Table of Contents: ¶

Importing Libraries

```
All the libraries are imported.
The functions are defined which will be required later.
The csv data is imported using pandas library
```

Data Cleaning

```
Data is cleaned to remove missing values.
Categorical data is encoded to numerical values.
Visualization of missing data is performed.
Performed label encoding and one hot encoding.
Final visualizaiton using heat map is done to check no missing data.
```

Exploratory Data Analysis

Feature Selection

```
Correlation plot is developed to look for most correlated features.
Random forest classifier is used to select the required important feature.
PCA and Scaling is performed to reduce the dimension of features selected.
```

Model Implementation and Hyperparameter Tuning

```
The train and test dataset is separated initially for the logistic regression mode
l.
First the logistic regression model is run with the training dataset and the result
is generated.

Tuning of hyperparameters: C and solver is done with the Grid Search.
Optimal parameters are found.

The training data set is again run using the logistic regression model with new par
ameters and results are obtained.
The model is tested on testing dataset and final conclusion is made.
```

Testing and Discussion

```
The train and test results are plotted on the bar plot of salary buckets.
```

Importing Libraries

In [1]:
```
# All the required libraries is imported and notebook is made ready for the co
de.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.ensemble.forest import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split, cross_val_score, learnin
g_curve, KFold, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score
import warnings
#warnings.filterwarnings("ignore", category=ConvergenceWarning)
warnings.filterwarnings("ignore")
from sklearn.metrics import make_scorer, confusion_matrix
import random
```

The explanation of previously defined function:

1. ms_perc: This function finds the missing values in the dataframe and calculates the percentage of missingness.
2. plot_learning_curve (Assignment 2 Tutorial): This function returns the learning curve plot.

In [2]:
```
# Function to find missing values

'''
Input = Dataframe
Output = missing percentage value
'''

def ms_perc(file):
    ms_ct = file.isna().sum()
    total = np.product(file.shape)
    ms_total = ms_ct.sum()
    ms_value = (ms_total/total)*100
    return ms_value
```

In [3]:
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=1,\
                        train_sizes=np.linspace(.1, 1.0, 5), scoring='accurac
y'):

    plt.figure(figsize=(10,6))
    plt.title(title)

    if ylim is not None:
        plt.ylim(*ylim)

    plt.xlabel("Training examples")
    plt.ylabel(scoring)

    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, c
v=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,\
                     train_scores_mean + train_scores_std, alpha=0.1, \
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,\
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")

    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",label="Training s
core")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-vali
dation score")
    plt.legend(loc="best")

    return plt
```

Now, the dataframe is imported using pandas library. The original kaggle data is copied to another variable if used later.

In [4]:
```
# Reading dataframe

kaggle_data = pd.read_csv('clean_kaggle_data_2020.csv', low_memory = False)
kaggle_original = kaggle_data.copy()
kaggle_data.head()
```

Out[4]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2
0	Duration (in seconds)	What is your age (# years)?	What is your gender? - Selected Choice	In which country do you currently reside?	What is the highest level of formal education ...	Select the title most similar to your current ...	For how many years have you been writing code ...	What programming languages do you use on a reg...	What programming languages do you use on a reg...
1	289287	30-34	Man	United States of America	Master’s degree	Data Engineer	5-10 years	Python	R
2	860	35-39	Man	Argentina	Bachelor’s degree	Software Engineer	10-20 years	NaN	NaN
3	507	30-34	Man	United States of America	Master’s degree	Data Scientist	5-10 years	Python	NaN
4	762	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	Python	NaN

5 rows × 357 columns

Data Cleaning

The kaggle_data is inspected by looking at column values and the missing percentage in the database.

In [5]:
```
kaggle_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10730 entries, 0 to 10729
Columns: 357 entries, Time from Start to Finish (seconds) to Q24_buckets
dtypes: float64(1), object(356)
memory usage: 29.2+ MB
```

In [6]:
```
kaggle_data.columns.values
```

Out[6]:
```
array(['Time from Start to Finish (seconds)', 'Q1', 'Q2', 'Q3', 'Q4',
       'Q5', 'Q6', 'Q7_Part_1', 'Q7_Part_2', 'Q7_Part_3', 'Q7_Part_4',
       'Q7_Part_5', 'Q7_Part_6', 'Q7_Part_7', 'Q7_Part_8', 'Q7_Part_9',
       'Q7_Part_10', 'Q7_Part_11', 'Q7_Part_12', 'Q7_OTHER', 'Q8',
       'Q9_Part_1', 'Q9_Part_2', 'Q9_Part_3', 'Q9_Part_4', 'Q9_Part_5',
       'Q9_Part_6', 'Q9_Part_7', 'Q9_Part_8', 'Q9_Part_9', 'Q9_Part_10',
       'Q9_Part_11', 'Q9_OTHER', 'Q10_Part_1', 'Q10_Part_2', 'Q10_Part_3',
       'Q10_Part_4', 'Q10_Part_5', 'Q10_Part_6', 'Q10_Part_7',
       'Q10_Part_8', 'Q10_Part_9', 'Q10_Part_10', 'Q10_Part_11',
       'Q10_Part_12', 'Q10_Part_13', 'Q10_OTHER', 'Q11', 'Q12_Part_1',
       'Q12_Part_2', 'Q12_Part_3', 'Q12_OTHER', 'Q13', 'Q14_Part_1',
       'Q14_Part_2', 'Q14_Part_3', 'Q14_Part_4', 'Q14_Part_5',
       'Q14_Part_6', 'Q14_Part_7', 'Q14_Part_8', 'Q14_Part_9',
       'Q14_Part_10', 'Q14_Part_11', 'Q14_OTHER', 'Q15', 'Q16_Part_1',
       'Q16_Part_2', 'Q16_Part_3', 'Q16_Part_4', 'Q16_Part_5',
       'Q16_Part_6', 'Q16_Part_7', 'Q16_Part_8', 'Q16_Part_9',
       'Q16_Part_10', 'Q16_Part_11', 'Q16_Part_12', 'Q16_Part_13',
       'Q16_Part_14', 'Q16_Part_15', 'Q16_OTHER', 'Q17_Part_1',
       'Q17_Part_2', 'Q17_Part_3', 'Q17_Part_4', 'Q17_Part_5',
       'Q17_Part_6', 'Q17_Part_7', 'Q17_Part_8', 'Q17_Part_9',
       'Q17_Part_10', 'Q17_Part_11', 'Q17_OTHER', 'Q18_Part_1',
       'Q18_Part_2', 'Q18_Part_3', 'Q18_Part_4', 'Q18_Part_5',
       'Q18_Part_6', 'Q18_OTHER', 'Q19_Part_1', 'Q19_Part_2',
       'Q19_Part_3', 'Q19_Part_4', 'Q19_Part_5', 'Q19_OTHER', 'Q20',
       'Q21', 'Q22', 'Q23_Part_1', 'Q23_Part_2', 'Q23_Part_3',
       'Q23_Part_4', 'Q23_Part_5', 'Q23_Part_6', 'Q23_Part_7',
       'Q23_OTHER', 'Q24', 'Q25', 'Q26_A_Part_1', 'Q26_A_Part_2',
       'Q26_A_Part_3', 'Q26_A_Part_4', 'Q26_A_Part_5', 'Q26_A_Part_6',
       'Q26_A_Part_7', 'Q26_A_Part_8', 'Q26_A_Part_9', 'Q26_A_Part_10',
       'Q26_A_Part_11', 'Q26_A_OTHER', 'Q27_A_Part_1', 'Q27_A_Part_2',
       'Q27_A_Part_3', 'Q27_A_Part_4', 'Q27_A_Part_5', 'Q27_A_Part_6',
       'Q27_A_Part_7', 'Q27_A_Part_8', 'Q27_A_Part_9', 'Q27_A_Part_10',
       'Q27_A_Part_11', 'Q27_A_OTHER', 'Q28_A_Part_1', 'Q28_A_Part_2',
       'Q28_A_Part_3', 'Q28_A_Part_4', 'Q28_A_Part_5', 'Q28_A_Part_6',
       'Q28_A_Part_7', 'Q28_A_Part_8', 'Q28_A_Part_9', 'Q28_A_Part_10',
       'Q28_A_OTHER', 'Q29_A_Part_1', 'Q29_A_Part_2', 'Q29_A_Part_3',
       'Q29_A_Part_4', 'Q29_A_Part_5', 'Q29_A_Part_6', 'Q29_A_Part_7',
       'Q29_A_Part_8', 'Q29_A_Part_9', 'Q29_A_Part_10', 'Q29_A_Part_11',
       'Q29_A_Part_12', 'Q29_A_Part_13', 'Q29_A_Part_14', 'Q29_A_Part_15',
       'Q29_A_Part_16', 'Q29_A_Part_17', 'Q29_A_OTHER', 'Q30',
       'Q31_A_Part_1', 'Q31_A_Part_2', 'Q31_A_Part_3', 'Q31_A_Part_4',
       'Q31_A_Part_5', 'Q31_A_Part_6', 'Q31_A_Part_7', 'Q31_A_Part_8',
       'Q31_A_Part_9', 'Q31_A_Part_10', 'Q31_A_Part_11', 'Q31_A_Part_12',
       'Q31_A_Part_13', 'Q31_A_Part_14', 'Q31_A_OTHER', 'Q32',
       'Q33_A_Part_1', 'Q33_A_Part_2', 'Q33_A_Part_3', 'Q33_A_Part_4',
       'Q33_A_Part_5', 'Q33_A_Part_6', 'Q33_A_Part_7', 'Q33_A_OTHER',
       'Q34_A_Part_1', 'Q34_A_Part_2', 'Q34_A_Part_3', 'Q34_A_Part_4',
       'Q34_A_Part_5', 'Q34_A_Part_6', 'Q34_A_Part_7', 'Q34_A_Part_8',
       'Q34_A_Part_9', 'Q34_A_Part_10', 'Q34_A_Part_11', 'Q34_A_OTHER',
       'Q35_A_Part_1', 'Q35_A_Part_2', 'Q35_A_Part_3', 'Q35_A_Part_4',
       'Q35_A_Part_5', 'Q35_A_Part_6', 'Q35_A_Part_7', 'Q35_A_Part_8',
       'Q35_A_Part_9', 'Q35_A_Part_10', 'Q35_A_OTHER', 'Q36_Part_1',
       'Q36_Part_2', 'Q36_Part_3', 'Q36_Part_4', 'Q36_Part_5',
       'Q36_Part_6', 'Q36_Part_7', 'Q36_Part_8', 'Q36_Part_9',
       'Q36_OTHER', 'Q37_Part_1', 'Q37_Part_2', 'Q37_Part_3',
       'Q37_Part_4', 'Q37_Part_5', 'Q37_Part_6', 'Q37_Part_7',
       'Q37_Part_8', 'Q37_Part_9', 'Q37_Part_10', 'Q37_Part_11',
```

```
'Q37_OTHER', 'Q38', 'Q39_Part_1', 'Q39_Part_2', 'Q39_Part_3',
'Q39_Part_4', 'Q39_Part_5', 'Q39_Part_6', 'Q39_Part_7',
'Q39_Part_8', 'Q39_Part_9', 'Q39_Part_10', 'Q39_Part_11',
'Q39_OTHER', 'Q26_B_Part_1', 'Q26_B_Part_2', 'Q26_B_Part_3',
'Q26_B_Part_4', 'Q26_B_Part_5', 'Q26_B_Part_6', 'Q26_B_Part_7',
'Q26_B_Part_8', 'Q26_B_Part_9', 'Q26_B_Part_10', 'Q26_B_Part_11',
'Q26_B_OTHER', 'Q27_B_Part_1', 'Q27_B_Part_2', 'Q27_B_Part_3',
'Q27_B_Part_4', 'Q27_B_Part_5', 'Q27_B_Part_6', 'Q27_B_Part_7',
'Q27_B_Part_8', 'Q27_B_Part_9', 'Q27_B_Part_10', 'Q27_B_Part_11',
'Q27_B_OTHER', 'Q28_B_Part_1', 'Q28_B_Part_2', 'Q28_B_Part_3',
'Q28_B_Part_4', 'Q28_B_Part_5', 'Q28_B_Part_6', 'Q28_B_Part_7',
'Q28_B_Part_8', 'Q28_B_Part_9', 'Q28_B_Part_10', 'Q28_B_OTHER',
'Q29_B_Part_1', 'Q29_B_Part_2', 'Q29_B_Part_3', 'Q29_B_Part_4',
'Q29_B_Part_5', 'Q29_B_Part_6', 'Q29_B_Part_7', 'Q29_B_Part_8',
'Q29_B_Part_9', 'Q29_B_Part_10', 'Q29_B_Part_11', 'Q29_B_Part_12',
'Q29_B_Part_13', 'Q29_B_Part_14', 'Q29_B_Part_15', 'Q29_B_Part_16',
'Q29_B_Part_17', 'Q29_B_OTHER', 'Q31_B_Part_1', 'Q31_B_Part_2',
'Q31_B_Part_3', 'Q31_B_Part_4', 'Q31_B_Part_5', 'Q31_B_Part_6',
'Q31_B_Part_7', 'Q31_B_Part_8', 'Q31_B_Part_9', 'Q31_B_Part_10',
'Q31_B_Part_11', 'Q31_B_Part_12', 'Q31_B_Part_13', 'Q31_B_Part_14',
'Q31_B_OTHER', 'Q33_B_Part_1', 'Q33_B_Part_2', 'Q33_B_Part_3',
'Q33_B_Part_4', 'Q33_B_Part_5', 'Q33_B_Part_6', 'Q33_B_Part_7',
'Q33_B_OTHER', 'Q34_B_Part_1', 'Q34_B_Part_2', 'Q34_B_Part_3',
'Q34_B_Part_4', 'Q34_B_Part_5', 'Q34_B_Part_6', 'Q34_B_Part_7',
'Q34_B_Part_8', 'Q34_B_Part_9', 'Q34_B_Part_10', 'Q34_B_Part_11',
'Q34_B_OTHER', 'Q35_B_Part_1', 'Q35_B_Part_2', 'Q35_B_Part_3',
'Q35_B_Part_4', 'Q35_B_Part_5', 'Q35_B_Part_6', 'Q35_B_Part_7',
'Q35_B_Part_8', 'Q35_B_Part_9', 'Q35_B_Part_10', 'Q35_B_OTHER',
'Q24_Encoded', 'Q24_buckets'], dtype=object)
```

In [7]:

```
kaggle_data.isnull().sum()
print(ms_perc(kaggle_data))
```

```
84.33299135124693
```

In the kaggle dataframe, there is 84.33% data missing.

It is seen that there are some columns with 'OTHER' option. So, it is irrelevant to keep those columns for the analysis as it does not give any specific information about the answers. These columns have free flow of text and have different key for every person. So, those columns with 'OTHER' text is dropped from the dataframe.

In [8]:
```
# dropping the columns with 'OTHER' as it does not make sense in the analysis.
other_col = [t for t in kaggle_data.columns if 'OTHER' in t]
kaggle_data = kaggle_data.drop(other_col, axis = 1)
kaggle_data.head()
```

Out[8]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2
0	Duration (in seconds)	What is your age (# years)?	What is your gender? - Selected Choice	In which country do you currently reside?	What is the highest level of formal education ...	Select the title most similar to your current ...	For how many years have you been writing code ...	What programming languages do you use on a reg...	What programming languages do you use on a reg...
1	289287	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	Python	R
2	860	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	NaN	NaN
3	507	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	Python	NaN
4	762	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	Python	NaN

5 rows × 328 columns

In [9]:
```
kaggle_data.columns.values
```

Out[9]:
```
array(['Time from Start to Finish (seconds)', 'Q1', 'Q2', 'Q3', 'Q4',
       'Q5', 'Q6', 'Q7_Part_1', 'Q7_Part_2', 'Q7_Part_3', 'Q7_Part_4',
       'Q7_Part_5', 'Q7_Part_6', 'Q7_Part_7', 'Q7_Part_8', 'Q7_Part_9',
       'Q7_Part_10', 'Q7_Part_11', 'Q7_Part_12', 'Q8', 'Q9_Part_1',
       'Q9_Part_2', 'Q9_Part_3', 'Q9_Part_4', 'Q9_Part_5', 'Q9_Part_6',
       'Q9_Part_7', 'Q9_Part_8', 'Q9_Part_9', 'Q9_Part_10', 'Q9_Part_11',
       'Q10_Part_1', 'Q10_Part_2', 'Q10_Part_3', 'Q10_Part_4',
       'Q10_Part_5', 'Q10_Part_6', 'Q10_Part_7', 'Q10_Part_8',
       'Q10_Part_9', 'Q10_Part_10', 'Q10_Part_11', 'Q10_Part_12',
       'Q10_Part_13', 'Q11', 'Q12_Part_1', 'Q12_Part_2', 'Q12_Part_3',
       'Q13', 'Q14_Part_1', 'Q14_Part_2', 'Q14_Part_3', 'Q14_Part_4',
       'Q14_Part_5', 'Q14_Part_6', 'Q14_Part_7', 'Q14_Part_8',
       'Q14_Part_9', 'Q14_Part_10', 'Q14_Part_11', 'Q15', 'Q16_Part_1',
       'Q16_Part_2', 'Q16_Part_3', 'Q16_Part_4', 'Q16_Part_5',
       'Q16_Part_6', 'Q16_Part_7', 'Q16_Part_8', 'Q16_Part_9',
       'Q16_Part_10', 'Q16_Part_11', 'Q16_Part_12', 'Q16_Part_13',
       'Q16_Part_14', 'Q16_Part_15', 'Q17_Part_1', 'Q17_Part_2',
       'Q17_Part_3', 'Q17_Part_4', 'Q17_Part_5', 'Q17_Part_6',
       'Q17_Part_7', 'Q17_Part_8', 'Q17_Part_9', 'Q17_Part_10',
       'Q17_Part_11', 'Q18_Part_1', 'Q18_Part_2', 'Q18_Part_3',
       'Q18_Part_4', 'Q18_Part_5', 'Q18_Part_6', 'Q19_Part_1',
       'Q19_Part_2', 'Q19_Part_3', 'Q19_Part_4', 'Q19_Part_5', 'Q20',
       'Q21', 'Q22', 'Q23_Part_1', 'Q23_Part_2', 'Q23_Part_3',
       'Q23_Part_4', 'Q23_Part_5', 'Q23_Part_6', 'Q23_Part_7', 'Q24',
       'Q25', 'Q26_A_Part_1', 'Q26_A_Part_2', 'Q26_A_Part_3',
       'Q26_A_Part_4', 'Q26_A_Part_5', 'Q26_A_Part_6', 'Q26_A_Part_7',
       'Q26_A_Part_8', 'Q26_A_Part_9', 'Q26_A_Part_10', 'Q26_A_Part_11',
       'Q27_A_Part_1', 'Q27_A_Part_2', 'Q27_A_Part_3', 'Q27_A_Part_4',
       'Q27_A_Part_5', 'Q27_A_Part_6', 'Q27_A_Part_7', 'Q27_A_Part_8',
       'Q27_A_Part_9', 'Q27_A_Part_10', 'Q27_A_Part_11', 'Q28_A_Part_1',
       'Q28_A_Part_2', 'Q28_A_Part_3', 'Q28_A_Part_4', 'Q28_A_Part_5',
       'Q28_A_Part_6', 'Q28_A_Part_7', 'Q28_A_Part_8', 'Q28_A_Part_9',
       'Q28_A_Part_10', 'Q29_A_Part_1', 'Q29_A_Part_2', 'Q29_A_Part_3',
       'Q29_A_Part_4', 'Q29_A_Part_5', 'Q29_A_Part_6', 'Q29_A_Part_7',
       'Q29_A_Part_8', 'Q29_A_Part_9', 'Q29_A_Part_10', 'Q29_A_Part_11',
       'Q29_A_Part_12', 'Q29_A_Part_13', 'Q29_A_Part_14', 'Q29_A_Part_15',
       'Q29_A_Part_16', 'Q29_A_Part_17', 'Q30', 'Q31_A_Part_1',
       'Q31_A_Part_2', 'Q31_A_Part_3', 'Q31_A_Part_4', 'Q31_A_Part_5',
       'Q31_A_Part_6', 'Q31_A_Part_7', 'Q31_A_Part_8', 'Q31_A_Part_9',
       'Q31_A_Part_10', 'Q31_A_Part_11', 'Q31_A_Part_12', 'Q31_A_Part_13',
       'Q31_A_Part_14', 'Q32', 'Q33_A_Part_1', 'Q33_A_Part_2',
       'Q33_A_Part_3', 'Q33_A_Part_4', 'Q33_A_Part_5', 'Q33_A_Part_6',
       'Q33_A_Part_7', 'Q34_A_Part_1', 'Q34_A_Part_2', 'Q34_A_Part_3',
       'Q34_A_Part_4', 'Q34_A_Part_5', 'Q34_A_Part_6', 'Q34_A_Part_7',
       'Q34_A_Part_8', 'Q34_A_Part_9', 'Q34_A_Part_10', 'Q34_A_Part_11',
       'Q35_A_Part_1', 'Q35_A_Part_2', 'Q35_A_Part_3', 'Q35_A_Part_4',
       'Q35_A_Part_5', 'Q35_A_Part_6', 'Q35_A_Part_7', 'Q35_A_Part_8',
       'Q35_A_Part_9', 'Q35_A_Part_10', 'Q36_Part_1', 'Q36_Part_2',
       'Q36_Part_3', 'Q36_Part_4', 'Q36_Part_5', 'Q36_Part_6',
       'Q36_Part_7', 'Q36_Part_8', 'Q36_Part_9', 'Q37_Part_1',
       'Q37_Part_2', 'Q37_Part_3', 'Q37_Part_4', 'Q37_Part_5',
       'Q37_Part_6', 'Q37_Part_7', 'Q37_Part_8', 'Q37_Part_9',
       'Q37_Part_10', 'Q37_Part_11', 'Q38', 'Q39_Part_1', 'Q39_Part_2',
       'Q39_Part_3', 'Q39_Part_4', 'Q39_Part_5', 'Q39_Part_6',
       'Q39_Part_7', 'Q39_Part_8', 'Q39_Part_9', 'Q39_Part_10',
       'Q39_Part_11', 'Q26_B_Part_1', 'Q26_B_Part_2', 'Q26_B_Part_3',
       'Q26_B_Part_4', 'Q26_B_Part_5', 'Q26_B_Part_6', 'Q26_B_Part_7',
```

```
       'Q26_B_Part_8', 'Q26_B_Part_9', 'Q26_B_Part_10', 'Q26_B_Part_11',
       'Q27_B_Part_1', 'Q27_B_Part_2', 'Q27_B_Part_3', 'Q27_B_Part_4',
       'Q27_B_Part_5', 'Q27_B_Part_6', 'Q27_B_Part_7', 'Q27_B_Part_8',
       'Q27_B_Part_9', 'Q27_B_Part_10', 'Q27_B_Part_11', 'Q28_B_Part_1',
       'Q28_B_Part_2', 'Q28_B_Part_3', 'Q28_B_Part_4', 'Q28_B_Part_5',
       'Q28_B_Part_6', 'Q28_B_Part_7', 'Q28_B_Part_8', 'Q28_B_Part_9',
       'Q28_B_Part_10', 'Q29_B_Part_1', 'Q29_B_Part_2', 'Q29_B_Part_3',
       'Q29_B_Part_4', 'Q29_B_Part_5', 'Q29_B_Part_6', 'Q29_B_Part_7',
       'Q29_B_Part_8', 'Q29_B_Part_9', 'Q29_B_Part_10', 'Q29_B_Part_11',
       'Q29_B_Part_12', 'Q29_B_Part_13', 'Q29_B_Part_14', 'Q29_B_Part_15',
       'Q29_B_Part_16', 'Q29_B_Part_17', 'Q31_B_Part_1', 'Q31_B_Part_2',
       'Q31_B_Part_3', 'Q31_B_Part_4', 'Q31_B_Part_5', 'Q31_B_Part_6',
       'Q31_B_Part_7', 'Q31_B_Part_8', 'Q31_B_Part_9', 'Q31_B_Part_10',
       'Q31_B_Part_11', 'Q31_B_Part_12', 'Q31_B_Part_13', 'Q31_B_Part_14',
       'Q33_B_Part_1', 'Q33_B_Part_2', 'Q33_B_Part_3', 'Q33_B_Part_4',
       'Q33_B_Part_5', 'Q33_B_Part_6', 'Q33_B_Part_7', 'Q34_B_Part_1',
       'Q34_B_Part_2', 'Q34_B_Part_3', 'Q34_B_Part_4', 'Q34_B_Part_5',
       'Q34_B_Part_6', 'Q34_B_Part_7', 'Q34_B_Part_8', 'Q34_B_Part_9',
       'Q34_B_Part_10', 'Q34_B_Part_11', 'Q35_B_Part_1', 'Q35_B_Part_2',
       'Q35_B_Part_3', 'Q35_B_Part_4', 'Q35_B_Part_5', 'Q35_B_Part_6',
       'Q35_B_Part_7', 'Q35_B_Part_8', 'Q35_B_Part_9', 'Q35_B_Part_10',
       'Q24_Encoded', 'Q24_buckets'], dtype=object)
```

Also, the first row is with the questions only. So, dropping the row with the question.

In [10]:
```python
print(kaggle_data.iloc[0])
```

```
Time from Start to Finish (seconds)                                Duration
(in seconds)
Q1                                                           What is your age
(# years)?
Q2                                               What is your gender? - Sele
cted Choice
Q3                                            In which country do you curren
tly reside?
Q4                                       What is the highest level of formal ed
ucation ...
                                                                 ...
Q35_B_Part_8                             In the next 2 years, do you hope to be
come mor...
Q35_B_Part_9                             In the next 2 years, do you hope to be
come mor...
Q35_B_Part_10                            In the next 2 years, do you hope to be
come mor...
Q24_Encoded
NaN
Q24_buckets
NaN
Name: 0, Length: 328, dtype: object
```

In [11]:
```
kaggle_data[kaggle_data['Q24_Encoded'].isna()]
```

Out[11]:

	Time from Start to Finish (seconds)	Q1	Q2	Q3	Q4	Q5	Q6	Q7_Part_1	Q7_Part_2	
0	Duration (in seconds)	What is your age (# years)?	What is your gender? - Selected Choice	In which country do you currently reside?	What is the highest level of formal education ...	Select the title most similar to your current ...	For how many years have you been writing code ...	What programming languages do you use on a reg...	What programming languages do you use on a reg...	pr

1 rows × 328 columns

In [12]:
```
kaggle_data = kaggle_data.dropna(subset=['Q24_Encoded'])
```

In [13]:
```
missing_val = kaggle_data.isnull().sum().sort_values(ascending=False)
print(missing_val)
```

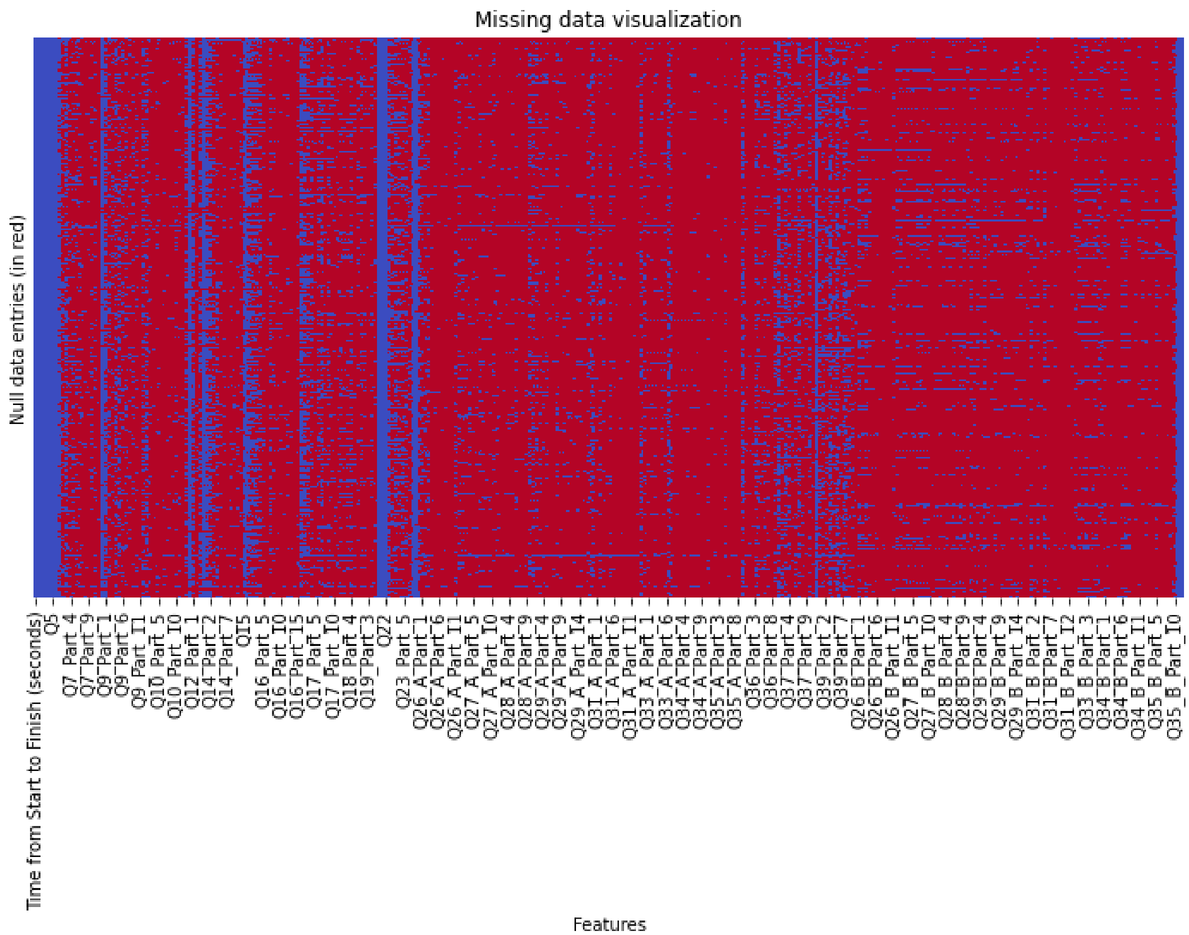
```
Q34_A_Part_9                       10685
Q31_A_Part_9                       10673
Q35_A_Part_4                       10672
Q31_A_Part_12                      10668
Q10_Part_6                         10665
                                    ...
Q21                                    0
Q22                                    0
Q24                                    0
Q24_Encoded                            0
Time from Start to Finish (seconds)    0
Length: 328, dtype: int64
```

The above array shows the number of missing value counts according to the columns. It is still required to clean the data.

Visualizing the missing data

In [14]:
```
fig, ax = plt.subplots(figsize=(12,6))
ax = sns.heatmap(kaggle_data.isnull(), cmap='coolwarm', yticklabels=False, cba
r=False, ax=ax)
ax.set_xlabel('Features')
ax.set_ylabel('Null data entries (in red)')
ax.set_title('Missing data visualization')
```

Out[14]:
```
Text(0.5, 1.0, 'Missing data visualization')
```



There are so many values missing and it is evident from the sns heatmap. The red color shows the missing values.

Inspecting all the question to sort out the irrelevant features

The following questions are irrelevant with the analysis and are dropped.

1. The first column "Time from start to finish" is irrelevant with the Salary.
2. Q8 - What programming language you recommend to learn first - This opinion is very subjective and the Q7 already provides the reasonable question about what language you use on regular basis.
3. Q39 - Media source on which they report data science does not have impact on salary

In [15]:
```
# removing those columns

Q8 = [t for t in kaggle_data.columns if 'Q8' in t]
Q39 = [t for t in kaggle_data.columns if 'Q39' in t]
drop_list = Q8 + Q39
drop_list.append('Time from Start to Finish (seconds)')
kaggle_data = kaggle_data.drop(drop_list, axis = 1)
```

In the dataframe, most of the questions are of categorical form having multiple choise questions and contains 'Part' in the column name. So, the columns having 'Part' in its name are one hot encoded.

There are two ways to encode the variables: Label encoding and One hot encoding. Both are explained over here.

Label Encoding: This method converts each value in the column to a number. If the value in the column repeats then that value will be replaced with the same number. If there are less variation of categorical values, then the label encoding will work best. The columns like, Age, education, profession can be label encoded very well.

One Hot Encoding: A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.
https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
(https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/)

Performing one hot encoding using the pandas get_dummies function to transform the categorical variables into the binary form.
https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd
(https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd)

In [16]:
```
cat_ques = [t for t in kaggle_data.columns if 'Part' in t]
print(cat_ques)
```

```
['Q7_Part_1', 'Q7_Part_2', 'Q7_Part_3', 'Q7_Part_4', 'Q7_Part_5', 'Q7_Part_
6', 'Q7_Part_7', 'Q7_Part_8', 'Q7_Part_9', 'Q7_Part_10', 'Q7_Part_11', 'Q7_Pa
rt_12', 'Q9_Part_1', 'Q9_Part_2', 'Q9_Part_3', 'Q9_Part_4', 'Q9_Part_5', 'Q9_
Part_6', 'Q9_Part_7', 'Q9_Part_8', 'Q9_Part_9', 'Q9_Part_10', 'Q9_Part_11',
'Q10_Part_1', 'Q10_Part_2', 'Q10_Part_3', 'Q10_Part_4', 'Q10_Part_5', 'Q10_Pa
rt_6', 'Q10_Part_7', 'Q10_Part_8', 'Q10_Part_9', 'Q10_Part_10', 'Q10_Part_1
1', 'Q10_Part_12', 'Q10_Part_13', 'Q12_Part_1', 'Q12_Part_2', 'Q12_Part_3',
'Q14_Part_1', 'Q14_Part_2', 'Q14_Part_3', 'Q14_Part_4', 'Q14_Part_5', 'Q14_Pa
rt_6', 'Q14_Part_7', 'Q14_Part_8', 'Q14_Part_9', 'Q14_Part_10', 'Q14_Part_1
1', 'Q16_Part_1', 'Q16_Part_2', 'Q16_Part_3', 'Q16_Part_4', 'Q16_Part_5', 'Q1
6_Part_6', 'Q16_Part_7', 'Q16_Part_8', 'Q16_Part_9', 'Q16_Part_10', 'Q16_Part
_11', 'Q16_Part_12', 'Q16_Part_13', 'Q16_Part_14', 'Q16_Part_15', 'Q17_Part_
1', 'Q17_Part_2', 'Q17_Part_3', 'Q17_Part_4', 'Q17_Part_5', 'Q17_Part_6', 'Q1
7_Part_7', 'Q17_Part_8', 'Q17_Part_9', 'Q17_Part_10', 'Q17_Part_11', 'Q18_Par
t_1', 'Q18_Part_2', 'Q18_Part_3', 'Q18_Part_4', 'Q18_Part_5', 'Q18_Part_6',
'Q19_Part_1', 'Q19_Part_2', 'Q19_Part_3', 'Q19_Part_4', 'Q19_Part_5', 'Q23_Pa
rt_1', 'Q23_Part_2', 'Q23_Part_3', 'Q23_Part_4', 'Q23_Part_5', 'Q23_Part_6',
'Q23_Part_7', 'Q26_A_Part_1', 'Q26_A_Part_2', 'Q26_A_Part_3', 'Q26_A_Part_4',
'Q26_A_Part_5', 'Q26_A_Part_6', 'Q26_A_Part_7', 'Q26_A_Part_8', 'Q26_A_Part_
9', 'Q26_A_Part_10', 'Q26_A_Part_11', 'Q27_A_Part_1', 'Q27_A_Part_2', 'Q27_A_
Part_3', 'Q27_A_Part_4', 'Q27_A_Part_5', 'Q27_A_Part_6', 'Q27_A_Part_7', 'Q27
_A_Part_8', 'Q27_A_Part_9', 'Q27_A_Part_10', 'Q27_A_Part_11', 'Q28_A_Part_1',
'Q28_A_Part_2', 'Q28_A_Part_3', 'Q28_A_Part_4', 'Q28_A_Part_5', 'Q28_A_Part_
6', 'Q28_A_Part_7', 'Q28_A_Part_8', 'Q28_A_Part_9', 'Q28_A_Part_10', 'Q29_A_P
art_1', 'Q29_A_Part_2', 'Q29_A_Part_3', 'Q29_A_Part_4', 'Q29_A_Part_5', 'Q29_
A_Part_6', 'Q29_A_Part_7', 'Q29_A_Part_8', 'Q29_A_Part_9', 'Q29_A_Part_10',
'Q29_A_Part_11', 'Q29_A_Part_12', 'Q29_A_Part_13', 'Q29_A_Part_14', 'Q29_A_Pa
rt_15', 'Q29_A_Part_16', 'Q29_A_Part_17', 'Q31_A_Part_1', 'Q31_A_Part_2', 'Q3
1_A_Part_3', 'Q31_A_Part_4', 'Q31_A_Part_5', 'Q31_A_Part_6', 'Q31_A_Part_7',
'Q31_A_Part_8', 'Q31_A_Part_9', 'Q31_A_Part_10', 'Q31_A_Part_11', 'Q31_A_Part
_12', 'Q31_A_Part_13', 'Q31_A_Part_14', 'Q33_A_Part_1', 'Q33_A_Part_2', 'Q33_
A_Part_3', 'Q33_A_Part_4', 'Q33_A_Part_5', 'Q33_A_Part_6', 'Q33_A_Part_7', 'Q
34_A_Part_1', 'Q34_A_Part_2', 'Q34_A_Part_3', 'Q34_A_Part_4', 'Q34_A_Part_5',
'Q34_A_Part_6', 'Q34_A_Part_7', 'Q34_A_Part_8', 'Q34_A_Part_9', 'Q34_A_Part_1
0', 'Q34_A_Part_11', 'Q35_A_Part_1', 'Q35_A_Part_2', 'Q35_A_Part_3', 'Q35_A_P
art_4', 'Q35_A_Part_5', 'Q35_A_Part_6', 'Q35_A_Part_7', 'Q35_A_Part_8', 'Q35_
A_Part_9', 'Q35_A_Part_10', 'Q36_Part_1', 'Q36_Part_2', 'Q36_Part_3', 'Q36_Pa
rt_4', 'Q36_Part_5', 'Q36_Part_6', 'Q36_Part_7', 'Q36_Part_8', 'Q36_Part_9',
'Q37_Part_1', 'Q37_Part_2', 'Q37_Part_3', 'Q37_Part_4', 'Q37_Part_5', 'Q37_Pa
rt_6', 'Q37_Part_7', 'Q37_Part_8', 'Q37_Part_9', 'Q37_Part_10', 'Q37_Part_1
1', 'Q26_B_Part_1', 'Q26_B_Part_2', 'Q26_B_Part_3', 'Q26_B_Part_4', 'Q26_B_Pa
rt_5', 'Q26_B_Part_6', 'Q26_B_Part_7', 'Q26_B_Part_8', 'Q26_B_Part_9', 'Q26_B
_Part_10', 'Q26_B_Part_11', 'Q27_B_Part_1', 'Q27_B_Part_2', 'Q27_B_Part_3',
'Q27_B_Part_4', 'Q27_B_Part_5', 'Q27_B_Part_6', 'Q27_B_Part_7', 'Q27_B_Part_
8', 'Q27_B_Part_9', 'Q27_B_Part_10', 'Q27_B_Part_11', 'Q28_B_Part_1', 'Q28_B_
Part_2', 'Q28_B_Part_3', 'Q28_B_Part_4', 'Q28_B_Part_5', 'Q28_B_Part_6', 'Q28
_B_Part_7', 'Q28_B_Part_8', 'Q28_B_Part_9', 'Q28_B_Part_10', 'Q29_B_Part_1',
'Q29_B_Part_2', 'Q29_B_Part_3', 'Q29_B_Part_4', 'Q29_B_Part_5', 'Q29_B_Part_
6', 'Q29_B_Part_7', 'Q29_B_Part_8', 'Q29_B_Part_9', 'Q29_B_Part_10', 'Q29_B_P
art_11', 'Q29_B_Part_12', 'Q29_B_Part_13', 'Q29_B_Part_14', 'Q29_B_Part_15',
'Q29_B_Part_16', 'Q29_B_Part_17', 'Q31_B_Part_1', 'Q31_B_Part_2', 'Q31_B_Part
_3', 'Q31_B_Part_4', 'Q31_B_Part_5', 'Q31_B_Part_6', 'Q31_B_Part_7', 'Q31_B_P
art_8', 'Q31_B_Part_9', 'Q31_B_Part_10', 'Q31_B_Part_11', 'Q31_B_Part_12', 'Q
31_B_Part_13', 'Q31_B_Part_14', 'Q33_B_Part_1', 'Q33_B_Part_2', 'Q33_B_Part_
3', 'Q33_B_Part_4', 'Q33_B_Part_5', 'Q33_B_Part_6', 'Q33_B_Part_7', 'Q34_B_Pa
rt_1', 'Q34_B_Part_2', 'Q34_B_Part_3', 'Q34_B_Part_4', 'Q34_B_Part_5', 'Q34_B
_Part_6', 'Q34_B_Part_7', 'Q34_B_Part_8', 'Q34_B_Part_9', 'Q34_B_Part_10', 'Q
```

```
34_B_Part_11', 'Q35_B_Part_1', 'Q35_B_Part_2', 'Q35_B_Part_3', 'Q35_B_Part_
4', 'Q35_B_Part_5', 'Q35_B_Part_6', 'Q35_B_Part_7', 'Q35_B_Part_8', 'Q35_B_Pa
rt_9', 'Q35_B_Part_10']
```

In [17]:
```
cat_encoded = pd.get_dummies(kaggle_data, columns=cat_ques)
cat_encoded
```

Out[17]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q11	Q13	Q15	Q20	.
1	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	A personal computer or laptop	2-5 times	1-2 years	10,000 or more employees	.
2	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	1000-9,999 employees	.
3	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	250-999 employees	.
4	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	1000-9,999 employees	.
5	35-39	Man	United States of America	Doctoral degree	Research Scientist	1-2 years	A personal computer or laptop	Never	Under 1 year	0-49 employees	.
...	...	...	...	...	...	...	...	...	...	...	.
10725	35-39	Man	Malaysia	I prefer not to answer	Machine Learning Engineer	1-2 years	A personal computer or laptop	Never	1-2 years	0-49 employees	.
10726	35-39	Man	Thailand	Bachelor's degree	Other	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	250-999 employees	.
10727	30-34	Man	Brazil	Master's degree	Research Scientist	< 1 years	A personal computer or laptop	Never	I do not use machine learning methods	0-49 employees	.
10728	22-24	Man	India	Bachelor's degree	Software Engineer	3-5 years	A cloud computing platform (AWS, Azure, GCP, h...	More than 25 times	1-2 years	10,000 or more employees	.
10729	22-24	Man	Pakistan	Master's degree	Machine Learning Engineer	< 1 years	A cloud computing platform (AWS, Azure, GCP, h...	Once	Under 1 year	0-49 employees	.

10729 rows × 315 columns

In [18]:
```
print(ms_perc(cat_encoded))
```

```
0.5782281222676413
```

It is observed that only 0.57% data is missing from the dataframe. As most of the columns contain 'Part' and those all are one hot encoded, the missing value percentage reduces less than one.

In [19]:
```
missing_val = cat_encoded.isnull().sum().sort_values(ascending=False)
print(missing_val[0:20])
```

```
Q32
9231
Q30
7216
Q38
1253
Q11
561
Q13
561
Q15
561
Q25
159
Q35_B_Part_10_None
0
Q19_Part_2_Encoder-decorder models (seq2seq, vanilla transformers)
0
Q19_Part_3_Contextualized embeddings (ELMo, CoVe)
0
Q19_Part_4_Transformer language models (GPT-3, BERT, XLnet, etc)
0
Q19_Part_5_None
0
Q23_Part_1_Analyze and understand data to influence product or business decis
ions                                                0
Q23_Part_2_Build and/or run the data infrastructure that my business uses for
storing, analyzing, and operationalizing data       0
Q23_Part_4_Build and/or run a machine learning service that operationally imp
roves my product or workflows                       0
Q23_Part_3_Build prototypes to explore applying machine learning to new areas
0
Q18_Part_6_None
0
Q23_Part_5_Experimentation and iteration to improve existing ML models
0
Q23_Part_6_Do research that advances the state of the art of machine learning
0
Q23_Part_7_None of these activities are an important part of my role at work
0
dtype: int64
```

Moreover, the column with the None entity (i.e, that includes the 'None') answer are also dropped to clean out the data.

In [20]:
```
none_col = [t for t in cat_encoded.columns if 'None' in t]
kaggle_encoded = cat_encoded.drop(none_col, axis=1)
kaggle_encoded
```

Out[20]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q11	Q13	Q15	Q20	.
1	30-34	Man	United States of America	Master’s degree	Data Engineer	5-10 years	A personal computer or laptop	2-5 times	1-2 years	10,000 or more employees	.
2	35-39	Man	Argentina	Bachelor’s degree	Software Engineer	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	1000-9,999 employees	.
3	30-34	Man	United States of America	Master’s degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	250-999 employees	.
4	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	1000-9,999 employees	.
5	35-39	Man	United States of America	Doctoral degree	Research Scientist	1-2 years	A personal computer or laptop	Never	Under 1 year	0-49 employees	.
...	...	...	...	...	...	...	...	...	...	...	.
10725	35-39	Man	Malaysia	I prefer not to answer	Machine Learning Engineer	1-2 years	A personal computer or laptop	Never	1-2 years	0-49 employees	.
10726	35-39	Man	Thailand	Bachelor’s degree	Other	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	250-999 employees	.
10727	30-34	Man	Brazil	Master’s degree	Research Scientist	< 1 years	A personal computer or laptop	Never	I do not use machine learning methods	0-49 employees	.
10728	22-24	Man	India	Bachelor’s degree	Software Engineer	3-5 years	A cloud computing platform (AWS, Azure, GCP, h...	More than 25 times	1-2 years	10,000 or more employees	.
10729	22-24	Man	Pakistan	Master’s degree	Machine Learning Engineer	< 1 years	A cloud computing platform (AWS, Azure, GCP, h...	Once	Under 1 year	0-49 employees	.

10729 rows × 288 columns

In [21]:
```
missing_val = kaggle_encoded.isnull().sum().sort_values(ascending=False)
print(missing_val[0:20])
```

```
Q32
9231
Q30
7216
Q38
1253
Q11
561
Q13
561
Q15
561
Q25
159
Q35_B_Part_9_ Domino Model Monitor
0
Q19_Part_2_Encoder-decorder models (seq2seq, vanilla transformers)
0
Q19_Part_3_Contextualized embeddings (ELMo, CoVe)
0
Q19_Part_4_Transformer language models (GPT-3, BERT, XLnet, etc)
0
Q23_Part_1_Analyze and understand data to influence product or business decis
ions                                                  0
Q23_Part_2_Build and/or run the data infrastructure that my business uses for
storing, analyzing, and operationalizing data       0
Q23_Part_4_Build and/or run a machine learning service that operationally imp
roves my product or workflows                         0
Q23_Part_3_Build prototypes to explore applying machine learning to new areas
0
Q18_Part_5_Generative Networks (GAN, VAE, etc)
0
Q23_Part_5_Experimentation and iteration to improve existing ML models
0
Q23_Part_6_Do research that advances the state of the art of machine learning
0
Q26_A_Part_1_ Amazon Web Services (AWS)
0
Q26_A_Part_2_ Microsoft Azure
0
dtype: int64
```

In [22]:
```
x,y = kaggle_encoded.shape
print('percentage missingness of column Q32 is ', (missing_val['Q32']/x)*100)
print('percentage missingness of column Q30 is ', (missing_val['Q30']/x)*100)
print('percentage missingness of column Q38 is ', (missing_val['Q38']/x)*100)
print('percentage missingness of column Q11 is ', (missing_val['Q11']/x)*100)
print('percentage missingness of column Q13 is ', (missing_val['Q13']/x)*100)
print('percentage missingness of column Q15 is ', (missing_val['Q15']/x)*100)
print('percentage missingness of column Q25 is ', (missing_val['Q25']/x)*100)
```

```
percentage missingness of column Q32 is  86.03784136452605
percentage missingness of column Q30 is  67.25696709851803
percentage missingness of column Q38 is  11.678628017522602
percentage missingness of column Q11 is  5.22881908845186
percentage missingness of column Q13 is  5.22881908845186
percentage missingness of column Q15 is  5.22881908845186
percentage missingness of column Q25 is  1.4819647683847517
```

The columns 'Q32', 'Q30', 'Q38', 'Q11', 'Q13', 'Q15', 'Q25' have the missing values and need to be analysed. The columns 'Q32' and 'Q30' have more than 50% missing values and so they are dropped. The rest of the columns are imputed with the mean or median as required.

In [23]:
```
kaggle_encoded = kaggle_encoded.drop(['Q32', 'Q30'], axis=1)
kaggle_encoded.isnull().sum().sort_values(ascending=False)
```

Out[23]:
```
Q38                                  1253
Q11                                   561
Q13                                   561
Q15                                   561
Q25                                   159
                                     ...
Q36_Part_1_ Plotly Dash                 0
Q35_A_Part_9_ Domino Model Monitor      0
Q35_A_Part_8_ Trains                    0
Q35_A_Part_7_ Polyaxon                  0
Q1                                      0
Length: 286, dtype: int64
```

In [24]:
```
kaggle_original[['Q11','Q13','Q15','Q25','Q38']]
```

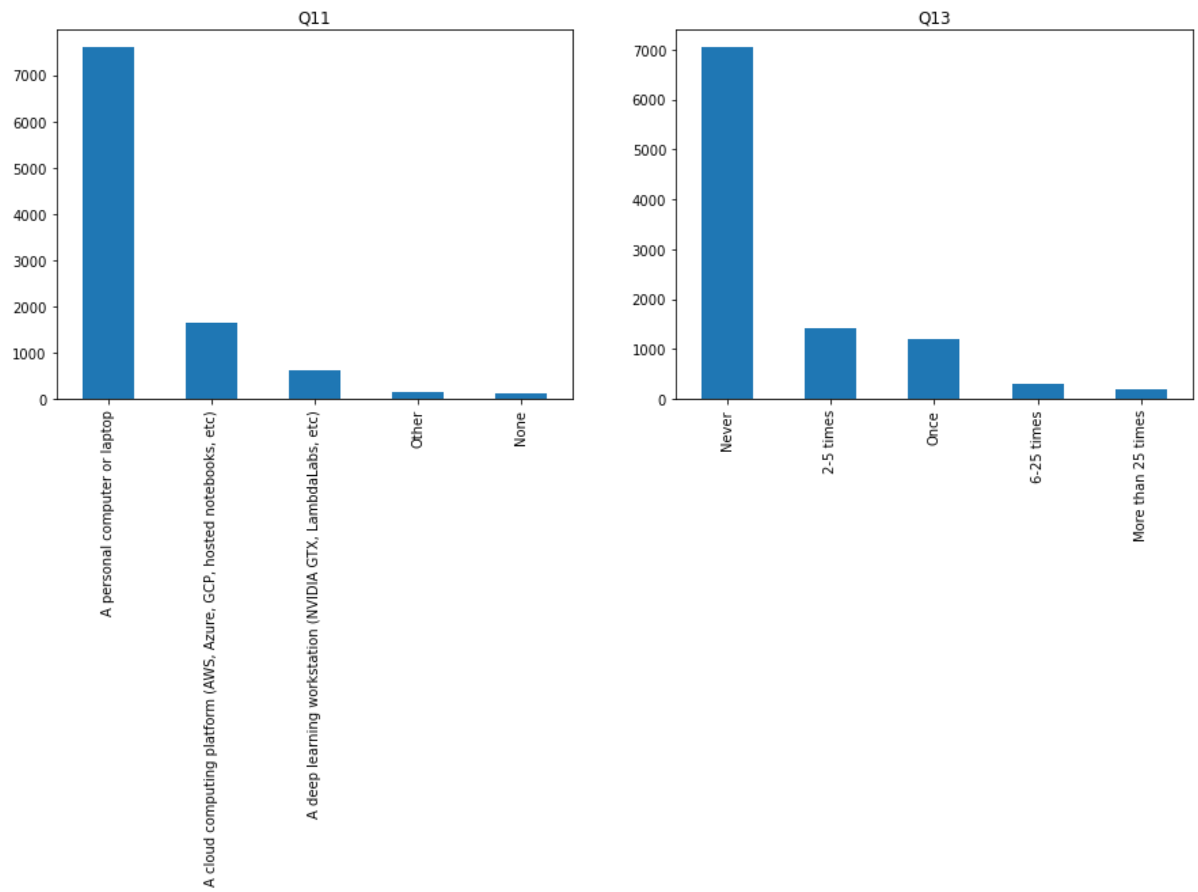
Out[24]:

	Q11	Q13	Q15	Q25	Q38
0	What type of computing platform do you use mos...	Approximately how many times have you used a T...	For how many years have you used machine learn...	Approximately how much money have you (or your...	What is the primary tool that you use at work ...
1	A personal computer or laptop	2-5 times	1-2 years	$100,000 or more($ USD)	Business intelligence software (Salesforce, Ta...
2	A personal computer or laptop	Never	I do not use machine learning methods	$0($ USD)	Basic statistical software (Microsoft Excel, G...
3	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	$10,000-$ 99,999	Local development environments (RStudio, Jupyt...
4	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	$10,000-$ 99,999	Cloud-based data software & APIs (AWS, GCP, Az...
...	...	...	...	...	...
10725	A personal computer or laptop	Never	1-2 years	$0($ USD)	Basic statistical software (Microsoft Excel, G...
10726	A personal computer or laptop	Never	I do not use machine learning methods	$0($ USD)	Local development environments (RStudio, Jupyt...
10727	A personal computer or laptop	Never	I do not use machine learning methods	$0($ USD)	NaN
10728	A cloud computing platform (AWS, Azure, GCP, h...	More than 25 times	1-2 years	$0($ USD)	Local development environments (RStudio, Jupyt...
10729	A cloud computing platform (AWS, Azure, GCP, h...	Once	Under 1 year	$0($ USD)	Local development environments (RStudio, Jupyt...

10730 rows × 5 columns

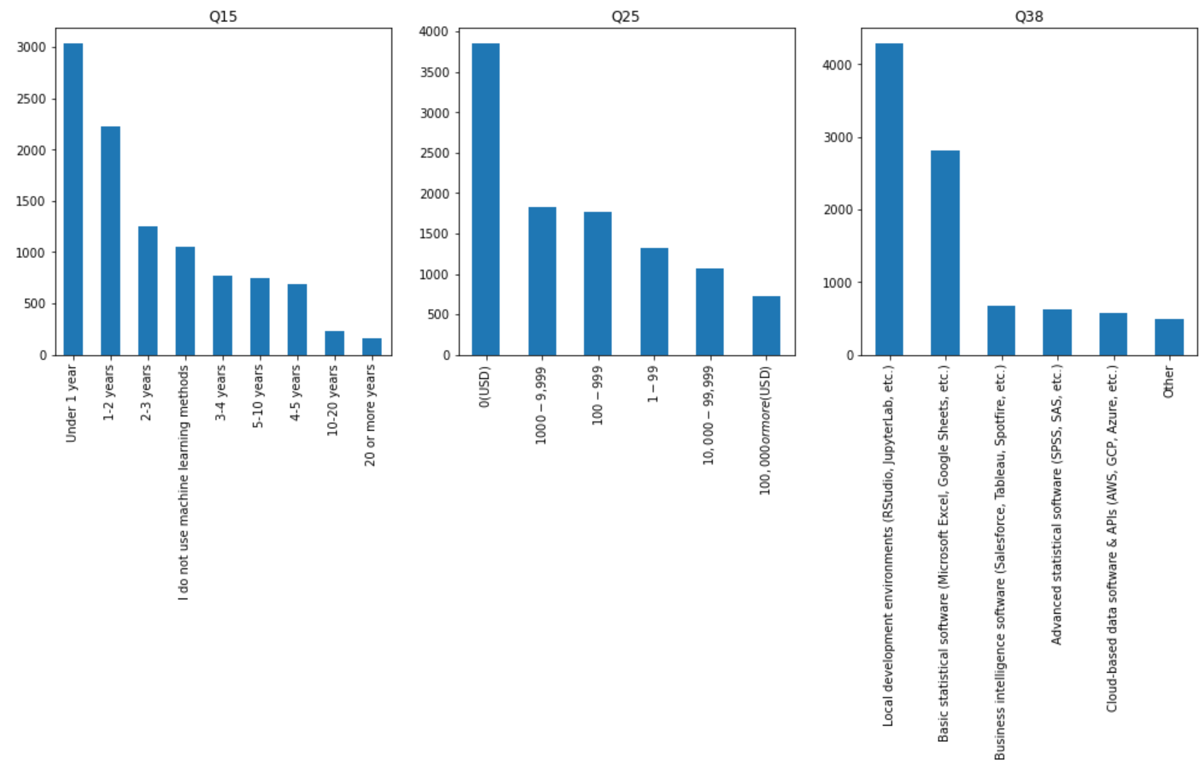
In [25]:
```
q11 = kaggle_encoded['Q11'].value_counts()
q13 = kaggle_encoded['Q13'].value_counts()
q15 = kaggle_encoded['Q15'].value_counts()
q25 = kaggle_encoded['Q25'].value_counts()
q38 = kaggle_encoded['Q38'].value_counts()

plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
q11.plot.bar()
plt.title('Q11')

plt.subplot(1,2,2)
q13.plot.bar()
plt.title('Q13')
```

Out[25]:
```
Text(0.5, 1.0, 'Q13')
```



In [26]:
```
plt.figure(figsize=(17,5))

plt.subplot(1,3,1)
q15.plot.bar()
plt.title('Q15')

plt.subplot(1,3,2)
q25.plot.bar()
plt.title('Q25')

plt.subplot(1,3,3)
q38.plot.bar()
plt.title('Q38')
```

Out[26]: Text(0.5, 1.0, 'Q38')



All the plots are skewed. So, to use mean for imputation does not seems reasonable. Therefore, the mode is used to impute all the columns.

In [27]:
```
def kag_mode(df):
    df.fillna(df.mode()[0],inplace=True)
kag_mode(kaggle_encoded['Q11'])
kag_mode(kaggle_encoded['Q13'])
kag_mode(kaggle_encoded['Q15'])
kag_mode(kaggle_encoded['Q25'])
kag_mode(kaggle_encoded['Q38'])
```

In [28]:
```
kaggle_encoded.isnull().sum().sort_values(ascending=False)
```

Out[28]:
```
Q35_B_Part_9_ Domino Model Monitor
0
Q23_Part_3_Build prototypes to explore applying machine learning to new areas
0
Q19_Part_1_Word embeddings/vectors (GLoVe, fastText, word2vec)
0
Q19_Part_2_Encoder-decorder models (seq2seq, vanilla transformers)
0
Q19_Part_3_Contextualized embeddings (ELMo, CoVe)
0

..
Q36_Part_1_ Plotly Dash
0
Q35_A_Part_9_ Domino Model Monitor
0
Q35_A_Part_8_ Trains
0
Q35_A_Part_7_ Polyaxon
0
Q1
0
Length: 286, dtype: int64
```

In [29]:
```
ms_perc(kaggle_encoded)
```

Out[29]:
```
0.0
```

In [30]:
```
kaggle_encoded
```

Out[30]:

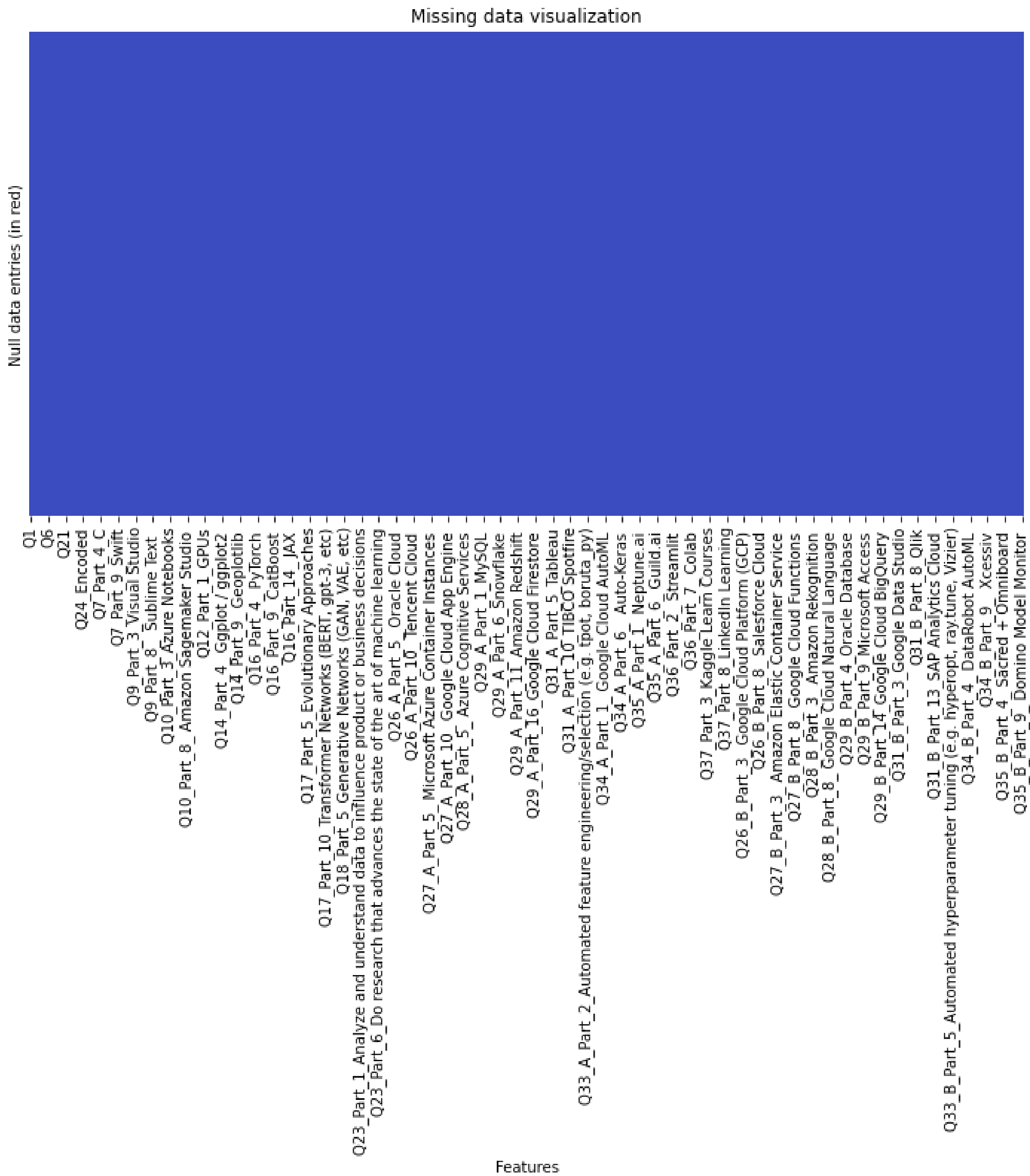
	Q1	Q2	Q3	Q4	Q5	Q6	Q11	Q13	Q15	Q20	.
1	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	A personal computer or laptop	2-5 times	1-2 years	10,000 or more employees	.
2	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	1000-9,999 employees	.
3	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	250-999 employees	.
4	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	1000-9,999 employees	.
5	35-39	Man	United States of America	Doctoral degree	Research Scientist	1-2 years	A personal computer or laptop	Never	Under 1 year	0-49 employees	.
...	...	...	...	...	...	...	...	...	...	...	.
10725	35-39	Man	Malaysia	I prefer not to answer	Machine Learning Engineer	1-2 years	A personal computer or laptop	Never	1-2 years	0-49 employees	.
10726	35-39	Man	Thailand	Bachelor's degree	Other	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	250-999 employees	.
10727	30-34	Man	Brazil	Master's degree	Research Scientist	< 1 years	A personal computer or laptop	Never	I do not use machine learning methods	0-49 employees	.
10728	22-24	Man	India	Bachelor's degree	Software Engineer	3-5 years	A cloud computing platform (AWS, Azure, GCP, h...	More than 25 times	1-2 years	10,000 or more employees	.
10729	22-24	Man	Pakistan	Master's degree	Machine Learning Engineer	< 1 years	A cloud computing platform (AWS, Azure, GCP, h...	Once	Under 1 year	0-49 employees	.

10729 rows × 286 columns

The dataframe is reduced to 10729 rows and 313 columns. The heat map is generated to visualize the missing data.

In [31]:
```
fig, ax = plt.subplots(figsize=(12,6))
ax = sns.heatmap(kaggle_encoded.isnull(), cmap='coolwarm', yticklabels=False, cbar=False, ax=ax)
ax.set_xlabel('Features')
ax.set_ylabel('Null data entries (in red)')
ax.set_title('Missing data visualization')
```

Out[31]:
```
Text(0.5, 1.0, 'Missing data visualization')
```



Exploratory Data Analysis

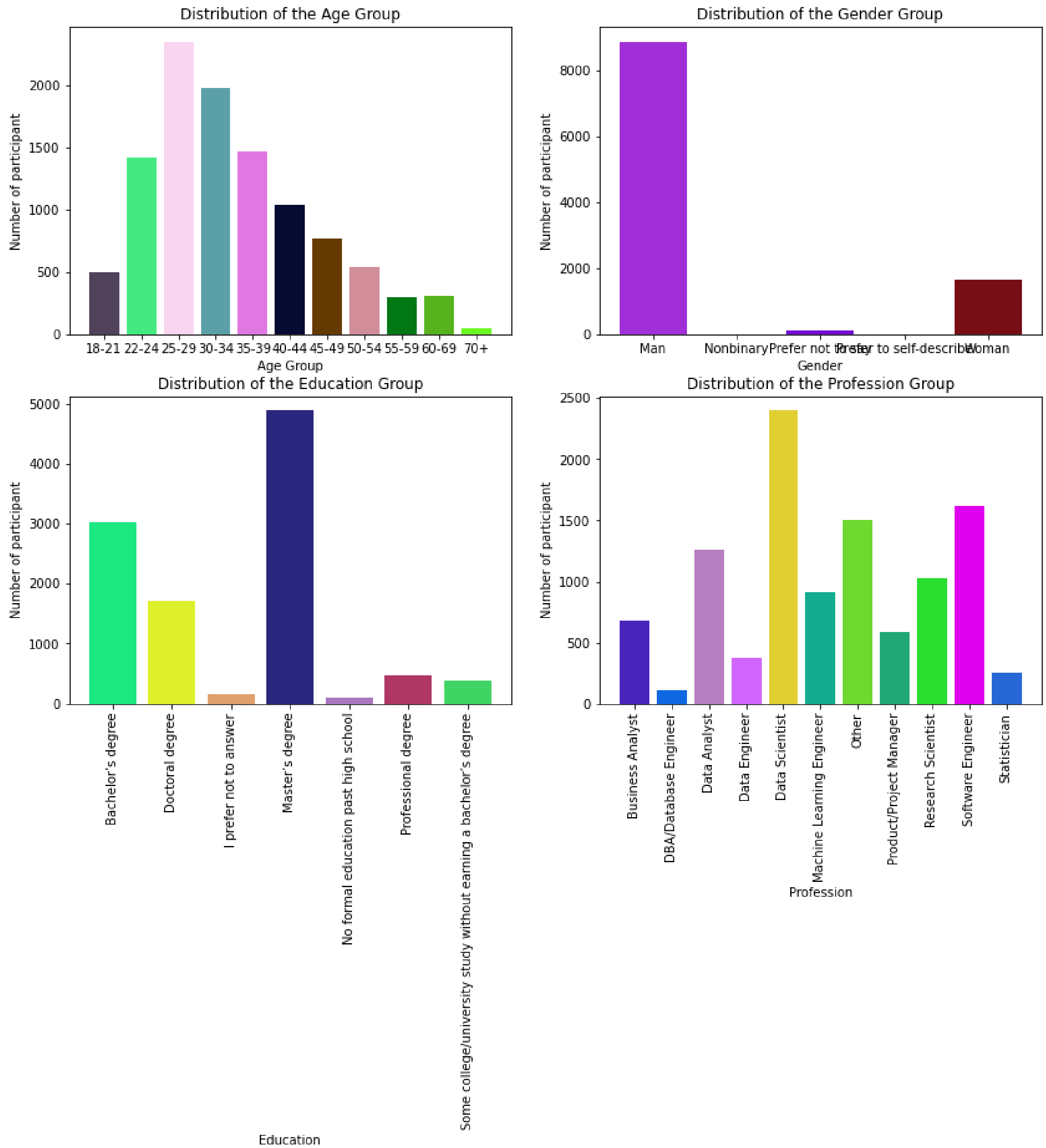
For the analysis of the dataframe, following things were analysed.

1. The distribution of Age, Gender, Education and Profession using the bar plot
2. Age and Gender distribution bar plot
3. Salary distribution according to Education
4. Salary distribution according to profession
5. Salary distribution according to years of coding experience
6. Salary distribution according to the money they spent on coding Q25

In [32]:
```
plt.figure(figsize=(14,10))

plt.subplot(2,2,1)
count = kaggle_encoded["Q1"].value_counts()
Age_count = pd.DataFrame({"Age":count.index, "total_count":count.values})
Age_count = Age_count.sort_values("Age")
print(Age_count)
plt.bar(Age_count.Age, Age_count.total_count, color=np.random.rand(11,3))
plt.xlabel('Age Group')
plt.ylabel('Number of participant')
plt.title('Distribution of the Age Group')

plt.subplot(2,2,2)
count = kaggle_encoded["Q2"].value_counts()
Gender_count = pd.DataFrame({"Gender":count.index, "total_count":count.values
})
Gender_count = Gender_count.sort_values("Gender")
print(Gender_count)
plt.bar(Gender_count.Gender, Gender_count.total_count, color=np.random.rand(11
,3))
plt.xlabel('Gender')
plt.ylabel('Number of participant')
plt.title('Distribution of the Gender Group')

plt.subplot(2,2,3)
count = kaggle_encoded["Q4"].value_counts()
Edu_count = pd.DataFrame({"Edu":count.index, "total_count":count.values})
Edu_count = Edu_count.sort_values("Edu")
print(Edu_count)
plt.bar(Edu_count.Edu, Edu_count.total_count, color=np.random.rand(11,3))
plt.xlabel('Education')
plt.ylabel('Number of participant')
plt.title('Distribution of the Education Group')
plt.xticks(rotation=90)

plt.subplot(2,2,4)
count = kaggle_encoded["Q5"].value_counts()
Prof_count = pd.DataFrame({"Profession":count.index, "total_count":count.value
s})
Prof_count = Prof_count.sort_values("Profession")
print(Prof_count)
plt.bar(Prof_count.Profession, Prof_count.total_count, color=np.random.rand(11
,3))
plt.xlabel('Profession')
plt.ylabel('Number of participant')
plt.title('Distribution of the Profession Group')
plt.xticks(rotation=90)
```

```
      Age  total_count
7   18-21          498
3   22-24         1424
0   25-29         2350
1   30-34         1979
2   35-39         1467
4   40-44         1042
5   45-49          771
6   50-54          536
9   55-59          301
8   60-69          309
10    70+           52
                  Gender  total_count
0                    Man         8872
4              Nonbinary           20
2      Prefer not to say          131
3  Prefer to self-describe         23
1                  Woman         1683
                                                 Edu  total_count
1                                  Bachelor’s degree         3013
2                                    Doctoral degree         1718
5                             I prefer not to answer          158
0                                    Master’s degree         4879
6                 No formal education past high school          106
3                                Professional degree          470
4  Some college/university study without earning ...          385
                    Profession  total_count
6             Business Analyst          678
10       DBA/Database Engineer          112
3                 Data Analyst         1260
8                Data Engineer          369
0               Data Scientist         2398
5    Machine Learning Engineer          918
2                        Other         1508
7      Product/Project Manager          590
4           Research Scientist         1028
1            Software Engineer         1620
9                 Statistician          248
```

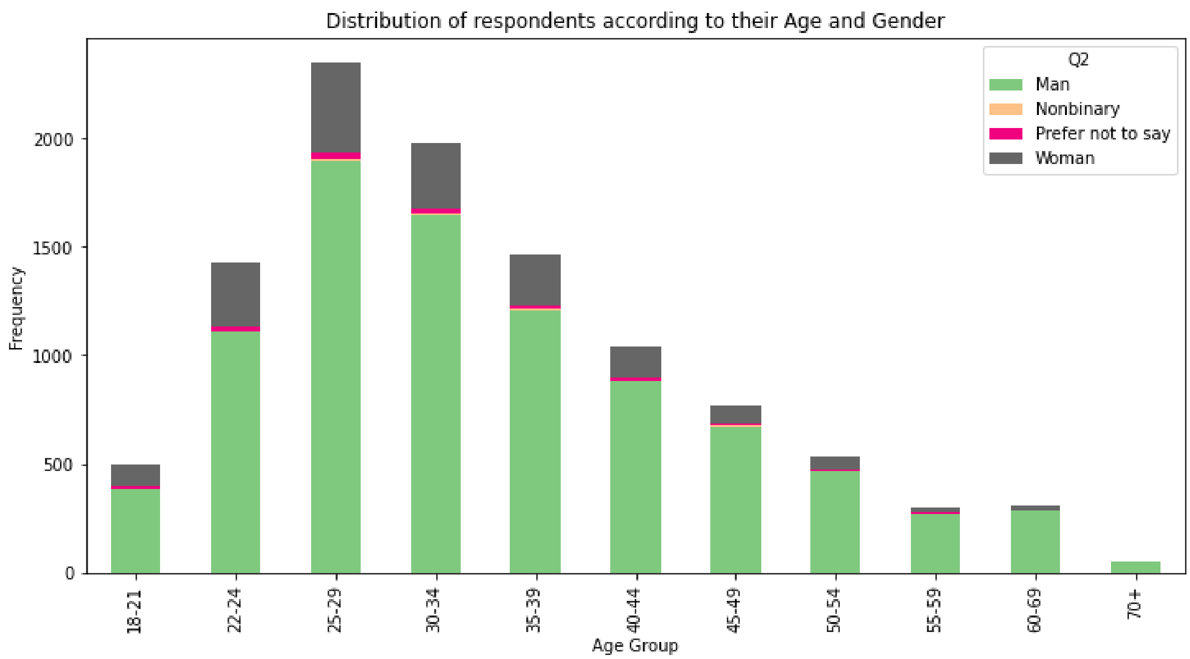
Out[32]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



The above EDA shows that the Man who accomplished Master's degree, having profession of Data Scientist among the age group of 25-29 are most of the survey respondants.

In [33]:
```
kaggle_encoded['Q2'] = kaggle_encoded['Q2'].replace('Prefer to self-describe', 'Prefer not to say')
df1 = kaggle_encoded[['Q1','Q2']]
df1['ct'] = 1
df1 = pd.pivot_table(df1, values = 'ct', index = 'Q1', columns = 'Q2', aggfunc = np.sum, fill_value =0)
ax = df1.plot.bar(stacked=True, figsize = (12,6), colormap='Accent')
ax.set_title('Distribution of respondents according to their Age and Gender')
ax.set_xlabel('Age Group')
ax.set_ylabel('Frequency')
```

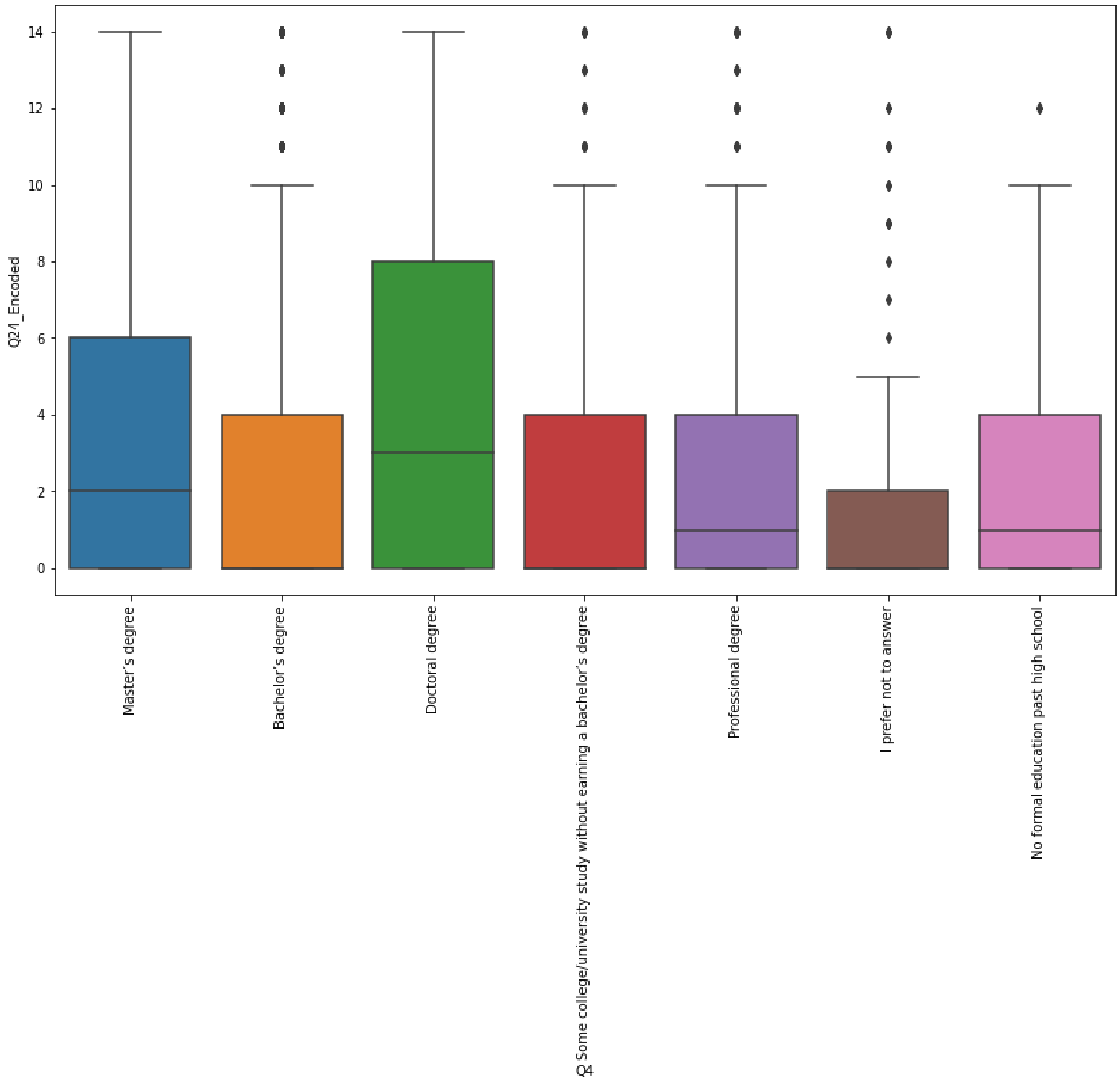
Out[33]:
```
Text(0, 0.5, 'Frequency')
```



Observation: The age group between 20 to 40 years are most of the survey respondants and those are mostly Man.

In [34]:
```
plt.figure(figsize=(14,8))
edu_sal = sns.boxplot(x='Q4', y='Q24_Encoded', data = kaggle_encoded)
edu_sal.set_xticklabels(edu_sal.get_xticklabels(), rotation=90)
```

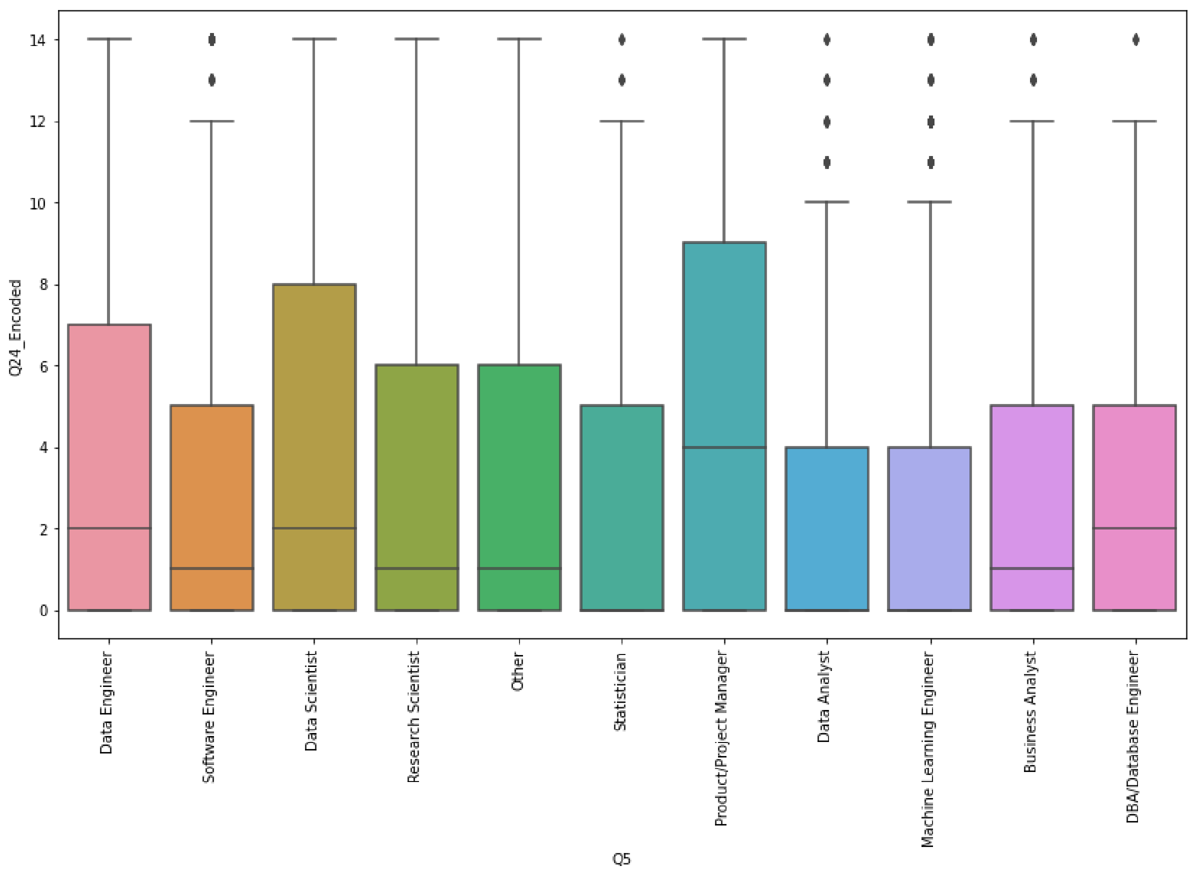
Out[34]:
```
[Text(0, 0, 'Master’s degree'),
 Text(1, 0, 'Bachelor’s degree'),
 Text(2, 0, 'Doctoral degree'),
 Text(3, 0, 'Some college/university study without earning a bachelor’s degre
e'),
 Text(4, 0, 'Professional degree'),
 Text(5, 0, 'I prefer not to answer'),
 Text(6, 0, 'No formal education past high school')]
```



Observation: Higher the education level, higher the salary. Doctoral candidates have higher mean than all other.

In [35]:
```
plt.figure(figsize=(14,8))
edu_sal = sns.boxplot(x='Q5', y='Q24_Encoded', data = kaggle_encoded)
edu_sal.set_xticklabels(edu_sal.get_xticklabels(), rotation=90)
```

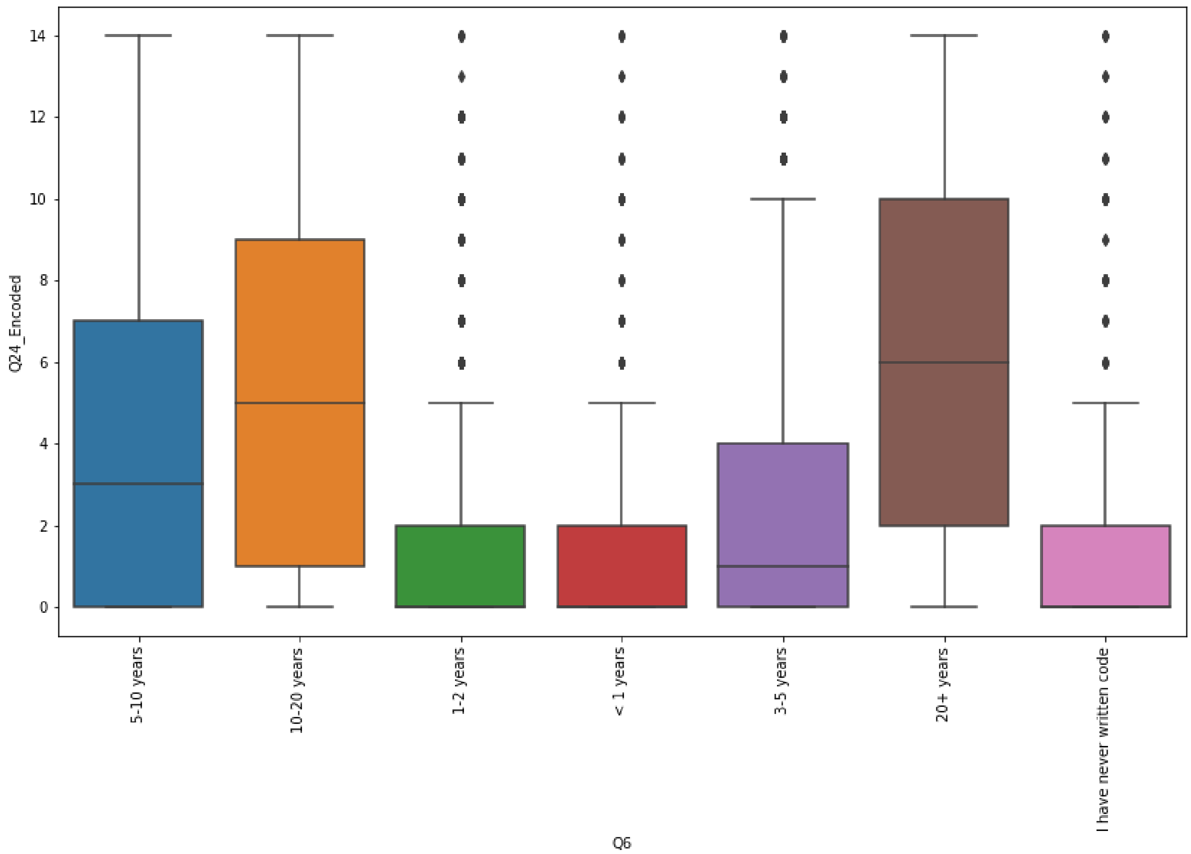
Out[35]:
```
[Text(0, 0, 'Data Engineer'),
 Text(1, 0, 'Software Engineer'),
 Text(2, 0, 'Data Scientist'),
 Text(3, 0, 'Research Scientist'),
 Text(4, 0, 'Other'),
 Text(5, 0, 'Statistician'),
 Text(6, 0, 'Product/Project Manager'),
 Text(7, 0, 'Data Analyst'),
 Text(8, 0, 'Machine Learning Engineer'),
 Text(9, 0, 'Business Analyst'),
 Text(10, 0, 'DBA/Database Engineer')]
```



Observation: The profession of Product Manager has higher mean salary than all other. Then, comes the data engineer and data scientist.

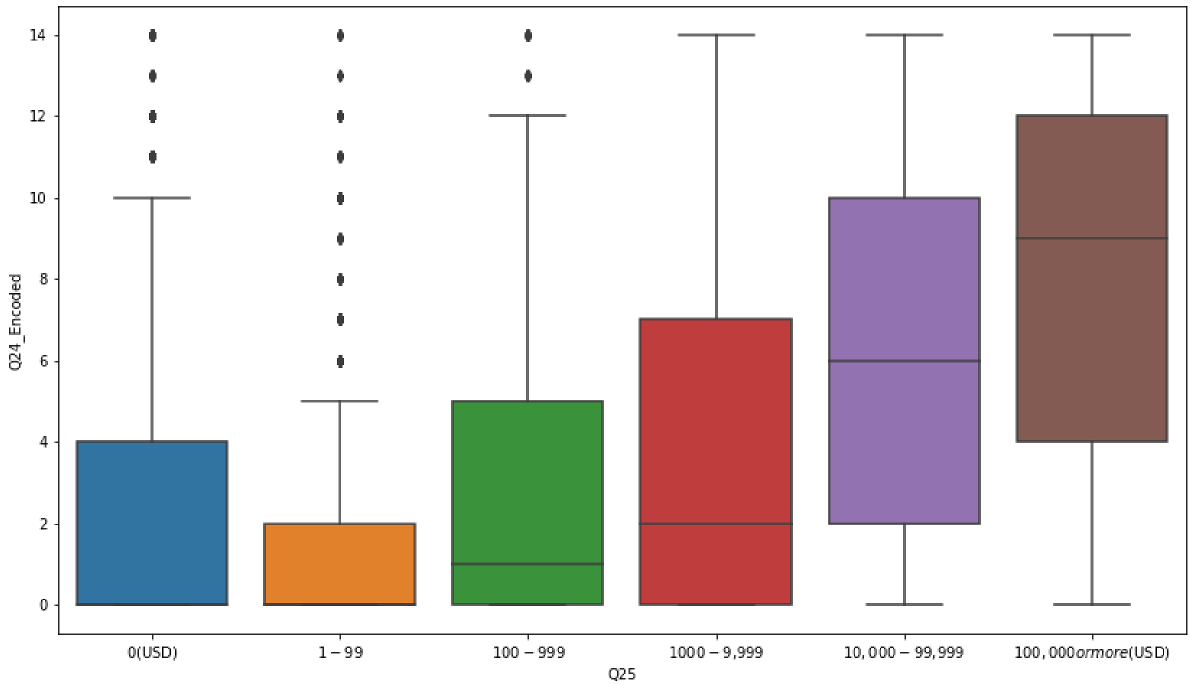
In [36]:
```
plt.figure(figsize=(14,8))
edu_sal = sns.boxplot(x='Q6', y='Q24_Encoded', data = kaggle_encoded)
edu_sal.set_xticklabels(edu_sal.get_xticklabels(), rotation=90)
```

Out[36]:
```
[Text(0, 0, '5-10 years'),
 Text(1, 0, '10-20 years'),
 Text(2, 0, '1-2 years'),
 Text(3, 0, '< 1 years'),
 Text(4, 0, '3-5 years'),
 Text(5, 0, '20+ years'),
 Text(6, 0, 'I have never written code')]
```



Observation: The salary is directly related to the experience. The person having 10-20 years or more than 20 years of experience earn more than others.

In [37]:
```
plt.figure(figsize=(14,8))
edu_sal = sns.boxplot(x='Q25', y='Q24_Encoded', data = kaggle_encoded, order = ['$0 ($USD)', '$1-$99', '$100-$999',

'$1000-$9,999', '$10,000-$99,999',

'$100,000 or more ($USD)'])
```



Observation: Here, the salary is also directly related to the money spent on the coding. So, more the investement made, more the chances of earning is there.

Feature Selection

There are around 286 column features in which most feature may not be related to the salary earned. So, the feature selection is an important step in classification data science problems.

Before that, the categorical values are there in the dataframe which is required to be converted into the numerical values.
Analysing the remaining categorical variables and converting them to numerical values by one hot encoding.

In [38]:
```
cat_col = kaggle_encoded.select_dtypes(include=['object']).columns
kaggle_encoded[cat_col].head()
```

Out[38]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q11	Q13	Q15	Q20	Q21
1	30-34	Man	United States of America	Master's degree	Data Engineer	5-10 years	A personal computer or laptop	2-5 times	1-2 years	10,000 or more employees	20+
2	35-39	Man	Argentina	Bachelor's degree	Software Engineer	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	1000-9,999 employees	0
3	30-34	Man	United States of America	Master's degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	250-999 employees	5-9
4	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	1000-9,999 employees	20+
5	35-39	Man	United States of America	Doctoral degree	Research Scientist	1-2 years	A personal computer or laptop	Never	Under 1 year	0-49 employees	1-2

In [39]:
```
#removing Q24 and Q24_buckets as we already have Q4_encoded and saving the new
dataframe with another name.
kaggle_new = kaggle_encoded.drop(['Q24','Q24_buckets'], axis=1)
kaggle_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10729 entries, 1 to 10729
Columns: 284 entries, Q1 to Q35_B_Part_9_ Domino Model Monitor
dtypes: float64(1), object(14), uint8(269)
memory usage: 4.4+ MB
```

In [40]:
```
cat_col = kaggle_new.select_dtypes(include=['object']).columns
kaggle_new[cat_col].head()
```

Out[40]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q11	Q13	Q15	Q20	Q21
1	30-34	Man	United States of America	Master’s degree	Data Engineer	5-10 years	A personal computer or laptop	2-5 times	1-2 years	10,000 or more employees	20+
2	35-39	Man	Argentina	Bachelor’s degree	Software Engineer	10-20 years	A personal computer or laptop	Never	I do not use machine learning methods	1000-9,999 employees	0
3	30-34	Man	United States of America	Master’s degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	3-4 years	250-999 employees	5-9
4	35-39	Man	Germany	Doctoral degree	Data Scientist	5-10 years	A cloud computing platform (AWS, Azure, GCP, h...	2-5 times	2-3 years	1000-9,999 employees	20+
5	35-39	Man	United States of America	Doctoral degree	Research Scientist	1-2 years	A personal computer or laptop	Never	Under 1 year	0-49 employees	1-2

Here, Q1, Q6, Q13, Q15, Q20, Q21, Q25 includes some bucket of values which can be mapped to the specific integer. So, those column can be label encoded (as it works for ordinal data). Those column have some sort of order.

In [41]:
```
# Encoding the ordinal data
# Here the ordinal columns are Q1, Q6, Q13, Q15, Q20, Q21, Q25
# So label encoding the given columns using the method shown in Tutorial.

# Q1
kaggle_new['Q1'].unique()

# Q6
kaggle_new['Q6'].unique()

# Q13
kaggle_new['Q13'].unique()

# Q15
kaggle_new['Q15'].unique()

# Q20
kaggle_new['Q20'].unique()

# Q21
kaggle_new['Q21'].unique()

# Q25
kaggle_new['Q25'].unique()
```

Out[41]:
```
array(['$100,000 or more ($USD)', '$0 ($USD)', '$10,000-$99,999',
       '$1-$99', '$1000-$9,999', '$100-$999'], dtype=object)
```

In [42]:
```
# Q1
kaggle_new['Q1'] = kaggle_new['Q1'].map({'18-21':0, '22-24':1, '25-29':2, '30-
34':3, '35-39':4, '40-44':5,
                                        '45-49':6, '50-54':7, '55-59':8, '60-6
9':9, '70+':10})
kaggle_new['Q1'] = kaggle_new['Q1'].astype(int)

# Q6
kaggle_new['Q6'] = kaggle_new['Q6'].map({'< 1 years':0, '1-2 years':1, '3-5 ye
ars':2, '5-10 years':3,
                                         '10-20 years':4, '20+ years':5,
                                        'I have never written code':6})
kaggle_new['Q6'] = kaggle_new['Q6'].astype(int)

# Q13
kaggle_new['Q13'] = kaggle_new['Q13'].map({'Never':0, 'Once':1, '2-5 times':2,
'6-25 times':3,
                                         'More than 25 times':4})
kaggle_new['Q13'] = kaggle_new['Q13'].astype(int)

# Q15
kaggle_new['Q15'] = kaggle_new['Q15'].map({'I do not use machine learning meth
ods':0, 'Under 1 year':1, '1-2 years':2,
                                           '2-3 years':3, '3-4 years':4, '4-5
 years':5, '5-10 years':6,
                                           '10-20 years':7, '20 or more years'
:8})
kaggle_new['Q15'] = kaggle_new['Q15'].astype(int)

# Q20
kaggle_new['Q20'] = kaggle_new['Q20'].map({'0-49 employees':0, '50-249 employe
es':1, '250-999 employees':2,
                                           '1000-9,999 employees':3, '10,000 o
r more employees':4})
kaggle_new['Q20'] = kaggle_new['Q20'].astype(int)

# Q21
kaggle_new['Q21'] = kaggle_new['Q21'].map({'0':0, '1-2':1, '3-4':2, '5-9':3,
'10-14':4, '15-19':5, '20+':6})
kaggle_new['Q21'] = kaggle_new['Q21'].astype(int)

# Q25
kaggle_new['Q25'] = kaggle_new['Q25'].map({'$0 ($USD)':0, '$1-$99':1, '$100-$9
99':2, '$1000-$9,999':3,
                                           '$10,000-$99,999':4, '$100,000 or m
ore ($USD)':5})
kaggle_new['Q25'] = kaggle_new['Q25'].astype(int)
```

Now, the rest columns are nominal data columns. So, those are one hot encoded except the column of Q3.

The Q3 includes the name of countries and it is analysed by grouping and then one hot encoding as shown later.

In [43]:
```python
cat_col = kaggle_new.select_dtypes(include=['object']).columns
cat_col = cat_col.drop('Q3')
print(cat_col)
kaggle_new[cat_col].head()
```

```
Index(['Q2', 'Q4', 'Q5', 'Q11', 'Q22', 'Q38'], dtype='object')
```

Out[43]:

	Q2	Q4	Q5	Q11	Q22	Q38
1	Man	Master’s degree	Data Engineer	A personal computer or laptop	We have well established ML methods (i.e., mod...	Business intelligence software (Salesforce, Ta...
2	Man	Bachelor’s degree	Software Engineer	A personal computer or laptop	No (we do not use ML methods)	Basic statistical software (Microsoft Excel, G...
3	Man	Master’s degree	Data Scientist	A cloud computing platform (AWS, Azure, GCP, h...	We have well established ML methods (i.e., mod...	Local development environments (RStudio, Jupyt...
4	Man	Doctoral degree	Data Scientist	A cloud computing platform (AWS, Azure, GCP, h...	We have well established ML methods (i.e., mod...	Cloud-based data software & APIs (AWS, GCP, Az...
5	Man	Doctoral degree	Research Scientist	A personal computer or laptop	We use ML methods for generating insights (but...	Local development environments (RStudio, Jupyt...

In [44]:
```
kaggle_df = pd.get_dummies(kaggle_new, columns=cat_col)
kaggle_df
```

Out[44]:

	Q1	Q3	Q6	Q13	Q15	Q20	Q21	Q25	Q24_Encoded	Q7_Part_1_Python	...	e: r (a mo pro
1	3	United States of America	3	2	2	4	6	5	10.0	1	...	
2	4	Argentina	4	0	0	3	0	0	1.0	0	...	
3	3	United States of America	3	2	4	2	3	4	11.0	1	...	
4	4	Germany	3	2	3	3	6	4	7.0	1	...	
5	4	United States of America	1	0	1	0	1	1	3.0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
10725	4	Malaysia	1	0	2	0	3	0	0.0	1	...	
10726	4	Thailand	4	0	0	2	0	0	1.0	0	...	
10727	3	Brazil	0	0	0	0	0	0	0.0	1	...	
10728	1	India	2	4	2	4	6	0	0.0	1	...	
10729	1	Pakistan	0	1	1	0	0	0	0.0	1	...	

10729 rows × 317 columns

In [45]:
```
cat_col = kaggle_df.select_dtypes(include=['object']).columns
print(cat_col)
```

```
Index(['Q3'], dtype='object')
```

The countries in Q3 are grouped into three sections, i.e. C1, C2, and C3. There are around 54 countries and it is sort in order according to their count. The first highest number of 18 countries are grouped in C1. The next 18 highest count countries are grouped in C2. The last 18 least included countries are grouped in C3.
So, there are now three groups C1, C2, and C3 instead of 54 countries in column Q3.

In [46]:
```
countries = kaggle_df.Q3
Ct_count = countries.value_counts()
Country = pd.DataFrame({"Country":Ct_count.index, "count":Ct_count.values})
C1 = Country[0:19]
C2 = Country[19:37]
C3 = Country[37:55]
```

In [47]:
```
kaggle_df = kaggle_df.replace(C1.Country.values, 'C1')
kaggle_df = kaggle_df.replace(C2.Country.values, 'C2')
kaggle_df = kaggle_df.replace(C3.Country.values, 'C3')
kaggle_df
```

Out[47]:

	Q1	Q3	Q6	Q13	Q15	Q20	Q21	Q25	Q24_Encoded	Q7_Part_1_Python	...	Q22_W a explorir M methoc (and ma one da put model in productio
1	3	C1	3	2	2	4	6	5	10.0	1	...	
2	4	C2	4	0	0	3	0	0	1.0	0	...	
3	3	C1	3	2	4	2	3	4	11.0	1	...	
4	4	C1	3	2	3	3	6	4	7.0	1	...	
5	4	C1	1	0	1	0	1	1	3.0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
10725	4	C3	1	0	2	0	3	0	0.0	1	...	
10726	4	C2	4	0	0	2	0	0	1.0	0	...	
10727	3	C1	0	0	0	0	0	0	0.0	1	...	
10728	1	C1	2	4	2	4	6	0	0.0	1	...	
10729	1	C2	0	1	1	0	0	0	0.0	1	...	

10729 rows × 317 columns

Now, the Q3 column is one hot encoded using pandas get_dummies function. So, there will be three new columns Q3_C1, Q3_C2, and Q3_C3 instead of column Q3.

In [48]:
```
kaggle_df = pd.get_dummies(kaggle_df, columns=['Q3'])
kaggle_df
```

Out[48]:

	Q1	Q6	Q13	Q15	Q20	Q21	Q25	Q24_Encoded	Q7_Part_1_Python	Q7_Part_2_R	...
1	3	3	2	2	4	6	5	10.0	1	1	...
2	4	4	0	0	3	0	0	1.0	0	0	...
3	3	3	2	4	2	3	4	11.0	1	0	...
4	4	3	2	3	3	6	4	7.0	1	0	...
5	4	1	0	1	0	1	1	3.0	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...
10725	4	1	0	2	0	3	0	0.0	1	0	...
10726	4	4	0	0	2	0	0	1.0	0	0	...
10727	3	0	0	0	0	0	0	0.0	1	0	...
10728	1	2	4	2	4	6	0	0.0	1	0	...
10729	1	0	1	1	0	0	0	0.0	1	0	...

10729 rows × 319 columns

In [49]:
```
kaggle_df.columns.values
```

Out[49]:
```
array(['Q1', 'Q6', 'Q13', 'Q15', 'Q20', 'Q21', 'Q25', 'Q24_Encoded',
       'Q7_Part_1_Python', 'Q7_Part_2_R', 'Q7_Part_3_SQL', 'Q7_Part_4_C',
       'Q7_Part_5_C++', 'Q7_Part_6_Java', 'Q7_Part_7_Javascript',
       'Q7_Part_8_Julia', 'Q7_Part_9_Swift', 'Q7_Part_10_Bash',
       'Q7_Part_11_MATLAB',
       'Q9_Part_1_Jupyter (JupyterLab, Jupyter Notebooks, etc) ',
       'Q9_Part_2_ RStudio ', 'Q9_Part_3_Visual Studio',
       'Q9_Part_4_Visual Studio Code (VSCode)', 'Q9_Part_5_ PyCharm ',
       'Q9_Part_6_  Spyder  ', 'Q9_Part_7_  Notepad++  ',
       'Q9_Part_8_  Sublime Text  ', 'Q9_Part_9_  Vim / Emacs  ',
       'Q9_Part_10_ MATLAB ', 'Q10_Part_1_ Kaggle Notebooks',
       'Q10_Part_2_Colab Notebooks', 'Q10_Part_3_Azure Notebooks',
       'Q10_Part_4_ Paperspace / Gradient ',
       'Q10_Part_5_ Binder / JupyterHub ', 'Q10_Part_6_ Code Ocean ',
       'Q10_Part_7_ IBM Watson Studio ',
       'Q10_Part_8_ Amazon Sagemaker Studio ',
       'Q10_Part_9_ Amazon EMR Notebooks ',
       'Q10_Part_10_Google Cloud AI Platform Notebooks ',
       'Q10_Part_11_Google Cloud Datalab Notebooks',
       'Q10_Part_12_ Databricks Collaborative Notebooks ',
       'Q12_Part_1_GPUs', 'Q12_Part_2_TPUs', 'Q14_Part_1_ Matplotlib ',
       'Q14_Part_2_ Seaborn ', 'Q14_Part_3_ Plotly / Plotly Express ',
       'Q14_Part_4_ Ggplot / ggplot2 ', 'Q14_Part_5_ Shiny ',
       'Q14_Part_6_ D3 js ', 'Q14_Part_7_ Altair ', 'Q14_Part_8_ Bokeh ',
       'Q14_Part_9_ Geoplotlib ', 'Q14_Part_10_ Leaflet / Folium ',
       'Q16_Part_1_  Scikit-learn ', 'Q16_Part_2_  TensorFlow ',
       'Q16_Part_3_ Keras ', 'Q16_Part_4_ PyTorch ',
       'Q16_Part_5_ Fast.ai ', 'Q16_Part_6_ MXNet ',
       'Q16_Part_7_ Xgboost ', 'Q16_Part_8_ LightGBM ',
       'Q16_Part_9_ CatBoost ', 'Q16_Part_10_ Prophet ',
       'Q16_Part_11_ H2O 3 ', 'Q16_Part_12_ Caret ',
       'Q16_Part_13_ Tidymodels ', 'Q16_Part_14_ JAX ',
       'Q17_Part_1_Linear or Logistic Regression',
       'Q17_Part_2_Decision Trees or Random Forests',
       'Q17_Part_3_Gradient Boosting Machines (xgboost, lightgbm, etc)',
       'Q17_Part_4_Bayesian Approaches',
       'Q17_Part_5_Evolutionary Approaches',
       'Q17_Part_6_Dense Neural Networks (MLPs, etc)',
       'Q17_Part_7_Convolutional Neural Networks',
       'Q17_Part_8_Generative Adversarial Networks',
       'Q17_Part_9_Recurrent Neural Networks',
       'Q17_Part_10_Transformer Networks (BERT, gpt-3, etc)',
       'Q18_Part_1_General purpose image/video tools (PIL, cv2, skimage, et
c)',
       'Q18_Part_2_Image segmentation methods (U-Net, Mask R-CNN, etc)',
       'Q18_Part_3_Object detection methods (YOLOv3, RetinaNet, etc)',
       'Q18_Part_4_Image classification and other general purpose networks (V
GG, Inception, ResNet, ResNeXt, NASNet, EfficientNet, etc)',
       'Q18_Part_5_Generative Networks (GAN, VAE, etc)',
       'Q19_Part_1_Word embeddings/vectors (GLoVe, fastText, word2vec)',
       'Q19_Part_2_Encoder-decorder models (seq2seq, vanilla transformers)',
       'Q19_Part_3_Contextualized embeddings (ELMo, CoVe)',
       'Q19_Part_4_Transformer language models (GPT-3, BERT, XLnet, etc)',
       'Q23_Part_1_Analyze and understand data to influence product or busine
ss decisions',
       'Q23_Part_2_Build and/or run the data infrastructure that my business
uses for storing, analyzing, and operationalizing data',
```

```
       'Q23_Part_3_Build prototypes to explore applying machine learning to n
ew areas',
       'Q23_Part_4_Build and/or run a machine learning service that operation
ally improves my product or workflows',
       'Q23_Part_5_Experimentation and iteration to improve existing ML model
s',
       'Q23_Part_6_Do research that advances the state of the art of machine
learning',
       'Q26_A_Part_1_ Amazon Web Services (AWS) ',
       'Q26_A_Part_2_ Microsoft Azure ',
       'Q26_A_Part_3_ Google Cloud Platform (GCP) ',
       'Q26_A_Part_4_ IBM Cloud / Red Hat ',
       'Q26_A_Part_5_ Oracle Cloud ', 'Q26_A_Part_6_ SAP Cloud ',
       'Q26_A_Part_7_ Salesforce Cloud ', 'Q26_A_Part_8_ VMware Cloud ',
       'Q26_A_Part_9_ Alibaba Cloud ', 'Q26_A_Part_10_ Tencent Cloud ',
       'Q27_A_Part_1_ Amazon EC2 ', 'Q27_A_Part_2_ AWS Lambda ',
       'Q27_A_Part_3_ Amazon Elastic Container Service ',
       'Q27_A_Part_4_ Azure Cloud Services ',
       'Q27_A_Part_5_ Microsoft Azure Container Instances ',
       'Q27_A_Part_6_ Azure Functions ',
       'Q27_A_Part_7_ Google Cloud Compute Engine ',
       'Q27_A_Part_8_ Google Cloud Functions ',
       'Q27_A_Part_9_ Google Cloud Run ',
       'Q27_A_Part_10_ Google Cloud App Engine ',
       'Q28_A_Part_1_ Amazon SageMaker ',
       'Q28_A_Part_2_ Amazon Forecast ',
       'Q28_A_Part_3_ Amazon Rekognition ',
       'Q28_A_Part_4_ Azure Machine Learning Studio ',
       'Q28_A_Part_5_ Azure Cognitive Services ',
       'Q28_A_Part_6_ Google Cloud AI Platform / Google Cloud ML Engine',
       'Q28_A_Part_7_ Google Cloud Video AI ',
       'Q28_A_Part_8_ Google Cloud Natural Language ',
       'Q28_A_Part_9_ Google Cloud Vision AI ', 'Q29_A_Part_1_MySQL ',
       'Q29_A_Part_2_PostgresSQL ', 'Q29_A_Part_3_SQLite ',
       'Q29_A_Part_4_Oracle Database ', 'Q29_A_Part_5_MongoDB ',
       'Q29_A_Part_6_Snowflake ', 'Q29_A_Part_7_IBM Db2 ',
       'Q29_A_Part_8_Microsoft SQL Server ',
       'Q29_A_Part_9_Microsoft Access ',
       'Q29_A_Part_10_Microsoft Azure Data Lake Storage ',
       'Q29_A_Part_11_Amazon Redshift ', 'Q29_A_Part_12_Amazon Athena ',
       'Q29_A_Part_13_Amazon DynamoDB ',
       'Q29_A_Part_14_Google Cloud BigQuery ',
       'Q29_A_Part_15_Google Cloud SQL ',
       'Q29_A_Part_16_Google Cloud Firestore ',
       'Q31_A_Part_1_Amazon QuickSight',
       'Q31_A_Part_2_Microsoft Power BI',
       'Q31_A_Part_3_Google Data Studio', 'Q31_A_Part_4_Looker',
       'Q31_A_Part_5_Tableau', 'Q31_A_Part_6_Salesforce',
       'Q31_A_Part_7_Einstein Analytics', 'Q31_A_Part_8_Qlik',
       'Q31_A_Part_9_Domo', 'Q31_A_Part_10_TIBCO Spotfire',
       'Q31_A_Part_11_Alteryx ', 'Q31_A_Part_12_Sisense ',
       'Q31_A_Part_13_SAP Analytics Cloud ',
       'Q33_A_Part_1_Automated data augmentation (e.g. imgaug, albumentation
s)',
       'Q33_A_Part_2_Automated feature engineering/selection (e.g. tpot, boru
ta_py)',
       'Q33_A_Part_3_Automated model selection (e.g. auto-sklearn, xcessiv)',
```

```
       'Q33_A_Part_4_Automated model architecture searches (e.g. darts, ena
s)',
       'Q33_A_Part_5_Automated hyperparameter tuning (e.g. hyperopt, ray.tun
e, Vizier)',
       'Q33_A_Part_6_Automation of full ML pipelines (e.g. Google AutoML, H20
Driverless AI)',
       'Q34_A_Part_1_ Google Cloud AutoML ',
       'Q34_A_Part_2_ H20 Driverless AI  ',
       'Q34_A_Part_3_ Databricks AutoML ',
       'Q34_A_Part_4_ DataRobot AutoML ', 'Q34_A_Part_5_  Tpot ',
       'Q34_A_Part_6_  Auto-Keras ', 'Q34_A_Part_7_  Auto-Sklearn ',
       'Q34_A_Part_8_  Auto_ml ', 'Q34_A_Part_9_  Xcessiv ',
       'Q34_A_Part_10_  MLbox ', 'Q35_A_Part_1_ Neptune.ai ',
       'Q35_A_Part_2_ Weights & Biases ', 'Q35_A_Part_3_ Comet.ml ',
       'Q35_A_Part_4_ Sacred + Omniboard ', 'Q35_A_Part_5_ TensorBoard ',
       'Q35_A_Part_6_ Guild.ai ', 'Q35_A_Part_7_ Polyaxon ',
       'Q35_A_Part_8_ Trains ', 'Q35_A_Part_9_ Domino Model Monitor ',
       'Q36_Part_1_ Plotly Dash ', 'Q36_Part_2_ Streamlit ',
       'Q36_Part_3_ NBViewer ', 'Q36_Part_4_ GitHub ',
       'Q36_Part_5_ Personal blog ', 'Q36_Part_6_ Kaggle ',
       'Q36_Part_7_ Colab ', 'Q36_Part_8_ Shiny ',
       'Q36_Part_9_I do not share my work publicly',
       'Q37_Part_1_Coursera', 'Q37_Part_2_edX',
       'Q37_Part_3_Kaggle Learn Courses', 'Q37_Part_4_DataCamp',
       'Q37_Part_5_Fast.ai', 'Q37_Part_6_Udacity', 'Q37_Part_7_Udemy',
       'Q37_Part_8_LinkedIn Learning',
       'Q37_Part_9_Cloud-certification programs (direct from AWS, Azure, GCP,
or similar)',
       'Q37_Part_10_University Courses (resulting in a university degree)',
       'Q26_B_Part_1_ Amazon Web Services (AWS) ',
       'Q26_B_Part_2_ Microsoft Azure ',
       'Q26_B_Part_3_ Google Cloud Platform (GCP) ',
       'Q26_B_Part_4_ IBM Cloud / Red Hat ',
       'Q26_B_Part_5_ Oracle Cloud ', 'Q26_B_Part_6_ SAP Cloud ',
       'Q26_B_Part_7_ VMware Cloud ', 'Q26_B_Part_8_ Salesforce Cloud ',
       'Q26_B_Part_9_ Alibaba Cloud ', 'Q26_B_Part_10_ Tencent Cloud ',
       'Q27_B_Part_1_ Amazon EC2 ', 'Q27_B_Part_2_ AWS Lambda ',
       'Q27_B_Part_3_ Amazon Elastic Container Service ',
       'Q27_B_Part_4_ Azure Cloud Services ',
       'Q27_B_Part_5_ Microsoft Azure Container Instances ',
       'Q27_B_Part_6_ Azure Functions ',
       'Q27_B_Part_7_ Google Cloud Compute Engine ',
       'Q27_B_Part_8_ Google Cloud Functions ',
       'Q27_B_Part_9_ Google Cloud Run ',
       'Q27_B_Part_10_ Google Cloud App Engine ',
       'Q28_B_Part_1_ Amazon SageMaker ',
       'Q28_B_Part_2_ Amazon Forecast ',
       'Q28_B_Part_3_ Amazon Rekognition ',
       'Q28_B_Part_4_ Azure Machine Learning Studio ',
       'Q28_B_Part_5_ Azure Cognitive Services ',
       'Q28_B_Part_6_ Google Cloud AI Platform / Google Cloud ML Engine',
       'Q28_B_Part_7_ Google Cloud Video AI ',
       'Q28_B_Part_8_ Google Cloud Natural Language ',
       'Q28_B_Part_9_ Google Cloud Vision AI ', 'Q29_B_Part_1_MySQL ',
       'Q29_B_Part_2_PostgresSQL ', 'Q29_B_Part_3_SQLite ',
       'Q29_B_Part_4_Oracle Database ', 'Q29_B_Part_5_MongoDB ',
       'Q29_B_Part_6_Snowflake ', 'Q29_B_Part_7_IBM Db2 ',
```

```
        'Q29_B_Part_8_Microsoft SQL Server ',
        'Q29_B_Part_9_Microsoft Access ',
        'Q29_B_Part_10_Microsoft Azure Data Lake Storage ',
        'Q29_B_Part_11_Amazon Redshift ', 'Q29_B_Part_12_Amazon Athena ',
        'Q29_B_Part_13_Amazon DynamoDB ',
        'Q29_B_Part_14_Google Cloud BigQuery ',
        'Q29_B_Part_15_Google Cloud SQL ',
        'Q29_B_Part_16_Google Cloud Firestore ',
        'Q31_B_Part_1_Microsoft Power BI',
        'Q31_B_Part_2_Amazon QuickSight',
        'Q31_B_Part_3_Google Data Studio', 'Q31_B_Part_4_Looker',
        'Q31_B_Part_5_Tableau', 'Q31_B_Part_6_Salesforce',
        'Q31_B_Part_7_Einstein Analytics', 'Q31_B_Part_8_Qlik',
        'Q31_B_Part_9_Domo', 'Q31_B_Part_10_TIBCO Spotfire',
        'Q31_B_Part_11_Alteryx ', 'Q31_B_Part_12_Sisense ',
        'Q31_B_Part_13_SAP Analytics Cloud ',
        'Q33_B_Part_1_Automated data augmentation (e.g. imgaug, albumentation
s)',
        'Q33_B_Part_2_Automated feature engineering/selection (e.g. tpot, boru
ta_py)',
        'Q33_B_Part_3_Automated model selection (e.g. auto-sklearn, xcessiv)',
        'Q33_B_Part_4_Automated model architecture searches (e.g. darts, ena
s)',
        'Q33_B_Part_5_Automated hyperparameter tuning (e.g. hyperopt, ray.tun
e, Vizier)',
        'Q33_B_Part_6_Automation of full ML pipelines (e.g. Google Cloud AutoM
L, H20 Driverless AI)',
        'Q34_B_Part_1_ Google Cloud AutoML ',
        'Q34_B_Part_2_ H20 Driverless AI  ',
        'Q34_B_Part_3_ Databricks AutoML ',
        'Q34_B_Part_4_ DataRobot AutoML ', 'Q34_B_Part_5_  Tpot ',
        'Q34_B_Part_6_  Auto-Keras ', 'Q34_B_Part_7_  Auto-Sklearn ',
        'Q34_B_Part_8_  Auto_ml ', 'Q34_B_Part_9_  Xcessiv ',
        'Q34_B_Part_10_  MLbox ', 'Q35_B_Part_1_ Neptune.ai ',
        'Q35_B_Part_2_ Weights & Biases ', 'Q35_B_Part_3_ Comet.ml ',
        'Q35_B_Part_4_ Sacred + Omniboard ', 'Q35_B_Part_5_ TensorBoard ',
        'Q35_B_Part_6_ Guild.ai ', 'Q35_B_Part_7_ Polyaxon ',
        'Q35_B_Part_8_ Trains ', 'Q35_B_Part_9_ Domino Model Monitor ',
        'Q2_Man', 'Q2_Nonbinary', 'Q2_Prefer not to say', 'Q2_Woman',
        'Q4_Bachelor’s degree', 'Q4_Doctoral degree',
        'Q4_I prefer not to answer', 'Q4_Master’s degree',
        'Q4_No formal education past high school',
        'Q4_Professional degree',
        'Q4_Some college/university study without earning a bachelor’s degre
e',
        'Q5_Business Analyst', 'Q5_DBA/Database Engineer',
        'Q5_Data Analyst', 'Q5_Data Engineer', 'Q5_Data Scientist',
        'Q5_Machine Learning Engineer', 'Q5_Other',
        'Q5_Product/Project Manager', 'Q5_Research Scientist',
        'Q5_Software Engineer', 'Q5_Statistician',
        'Q11_A cloud computing platform (AWS, Azure, GCP, hosted notebooks, et
c)',
        'Q11_A deep learning workstation (NVIDIA GTX, LambdaLabs, etc)',
        'Q11_A personal computer or laptop', 'Q11_None', 'Q11_Other',
        'Q22_I do not know', 'Q22_No (we do not use ML methods)',
        'Q22_We are exploring ML methods (and may one day put a model into pro
duction)',
```

```
       'Q22_We have well established ML methods (i.e., models in production f
or more than 2 years)',
       'Q22_We recently started using ML methods (i.e., models in production
for less than 2 years)',
       'Q22_We use ML methods for generating insights (but do not put working
models into production)',
       'Q38_Advanced statistical software (SPSS, SAS, etc.)',
       'Q38_Basic statistical software (Microsoft Excel, Google Sheets, et
c.)',
       'Q38_Business intelligence software (Salesforce, Tableau, Spotfire, et
c.)',
       'Q38_Cloud-based data software & APIs (AWS, GCP, Azure, etc.)',
       'Q38_Local development environments (RStudio, JupyterLab, etc.)',
       'Q38_Other', 'Q3_C1', 'Q3_C2', 'Q3_C3'], dtype=object)
```

Feature Selection Method 1:

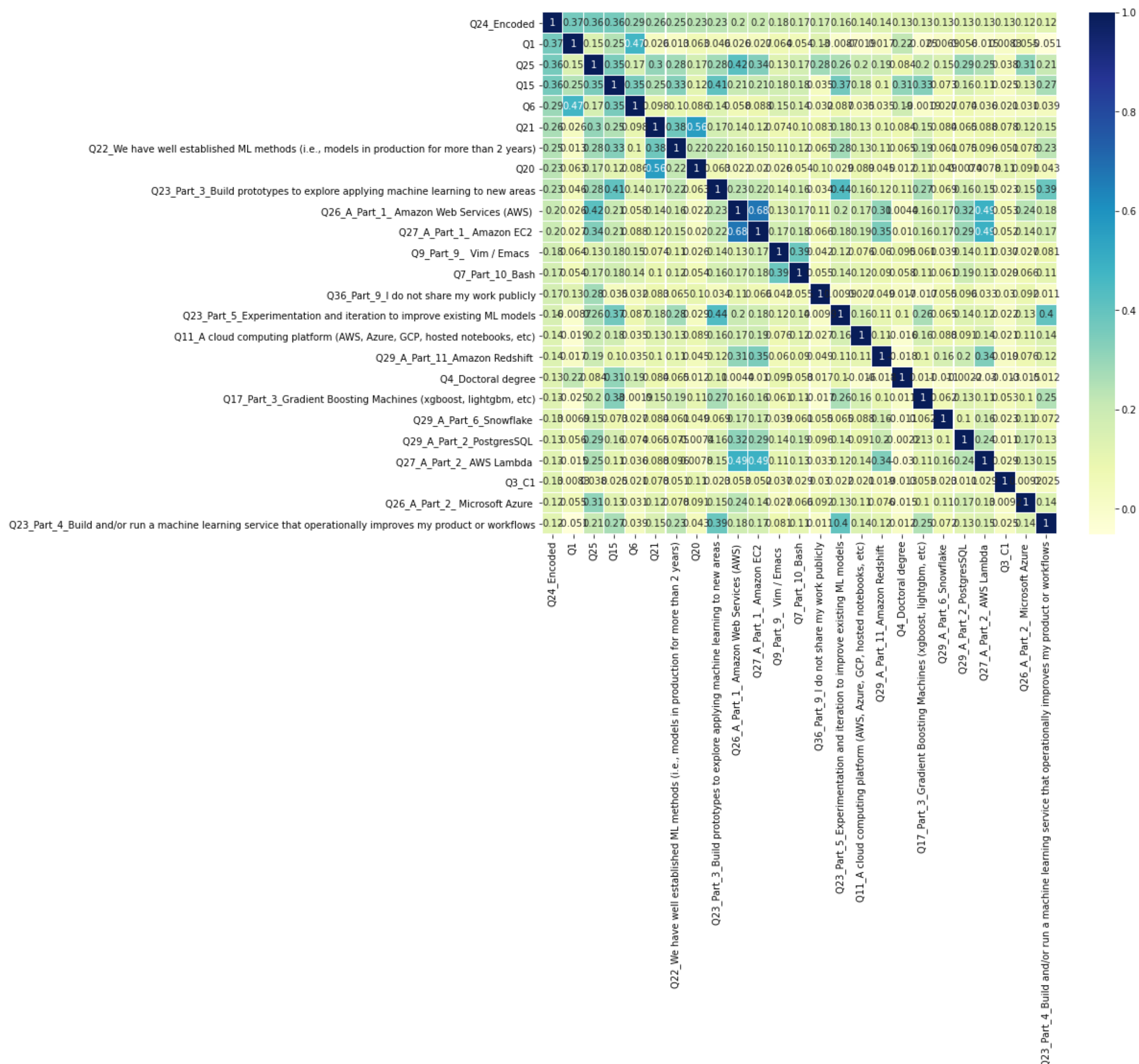
Correlation plot: Correlation states how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable).
https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e (https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e)

In [50]:
```
corr = kaggle_df.corr()
rel_corr = (corr['Q24_Encoded']).sort_values(ascending=False)
print(rel_corr[0:15])
```

```
Q24_Encoded
1.000000
Q1
0.367102
Q25
0.362844
Q15
0.359976
Q6
0.291105
Q21
0.263443
Q22_We have well established ML methods (i.e., models in production for more
than 2 years)    0.245233
Q20
0.234390
Q23_Part_3_Build prototypes to explore applying machine learning to new areas
0.229095
Q26_A_Part_1_ Amazon Web Services (AWS)
0.204226
Q27_A_Part_1_ Amazon EC2
0.198860
Q9_Part_9_  Vim / Emacs
0.177874
Q7_Part_10_Bash
0.169667
Q36_Part_9_I do not share my work publicly
0.169297
Q23_Part_5_Experimentation and iteration to improve existing ML models
0.156142
Name: Q24_Encoded, dtype: float64
```

In [51]:
```
corr = kaggle_df.corr()

cols = corr.nlargest(25, 'Q24_Encoded')['Q24_Encoded'].index

cm = np.corrcoef(kaggle_df[cols].values.T)
f, ax = plt.subplots(figsize =(12, 10))

sns.heatmap(cm, ax = ax, cmap ="YlGnBu", annot=True,
            linewidths = 0.1, yticklabels = cols.values,
                              xticklabels = cols.values)
```

Out[51]: <AxesSubplot:>



Observation: The following features are higly correlated with the Q24_Encoded. Note that correlation methods mainly works for the label encoded features.
The relevant correlated columns are:

- Q1 - Age group
- Q25 - money spent on coding
- Q15 - ML experience
- Q6 - Coding experience
- Q21 - number of individuals are responsible for data science workloads at your place of business

In [52]:
```
# The Q24_Encoded column is dropped from the dataframe and the x and y datafra
me is separated.
dfx = kaggle_df.drop(columns = ['Q24_Encoded'])
dfy = kaggle_df[['Q24_Encoded']]
```

Feature Selection Method 2:

Random Forest Classifier: The random forest is a model made up of many decision trees. It does not simply average the predicition of trees (which we could call a “forest”), this model uses two key concepts that gives it the name random:

- Random sampling of training data points when building trees
- Random subsets of features considered when splitting nodes

https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76 (https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76)

In [53]:
```
classifier = SelectFromModel(RandomForestClassifier(random_state=42))
classifier.fit(dfx, dfy)
classifier.get_support()
```

Out[53]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True, False, False,  True, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True, False, False,  True, False, False, False,
       False, False, False, False,  True, False,  True,  True,  True,
        True,  True, False, False, False, False, False,  True,  True,
        True,  True, False, False,  True,  True, False, False, False,
       False, False, False,  True,  True,  True,  True, False,  True,
        True, False,  True, False,  True,  True,  True,  True, False,
       False, False, False, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True,  True,  True,  True,  True, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False,  True, False,  True,
        True, False,  True,  True,  True,  True,  True, False,  True,
        True,  True, False,  True,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
        True,  True,  True, False,  True, False, False, False,  True,
       False,  True, False,  True, False,  True,  True,  True,  True,
       False,  True, False,  True, False, False,  True,  True,  True,
        True,  True,  True, False,  True, False, False,  True, False,
        True,  True, False])
```

In [54]:
```
selected_features= dfx.columns[(classifier.get_support())]
len(selected_features)
```

Out[54]: 107

In [55]:
```
selected_features
```

Out[55]:
```
Index(['Q1', 'Q6', 'Q13', 'Q15', 'Q20', 'Q21', 'Q25', 'Q7_Part_1_Python',
       'Q7_Part_2_R', 'Q7_Part_3_SQL',
       ...
       'Q22_I do not know', 'Q22_No (we do not use ML methods)',
       'Q22_We are exploring ML methods (and may one day put a model into pro
duction)',
       'Q22_We have well established ML methods (i.e., models in production f
or more than 2 years)',
       'Q22_We recently started using ML methods (i.e., models in production
for less than 2 years)',
       'Q22_We use ML methods for generating insights (but do not put working
models into production)',
       'Q38_Basic statistical software (Microsoft Excel, Google Sheets, et
c.)',
       'Q38_Local development environments (RStudio, JupyterLab, etc.)',
       'Q3_C1', 'Q3_C2'],
      dtype='object', length=107)
```

In [56]:
```python
slfeatures = selected_features.values.tolist()
print(slfeatures)
```

```
['Q1', 'Q6', 'Q13', 'Q15', 'Q20', 'Q21', 'Q25', 'Q7_Part_1_Python', 'Q7_Part_
2_R', 'Q7_Part_3_SQL', 'Q7_Part_4_C', 'Q7_Part_5_C++', 'Q7_Part_6_Java', 'Q7_
Part_7_Javascript', 'Q7_Part_10_Bash', 'Q9_Part_1_Jupyter (JupyterLab, Jupyte
r Notebooks, etc) ', 'Q9_Part_2_ RStudio ', 'Q9_Part_3_Visual Studio', 'Q9_Pa
rt_4_Visual Studio Code (VSCode)', 'Q9_Part_5_ PyCharm ', 'Q9_Part_6_  Spyder
', 'Q9_Part_7_  Notepad++  ', 'Q9_Part_8_  Sublime Text  ', 'Q9_Part_9_  Vim
/ Emacs  ', 'Q10_Part_1_ Kaggle Notebooks', 'Q10_Part_2_Colab Notebooks', 'Q1
0_Part_5_ Binder / JupyterHub ', 'Q12_Part_1_GPUs', 'Q14_Part_1_ Matplotlib
', 'Q14_Part_2_ Seaborn ', 'Q14_Part_3_ Plotly / Plotly Express ', 'Q14_Part_
4_ Ggplot / ggplot2 ', 'Q14_Part_5_ Shiny ', 'Q16_Part_1_  Scikit-learn ', 'Q
16_Part_2_  TensorFlow ', 'Q16_Part_3_ Keras ', 'Q16_Part_4_ PyTorch ', 'Q16_
Part_7_ Xgboost ', 'Q16_Part_8_ LightGBM ', 'Q17_Part_1_Linear or Logistic Re
gression', 'Q17_Part_2_Decision Trees or Random Forests', 'Q17_Part_3_Gradien
t Boosting Machines (xgboost, lightgbm, etc)', 'Q17_Part_4_Bayesian Approache
s', 'Q17_Part_6_Dense Neural Networks (MLPs, etc)', 'Q17_Part_7_Convolutional
Neural Networks', 'Q17_Part_9_Recurrent Neural Networks', 'Q18_Part_1_General
purpose image/video tools (PIL, cv2, skimage, etc)', 'Q18_Part_2_Image segmen
tation methods (U-Net, Mask R-CNN, etc)', 'Q18_Part_3_Object detection method
s (YOLOv3, RetinaNet, etc)', 'Q18_Part_4_Image classification and other gener
al purpose networks (VGG, Inception, ResNet, ResNeXt, NASNet, EfficientNet, e
tc)', 'Q23_Part_1_Analyze and understand data to influence product or busines
s decisions', 'Q23_Part_2_Build and/or run the data infrastructure that my bu
siness uses for storing, analyzing, and operationalizing data', 'Q23_Part_3_B
uild prototypes to explore applying machine learning to new areas', 'Q23_Part
_4_Build and/or run a machine learning service that operationally improves my
product or workflows', 'Q23_Part_5_Experimentation and iteration to improve e
xisting ML models', 'Q23_Part_6_Do research that advances the state of the ar
t of machine learning', 'Q26_A_Part_1_ Amazon Web Services (AWS) ', 'Q26_A_Pa
rt_2_ Microsoft Azure ', 'Q26_A_Part_3_ Google Cloud Platform (GCP) ', 'Q27_A
_Part_1_ Amazon EC2 ', 'Q29_A_Part_1_MySQL ', 'Q29_A_Part_2_PostgresSQL ', 'Q
29_A_Part_3_SQLite ', 'Q29_A_Part_4_Oracle Database ', 'Q29_A_Part_5_MongoDB
', 'Q29_A_Part_8_Microsoft SQL Server ', 'Q31_A_Part_2_Microsoft Power BI',
'Q31_A_Part_5_Tableau', 'Q35_A_Part_5_ TensorBoard ', 'Q36_Part_4_ GitHub ',
'Q36_Part_6_ Kaggle ', 'Q36_Part_7_ Colab ', 'Q36_Part_9_I do not share my wo
rk publicly', 'Q37_Part_1_Coursera', 'Q37_Part_2_edX', 'Q37_Part_3_Kaggle Lea
rn Courses', 'Q37_Part_4_DataCamp', 'Q37_Part_6_Udacity', 'Q37_Part_7_Udemy',
'Q37_Part_8_LinkedIn Learning', 'Q37_Part_10_University Courses (resulting in
a university degree)', 'Q26_B_Part_1_ Amazon Web Services (AWS) ', 'Q29_B_Par
t_1_MySQL ', 'Q2_Man', 'Q2_Woman', 'Q4_Bachelor’s degree', 'Q4_Doctoral degre
e', 'Q4_Master’s degree', 'Q5_Business Analyst', 'Q5_Data Analyst', 'Q5_Data
Scientist', 'Q5_Other', 'Q5_Product/Project Manager', 'Q5_Research Scientis
t', 'Q5_Software Engineer', 'Q11_A cloud computing platform (AWS, Azure, GCP,
hosted notebooks, etc)', 'Q11_A personal computer or laptop', 'Q22_I do not k
now', 'Q22_No (we do not use ML methods)', 'Q22_We are exploring ML methods
(and may one day put a model into production)', 'Q22_We have well established
ML methods (i.e., models in production for more than 2 years)', 'Q22_We recen
tly started using ML methods (i.e., models in production for less than 2 year
s)', 'Q22_We use ML methods for generating insights (but do not put working m
odels into production)', 'Q38_Basic statistical software (Microsoft Excel, Go
ogle Sheets, etc.)', 'Q38_Local development environments (RStudio, JupyterLa
b, etc.)', 'Q3_C1', 'Q3_C2']
```

There are 107 features selected in the step of features selection.
Therefore, it seems there are many features and need to reduce the dimension of the features selected. Thus, PCA and scaling is performed in further steps

PCA and Scaling

PCA: PCA is effected by scale so you need to scale the features in your data before applying PCA. Using StandardScaler from scikit-learn library to standardize the dataset's features onto unit scale (mean = 0 and variance = 1) which is a requirement for the optimal performance of many machine learning algorithms.
PCA is mainly used to reducee the dimension of dataset and spped up the machine learning algorithm.
[https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60 (https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60)](https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60)

In [57]:
```
from sklearn.preprocessing import StandardScaler
features = selected_features          # using the features selected by random
forest classifier
# Separating out the features
x = dfx.loc[:, features].values
x = StandardScaler().fit_transform(x)
```

In [58]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.90,random_state=0)

Xn = pca.fit_transform(x)
```

In [59]:
```
Xn = pd.DataFrame(Xn)
Xn.shape
```

Out[59]:
```
(10729, 77)
```

In [60]:
```
Yn = pd.DataFrame(dfy)
Yn.shape
```

Out[60]:
```
(10729, 1)
```

Model Implementation

Implementing ordinal logistic regression model on the training set separated before.

In [61]:
```
model = LogisticRegression()

X_train, X_test, y_train, y_test = train_test_split(Xn, Yn, test_size=0.30, ra
ndom_state=42)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(7510, 77) (7510, 1)
(3219, 77) (3219, 1)
```

In [62]:
```
model = LogisticRegression()
scaler = StandardScaler()
kfold = KFold(n_splits=10)
kfold.get_n_splits(X_train)

accuracy = np.zeros(10)
np_idx = 0

for train_idx, test_idx in kfold.split(X_train):
    X_train1, X_test1 = X_train.values[train_idx], X_train.values[test_idx]
    y_train1, y_test1 = y_train.values[train_idx], y_train.values[test_idx]

    X_train1 = scaler.fit_transform(X_train1)
    X_test1 = scaler.transform(X_test1)

    model.fit(X_train1, y_train1)

    predictions = model.predict(X_test1)
    prediction_prob = model.predict_proba(X_test1)

    ACC = accuracy_score(predictions,y_test1)
    accuracy[np_idx] = ACC*100
    np_idx += 1
    print (ACC)


print ("Average Score/mean: {}%({}%)".format(round(np.mean(accuracy),3),(round
(np.std(accuracy),3))))
```

```
0.43142476697736354
0.40745672436750996
0.4380825565912117
0.4047936085219707
0.4167776298268975
0.4167776298268975
0.4194407456724368
0.42876165113182424
0.4167776298268975
0.4474034620505992
Average Score/mean: 42.277%(1.276%)
```

In [63]:
```
test_predict= model.predict(X_test)
accuracy_score(test_predict,y_test)
```

Out[63]:
```
0.391425908667288
```

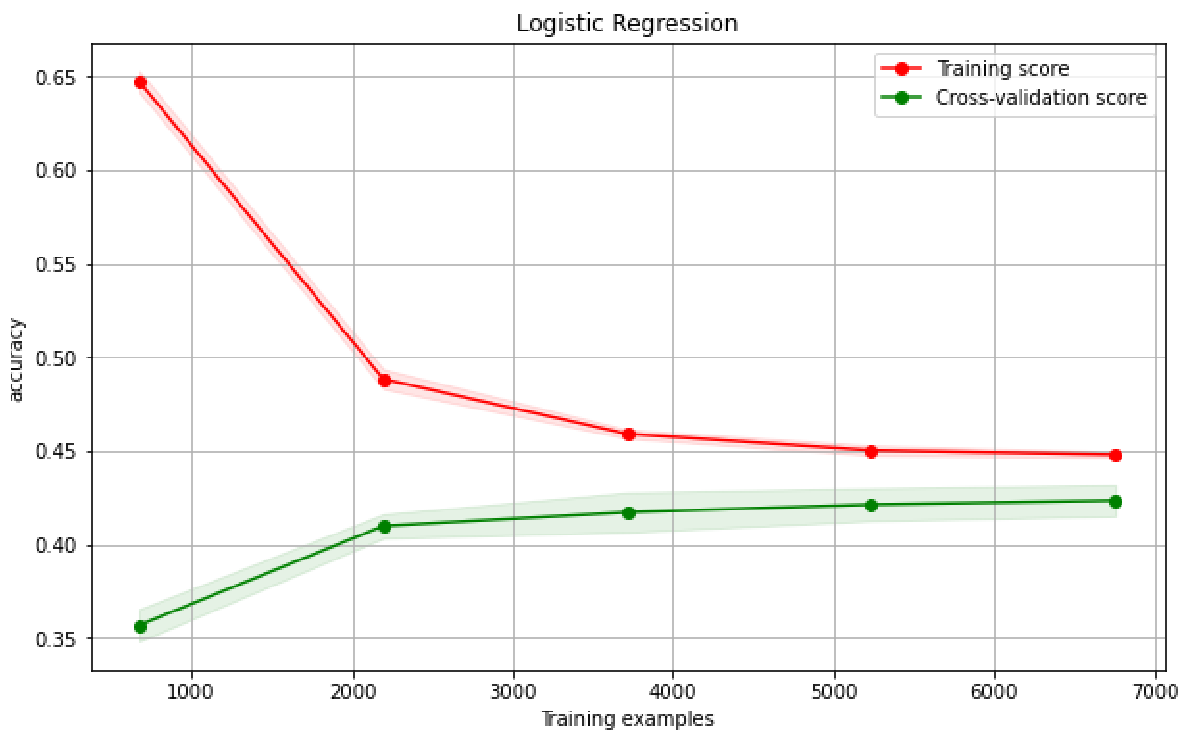
The mean and standard deviation of the accuracy score is 42.277% and 1.276% respectively.
Here, the accuracy score is selected as a metric to measure the performance. This is because the analysis will be simple and easy to accomplish.

When building a model, we want one that can generalize (low bias), and have similar accuracies across testing sets (low variance).

KFold Cross Validation is a common method where the training set is split into k equal sizes. Then of the k subsamples, a single sample is used for testing, and the remaining k-1 samples are used for training. This process continues k times, and each time a different sample is used for testing. This results in each sample being tested once. At the end of this we get 10 accuracies for the model and, from this, we can get the average accuracy, and the standard deviation of the accuracy. The higher the average accuracy, the lower the bias. The lower the standard deviation, the lower the variance. This better represents the true performance of the model on the training set.

In [64]:
```
plot_learning_curve(model,'Logistic Regression', X_train, y_train, cv=10)
```

Out[64]:
```
<module 'matplotlib.pyplot' from '/home/jupyterlab/conda/envs/python/lib/python3.6/site-packages/matplotlib/pyplot.py'>
```



Bias Variance Trade off:

The above learning curve was plotted to check whether the model has high bias or high variance.

Hyperparameter tuning with grid search

The hyperparameters for the model is listed below:

- penalty{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}
- C, default=1.0 :Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- solver{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=‘lbfgs’ :Algorithm to use in the optimization problem.
- intercept_scalingfloat, default=1
- class_weightdict or ‘balanced’, default=None
- random_stateint, RandomState instance, default=None

These are some hyperparameters that can be tuned for study purpose.
In this study, the parameter C and solver is varied as shown in the Raw block below.

The following block for parameter tuning is run and the result is found. Then, the block is converted to Raw type.

model = LogisticRegression() scaler = StandardScaler() kfold = KFold(n_splits=10) kfold.get_n_splits(X_train) best_model = model best_params = {} best_accuracy = 0 best_std = 0 for C in [0.001, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100]: #0.001, 0.01, 0.05, 0.1, 0.5 1, 5, 10, 100 for solver in ['sag', 'saga', 'newton-cg', 'lbfgs']: #'newton-cg','lbfgs', 'sag', 'saga' model = LogisticRegression(C=C, solver=solver) accuracy = np.zeros(10) np_idx = 0 for train_idx, test_idx in kfold.split(X_train): X_train2, X_test2 = X_train.values[train_idx], X_train.values[test_idx] y_train2, y_test2 = y_train.values[train_idx], y_train.values[test_idx] X_train2 = scaler.fit_transform(X_train2) X_test2 = scaler.transform(X_test2) model.fit(X_train2, y_train2) predictions = model.predict(X_test2) prediction_prob = model.predict_proba(X_test2) ACC = accuracy_score(predictions,y_test2) # np_idx += 1 # TN = confusion_matrix(y_test2, predictions)[0][0] # FP = confusion_matrix(y_test2, predictions)[0][1] # FN = confusion_matrix(y_test2, predictions)[1][0] # TP = confusion_matrix(y_test2, predictions)[1][1] # total = TN + FP + FN + TP # ACC = (TP + TN) / float(total) accuracy[np_idx] = ACC*100 np_idx += 1 if np.mean(accuracy) > best_accuracy: best_model = model best_params = {'C':C, 'solver':solver} best_accuracy = np.mean(accuracy) best_std = np.std(accuracy) print (best_params) print ("Best Score: {}% ({}%)".format(round(best_accuracy,3),round(best_std,3))) # 0.01 sag print ("\nThe optimal log model uses C={}, and a {} solver, and has a cross validation score of {}% with a standard deviation of {}%".format(best_params['C'],best_params['solver'],round(best_accuracy,3),round(best_std,3)))

Implementing new optimal model

The following block will implement the new parameters to the training set of the model.

In [65]:
```
import numpy as np
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

model = LogisticRegression(C=0.01, solver='sag')
scaler = StandardScaler()
kfold = KFold(n_splits=10)
kfold.get_n_splits(X_train)

accuracy = np.zeros(10)
np_idx = 0

for train_idx, test_idx in kfold.split(X_train):
    X_train1, X_test1 = X_train.values[train_idx], X_train.values[test_idx]
    y_train1, y_test1 = y_train.values[train_idx], y_train.values[test_idx]

    X_train1 = scaler.fit_transform(X_train1)
    X_test1 = scaler.transform(X_test1)

    model.fit(X_train1, y_train1)

    predictions = model.predict(X_test1)
    prediction_prob1 = model.predict_proba(X_test1)

    ACC = accuracy_score(predictions,y_test1)
    accuracy[np_idx] = ACC*100
    np_idx += 1

    print (ACC)


print ("Average Score/mean: {}%({}%)".format(round(np.mean(accuracy),3),round(
np.std(accuracy),3)))
```

```
0.4420772303595206
0.41544607190412786
0.43142476697736354
0.4047936085219707
0.4207723035952064
0.43009320905459386
0.42609853528628494
0.4340878828229028
0.42609853528628494
0.4434087882822903
Average Score/mean: 42.743%(1.114%)
```

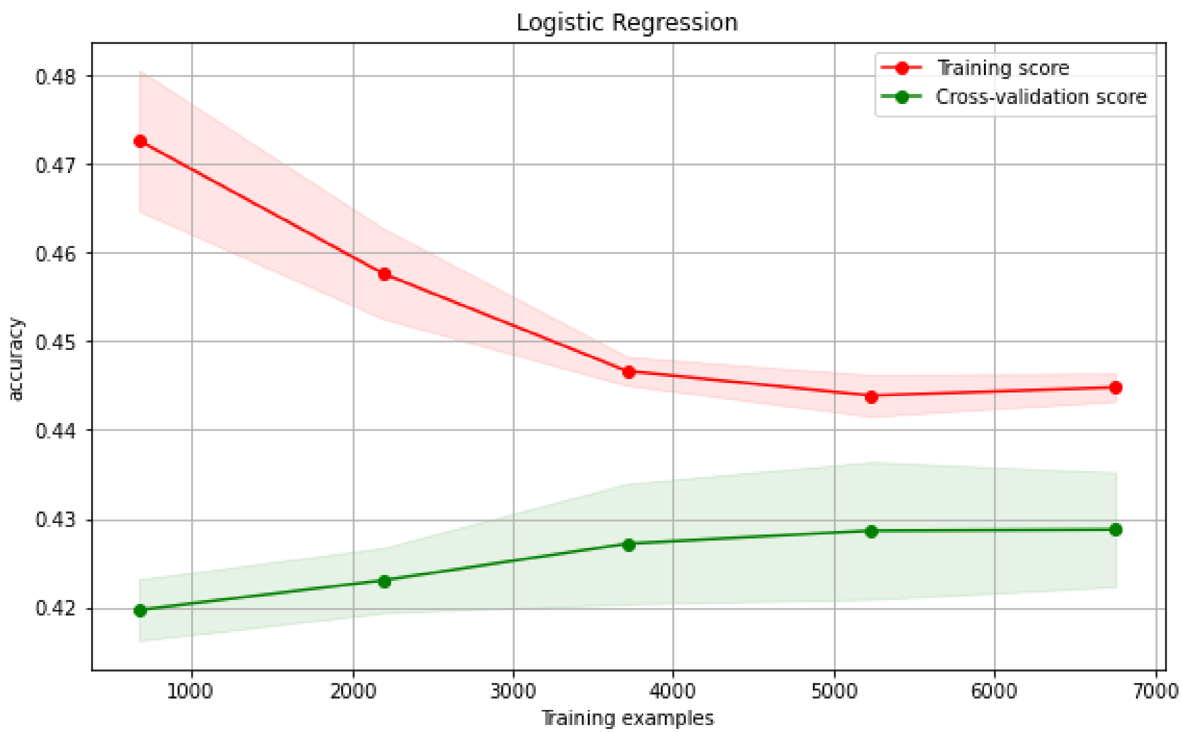
In [66]:
```
test_predict= model.predict(X_test)
accuracy_score(test_predict,y_test)
```

Out[66]:
```
0.40478409443926683
```

Observation: The average score with the optimized parameters is 42.743% while that for the original one is 42.277%. Therefore, the score is improved slightly.
Whereas, the standard deviation is slightly lowered to the value of 1.114% from the value of 1.276%.

In [67]:
```
plot_learning_curve(model,'Logistic Regression', X_train, y_train, cv=10)
```

Out[67]:
```
<module 'matplotlib.pyplot' from '/home/jupyterlab/conda/envs/python/lib/pyth
on3.6/site-packages/matplotlib/pyplot.py'>
```



Optimal model with xtest

Now the model is run for the testing dataset to see the output performance of the model.

In [68]:
```
model = LogisticRegression(C=0.01, solver='sag')
scaler = StandardScaler()
kfold = KFold(n_splits=10)
kfold.get_n_splits(X_test)

accuracy = np.zeros(10)
np_idx = 0

for train_idx, test_idx in kfold.split(X_test):
    X_train1, X_test1 = X_train.values[train_idx], X_train.values[test_idx]
    y_train1, y_test1 = y_train.values[train_idx], y_train.values[test_idx]

    X_train1 = scaler.fit_transform(X_train1)
    X_test1 = scaler.transform(X_test1)

    model.fit(X_train1, y_train1)

    predictions = model.predict(X_test1)
    prediction_prob2 = model.predict_proba(X_test1)

    ACC = accuracy_score(predictions,y_test1)
    accuracy[np_idx] = ACC*100
    np_idx += 1

    print (ACC)


print ("Average Score/mean: {}%({}%)".format(round(np.mean(accuracy),3),round(
np.std(accuracy),3)))
```

```
0.4503105590062112
0.40993788819875776
0.43167701863354035
0.3695652173913043
0.4254658385093168
0.4472049689440994
0.4161490683229814
0.40372670807453415
0.40993788819875776
0.3925233644859813
Average Score/mean: 41.565%(2.327%)
```

Testing and Discussion:

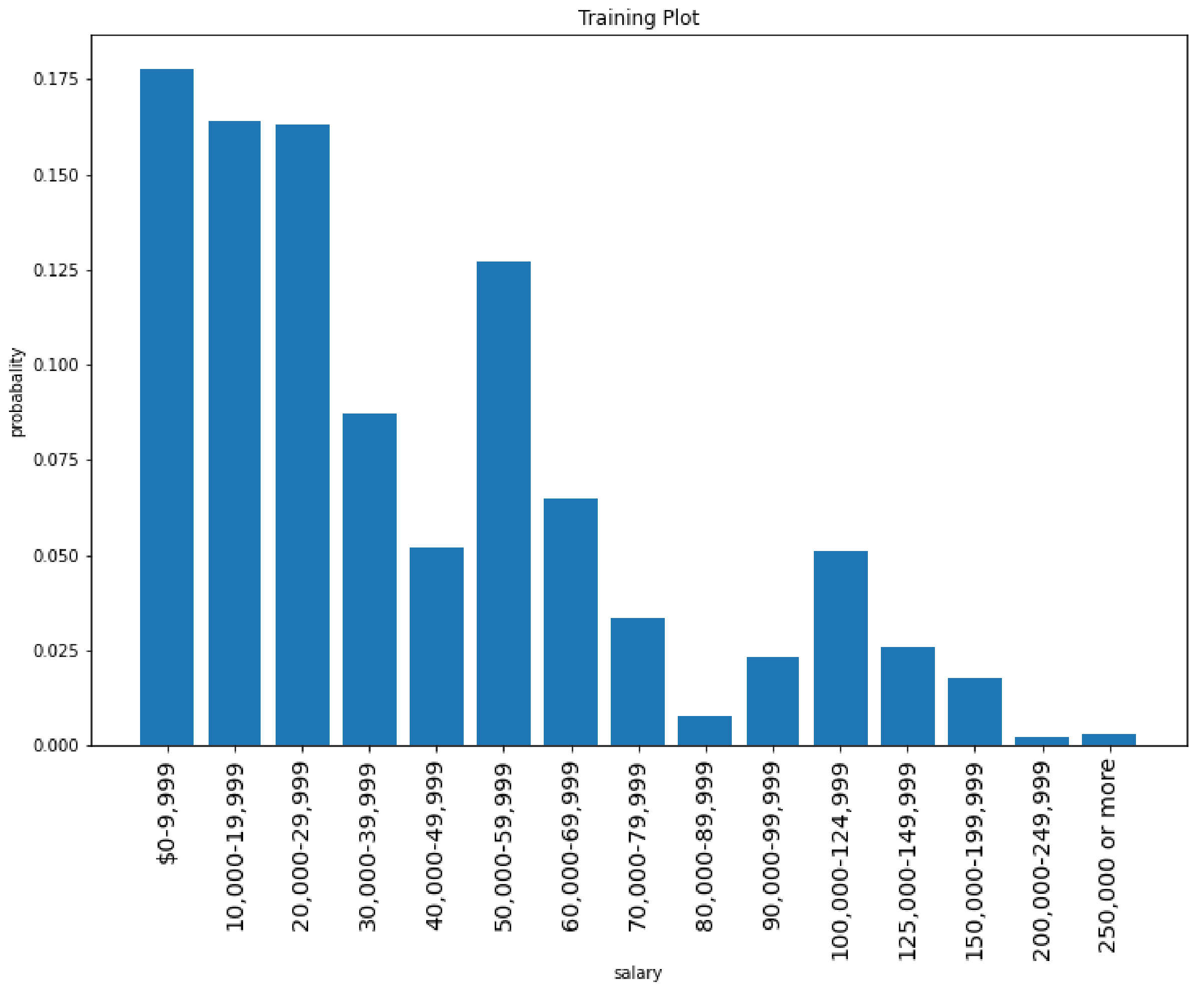
The model does not look overfitting or underfitting. It is a good fit. The test data score was around 41.565% while that of train data score is 42.743%. This is because of the ratio for training and testing dataset split. If more features are included then the accuracy can be increased. The probability of training and testing set was been plotted with respect to the salary

In [69]:
```
print(prediction_prob)
print(prediction_prob1)
```

```
[[0.1060254  0.0731801  0.07344407 ... 0.04526051 0.01879255 0.02896621]
 [0.17778661 0.16413933 0.1630776  ... 0.01758905 0.00225849 0.00296225]
 [0.18348275 0.20670725 0.11203677 ... 0.01008983 0.00202845 0.00769276]
 ...
 [0.03202061 0.05923028 0.05926194 ... 0.13817988 0.01036864 0.03957216]
 [0.17937954 0.20475789 0.04710881 ... 0.03001754 0.01412644 0.00498171]
 [0.35403341 0.14073924 0.05532429 ... 0.00364104 0.00119275 0.00386412]]
[[0.14134391 0.08401826 0.07771126 ... 0.04847339 0.01856671 0.02717593]
 [0.20427477 0.15778232 0.14046208 ... 0.02672031 0.00579981 0.00733136]
 [0.20995527 0.185065   0.09925544 ... 0.02297156 0.00764666 0.01389922]
 ...
 [0.05485318 0.07174313 0.06714375 ... 0.12688866 0.01684764 0.03141113]
 [0.21010422 0.18435205 0.05072987 ... 0.03709195 0.01942869 0.01270885]
 [0.36057881 0.13705048 0.05881052 ... 0.008021   0.00477022 0.00999506]]
```

In [70]:
```
columns =  '$0-9,999', '10,000-19,999', '20,000-29,999', '30,000-39,999', '40,000-49,999', '50,000-59,999', '60,000-69,999', '70,000-79,999', '80,000-89,999', '90,000-99,999', '100,000-124,999', '125,000-149,999', '150,000-199,999', '200,000-249,999', '250,000 or more'
```

In [71]:
```
train_plt = pd.DataFrame(prediction_prob, columns = columns)

plt.figure(figsize = (12,8))
plt.bar(columns, train_plt.loc[1])
plt.xlabel('salary')
plt.ylabel('probabality')
plt.xticks(rotation=90,fontsize='x-large')
plt.title('Training Plot')
plt.show()
```



In [72]:
```
test_plt = pd.DataFrame(prediction_prob1, columns = columns)
plt.figure(figsize = (12,8))
plt.bar(columns, test_plt.loc[1])
plt.xlabel('salary')
plt.ylabel('probabality')
plt.xticks(rotation=90,fontsize='x-large')
plt.title('Testing Plot')
plt.show()
```