Product Sales task for bupa

Business Goals

Bupa is an international healthcare group that provides a wide range of healthcare services and insurance products. The company operates globally and is focused on delivering high-quality healthcare solutions to individuals, families, and businesses.

Bupa's primary areas of operation include: Health Insurance, Hospitals and Clinics, Care Homes and Retirement Villages, Dental Care, Corporate Health and Wellbeing and International Health Services.

The data collected is from the pipeline. The main goal of this report is to outline the following:

- Where the revenue is coming from
- Understand what the data means
- Validate the data
- Anything that can help in the world of sales
- Suggestions on how the data system can be improved

Here I assume that our primary business concern (from what we can derive from the data) would be Bupa's sales and revenue, and to outline any problems or suggestions to improve profitability.

Data Cleansing and Data Validation

Products.xlsx

"Products.xlsx" provides a list of Products and the Group of which these Products are associated with. It contains 43 rows and 2 columns.

```
import pandas as pd

products = pd.read_excel("Products.xlsx")
print("- Amount of Unique products = " ,products['Product'].nunique())
print("- List of all product groups : ",products['Product
Group'].unique())
```

```
- Amount of Unique products =  43
- List of all product groups :  ['Occupational Health' 'Primary Care'
'Be.Me' 'Health Stations'
 'Health Assessment' 'SmartDNA' 'Flu' 'Mental Health' 'Health Talks'
'MSK'
 'Onsite Health Checks' 'UNKNOWN' 'Wellbeing pot']
```

The amount of unique products are the same as the amount of products, showing that they are no duplicate products listed in "Products.xlsx". We have 13 unique product groups, where unfortuntantely, some are unknown. Lets have a look at how many unknown product groups they are.

```
print("- Number of Unknown product groups = ", products['Product 
Group'].isin(['UNKNOWN']).sum())
```

```
- Number of Unknown product groups =  1
```

Which product as an unknown product group?

```
print(products[products["Product Group"] == "UNKNOWN"])
```

```
         Product Product Group
36  PMI – Select       UNKNOWN
```

I'm not familiar with this product, so I will leave the Product Group as "Unknown" for now.

Sales Team.xlsx

"Sales Team.xlsx" contains 20 rows and 2 columns. One column is called "Sales Person" and the other is "Sales Manager".

```
sales = pd.read_excel("Sales Team.xlsx")
print("- Amount of Unique sales people = " ,sales['Sales 
Person'].nunique())
print("- List of all sales managers : ",sales['Sales 
Manager'].unique())
print("- How many sales managers are in the Sales people column? - ", 
sales['Sales Person'].isin(sales['Sales Manager'].unique()).sum())
```

```
- Amount of Unique sales people =  20
- List of all sales managers :  ['Kaine Thomas' 'Leona Pugh' 'Adele 
Pearson' 'Alexandros Ware']
- How many sales managers are in the Sales people column? -  0
```

From above, it is shown that they are 20 sales people and 4 managers, totalling 24 sales staff. No null values were found so we have successfully validated the data. No cleansing was required.

pipeline_data.xlsx

"pipeline_data.xlsx" has 999 rows and 7 columns.

- Owner : Sales member selling the product
- Potential Customer : Customer name
- Contract Start Date : Start of contract, In Q3 and Q4 of 2023.
- Contract Value ; Value of Product in £
- Direct Opportunity: Should be either Direct or Broker
- Product : What product was sold

- Status : Should be either Won or Lost

```
pipeline_data = pd.read_excel("pipeline_data.xlsx")
print(pipeline_data.isna().sum())
```

```
Owner                   3
Potential Customer      0
Contract Start Date     0
Contract Value         43
Direct Opportunity      0
Product                43
Status                  0
dtype: int64
```

The above table tells us how many null values are in each column. We can see that the Owner, Contract Value and Product columns have null values, where the latter two have the same amount of null values (43). Are these linked to one another?

```
print("Number of rows where both Contract Value and Product values are null = ",pipeline_data[pipeline_data['Contract Value'].isnull() & pipeline_data['Product'].isnull()].shape[0])
```

```
Number of rows where both Contract Value and Product values are null = 43
```

We see when the Contract value is null, the product is null as well. As these rows do not provide much insight, I decided to remove them. I also remove unknown owners as they are only 3 and we can afford to drop these.

```
pipeline_data = pipeline_data[~pipeline_data['Contract Value'].isnull()]
pipeline_data = pipeline_data[~pipeline_data['Owner'].isnull()]
print(pipeline_data.isna().sum())
```

```
Owner                  0
Potential Customer     0
Contract Start Date    0
Contract Value         0
Direct Opportunity     0
Product                0
Status                 0
dtype: int64
```

Lets check each type of each category.

```
print(pipeline_data.dtypes)
```

```
Owner                  object
Potential Customer     object
Contract Start Date    object
Contract Value         object
Direct Opportunity     object
Product                object
```

```
Status                    object
dtype: object
```

```
pipeline_data[['Owner','Potential Customer','Product']] =
pipeline_data[['Owner','Potential
Customer','Product']].astype("string")
pipeline_data[['Direct Opportunity','Status']] =
pipeline_data[['Direct Opportunity','Status']].astype("category")
#pipeline_data["Contract Value"] = pipeline_data["Contract
Value"].astype("int64")
#pipeline_data['Contract Start Date'] =
pd.to_datetime(pipeline_data['Contract Start
Date'],dayfirst=True).dt.strftime('%d-%m-%Y')
```

Here, The column "Contract Value" is not all numeric and the "Contract Start Date" values are not all dates. I use "#" so not to print out the errors for cleanliness purposes. Lets find what values are non numeric in "Contract Value" column.

Cleansing and Validating 'Contract Value'

```
import numpy as np

non_numeric_values = pipeline_data.loc[~pipeline_data['Contract
Value'].apply(lambda x: pd.to_numeric(str(x),
errors='coerce')).notnull()
]['Contract Value']
print(non_numeric_values)
```

```
65          aa
533     £70.00
717        tbc
920        tbc
Name: Contract Value, dtype: object
```

```
pipeline_data =
pipeline_data.drop(pipeline_data[pipeline_data['Contract
Value'].isin(['aa', 'tbc'])].index)
pipeline_data['Contract Value'] = pipeline_data['Contract
Value'].replace(['£70.00'],[70])
```

Lets check the limits of the contract values.

```
pd_sorted = pipeline_data.sort_values(by='Contract Value',
ascending=False)
print(pd_sorted.head(7))
```

```
                 Owner           Potential Customer  Contract Start
Date  \
376       Nikodem Moss                  Simonis Inc  2023-08-20
00:00:00
413      Alyssia Walsh  Kuhic, Steuber and MacGyver  2023-08-27
00:00:00
```

```
200        Ehsan Knapp    Kilback, Bauch and Russel  2023-07-12
00:00:00
772  Brendan Patterson                  Wolf - Conn  2023-11-14
00:00:00
543  Brendan Patterson          Greenholt - Kerluke  2023-09-23
00:00:00
201        Ehsan Knapp  Luettgen, Langosh and Sauer  2023-07-12
00:00:00
726        Tess Church            Heidenreich Group  2023-11-03
00:00:00

     Contract Value Direct Opportunity  \
376        350000.0             Direct
413        300000.0             Broker
200        291900.0             Broker
772        225000.0             Broker
543        225000.0             Broker
201        168000.0             Broker
726        119880.0             Direct

                                      Product Status
376  Absence management --Management referral   Lost
413                Clinics-Health Assessment     Won
200                Clinics-Health Assessment    Lost
772                Clinics-Health Assessment     Won
543                Clinics-Health Assessment    Lost
201                Clinics-Health Assessment    Lost
726                Clinics-Health Assessment    Lost
```

Since the outliers are all "Clinics-Health Assessment" Products, these prices are validated. The highest contract value of 350000 is a different product but it is not too higher than the second highest price, so I do not remove any of these rows.

Cleansing and Validating 'Contract Start Date'

```
non_date_values = pipeline_data['Contract Start
Date'].loc[pipeline_data['Contract Start
Date'].str.contains('/').fillna(False)]

print(non_date_values)
```

```
995      17/082023
996    21//12/2023
997    26//10/2023
Name: Contract Start Date, dtype: object
```

And we also have dates that are not valid, for example:

```
start_dates_valid = pipeline_data[~pipeline_data['Contract Start
Date'].isin(non_date_values)]['Contract Start Date']
print(start_dates_valid.tail(5))
```

```
990    2029-06-25 00:00:00
991    2029-11-01 00:00:00
992    2029-11-06 00:00:00
993    3023-10-04 00:00:00
994    3023-11-08 00:00:00
Name: Contract Start Date, dtype: object
```

In total, we have 3 dates that have been incorrect formatting, 4 dates that are in 2029, and 2 dates that are in 3023. These are most likely errors and are meant to be 2023. We can fix the 3 incorrectly formatted dates as well.

```
#dates correctly formatted
pipeline_data['Contract Start Date'] = pipeline_data['Contract Start Date'].replace(['17/082023','21//12/2023','26//10/2023'],
['17/08/2023','21/12/2023','26/10/2023'])

#2029 & 3023 -> 2023
fixed_times = ['2023-08-17 00:00:00','2023-11-21 00:00:00','2023-10-26 00:00:00','2023-10-04 00:00:00','2023-11-08 00:00:00']
for i in range(990,995):
    pipeline_data.loc[i, 'Contract Start Date'] = pd.to_datetime(fixed_times[i-990])

#convert these dates to day-month-year format for easy reading
pipeline_data['Contract Start Date'] = pd.to_datetime(pipeline_data['Contract Start Date'],dayfirst=True).dt.strftime('%d-%m-%Y')
```

Validating 'Direct Opportunity', 'Potential Customer' and 'Status'

```
pipeline_data.nunique()
```

```
Owner                  12
Potential Customer    912
Contract Start Date   211
Contract Value        390
Direct Opportunity      2
Product                19
Status                  2
dtype: int64
```

Status and Direct opportunity are both valid, with 2 options each (Won or Lost and Broker or Direct). The Potential Customer column have mostly unique customers in with not many repeat customers. It would be difficult and not very useful to check each one of these for spelling or grammar mistakes etc, so we can leave it how it is.

Validating 'Product' & 'Owner' column

They are 12 owners and 19 products. Let us check if these Owners are in the sales people column for the other Sales team.xlsx file, and the products are the same in the products.xlsx file.

```
print("Are all Products in Products.xlsx?
",all(pipeline_data['Product'].unique().isin(products['Product'])))
print("Are all Owners in Sales Team.xlsx?
",all(pipeline_data['Owner'].unique().isin(sales['Sales Person'])))
```

```
Are all Products in Products.xlsx?  True
Are all Owners in Sales Team.xlsx?  False
```

Let us have a look on which owner in not in the sales team.

```
unique_owners = pd.Series(pipeline_data['Owner'].unique())
for owner in unique_owners:
    if owner in sales['Sales Person'].values:
        print(owner, ", in sales team")
    else:
        print(owner, ", not in sales team")
```

```
Tess Church , in sales team
Syed Nicholson , in sales team
Rehan Holloway , in sales team
Eugene Shelton , in sales team
Jensen Duke , not in sales team
Ehsan Knapp , in sales team
Elodie Carlson , in sales team
Brendan Patterson , in sales team
Albie Buck , in sales team
Nikodem Moss , in sales team
Alyssia Walsh , in sales team
Chaim Pope , in sales team
```

Jensen Duke is not in the sales team, neither as a sales person or manager. How many sales has Jensen Duke done?

```
print("How many times does Jensen Duke appear?
",pipeline_data['Owner'].value_counts()['Jensen Duke'])
```

```
How many times does Jensen Duke appear?  6
```

Jensen Duke may be a former sales member, but as he has only done 6 sales, so he can be safely dropped from the dataset.

```
pipeline_data =
pipeline_data.drop(pipeline_data[pipeline_data['Owner'] == 'Jensen
Duke'].index)
```

Final results of cleansing and validation of pipeline_data.xlsx

- The total rows have decreased from 999 to 944, as we have dropped 43 rows due to empty products/value, 3 rows due to unknown owners, 3 rows due to invalid contract values and 6 rows due to invalid sales member.
- Dates from 2029 and 3023 where changed to 2023, and wrongly formatted dates were fixed.

- Products and Owners are all valid products and sales team members.
- Wrongly inputted numbers in contract value were fixed.
- 'Direct Opportunity' and 'Status' have valid categories and 'Potential Customer' column are mostly unique values.

Data exploration and visualisation

Merging our datasets together

Let us merge the sales manager from sales and let us merge the product group from products to our pipeline_data.

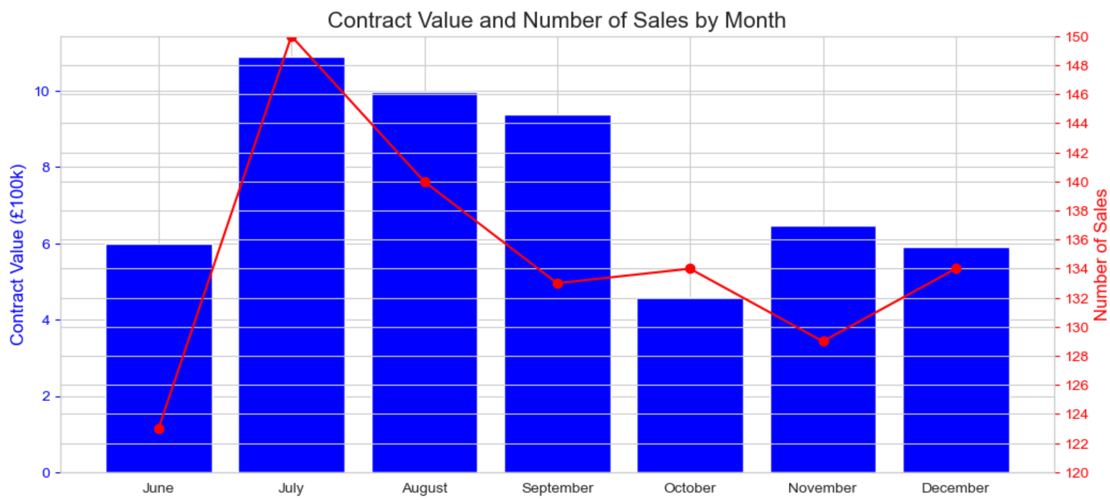
```
df = pd.merge(pipeline_data, products, on='Product')
df = pd.merge(df, sales, left_on='Owner', right_on = 'Sales Person')
```

Seasonality vs sales

We look at sales amount and sales value in this section. Please note that June is included, even though it is not in Q3.

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import MultipleLocator

df['Contract Start Date'] = pd.to_datetime(df['Contract Start Date'],
dayfirst=True)
df['Month'] = df['Contract Start Date'].dt.strftime('%B')
month_order = [ 'June', 'July', 'August', 'September', 'October',
'November', 'December']
df['Month'] = pd.Categorical(df['Month'], categories=month_order,
ordered=True)

monthly_data = df.groupby('Month').agg({'Contract Value': 'sum',
'Sales Person': 'count'}).reset_index()
monthly_data['Contract Value'] = monthly_data['Contract Value']/100000
monthly_data = monthly_data.sort_values('Month')

fig, ax1 = plt.subplots(figsize=(12, 6))

ax1.bar(monthly_data['Month'], monthly_data['Contract Value'],
color='blue')
ax1.set_ylabel('Contract Value (£100k)', color='blue', fontsize=12)
ax1.tick_params('y', colors='blue')

ax2 = ax1.twinx()
ax2.plot(monthly_data['Month'], monthly_data['Sales Person'],
color='red', marker='o')
ax2.set_ylabel('Number of Sales', color='red', fontsize=12)
```

```
ax2.tick_params('y', colors='red')
ax2.yaxis.set_major_locator(MultipleLocator(2))
ax2.set_ylim(120, 150)

plt.subplots_adjust(bottom=0.2)
plt.title('Contract Value and Number of Sales by Month' , fontsize=15)
plt.xlabel('Month')
plt.show()
```



This graph shows and increase in sales and revenue in July & August. The winter months perform relatively poorly but June's number of sales to contract value, shows increases sales of more expensive products.

How are different Sales Managers teams performing?

Let us first look at each Sales Manager and how many Sales people are managed by them.

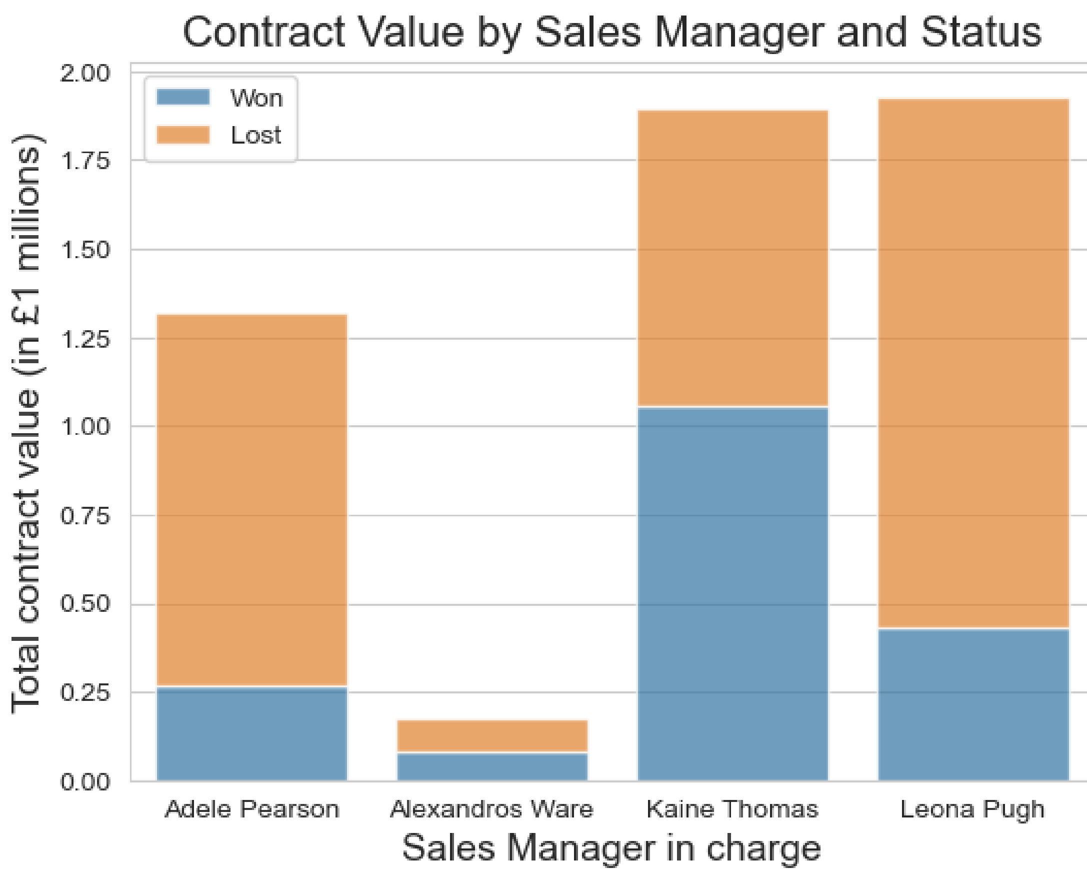
```
print(sales['Sales Manager'].value_counts())
```

```
Kaine Thomas       5
Leona Pugh         5
Adele Pearson      5
Alexandros Ware    5
Name: Sales Manager, dtype: int64
```

As each manager has an equal number of staff in their team, let us look at their total sales for both quarters.

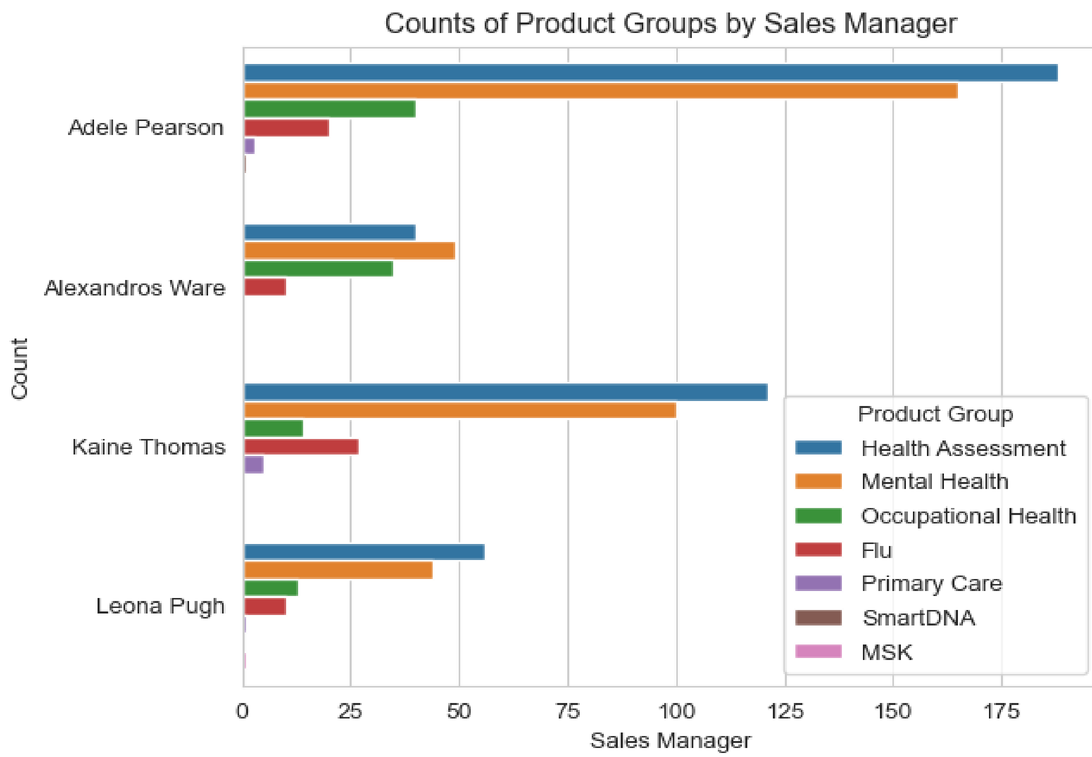
```
grouped = df.groupby(['Sales Manager', 'Status'])['Contract
Value'].sum().unstack().fillna(0)
grouped = grouped/1000000

sns.set_style('whitegrid')
sns.barplot(x=grouped.index, y='Won', data=grouped, color='#1f77b4',
alpha=0.7,label='Won')
```

```
sns.barplot(x=grouped.index, y='Lost', data=grouped, color='#ff7f0e',
alpha=0.7, label='Lost', bottom=grouped['Won'])

plt.xlabel('Sales Manager in charge',fontsize=14)
plt.ylabel('Total contract value (in £1 millions)',fontsize=14)
plt.title('Contract Value by Sales Manager and Status',fontsize=16)
plt.legend()
plt.show()
```



Why does Mr Ware's team have such low sales? Do all teams sell the same products?

```
product_counts = df.groupby('Sales Manager')['Product
Group'].value_counts().reset_index(name='Count')


sns.set_style('whitegrid')
sns.barplot(y='Sales Manager', x='Count', hue='Product Group',
data=product_counts)

plt.title('Counts of Product Groups by Sales Manager')
plt.xlabel('Sales Manager')
plt.ylabel('Count')
plt.show()
```



We can see that Mr Ware's team and Mrs Pughs team both sell much lower amounts than their fellow managers. However, Mrs Pugh's team returns a much higher total value over Mr Ware. We can also see that Mrs Pearson's team had the most contracts, but failed to convert these to good returns relatively.

- Moving on forward, the Product Groups 'MSK' and 'SmartDNA' have only one sale each, which were both lost. Let us remove these as they serve little use and disrupt the data.

```
df = df[~df['Product Group'].isin(['MSK', 'SmartDNA'])]
```

How does each sales person contribution weigh?

Let us first look at how many sales did each sales person make.

```
print(df['Owner'].value_counts())
```

```
Tess Church          341
Syed Nicholson       217
Rehan Holloway       131
Ehsan Knapp          105
Eugene Shelton        74
Brendan Patterson     50
Elodie Carlson        18
Albie Buck             2
Nikodem Moss           1
Alyssia Walsh          1
```

```
Chaim Pope              1
Name: Owner, dtype: int64
```

This shows that our data is heavily skewed to sales people like Tess Church and Syed Nicholson. As stated in the Assignment details, this is all the data from Q3 and Q4. But how can the majority of sales be made from 4 sales people, and 9 sales persons apparently made no sales within Q3 and Q4. Further research internally must be made to see what the case is.

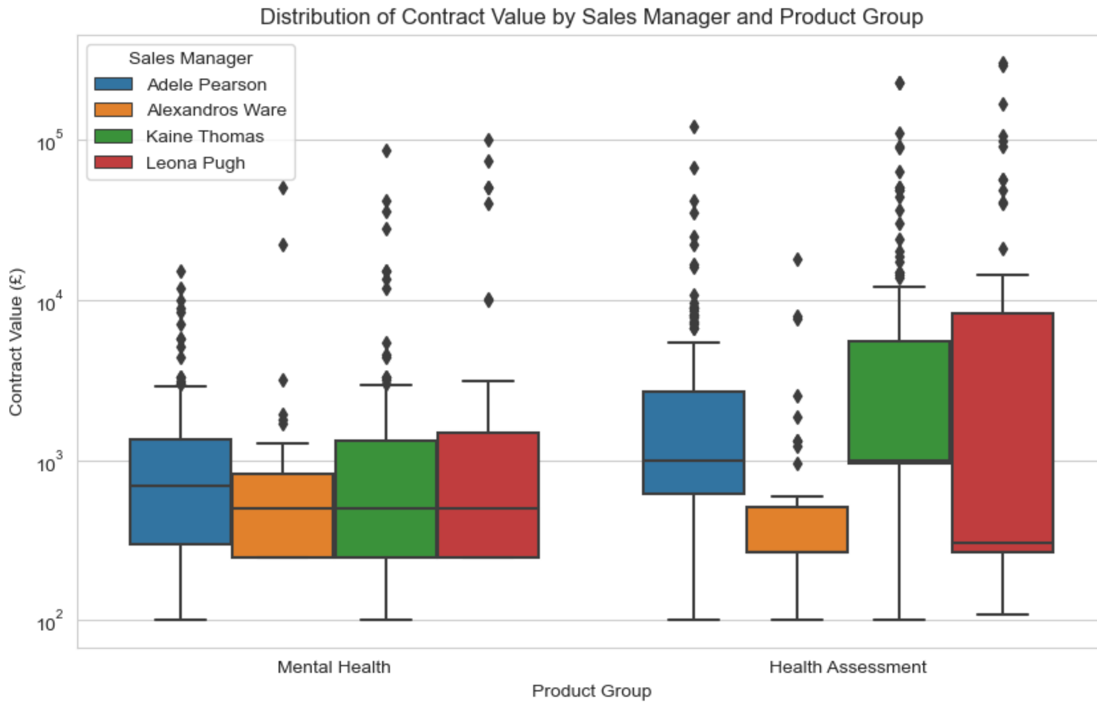
Tess Church is in Adele Pearson's team and Syed Nicholson is in Kaine Thomas' team which explains the skew. Could it be that for sales people with less sales, may have recorded more than one quantity of the product sale as 1 sale?

```
print(df['Product Group'].value_counts())
```

```
Health Assessment      405
Mental Health          358
Occupational Health    102
Flu                     67
Primary Care             9
Name: Product Group, dtype: int64
```

I will only use the Product groups Health Assessment and Mental Health, as all managers have sufficient sales in each of these categories.

```
plt.figure(figsize=(10, 6))
df2 = df[df['Product Group'].isin(['Health Assessment','Mental
Health'])]
ax = sns.boxplot(data=df2, hue='Sales Manager', y='Contract Value',
x='Product Group', hue_order = ['Adele Pearson', 'Alexandros Ware',
'Kaine Thomas', 'Leona Pugh'])

plt.xlabel('Product Group')
plt.ylabel('Contract Value (£)')
plt.title('Distribution of Contract Value by Sales Manager and Product
Group')
ax.set_yscale('log')
```



The graph above looks at the contract value distribution for the top two product categories. The mental Health product category is consistent with each manager but the Health Assessment product category varies wildly. Perhaps this is due to Products in the product category not being similar prices?

Looking at most commonly sold products

Let us first look at the number of products in each product group.

```
print(products.groupby('Product Group')
['Product'].count().sort_values(ascending=False))
```

```
Product Group
Occupational Health     9
Mental Health           8
Health Assessment       6
Primary Care            5
MSK                     4
Flu                     2
Health Stations         2
SmartDNA                2
Be.Me                   1
Health Talks            1
Onsite Health Checks    1
UNKNOWN                 1
Wellbeing pot           1
Name: Product, dtype: int64
```

Occupational Health, Mental Health and Health Assessment have the most products, which is why these product groups generate the most revenue. What about the average contract price for the Mental Health and Health Assessment groups?

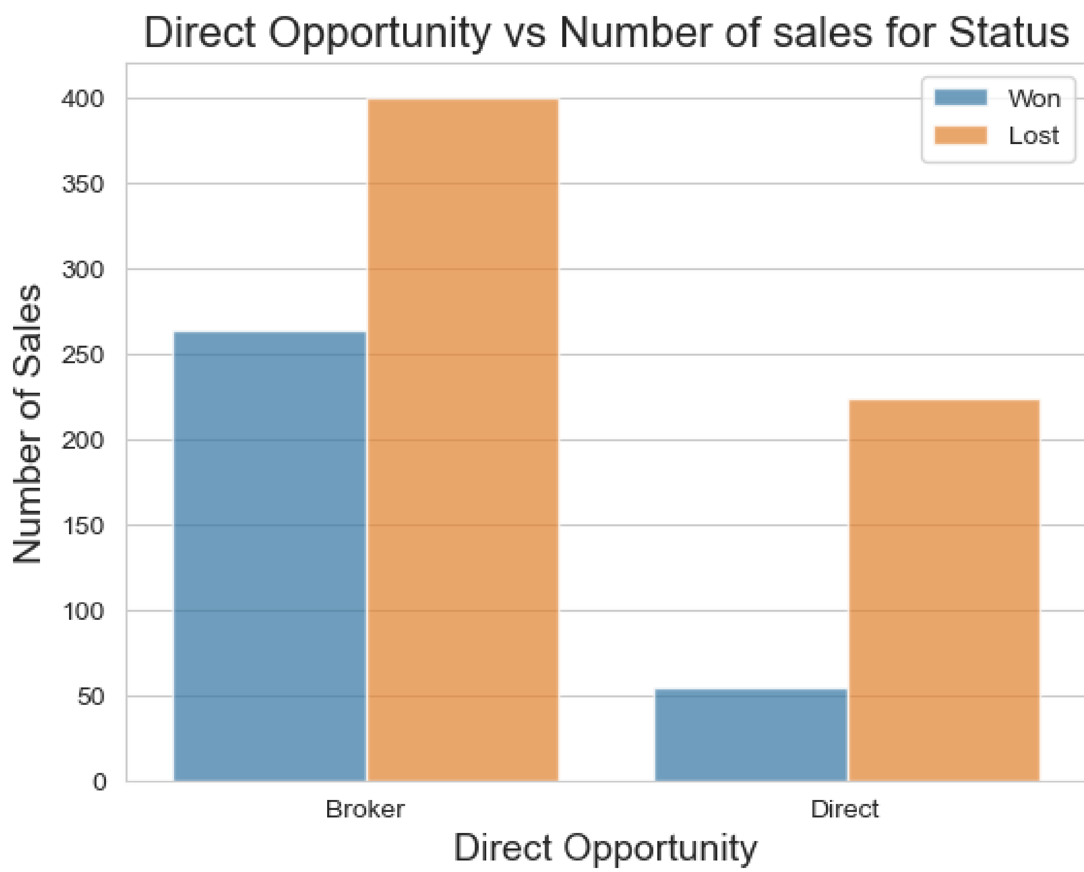
```
import numpy as np

product_stats = df2.groupby(['Product', 'Product Group'])['Contract
Value'].agg(['median', lambda x: np.quantile(x, 0.25), lambda x:
np.quantile(x, 0.75), 'mean', 'count']).reset_index()

product_stats['Median'] = product_stats['median'].round(2)
product_stats['Lower Quartile'] = product_stats['<lambda_0>'].round(2)
product_stats['Upper Quartile'] = product_stats['<lambda_1>'].round(2)
product_stats['Average Price'] = product_stats['mean'].round(2)
product_stats = product_stats.drop(['median', '<lambda_0>',
'<lambda_1>', 'mean'], axis=1)

product_stats.columns = ['Product', 'Product Group', 'Count', 'Upper
Quartile', 'Lower Quartile', 'Median', 'Average Price']
product_stats['IQR'] = product_stats['Upper Quartile'] -
product_stats['Lower Quartile']
product_stats.drop(['Lower Quartile', 'Upper Quartile'], axis=1,
inplace=True)
product_stats.sort_values('Count', ascending = False, inplace=True)
order = ['Product', 'Product Group', 'Count',  'Average Price',
'Median','IQR']

print(product_stats[order])
```

```
                              Product      Product Group  Count  \
1          Clinics-Health Assessment  Health Assessment    404
3         Mental Health Services –EAP      Mental Health    327
0          Clinics - Onsite Services      Mental Health     26
4  Mental Health Services –Resilience      Mental Health      3
5        Mental Health – Healthy Minds      Mental Health      2
2                   Health Assessment  Health Assessment      1

   Average Price    Median      IQR
1        8896.50   3004.75   384.00
3        1641.14   1214.10   247.50
0       15078.69  15000.00   137.00
4       17260.67  25475.00    59.00
5       17772.50  19886.25  2113.75
2        5000.00   5000.00     0.00
```

We can see that our most commonly sold products are Clinics-Health Assessment and mental Health Services - EAP respectively. The Clinics-Health Assessment product contract value is much higher value than its counterpart which is why Health Assessment has higher sales revenue. The other products are sold less but have a much higher price. We can also see from the differences between average price and median price, that some sales most likely have different quanitities of products sold.

How does Direct & Broker opportunity affect sales?

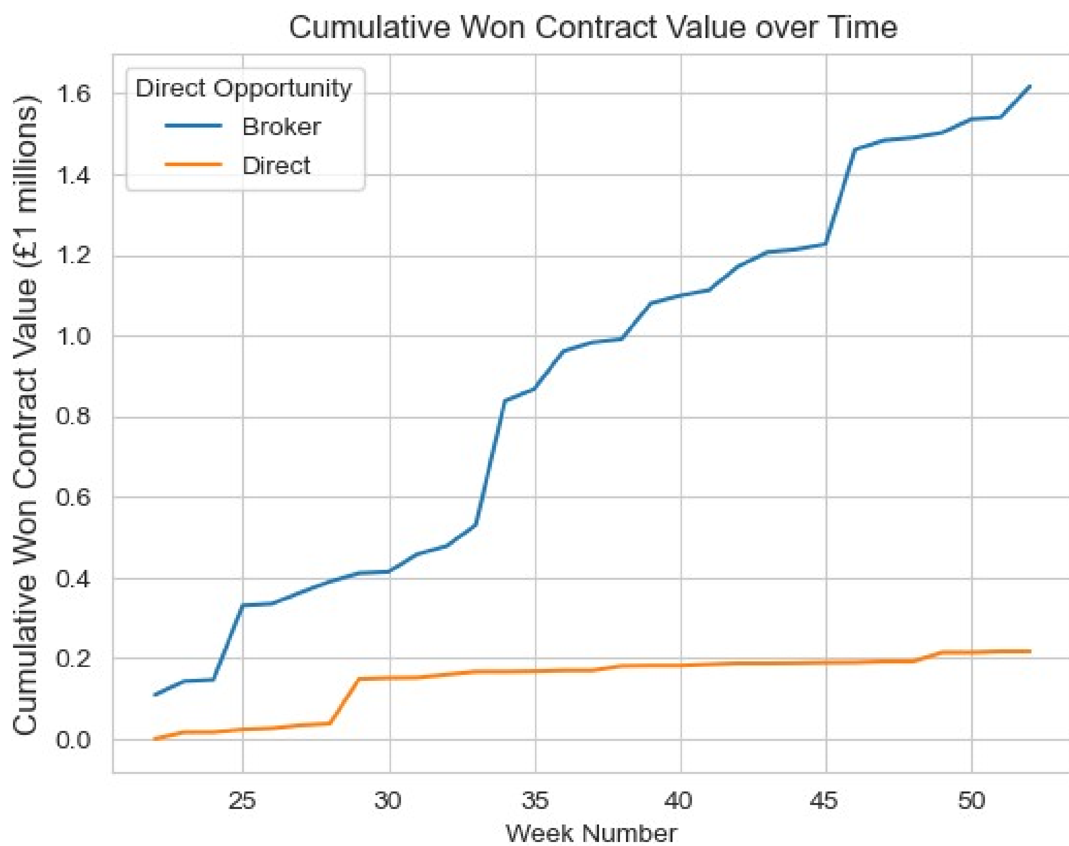
```
grouped = df.groupby('Direct Opportunity')
['Status'].value_counts().reset_index(name='Count')

sns.set_style('whitegrid')
sns.barplot(x='Direct Opportunity', y='Count', data=grouped,
hue='Status', hue_order=['Won','Lost'], alpha=0.7)

plt.xlabel('Direct Opportunity',fontsize=14)
plt.ylabel('Number of Sales',fontsize=14)
plt.title('Direct Opportunity vs Number of sales for
Status',fontsize=16)
plt.legend()
plt.show()
```



Brokers have a win ratio of 39.6% (664 sales) whilst Direct has a win ratio of 19.4% (273 sales). Does this translate to the same for sales across Q3 and Q4.

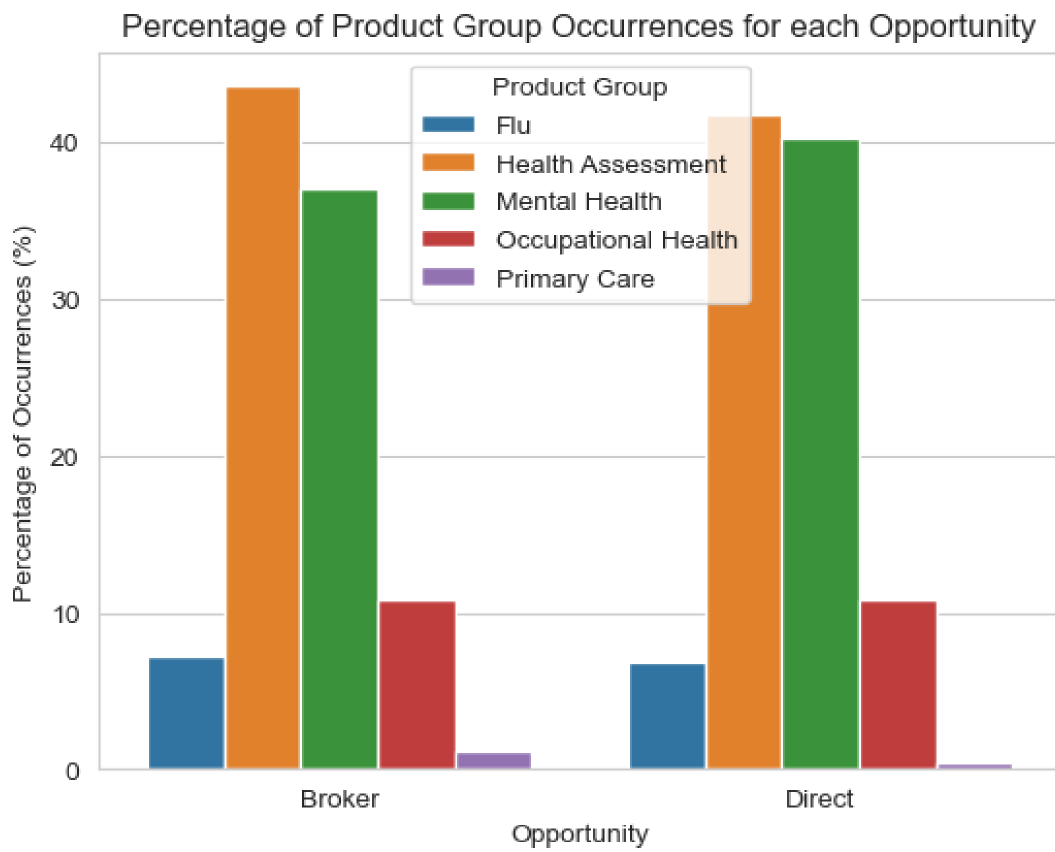
```
df['Week'] = pd.to_datetime(df['Contract Start
Date'],dayfirst=True).dt.isocalendar().week
```

```
df2 = df[df['Status'] == 'Won'].copy()
df2.sort_values('Week', inplace=True)
df2 = df2[['Direct Opportunity', 'Week', 'Contract Value']]
df2 = df2.groupby(['Direct Opportunity', 'Week'])['Contract
Value'].sum().reset_index()
df2['Cumulative Contract Value'] = df2.groupby('Direct Opportunity')
['Contract Value'].cumsum()/1000000

df2['Week'] = df2['Week'].astype('string').astype(int) #This is to
turn the Week type to integer

sns.lineplot(data=df2, x='Week', y='Cumulative Contract Value',
hue='Direct Opportunity')

plt.xlabel('Week Number')
plt.ylabel('Cumulative Won Contract Value (£1 millions)', fontsize
=12)
plt.title('Cumulative Won Contract Value over Time')
plt.show()
```



Using a Broker is much more effective than Direct. Broker has £1.6 million in sales, and Direct only slightly over £200k. It seems like Direct contribution to sales is fairly flat where they was one good week on week 28. Why is this? Is the type of product sold different?

```python
grouped_df = df.groupby(['Direct Opportunity', 'Product
Group']).size().reset_index(name='Count')
total_count = grouped_df.groupby('Direct Opportunity')
['Count'].transform('sum')
grouped_df['Percentage'] = grouped_df['Count'] / total_count * 100

sns.barplot(data=grouped_df, x='Direct Opportunity', y='Percentage',
hue='Product Group')
plt.xlabel('Opportunity')
plt.ylabel('Percentage of Occurrences (%)')
plt.title('Percentage of Product Group Occurrences for each
Opportunity')
plt.show()
```



Broker and Direct sell the same products at the same frequency. After looking at the data, it seems like using a Broker is much more efficent than Direct approach. Let us do a t-test to see if there is a significant difference between 'Contract Value' and the 'Direct' and 'Broker' opportunities. I take the null Hypothesis that they is no difference between Contract Value for the opportunities.

```python
import scipy.stats as stats

direct_values = df.loc[df['Direct Opportunity'] == 'Direct', 'Contract
Value']
```

```
broker_values = df.loc[df['Direct Opportunity'] == 'Broker', 'Contract
Value']

t_statistic, p_value = stats.ttest_ind(direct_values, broker_values)

print('T-Statistic:', t_statistic.round(3))
print('P-Value:', p_value.round(3))
```

```
T-Statistic: -0.493
P-Value: 0.622
```

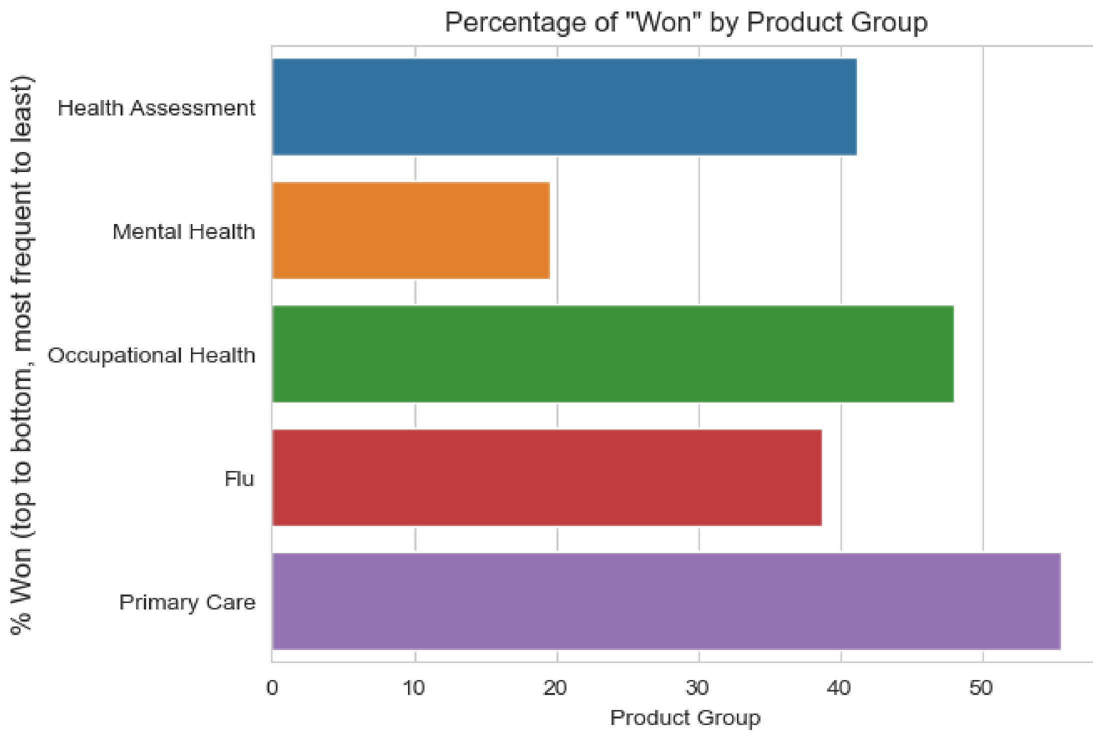
The t-statistic measures how different the average 'Contract Value' is between the two groups. A negative t-statistic means that, on average, the 'Direct' opportunities have a slightly lower 'Contract Value' compared to the 'Broker' opportunities.

The p-value tells us the probability of observing such a difference in 'Contract Value' between the groups by chance alone. In this case, the p-value of 0.6222 is above our significance level of 0.05, which tells us to reject the alternative hypothesis that they is a significant difference between contract value for each opportunity.

What affects whether a sale is won or lost?

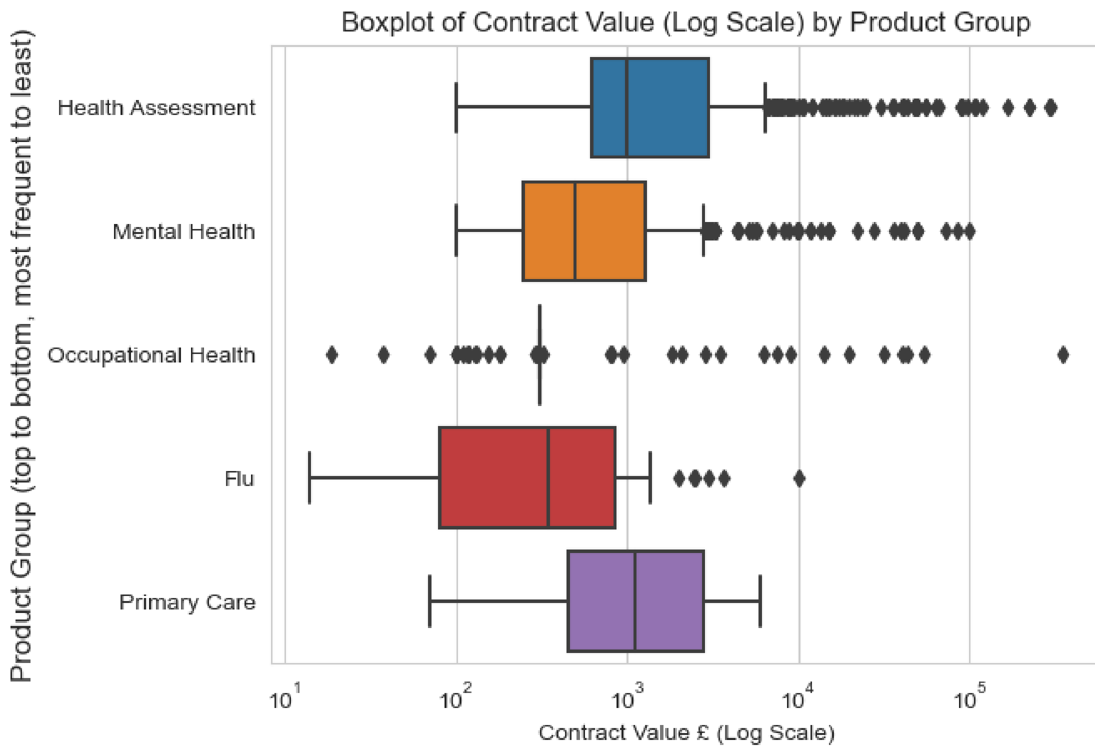
Lets see if whether product group affects if the sale is are won or lost.

```
grouped_df = df[df['Status'] == 'Won'].groupby('Product
Group').size().reset_index(name='Won Count')

group_freq = df['Product Group'].value_counts()
sorted_groups = group_freq.index.tolist()

total_count = df.groupby('Product
Group').size().reset_index(name='Total Count')
merged_df = pd.merge(grouped_df, total_count, on='Product Group',
how='outer')
merged_df['Percentage Won'] = (merged_df['Won Count'] /
merged_df['Total Count']) * 100

sns.barplot(data=merged_df, y='Product Group', x='Percentage Won',
order = sorted_groups)
plt.xlabel('Product Group')
plt.ylabel('% Won (top to bottom, most frequent to least)', fontsize =
12)
plt.title('Percentage of "Won" by Product Group')
plt.show()
```



The results read as 41.2% win ratio for Health Assessment, 19.6% for Mental Health, 48.1% for Occupational Health, 38.8% for flu and 55.6% for Primary Care. Mental Health is the weakest here, do the contract price for this product group justify the low conversion rate?

```
sns.set_style('whitegrid')
ax = sns.boxplot(y='Product Group', x='Contract Value', data=df,
order=sorted_groups)

ax.set_xscale('log')

plt.title('Boxplot of Contract Value (Log Scale) by Product Group')
plt.ylabel('Product Group (top to bottom, most frequent to least)',
fontsize = 12)
plt.xlabel('Contract Value £ (Log Scale)')

plt.show()
```



Using the above two figures, it can be seen that Primary Care products are the most superior with the highest average Contract Value and the highest win ratio. Health Assessment is also very good in both domains. Mental Health does offer the least win ratio for okay Contract values. Overall this product group may not be the best, mainly due to the win ratio.

Using Machine Learning to predict whether the status is won or lost

Let us build a simple machine learning model for our classification problem to see if a model can predict whether the status is won or lost. I will use scikit learn here, and split the data to test and train. I then use the Random Forest Classifier model to build our model and test its effectiveness.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

features = ['Owner','Contract Value', 'Direct Opportunity','Product Group', 'Week']
target = 'Status'
selected_data = df[features + [target]].copy()

categorical_cols = ['Direct Opportunity', 'Product Group','Owner']
encoded_data = pd.get_dummies(selected_data, columns=categorical_cols)
```

```
X = encoded_data.drop(target, axis=1)
y = encoded_data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

classification_rep = classification_report(y_test, y_pred)
print(classification_rep)
```

```
              precision    recall  f1-score   support

        Lost       0.76      0.74      0.75       129
         Won       0.48      0.50      0.49        60

    accuracy                           0.67       189
   macro avg       0.62      0.62      0.62       189
weighted avg       0.67      0.67      0.67       189
```

Our model predicts 76% of Won Status correct and 48% of Lost Status correct. This is not a good model to use to figure out what the status would be from our data. This is mainly due to the lack of data in certain key fields. I go through these in the Conclusions & Recommendations section below.

Conclusion & Recommendations

Recommendations on improving data

- Only 7 product groups out of 13 product groups listed where present in the pipeline data. Furthermore, 2 of these product groups where only listed once, effectively making it so 5 product groups could only be analysed. I would suggest in collecting data for these other product groups for further analysis.
- Only 11 sales peoples' sales where recorded, and most of the pipeline_data given are from 4-5 sales people. This heavily skews our data whilst looking at manager and sales persons contributions relatively. I would suggest to look into whether they are unrecorded sales by different sales peoples or whether some sales are incorrectly listed as some other sales persons sale.
- Add a quantity column to show how much of the product was sold.
- Customer Demographics: Potential Customer column provides no insight as most of potential customers are unique. Collect information about the customers, such as industry, location, or company size. This information can provide insights into customer behavior and preferences.
- Adding on to the previous points, Customer Interactions and Feedback & surveys are another good way of collecting more data. Feedback can include reasons for

choosing or not choosing the product/service, satisfaction levels, or suggestions for improvement. Customer Interactions could be: number of meetings, calls, or emails exchanged, as well as the duration of these interactions.

- Salesperson Experience: Include data on the experience level or performance metrics of the salesperson handling the opportunity. This can help capture the impact of salesperson expertise on the outcome.
- Competitive Analysis: Gather data on competitors and their activities in the market. This can provide insights into competitive dynamics and help identify factors influencing the outcome.

Recommendations on what was found through data analysis

- Q3 preforms better than Q4 in the number of sales and total contract value department. Further analysis can be done to see why this is the case.
- Mental health products provide the least win rate and average Contract Values. As it is our second most contracted product group, I would considering in to looking further on why the conversion rate is so low.
- Data is not sufficent enough to see why some contracts are lost or won, more data is required.
- Brokers provide a big etch over Direct opportunities, I would suggest to start to look at why Direct is much less profitable and consider finding ways on improving this method.
- Out of the four managers, Mr Ware's team performed the weakest. As the other teams had many more recorded sales, Mr Ware;s team had the same amount of sales as Leona Pugh, but with much less revenue. I would look further on why his team has such a low performance, keeping in mind that some sales may have not been recorded.
- Leona Pugh's team has higher revenue despite selling less products than Adele Pearsons team and selling the same products. Why is this? Could Tess Church's performance be a point of interest?
- Kaine Thomas' team has the highest win ratio and good amount of products sold. I would highly suggest to look into how Kaine's team is performing well and instruct other managers of any insight found.

Final review

The following Business goals were reviewed and analysed as following:

- Validate the data : This one done at the start and the data was cleaned and validated
- Where the revenue is coming from : The report delves deep into revenue connection for each variable.
- Understand what the data means : Throughout the report, the data is well looked at and interepted.
- Anything that can help in the world of sales : The report looks at sales people and sales managers at their performance and statistics.

- Suggestions on how the data system can be improved: Outlined in the section above on how the data can be approved.