Appendix B (Code and Documentation)

The code will be stated in blue

Loading of necessary libraries

```
#load necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from scipy.stats import chi2_contingency
from statsmodels.stats.proportion import proportions_ztest
import seaborn as sns
```

Documentation:

1. We loaded the necessary Python libraries and modules that would be used in subsequent codes

Data Cleaning for Survey Data

Code:

```
#loading the given dataset from Ministry of Health survey
MOH_data = pd.read_csv('lung cancer survey.csv')

#it was observed there are a series of rows where there are missing values across all columns at the bottom of the data frame
plt.figure(figsize=(12, 6))
sns.heatmap(MOH_data.isna(), cbar=False, cmap='viridis')  # 'viridis' is the color map
plt.title("Missing Values Heatmap")
plt.show()

#removing rows with missing values
MOH_data = MOH_data.dropna()

#removed underscores from column names to standardised
MOH_data.columns = MOH_data.columns.str.replace('_', ' ')
```

Documentation:

1. We loaded the given lung cancer dataset to a Pandas data frame called “MOH_data”.
2. We then created a heatmap using plt.figure() to identify the location of missing values within the data frame.
3. It was observed that there was a segment of missing values across all columns in the lower part of the dataframe.
4. We then removed the entire chunk of missing values from the dataframe using the dropna() method
5. Additionally, we standardised the column names in the data frame by removing underscores where present

Figure 1: Ten Most Frequent Incidences of Cancers by Gender (2018-2022)

Code:

```
#Loading dataset from Singapore Cancer Registry Annual Report 2022 and selecting the sheet for cancer incidence as a pandas dataframe
cancer_incidence = pd.read_excel('National Cancer Registry 2022.xlsx', sheet_name='Table 1.1.2 (Incidence)')

# Filter data for Male and Female
male_incidence_data = cancer_incidence[cancer_incidence['Gender'] == 'Male']
female_incidence_data = cancer_incidence[cancer_incidence['Gender'] == 'Female']

# Create subplots
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot for Male
bars_male = axs[0].bar(male_incidence_data['Cancer Type'], male_incidence_data['Count'], color='blue')
axs[0].set_title('Male')
axs[0].set_ylabel('Number of Incidences')
axs[0].set_xlabel('Cancer Type')
axs[0].set_xticklabels(male_incidence_data['Cancer Type'], rotation=45, ha='right')

# Add count values on top of bars for Male
for bar in bars_male:
    axs[0].text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                f'{int(bar.get_height())}', ha='center', va='bottom')

# Plot for Female
bars_female = axs[1].bar(female_incidence_data['Cancer Type'], female_incidence_data['Count'], color='pink')
axs[1].set_title('Female')
axs[1].set_ylabel('Number of Incidences')
axs[1].set_xlabel('Cancer Type')
```

```
axs[1].set_xticklabels(female_incidence_data['Cancer Type'], rotation=45, ha='right')

# Add count values on top of bars for Female
for bar in bars_female:
    axs[1].text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                f'{int(bar.get_height())}', ha='center', va='bottom')

# Annotate Lung Cancer columns
axs[0].annotate('Lung Cancer is the 3rd most frequently occurring cancer for Male',
                xy=('Lung', 6100), xytext=(+115, +40), textcoords='offset points',
                ha='center', va='bottom', fontsize=8,
                arrowprops=dict(arrowstyle='->', color='black'))

axs[1].annotate('Lung Cancer is the 3rd most frequently occurring cancer for
Female',
                xy=('Lung', 4100), xytext=(+110, +50), textcoords='offset points',
                ha='center', va='bottom', fontsize=8,
                arrowprops=dict(arrowstyle='->', color='black'))

# Set the overall title
plt.suptitle("Figure 1: Ten Most Frequent Incidences of Cancers by Gender
(2018-2022)")

# Add gridlines
axs[0].yaxis.grid(True)
axs[1].yaxis.grid(True)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.96])  # Adjust layout to make space for the suptitle
plt.show()
```

Documentation:

1. Firstly, we wanted to visualise the dataset we retrieved from the National Cancer Registry on the incidence rates of the ten most frequent incident cancers.
2. We then compiled the data downloaded into one excel sheet for ease of access.
3. Next, we loaded the data into python using pandas into a pandas dataframe.
4. We then filtered the data based on gender into **male_incidence_data** and **female_incidence_data**.
5. We then initialised matplotlib to set up 2 subplots adjusting figsize to 12,6 to fit nicely into the frame.
6. The bar plot for males was named **bars_male**, and we changed the title and x and y labels. We also used a for loop to add count values on top of the bars for males. Then, we repeated the same process for females.
7. We used axs[].annotate to annotate the lung cancer columns to show how frequently occuring it was compared to other common types of cancers.
8. We set the overall title of the plot with plt.suptitle() and added gridlines with axs[0].yaxis.grid(True).
9. Finally, we adjusted the layout with plt.tight_layout() and displayed the plot with plt.show().

Figure 2: Cancer Deaths by Gender (2018-2022)

Code:

```
# Load the dataset for cancer mortality from the Singapore Cancer Registry Annual Report 2022
cancer_mortality = pd.read_excel('National Cancer Registry 2022.xlsx', sheet_name='Table 1.1.2 (Mortality)')

# Filter data for Male and Female
male_mortality_data = cancer_mortality[cancer_mortality['Gender'] == 'Male']
female_mortality_data = cancer_mortality[cancer_mortality['Gender'] == 'Female']

# Create subplots
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot for Male
bars_male = axs[0].bar(male_mortality_data['Cancer Type'], male_mortality_data['Count'], color='blue')
axs[0].set_title('Male')
axs[0].set_ylabel('Number of Mortalities')
axs[0].set_xlabel('Cancer Type')
axs[0].set_xticklabels(male_mortality_data['Cancer Type'], rotation=45, ha='right')

# Add count values on top of bars for Male
for bar in bars_male:
    axs[0].text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                f'{int(bar.get_height())}', ha='center', va='bottom')

# Plot for Female
bars_female = axs[1].bar(female_mortality_data['Cancer Type'], female_mortality_data['Count'], color='pink')
axs[1].set_title('Female')
axs[1].set_ylabel('Number of Mortalities')
axs[1].set_xlabel('Cancer Type')
axs[1].set_xticklabels(female_mortality_data['Cancer Type'], rotation=45, ha='right')
```

```
# Add count values on top of bars for Female
for bar in bars_female:
    axs[1].text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                f'{int(bar.get_height())}', ha='center', va='bottom')

# Annotate column for Lung Cancer
axs[0].annotate(f'Lung Cancer has the most number of mortalities for Male',
                xy=('Lung', 3900), xytext=(+180, -40), textcoords='offset points',
                ha='center', va='bottom', fontsize=8,
                arrowprops=dict(arrowstyle='->', color='black'))

axs[1].annotate(f'Lung Cancer has the 3rd most number of mortalities for Female',
                xy=('Lung', 2180), xytext=(+100, +15), textcoords='offset points',
                ha='center', va='bottom', fontsize=8,
                arrowprops=dict(arrowstyle='->', color='black'))

# Set the overall title
plt.suptitle("Figure 2: Cancer Deaths by Gender (2018-2022)")

# Add gridlines
axs[0].yaxis.grid(True)
axs[1].yaxis.grid(True)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.96])  # Adjust layout to make space for the suptitle
plt.show()
```

Documentation:

1. Firstly, we wanted to visualise the dataset we retrieved from the National Cancer Registry on the mortality rates of the ten most frequent incident cancers.
2. We then compiled the data downloaded into one excel sheet for ease of access.
3. Next, we loaded the data into python as a pandas dataframe using the pandas library.
4. We then filtered the data based on gender into **male_mortality_data** and **female_mortality_data**.
5. We then initialised matplotlib to set up 2 subplots adjusting figsize to 12,6 to fit nicely into the frame.
6. The bar plot for males was named **bars_male**, and we changed the title and x and y labels. We also used a for loop to add count values on top of the bars for male. Then, we repeated the same process for females.
7. We used axs[].annotate to annotate the lung cancer columns to show how the mortality rate of lung cancer compares with other common cancers.
8. We set the overall title of the plot with plt.suptitle() and added gridlines with axs[0].yaxis.grid(True).
9. Finally, we adjusted the layout with plt.tight_layout and displayed the plot with plt.show().

Figure 3: Crude Incidence Rate, Mortality Rate, and Five-year Age-Standardised Relative Survival Rate for Lung Cancer for Males (1970-2020).

```
#loading dataset from Singapore Cancer Registry Annual Report 2022 and selecting the sheet for cancer incidence as a pandas dataframe
male_incidence_mortality_overtime = pd.read_excel('National Cancer Registry 2022.xlsx', sheet_name='Table 3.1.1 (Lung Cancer)')

# Remove brackets and asterisks from column names
male_incidence_mortality_overtime.columns = (
    male_incidence_mortality_overtime.columns
    .str.replace(r'\s*\(.*?\)\*?', '', regex=True)  # Remove brackets and asterisks
    .str.strip()  # Remove leading/trailing whitespace
)

# Function to clean specific columns' values
def clean_values(column):
    return column.str.extract(r'(\d+\.\d+)')[0].astype(float)  # Extract numeric values

# Clean values in the specific columns
male_incidence_mortality_overtime['CIR'] =
clean_values(male_incidence_mortality_overtime['CIR'])
male_incidence_mortality_overtime['CMR'] =
clean_values(male_incidence_mortality_overtime['CMR'])
male_incidence_mortality_overtime['ASRS'] =
clean_values(male_incidence_mortality_overtime['ASRS'])

# Create a figure and axis objects
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot CMR and CIR on the left y-axis
ax1.plot(male_incidence_mortality_overtime['Year'],
male_incidence_mortality_overtime['CMR'],
        color='blue', marker='o', label='CMR (LHS)')
```

```
ax1.plot(male_incidence_mortality_overtime['Year'],
male_incidence_mortality_overtime['CIR'],
        color='green', marker='o', label='CIR (LHS)')
ax1.set_xlabel("Year", fontsize=14, labelpad = 15) #change made here to increase
gap from xlabel
ax1.set_ylabel("CMR and CIR (per 100,000 population)", fontsize=12)
ax1.tick_params(axis='y', labelcolor='black')
ax1.spines['left'].set_linewidth(2)
ax1.spines['left'].set_color('black')
ax1.tick_params(direction='out', width=2)
ax1.set_ylim(0,80)

# Tilt the x-axis labels by 45 degrees
plt.xticks(rotation=45, fontsize = 8) #fontsize edited



# Create a second y-axis for ASRS
ax2 = ax1.twinx()
ax2.plot(male_incidence_mortality_overtime['Year'],
male_incidence_mortality_overtime['ASRS'],
        color='orange', marker='o', label='ASRS (RHS)')
ax2.set_ylabel("ASRS (%)", fontsize=12)
ax2.tick_params(axis='y', labelcolor='black')
ax2.spines['right'].set_linewidth(2)
ax2.spines['right'].set_color('black')
ax2.tick_params(direction='out', width=2)
ax2.set_ylim(0,40)

# Add Gridlines
ax1.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.5)
ax2.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.5)

# Title and Legend
```

```
ax1.set_title("Figure 3: Crude Incidence Rate, Mortality Rate, and Five-year
Age-Standardised Relative Survival Rate \nfor Male Lung Cancer Patients
(1970-2020)",
          pad = 15, fontsize = 12) #change made here to max labelling easier

# Combine legends from both axes in the top left corner
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines + lines2, labels + labels2, loc='upper left')


# Add text annotation at the bottom
plt.figtext(0.5, 0.03, "CMR: Crude Mortality Rate\nCIR: Crude Incidence
Rate\nASRS: Age-standardised Survival Rate",
           ha='center', fontsize=10)

# Adjust layout to make room for the text
plt.tight_layout(rect=[0, 0.1, 1, 1])  # Adjust layout to fit text

# Show the plot
plt.show()
```

Documentation:

1. This visualisation was meant to depict the trends between 3 factors:
   - Male Crude Mortality rate
   - Male Crude Incidence rate
   - Male Age-Standardised Survival Rate
2. First, we assigned the data from the 'National Cancer Registry 2022.xlsx, specifically the sheet 'Table 3.1.1 (Lung Cancer)' into **male_incidence_mortality_overtime.**
3. Because the original documentation used a mean value and confidence range, we removed these through the .str.replace function, looking for brackets and asterisks as well as the .str.strip function to remove any leading or trailing whitespace.
4. After that, in order to only get numeric values, we defined a function called clean_values that would go through a column to extract only numeric values.
5. After that, the function was run through the 'CIR', 'CMR' and 'ASRS' within the **male_incidence_mortality_overtime** and was used to overwrite the values within the aforementioned columns.
6. Then we began plot visualisation by creating a subplot with size 10 by 6.
7. We then plotted CMR and CIR on the left y axis, using different colors, blue and green respectively.
8. The labels and years were set and then it was given the limit 0 to 80.
9. After that, due to the length of the values on the x-axis, they were adjusted 45 degrees and given a smaller font size of 8.
10. Because ASRS had different values assigned to it, mainly by percentage as opposed to the per 100,000 population in the left y axis, a new axis was made through ax1.twinx() which was assigned into **ax2.** The values in 'ASRS' were given the color orange and drawn on the line plot.
11. After that, for better clarity, gridlines were made with linewidth 0.5 and alpha 0.5 with a dashed linestyle.
12. The title was then set, with pad = 15 and fontsize = 12 as the values, because the title would be too close to the plot.
13. After that a legend was made for better illustration, which was then combined with the legend from both the axes and placed in the upper left.

14. An extra annotation was placed at the bottom through plt.figtext because CMR, CIR, ASRS would need to be explained as it was not clearcut what it meant.
15. Then a tight_layout was added to make room for the text and finally plt.show was used to show the plot.

Figure 4: Crude Incidence Rate, Mortality Rate, and Five-year Age-Standardised Relative Survival Rate for Lung Cancer for Females (1970-2020).

```
#loading dataset from Singapore Cancer Registry Annual Report 2022 and selecting the sheet for cancer incidence as a pandas dataframe
female_incidence_mortality_overtime = pd.read_excel('National Cancer Registry 2022.xlsx', sheet_name='Table 3.1.2 (Lung Cancer)')

# Remove brackets and asterisks from column names
female_incidence_mortality_overtime.columns = (
    male_incidence_mortality_overtime.columns
    .str.replace(r'\s*\(.*?\)\*?', '', regex=True)  # Remove brackets and asterisks
    .str.strip()  # Remove leading/trailing whitespace
)

# Function to clean specific columns' values
def clean_values(column):
    return column.str.extract(r'(\d+\.\d+)')[0].astype(float)  # Extract numeric values

# Clean values in the specific columns
female_incidence_mortality_overtime['CIR'] =
clean_values(female_incidence_mortality_overtime['CIR'])
female_incidence_mortality_overtime['CMR'] =
clean_values(female_incidence_mortality_overtime['CMR'])
female_incidence_mortality_overtime['ASRS'] =
clean_values(female_incidence_mortality_overtime['ASRS'])

# Create a figure and axis objects
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot CMR and CIR on the left y-axis
ax1.plot(female_incidence_mortality_overtime['Year'],
female_incidence_mortality_overtime['CMR'],
        color='blue', marker='o', label='CMR (LHS)')
```

```
ax1.plot(female_incidence_mortality_overtime['Year'],
female_incidence_mortality_overtime['CIR'],
        color='green', marker='o', label='CIR (LHS)')
ax1.set_xlabel("Year", fontsize=14, labelpad = 15) #change made here to increase
gap from xlabel
ax1.set_ylabel("CMR and CIR (per 100,000 population)", fontsize=12)
ax1.tick_params(axis='y', labelcolor='black')
ax1.spines['left'].set_linewidth(2)
ax1.spines['left'].set_color('black')
ax1.tick_params(direction='out', width=2)
ax1.set_ylim(0,80)

# Tilt the x-axis labels by 45 degrees
plt.xticks(rotation=45, fontsize = 8) #fontsize edited

# Create a second y-axis for ASRS
ax2 = ax1.twinx()
ax2.plot(female_incidence_mortality_overtime['Year'],
female_incidence_mortality_overtime['ASRS'],
        color='orange', marker='o', label='ASRS (RHS)')
ax2.set_ylabel("ASRS (%)", fontsize=12)
ax2.tick_params(axis='y', labelcolor='black')
ax2.spines['right'].set_linewidth(2)
ax2.spines['right'].set_color('black')
ax2.tick_params(direction='out', width=2)
ax2.set_ylim(0,40)

# Add Gridlines
ax1.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.5)
ax2.grid(visible=True, which='both', linestyle='--', linewidth=0.5, alpha=0.5)

# Title and Legend
ax1.set_title("Figure 4: Crude Incidence Rate, Mortality Rate, and Five-year
Age-Standardised Relative Survival Rate \nfor Female Lung Cancer Patients",
```

```
            pad = 15, fontsize = 12)

# Combine legends from both axes in the top left corner
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines + lines2, labels + labels2, loc='upper left')

# Add text annotation at the bottom
plt.figtext(0.5, 0.03, "CMR: Crude Mortality Rate\nCIR: Crude Incidence
Rate\nASRS: Age-standardised Survival Rate",
            ha='center', fontsize=10)

# Adjust layout to make room for the text
plt.tight_layout(rect=[0, 0.1, 1, 1])  # Adjust layout to fit text

# Show the plot
plt.show()
```

Documentation:

1. This visualisation was meant to depict the trends between 3 factors:
   - Female crude mortality rate
   - Female crude incidence rate
   - Female Age-Standardised Survival Rate
2. First, we assigned the data we found 'National Cancer Registry 2022.xlsx',specifically the sheet 'Table 3.1.2 (Lung Cancer)' into **female_incidence_mortality_overtime.**
3. Because the original documentation used a mean value and confidence range, we removed these through the .str.replace function, looking for brackets and asterisks as well as the .str.strip function to remove any leading or trailing whitespace.
4. After that, in order to only get numeric values, we defined a function called clean_values that would go through a column to extract only numeric values.
5. After that, the function was run through the 'CIR', 'CMR' and 'ASRS' within the **female_incidence_mortality_overtime** and was used to overwrite the values within the aforementioned columns.
6. Then we began plot visualisation by creating a subplot with size 10 by 6.
7. We then plotted CMR and CIR on the left y-axis, using different colors, blue and green respectively.
8. The labels and years were set and then it was given the limit 0 to 80.
9. After that, due to the length of the values on the x-axis, they were adjusted 45 degrees and given a smaller font size of 8.
10. Because ASRS had different values assigned to it, mainly by percentage as opposed to the per 100,000 population in the left y-axis, a new axis was made through ax1.twinx() which was assigned into **ax2.** The values in 'ASRS' were given the color orange and drawn on the line plot.
11. After that, for better clarity, gridlines were made with linewidth 0.5 and alpha 0.5 with a dashed linestyle.
12. The title was then set, with pad = 15 and fontsize = 12 as the values, because the title would be too close to the plot.
13. After that a legend was made for better illustration, which was then combined with the legend from both the axes and placed in the upper left.

14. An extra annotation was placed at the bottom through plt.figtext because CMR, CIR, ASRS would need to be explained as it was not clear cut what it meant.
15. Then a tight_layout was added to make room for the text and finally plt.show was used to show the plot.

Figure 5: Estimated Number of Deaths (2022-2035), Both Genders - Trachea, Bronchus, and Lung Cancer

Code:

```
# Load the dataset
mortality_prediction = pd.read_excel('WHO - Cancer Mortality and Incidence Prediction.xlsx', sheet_name='Table 1 (Mortality Prediction)')

# Prepare data for plotting
years = mortality_prediction['Year']
predictions = mortality_prediction['Prediction Mortality']

# Create the bar chart with wider bars and values at the top
plt.figure(figsize=(6, 6))
bars = plt.bar(years, predictions, color='blue', width=2)

# Add the values at the top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), ha='center', va='bottom')

# Customize the chart
plt.title('Figure 5: Estimated Number of Deaths (2022-2035), Both Genders - Trachea, Bronchus, and Lung Cancer')
plt.xlabel('Year')
plt.ylabel('Number of Deaths')
plt.xticks(years)  # Show all years on x-axis
plt.grid(axis='y', linestyle='--', alpha=1)

# Show the updated plot
plt.tight_layout()
plt.show()
```

Documentation:

1. For this code, we wanted to visualise the dataset we retrieved from WHO on predictions on Mortality Rate from Trachea, Bronchus, and Lung Cancer.
2. Firstly, we assigned **mortality_prediction** as data from the excel 'WHO - Cancer Mortality and Incidence Prediction.xlsx', specifically sheet Table 1 (Mortality Prediction).
3. In preparing the data, we assigned the 'Year' column from **mortality_prediction** into **years** and the 'Prediction Mortality' column from **mortality_prediction** into **predictions.**
4. After that, we began the preparation for the plot by creating a figure with size 6 by 6 and assigned **bars** as the bar plot with variable **years** and **predictions.** Further customisation with it being blue and width being 2 was added.
5. After that, in order to see the values at the top of the respective bars, we ran a for loop where within **bars**, it would place a text above each bar by getting its height and x values.
6. In order to customise the plot further, a title was added together with the x and y axis labels and the years involved.
7. For further visual aid, a grid was added with alpha being 1 and linestyle being dashed.
8. Finally, we applied the tight layout for better visualisation and thus used plt.show() to output the plot.

Figure 6: Estimated Number of New Incidences (2022-2035), Both Genders - Trachea, Bronchus, and Lung Cancer

Code:

```
mortality_prediction = pd.read_excel('WHO - Cancer Mortality and Incidence Prediction.xlsx', sheet_name='Table 2 (Incidence Prediction)')

# Prepare data for plotting
years = mortality_prediction['Year']
predictions = mortality_prediction['Prediction Incidence']

# Create the bar chart with wider bars and values at the top
plt.figure(figsize=(6, 6))
bars = plt.bar(years, predictions, color='blue', width=2)

# Add the values at the top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), ha='center', va='bottom')

# Customize the chart
plt.title('Figure 6: Estimated Number of New Incidences (2022-2035), Both Genders - Trachea, Bronchus, and Lung Cancer')
plt.xlabel('Year')
plt.ylabel('Number of Incidences')
plt.xticks(years)  # Show all years on x-axis
plt.grid(axis='y', linestyle='--', alpha=1)

# Show the updated plot
plt.tight_layout()
plt.show()
```

Documentation:

1. For this code, we wanted to visualise the dataset we retrieved from WHO on predictions on Mortality Rate from Trachea, Bronchus, and Lung Cancer.
2. Firstly, we assigned **mortality_prediction** as a data from the excel 'WHO - Cancer Mortality and Incidence Prediction.xlsx', specifically sheet Table 1 (Mortality Prediction).
3. In preparing the data, we assigned the 'Year' column from **mortality_prediction** into **years** and the 'Prediction Mortality' column from **mortality_prediction** into **predictions.**
4. After that, we began the preparation for the plot by creating a figure with size 6 by 6 and assigned **bars** as the bar plot with variable **years** and **predictions.** Further customisation with it being blue and width being 2 was added.
5. After that, in order to see the values at the top of the respective bars, we ran a for loop where within **bars**, it would place a text above each bar by getting its height and x values.
6. In order to customise the plot further, a title was added together with the x and y axis labels and the years involved.
7. For further visual aid, a grid was added with alpha being 1 and linestyle being dashed.
8. Finally, we applied the tight layout for better visualisation and thus used plt.show() to output the plot.

Figure 7: Probability of Having Lung Cancer by Lifestyle Choice

```
# Calculate the counts and proportions for each group
smoking_only = MOH_data[(MOH_data['SMOKING'] == 1) & (MOH_data['ALCOHOL
CONSUMING'] == 0)]['LUNG_CANCER'].mean()
alcohol_only = MOH_data[(MOH_data['SMOKING'] == 0) & (MOH_data['ALCOHOL
CONSUMING'] == 1)]['LUNG_CANCER'].mean()
both = MOH_data[(MOH_data['SMOKING'] == 1) & (MOH_data['ALCOHOL
CONSUMING'] == 1)]['LUNG_CANCER'].mean()
none = MOH_data[(MOH_data['SMOKING'] == 0) & (MOH_data['ALCOHOL
CONSUMING'] == 0)]['LUNG_CANCER'].mean()

# Prepare data for the bar chart and convert proportions to percentages
groups = ['Smoking Only', 'Alcohol Only', 'Both', 'None']
proportions = [smoking_only * 100, alcohol_only * 100, both * 100, none * 100]  #
Convert to percentages

# Create a DataFrame to sort the data
data = pd.DataFrame({'Group': groups, 'Proportion': proportions})
data = data.sort_values(by='Proportion')  # Sort in ascending order

# Create the bar chart with different colors
colors = ['red', 'blue', 'green', 'orange']
bars = plt.bar(data['Group'], data['Proportion'], color=colors)

# Add percentage labels on top of the bars
for i, proportion in enumerate(data['Proportion']):
    plt.text(i, proportion, f'{proportion:.2f}%', ha='center', va='bottom')

# Set chart labels and title
plt.ylabel('Probability of having Lung Cancer (%)')
plt.title('Figure 7: Probability of Having Lung Cancer by Lifestyle Choice', y = 1.005)
#raised the title up abit
```

```
# Add gridlines and set the upper limit for y-axis
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.ylim(0, 100)

# Show the plot
plt.show()

#######

# Check using Chi-Square Test to see if the values are significant:
# Create a contingency table for the four lifestyle groups and lung cancer
# Calculate the counts for each category (Lung Cancer: 0 or 1)
smoking_only_counts = MOH_data[(MOH_data['SMOKING'] == 1) &
(MOH_data['ALCOHOL CONSUMING'] == 0)]['LUNG_CANCER'].value_counts()
alcohol_only_counts = MOH_data[(MOH_data['SMOKING'] == 0) &
(MOH_data['ALCOHOL CONSUMING'] == 1)]['LUNG_CANCER'].value_counts()
both_counts = MOH_data[(MOH_data['SMOKING'] == 1) & (MOH_data['ALCOHOL
CONSUMING'] == 1)]['LUNG_CANCER'].value_counts()
none_counts = MOH_data[(MOH_data['SMOKING'] == 0) & (MOH_data['ALCOHOL
CONSUMING'] == 0)]['LUNG_CANCER'].value_counts()

# Ensure all categories have both '0' and '1' counts
smoking_only = [smoking_only_counts.get(0, 0), smoking_only_counts.get(1, 0)]
alcohol_only = [alcohol_only_counts.get(0, 0), alcohol_only_counts.get(1, 0)]
both = [both_counts.get(0, 0), both_counts.get(1, 0)]
none = [none_counts.get(0, 0), none_counts.get(1, 0)]

# Create the contingency table
contingency_table = pd.DataFrame([smoking_only, alcohol_only, both, none],
                                 index=['Smoking Only', 'Alcohol Only', 'Both', 'None'],
                                 columns=['No Lung Cancer', 'Lung Cancer'])

# Apply Chi-Square Test
chi2, p, dof, expected = chi2_contingency(contingency_table)
```

```
# Print Chi-Square Test results
print(f'Chi-Square Statistic: {chi2:.4f}")
print(f'P-value: {p:.8f}")

# Check for significance
if p < 0.05:
    print("The distribution of lung cancer across the four groups is statistically
significant (p < 0.05).")
else:
    print("The distribution of lung cancer across the four groups is not statistically
significant (p >= 0.05).")
```

Documentation

1. With this code, we wanted to visualise the probability of having lung cancer according to each person's lifestyle choice: Non-smokers & Non-drinkers, smokers, drinkers and those who smoke & consume alcohol.
2. Firstly, we separated the first four categories and defined them.
   - The default category in the code was non smokers and non drinkers and we put it under 'none'.
   - If they smoked and did not consume alcohol, they were placed into the category 'smoking_only'.
   - For those who only consumed alcohol, they were placed into 'alcohol_only'.
   - If they smoked and drank alcohol, they were placed into 'Both'.
3. They were grouped and assigned into **groups** containing 'Smoking Only', 'Alcohol Only', 'Both' and 'None'.
4. In order to calculate percentages, we multiplied values within 'smoking_only', 'alcohol_only', 'both' and 'none' by 100 each and assigned them into **proportions.**
5. They were then put into dataframe **data** where columns 'Group' were values from **groups** and 'Proportion' was values from **proportion.** They were then sorted by 'Proportion' in ascending order.
6. To prepare the barplot, **colors** was a variable made containing 'red', 'blue', 'green' and 'orange'.
7. **Bars** was also made to include the bar plot data where it was differentiated by **colors.**
8. In order to see the exact probability values, a for loop was made where when enumerated through 'Proportion', the value found had a text made from it, which showed the value subject to 2 decimal places.
9. After that, we applied titles and labels through plt.title, plt.xlabel, and plt.ylabel.
10. We then added a grid, with linewidth and alpha at 0.7 and linestyle as a dashed style.
11. We then applied a y axis limit from 0 to 100.
12. Finally, we used plt.show() to output the graph.
13. In order to know if the value was statistically significant, we performed a chi square test. This was done by creating a contingency table through finding the

counts of each category and assigning it **smoking_only_counts**, **alcohol_only_counts, both_counts** and **none_counts.**

14. As the values were all 1 and 0, a test was done to check if it only got those values through the .get function.
15. A contingency table was then made through pd.Dataframe.
16. The chi square test was then applied with **chi2, p, dof, expected** as the values that wanted to be found.
17. The results we wanted were the Chi Square Statistic which was 437.3603 and p-value which was 0.00000000.
18. Then we checked if it was accurate and the results showed: The distribution of lung cancer across the four groups is statistically significant ($p < 0.05$).

Figure 8: Lung Cancer Density Across Age by Gender

Code:

```
# Map the GENDER values to 'Male' and 'Female'
MOH_data['GENDER'] = MOH_data['GENDER'].map({0: 'Female', 1: 'Male'})

# Plot KDE for both genders, remove the fill
sns.kdeplot(data=MOH_data, x='AGE', hue='GENDER', common_norm=False,
fill=False)

# Highlight age range 42 to 52 with a yellow box
plt.axvspan(42, 52, color='yellow', alpha=0.3)

#Set axis labels and titles
plt.xlabel('Age')
plt.ylabel('Number of Incidences per 100 people')
plt.title('Figure 8: Lung Cancer Incidences Across Age by Gender')

# Modify y-axis tick labels by multiplying the values by 100
yticks = plt.gca().get_yticks()  # Get current y-tick values
plt.gca().set_yticklabels([f'{int(tick * 100)}' for tick in yticks])  # Multiply by 100

#Add plot annotations
plt.annotate(
    'Most likely to get lung cancer',  # Annotation text
    xy=(43, 0.044),  # Point where the annotation will be placed
    xytext=(13, 0.048),  # Position of the text
    arrowprops=dict(arrowstyle='->', color='black', lw=1.0),  # Arrow properties
    fontsize=8,
    color='black'
)

# Show the plot
plt.show()
```

Documentation

1. From the ‘MOH_data’ data set, we wanted to see the prevalence of lung cancer across age by gender using a Kernel Density Estimate (KDE) plot.
2. We loaded the dataset using pandas as MOH_data.
3. We then mapped the GENDER values from numeric (0 and 1) to categorical (Female and Male).
4. We imported the seaborn library as sns, and used the sns.kdeplot() function to generate a KDE plot for the AGE variable, separated by GENDER, with no fill. We added suitable x and y labels (‘Age’ and ‘Number of Incidences per 100 people’), as well as the title (‘Figure 8: Lung Cancer Incidences Across Age by Gender’).
5. We highlighted the AGE range of 42 to 52 in yellow using plt.axvspan().
6. We modified the y-axis tick labels by multiplying the values by 100 to change the y-axis to represent the number of incidences per 100 individuals instead of the probability density.
7. We also added an annotation pointing at the highlighted area using plt.annotate(), with the text (Most likely to get lung cancer).
8. Finally, we displayed the plot using plt.show().

Figure 9: Pie Chart of Proportion of Lung Cancer Cases Given the Presence of Each Symptom

Code:

```
# List of symptoms
symptoms = ['FATIGUE ', 'WHEEZING', 'COUGHING',
            'SHORTNESS OF BREATH', 'SWALLOWING DIFFICULTY',
            'CHEST PAIN', 'YELLOW_FINGERS']

# Create subplots for multiple pie charts
fig, axes = plt.subplots(2, 4, figsize=(16, 8))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Loop through each symptom and create a pie chart
for i, symptom in enumerate(symptoms):
    # Calculate the proportion of lung cancer for each symptom (mean for binary 0/1)
    lung_cancer_proportion = MOH_data[MOH_data[symptom] ==
1]['LUNG_CANCER'].mean()

    # Proportion for pie chart
    proportions = [lung_cancer_proportion, 1 - lung_cancer_proportion]

    # Create the pie chart without labels on the pie itself
    axes[i].pie(proportions, autopct='%1.1f%%', startangle=90, colors=['orange',
'lightgrey'])

    # Title for each pie chart
    axes[i].set_title(f'{symptom}')

# Add the main title
plt.suptitle('Figure 9: Proportion of Lung Cancer Cases Given the Presence of Each
Symptom', fontsize=16)
```

```
# Adjust layout to prevent overlap and raise the title
plt.subplots_adjust(top=0.85)  # Shift the title higher

# Create a custom legend and shift it slightly to the left
fig.legend(['Have Lung Cancer', 'No Lung Cancer'], loc='lower right', fontsize=12,
markerscale=1.5,
          bbox_to_anchor=(0.87, 0.25), fancybox=True, shadow=True)

# Remove the last subplot (the empty one)
axes[-1].axis('off')  # Hide the last subplot

# Show the pie charts
plt.show()

# Initialize a dictionary to hold results
ztest_results = {}

# Loop through each symptom to perform Two-Proportion Z-Test
for symptom in symptoms:
    # Calculate counts of lung cancer cases with and without the symptom
    count_with_symptom = MOH_data[(MOH_data[symptom] == 1) &
(MOH_data['LUNG_CANCER'] == 1)].shape[0]
    count_without_symptom = MOH_data[(MOH_data[symptom] == 0) &
(MOH_data['LUNG_CANCER'] == 1)].shape[0]

    # Total counts of individuals with and without the symptom
    total_with_symptom = MOH_data[MOH_data[symptom] == 1].shape[0]
    total_without_symptom = MOH_data[MOH_data[symptom] == 0].shape[0]

    # Perform the Z-test
    count = [count_with_symptom, count_without_symptom]
    nobs = [total_with_symptom, total_without_symptom]

    z_stat, p_value = proportions_ztest(count, nobs)
```

```
    # Store results
    ztest_results[symptom] = {
        'z_statistic': z_stat,
        'p_value': p_value
    }

# Convert results to DataFrame for easy viewing
ztest_results_df = pd.DataFrame(ztest_results).T

# Display the results
print(ztest_results_df)
```

Documentations:

1. From the 'MOH_data' data frame, we wanted to find out the probability of having lung cancer given that an individual has a particular symptom, particularly FATIGUE, WHEEZING, COUGHING, SHORTNESS OF BREATH, SWALLOWING DIFFICULTY, CHEST PAIN and YELLOW_FINGERS.
2. We listed down the symptoms of interest in the **symptoms** list.
3. We created subplots for multiple pie charts using plt.subplots() and assigned it to variable **fig, axes**. We also flattened the axes array using axes.flatten() for easier indexing.
4. We created a for loop to iterate over each symptom in the **symptoms** list, where the enumerate function returns both the index i and the value **symptom** for each iteration. In the for loop, MOH_data[MOH_data[symptom] ==1] filters the DataFrame to include only those rows where the current symptom is present (ie. equals 1). ['LUNG_CANCER'].mean calculates the mean of the 'LUNG_CANCER' column for the filtered rows. Since 'LUNG_CANCER' is binary (0 for no lung cancer, 1 for lung cancer), the mean value represents the proportion of individuals with lung cancer among those who have the specific symptom. A list named **proportions** is created that contains two values, the first is the proportion of lung cancer cases for the current symptom, and the second represents the proportion of individuals without lung cancer who have the symptom. The **labels** list creates a list of

labels for the pie chart segments, the first represents the proportion of individuals with lung cancer, and the second represents those without lung cancer. We call the function axes[i].pie() to create a pie chart on the subplot defined by the i-th index of the axes array, and axes[i].set_title sets the title of the current pie chart.

5. We added the main title using plt.suptitle() and adjusted the layout to prevent overlap using plt.tight_layout().
6. We created a custom legend using fig.legend() and removed the empty subplot with axes[-1].axis('off') and displayed the pie charts using plt.show().
7. To find out whether the proportion of lung cancer cases given that an individual has a particular symptom is statistically significant, we conducted a two-proportion Z-test by importing proportions_ztest from the statsmodels.stats.proportion library.
8. We created a dictionary **ztest_results** to hold our results.
9. We then used a for loop that iterates over each symptom in the **symptoms** list. We count how many individuals have both the symptom (ie. equals 1) and lung cancer (where 'LUNG_CANCER' equals 1) and store it in object **count_with_symptom,** and .shape[0] returns the number of rows which corresponds to the count of individuals who have the symptom and lung cancer. The count_without_symptom variable does the same except for individuals who do not have the symptom but do have lung cancer. total_with_symptom and total_without_symptom counts the total number of individuals who have the symptom and those who do not respectively. The count list contains the counts of lung cancer cases with and without the symptom and the nobs list contains the total counts of individuals with and without the symptom. The proportions_ztest function is called with the count and nobs lists as arguments. It returns the Z-statistic (z_stat) and the p-value (p_value) for the hypothesis test, which assesses whether the proportion of lung cancer is significantly different for individuals with the symptom compared to those without. **ztest_results[symptom]** stores the results of the current Z-test, and **ztest_results_df** converts results to a DataFrame for easy viewing, while print(ztest_results_df) displays the results.

Figure 10: Proportion of Smokers Influenced by Peer Pressure

```
# Filter data for smokers and calculate the counts of peer pressure
smokers = MOH_data[MOH_data['SMOKING'] == 1]
peer_pressure_counts = smokers['PEER PRESSURE'].value_counts()

# Create labels for the pie chart
labels = ['Peer Pressured', 'Not Peer Pressured']
sizes = [peer_pressure_counts[1], peer_pressure_counts[0]]  # Count of '1' (peer
pressured) and '0' (not peer pressured)
colors = ['#66b3ff', '#ff9999']
explode = (0.1, 0)  # "explode" the first slice (Peer Pressured)

# Plotting the pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, colors=colors, explode=explode, autopct='%1.1f%%',
startangle=140, shadow=True)
plt.title('Figure 10: Proportion of Smokers Influenced by Peer Pressure')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()

# Calculate the counts for each category
anxiety_smoker_count = len(MOH_data[(MOH_data['ANXIETY'] == 1) &
(MOH_data['SMOKING'] == 1)])
peer_pressure_smoker_count = len(MOH_data[(MOH_data['PEER PRESSURE'] ==
1) & (MOH_data['SMOKING'] == 1)])
anxiety_alcohol_count = len(MOH_data[(MOH_data['ANXIETY'] == 1) &
(MOH_data['ALCOHOL CONSUMING'] == 1)])

# Total counts of smokers and alcohol consumers
smoker_count = len(MOH_data[MOH_data['SMOKING'] == 1])
alcohol_count = len(MOH_data[MOH_data['ALCOHOL CONSUMING'] == 1])

# Calculate the probabilities
```

```
anxious_given_smoker = anxiety_smoker_count / smoker_count
peer_pressure_given_smoker = peer_pressure_smoker_count / smoker_count
anxiety_given_alcohol = anxiety_alcohol_count / alcohol_count

# Creating Contingency Tables for the Chi-Square Test
contingency_anxiety_smoking = pd.crosstab(MOH_data['ANXIETY'],
MOH_data['SMOKING'])
contingency_peer_pressure_smoking = pd.crosstab(MOH_data['PEER
PRESSURE'], MOH_data['SMOKING'])
contingency_anxiety_alcohol = pd.crosstab(MOH_data['ANXIETY'],
MOH_data['ALCOHOL CONSUMING'])

chi2_anxiety_smoking, p_anxiety_smoking, _, _ =
chi2_contingency(contingency_anxiety_smoking)
chi2_peer_pressure_smoking, p_peer_pressure_smoking, _, _ =
chi2_contingency(contingency_peer_pressure_smoking)
chi2_anxiety_alcohol, p_anxiety_alcohol, _, _ =
chi2_contingency(contingency_anxiety_alcohol)

# Display results for Anxiety, Peer Pressure, and Alcohol
print(f"Probability of Anxiety Given Smoker: {anxious_given_smoker:.4f}, P-value:
{p_anxiety_smoking:.4f}")
print(f"Probability of Peer Pressure Given Smoker:
{peer_pressure_given_smoker:.4f}, P-value: {p_peer_pressure_smoking:.4f}")
print(f"Probability of Anxiety Given Alcohol Consumption: {anxiety_given_alcohol:.4f},
P-value: {p_anxiety_alcohol:.4f}")
```

Documentation:

1. The purpose of this visualisation will be to see how many smokers are peer pressured.
2. Firstly, we create two variables: **smokers,** where it filters those who smoke from the data and also **peer_pressure_counts** which counts the number of people who had peer pressure.
3. To prepare the pie chart, we created labels and sizes, for example 'Peer Pressured' and 'Not Peer Pressured' as well as colors #66b3ff and #ff9999 to be used in the pie chart. We then used explode to create a slice, to show the separation better.
4. After, we made the pie chart by creating a figure with size of 8 by 6. We then created the pie chart through plt.pie where the start angle was adjusted for better visualisation and shadow was placed to be seen for visualisation purposes as well. Auto percentage was used to make it more exact for each slice being up to 1 decimal precision.
5. We then gave it the appropriate title of 'Figure 10: Proportion of Smokers Influenced by Peer Pressure', added an axis as equal to ensure the pie is drawn as a perfect simple.
6. We finally displayed the chart with plt.show().
7. In order to know if the value was statistically significant, we performed a chi square test.
8. To prepare for the test, counts were made with variables:
   -anxiety_smoker_count: counts of those with 'ANXIETY' and 'SMOKING'
   -peer_pressure_smoker_count: counts of those with 'PEER PRESSURE' and 'SMOKING'
   -anxiety_alcohol_count: counts of those with 'ANXIETY' and 'ALCOHOL CONSUMING'.
9. After that, to get the denominator, we do **smoker_count** which would be the number of smokers and **alcohol_count** which would be the number of those who consume alcohol.
10. Probabilities were then calculated and assigned into variables named **anxious_given_smoker, peer_pressure_given_smoker** and **anxiety_given_alcohol.**

11. Then, a contingency table through pd.crosstab with columns within **MOH_data** was made where the different variables meant:
    - **contigency_anxiety_smoking:** ‘ANXIETY’ & ‘SMOKING’
    - **contigency_peer_pressure:** ‘PEER PRESSURE’ & ‘SMOKING’
    - **contigency_anxiety_alcohol:** ‘ANXIETY’ & ‘ALCOHOL CONSUMING’
12. The chi square values were then calculated and to check for the results, the probability and p-values were printed out:
    - Probability of Anxiety Given Smoker: 0.4627 with its p-value being 0.2109.
    - Probability of Peer Pressure Given Smoker: 0.4979 with its p-value being 0.0124.
    - Probability of Anxiety Given Alcohol Consumption: 0.4555 with its p–value being 0.8872.

Figure 11: Share of Diagnosis Stage for Males and Females

Code:

```
# Load the datasets
male_diagnosis_stage = pd.read_excel('National Cancer Registry 2022.xlsx',
sheet_name='Table 3.2.1 (Lung Cancer)')
female_diagnosis_stage = pd.read_excel('National Cancer Registry 2022.xlsx',
sheet_name='Table 3.2.2 (Lung Cancer)')

# Create a figure with two subplots side-by-side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# Bar chart for male diagnosis stage (Skyblue color)
ax1.bar(male_diagnosis_stage['Stage'], male_diagnosis_stage['Percentage (%)'],
color='skyblue')
for i, percentage in enumerate(male_diagnosis_stage['Percentage (%)']):
    ax1.text(i, percentage + 0.5, f'{percentage:.2f}%', ha='center')
ax1.set_ylabel('Percentage (%)')
ax1.set_title('Share of Diagnosis Stage for Males')
ax1.grid(True, axis='y', linestyle='--', alpha=0.7)
ax1.set_ylim(0, 75)

# Bar chart for female diagnosis stage (Pink color)
ax2.bar(female_diagnosis_stage['Stage'], female_diagnosis_stage['Percentage (%)'],
color='pink')
for i, percentage in enumerate(female_diagnosis_stage['Percentage (%)']):
    ax2.text(i, percentage + 0.5, f'{percentage:.2f}%', ha='center')
ax2.set_ylabel('Percentage (%)')
ax2.set_title('Share of Diagnosis Stage for Females')
ax2.grid(True, axis='y', linestyle='--', alpha=0.7)
ax2.set_ylim(0, 75)

# Add a unified figure title
fig.suptitle('Figure 11: Share of Diagnosis Stage for Males and Females',
fontsize=16)
```

```
# Adjust layout to prevent overlap
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust rect to make room for the suptitle

# Show the combined plot
plt.show()
```

Documentation:

1. We wanted to find out the share of cancer diagnosis stages for both genders from the dataset retrieved from National Cancer Registry.
2. We used pandas to load the diagnosis stage data of both genders into male_diagnosis_stage and female_diagnosis_stage.
3. We then created a figure with two subplots side-by-side using plt.subplots()
4. We created a bar chart for male and female diagnosis stage with ax.bar(), then added a unified figure title using fig.suptitle().
5. Finally, we adjusted the layout using plt.tight_layout and displayed the plot with plt.show().