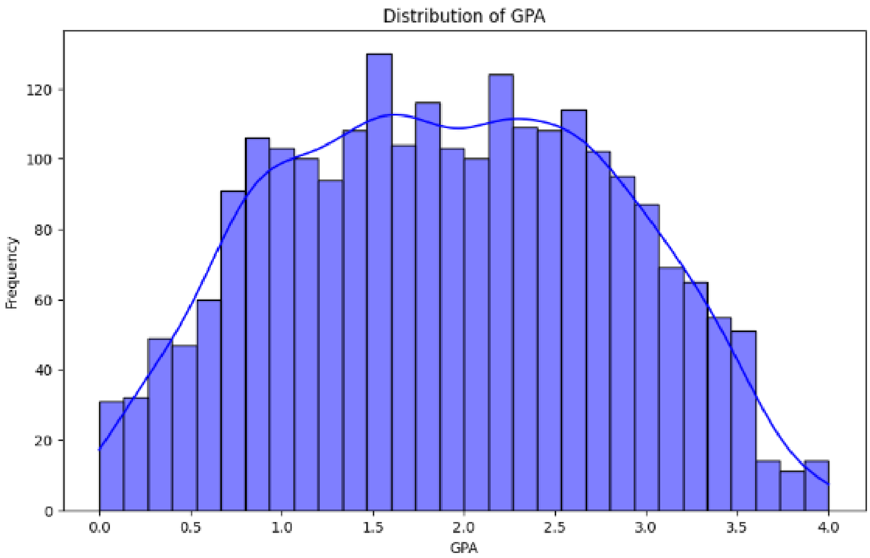
Checking Null Values

[8]:
```
df.isnull().sum()
```

[8]:
```
StudentID            0
Age                  0
Gender               0
Ethnicity            0
ParentalEducation    0
StudyTimeWeekly      0
Absences             0
Tutoring             0
ParentalSupport      0
Extracurricular      0
Sports               0
Music                0
Volunteering         0
GPA                  0
GradeClass           0
dtype: int64
```

Exploratory Data Analysis and Visualisation

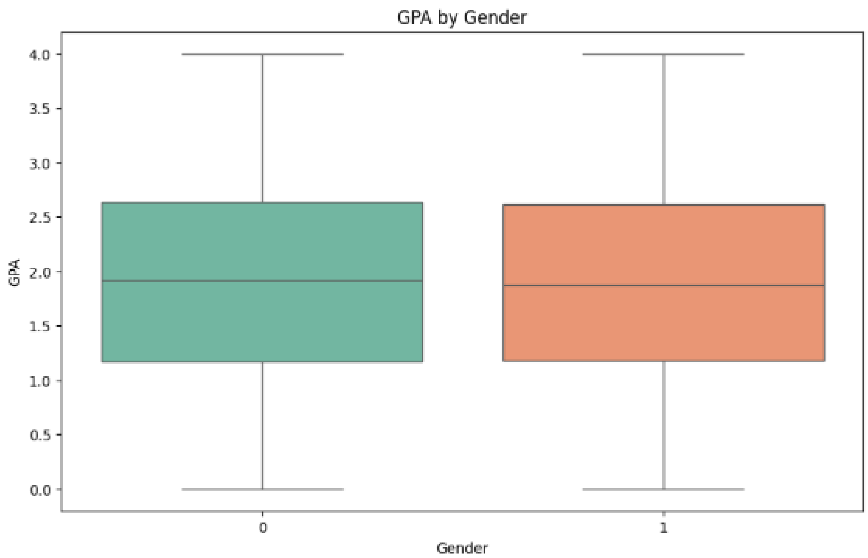
[11]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['GPA'], kde=True, bins=30, color='blue')
plt.title('Distribution of GPA')
plt.xlabel('GPA')
plt.ylabel('Frequency')
plt.show()
```



Above graph shows that most of the students scored GPA greater than 1.5 with most ranging between 1.5 and 2.5 and the no. decreasing as it reached 4.0.

[12]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='GPA', data=df, palette='Set2')
plt.title('GPA by Gender')
plt.xlabel('Gender')
plt.ylabel('GPA')
plt.show()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_16956\2646191905.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Gender', y='GPA', data=df, palette='Set2')
```

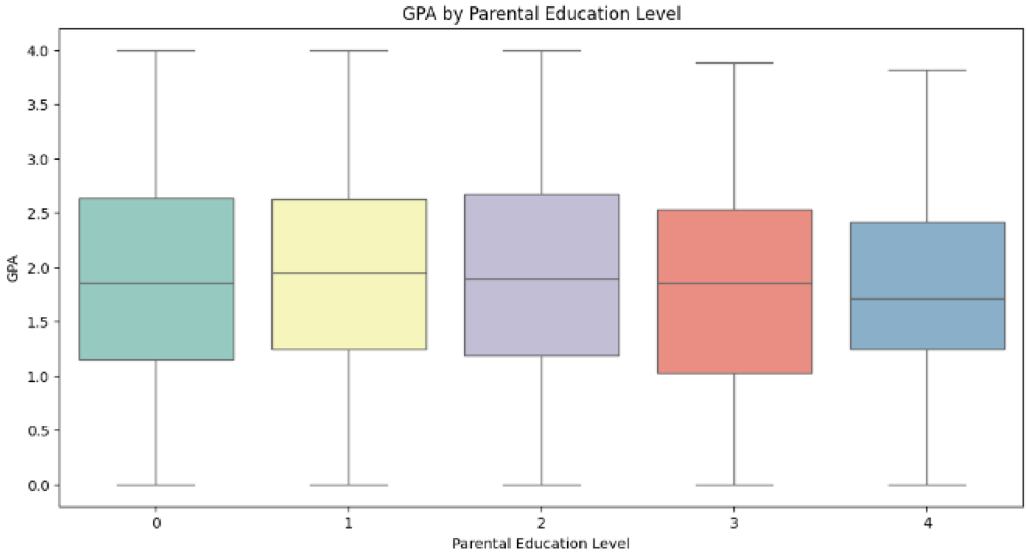


Above box plot shows that median GPA value of males is greater than females though they have the same minimum and maximum value.

[13]:

```python
plt.figure(figsize=(12, 6))
sns.boxplot(x='ParentalEducation', y='GPA', data=df, palette='Set3')
plt.title('GPA by Parental Education Level')
plt.xlabel('Parental Education Level')
plt.ylabel('GPA')
plt.show()
```

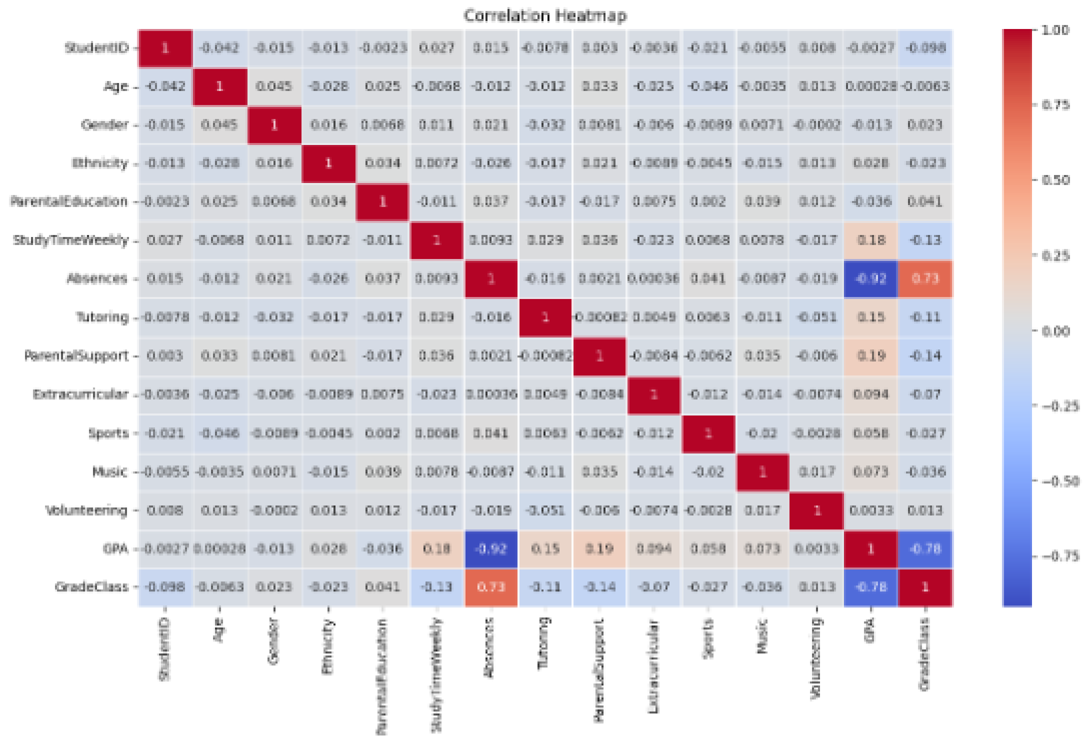
```
C:\Users\admin\AppData\Local\Temp\ipykernel_16956\853625335.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='ParentalEducation', y='GPA', data=df, palette='Set3')
```



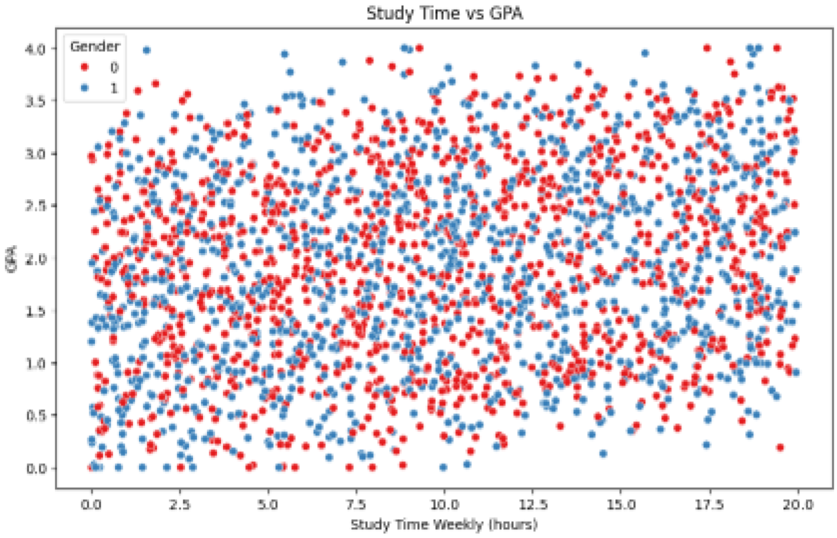
Children of the parents educated upto 0,1 and 2 level have scored better GPA as compared to level 3 and 4.

[14]:

```python
plt.figure(figsize=(14, 8))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

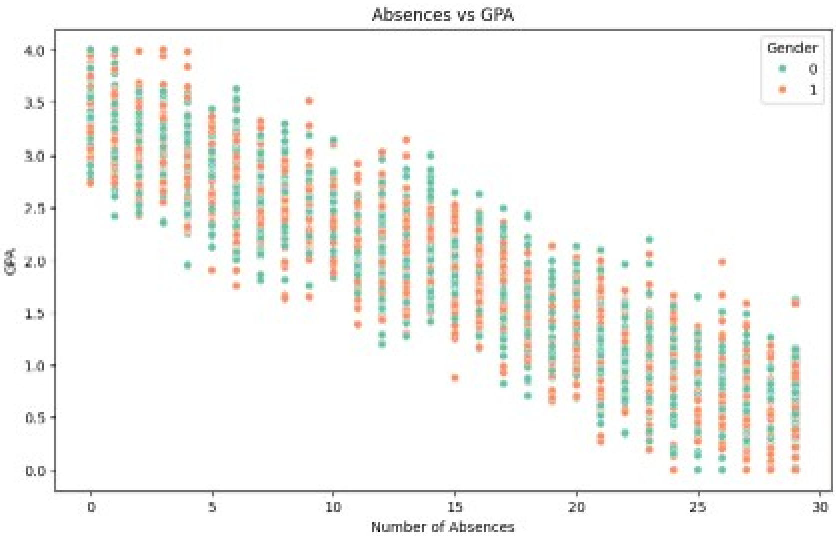


[15]:
```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x='StudyTimeWeekly', y='GPA', data=df, hue='Gender', palette='Set1')
plt.title('Study Time vs GPA')
plt.xlabel('Study Time Weekly (hours)')
plt.ylabel('GPA')
plt.show()
```




```
More study time means better GPA.
```

[16]:
```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Absences', y='GPA', data=df, hue='Gender', palette='Set2')
plt.title('Absences vs GPA')
plt.xlabel('Number of Absences')
plt.ylabel('GPA')
plt.show()
```




```
Students having less absences scored better than those having more absences.
```

CONCLUSION

1. Most of the students scored GPA greater than 1.5 with most ranging between 1.5 and 2.5 and the no. decreasing as it reached 4.0.

2. Median GPA value of males is greater than females.

3. Children of the parents educated upto 0,1 and 2 level have scored better GPA as compared to level 3 and 4.

4. More study time means better GPA.

5. Students having less absences scored better than those having more absences.

RECOMMENDATIONS

1. Absences should be less.
2. More time should be given to study.
3. Parents should be educated atleast upto 1^{st} or 2^{nd} level.

TASK 2

E-COMMERCE SALES ANALYSIS

PROBLEM STATEMENT:

Work with a dataset of e-commerce transactions including product details, prices,and purchase timestamps.

Analyze sales trends over time and identify peak selling periods. Explore the distribution of product prices and customer spending habits.

Segment customers based on purchasing behavior or demographic information. Generate insights to optimize product offerings and marketing strategies.

Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading Dataset

[13]:
```
df = pd.read_csv(r"e-commerce_sales_data.csv", encoding='unicode_escape')
df.head()
```

[13]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

Data Cleaning

[14]:
```
df.dtypes
```

[14]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

[15]:
```
df.describe(include='all').T
```

[15]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
InvoiceNo	541909	25900	573585	1114	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockCode	541909	4070	85123A	2313	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Description	540455	4223	WHITE HANGING HEART T-LIGHT HOLDER	2369	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Quantity	541909.0	NaN	NaN	NaN	9.55225	218.081158	-80995.0	1.0	3.0	10.0	80995.0
InvoiceDate	541909	23260	10/31/2011 14:41	1114	NaN	NaN	NaN	NaN	NaN	NaN	NaN
UnitPrice	541909.0	NaN	NaN	NaN	4.611114	96.759853	-11062.06	1.25	2.08	4.13	38970.0
CustomerID	406829.0	NaN	NaN	NaN	15287.69057	1713.600303	12346.0	13953.0	15152.0	16791.0	18287.0
Country	541909	38	United Kingdom	495478	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[17]:
```
df.isnull().sum()
```

[17]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

Exploratory Data Analysis

[33]:
```
# Displaying columns containing at least one value of 0 (zero).
df.columns[df.isin([0]).any()]
```

[33]:
```
Index([], dtype='object')
```

[18]:
```
# Calculating the mean of 'UnitPrice'
unit_price_mean = df['UnitPrice'].mean()
```

[19]:
```
df['UnitPrice'] = df['UnitPrice'].replace(0, unit_price_mean)
```

[22]:
```
df['Description'].fillna('No description available')
df.isnull().sum()
```

[22]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

[23]:
```
# Replace missing value in 'CustomerID'
def fill_customer_id(row):
    if pd.isna(row['CustomerID']):
        return f"Customer of Invoice Nº {row['InvoiceNo']}"
    else:
        return row['CustomerID']

df['CustomerID'] = df.apply(fill_customer_id, axis=1)
df.isnull().sum()
```

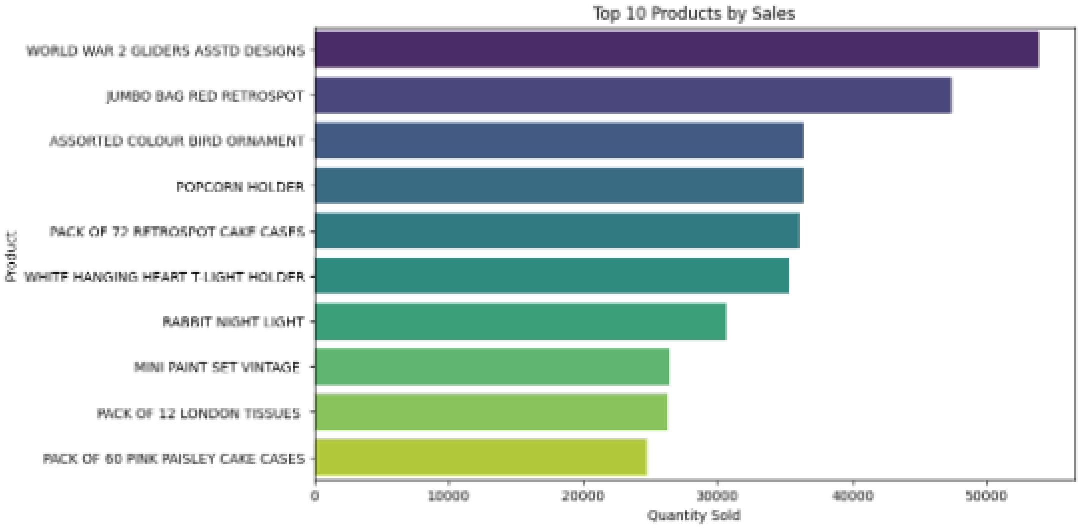
[23]:
```
InvoiceNo         0
StockCode         0
Description    1454
Quantity          0
InvoiceDate       0
UnitPrice         0
CustomerID        0
Country           0
dtype: int64
```

[24]:
```
#Change the 'InvoiceDate' column to datetime objects.
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   541909 non-null  object
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 33.1+ MB
```

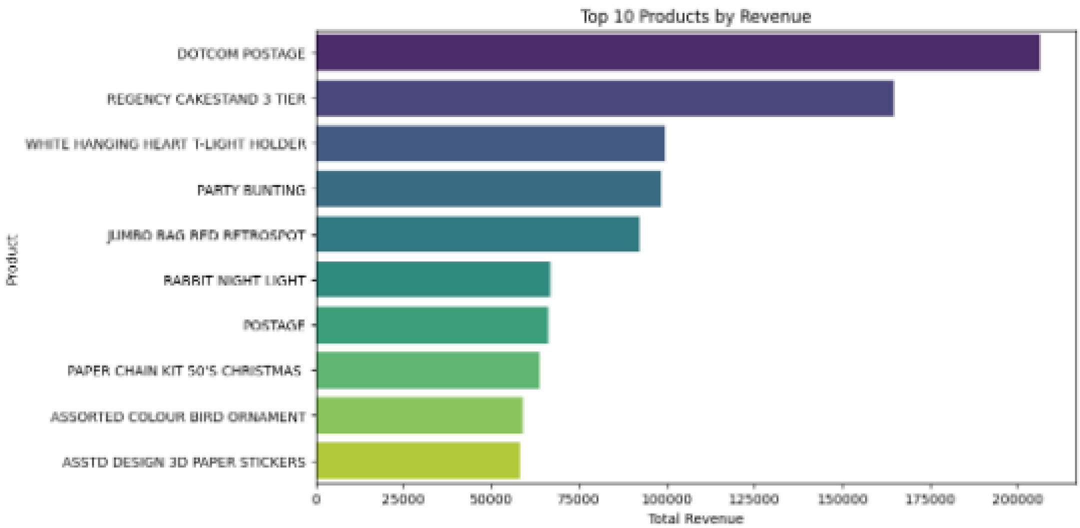
Data Visualisation

[26]:
```
sales_by_product = df.groupby('Description')['Quantity'].sum().sort_values(ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=sales_by_product.values, y=sales_by_product.index, palette='viridis')
plt.xlabel('Quantity Sold')
plt.ylabel('Product')
plt.title('Top 10 Products by Sales')
plt.show()
```



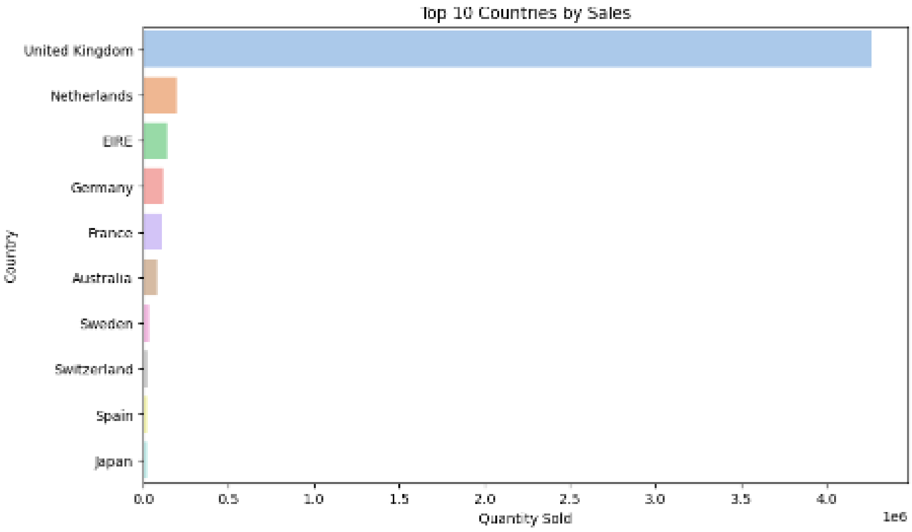
The best-selling products include "WORLD WAR 2 GLIDERS ASSTD DESIGNS" and "JUMBO BAG RED RETROSPOT"

[27]:
```
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']
top_revenue_products = df.groupby('Description')['TotalPrice'].sum().sort_values(ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x=top_revenue_products.values, y=top_revenue_products.index, palette='viridis')
plt.xlabel('Total Revenue')
plt.ylabel('Product')
plt.title('Top 10 Products by Revenue')
plt.show()
```



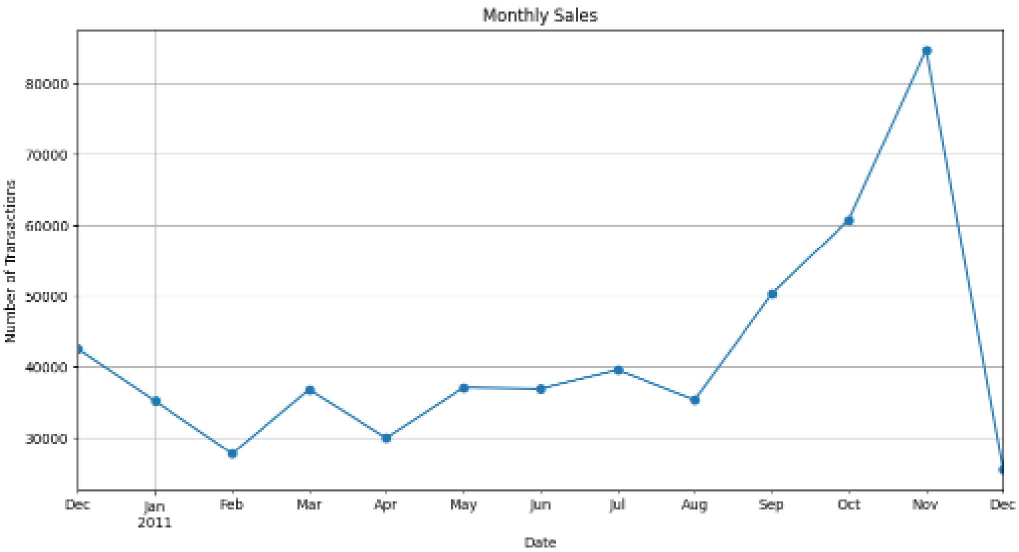
The products that generate the highest revenue are "DOTCOM POSTAGE" and "REGENCY CAKESTAND 3 TIER".

[28]:
```
sales_by_country = df.groupby('Country')['Quantity'].sum().sort_values(ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x=sales_by_country.values, y=sales_by_country.index, palette='pastel')
plt.xlabel('Quantity Sold')
plt.ylabel('Country')
plt.title('Top 10 Countries by Sales')
plt.show()
```



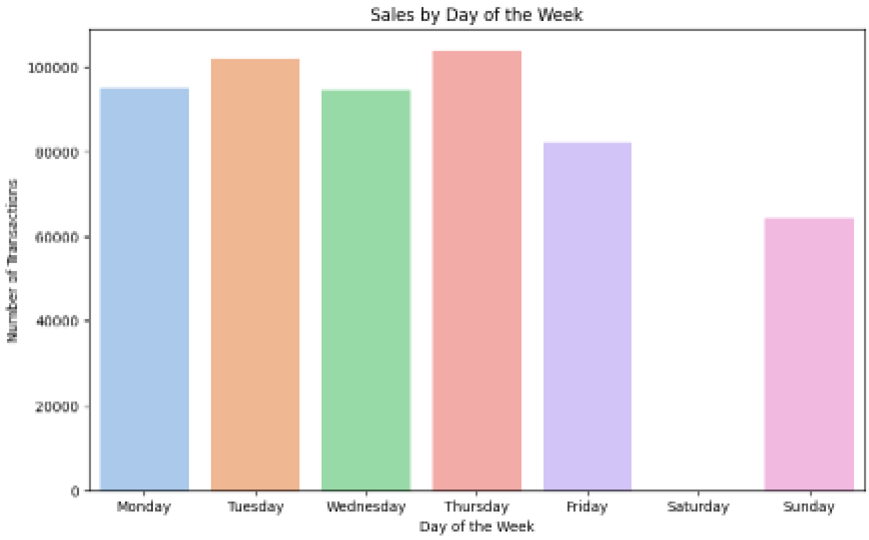
The United Kingdom leads in sales, followed by the Netherlands and EIRE. This data highlights the importance of the domestic market and potential opportunities in key international markets

[29]:
```
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')
monthly_sales = df.groupby('YearMonth').size()

plt.figure(figsize=(12, 6))
monthly_sales.plot(marker='o')
plt.xlabel('Date')
plt.ylabel('Number of Transactions')
plt.title('Monthly Sales')
plt.grid(True)
plt.show()
```



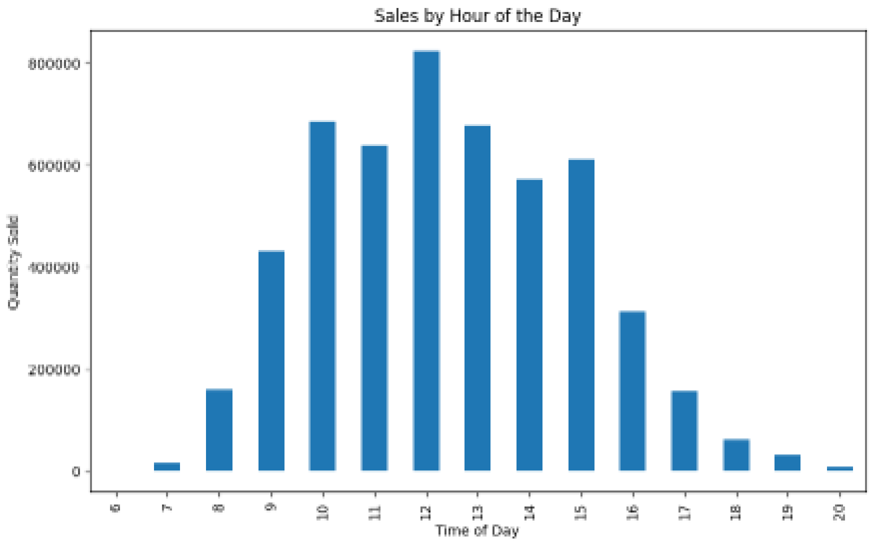
Monthly sales show a seasonal pattern with significant peaks from September to November. These peaks can be attributed to events such as holidays or successful promotional campaigns.

[30]:
```
df['DayOfWeek'] = df['InvoiceDate'].dt.day_name()
weekday_sales = df.groupby('DayOfWeek').size().reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])

plt.figure(figsize=(10, 6))
sns.barplot(x=weekday_sales.index, y=weekday_sales.values, palette='pastel')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Transactions')
plt.title('Sales by Day of the Week')
plt.show()
```



Sales show a relatively even distribution throughout the week, with a slight decrease on Fridays and Saturdays.

[31]:
```
df['Hour'] = df['InvoiceDate'].dt.hour
sales_by_hour = df.groupby('Hour')['Quantity'].sum()

plt.figure(figsize=(10, 6))
sales_by_hour.plot(kind='bar')
plt.xlabel('Time of Day')
plt.ylabel('Quantity Sold')
plt.title('Sales by Hour of the Day')
plt.show()
```



Peak sales hours are observed between 9:00 AM and 3:00 PM, with a peak at 12:00 PM.

CONCLUSION AND SUGGESTIONS

1. The best-selling products include "WORLD WAR 2 GLIDERS ASSTD DESIGNS" and "JUMBO BAG RED RETROSPOT". These items represent popular products that should be kept in inventory due to their high demand.

2. The products that generate the highest revenue are "DOTCOM POSTAGE" and "REGENCY CAKESTAND 3 TIER". These items not only have high sales volumes but also contribute significantly to the store's total revenue.

3. The United Kingdom leads in sales, followed by the Netherlands and EIRE. These data highlight the importance of the domestic market and potential opportunities in key international markets.

4. Monthly sales show a seasonal pattern with significant peaks from September to November. These peaks can be attributed to events such as holidays or successful promotional campaigns.

5. Sales show a relatively even distribution throughout the week, with a slight decrease on Fridays and Saturdays. This suggests that marketing and promotion strategies should consider adjustments to boost sales on these days.

6. Peak sales hours are observed between 9:00 AM and 3:00 PM, with a peak at 12:00 PM. This pattern indicates optimal times to implement promotional strategies and efficiently manage resources during periods of high demand.

TASK 3

HEALTHCARE DATA ANALYSIS

PROBLEM STATEMENT:

Analyze a dataset containing patient health records, including demographics, medical conditions, and treatment outcomes.

Investigate common health conditions and their prevalence within different demographic groups.

Identify factors that contribute to patient readmission rates or treatment success.

Visualize trends in patient health metrics over time.

Suggest potential interventions or improvements in healthcare delivery based on the analysis.

Importing Libraries

[30]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading Dataset

[31]:
```
df = pd.read_csv(r"C:\Users\admin\Downloads\healthcare_dataset.csv\healthcare_dataset.csv")
df
```

[31]:

	Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type	Discharge Date	Medication
0	Bobby JacksOn	30	Male	B-	Cancer	2024-01-31	Matthew Smith	Sons and Miller	Blue Cross	18856.281306	328	Urgent	2024-02-02	Paracetamol
1	LesLie TErRy	62	Male	A+	Obesity	2019-08-20	Samantha Davies	Kim Inc	Medicare	33643.327287	265	Emergency	2019-08-26	Ibuprofen
2	DaNnY sMitH	76	Female	A-	Obesity	2022-09-22	Tiffany Mitchell	Cook PLC	Aetna	27955.096079	205	Emergency	2022-10-07	Aspirin
3	andrEw waTtS	28	Female	O+	Diabetes	2020-11-18	Kevin Wells	Hernandez Rogers and Vang,	Medicare	37909.782410	450	Elective	2020-12-18	Ibuprofen
4	adrIENNE bEll	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.317814	458	Urgent	2022-10-09	Penicillin
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
55495	eLIZABeTH jaCkSOn	42	Female	O+	Asthma	2020-08-16	Joshua Jarvis	Jones-Thompson	Blue Cross	2650.714952	417	Elective	2020-09-15	Penicillin
55496	KYle pERez	61	Female	AB-	Obesity	2020-01-23	Taylor Sullivan	Tucker-Moyer	Cigna	31457.797307	316	Elective	2020-02-01	Aspirin
55497	HEATher WaNG	38	Female	B+	Hypertension	2020-07-13	Joe Jacobs DVM	and Mahoney Johnson Vasquez,	UnitedHealthcare	27620.764717	347	Urgent	2020-08-10	Ibuprofen
55498	JENNiFER JOneS	43	Male	O-	Arthritis	2019-05-25	Kimberly Curry	Jackson Todd and Castro,	Medicare	32451.092358	321	Elective	2019-05-31	Ibuprofen
55499	jAMES GARCiA	53	Female	O+	Arthritis	2024-04-02	Dennis Warren	Henry Sons and	Aetna	4010.134172	448	Urgent	2024-04-29	Ibuprofen

55500 rows × 15 columns

[32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Name                55500 non-null  object 
 1   Age                 55500 non-null  int64  
 2   Gender              55500 non-null  object 
 3   Blood Type          55500 non-null  object 
 4   Medical Condition   55500 non-null  object 
 5   Date of Admission   55500 non-null  object 
 6   Doctor              55500 non-null  object 
 7   Hospital            55500 non-null  object 
 8   Insurance Provider  55500 non-null  object 
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64  
 11  Admission Type      55500 non-null  object 
 12  Discharge Date      55500 non-null  object 
 13  Medication          55500 non-null  object 
 14  Test Results        55500 non-null  object 
dtypes: float64(1), int64(2), object(12)
memory usage: 6.4+ MB
```

[ ]:
```
df['Date of Admission'] = pd.to_datetime(df['Date of Admission'])
```

[ ]:
```
df['Discharge Date'] = pd.to_datetime(df['Discharge Date'])
```

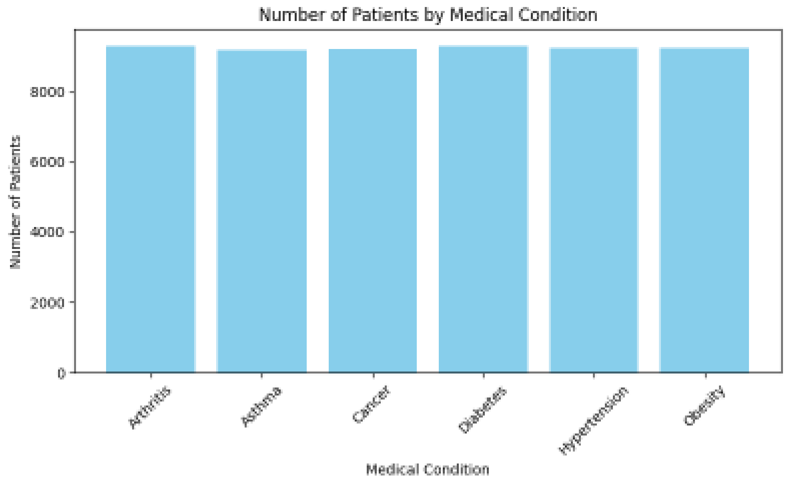
[29]:
```
df.drop(['Name','Room Number'],axis=1,inplace=True)
df.head()
```

[29]:

	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Admission Type	Discharge Date	Medication	Test Results
0	30	Male	B-	Cancer	2024-01-31	Matthew Smith	Sons and Miller	Blue Cross	18856.281306	Urgent	2024-02-02	Paracetamol	Normal
1	62	Male	A+	Obesity	2019-08-20	Samantha Davies	Kim Inc	Medicare	33643.327287	Emergency	2019-08-26	Ibuprofen	Inconclusive
2	76	Female	A-	Obesity	2022-09-22	Tiffany Mitchell	Cook PLC	Aetna	27955.096079	Emergency	2022-10-07	Aspirin	Normal
3	28	Female	O+	Diabetes	2020-11-18	Kevin Wells	Hernandez Rogers and Vang,	Medicare	37909.782410	Elective	2020-12-18	Ibuprofen	Abnormal
4	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.317814	Urgent	2022-10-09	Penicillin	Abnormal

[15]:
```
grouped_df = df.groupby('Medical Condition').size().reset_index(name='Number of Patients')
```

[16]:
```
plt.figure(figsize=(8, 5))
plt.bar(grouped_df['Medical Condition'], grouped_df['Number of Patients'], color='skyblue')
plt.xlabel('Medical Condition')
plt.ylabel('Number of Patients')
plt.title('Number of Patients by Medical Condition')
plt.xticks(rotation=45)  # Rotate the x labels if needed
plt.tight_layout()  # Adjust layout to make room for rotated labels
plt.show()
```



Almost all of the bars are equal, leaving no conclusion in analysing common medical condition. So to overcome this problem, I am going to divide the data into different age groups.

[17]:
```
bins = [0, 18, 35, 55, float('inf')]
labels = ['0-18', '18-35', '35-55', '55+']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
```

[18]:
```
grouped_df = df.groupby(['Medical Condition', 'Age Group'],observed=False).size().reset_index(name='Number of Patients')
grouped_df
```

[18]:

	Medical Condition	Age Group	Number of Patients
0	Arthritis	0-18	21
1	Arthritis	18-35	2269
2	Arthritis	35-55	2789
3	Arthritis	55+	4229
4	Asthma	0-18	20
5	Asthma	18-35	2277
6	Asthma	35-55	2714
7	Asthma	55+	4174
8	Cancer	0-18	21
9	Cancer	18-35	2273
10	Cancer	35-55	2684
11	Cancer	55+	4249
12	Diabetes	0-18	16
13	Diabetes	18-35	2239
14	Diabetes	35-55	2769
15	Diabetes	55+	4280
16	Hypertension	0-18	11
17	Hypertension	18-35	2259
18	Hypertension	35-55	2702
19	Hypertension	55+	4273
20	Obesity	0-18	27
21	Obesity	18-35	2288
22	Obesity	35-55	2756
23	Obesity	55+	4160

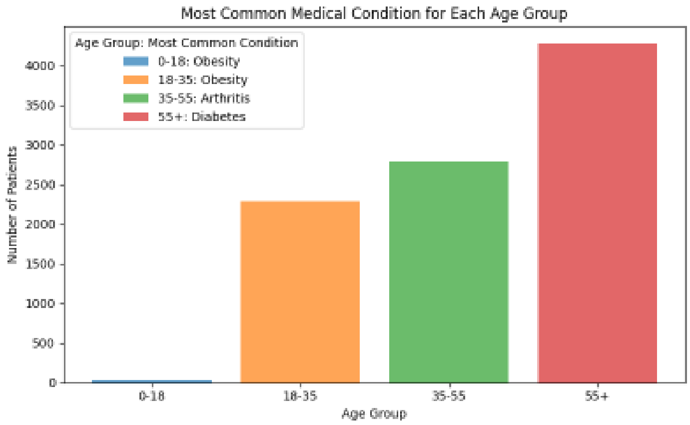
As we have an age grouped data and we can derive for which age group which is the most common medical condition.

[19]:
```
most_common = grouped_df.loc[grouped_df.groupby('Age Group', observed=False)['Number of Patients'].idxmax()]
most_common
```

[19]:

	Medical Condition	Age Group	Number of Patients
20	Obesity	0-18	27
21	Obesity	18-35	2288
2	Arthritis	35-55	2789
15	Diabetes	55+	4280

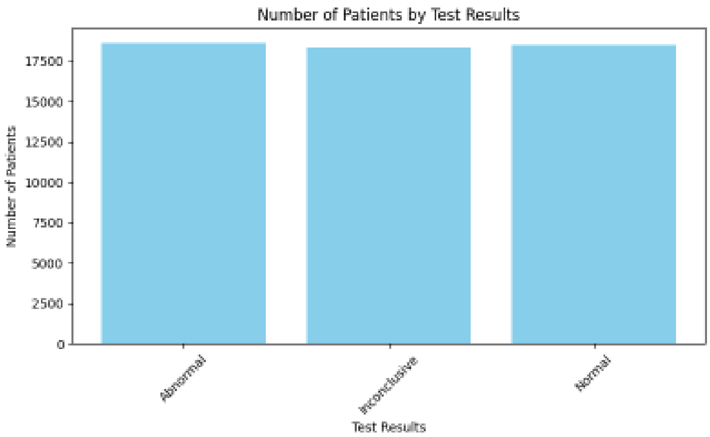
[28]:
```python
plt.figure(figsize=(8, 5))
for age_group, data in most_common.groupby('Age Group', observed=False):
    plt.bar(data['Age Group'], data['Number of Patients'], label=f"{age_group}: {data["Medical Condition"].values[0]}", alpha=0.7)
plt.xlabel('Age Group')
plt.ylabel('Number of Patients')
plt.title('Most Common Medical Condition for Different Age Group')
plt.legend(title='Age Group: Most Common Condition')
plt.tight_layout()
plt.show()
```




```
Above bar plot shows the most common medical condition in different age groups.
```

[33]:
```python
grouped_df1 = df.groupby('Test Results').size().reset_index(name='Number of Patients')
```

[35]:
```python
plt.figure(figsize=(8, 5))
plt.bar(grouped_df1['Test Results'], grouped_df1['Number of Patients'], color='skyblue')
plt.xlabel('Test Results')
plt.ylabel('Number of Patients')
plt.title('Number of Patients by Test Results')
plt.xticks(rotation=45)  # Rotate the x labels if needed
plt.tight_layout()  # Adjust layout to make room for rotated labels
plt.show()
```



[36]:
```python
bins = [0, 18, 35, 55, float('inf')]
labels = ['0-18', '18-35', '35-55', '55+']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
```

[37]:
```python
grouped_df1 = df.groupby(['Test Results', 'Age Group'],observed=False).size().reset_index(name='Number of Patients')
grouped_df1
```

[37]:

	Test Results	Age Group	Number of Patients
0	Abnormal	0-18	47
1	Abnormal	18-35	4483
2	Abnormal	35-55	5544
3	Abnormal	55+	8553
4	Inconclusive	0-18	33
5	Inconclusive	18-35	4443
6	Inconclusive	35-55	5459
7	Inconclusive	55+	8421
8	Normal	0-18	36
9	Normal	18-35	4679
10	Normal	35-55	5411
11	Normal	55+	8391

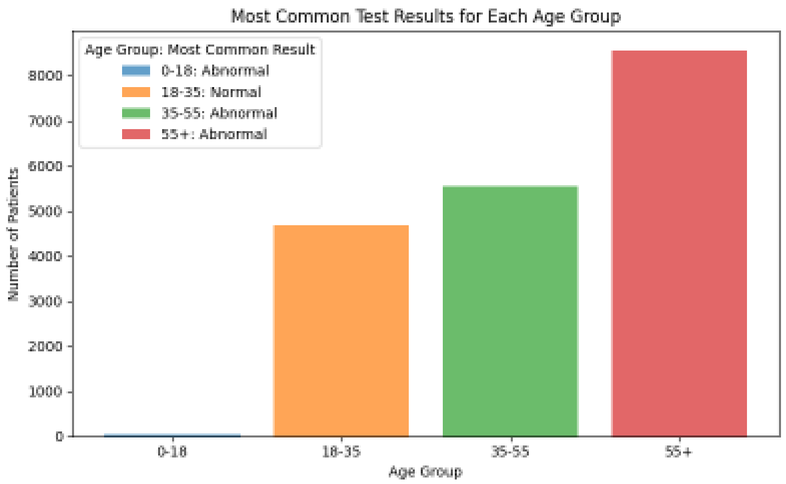
[39]:
```python
most_common1 = grouped_df1.loc[grouped_df1.groupby('Age Group', observed=False)["Number of Patients"].idxmax()]
most_common1
```

[39]:

	Test Results	Age Group	Number of Patients
0	Abnormal	0-18	47
9	Normal	18-35	4679
2	Abnormal	35-55	5544
3	Abnormal	55+	8553

[40]:

```
plt.figure(figsize=(8, 5))
for age_group, data in most_common1.groupby('Age Group', observed=False):
    plt.bar(data['Age Group'], data['Number of Patients'], label=f'{age_group}: {data["Test Results"].values[0]}', alpha=0.7)
plt.xlabel('Age Group')
plt.ylabel('Number of Patients')
plt.title('Most Common Test Results for Each Age Group')
plt.legend(title='Age Group: Most Common Result')
plt.tight_layout()
plt.show()
```




```
Above bar plot shows the common test result among various age groups.
```

CONCLUSION AND SUGGESTIONS

1. As we have analyzed that diabetes, arthritis, obesity are the common medical conditions among age groups 55+, 35-55 and 0-35. So such age groups should take preventive measures and also govt. should make people aware of the same.

2. From 'Most Common Test Results for Each Age Group' bar plot, we find that 18-35 age group has normal test result so they are not likely to be re-admitted, whereas 0-18 age group must be taken proper care of, and age groups 35-55 and 55+ have abnormal results so they are most likely to be re-admitted.

TASK 4

WEATHER DATA ANALYSIS

PROBLEM STATEMENT:

Use a dataset containing historical weather data, including temperature, precipitation, and wind speed. Analyze seasonal weather patterns and trends over time. Identify correlations between weather variables (e.g., temperature and precipitation). Visualize weather data using line graphs, heatmaps, or box plots. Extract insights about climate trends or extreme weather events from the analysis.

Importing Libraries

```
[1]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
```

Loading Dataset

```
[3]: df = pd.read_csv(r"C:\Users\admin\Downloads\weatherHistory.csv\weatherHistory.csv")
     df
```

[3]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

96453 rows × 12 columns

Data Cleaning

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  object
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: float64(8), object(4)
memory usage: 8.8+ MB
```

[9]:
```
df['Formatted Date'] = pd.to_datetime(df['Formatted Date'],utc=True)
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  datetime64[ns, UTC]
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: datetime64[ns, UTC](1), float64(8), object(3)
memory usage: 8.8+ MB
```

Checking Null Values

[11]:
```
df.isnull().sum()
```

[11]:
```
Formatted Date                0
Summary                       0
Precip Type                 517
Temperature (C)               0
Apparent Temperature (C)      0
Humidity                      0
Wind Speed (km/h)             0
Wind Bearing (degrees)        0
Visibility (km)               0
Loud Cover                    0
Pressure (millibars)          0
Daily Summary                 0
dtype: int64
```

Exploratory Data Analysis

[12]:
```
df.duplicated().sum()
```

[12]: 24

[13]:
```
df.drop_duplicates(inplace=True)
```

[15]:
```
df.drop('Loud Cover', axis=1, inplace=True)
```

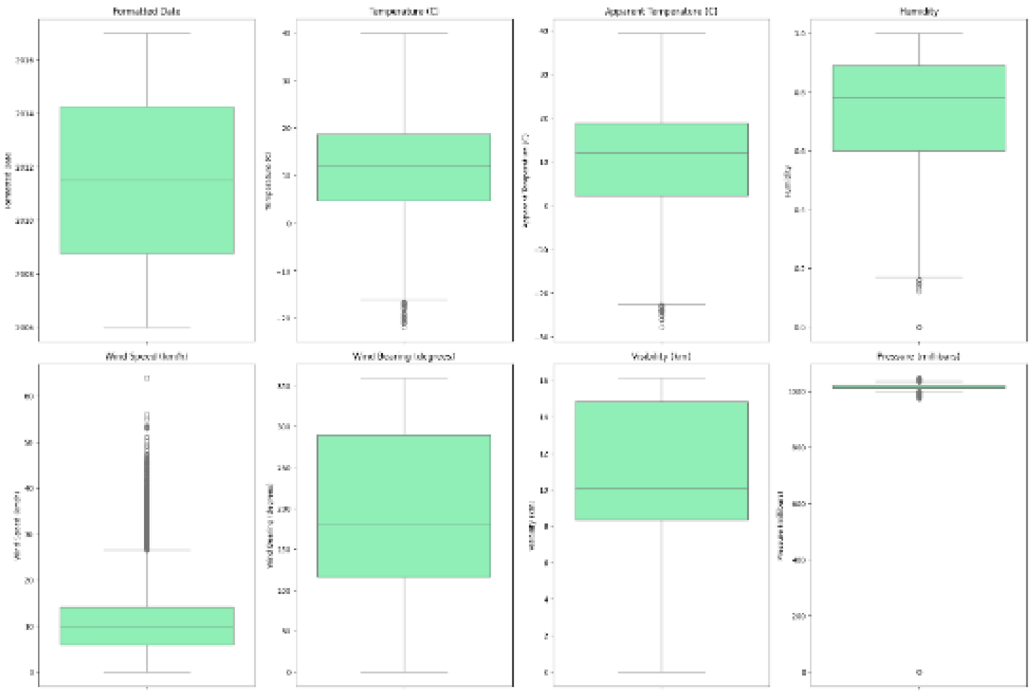
[16]:
```
df.columns
```

[16]:
```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
       'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Wind Bearing (degrees)', 'Visibility (km)', 'Pressure (millibars)',
       'Daily Summary'],
      dtype='object')
```

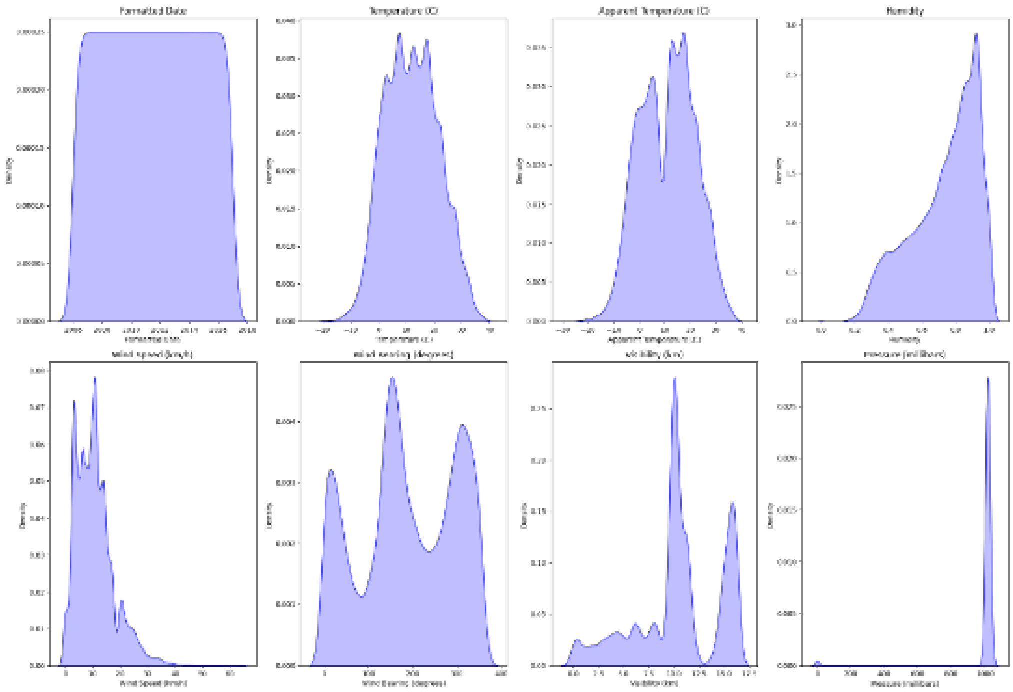
[18]:
```
df_num=df.select_dtypes(exclude=object)
df_cat=df.select_dtypes(include=object)
```

Data Visualisation

[21]:
```
plt.figure(figsize=(20,20))

re=1
for i in df_num.columns:
    plt.subplot(3,4,re)
    sns.boxplot(df[i], palette='rainbow')
    plt.title(i)
    re+=1
plt.tight_layout()
```



[22]:
```
plt.figure(figsize=(20,20))

re=1
for i in df_num.columns:
    plt.subplot(3,4,re)
    sns.kdeplot(df[i], fill=True,color='blue')
    plt.title(i)
    re+=1
plt.tight_layout()
```



[23]:
```
df.nunique()
```

[23]:
```
Formatted Date              96429
Summary                        27
Precip Type                     2
Temperature (C)              7574
Apparent Temperature (C)     8984
Humidity                       90
Wind Speed (km/h)            2484
Wind Bearing (degrees)        360
Visibility (km)               949
Pressure (millibars)         4979
Daily Summary                 214
dtype: int64
```

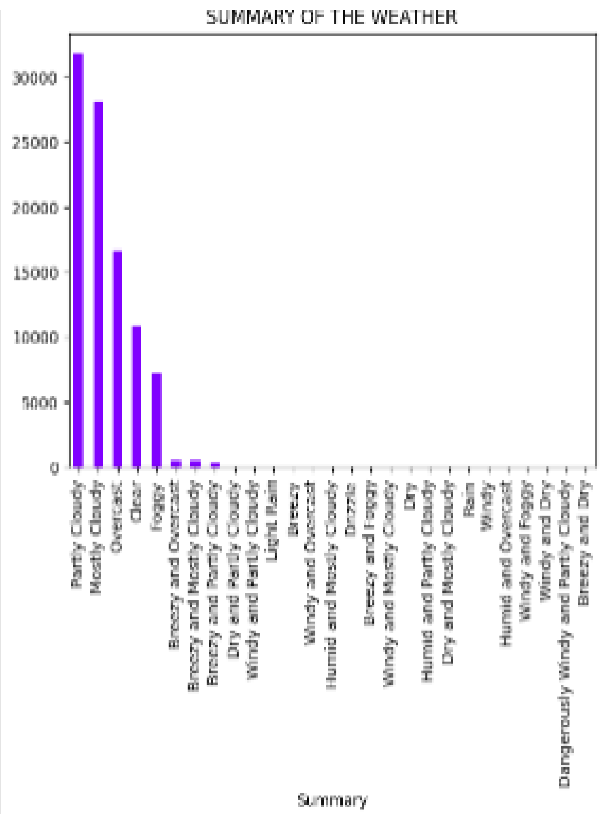
[24]:
```
df['Formatted Date'].nunique()
```

[24]: 96429

[25]:
```
df['Summary'].value_counts()
```

[25]:
```
Summary
Partly Cloudy                          31726
Mostly Cloudy                          28094
Overcast                               16597
Clear                                  10873
Foggy                                   7148
Breezy and Overcast                      528
Breezy and Mostly Cloudy                 516
Breezy and Partly Cloudy                 386
Dry and Partly Cloudy                     86
Windy and Partly Cloudy                   67
Light Rain                                63
Breezy                                    54
Windy and Overcast                        45
Humid and Mostly Cloudy                   40
Drizzle                                   39
Breezy and Foggy                          35
Windy and Mostly Cloudy                   35
Dry                                       34
Humid and Partly Cloudy                   17
Dry and Mostly Cloudy                     14
Rain                                      10
Windy                                      8
Humid and Overcast                         7
Windy and Foggy                            4
Windy and Dry                              1
Dangerously Windy and Partly Cloudy        1
Breezy and Dry                             1
Name: count, dtype: int64
```

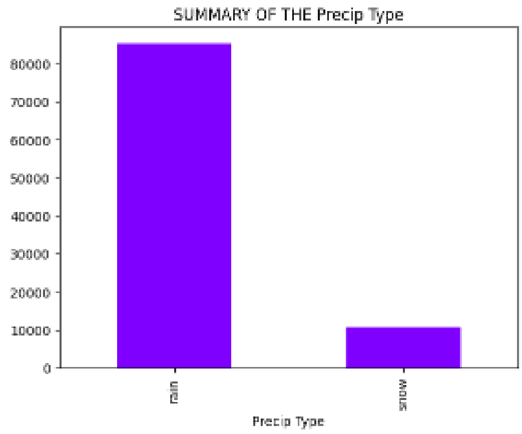
[26]:
```
df['Summary'].value_counts().plot(kind='bar', cmap='rainbow')
plt.title('SUMMARY OF THE WEATHER')
plt.show()
```



[27]:
```
df['Precip Type'].unique()
```

[27]: array(['rain', 'snow', nan], dtype=object)

[28]:
```
df['Precip Type'].value_counts().plot(kind='bar', cmap='rainbow')
plt.title('SUMMARY OF THE Precip Type')
plt.show()
```

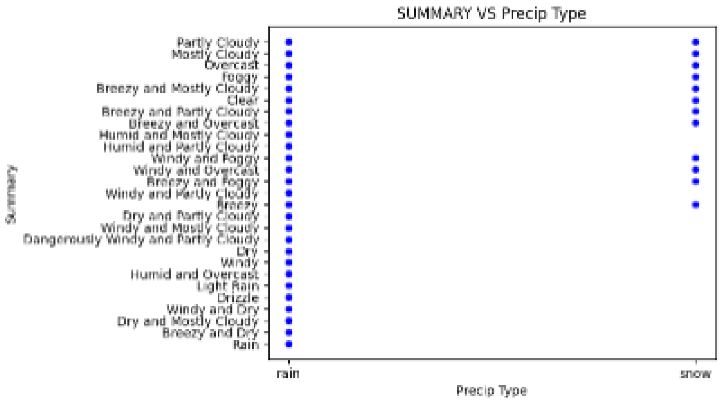


[29]:
```
df['Daily Summary'].value_counts()
```

[29]:
```
Daily Summary
Mostly cloudy throughout the day.                                                          20085
Partly cloudy throughout the day.                                                           9981
Partly cloudy until night.                                                                  6169
Partly cloudy starting in the morning.                                                      5184
Foggy in the morning.                                                                       4201
                                                                                            ...
Breezy starting overnight continuing until morning and foggy overnight.                       24
Mostly cloudy throughout the day and breezy starting overnight continuing until afternoon.    24
Partly cloudy starting in the morning and breezy starting in the afternoon continuing until evening.    24
Rain until afternoon.                                                                         24
Foggy starting overnight continuing until morning and breezy in the afternoon.                23
Name: count, Length: 214, dtype: int64
```

[30]:
```
df.drop('Daily Summary', axis=1, inplace=True)
```

[31]:
```
sns.scatterplot(y=df['Summary'], x=df['Precip Type'], color='blue')
plt.title('SUMMARY VS Precip Type')
plt.show()
```



[32]:
```
df_new_num=df.drop(['Formatted Date','Summary','Precip Type' ], axis=1)
```

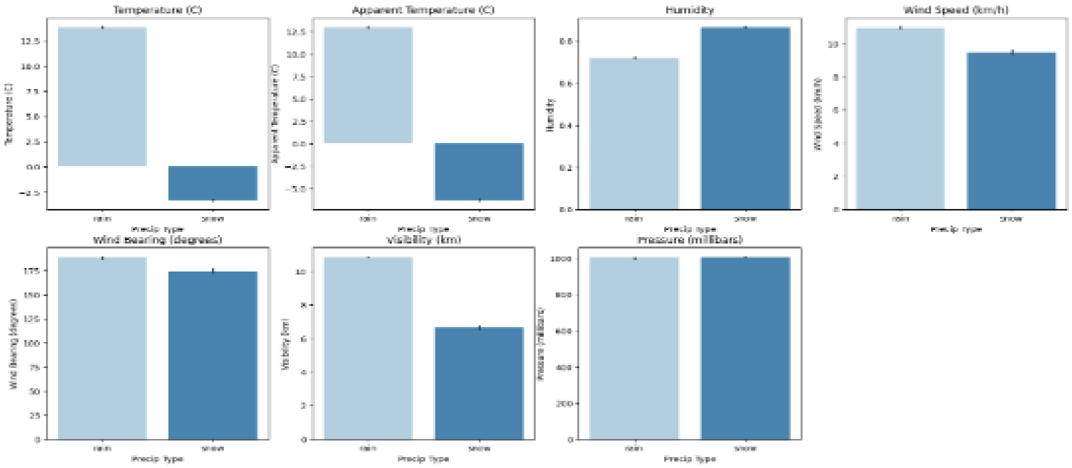
[33]:
```
df_new_num
```

[33]:

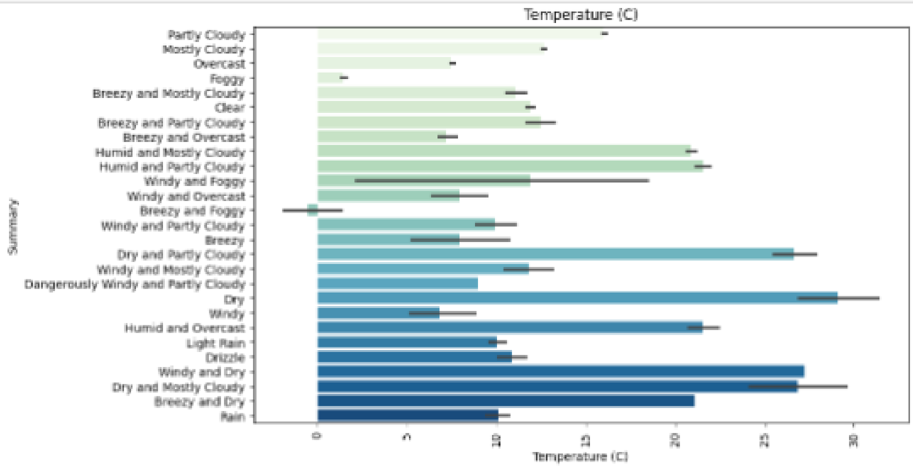
	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Pressure (millibars)
0	9.472222	7.388889	0.89	14.1197	251.0	15.8263	1015.13
1	9.355556	7.227778	0.86	14.2646	259.0	15.8263	1015.63
2	9.377778	9.377778	0.89	3.9284	204.0	14.9569	1015.94
3	8.288889	5.944444	0.83	14.1036	269.0	15.8263	1016.41
4	8.755556	6.977778	0.83	11.0446	259.0	15.8263	1016.51
...	...	...	...	...	...	...	...
96448	26.016667	26.016667	0.43	10.9963	31.0	16.1000	1014.36
96449	24.583333	24.583333	0.48	10.0947	20.0	15.5526	1015.16
96450	22.038889	22.038889	0.56	8.9838	30.0	16.1000	1015.66
96451	21.522222	21.522222	0.60	10.5294	20.0	16.1000	1015.95
96452	20.438889	20.438889	0.61	5.8765	39.0	15.5204	1016.16

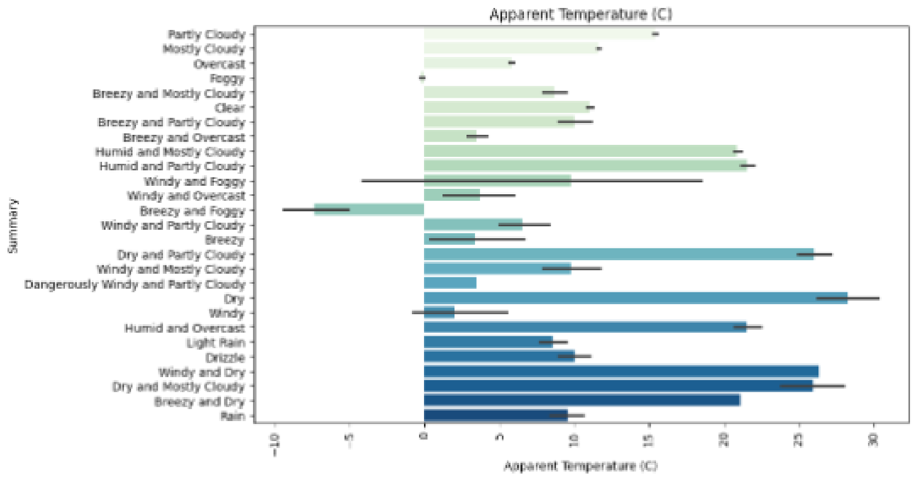
96429 rows × 7 columns

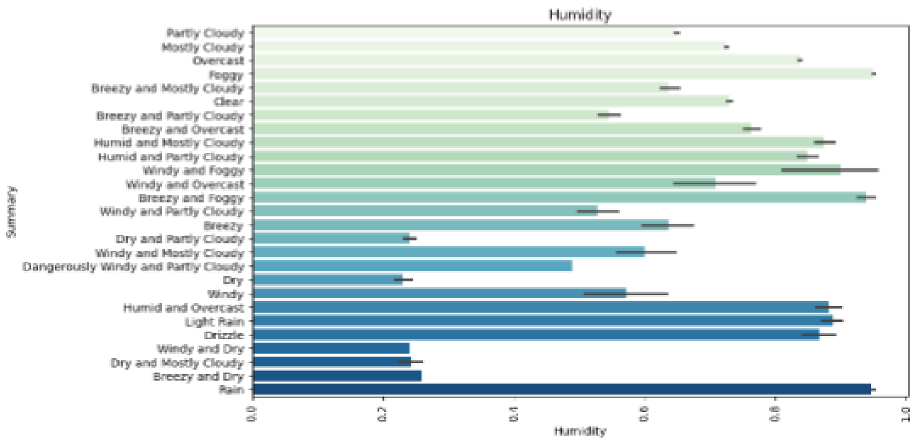
[34]:
```
plt.figure(figsize=(20,10))
re=1
for i in df_new_num.columns:
    plt.subplot(2,4,re)
    sns.barplot(x=df['Precip Type'], y=df_new_num[i],palette='Blues')
    re+=1
    plt.title(i)

plt.show()
```

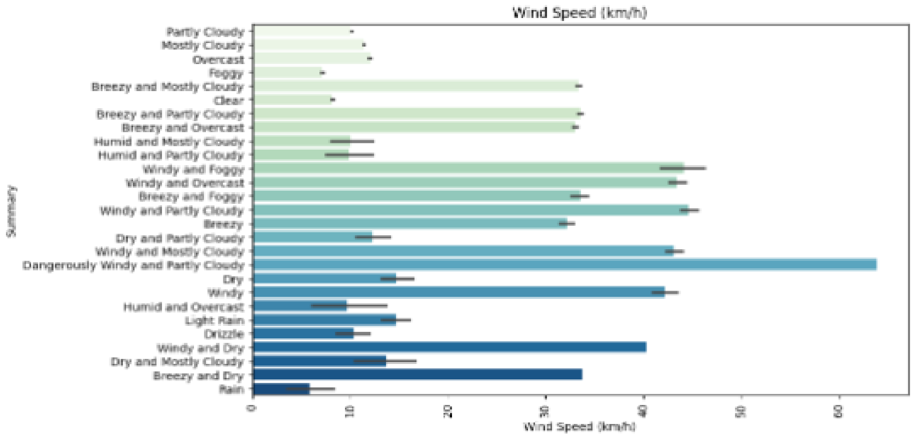


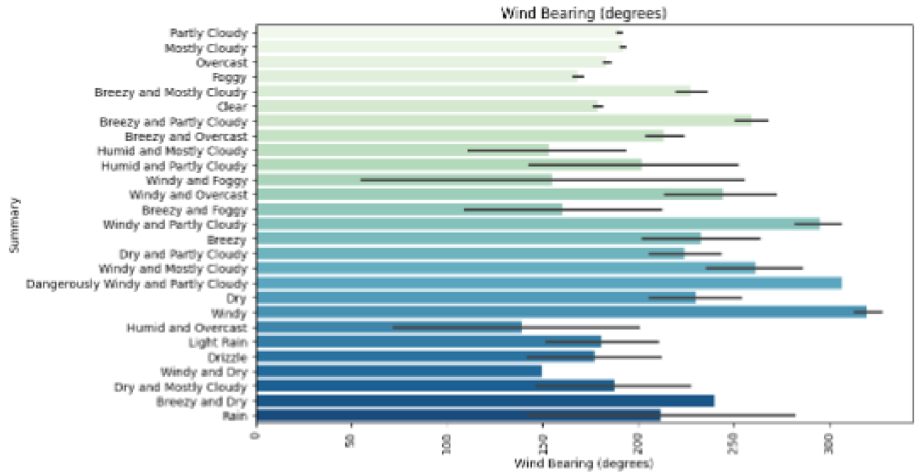
[35]:
```
plt.figure(figsize=(10,50))
re=1
for i in df_new_num.columns:
    plt.subplot(7,1,re)
    sns.barplot(y=df['Summary'], x=df_new_num[i],palette='GnBu')
    re+=1
    plt.title(i)
    plt.xticks(rotation=90)

plt.show()
plt.tight_layout()
```

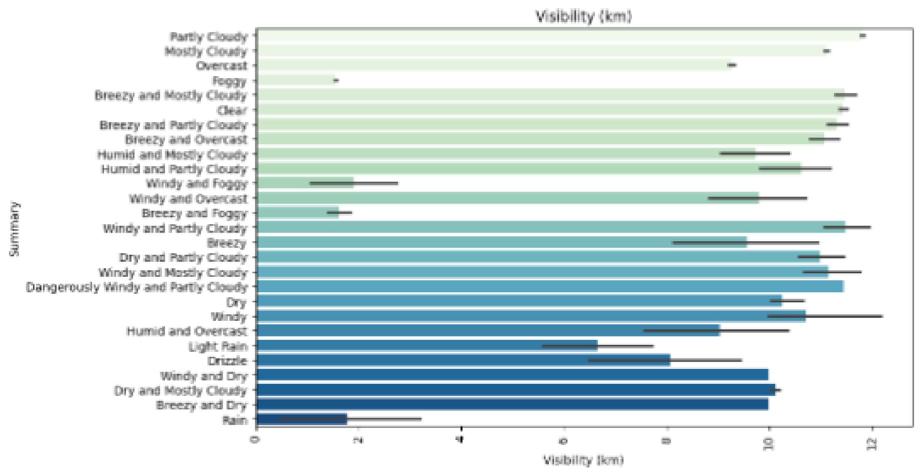


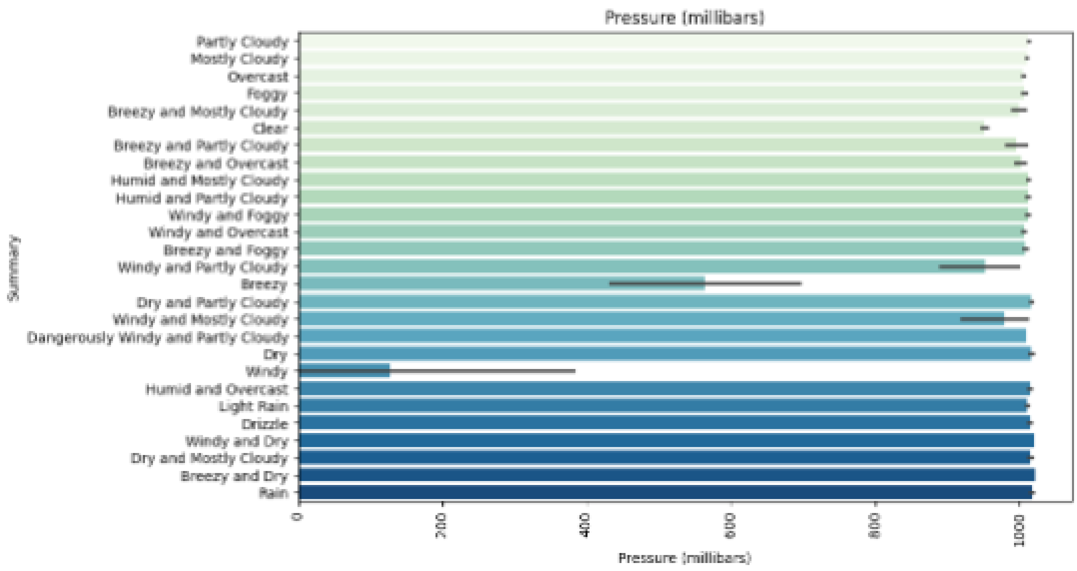



Humidity
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain
Summary
0.0
0.2
0.4
0.6
0.8
1.0
Humidity


Wind Speed (km/h)
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain
Summary
0
10
20
30
40
50
60
Wind Speed (km/h)


Wind Bearing (degrees)
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain
Summary
0
50
100
150
200
250
300
Wind Bearing (degrees)


Visibility (km)
Partly Cloudy
Mostly Cloudy
Overcast
Foggy
Breezy and Mostly Cloudy
Clear
Breezy and Partly Cloudy
Breezy and Overcast
Humid and Mostly Cloudy
Humid and Partly Cloudy
Windy and Foggy
Windy and Overcast
Breezy and Foggy
Windy and Partly Cloudy
Breezy
Dry and Partly Cloudy
Windy and Mostly Cloudy
Dangerously Windy and Partly Cloudy
Dry
Windy
Humid and Overcast
Light Rain
Drizzle
Windy and Dry
Dry and Mostly Cloudy
Breezy and Dry
Rain
Summary
0
2
4
6
8
10
12
Visibility (km)



<Figure size 640x480 with 0 Axes>

<u>CONCLUSION</u>

1. Year, and day has no effect on precipitation and weather summary while both mainly depends on the month, which is season.

2. As timezone changes there will be a change in temperature which may change overall daily weather condition.

3. Temperature on the dry day is the highest while temperature on the foggy abd breezy day is the lowest.

4. Pressure will be low on windy and breezy days.

TASK 6

EMPLOYEE TURNOVER ANALYSIS

PROBLEM STATEMENT:

Work with HR data containing employee demographics, performance metrics, and turnover rates. Analyze trends in employee turnover over time and identify common reasons for leaving. Explore correlations between turnover rates and factors such as job satisfaction or salary. Visualize turnover data using bar charts, pie charts, or heatmaps. Develop strategies to reduce employee turnover based on insights gained from the analysis.

Importing Libraries ¶

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Loading Dataset

[12]:
```
df = pd.read_csv(r"C:\Users\admin\Downloads\Employee Turnover Analytics Data set.csv")
df.head()
```

[12]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

Data Cleaning

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

[13]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
14994	0.40	0.57	2	151	3	0	1	0	support	low
14995	0.37	0.48	2	160	3	0	1	0	support	low
14996	0.37	0.53	2	143	3	0	1	0	support	low
14997	0.11	0.96	6	280	4	0	1	0	support	low
14998	0.37	0.52	2	158	3	0	1	0	support	low

[14]:
```
df.shape
```

[14]: (14999, 10)

[15]:
```
df.isna().sum()
```

```
[15]: satisfaction_level       0
      last_evaluation          0
      number_project           0
      average_montly_hours     0
      time_spend_company       0
      Work_accident            0
      left                     0
      promotion_last_5years    0
      sales                    0
      salary                   0
      dtype: int64
```

```
[16]: df.describe().T
```

[16]:

	count	mean	std	min	25%	50%	75%	max
satisfaction_level	14999.0	0.612834	0.248631	0.09	0.44	0.64	0.82	1.0
last_evaluation	14999.0	0.716102	0.171169	0.36	0.56	0.72	0.87	1.0
number_project	14999.0	3.803054	1.232592	2.00	3.00	4.00	5.00	7.0
average_montly_hours	14999.0	201.050337	49.943099	96.00	156.00	200.00	245.00	310.0
time_spend_company	14999.0	3.498233	1.460136	2.00	3.00	3.00	4.00	10.0
Work_accident	14999.0	0.144610	0.351719	0.00	0.00	0.00	0.00	1.0
left	14999.0	0.238083	0.425924	0.00	0.00	0.00	0.00	1.0
promotion_last_5years	14999.0	0.021268	0.144281	0.00	0.00	0.00	0.00	1.0

```
[17]: df['sales'].value_counts()
```

```
[17]: sales
      sales          4140
      technical      2720
      support        2229
      IT             1227
      product_mng     902
      marketing       858
      RandD           787
      accounting      767
      hr              739
      management      630
      Name: count, dtype: int64
```

Exploratory Data Analysis and Visualisation

```
[18]: df.rename(columns={'sales':'department'},inplace=True)
      df.columns
```

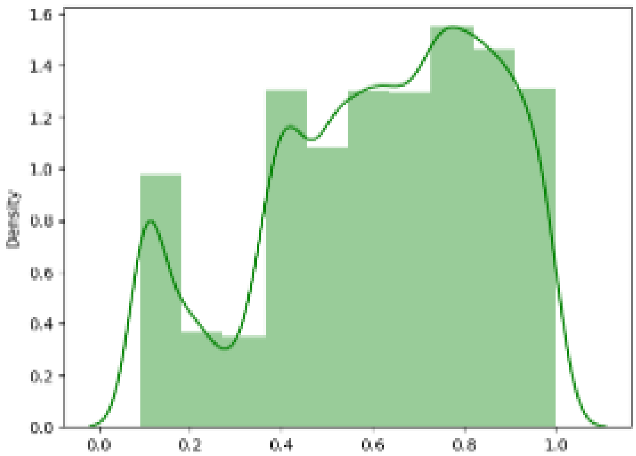
```
[18]: Index(['satisfaction_level', 'last_evaluation', 'number_project',
             'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
             'promotion_last_5years', 'department', 'salary'],
            dtype='object')
```

```
[19]: df['salary'].value_counts()
```

```
[19]: salary
      low       7316
      medium    6446
      high      1237
      Name: count, dtype: int64
```

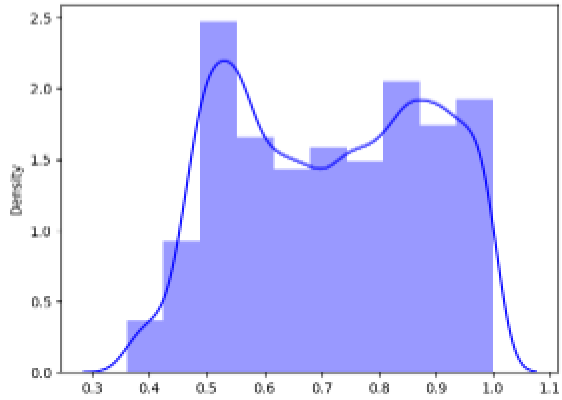
```
[21]: sns.distplot(x=df['satisfaction_level'], hist=True,bins=10 ,color='green')
```

```
[21]: <Axes: ylabel='Density'>
```



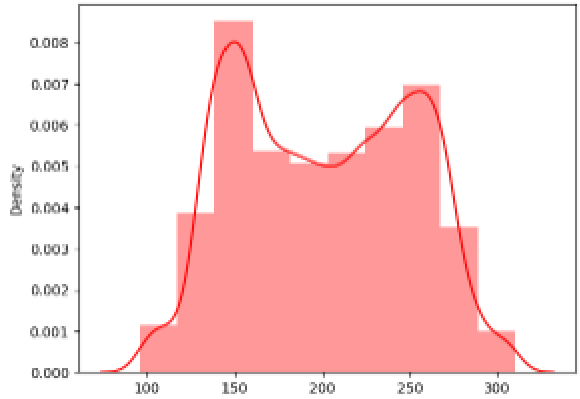

```
[22]: sns.distplot(x=df['last_evaluation'], hist=True,bins=10 ,color='blue')
```

```
[22]: <Axes: ylabel='Density'>
```



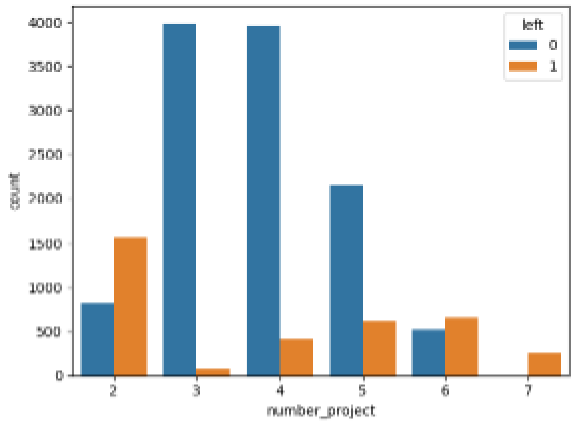
[23]:
```
sns.distplot(x=df['average_montly_hours'], hist=True,bins=10 ,color='red')
```

[23]:
```
<Axes: ylabel='Density'>
```



[42]:
```
sns.countplot(data=df, x='number_project', hue='left')
```

[42]:
```
<Axes: xlabel='number_project', ylabel='count'>
```



CONCLUSION

1. Employees with experience between 3 to 5 yrs are likely to leave.

2. employees who worked less than 160 Hrs were most likely to quit.

3. employees who worked more than 220 Hrs on an average were at medium risk of quiting the company.

4. Employees who worked 200 Hrs mothly on an average were pretty happy and did not quit from the company.

5. Employees with satisfaction level less than 0.4 were most likely to leave the company.

6. Employees with satisfaction level of 0.6 were happy at the company and will not quit.

7. Employees with satisfaction level of 0.8 and higher had some chance of quiting.

8. Employees with low and medium salary had more chance of quitting the others.

9. Employees who belongs to sales, technical and support departments are more prone to leaving the company.


Thank
You