Dashboard Creation Using Cognos/Looker

Project Description / Business Task

Dashboards are powerful tools that enable stakeholders to visualize and interpret data, uncovering trends and actionable insights. In this lab, you will use IBM Cognos Analytics or Google Looker to design an interactive dashboard that highlights key performance metrics for HealthFirst Care. This dashboard will serve as a decision-support tool, addressing areas such as patient wait times, resource utilization, and patient satisfaction levels.

Import Libraries

In [1]:

```
import numpy as np
#from numpy import count_nonzero, median, mean
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import random

# Plotly
import plotly.express as px
import plotly.offline as py
import plotly.graph_objs as go

import datetime
from datetime import datetime, timedelta, date

%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)
#sns.set(rc={'figure.figsize':(14,10)})

plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns',None)
```

```
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```
Autosaving every 60 seconds

Import Data

In [2]:

```
df1 = pd.read_csv("Cleaned_Appointment.csv", parse_dates=["Date"], dayfirst=True)
```

In [3]:

```
df1.head()
```

Out[3]:

	index	AppointmentID	PatientID	DoctorID	Department	Date	Time	Status
0	1	A7515	P16262	D1	Orthopedics	2024-09-10	13:00	No Show
1	2	A7537	P99601	D30	Cardiology	2024-12-17	15:00	Completed
2	3	A8866	P55235	D30	Neurology	2024-07-09	15:30	Rescheduled
3	4	A3039	P93372	D23	Pediatric	2024-08-12	18:00	Cancelled
4	5	A3237	P84444	D1	Pediatric	2024-03-06	17:30	Rescheduled

In [4]:

```
df1.dtypes
```

Out[4]:

```
index                  int64
AppointmentID         object
PatientID             object
DoctorID              object
Department            object
Date          datetime64[ns]
Time                  object
Status                object
dtype: object
```

In [5]:

```
df2 = pd.read_csv("Cleaned_Feedback.csv")
```

In [6]:

```
df2.head()
```

Out[6]:

	index	FeedbackID	PatientID	Department	Feedback Score	Comments	WaitTime
0	0	F1000	P99601	Cardiology	4.00	Delayed response	45.00
1	1	F1001	P36565	Orthopedics	3.00	Needs improvement	45.00
2	3	F1003	P67499	Neurology	9.00	Amazing team	30.00
3	4	F1004	P49994	Pediatric	8.00	Great doctors	45.00

	index	FeedbackID	PatientID	Department	Feedback Score	Comments	WaitTime
4	5	F1005	P31843	Pediatric	5.00	Satisfactory	45.00

In [7]:

```
df3 = pd.read_csv("Cleaned_Pivot1.csv", parse_dates=["Date"], dayfirst=True)
```

In [8]:

```
df3.head()
```

Out[8]:

	PatientID	Department_x	Date	Time	Status	Feedback Score	WaitTime	Satisfaction Level
0	P16262	Orthopedics	2024-09-10	13:00	No Show	9.00	30.00	High
1	P99601	Cardiology	2024-12-17	15:00	Completed	4.00	45.00	High
2	P55235	Neurology	2024-07-09	15:30	Rescheduled	6.00	45.00	High
3	P93372	Pediatric	2024-08-12	18:00	Cancelled	4.00	45.00	High
4	P84444	Pediatric	2024-03-06	17:30	Rescheduled	8.00	45.00	High

In [9]:

```
df4 = pd.read_csv("Cleaned_Resource.csv", parse_dates=["Date"], dayfirst=True)
```

In [10]:

```
df4.head()
```

Out[10]:

	index	ResourceID	Department	ResourceType	UsageHours	Date	Availability
0	0	R5487	Cardiology	Nurse	2	2024-07-28	Available
1	1	R8097	Orthopedics	Doctor	0	2024-09-10	Unavailable
2	2	R2557	General Medicine	Room	0	2024-12-17	Under Maintenance
3	3	R9024	Neurology	Doctor	9	2024-07-09	Available
4	5	R2629	Oncology	Doctor	4	2024-03-06	Available

Histogram

In [11]:

```
fig = px.histogram(data_frame=df2, x="Feedback Score", nbins=20, title="Histogram of Feedback Score")
fig.show()
```

In [12]:

```
fig = px.histogram(data_frame=df2, x="WaitTime", nbins=5, title="Histogram of Wait Time")
fig.update_traces(marker_color='orange')
fig.update_layout(
   xaxis_title="Wait Times",
   yaxis_title="Count"
)
fig.show()
```

Bar Plots

In [13]:

```
fig = px.bar(data_frame=df4, x="Department", y="UsageHours", title="Usage Hours by Dept")
fig.update_traces(marker_color='red')
fig.update_layout(
   xaxis_title="Department",
   yaxis_title="Usage Hours"
)
# Show the figure
fig.show()
```

In [14]:

```
fig = px.bar(data_frame=df4, x="ResourceType", y="UsageHours", title="Usage Hours by Resource Type")
fig.update_traces(marker_color='lime')
fig.update_layout(
   xaxis_title="Resource Type",
   yaxis_title="Usage Hours"
)
# Show the figure
fig.show()
```

Scatter Plots

In [15]:

```
fig = px.scatter(
   data_frame=df2,
   x="Feedback Score",
   y="WaitTime",
   title="Scatterplot")
fig.update_layout(
   xaxis_title="Feedback Score",
   yaxis_title="Wait Time"
)
fig.show()
```

In [16]:

```
# fig = px.scatter(data_frame=df, x="", y="", color="continent", title="",
#                  size="pop", size_max=10, hover_name="country")
# fig.show()
```

Line Plots

What is a line plot and why use it?

A line chart or line plot is a type of plot which displays information as a series of data points called 'markers' connected by straight line segments. It is a basic type of chart common in many fields. Use line

plot when you have a continuous data set. These are best suited for trend-based visualizations of data over a period of time.

Line plot is a handy tool to display several dependent variables against one independent variable. However, it is recommended that no more than 5-10 lines on a single graph; any more than that and it becomes difficult to interpret.

In [17]:
```
fig = px.line(data_frame=df3, x="Date", y="WaitTime", title="Wait Time Trends 1", color="Department_x")
fig.update_layout(
   xaxis_title="Date",
   yaxis_title="Wait Time"
)
fig.show()
```

In [18]:
```
fig = px.line(data_frame=df3, x="Time", y="WaitTime", title="Wait Time Trends 2", color="Department_x")
fig.update_layout(
   xaxis_title="Time",
   yaxis_title="Wait Time"
)
fig.show()
```

In [19]:
```
fig = px.scatter(data_frame=df2, x="Feedback Score", y="WaitTime", trendline="ols", title="Trendline")
fig.update_layout(
   xaxis_title="Feedback Score",
   yaxis_title="Wait Time"
)
fig.show()
```

In [20]:
```
# fig = px.scatter_matrix(data_frame=df, title="Heatmap", width=2000, height=2000,
#                         labels={col:col.replace('_', ' ') for col in df.columns})
# fig.show()
```

Box Plots

A box plot is a way of statistically representing the *distribution* of the data through five main dimensions:

- **Minimum:** The smallest number in the dataset excluding the outliers.
- **First quartile:** Middle number between the minimum and the median.
- **Second quartile (Median):** Middle number of the (sorted) dataset.
- **Third quartile:** Middle number between median and maximum.
- **Maximum:** The largest number in the dataset excluding the outliers.

In [21]:
```
fig = px.box(data_frame=df3, y="Feedback Score", title="Feedback Score Boxplot")


fig.show()
```

In [22]:
```
fig = px.box(data_frame=df3, y="WaitTime", title="Wait Time Boxplot")
```

```
fig.show()
```

Violin Plots

In [23]:

```
fig = px.violin(data_frame=df4, y="UsageHours", title="Violinplot for Usage Hours")
fig.show()
```

Heatmap

In [24]:

```
# Pivot the DataFrame
heatmap_data = df4.pivot_table(values='UsageHours',
                               index='Department',
                               columns='ResourceType',
                               aggfunc='sum')  # Summing usage hours for each resource type per department

# Create the heatmap
fig = px.imshow(heatmap_data,
                color_continuous_scale='YlOrRd',  # Color scale from yellow to red (low to high usage)
                text_auto=True,  # Show values inside the cells
                labels={'x': 'Resource Type', 'y': 'Department', 'color': 'Usage Hours'},
                title="Resource Usage Heatmap by Department")

# Resize the heatmap
fig.update_layout(
    width=1000,  # Set the width of the heatmap
    height=700,  # Set the height of the heatmap
)


# Show the figure
fig.show()
```

Area Plot

In [25]:

```
# fig = px.area(data_frame=df, x="", y="", color="continent", line_group="country", title="",
#               labels={'actual_productivity': 'Actual Productivity'})
# fig.show()
```

Pie Charts

A pie chart is a circular graphic that displays numeric proportions by dividing a circle (or pie) into proportional slices. You are most likely already familiar with pie charts as it is widely used in business and media. We can create pie charts in Matplotlib by passing in the kind=pie keyword.

- autopct - is a string or function used to label the wedges with their numeric value. The label will be placed inside the wedge. If it is a format string, the label will be fmt%pct.
- startangle - rotates the start of the pie chart by angle degrees counterclockwise from the x-axis.
- shadow - Draws a shadow beneath the pie (to give a 3D feel).

In [26]:

```
df3.columns
```

Out[26]:

```
Index(['PatientID', 'Department_x', 'Date', 'Time', 'Status', 'Feedback Score', 'WaitTime', 'Satisfaction Level'],
dtype='object')
```

In [27]:

```
# Group by 'Satisfaction Level' and count the number of patients in each category
satisfaction_counts = df3.groupby('Satisfaction Level')['PatientID'].count().reset_index()
satisfaction_counts
```

Out[27]:

	Satisfaction Level	PatientID
0	High	164
1	Low	27

In [28]:

```
# Calculate the percentage of patients in each category
satisfaction_counts['Percentage'] = (satisfaction_counts['PatientID'] / satisfaction_counts['PatientID'].sum()) *
100
satisfaction_counts
```

Out[28]:

	Satisfaction Level	PatientID	Percentage
0	High	164	85.86
1	Low	27	14.14

In [29]:

```
# Create a pie chart
fig = px.pie(
    satisfaction_counts,
    names='Satisfaction Level',
    values='Percentage',
    title='Patient Satisfaction Levels (High vs Low)',
    color='Satisfaction Level',  # Color by Satisfaction Level
    color_discrete_map={'High': 'green', 'Low': 'red'},  # Custom colors
    labels={'Satisfaction Level': 'Satisfaction Level', 'Percentage': 'Percentage of Patients'}
)

fig.update_layout(
    width=1000,  # Set the width
    height=700,  # Set the height
)
```

```
# Show the figure
fig.show()
```

Geospatial Analysis

In [30]:

```
# fig = px.choropleth(data_frame=df, locations="iso_alpha", color="lifeExp", hover_name="country",
#                     animation_frame="year", title="",
#                     color_continuous_scale=px.colors.sequential.Plasma, projection="natural earth")
# fig.show()
```

Python code done by Dennis Lam