REPORT WRITING

ON

A CENSUS PROJECT

Artificial Intelligence and Data sciences

Table of Content

Introduction

The UK census is not just a simple counting exercise. It is a comprehensive and meticulous national effort that collects essential information, such as street names, surnames, occupations, and marital statuses. The data goes beyond mere numbers, delving into the intricacies of households and individual lives. This thorough process guarantees a detailed and precise depiction of the population's dynamics, going beyond the surface to capture the diverse socio-economic makeup. With a focus on critical factors like residential details, family structures, and professional roles, the census becomes a valuable resource for policymakers, researchers, and communities alike, offering valuable insights into societal trends, resource distribution, and national planning for the future.

Aim

The aim of this project to perform a census exercise That compares different people across the nation and to provide the government with accurate statistics of the population to enable better planning, to develop policies, and to allocate certain funding.

Tool/ Skills needed to achieve this goal

- Python programming language
- Pandas
- NumPy
- Matplotlib
- Seaborn

Python Programming language: This is an object-oriented programming language that can be used to send instructions to the computer to perform certain tasks. Python is especially useful in both web development and data analysis. The power of Python in data analysis is evident in its extensive libraries, such as pandas and NumPy. Its ease of use and flexibility make it the top

choice for data manipulation and exploration tasks. With this language, extracting valuable insights for informed decision-making becomes a smooth and efficient process.

Pandas: This is a python library which task is to manipulate, wrangle, adjust data for better exploration of data. Python's data manipulation library is renowned for its ability to handle and transform data with its robust toolkit. Its user-friendly features allow for seamless manipulation and investigation of data, solidifying its necessity in tasks involving data analysis and exploration.

NumPy: This is also a python library used to perform numerical calculations in other tasks. a crucial component of the Python library, streamlines numerical tasks and array manipulation. By improving computation speed and offering a wide range of mathematical capabilities, it serves as the foundation for numerous scientific and data analysis programs. With its advanced array objects, NumPy boosts the efficiency of mathematical operations, playing a crucial role in bolstering Python's reputation for robust scientific computing and data analysis.

Matplotlib: This is a data visualization python library, which can be used to create interactive charts across analysis in the system. With Matplotlib, a robust Python library, you can produce stunning visual representations. It offers a wide range of options for everything from basic line graphs to intricate 3D plots, giving you the freedom to effectively communicate data insights. This indispensable tool is crucial for any scientific or data analysis project.

Seaborn: This is also a python library built on the matplotlib library, it is a data visualization library that can be used to create interactive charts that will provide insight based on the data is fed upon. Seaborn, which is built upon the foundation of Matplotlib, elevates data visualization with its sophisticated user interface. It streamlines the process of creating visually compelling and informative statistical graphs. With its advanced design, Seaborn is an invaluable tool for effectively representing intricate datasets and patterns.

DATA STEPS

- Data Acquisition
- Data Wrangling/Data Cleaning
- Exploratory Data Analysis (EDA)
- Recommendations

DATA ACQUISITION:

This is the process of acquiring data into the system by using the data sources. There are various sources that can be used to get data into the system which includes the csv, SQL, Json and many more. Data acquisition entails accumulating data from various sources, including CSV files, SQL databases, and JSON streams. This crucial stage sets the groundwork for further analysis and enables a thorough comprehension of the data. It employs effective techniques to seamlessly import, structure, and combine different datasets, enabling businesses to unlock valuable insights for informed decision-making and strategic planning.

Data Acquisition

This is the act of acquiring the data into the environment.

```
In [367]: #creating a variable that brings the data to the environment which is census_data
          census_data = pd.read_csv('census04.csv')
          census_data.head(10)
```

Out[367]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
0	1	Jones Crossing	Victor	Richardson	61	Head	Single	Male	Lawyer	NaN	Christian
1	1	Jones Crossing	Duncan	Clark	62	NaN	Single	Male	Energy engineer	NaN	Christian
2	1	Jones Crossing	Sian	Page	21	NaN	Single	Female	University Student	NaN	Methodist
3	1	Jones Crossing	Kevin	Jones	44	NaN	Single	Male	Set designer	NaN	Methodist
4	1	Jones Crossing	Derek	Stephenson	28	NaN	Single	Male	Psychotherapist	NaN	Muslim
5	2	Jones Crossing	Wendy	Stevens	49	Head	Married	Female	Administrator, arts	NaN	Christian
6	2	Jones Crossing	Benjamin	Stevens	51	Husband	Married	Male	Engineer, production	NaN	Christian
7	2	Jones Crossing	Adam	Stevens	19	Son	Divorced	Male	University Student	NaN	Christian
8	2	Jones Crossing	Leon	Stevens	17	Son	NaN	Male	Student	NaN	NaN
9	3	Jones Crossing	Megan	Wilkinson	36	Head	Married	Female	Tour manager	NaN	Muslim

Fig1: shows that data is acquired using the panda's library from the systems directory.

The data acquired is nothing more than the census data the includes a wide range of information which starts from the house number, street, first and surname, age, relationship to the head, marital status, gender, occupancy, infirmity, religion.

Data Wrangling/Data Cleaning:

Data cleaning is a process of removal of irrelevancies or unformatted words or character away from the main data. This data can be words like an empty string, Na values, inappropriate usage of words at a position in brackets. Data cleaning is the meticulous and essential process of rectifying inaccuracies, handling missing values, and resolving inconsistencies within datasets. Through addressing outliers and standardizing formats, this crucial step guarantees the accuracy and integrity of data. By effectively

cleaning data, the reliability of analyses is significantly improved, resulting in trustworthy insights for informed decision-making and robust statistical modeling across various fields.

To wrangle / clean your data you have explore the data to know what is in the data.

Data Wrangling

Familarizing with the data which involves shape, size , column information , number of Nan values e.t.c

```
In [368]: #checking the data types
          census_data.dtypes

Out[368]: House Number                    int64
          Street                         object
          First Name                     object
          Surname                        object
          Age                            object
          Relationship to Head of House  object
          Marital Status                 object
          Gender                         object
          Occupation                     object
          Infirmity                      object
          Religion                       object
          dtype: object
```

Fig2: shows the data type of each column in the dataframe

```
In [369]: #getting to know the columns
          census_data.columns

Out[369]: Index(['House Number', 'Street', 'First Name', 'Surname', 'Age',
                 'Relationship to Head of House', 'Marital Status', 'Gender',
                 'Occupation', 'Infirmity', 'Religion'],
                dtype='object')
```

Fig3: shows the column names in the dataframe

```
In [370]: #getting to know the rows and columns available in this dataset
          census_data.shape

Out[370]: (8237, 11)
```

Fig4: shows the shape of the data

```
          census_data.isnull().sum()

Out[372]: House Number                    0
          Street                         0
          First Name                     0
          Surname                        0
          Age                            0
          Relationship to Head of House    627
          Marital Status                1989
          Gender                           0
          Occupation                       0
          Infirmity                     8163
          Religion                      4865
          dtype: int64
```

It shows that the Marital Status and Religion has huge number of Nan Values

Fig5: shows the amount of nan values available on each column.

```
In [373]: #getting more information
          census_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8237 entries, 0 to 8236
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   House Number                    8237 non-null   int64 
 1   Street                          8237 non-null   object
 2   First Name                      8237 non-null   object
 3   Surname                         8237 non-null   object
 4   Age                             8237 non-null   object
 5   Relationship to Head of House   7610 non-null   object
 6   Marital Status                  6248 non-null   object
 7   Gender                          8237 non-null   object
 8   Occupation                      8237 non-null   object
 9   Infirmity                       74 non-null     object
 10  Religion                        3372 non-null   object
dtypes: int64(1), object(10)
memory usage: 708.0+ KB
```

Fig6: show the general data information that includes the nan values and data type.

Creating a reusable functions to avoid repeats and performance to get the unique data each series contains.

```
In [374]: #Digging Dip into each feature in the dataset
          def CheckUnique(sub_data):
              data = census_data[sub_data].unique() # check for uniqueness in the columns


              return data
```

Fig7: creating a reuseable function to check for unique values for each column, specifying column name as the parameter.

Checking for uniqueness for all Series

```
In [375]: CheckUnique('House Number')
```

```
Out[375]: array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
                  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
                  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
                  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
                  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
                  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
                  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
                  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
                 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
                 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
                 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
                 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
                 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167])
```

Fig8: checking for the house number unique values

```
In [379]: CheckUnique('Age')
```

```
Out[379]: array(['61', '62', '21', '44', '28', '49', '51', '19', '17', '36', '38',
                 '6', '2', '43', '41', '15', '42', '5', '3', '73', '40', '13', '11',
                 '7', '64', '39', '34', '48', '45', '14', '12', '50', '9', '58',
                 '16', '10', '52', '54', '22', '63', '35', '26', '30', '33', '1',
                 '25', '18', '46', '8', '37', '4', '29', '47', '53', '31', '24',
                 '0', '55', '27', '69', '59', '66', '68', '20', '81', '32', '60',
                 '23', '57', '56', '75', '65', '71', '67', '83', '76', '77', '84',
                 '80', '85', '87', '70', '72', '79', '78', '89', '91', '88', '92',
                 '74', '82', '104', '98', '90', '86', '97', '94', '96', ' ', '93',
                 '49.0300851496455', '4.0', '3.0', '99', '57.90370851747044',
                 '56.90370851747044', '10.0', '7.0', '59.57715584216195', '17.0',
                 '15.0', '103', '102', '95', '106', '100', '101', '105', '107',
                 '62.77766726084468'], dtype=object)
```

Fig9: checking for the age unique values

```
In [380]: CheckUnique('Relationship to Head of House')

Out[380]: array(['Head', nan, 'Husband', 'Son', 'Daughter', 'Wife', 'Grandson',
                 'Granddaughter', 'Nephew', 'Partner', 'Lodger', 'Sibling',
                 'Adopted Son', 'Adopted Granddaughter', 'Step-Daughter',
                 'Step-Son', 'Visitor', 'Cousin', 'Adopted Daughter', 'Neice', ' ',
                 'Daughter-in-law'], dtype=object)
```

Fig10: checking for the Relationship to the head of house unique values

```
In [381]: CheckUnique('Marital Status')

Out[381]: array(['Single', 'Married', 'Divorced', nan, 'Widowed'], dtype=object)
```

Fig11: checking for the marital status unique values

```
In [382]: click to scroll output; double click to hide

Out[382]: array(['Male', 'Female', ' '], dtype=object)
```

Fig12: checking for the gender unique values

```
In [383]: CheckUnique('Occupation')

Out[383]: array(['Lawyer', 'Energy engineer', 'University Student', ...,
                 'Retired Location manager', 'Retired Engineer, water',
                 'Retired Doctor, hospital'], dtype=object)
```

Fig13: checking for the occupation unique values

```
In [384]: CheckUnique('Infirmity')

Out[384]: array([nan, 'Disabled', 'Mental Disability', 'Physical Disability',
                 'Unknown Infection', ' ', 'Deaf', 'Blind'], dtype=object)
```

Fig14: checking for the Infirmity unique values

```
In [385]: CheckUnique('Religion')

Out[385]: array(['Christian', 'Methodist', 'Muslim', nan, 'Catholic', 'Jewish',
                 'Sikh', 'Private', 'Jedi', 'Agnostic', 'Undecided', 'Baptist',
                 'Hindu', ' ', 'Orthodoxy', 'Sith', 'Pagan', 'Bahai'], dtype=object)
```

Fig15: checking for the religion unique values

To remove empty strings and replace wrong words, we will create a function that can handle that across board.

```
In [387]: #creating a function that could check for empty strings, then replace with a new data then return the uniqueness
          def removeStringAndReplace(sub_data , new_data , index = None):
              first_filter = census_data.loc[census_data[sub_data] == ' ']

              if len(first_filter) > 1:
                  second_filter = census_data.filter(items = [index] , axis = 0).replace(' ' , new_data)
                  census_data.update(second_filter)
                  return second_filter

              else:
                  #replace the data with your prefered data
                  census_data[sub_data] = census_data[sub_data].replace(' ' , new_data)
```

Fig16: This is a reusable function that will be used for removing empty string and replace with a new data specified in the parameter. If it is more than one, then filter via the index then replace for that row and update the global census data with the new data.

Reformatting the Age data

This is to check the missing data and make relevant analysis to know the rightful data to be replaced

census_data.loc[census_data['Age'] == ' ']

Since the age is missing, our analysis shows that the missing data has a marital status being divorced and the relationship to Head of House is Head. That shows the person is more than 18. And the median of the Age is 35

```
In [388]: removeStringAndReplace('Age' , 35)
```

Fig 17: This reusable function is used to replace empty with the median age which is 35.

```
In [389]: #converting my age column to numerics and round it up to an integer
          census_data['Age'] = pd.to_numeric(census_data['Age'], errors='coerce').round(0)
          census_data['Age'] = census_data['Age'].astype('int')
          # since there is no zero age too , we will be replacing the age with the the median of the distribution
          census_data['Age'].replace(0 , census_data['Age'].median() , inplace = True)
          #checks to see the unique figure in the data
          CheckUnique('Age')

Out[389]: array([ 61,  62,  21,  44,  28,  49,  51,  19,  17,  36,  38,   6,   2,
                  43,  41,  15,  42,   5,   3,  73,  40,  13,  11,   7,  64,  39,
                  34,  48,  45,  14,  12,  50,   9,  58,  16,  10,  52,  54,  22,
                  63,  35,  26,  30,  33,   1,  25,  18,  46,   8,  37,   4,  29,
                  47,  53,  31,  24,  55,  27,  69,  59,  66,  68,  20,  81,  32,
                  60,  23,  57,  56,  75,  65,  71,  67,  83,  76,  77,  84,  80,
                  85,  87,  70,  72,  79,  78,  89,  91,  88,  92,  74,  82, 104,
                  98,  90,  86,  97,  94,  96,  93,  99, 103, 102,  95, 106, 100,
                 101, 105, 107])
```

Fig18: We must convert the Age from string to number and replace zero with the median.

Working on Relationship to the Head data

```
In [390]: # there are some typographical error in the relationship to the Head sub data which is neice to niece
census_data['Relationship to Head of House'].replace('Neice' , 'Niece' , inplace = True)
```

Fig19: adjusting the spelling for Niece in the column

Since the Age is over 18 and he is married, the Relationship to the Head is Head for the first missing row.

Checking for the analysis on the missing data in the Relationship to the Head data

```
In [391]: census_data.loc[census_data['Relationship to Head of House'] == ' ']
```

Out[391]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
1967	52	Vulture Parkway	Leanne	Matthews	44		Divorced	Female	Arts development officer	NaN	NaN
7441	54	Hazel Ford	Liam	Daly	15		NaN	Male	Student	NaN	NaN

Fig 20: checking for empty strings

```
In [392]: removeStringAndReplace('Relationship to Head of House' , 'Head' , 1967)
```

Out[392]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
1967	52	Vulture Parkway	Leanne	Matthews	44	Head	Divorced	Female	Arts development officer	NaN	NaN

Fig21: using the function to filter out a specific row via its index then replace it with the right data.

```
In [393]: #Working on the Nan Values on the Relationship to the Head data
census_data[(census_data['Relationship to Head of House'].isnull()) & (census_data['Marital Status'] == 'Single')& (
```

Out[393]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
1	1	Jones Crossing	Duncan	Clark	62	NaN	Single	Male	Energy engineer	NaN	Christian
2	1	Jones Crossing	Sian	Page	21	NaN	Single	Female	University Student	NaN	Methodist
3	1	Jones Crossing	Kevin	Jones	44	NaN	Single	Male	Set designer	NaN	Methodist
4	1	Jones Crossing	Derek	Stephenson	28	NaN	Single	Male	Psychotherapist	NaN	Muslim
58	16	Jones Crossing	Ryan	Price	41	NaN	Single	Male	Designer, furniture	NaN	Catholic
...	...	...	...	...	...	...	...	...	...	...	...
8121	2	Law Flat	Lynn	Richards	26	NaN	Single	Female	Editor, magazine features	NaN	NaN
8125	5	Law Flat	Charlotte	Dennis	63	NaN	Single	Female	Environmental manager	NaN	Christian
8129	8	Law Flat	Arthur	Turner	30	NaN	Single	Male	Dentist	NaN	Catholic
8130	8	Law Flat	Samuel	Nixon	43	NaN	Single	Male	Set designer	Physical Disability	Christian
8131	8	Law Flat	Norman	Hurst	38	NaN	Single	Male	Theme park manager	NaN	Catholic

553 rows × 11 columns

Fig22: filtering the singles with empty values so as to replace with the right data

```
In [394]: census_data[(census_data['Relationship to Head of House'].isnull()) & (census_data['Marital Status'] != 'Single') &
```

Out[394]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
682	25	St.Lukecoffer Crescent	Jodie	Evans	36	NaN	Married	Female	Travel agency manager	NaN	NaN
683	25	St.Lukecoffer Crescent	Wayne	Evans	40	NaN	Married	Male	Translator	NaN	NaN
741	47	St.Lukecoffer Crescent	Anna	O'Sullivan	39	NaN	Married	Female	Financial manager	NaN	Christian
742	47	St.Lukecoffer Crescent	Alexander	O'Sullivan	40	NaN	Married	Male	Engineer, biomedical	NaN	NaN
863	7	Richardson Estates	Eleanor	Hughes	25	NaN	Divorced	Female	Designer, industrial/product	NaN	Christian
1104	101	Richardson Estates	Emma	Cox	51	NaN	Divorced	Female	Surgeon	NaN	Methodist
1393	2	Dumnonia Road	Sophie	Taylor	52	NaN	Married	Female	Development worker, community	NaN	Catholic
1394	2	Dumnonia Road	Lewis	Taylor	50	NaN	Married	Male	Financial controller	NaN	Catholic
1396	2	Dumnonia Road	Daniel	Taylor	25	NaN	Divorced	Male	Public relations officer	NaN	Catholic
2170	13	Jones Corners	Harry	Byrne	31	NaN	Divorced	Male	Engineer, automotive	NaN	NaN
2627	47	Osborne Isle	Hannah	Bryan	36	NaN	Married	Female	Patent attorney	NaN	NaN
2628	47	Osborne Isle	Leon	Bryan	36	NaN	Married	Male	Unemployed	NaN	NaN

Fig23: filtering the non-singles with empty values to replace with the right data.

```
In [395]: census_data_missing_single_male = census_data[(census_data['Relationship to Head of House'].isnull()) & (census_data
          census_data_missing_single_male_filtered = census_data['Relationship to Head of House'].replace(np.nan , 'Head')
          census_data.update(census_data_missing_single_male_filtered)

In [396]: sus_data_missing_single_female = census_data[(census_data['Relationship to Head of House'].isnull()) & (census_data['
          sus_data_missing_single_female_filtered = census_data['Relationship to Head of House'].replace(np.nan , 'Head')
          sus_data.update(census_data_missing_single_female_filtered)

In [397]: ıs_data_missing_not_single_male = census_data[(census_data['Relationship to Head of House'].isnull()) & (census_data[
          ıs_data_missing_not_single_male_filtered = census_data['Relationship to Head of House'].replace(np.nan , 'Husband')
          ıs_data.update(census_data_missing_not_single_male_filtered)

In [398]: census_data_missing_not_single_female = census_data[(census_data['Relationship to Head of House'].isnull()) & (censu
          census_data_missing_not_single_female_filtered = census_data['Relationship to Head of House'].replace(np.nan , 'Wife
          census_data.update(census_data_missing_not_single_female_filtered)
```

Fig24: Replacing the appropriate data for different relationships to the head with good logics to remove lies in the data

```
In [401]: removeStringAndReplace('Relationship to Head of House' , 'Son' , 7441)

In [402]: census_data.loc[census_data['Relationship to Head of House'] == ' ']
```

Out[402]:

House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion

Fig25: After all processing, we have no empty strings and no null values

While exploring, we discovered some lies given by the households, some are individuals. We tend to use our analytical skills to fish them out to avoid inconsistencies in the results.

Analysis trying to shed some light on the age census that shows lies or deceit.

In [430]:
```
#ages before 18th year birthday and they are the head of the house
census_data.loc[(census_data['Age'] < 18) & (census_data['Relationship to Head of House'] == 'Head')]
```

Out[430]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
264	28	Windy Dale	Tracy	Vaughan	17	Head	Married	Female	Student	No Infirmity	No Religion
743	47	St.Lukecoffer Crescent	Lindsey	O'Sullivan	16	Head	Single	Female	Student	No Infirmity	No Religion
1397	2	Dumnonia Road	Eric	Taylor	17	Head	Single	Male	Student	No Infirmity	No Religion
2629	47	Osborne Isle	Julia	Green	17	Head	Single	Female	Student	No Infirmity	No Religion
2630	47	Osborne Isle	Jack	Green	16	Head	Single	Male	Student	No Infirmity	No Religion
2631	47	Osborne Isle	Kathleen	Green	16	Head	Single	Female	Student	No Infirmity	No Religion
3055	32	Shepherd Crescent	Gerald	Scott	16	Head	Single	Male	Student	No Infirmity	No Religion
3056	32	Shepherd Crescent	Lindsey	Scott	13	Head	Single	Female	Student	No Infirmity	No Religion
6755	19	Jarvis Locks	Ashley	Vaughan-Wilson	13	Head	Single	Male	Student	No Infirmity	No Religion
7987	1	Orchardnip Road	Brenda	Stone	17	Head	Single	Female	Unemployed	No Infirmity	No Religion

In [431]:
```
# This is more filtering to drill down the liers in the data
census_data.loc[(census_data['Age'] < 50) & (census_data['Relationship to Head of House'] == 'Head') & (census_data[
```

Out[431]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
391	25	Taylor Radial	Jamie	Duncan	18	Head	Divorced	Male	Student	No Infirmity	No Religion
5428	9	Drummer Motorway	Tracy	Simpson	18	Head	Divorced	Female	Student	No Infirmity	No Religion

General analysis of the whole Dataset

Based on this data it is shown that there are Age below 18 and they refer to themselves as the head of the household.

Working on the Marital Status Data

```
In [404]: # creating the analysis on the marital status filter out the Nan values
          census_data[census_data['Marital Status'].isnull()]
```

Out[404]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
8	2	Jones Crossing	Leon	Stevens	17	Son	NaN	Male	Student	NaN	NaN
11	3	Jones Crossing	Jean	Wilkinson	6	Daughter	NaN	Female	Student	NaN	NaN
12	3	Jones Crossing	Natasha	Rhodes	2	Daughter	NaN	Female	Child	NaN	NaN
15	4	Jones Crossing	Amelia	Faulkner	17	Daughter	NaN	Female	Student	NaN	NaN
16	4	Jones Crossing	Michael	Faulkner	15	Son	NaN	Male	Student	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
8221	1	England Folly	Kenneth	Russell	11	Son	NaN	Male	Student	NaN	NaN
8222	1	England Folly	Carolyn	Russell	11	Daughter	NaN	Female	Student	NaN	NaN
8223	1	England Folly	Kerry	Brooks	10	Daughter	NaN	Female	Student	NaN	NaN
8229	1	Scepter Factory	Ruth	Frost	9	Adopted Daughter	NaN	Female	Student	NaN	NaN
8236	1	Oyster Obervatory	Nigel	Andrews	2	Son	NaN	Male	Child	NaN	NaN

1989 rows × 11 columns

Fig26: checking for nan values in the marital status column.

IF THE AGE IS GREATER THAN 18 YEARS THEN THE MARITAL STATUS WON'T BE SINGLE, IF THE OCCUPATION OF THE NAN VALUE IS CHILD IT HAS TO BE EITHER SON OR DAUGTHER DEPENDING ON THE GENDER.

```
In [405]: marital_status_filtered = census_data[census_data['Marital Status'].isnull()]
          age_less_than_eighteen = marital_status_filtered[marital_status_filtered['Age'] < 18]
          age_less_than_eighteen_replaced = age_less_than_eighteen['Marital Status'].replace(np.nan , 'Single')
          census_data.update(age_less_than_eighteen_replaced)
```

Fig27: replacing nan values with age less than 18 with marital status Single.

```
In [408]: # writing a python on how to categorize the rest based on analysis
          marital_status_filtered = census_data[(census_data['Marital Status'].isnull()) & (census_data['Occupation'] == 'Chil
          marital_status_child = marital_status_filtered['Marital Status'].replace(np.nan , 'Single')
          census_data.update(marital_status_child)
```

Fig28: Working on the data occupation as child should be changed as single status

Working on the Gender Data

```
In [409]: #searching for empty data in the gender columns
          census_data.loc[census_data['Gender'] == ' ']
```

Out[409]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
5968	26	Murray Coves	Craig	Reed	56	Partner	Single		Paediatric nurse	NaN	NaN

Fig29: checking for the empty string

```
In [410]: #using the function to remove and replace the empty string with the right data
          removeStringAndReplace('Gender' , 'Male')
```

Fig30: replacing the empty gender column with male

```
In [411]: #checking for any null values
          census_data['Gender'].isnull().sum()

Out[411]: 0
```

Fig31: checking for sum of empty values left in the column

Working on the Occupation Data

```
In [412]: CheckUnique('Occupation')

Out[412]: array(['Lawyer', 'Energy engineer', 'University Student', ...,
                 'Retired Location manager', 'Retired Engineer, water',
                 'Retired Doctor, hospital'], dtype=object)
```

Fig32: checking for misspelled words in the column.

```
In [41]:  click to scroll output; double click to hide
          in the occupation columns and adjusting some of the occupation list
          census_data['Occupation'].replace('Retired Engineer, water' ,'Retired Water Engineer', inplace=True)

In [414]: #searching for empty data in the occupation columns and adjusting some of the occupation list
          census_data['Occupation'].replace('Retired Doctor, hospital' ,'Retired Doctor', inplace=True)
```

Fig33: change the misspelled words in the column

```
In [415]: # checking if there is any missing values
          census_data['Occupation'].isnull().sum()

Out[415]: 0
```

Fig34: checking for nan values in the column

Working on the Infirmity Data

```
In [416]: #Checking the uniqueness of this Field
          CheckUnique('Infirmity')

Out[416]: array([nan, 'Disabled', 'Mental Disability', 'Physical Disability',
                 'Unknown Infection', ' ', 'Deaf', 'Blind'], dtype=object)
```

Fig35: checking for unique values in the infirmity data

```
In [417]: #searching for empty data in the infirmity columns
          census_data.loc[census_data['Infirmity'] == ' ']
```

Out[417]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
1498	36	Dumnonia Road	Luke	Stevenson	43	Head	Single	Male	Advertising account planner		Jewish
2166	12	Jones Corners	Connor	Robinson	23	Son	Divorced	Male	Unemployed		Catholic
2565	30	Osborne Isle	Glenn	Wallace	53	Husband	Married	Male	Medical secretary		Catholic
3161	3	Fletcher Spur	Elaine	Robson	43	Head	Married	Female	Passenger transport manager		Catholic
3758	1	Walters Inlet	Kate	Holland	5	Daughter	Single	Female	Student		NaN
4029	14	Coronation Drive	Abdul	Walker	71	Husband	Married	Male	Retired Secondary school teacher		NaN
4109	23	Elliott Bypass	Molly	Lynch	45	Head	Single	Female	Arts development officer		Christian
4889	149	Williams Roads	Jonathan	Webb	37	Husband	Married	Male	Unemployed		Sikh
5076	24	Edwards Fall	Malcolm	Dyer	56	Head	Single	Male	Editor, commissioning		NaN
6035	17	Holden Locks	Patricia	Smith	43	Head	Divorced	Female	Counselling psychologist		Methodist
6919	48	Jarvis Locks	Shannon	Ward	5	Daughter	Single	Female	Student		NaN

Fig36: checking for the empty strings in the data.

```
In [421]: #replace the empty data with the most frequent data
          census_data['Infirmity'].replace(' ' , 'No Infirmity' , inplace=True)
```

Fig37: replace the empty string with No Infirmity

```
In [422]: #removing the missing values in the infimity column
          census_data['Infirmity'].fillna('No Infirmity',inplace=True)
```

Fig38: replace the nan values with No infirmity

Working on the Religion Data

```
In [425]: #most frequent data in the religion column and the value counts
          census_data['Religion'].value_counts()
```

Out[425]:
```
Religion
Christian    1792
Catholic      897
Methodist     494
Muslim        108
Sikh           42
Jewish         26
Private         3
                1
Pagan           1
Sith            1
Orthodoxy       1
Agnostic        1
Hindu           1
Baptist         1
Undecided       1
Jedi            1
Bahai           1
Name: count, dtype: int64
```

Fig39: checking the value counts of several types of religion we have

```
In [426]: #searching for empty data in the religion column
          census_data.loc[census_data['Religion'] == ' ']
```

Out[426]:

click to scroll output; double click to hide

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion
5902	17	Murray Coves	Amanda	Sanders	20	Lodger	Single	Female	University Student	No Infirmity	

Fig40: checking empty strings in the religion column data

```
In [427]: #replace the empty data with the most frequent data
          census_data['Religion'].replace(' ' , 'No Religion' , inplace=True)

In [428]: # replace the na values in the religion data
          census_data['Religion'].fillna( 'No Religion' , inplace=True)
```

Fig41: replacing empty strings and null values with No Religion

```
In [429]: #checking if the data has null
          census_data.isnull().sum()
```

click to scroll output; double click to hide

```
Out[429]: House Number                    0
          Street                          0
          First Name                      0
          Surname                         0
          Age                             0
          Relationship to Head of House   0
          Marital Status                  0
          Gender                          0
          Occupation                      0
          Infirmity                       0
          Religion                        0
          dtype: int64
```

Fig42: This shows we have no null values, empty string, and unformatted values in your dataset.

This is the end of the cleaning process!!!!

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial process of systematically examining and representing datasets to reveal underlying patterns, trends, and connections. By using tools such as summary statistics, charts, and graphs, EDA aids in comprehending the distribution and organization of the data. This initial investigation assists in formulating hypotheses, informing decision-making, and conducting further analysis in diverse fields, thereby providing a holistic understanding of the features of the dataset.

We will cover insights showing why we are making these decisions.

EXERCISE A(a)

What should be built on an unoccupied plot of land that the local government wishes to develop?

High-density housing. This should be built if the population is significantly expanding.

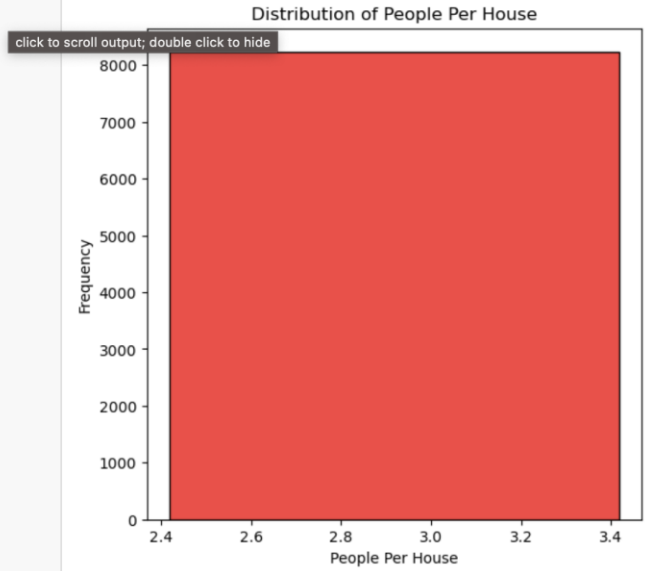
This should be high density housing because the number of occupancy levels is increasing by the number of people per house.

In [450]:
```
#Converting the house number to a string value so it can concatenate  with the Street
census_data['House Number'] = census_data['House Number'].astype('str')
# Concatenating the House Number with the street Name to get the House Number and Street Name together
census_data['Household'] = census_data['Street'] + ' ' + census_data['House Number']
#checking the number of unique house holds for each house
total_households = census_data['Household'].nunique()
#getting the total population of the census data
census_data['Total Population'] = general_population
#getting the total number of households in the data
census_data['Number of Houses'] = total_households
#getting the total people per each house
census_data['People_Per_House'] = census_data['Total Population'] / census_data['Number of Houses']

# visualizing distribution of people per house
fig, ax = plt.subplots(figsize=(6, 6))
sns.histplot(census_data['People_Per_House'], color = 'red')
plt.title('Distribution of People Per House')
plt.xlabel('People Per House')
plt.ylabel('Frequency')
plt.show()

# Calculate the average people per house
avg_people_per_house = census_data['People_Per_House'].mean()

# Determine if housing is underused, overused, or balanced
if avg_people_per_house < 1:
    print('Housing is underused.')
elif 1 <= avg_people_per_house <= 2:
    print('Housing is reasonably used.')
else:
    print('Housing is overused.')
```




```
Housing is overused.
```

Based on this analysis, it shows that the total average number of people per household is 3. It means the household might be overused, that brings about creating more households for people to stay in.

EXERCISE A(b)

low-density housing. This should be built if the population is “affluent” and there is demand for large family housing.

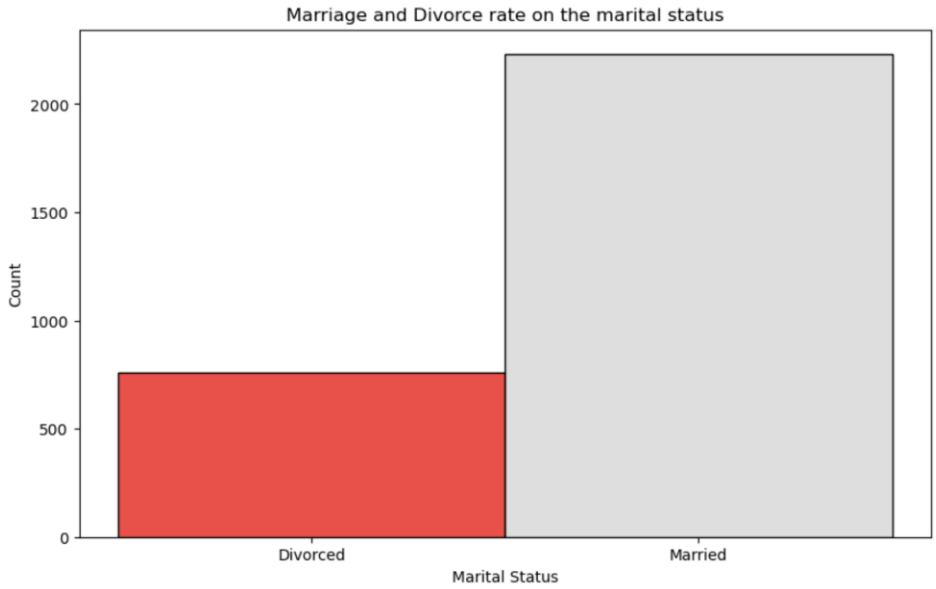
There is also a need for large family housing too because we have more married people than divorced people.

```
In [87]: fig, ax = plt.subplots(figsize=(10, 6))

         #plotting the graph
         sns.histplot(data=census_divorced, x = 'Marital Status'  , color = 'red',  ax=ax)
         sns.histplot(data=census_married , x = 'Marital Status'  , color = 'lightgray',  ax=ax)

         # Labelling
         plt.title('Marriage and Divorce rate on the marital status')

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



Based on the charts, it shows we have more married couples than divorced. Based on this analysis we will need low-density housing for those areas that have such conditions.

EXERCISE A(c)

Train station. There are potentially a lot of commuters in the town and building a train station could take pressure off the roads. But how will you identify commuters?

In [88]:
```
for i in census_data['Occupation'].unique():
    print(i)
```

```
Associate Professor
Engineer, manufacturing systems
Analytical chemist
Journalist, magazine
Advertising account executive
Building surveyor
Psychotherapist, child
Immigration officer
Conference centre manager
Copywriter, advertising
Designer, furniture
Engineer, chemical
Publishing rights manager
Research scientist (life sciences)
Trading standards officer
Operations geologist
Cartographer
Land
Occupational therapist
Best boy
```

In [89]:
```
#filtering out the university students in the list in the occupation
uni_students = census_data[census_data['Occupation'] == 'University Student']
#getting the total length of the university student
num_uni_students = len(uni_students)
#printing the number of the university students
print(f'The number of university students: {num_uni_students}')
#getting other professionals that is not a university student as a list
census_professionals_filtered = census_data['Occupation'].unique()
census_professionals_filtered = list(dict.fromkeys(census_professionals_filtered))
del census_professionals_filtered[2]

#assigning the commuter professions
commuter_professions = census_professionals_filtered

#check to see if it is likely they are on the list
likely_commuters = census_data[census_data['Occupation'].isin(commuter_professions)]
num_likely_commuters = len(likely_commuters)
print(f'The number of likely commuters in other professions: {num_likely_commuters}')
print(likely_commuters['Occupation'].value_counts())
```

```
The number of university students: 543
The number of likely commuters in other professions: 7694
Occupation
Student                                          1603
Unemployed                                        512
Child                                             494
Chiropractor                                       18
TEFL teacher                                       15
                                                 ...
Retired Chemist, analytical                         1
Retired Emergency planning/management officer       1
Retired Archivist                                   1
Retired Computer games developer                    1
Retired Doctor                                      1
Name: count, Length: 1018, dtype: int64
```

There is a need for a train station since we have a lot of other professionals working in the town. We have the students being the highest number of people in the town while we have other professionals moving to other places.

EXERCISE A(d)

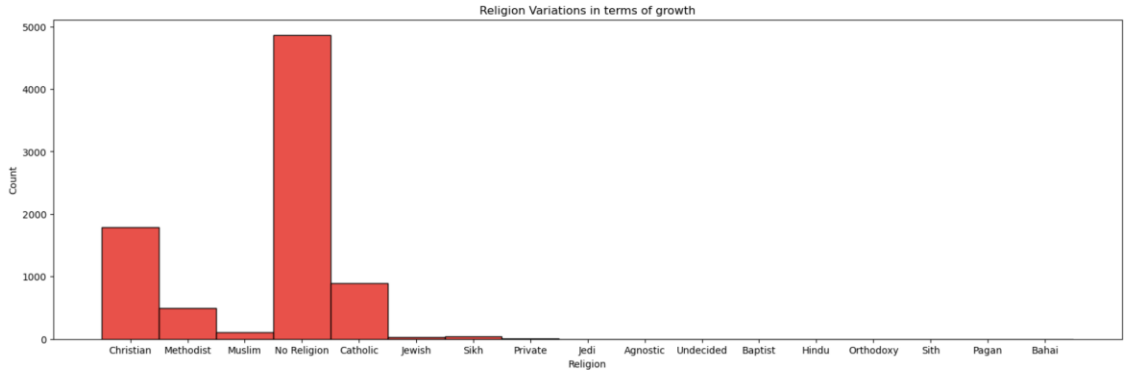
Religious building. There is already one place of worship for Catholics in the town. Is their demand for a second Church (if so, which denomination?), or for a different religious building?

In [90]:
```
fig, ax = plt.subplots(figsize=(20, 6))

#plotting the graph
sns.histplot(data=census_data , x = 'Religion'  , color = 'red',  ax=ax)

# Labelling
plt.title('Religion Variations in terms of growth')

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



Based on the religious denominations apart from Catholics, we have a lot of Christians, methodists too. We can have another church built there apart. This shows that there are more of less religious people in the Uk.

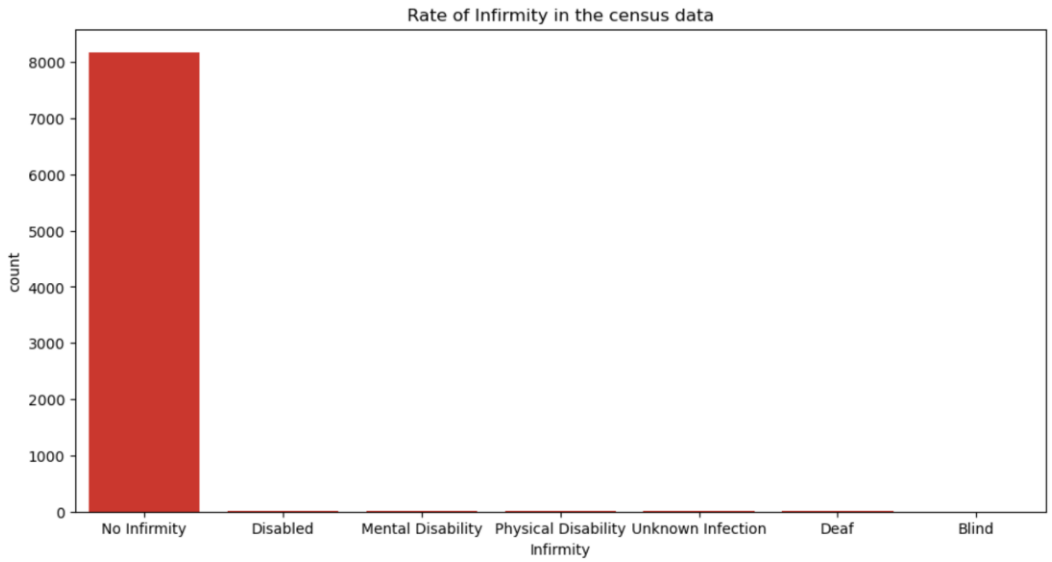
EXERCISE A(e)

Emergency medical building. Not a full hospital, but a minor injuries center. This should be built if there are many injuries or future pregnancies in the population.

In terms of health, most analysis will be drilled down on the infirmity column to explore more.

Based on the Analysis made, it is advisable to have a small medical facility due to a smaller number of medical conditions. but the rate of getting pregnancies might also increase due to the high number of married couples too.

In [91]:
```
fig , ax = plt.subplots(figsize = (12,6))
sns.countplot(census_data, x="Infirmity" , color = 'red')
plt.title('Rate of Infirmity in the census data');
```

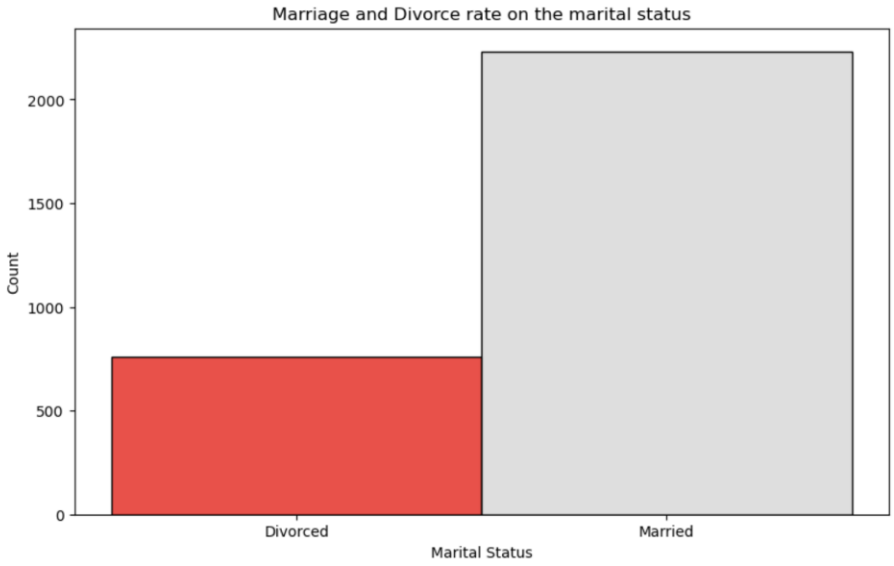


In [92]:
```
fig, ax = plt.subplots(figsize=(10, 6))

#plotting the graph
sns.histplot(data=census_divorced, x = 'Marital Status'  , color = 'red',  ax=ax)
sns.histplot(data=census_married , x = 'Marital Status'  , color = 'lightgray',  ax=ax)

# Labelling
plt.title('Marriage and Divorce rate on the marital status')

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



Exercise B(A)

Employment and training. If there is evidence of a lot of unemployment, we should re-train people in new skills.

```
In [93]: census_filtered = census_data[census_data['Occupation'] == 'Unemployed']
         census_filtered
```

Out[93]:

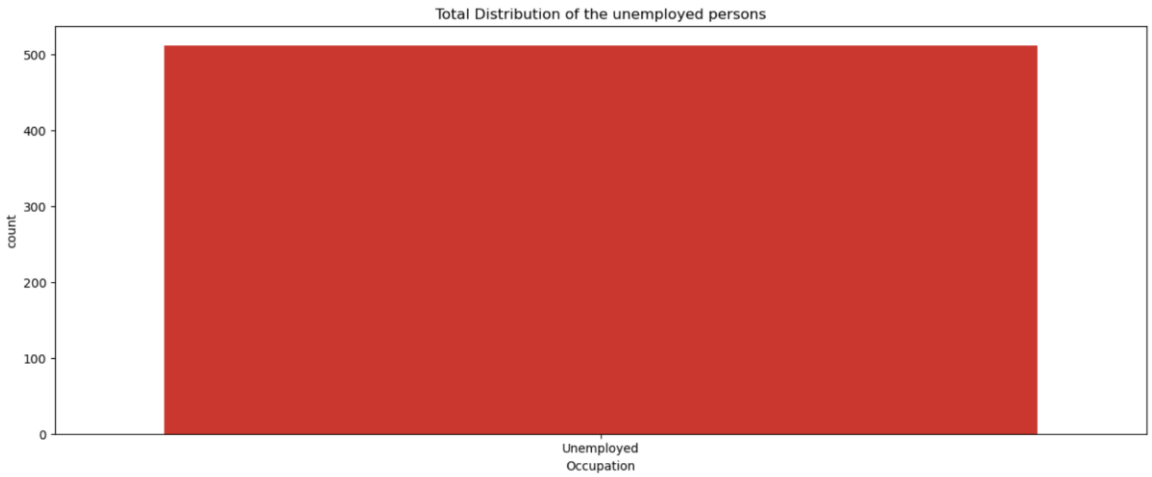
	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion	Household	Total Population	Number of Houses	People
17	5	Jones Crossing	Charlene	Turner	42	Head	Single	Female	Unemployed	No Infirmity	Christian	Jones Crossing 5	8237	2823	
47	14	Jones Crossing	Georgia	Parker-Jenkins	44	Head	Married	Female	Unemployed	No Infirmity	Catholic	Jones Crossing 14	8237	2823	
80	22	Jones Crossing	Sheila	Davies	37	Head	Single	Female	Unemployed	No Infirmity	Catholic	Jones Crossing 22	8237	2823	
106	29	Jones Crossing	Beth	Bell	53	Head	Single	Female	Unemployed	No Infirmity	No Religion	Jones Crossing 29	8237	2823	
111	30	Jones Crossing	Mary	Ahmed	31	Head	Single	Female	Unemployed	No Infirmity	No Religion	Jones Crossing 30	8237	2823	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8033	7	Metropolis Street	Janet	Thomas	39	Head	Single	Female	Unemployed	No Infirmity	Catholic	Metropolis Street 7	8237	2823	
8039	9	Metropolis Street	Julie	Riley	40	Head	Divorced	Female	Unemployed	No Infirmity	No Religion	Metropolis Street 9	8237	2823	
8083	8	Marshall Pass	Hilary	Mistry	26	Head	Divorced	Female	Unemployed	No Infirmity	No Religion	Marshall Pass 8	8237	2823	
8091	11	Marshall Pass	Shaun	Woodward	24	Head	Single	Male	Unemployed	No Infirmity	No Religion	Marshall Pass 11	8237	2823	
8188	1	Frost Factory	Andrea	Shah	49	Head	Divorced	Female	Unemployed	No Infirmity	No Religion	Frost Factory 1	8237	2823	

512 rows × 15 columns

```
In [94]: census_filtered.shape
```

Out[94]: (512, 15)

```
In [95]: fig, ax = plt.subplots(figsize=(16, 6))
         sns.countplot(data = census_filtered , x = 'Occupation' , color = 'red')
         plt.title('Total Distribution of the unemployed persons')
         plt.show()
```



Based on the analysis made, it is explained that we have 512 people that are unemployed in the census data

There is a need to train people for new skills so they can benefit more from the skill acquisition.

```
In [96]: for i in census_data['Occupation'].unique():
             print(i)
```

```
Lawyer
Energy engineer
University Student
Psychotherapist
Administrator, arts
Engineer, production
Student
Tour manager
Chief of Staff
Child
Furniture designer
Research scientist (medical)
Unemployed
Retired Designer, multimedia
Research scientist (physical sciences)
Financial controller
Adult guidance worker
Media buyer
Equities trader
Engineer, control and instrumentation
Ergonomist
Immunologist
Associate Professor
Engineer, manufacturing systems
Analytical chemist
```

click to scroll output; double click to hide

Exercise B(b)

Old age care. If there is evidence for increasing numbers of retired people in future years, The town will need to allocate more funding for end-of-life care.

Based on my research it is stated that ages above 66 are retired in the Uk.

In [97]:
```
# Find unique values in the 'player_name' column containing the word 'retired'
retired_players = census_data[census_data['Occupation'].str.contains('Retired', case=False)]
retired_players
```

Out[97]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion	Household	Total Population	Number of Houses	Peop
20	6	Jones Crossing	Norman	Marshall	73	Head	Widowed	Male	Retired Designer, multimedia	No Infirmity	Christian	Jones Crossing 6	8237	2823	
126	1	Ripon Cove	June	Hughes	69	Head	Divorced	Female	Retired Theatre manager	No Infirmity	Christian	Ripon Cove 1	8237	2823	
146	13	Ripon Cove	Mandy	Davies	68	Head	Single	Female	Retired Therapist, occupational	No Infirmity	No Religion	Ripon Cove 13	8237	2823	
154	18	Ripon Cove	Bethany	Harvey	81	Head	Single	Female	Retired Ergonomist	No Infirmity	No Religion	Ripon Cove 18	8237	2823	
289	35	Windy Dale	Cameron	Morris	75	Head	Married	Male	Retired Geophysical data processor	No Infirmity	Christian	Windy Dale 35	8237	2823	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8118	1	Law Flat	Louise	Leach	78	Head	Widowed	Female	Retired Water Engineer	No Infirmity	Christian	Law Flat 1	8237	2823	
8126	6	Law Flat	Danielle	Harper	71	Head	Widowed	Female	Retired Ranger/warden	No Infirmity	Christian	Law Flat 6	8237	2823	
8127	7	Law Flat	Naomi	Green	71	Head	Single	Female	Retired International aid/development worker	No Infirmity	No Religion	Law Flat 7	8237	2823	
8132	9	Law Flat	Kimberley	Curtis	70	Head	Divorced	Female	Retired Doctor	No Infirmity	Christian	Law Flat 9	8237	2823	
8183	1	Green Manor	Abigail	Reeves	80	Head	Widowed	Female	Retired Financial risk analyst	No Infirmity	No Religion	Green Manor 1	8237	2823	

584 rows × 15 columns

In [98]:
```
retired_players.shape
```

Out[98]: (584, 15)

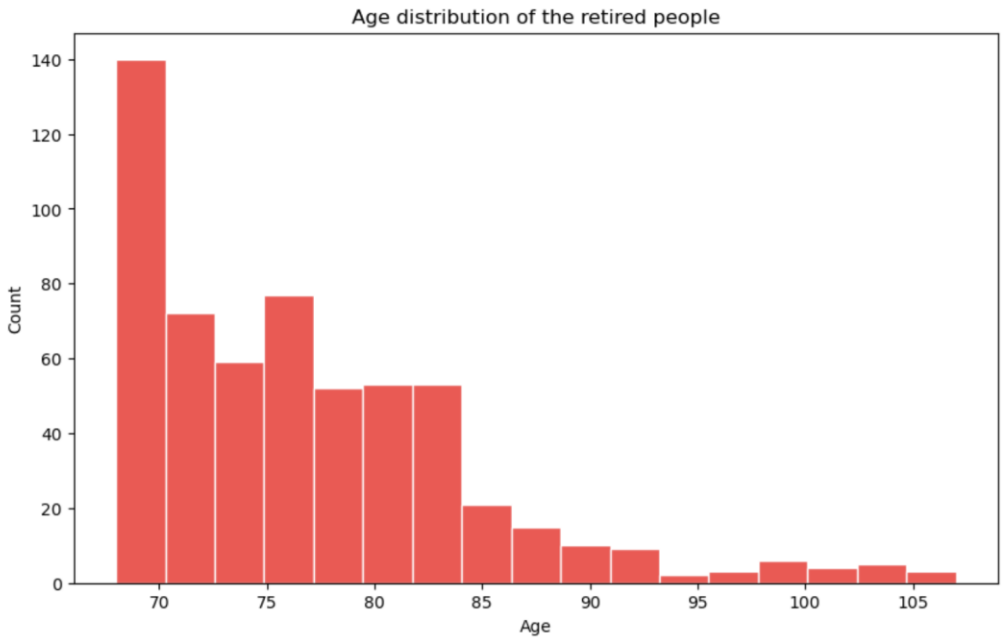
There are no retired people under 66, this shows the analysis was right.

In [99]:
```
retired_players[retired_players['Age'] < 66]
```

Out[99]:

	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion	Household	Total Population	Number of Houses	People_Per_House

```
In [100]: plt.figure(figsize=(10, 6))
          sns.histplot(data = retired_players , x = 'Age' , color = 'red', edgecolor='white', alpha=0.7)
          plt.title("Age distribution of the retired people");
```



There are less increasing number of unemployed people in the society with 584 out of the 8237 of the total population.

Exercise B(c)

Increase spending for schooling. If there is evidence of a growing population of school-aged children (new births, or families moving into the town), then school spending should increase.

In [101]:
```
census_data_school_age_children = census_data[ census_data['Age'] < 2]
census_data_school_age_children
```

Out[101]:

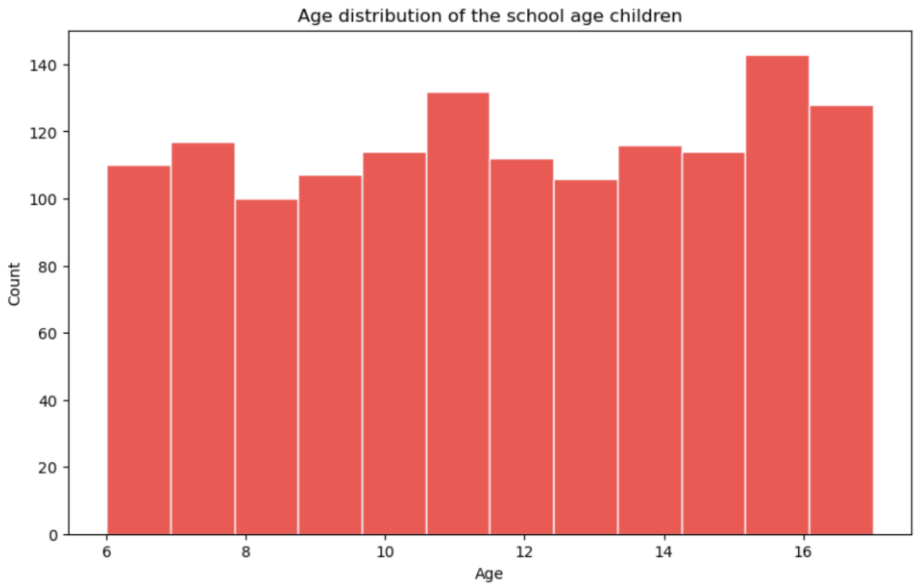
	House Number	Street	First Name	Surname	Age	Relationship to Head of House	Marital Status	Gender	Occupation	Infirmity	Religion	Household	Total Population	Number of Houses	Pe
66	18	Jones Crossing	Kelly	Knight	1	Daughter	Single	Female	Child	No Infirmity	No Religion	Jones Crossing 18	8237	2823	
174	6	O'Sullivan Cape	Jade	Hughes	1	Daughter	Single	Female	Child	No Infirmity	No Religion	O'Sullivan Cape 6	8237	2823	
224	14	Windy Dale	Jack	Brown	1	Son	Single	Male	Child	No Infirmity	No Religion	Windy Dale 14	8237	2823	
239	18	Windy Dale	Antony	Davison	1	Son	Single	Male	Child	No Infirmity	No Religion	Windy Dale 18	8237	2823	
345	9	Taylor Radial	Samuel	Elliott	1	Son	Single	Male	Child	No Infirmity	No Religion	Taylor Radial 9	8237	2823	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7830	15	ExcaliburBells Creek	Benjamin	Watson	1	Son	Single	Male	Child	No Infirmity	No Religion	ExcaliburBells Creek 15	8237	2823	
7941	30	Williams Street	Liam	Leach	1	Son	Single	Male	Child	No Infirmity	No Religion	Williams Street 30	8237	2823	
7956	36	Williams Street	Ashley	Thomas	1	Son	Single	Male	Child	No Infirmity	No Religion	Williams Street 36	8237	2823	
8176	1	Pendragon Granary	Ann	Tucker	1	Granddaughter	Single	Female	Child	No Infirmity	No Religion	Pendragon Granary 1	8237	2823	
8191	1	Frost Factory	Helen	Shah	1	Daughter	Single	Female	Child	No Infirmity	No Religion	Frost Factory 1	8237	2823	

77 rows × 15 columns

In [103]:
```
census_data_school_age_children.shape
```

Out[103]: (1399, 15)

In [104]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data = census_data_school_age_children , x = 'Age'  ,  color = 'red', edgecolor='white', alpha=0.7)
plt.title("Age distribution of the school age children");
plt.show();
```



We have more than A thousand children in school, we need to increase funding for schools.

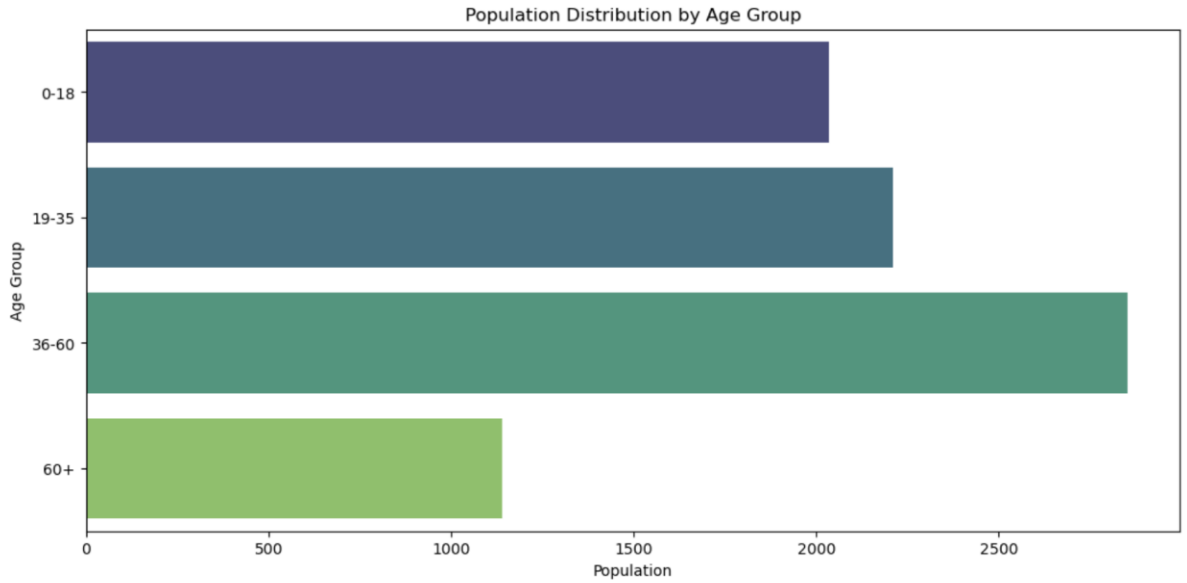
Exercise B(D)

General infrastructure. If the town is expanding, then services (waste collection; road maintenance, etc.) will require more investment.

In [105]:
```
# this will split the ages into different categories
groups_of_ages = pd.cut(census_data['Age'], bins=[0, 18, 35, 60, float('inf')], labels=['0-18', '19-35', '36-60', '6
age_group_using_population = census_data.groupby(groups_of_ages).size().reset_index(name='Population')
print(age_group_using_population)
```

```
     Age  Population
0   0-18        2034
1  19-35        2210
2  36-60        2853
3    60+        1140
```

In [106]:
```
# Categorical visualization with Seaborn
plt.figure(figsize=(13, 6))
sns.barplot(x='Population', y='Age', data=age_group_using_population, palette='viridis')
plt.title('Population Distribution by Age Group')
plt.xlabel('Population')
plt.ylabel('Age Group')
plt.show()
```



This shows that there are more young people in the census data than older people. Which means that we might have more death rate than birth rate.

This shows that at this rate we will have the population expanding, and there is a huge amount of increasing workforce.

This will give rise to the need for infrastructure in the economy.

CONCLUSION

In summary, the UK census goes beyond simply counting people by providing a crucial perspective on society. Its comprehensive recording of specific information, ranging from street names to occupations, allows for a deep comprehension of the nation's makeup. This all-encompassing method guarantees an accurate portrayal of socio-economic variety, equipping leaders with invaluable insights. The census is more than just a statistical project; it is a strategic tool for informed decision-making, reflecting the intricate complexities of individual lives and contributing to the development of a dynamic and adaptable society, ready to face any future challenges and prospects.