Housing Markets Trends for Inner Melbourne Suburbs

Project 1 – Team 2

Violet Bui, Elizabeth Dashwood, Lindsay McCulloch and Vrinda Patel

How was this data sourced?

CITY OF MELBOURNE

AUSTRALIA
Australian Bureau of Statistics

RESERVE BANK OF AUSTRALIA

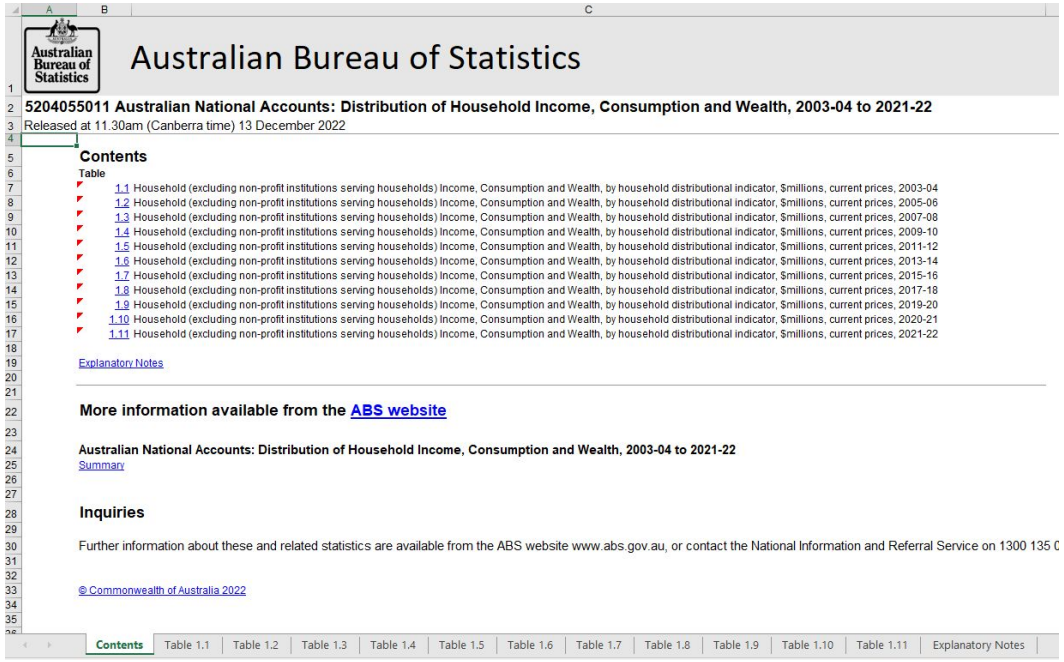

Residence Prices Against
Household Income

What trends can be observed in median income per 2 year period vs median residence cost per 2 year period?

THE DATA

The data source used for this analysis was the median household income census data from the Australian Bureau of Statistics (ABS).

This data was downloaded as a CSV file containing median household income data in two year increments from 2003-2004



Australian Bureau of Statistics

5204055011 Australian National Accounts: Distribution of Household Income, Consumption and Wealth, 2003-04 to 2021-22

Released at 11.30am (Canberra time) 13 December 2022

Contents

Table

1.1 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2003-04
1.2 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2005-06
1.3 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2007-08
1.4 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2009-10
1.5 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2011-12
1.6 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2013-14
1.7 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2015-16
1.8 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2017-18
1.9 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2019-20
1.10 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2020-21
1.11 Household (excluding non-profit institutions serving households) Income, Consumption and Wealth, by household distributional indicator, $millions, current prices, 2021-22

Explanatory Notes

More information available from the ABS website

Australian National Accounts: Distribution of Household Income, Consumption and Wealth, 2003-04 to 2021-22

Summary

Inquiries

Further information about these and related statistics are available from the ABS website www.abs.gov.au, or contact the National Information and Referral Service on 1300 135 0

© Commonwealth of Australia 2022

Contents | Table 1.1 | Table 1.2 | Table 1.3 | Table 1.4 | Table 1.5 | Table 1.6 | Table 1.7 | Table 1.8 | Table 1.9 | Table 1.10 | Table 1.11 | Explanatory Notes

```python
# Define the row indices to skip
rows_to_skip = list(range(0, 6)) + [7] + list(range(9, 26)) + list(range(27, 40)) + list(range(42, 127))

# Read ABS income data file and store into Pandas DataFrames
dfs = pd.read_excel(household_income_data, sheet_name=None, skiprows=rows_to_skip)

# Create an empty list to store DataFrames with tab names as a column
dfs_with_tab_name = []

# Flag to skip the first and last tab
skip_first_tab = True
skip_last_tab = False

# Iterate over each sheet in the dictionary
for sheet_name, df in dfs.items():
    if skip_first_tab:
        skip_first_tab = False
        continue  # Skip processing the first tab

    if skip_last_tab:
        break  # Exit the loop if the last tab is reached

    # Determine the column indices dynamically based on the actual number of columns
    num_cols = min(7, len(df.columns))

    # Ensure we don't exceed the number of columns in the DataFrame
    columns_to_read = list(range(num_cols))

    # Select only the specified columns
    df_selected = df.iloc[:, columns_to_read].copy()

    # Add a new column with the tab name to the selected DataFrame using loc
    df_selected.loc[:, 'Year/s'] = sheet_name

    # Append the modified DataFrame to the list
    dfs_with_tab_name.append(df_selected)

# Concatenate all DataFrames in the list into a single DataFrame
combined_df = pd.concat(dfs_with_tab_name, ignore_index=True)
```

INTERPRETATIONS OF THE DATA

Year/s	2003-2004	2005-2006	2007-2008	2009-2010	2011-2012	2013-2014	2015-2016	2017-2018	2019-2020
Household income									
Estimated number of households in population	7954585.0	8160856.0	8327818.0	8664857.0	8912566.0	9048583.0	9246191.0	9554316.0	10016972.0
Gross disposable income	536356.0	608301.0	725929.0	827246.0	939989.0	1022855.0	1095050.0	1165497.0	1282944.0
Total gross income	711537.0	819132.0	985903.0	1083258.0	1248033.0	1336473.0	1433237.0	1530133.0	1656039.0
Total income payable	175181.0	210831.0	259974.0	256011.0	308044.0	313618.0	338187.0	364636.0	373095.0

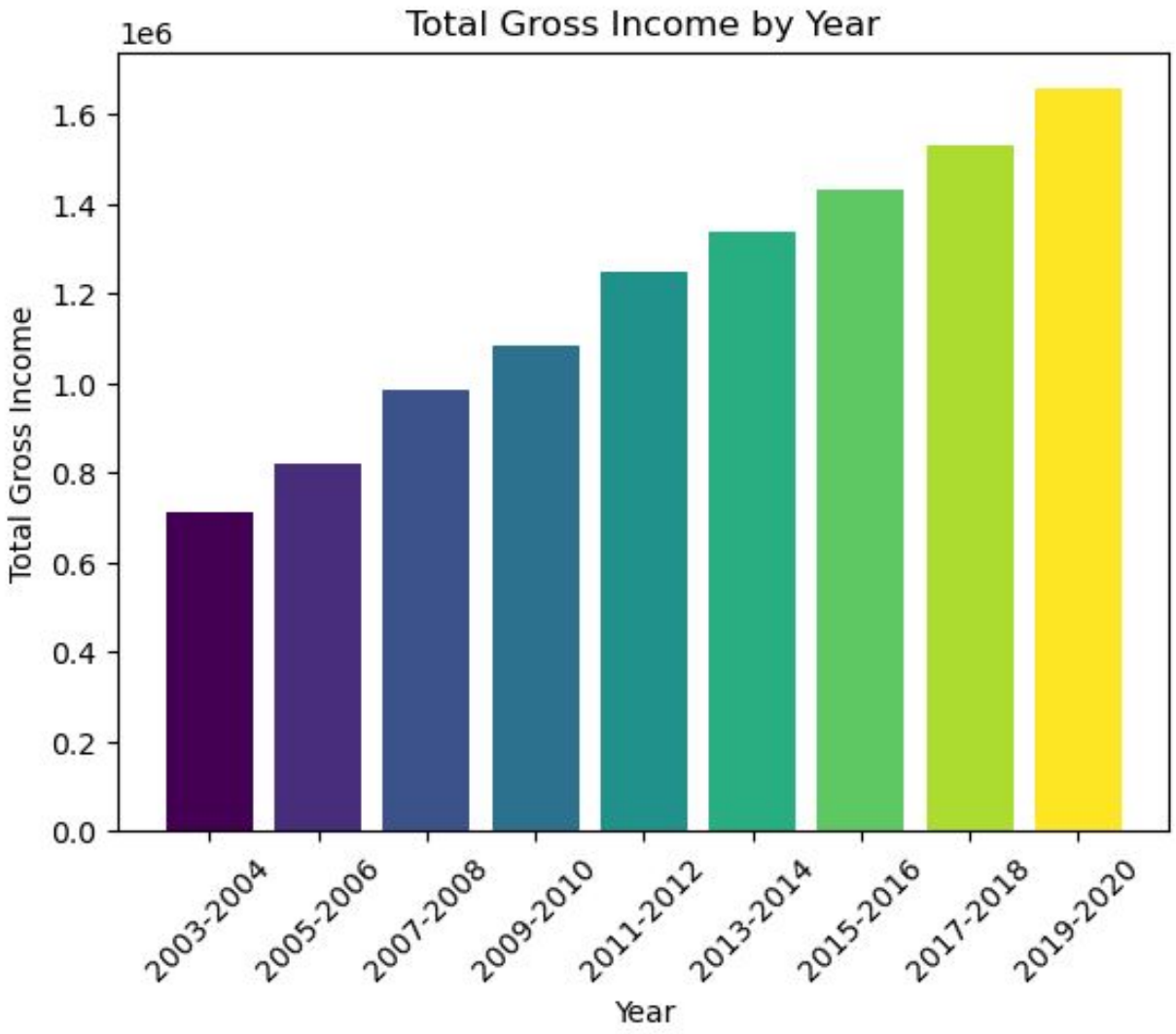
From the above pivot table we can see that Total Gross Income has more than doubled at a total increase of *2.33 from 2003-2020

Note: all income points are measured in Millions (m) and years measured in two year increments.

```
# Create a pivot table with 'Category' as rows, 'Year' as columns, and 'Value' as values
pivot_df = cleaned_df.pivot_table(columns='Year/s', values='All households (b)', aggfunc='sum')
```

```
# rename the index as "Household income"
pivot_df = cleaned_df.pivot_table(index='Unnamed: 0', columns='Year/s', values='All households (b)', aggfunc='sum')
pivot_df = pivot_df.rename_axis('Household income')
```

```
household_income_df = pivot_df
household_income_df.head()
```



This graph depicts the **Total Gross Income** in Victoria from 2003-2020

Observations:

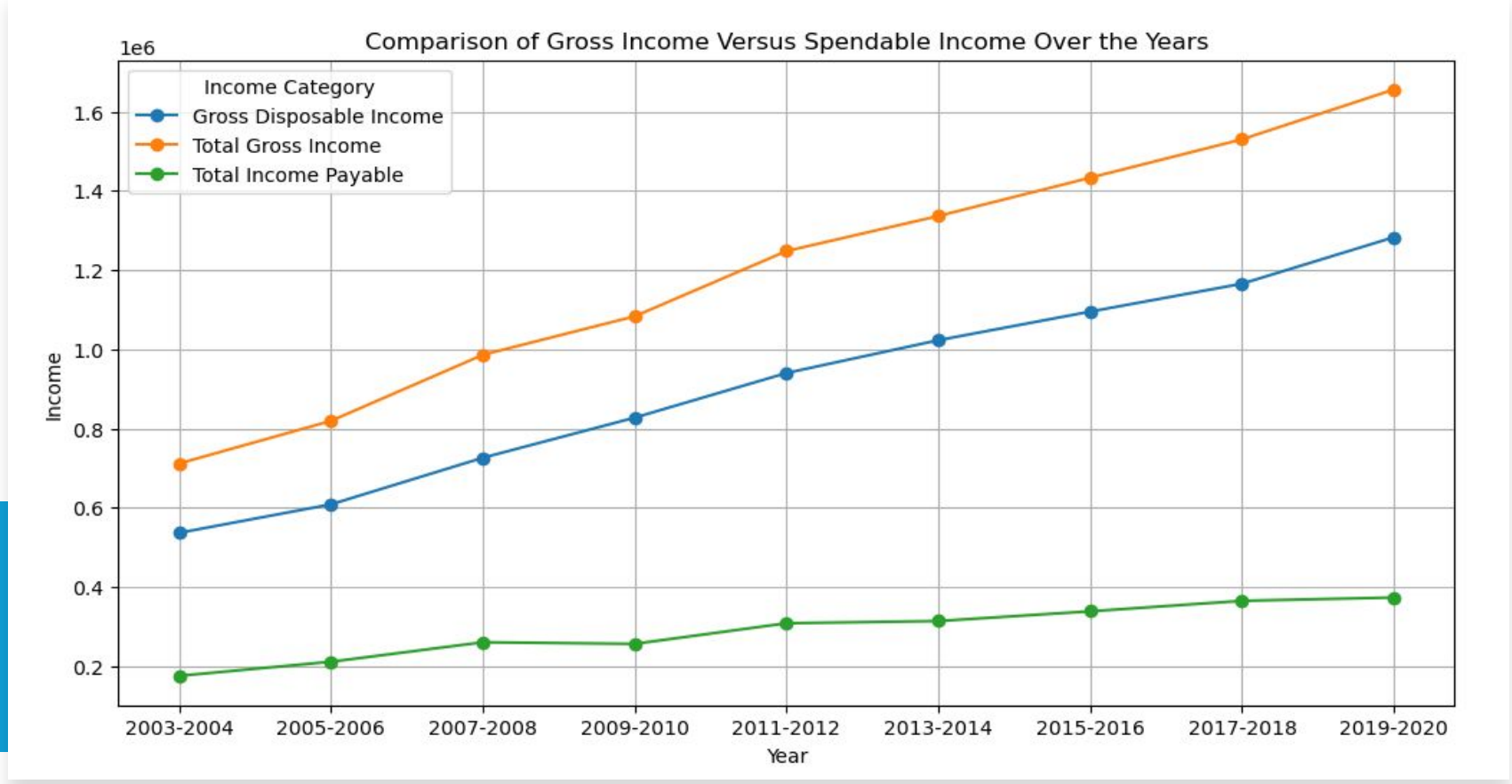
- Data for Gross income is negatively skewed
- Total Gross Income per two year period continued to increase and did not decrease at any point during the 17 year period of measure

```
# Extract the data for 'Total Gross income' from a specific row in the DataFrame
total_gross_income_data = household_income_df.loc['Total gross income']

# Set the X-axis values to be the years
years = total_gross_income_data.index

# Set the Y-axis values to be the 'Total Gross income' values
total_gross_income_values = total_gross_income_data.values

# Define a color map for each year
colors = plt.cm.viridis(np.linspace(0, 1, len(years)))

# Create a bar graph with different colors for each year
plt.bar(years, total_gross_income_values, color=colors)
plt.xlabel('Year')
plt.ylabel('Total Gross Income')
plt.title('Total Gross Income by Year')
plt.xticks(rotation=45)  # Rotate X-axis labels for better visibility
plt.show()
```

VISUALISING THE DATA

This line graph depicts the **Total Income Data** as sourced from the ABS Census

```
# Set 'Year/s' column as the index
income_df.set_index('Year/s', inplace=True)


# Plot the line graph
income_df.plot(kind='line', marker='o', figsize=(12, 6))
plt.xlabel('Year')
plt.ylabel('Income')
plt.title('Comparison of Gross Income Versus Spendable Income Over the Years')
plt.legend(title='Income Category')
plt.grid(True)
plt.show()
```



Observations:

- There is no major increases in any of the income data categories
- Total Income payable was the only category to see a decrease between the 4 year period of 2007/2008 and 2009//2010 before again increasing per two year period

type	2003-2004	2005-2006	2007-2008	2009-2010	2011-2012	2013-2014	2015-2016
House/Townhouse	167.571429	170.857143	187.571429	173.285714	146.714286	160.857143	136.285714
Residential Apartment	609.909091	572.454545	932.636364	1233.363636	1091.545455	1353.363636	686.909091

```
# Modifying house_price_df for 2 year ranges, to compare with Income Data
starting_years = list(range(2003,2015 + 1,2))

# empty list to store new dictionaries
median_price_two_year_dict = {}
sum_transactions_two_year_dict = {}

# loop through year ranges
for year in starting_years:
    st_year = house_price_df[house_price_df["sale_year"] == str(year)]
    en_year = house_price_df[house_price_df["sale_year"] == str(year + 1)]

    # get intersection
    two_year = pd.merge(st_year, en_year, how ='inner', on =['small_area', 'type'])

    # get rid of "Docklands House/Townhouse" and "Southbank House/Townhouse"
    two_year = two_year.drop(two_year[(two_year['small_area'] == 'Docklands') & (two_year['type'] == 'House/Townhouse')].
    two_year = two_year.drop(two_year[(two_year['small_area'] == 'Southbank') & (two_year['type'] == 'House/Townhouse')].
    two_year = two_year.reset_index(drop=True)

    # add together transaction counts, and average median prices
    average_median_2y = (two_year["median_price_x"] + two_year["median_price_y"])/2
    two_year["average_median_2y"] = average_median_2y
    sum_transactions_2y = two_year["transaction_count_x"] + two_year["transaction_count_y"]
    two_year["sum_transactions_2y"] = sum_transactions_2y

    # split into types
    two_year_groupby = two_year.groupby(["type"])
    two_year_ave_median = two_year_groupby["average_median_2y"].mean()
    two_year_sum_transactions = two_year_groupby["sum_transactions_2y"].mean()

    # put into dictionaries
    median_price_two_year_dict[f"%s-%s" % (year, year + 1)] = two_year_ave_median
    sum_transactions_two_year_dict[f"%s-%s" % (year, year + 1)] = two_year_sum_transactions
```

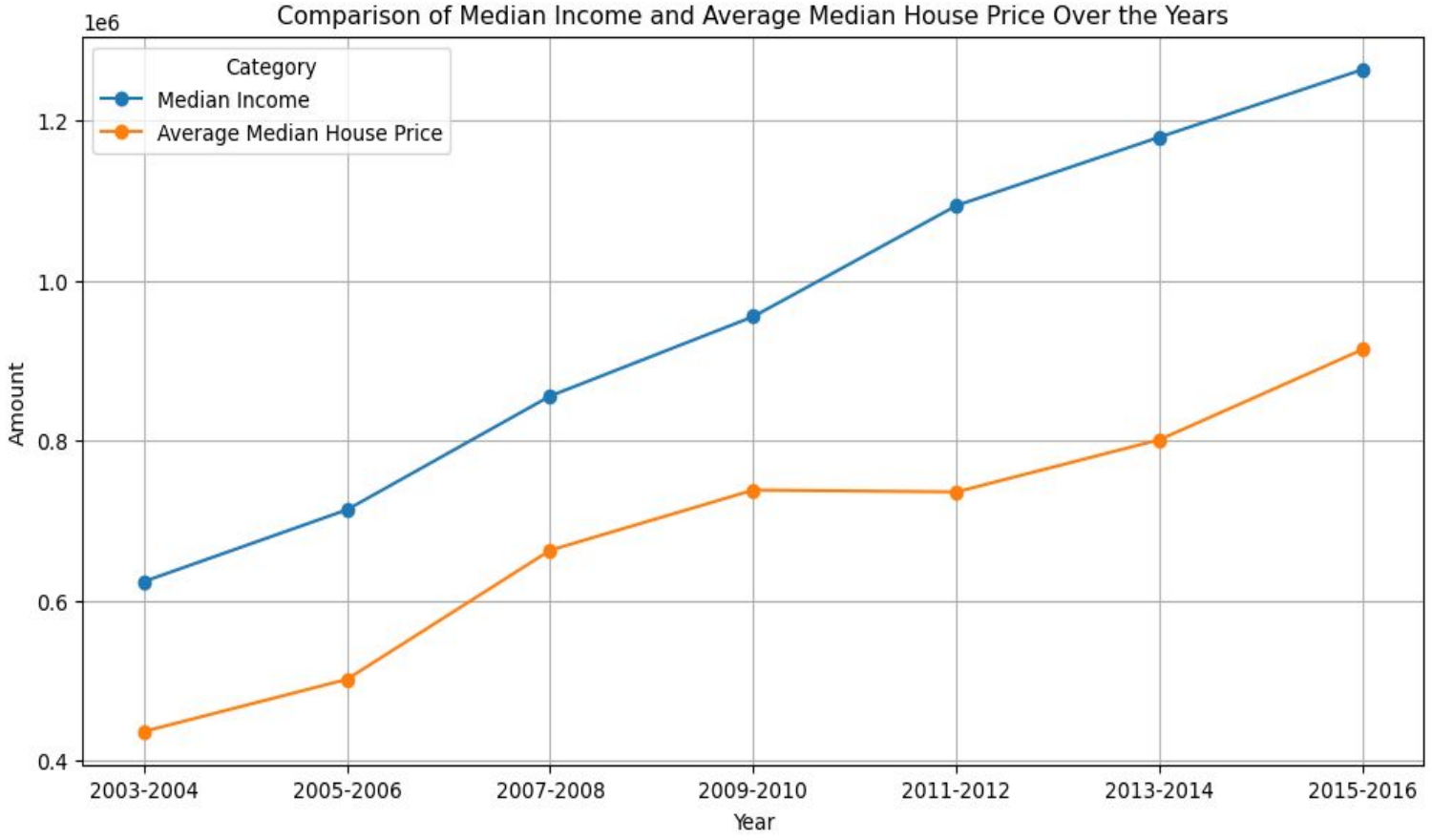
```
# median price DF
median_price_two_year_df = pd.DataFrame(median_price_two_year_dict)
median_price_two_year_df.head(17)
```

THE RESIDENCE PRICE DATA

As our API call from the City of Melbourne Open Data was available from the years 2000 to 2016, for the purpose of analysis the data was limited to show data from 2003-2016 in two year intervals to then compare with the Total Gross Income Data

COMPARING THE DATA

This bar graph depicts the **Median Household Income** in Victoria and the **Average Median House Price** in Melbourne Suburbs from 2003-2016




```
# Analyze trends in average median house prices and median income from 2003 to 2016
# Calculate the average median house price for every two years
average_median_price = median_price_two_year_df.mean()
average_median_price
```

```
2003-2004    436554.568182
2005-2006    501783.061688
2007-2008    663090.819805
2009-2010    738372.889610
2011-2012    736070.292208
2013-2014    801391.461039
2015-2016    914250.324675
dtype: float64
```

```
# Median income by year vs average median house price by year
# Create a DataFrame with the median income and average median house price for every tw
income_vs_price_df = pd.concat([median_income, average_median_price], axis=1)
income_vs_price_df = income_vs_price_df.drop(["2017-2018", "2019-2020"])
income_vs_price_df.columns = ["Median Income", "Average Median House Price"]
income_vs_price_df

# Plot the line graph
income_vs_price_df.plot(kind='line', marker='o', figsize=(12, 6))
plt.xlabel('Year')
plt.ylabel('Amount')
plt.title('Comparison of Median Income and Average Median House Price Over the Years')
plt.legend(title='Category')
plt.grid(True)
plt.show()
```

Observations:

- Median House prices overall have also more than doubled by *2.09 across the combined housing types; House/Townhouse and Residential apartments.
- Unlike Median Income which had maintained an increasing upwards trend per two year period , the Median Residence Prices saw a decrease between the 4 year period of 2009/2010 and 2011/2012 before again increasing per two year period

What trends can be observed in median income per 2 year period vs median residence cost per 2 year period?

It can be hypothesised that the Median House Price for the next two year period 2017-2018 will maintain this same average difference of *1.59 between Median House Prices and Median Household Income, whilst both continuing to increase per two year period.

Limitations:

- Data is that is not current and cannot provide comparable information from 2016 onwards meaning we cannot predict future trends
- Average household income was based on Victorian census data whereas the House Price data is based on Melbourne suburbs - meaning the data could be skewed based on proximity to Melbourne and may not provide a true indicator of all median house and income totals.

Household Affordability Index

What is the housing affordability index and what conclusions can be made based on various factors?

Interest Data of 2011 to 2016 used from RBA
https://www.rba.gov.au/statistics/cash-rate/

```
# file path for the xlsx file Interest rate data
interest_rate_data = Path("/Users/vrindapatel/Documents/GitHub/project-one-team2/Resources/Interest rate data.csv")

# Read the CSV file into a DataFrame
interest_rate_df = pd.read_csv(interest_rate_data)

# Display the first 5 rows of the DataFrame
interest_rate_df.head()
```

✓ 0.0s

Data Cleaning and Processing

```
# Get the available sheet names in the CSV file
interest_rate_data_df = pd.read_csv(interest_rate_data)
interest_rate_data_df.head()

# Drop the first eleven rows
interest_rate_data_df = interest_rate_data_df.drop([0,1,2,3,4,5,6,7,8,9,10])
interest_rate_data_df.head()

# Drop the column "Unnamed: 2" to "Unnamed: 18"
interest_rate_data_df = interest_rate_data_df.drop(interest_rate_data_df.iloc[:, 2:19], axis = 1)
interest_rate_data_df.head()

# Rename the columns
interest_rate_data_df.columns = ["Date", "Interest Rate"]
interest_rate_data_df.head()
```

```
if 'Date' in interest_rate_data_df.columns:
    interest_rate_data_df["Date"] = pd.to_datetime(interest_rate_data_df["Date"]).dt.to_period('Y')

# change date to year
interest_rate_data_df["Date"] = interest_rate_data_df["Date"].dt.year

# Change column name
interest_rate_data_df = interest_rate_data_df.rename(columns={"Date": "Year"})

# Set the 'Date' column as the index
interest_rate_data_df.set_index('Year', inplace=True)
interest_rate_data_df.head()
```

✓ 0.0s

	Interest Rate
Year	
2011	4.75
2011	4.75
2011	4.75
2011	4.75
2011	4.75

Grouping data by Year and Calculating Median Interest rate

```
interest_rate_data_df['Interest Rate'] = pd.to_numeric(interest_rate_data_df['Interest Rate'], errors='coerce')

# Assuming your DataFrame has a column 'Year' with the year of each interest rate
median_interest_rate = interest_rate_data_df.groupby('Year')['Interest Rate'].median()
median_interest_rate.head()

median_interest_rate = median_interest_rate[median_interest_rate.index != -1]
def group_years(year):
    if 2011 <= year <= 2012:
        return '2011-2012'
    elif 2013 <= year <= 2014:
        return '2013-2014'
    elif 2015 <= year:
        return '2015-2016'
    else:
        return 'Other'

median_interest_rate = median_interest_rate.groupby(group_years).mean()
median_interest_rate.head()
```

```
Year
2011-2012    4.125
2013-2014    2.625
2015-2016    1.740
Name: Interest Rate, dtype: float64
```

```
# create a dataframe for the median interest rate, average median house price, and median income
interest_vs_price_vs_income_df = pd.concat([median_interest_rate, average_median_price, median_income], axis=1)
interest_vs_price_vs_income_df.columns = ["Median Interest Rate", "Average Median House Price", "Median Income"]
interest_vs_price_vs_income_df
```

✓ 0.0s

	Median Interest Rate	Average Median House Price	Median Income
2011-2012	4.125	736070.292208	1094011.0
2013-2014	2.625	801391.461039	1179664.0
2015-2016	1.740	914250.324675	1264143.5
2003-2004	NaN	436554.568182	623946.5
2005-2006	NaN	501783.061688	713716.5
2007-2008	NaN	663090.819805	855916.0
2009-2010	NaN	738372.889610	955252.0
2017-2018	NaN	NaN	1347815.0
2019-2020	NaN	NaN	1469491.5

```
# drop the Nan values
interest_vs_price_vs_income_df = interest_vs_price_vs_income_df.dropna()
interest_vs_price_vs_income_df
```

✓ 0.0s

	Median Interest Rate	Average Median House Price	Median Income
2011-2012	4.125	736070.292208	1094011.0
2013-2014	2.625	801391.461039	1179664.0
2015-2016	1.740	914250.324675	1264143.5

Merged the Median Interest rate data to Average Median House Price and Median Income

Affordability Index Calculation

```
# create affordibility index
interest_vs_price_vs_income_df.loc[:, "Affordability Index"] = interest_vs_price_vs_income_df["Average Median House Price"] / interest_vs_price_vs_income_df["Median Income"] * 10
interest_vs_price_vs_income_df
```

✓ 0.0s Python

```
/var/folders/kr/dnn4yz45639_qpcw2723y4t80000gn/T/ipykernel_40513/1425799588.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  interest_vs_price_vs_income_df.loc[:, "Affordability Index"] = interest_vs_price_vs_income_df["Average Median House Price"] / interest_vs_price_vs_income_df["Median Income"] * 100
```

	Median Interest Rate	Average Median House Price	Median Income	Affordability Index
2011-2012	4.125	736070.292208	1094011.0	67.281800
2013-2014	2.625	801391.461039	1179664.0	67.933874
2015-2016	1.740	914250.324675	1264143.5	72.321720

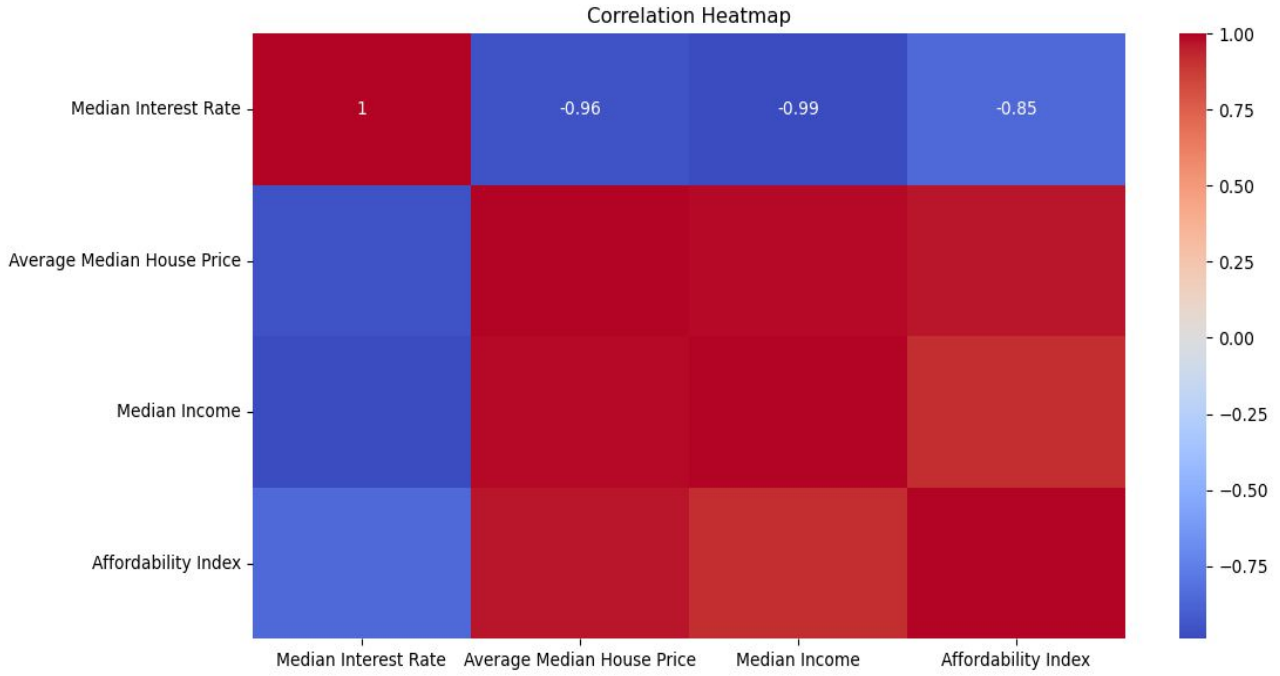
```
# plot heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(interest_vs_price_vs_income_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

✓ 0.3s Python

Affordability Index Heatmap

Conclusion:

- **Median Interest Rate vs. Average Median House Price:** There's a strong negative correlation (dark blue) of approximately -0.96. As interest rates increase, house prices tend to decrease (and vice versa).
- **Affordability Index vs. Average Median House Price:** Another strong negative correlation (dark blue) of approximately -0.99. As house prices rise, affordability decreases significantly.
- **Affordability Index vs. Median Income:** A negative correlation (light blue) of approximately -0.85. As median income increases, affordability improves, but not as dramatically as house prices decrease affordability.




Residence Prices Against
Population
By Suburbs

Does the suburb population have any correlation with residence prices?

Jupyter Lab Code

2021 file load

```
# 2021 Files to Load
population_data_to_load_2021 = ("Resources/2021Census_G03_VIC_SAL.csv")
suburb_name_data_to_load_2021 = ("Resources/2021Census_geog_desc_1st_2nd_3rd_release.xlsx")
# Reference: https://www.abs.gov.au/census/find-census-data/datapacks?release=2021&product=GCP&geography=ALL&header=S
```

```
# Read Population data into Pandas DataFrames
population_data_2021_df = pd.read_csv(population_data_to_load_2021)

# Read population and Suburb data into Pandas DataFrames and specifiy which worksheet number to be read in suburb file
sheet_name = 0
suburb_name_data_2021_df = pd.read_excel(suburb_name_data_to_load_2021, sheet_name = 5)
# Reference:https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html
```

```
# Rename column in Population data for allow DataFrames to merge
population_data_2021_df= population_data_2021_df.rename(columns={"SAL_CODE_2021":"Census_Code_2021"})
#population_data_2021_df.head()
```

```
# Merge Population Data and Suburb Data DataFrames
population_and_suburb_merged_2021_df = pd.merge(population_data_2021_df, suburb_name_data_2021_df, how = "left", on=["Census_Code_2021", "Census_Code_2021"])
```

```
# Set Index to AGSS_Code_2021 and rename columns to meaningful headings
population_and_suburb_data_2021_df = population_and_suburb_merged_2021_df.set_index(["AGSS_Code_2021"])
population_and_suburb_data_2021_df = population_and_suburb_data_2021_df.rename(columns={"Total_Total":"Total_Population","Census_Name_2021":"Suburb_Name"})
```

```
# Create final Dataframe with only the required columns
population_and_suburb_2021_df = population_and_suburb_data_2021_df[["Suburb_Name","Total_Population","Area sqkm"]]
population_and_suburb_2021_final_df = pd.DataFrame(population_and_suburb_2021_df)
#population_and_suburb_2021_final_df.head()
```

```
# Create list of required suburbs
City_of_Melbourne_Suburbs_2021 = population_and_suburb_2021_final_df[(population_and_suburb_2021_final_df["Suburb_Name"]=="East Melbourne")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Parkville (Vic.)")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="North Melbourne")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="South Yarra")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Southbank")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Docklands")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Carlton (Vic.)")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="West Melbourne")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Kensington (Vic.)")|
(population_and_suburb_2021_final_df["Suburb_Name"]=="Melbourne")]
```

Jupyter Lab Code

```
# Create DataFrame for City of Melbourne suburbs
City_of_Melbourne_Suburbs_2021_df = pd.DataFrame(City_of_Melbourne_Suburbs_2021)
#City_of_Melbourne_Suburbs_2021_df
```

```
# Add Census Year to the DataFrame:
year = ["2021","2021","2021","2021","2021","2021","2021","2021","2021","2021"]
City_of_Melbourne_Suburbs_2021_df["Census_Year"] = year
City_of_Melbourne_Suburbs_2021_df
```

AGSS_Code_2021	Suburb_Name	Total_Population	Area sqkm	Census_Year
20495	Carlton (Vic.)	15896	1.7728	2021
20766	Docklands	15547	3.1480	2021
20830	East Melbourne	5358	1.8557	2021
21327	Kensington (Vic.)	10545	2.1373	2021
21640	Melbourne	58262	6.5866	2021
21966	North Melbourne	15040	2.3554	2021
22038	Parkville (Vic.)	7756	3.9981	2021
22314	South Yarra	24609	3.5475	2021
22315	Southbank	22717	1.5640	2021
22757	West Melbourne	7934	6.5822	2021

Jupyter Lab Code

Population Visualisation

```
# Merge Population Data and Suburb Data DataFrames for Census Years 2011, 2016 and 2021
merged_population_df = pd.merge(City_of_Melbourne_Suburbs_2011_df, City_of_Melbourne_Suburbs_2016_df, how = "left", on=["Suburb_Name", "Suburb_Name"])
```

```
merged_population_df = pd.merge(merged_population_df, City_of_Melbourne_Suburbs_2021_df, how = "left", on=["Suburb_Name", "Suburb_Name"])
```

```
# Rename columns for charts
merged_population_renamed_df = merged_population_df.rename(columns=
                                                           {"Total_Population_x":"Total_Population_2011",
                                                            "Total_Population_y":"Total_Population_2016",
                                                            "Total_Population":"Total_Population_2021",
                                                            "Census_Year_x":"Census_Year_2011",
                                                            "Census_Year_y":"Census_Year_2016",
                                                            "Census_Year": "Census_Year_2021",
                                                            "Area sqkm_x": "Area_sqkm_2011",
                                                            "Area sqkm_y": "Area_sqkm_2016",
                                                            "Area sqkm":"Area_sqkm_2021"})
```

```
merged_population_renamed_df
```

	Suburb_Name	Total_Population_2011	Area_sqkm_2011	Census_Year_2011	Total_Population_2016	Area_sqkm_2016	Census_Year_2016	Total_Population_2021	Area_sqkm_2021
0	Carlton (Vic.)	13742	1.69631	2011	18829	1.7488	2016	15896	1.7728
1	Docklands	6756	3.00192	2011	12486	3.1479	2016	15547	3.1480
2	East Melbourne	6073	1.39918	2011	6196	1.8557	2016	5358	1.8557
3	Kensington (Vic.)	9616	2.14699	2011	10621	2.1306	2016	10545	2.1373
4	Melbourne	38560	6.23214	2011	60057	6.5045	2016	58262	6.5866
5	North Melbourne	12134	2.38918	2011	15395	2.3619	2016	15040	2.3554
6	Parkville (Vic.)	7895	4.04870	2011	9188	4.0492	2016	7756	3.9981
7	South Yarra	19670	3.64949	2011	25237	3.5476	2016	24609	3.5475
8	Southbank	13832	1.26233	2011	22397	1.5639	2016	22717	1.5640
9	West Melbourne	3851	6.59269	2011	5588	6.5821	2016	7934	6.5822

Jupyter Lab Code

```
population_only_df = pd.DataFrame(population_reduced_df[["Suburb_Name",
                                                         "Total_Population_2011",
                                                         "Total_Population_2016",
                                                         "Total_Population_2021"]])
```

```
population_only_df
```

	Suburb_Name	Total_Population_2011	Total_Population_2016	Total_Population_2021
0	Carlton (Vic.)	13742	18829	15896
1	Docklands	6756	12486	15547
2	East Melbourne	6073	6196	5358
3	Kensington (Vic.)	9616	10621	10545
4	Melbourne	38560	60057	58262
5	North Melbourne	12134	15395	15040
6	Parkville (Vic.)	7895	9188	7756
7	South Yarra	19670	25237	24609
8	Southbank	13832	22397	22717
9	West Melbourne	3851	5588	7934

```
# Set index to "Suburb_Name"
population_only_df = population_only_df.set_index("Suburb_Name")
```

```
# Create variables for suburbs included in graph
suburb1 = "Carlton (Vic.)"
suburb2 = "Docklands"
suburb3 = "East Melbourne"
suburb4 = "Kensington (Vic.)"
suburb5 = "Melbourne"
suburb6 = "North Melbourne"
suburb7 = "Parkville (Vic.)"
suburb8 = "South Yarra"
suburb9 = "Southbank"
suburb10 = "West Melbourne"
```

```
# Set variable to look at Total_Population
population_comparison = "Total_Population_"
```

```
# Create a Series for comparing population from 2011 to 2021
suburb1_total_population = population_only_df.loc[suburb1,[f"{population_comparison}2011",
                                                           f"{population_comparison}2016",
                                                           f"{population_comparison}2021"]]
```

```
suburb10_total_population = population_only_df.loc[suburb10,[f"{population_comparison}2011",
                                                             f"{population_comparison}2016",
                                                             f"{population_comparison}2021"]]
```

Merging population and median house price data

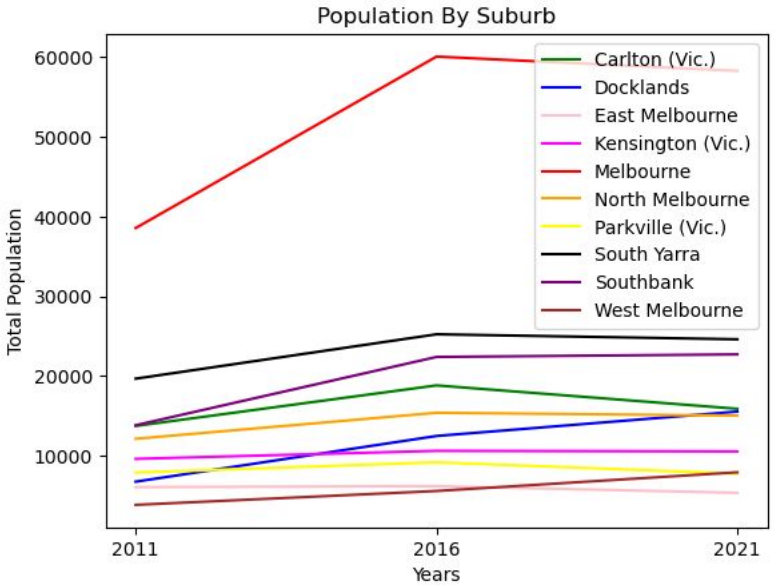
From median house price data which Violet accessed via API

	sale_year	small_area	type	median_price	transaction_count
0	2016	East Melbourne	House/Townhouse	1940000.0	35
1	2016	East Melbourne	Residential Apartment	650000.0	129
2	2016	Melbourne (CBD)	House/Townhouse	NaN	2
3	2016	Melbourne (CBD)	Residential Apartment	435000.0	1364
4	2016	North Melbourne	Residential Apartment	515000.0	231
5	2016	Parkville	House/Townhouse	1781250.0	32
6	2016	Parkville	Residential Apartment	500000.0	63
7	2016	South Yarra	House/Townhouse	2125000.0	34
8	2016	Southbank	Residential Apartment	565000.0	701
9	2016	Carlton	Residential Apartment	508000.0	162
10	2016	Docklands	House/Townhouse	NaN	5
11	2016	North Melbourne	House/Townhouse	861000.0	87
12	2016	South Yarra	Residential Apartment	555000.0	86
13	2016	West Melbourne (Residential)	Residential Apartment	475000.0	115
14	2016	NA	Residential Apartment	NaN	1
15	2016	Carlton	House/Townhouse	1110000.0	57
16	2016	Docklands	Residential Apartment	591000.0	371
17	2016	Kensington	House/Townhouse	816750.0	172
18	2016	Kensington	Residential Apartment	430000.0	126
19	2016	Melbourne (Remainder)	Residential Apartment	582500.0	80
20	2016	West Melbourne (Residential)	House/Townhouse	900000.0	26
21	2016	NA	House/Townhouse	NaN	4

Suburb_Name	Total_Population_2011	Total_Population_2016	Total_Population_2021	2016_Median_Residential_Apt_Price	2016_Median_Hou
Carlton (Vic.)	12742	18829	15896	508000	
Docklands	8756	12466	15547	591000	
East Melbourne	6073	6196	5358	650000	
Kensington (Vic.)	9616	10621	10545	430000	
Melbourne	36560	60057	56262	435000	
North Melbourne	12154	13395	15040	515000	
Parkville (Vic.)	7995	9168	7756	500000	
South Yarra	19670	25237	24609	555000	
Southbank	13832	22307	22717	565000	
West Melbourne	3851	5568	7934	475000	

Jupyter Lab Code

```
# Years for the x axis
years = [2011, 2016, 2021]

# Plot lines for suburbs by poulation
plt.plot(years,suburb1_total_population, color="green", label=suburb1)
plt.plot(years,suburb2_total_population, color="blue", label=suburb2)
plt.plot(years,suburb3_total_population, color="pink", label=suburb3)
plt.plot(years,suburb4_total_population, color="magenta", label=suburb4)
plt.plot(years,suburb5_total_population, color="red", label=suburb5)
plt.plot(years,suburb6_total_population, color="orange", label=suburb6)
plt.plot(years,suburb7_total_population, color="yellow", label=suburb7)
plt.plot(years,suburb8_total_population, color="black", label=suburb8)
plt.plot(years,suburb9_total_population, color="purple", label=suburb9)
plt.plot(years,suburb10_total_population, color="Brown", label=suburb10)
# Plot legend in the best place as determined by matplotlib
plt.legend(loc="best")
plt.title("Population By Suburb")
plt.xlabel("Years")
plt.xticks(np.arange(min(years), max(years)+1,5.0))
plt.ylabel("Total Population")
plt.show()
```



Merging population and median house price data

Jupyter Lab Code

From median house price data which Violet accessed via API

[10]:

	sale_year	small_area	type	median_price	transaction_count
0	2016	East Melbourne	House/Townhouse	1940000.0	35
1	2016	East Melbourne	Residential Apartment	650000.0	129
2	2016	Melbourne (CBD)	House/Townhouse	NaN	2
3	2016	Melbourne (CBD)	Residential Apartment	435000.0	1264
4	2016	North Melbourne	Residential Apartment	515000.0	231
5	2016	Parkville	House/Townhouse	1761250.0	32
6	2016	Parkville	Residential Apartment	500000.0	63
7	2016	South Yarra	House/Townhouse	2135000.0	34
8	2016	Southbank	Residential Apartment	565000.0	701
9	2016	Carlton	Residential Apartment	508000.0	162
10	2016	Docklands	House/Townhouse	NaN	5
11	2016	North Melbourne	House/Townhouse	861000.0	87
12	2016	South Yarra	Residential Apartment	555000.0	88
13	2016	West Melbourne (Residential)	Residential Apartment	475000.0	115
14	2016	NA	Residential Apartment	NaN	1
15	2016	Carlton	House/Townhouse	1110000.0	57
16	2016	Docklands	Residential Apartment	591000.0	371
17	2016	Kensington	House/Townhouse	818750.0	172
18	2016	Kensington	Residential Apartment	430000.0	128
19	2016	Melbourne (Remainder)	Residential Apartment	582500.0	80
20	2016	West Melbourne (Residential)	House/Townhouse	900000.0	26
21	2016	NA	House/Townhouse	NaN	4

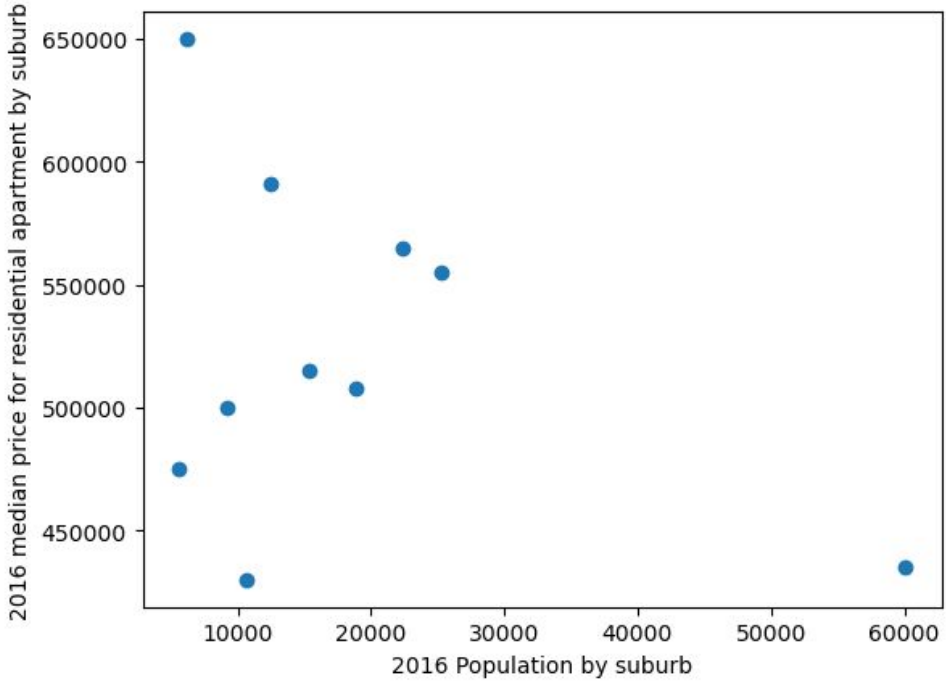
Suburb_Name	Total_Population_2011	Total_Population_2016	Total_Population_2021	2016_Median_Residential_Apt_Price	2016_Median_Hou
Carlton (Vic.)	13742	18829	15896	508000	
Docklands	6756	12486	15547	591000	
East Melbourne	6073	6196	5358	650000	
Kensington (Vic.)	9616	10621	10545	430000	
Melbourne	38560	60057	58262	435000	
North Melbourne	12134	15395	15040	515000	
Parkville (Vic.)	7895	9188	7756	500000	
South Yarra	19670	25237	24609	555000	
Southbank	13832	22397	22717	565000	
West Melbourne	3851	5588	7934	475000	

Correlation between population and median house prices

No correlation between population and residential apartments

```
# Scatter plot for population and median price for residential apartment with correlation calculation
population = population_copy_for_pricing_df.iloc[:,1]
median_res_apt_price= population_copy_for_pricing_df.iloc[:,3]
correlation = st.pearsonr(population,median_res_apt_price)
plt.scatter(population,median_res_apt_price)
plt.xlabel('2016 Population by suburb')
plt.ylabel('2016 median price for residential apartment by suburb')
print(f"The correlation between both factors is {round(correlation[0],2)}")
plt.show()
```

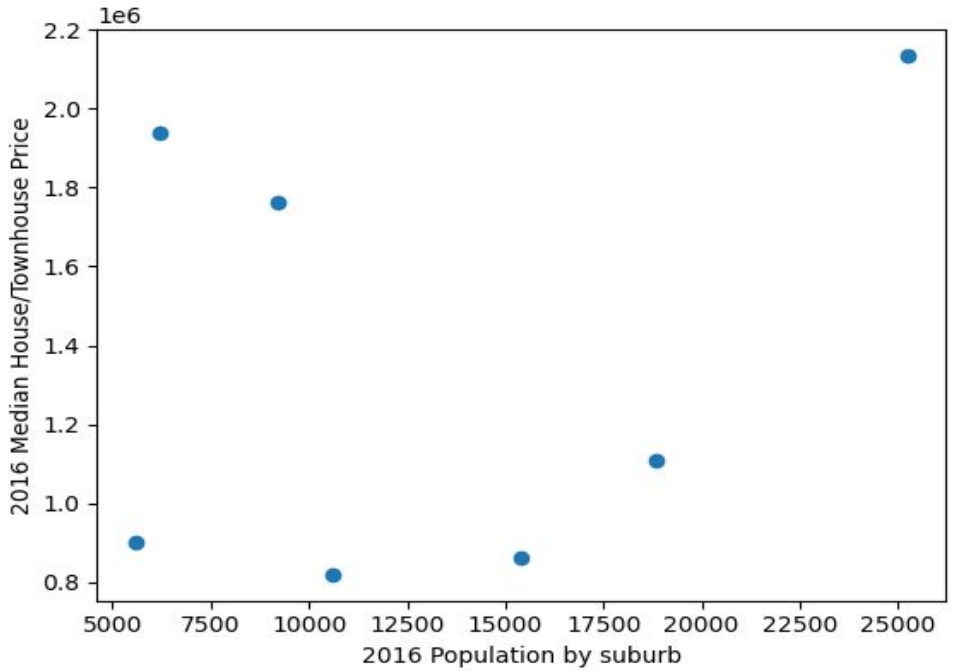
```
The correlation between both factors is -0.37
```



Slight correlation between population and houses & townhouses

```
# Scatter plot for population and median price for house/townhouse with correlation calculation
population = population_copy_for_house_pricing_df.iloc[:,1]
median_hse_town_price= population_copy_for_house_pricing_df.iloc[:,4]
correlation = st.pearsonr(population,median_hse_town_price)
plt.scatter(population,median_hse_town_price)
plt.xlabel('2016 Population by suburb')
plt.ylabel('2016 Median House/Townhouse Price')
print(f"The correlation between both factors is {round(correlation[0],2)}")
plt.show()
```

```
The correlation between both factors is 0.25
```




Residence Price Increases
Against
Population Growth
By Suburbs

Does suburb population growth have any correlation with house prices?

Jupyter Lab Code

```
# calculate population growth rate for each suburb from 2011 to 2016
population_growth_rate_df = pd.merge(population_and_suburb_2011_final_df, population_and_suburb_2016_final_df, on='Suburb_Name', how='inner')
population_growth_rate_df["Population Growth Rate"] = ((population_growth_rate_df["Total_Population_y"] - population_growth_rate_df["Total_Population_x"]) / \
                                                       population_growth_rate_df["Total_Population_x"]) * 100
population_growth_rate_df = population_growth_rate_df[["Suburb_Name", "Population Growth Rate"]]
population_growth_rate_df.head()
```

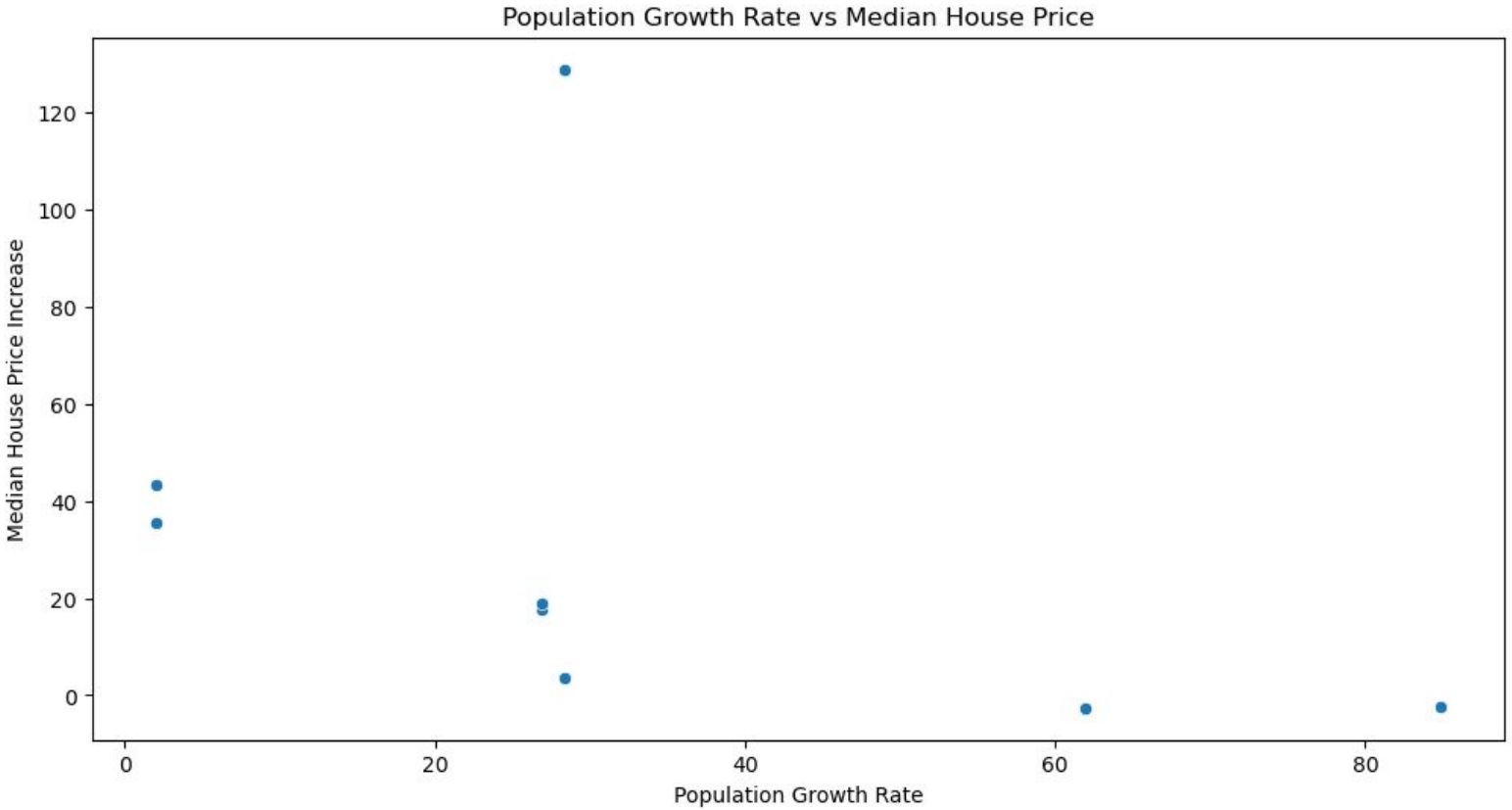
	Suburb_Name	Population Growth Rate
0	Abbotsford (Vic.)	68.725224
1	Aberfeldie	9.637188
2	Acheron	-46.037736
3	Aireys Inlet	16.716418
4	Airport West	7.906169

```
# calculate residential price increase from 2011 to 2016
house_price_2016 = house_price_df[house_price_df["sale_year"]==2016]
house_price_2011 = house_price_df[house_price_df["sale_year"]==2011]
house_price_span = pd.merge(house_price_2016,house_price_2011,on=["small_area","type"],how="inner")

res_price_increase = (house_price_span["median_price_x"] - house_price_span["median_price_y"]) / house_price_span["median_price_y"] * 100
house_price_span["res_price_increase"] = res_price_increase
house_price_span.head()
```

	sale_year_x	small_area	type	median_price_x	transaction_count_x	sale_year_y	median_price_y	transaction_count_y	res_price_increase
0	2016	East Melbourne	House/Townhouse	1940000.0	35	2011	1352500.0	30	43.438078
1	2016	East Melbourne	Residential Apartment	650000.0	129	2011	480000.0	139	35.416667
2	2016	Melbourne (CBD)	Residential Apartment	435000.0	1264	2011	464000.0	2437	-6.250000
3	2016	North Melbourne	Residential Apartment	515000.0	231	2011	438000.0	208	17.579909
4	2016	Parkville	House/Townhouse	1761250.0	32	2011	975000.0	34	80.641026

Correlation between population growth rate and median house prices

No correlation between population growth rate and median house price



	Population Growth Rate	res_price_increase
Population Growth Rate	1.000000	-0.397954
res_price_increase	-0.397954	1.000000

Jupyter Lab Code

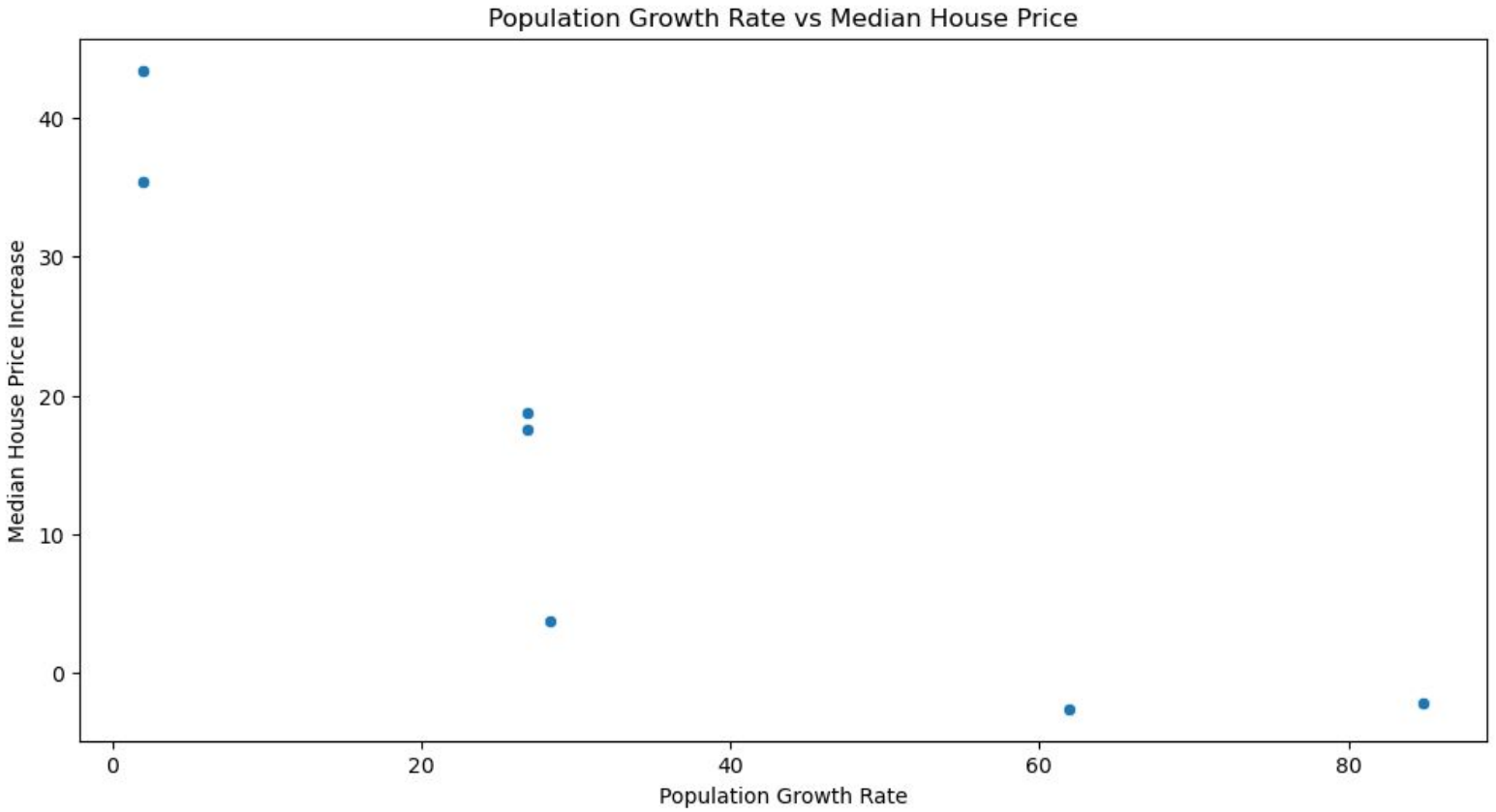
```
# get rid of outlier
house_price_pop_growth_sorted = house_price_pop_growth.sort_values(by="res_price_increase", ascending=False).reset_index()
house_price_pop_growth_sorted.head()
```

	index	sale_year_x	small_area	type	median_price_x	transaction_count_x	sale_year_y	median_price_y	transaction_count_y	res_price_increase	Population Growth Rate
0	4	2016	South Yarra	House/Townhouse	2135000.0	34	2011	932500.0	32	128.954424	28.301983
1	0	2016	East Melbourne	House/Townhouse	1940000.0	35	2011	1352500.0	30	43.438078	2.025358
2	1	2016	East Melbourne	Residential Apartment	650000.0	129	2011	480000.0	139	35.416667	2.025358
3	3	2016	North Melbourne	House/Townhouse	861000.0	87	2011	724750.0	110	18.799586	26.874897
4	2	2016	North Melbourne	Residential Apartment	515000.0	231	2011	438000.0	208	17.579909	26.874897

```
# drop largest increase
house_price_pop_growth_sorted = house_price_pop_growth_sorted.drop([0])
house_price_pop_growth_sorted.head()
```

Correlation between population growth rate and median house prices

No correlation between population growth rate and median house price



	Population Growth Rate	res_price_increase
Population Growth Rate	1.000000	-0.879557
res_price_increase	-0.879557	1.000000

Thanks for listening.