Data Analysis of the Data Science Job Posting On GlassDoor

Table of Contents

Introduction

Project Background:

The Data Science field has experienced tremendous growth over the past decade. Companies across various industries, including technology, finance, healthcare, and retail, are increasingly leveraging data to make informed decisions. As a result, Data Professionals, who can turn raw data into actionable insights, have become one of the most in-demand professionals globally.

Within this expansive field, a myriad of titles and positions exist, reflecting a spectrum of responsibilities. Examples include Data Analysts, Data Scientists, Data Engineers, and Machine Learning Scientists. While this project does not intend to clarify the precise definitions of these roles, it aims to provide a comprehensive snapshot of the current Data Science job market, as seen through the job postings on Glassdoor.

Additionally, I hope that the findings of this project will not only assist me but also aid other job seekers in uncovering valuable insights for navigating the data science job market.

Primary Objective:

This project analyzes a dataset of Data Science job postings from Glassdoor to uncover key insights into the job market. The analysis uses various visualizations to explore aspects such as salary estimates, company sizes, and geographical distribution of job postings.

Data Set:

The dataset contains 672 observations and 14 variables, each observation represents a unique job posting for a Data Science-related position on Glassdoor, a popular job search website. The variables provide detailed information about each job posting.

Link: https://www.kaggle.com/datasets/rashikrahmanpritom/data-science-job-posting-on-glassdoor

Methodology:

1. Data Cleaning:

- Refine the 'Job Title' column, consolidating redundant entries into 8 standard, commonly used titles.
- Split the 'Salary Estimate' column into 'Salary Lower Bound', 'Salary Upper Bound', and 'Salary Midpoint'.
- Transform the 'Location' column into separate 'Country', 'State', and 'City' columns, standardizing states with their abbreviations.
- Perform essential cleaning procedures on other columns, tailored to their respective values and data types.

2. Exploratory Data Analysis (EDA):

- Show the distribution of key variables, such as job titles, company sizes, and business sectors.
- Visualize the geographical distribution of job postings across the United States and their average salary midpoints.
- Utilize various graphs to explore the relationships between average salary midpoints and other company characteristics.

Data Preparation and Cleaning:

Importation:

In [8]:
```
# import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import fuzzywuzzy
from fuzzywuzzy import process
```

```python
import re
import geopandas as gpd
import plotly.express as px


# some display settings:
pd.set_option('display.max_columns', 100)


# load the data
data = pd.read_csv("Uncleaned_DS_jobs.csv", index_col='index')
```

Understand the data:

In [9]:
```python
data.head(10)
```

Out[9]:

	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Hea
index							
0	Sr Data Scientist	$137K-$ 171K (Glassdoor est.)	Description\n\nThe Senior Data Scientist is re...	3.1	Healthfirst\n3.1	New York, NY	Ne
1	Data Scientist	$137K-$ 171K (Glassdoor est.)	Secure our Nation, Ignite your Future\n\nJoin ...	4.2	ManTech\n4.2	Chantilly, VA	He
2	Data Scientist	$137K-$ 171K (Glassdoor est.)	Overview\n\n\nAnalysis Group is one of the lar...	3.8	Analysis Group\n3.8	Boston, MA	E
3	Data Scientist	$137K-$ 171K (Glassdoor est.)	JOB DESCRIPTION:\n\nDo you have a passion for ...	3.5	INFICON\n3.5	Newton, MA	E S
4	Data Scientist	$137K-$ 171K (Glassdoor est.)	Data Scientist\nAffinity Solutions / Marketing...	2.9	Affinity Solutions\n2.9	New York, NY	Ne
5	Data Scientist	$137K-$ 171K (Glassdoor est.)	About Us:\n\nHeadquartered in beautiful Santa ...	4.2	HG Insights\n4.2	Santa Barbara, CA	B
6	Data Scientist / Machine Learning Expert	$137K-$ 171K (Glassdoor est.)	Posting Title\nData Scientist / Machine Learni...	3.9	Novartis\n3.9	Cambridge, MA	S
7	Data Scientist	$137K-$ 171K (Glassdoor est.)	Introduction\n\nHave you always wanted to run ...	3.5	iRobot\n3.5	Bedford, MA	Be
8	Staff Data Scientist - Analytics	$137K-$ 171K (Glassdoor est.)	Intuit is seeking a Staff Data Scientist to co...	4.4	Intuit - Data\n4.4	San Diego, CA	
9	Data Scientist	$137K-$ 171K (Glassdoor est.)	Ready to write the best chapter of your career...	3.6	XSELL Technologies\n3.6	Chicago, IL	C

In [10]:
```
data.shape
```

Out[10]:
```
(672, 14)
```

In [11]:
```
data.dtypes
```

Out[11]:
```
Job Title              object
Salary Estimate        object
Job Description        object
Rating                float64
Company Name           object
Location               object
Headquarters           object
Size                   object
Founded                 int64
Type of ownership      object
Industry               object
Sector                 object
Revenue                object
Competitors            object
dtype: object
```

In [12]:
```
data.describe(include = 'all')
```

Out[12]:

	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters	
count	672	672	672	672.000000	672	672	672	
unique	172	30	489	NaN	432	207	229	
top	Data Scientist	$79K-$ 131K (Glassdoor est.)	Job Overview: The Data Scientist is a key memb...	NaN	Hatch Data Inc	San Francisco, CA	New York, NY	e
freq	337	32	12	NaN	12	69	33	
mean	NaN	NaN	NaN	3.518601	NaN	NaN	NaN	
std	NaN	NaN	NaN	1.410329	NaN	NaN	NaN	
min	NaN	NaN	NaN	-1.000000	NaN	NaN	NaN	
25%	NaN	NaN	NaN	3.300000	NaN	NaN	NaN	
50%	NaN	NaN	NaN	3.800000	NaN	NaN	NaN	
75%	NaN	NaN	NaN	4.300000	NaN	NaN	NaN	
max	NaN	NaN	NaN	5.000000	NaN	NaN	NaN	

In [13]:
```
# check data index uniqueness
data.index.duplicated(keep = 'first').sum()
```

Out[13]:
```
0
```

In [14]:
```
# check the number of missing values in each column
data.isna().sum(axis = 0)
```

Out[14]:
```
Job Title            0
Salary Estimate      0
Job Description      0
Rating               0
Company Name         0
Location             0
Headquarters         0
Size                 0
Founded              0
Type of ownership    0
Industry             0
Sector               0
Revenue              0
Competitors          0
dtype: int64
```

We have checked that the data has a non-duplicate index list, and there are no missing values in the data set. Now, let's dive into each column and do some data cleaning.

Data cleaning:

1. Job Title Cloumn

In [15]:
```python
# record the original title just in case
data['old title'] = data['Job Title']

# create a coloum to record whether a title has been integrated
data['integrated title'] = pd.Series(np.zeros(len(data)), dtype = int)

# remove whitespace, change into lower cases
data['Job Title'] = data['Job Title'].str.strip().str.lower()

# check unique values of the titles and their counts
title_counts = data['Job Title'].value_counts()
print(title_counts)

# get the count of different occurance frequencies
title_occurance_freq = title_counts.value_counts().sort_index().to_frame()
title_occurance_freq.index.name = 'occurance frequency'
title_occurance_freq.columns = ['count']
print(title_occurance_freq, '\n'*2)

print('Titles that have an occurance of at least 5 times cover ',
      (title_counts[title_counts >= 5].sum())/data.shape[0]*100,
      '% of the raw data')
```

```
data scientist                                                337
data engineer                                                  26
senior data scientist                                          19
machine learning engineer                                      16
data analyst                                                   12
                                                             ...
data science instructor                                         1
business data analyst                                           1
purification scientist                                          1
data engineer, enterprise analytics                             1
ai/ml - machine learning scientist, siri understanding          1
Name: Job Title, Length: 172, dtype: int64
                     count
occurance frequency
1                       96
2                       55
3                       11
4                        3
5                        1
6                        1
12                       1
16                       1
19                       1
26                       1
337                      1


Titles that have an occurance of at least 5 times cover  62.648809523809526
% of the raw data
```

The raw dataset contains 172 unique titles; however, 96 of these titles appear only once, 55 appear twice, and titles occurring at least five times comprise approximately 62% of our raw data. To ensure sufficient data for each title we analyze, we will retain only those titles that occur five times or more, and we will consolidate the remaining titles into these more frequently occurring categories.

To accomplish this, we will first assess the similarities between the titles that appear less than five times and those that appear at least five times. Based on this assessment, we will then group similar titles under a common title.

In [16]:
```python
# Now we use the extract function from fuzzywuzzy to assess the title
# similarity, it allows you to input a title, and the algorithm will
# find the top n most similar titles each with a similar score.
titles_show_up_5 = title_counts[title_counts >= 5].index.to_list()

for i in titles_show_up_5:
    print('These are the similar titles to ', i, '\n',
          fuzzywuzzy.process.extract(i, data['Job Title'].unique(),
                                     limit = 20,
                                     scorer = fuzzywuzzy.fuzz.token_sort_rat
          '\n'*2)
```

```
These are the similar titles to  data scientist
 [('data scientist', 100), ('sr data scientist', 90), ('(sr.) data scientist
-', 90), ('ai data scientist', 90), ('sr. data scientist', 90), ('data scien
tist - risk', 85), ('lead data scientist', 85), ('data scientist 3 (718)', 8
2), ('staff data scientist', 82), ('sr. data scientist ii', 82), ('senior da
ta scientist', 80), ('data scientist (ts/sci)', 80), ('ai ops data scientis
t', 80), ('data scientist - contract', 76), ('associate data scientist', 7
4), ('principal data scientist', 74), ('ngs scientist', 74), ('geospatial da
ta scientist', 72), ('experienced data scientist', 70), ('human factors scie
ntist', 70)]


These are the similar titles to  data engineer
 [('data engineer', 100), ('jr. data engineer', 90), ('big data engineer', 8
7), ('data engineer - kafka', 81), ('data engineer (remote)', 79), ('senior
data engineer', 79), ('data analyst/engineer', 76), ('software data enginee
r', 74), ('data analytics engineer', 72), ('cloud data engineer (azure)', 6
8), ('staff bi and data engineer', 67), ('data modeler', 64), ('tableau data
engineer 20-0117', 62), ('data science manager', 61), ('sr data scientist',
60), ('(sr.) data scientist -', 60), ('software engineer - data science', 6
0), ('data science software engineer', 60), ('sr. data scientist', 60), ('da
ta scientist', 59)]


These are the similar titles to  senior data scientist
 [('senior data scientist', 100), ('sr data scientist', 89), ('(sr.) data sc
ientist -', 89), ('sr. data scientist', 89), ('senior analyst/data scientis
t', 84), ('sr. data scientist ii', 83), ('data scientist', 80), ('senior dat
a scientist - algorithms', 79), ('staff data scientist', 78), ('data scienti
st (ts/sci)', 76), ('senior data scientist - r&d oncology', 76), ('ai data s
cientist', 74), ('senior data engineer', 73), ('senior data & machine learni
ng scientist', 71), ('data scientist - risk', 70), ('lead data scientist', 7
0), ('senior principal data scientist (python/r)', 69), ('data scientist 3
(718)', 68), ('senior clinical data scientist programmer', 68), ('data scien
tist technical specialist', 68)]


These are the similar titles to  machine learning engineer
 [('machine learning engineer', 100), ('machine learning engineer, sr.', 9
4), ('senior machine learning engineer', 88), ('machine learning engineer/sc
ientist', 83), ('machine learning scientist / engineer', 83), ('data scienti
st(s)/machine learning engineer', 75), ('data scientist / machine learning e
xpert', 63), ('scientist - machine learning', 63), ('data scientist/machine
learning', 61), ('data & machine learning scientist', 61), ('data scientist
- machine learning', 61), ('data scientist machine learning', 61), ('computa
tional scientist, machine learning', 58), ('machine learning scientist - bay
area, ca', 57), ('data scientist, applied machine learning - bay area', 55),
('data integration and modeling engineer', 54), ('senior data & machine lear
ning scientist', 54), ('data engineer (remote)', 53), ('senior data enginee
r', 53), ('principal data scientist - machine learning', 52)]


These are the similar titles to  data analyst
 [('data analyst', 100), ('data analyst i', 92), ('data analyst ii', 89),
('sr data analyst', 89), ('sr. data analyst', 89), ('rfp data analyst', 86),
('global data analyst', 77), ('senior data analyst', 77), ('data science ana
lyst', 75), ('business data analyst', 73), ('data analyst/engineer', 73),
('clinical data analyst', 73), ('e-commerce data analyst', 69), ('operations
data analyst', 69), ('say business data analyst', 65), ('data analytics engi
neer', 63), ('report writer-data analyst', 63), ('analytics manager', 62),
('health plan data analyst, sr', 62), ('senior analyst/data scientist', 59)]


These are the similar titles to  senior data analyst
```

```
 [('senior data analyst', 100), ('sr data analyst', 88), ('sr. data analys
t', 88), ('data analyst i', 85), ('data analyst/engineer', 85), ('data analy
st ii', 82), ('data science analyst', 82), ('rfp data analyst', 80), ('senio
r analyst/data scientist', 79), ('data analyst', 77), ('data analytics engin
eer', 76), ('operations data analyst', 76), ('intelligence data analyst, sen
ior', 75), ('global data analyst', 74), ('data analyst - unilever prestige',
73), ('report writer-data analyst', 71), ('health plan data analyst, sr', 7
0), ('in-line inspection data analyst', 68), ('e-commerce data analyst', 6
7), ('staff data scientist - analytics', 65)]


These are the similar titles to  senior data engineer
 [('senior data engineer', 100), ('jr. data engineer', 83), ('software data
engineer', 81), ('data engineer (remote)', 80), ('data engineer', 79), ('sof
tware engineer - data science', 76), ('data science software engineer', 76),
('senior data scientist', 73), ('data engineer - kafka', 72), ('big data eng
ineer', 70), ('senior data & machine learning scientist', 66), ('staff bi an
d data engineer', 65), ('applied ai scientist / engineer', 65), ('data scien
ce manager', 65), ('data scientist/machine learning', 63), ('data & machine
learning scientist', 63), ('data scientist - machine learning', 63), ('data
engineer, enterprise analytics', 63), ('data analyst/engineer', 63), ('data
scientist machine learning', 63)]
```

We observe that most similar titles differ primarily in their wording or formatting, such as 'Sr. Data Analyst', 'Senior Data Analyst', and 'Experienced Data Analyst'.

To streamline these variations, we propose integrating titles that appear fewer than 5 times using a series of conditional checks. Furthermore, we've identified several management-level titles that don't align with the most frequent titles. To address this, we plan to create a new category, 'Data Science Manager', to encompass all manager-level titles.

In [17]:
```
# Start to integrate the titles that appear less than 5 times, which
# means we will only keep the titles below and integrate others into them.
titles_we_keep = titles_show_up_5.copy()
titles_we_keep.append('data science manager')
titles_we_keep
```

Out[17]:
```
['data scientist',
 'data engineer',
 'senior data scientist',
 'machine learning engineer',
 'data analyst',
 'senior data analyst',
 'senior data engineer',
 'data science manager']
```

In [18]:
```
# Now we use a series of if-else check to consolidate the titles. To do that
# we start by naming some key words to identify certain titles.

senior = ['senior', 'sr', 'experienced', 'ii', 'iii', 'staff']
manager = ['manager', 'management', 'lead', 'principal', 'director',
           'president', 'vp']

for i in range(0, len(data)):
    if data.loc[i, 'Job Title'] not in titles_we_keep:
        data.loc[i, 'integrated title'] = 1
        title = data.loc[i, 'Job Title']

        if any(key in title for key in manager):
            data.loc[i, 'Job Title'] = 'data science manager'
        elif 'machine learning' in title:
            data.loc[i, 'Job Title'] = 'machine learning engineer'
        elif any(key in title for key in senior):
            if 'engineer' in title:
                data.loc[i, 'Job Title'] = 'senior data engineer'
            elif 'analyst' in title:
                data.loc[i, 'Job Title'] = 'senior data analyst'
            elif 'scientist' in title:
                data.loc[i, 'Job Title'] = 'senior data scientist'
        elif 'engineer' in title:
            data.loc[i, 'Job Title'] = 'data engineer'
        elif 'analyst' in title:
            data.loc[i, 'Job Title'] = 'data analyst'
        elif 'scientist' in title:
            data.loc[i, 'Job Title'] = 'data scientist'
```

In [19]:

```
# There are still some titles with unique wordings that are not integrated,
# we can integrate using the following conditional checks.
titles_need_change_manually = data['Job Title'].value_counts().index.to_list

for i in range(0, len(data)):
    if data.loc[i, 'Job Title'] in titles_need_change_manually:
        data.loc[i, 'integrated title'] = 1
        if ('environmental data science' in title or
           'it partner digital health technology and data science' in title)
            data.loc[i, 'Job Title']  = 'data analyst'
        else:
            data.loc[i, 'Job Title']  = 'data scientist'

# see results
data['Job Title'].value_counts()
```

Out[19]:

```
data scientist               421
data engineer                 63
senior data scientist         51
machine learning engineer     45
data analyst                  35
data science manager          30
senior data analyst           20
senior data engineer           7
Name: Job Title, dtype: int64
```

We observe that the conditional checks have effectively managed most of the titles that appear only once or twice, and we further refined the integration manually for the unusual cases. Ultimately, we consolidated all the titles into the 8 common categories we have chosen. For those who wish to validate the process, the following code provides a comparison between the original titles and the integrated ones.

In [20]:

```
#pd.set_option('display.max_rows', None)
#data.loc[data['integrated title'] == 1,  ['Job Title', 'old title']]
```

2. Salary Estimate column

The "salary estimate" column is not numerical, but consists of strings that contain upper and lower limit estimates provided by Glassdoor or employers. For subsequent analysis, it is necessary to extract these lower and upper limits and convert them into integers. Before proceeding with this, we also want to examine the format of this column to ensure there are no unexpected values.

In [21]:
```python
# extract the column
salary = data['Salary Estimate']
print(salary.head(5), '\n'*2)

# there are 20 observations (around 3%) do not contain 'Glassdoor est',
# instead, they contain 'Employer est.'
print(salary[~salary.str.contains('(Glassdoor est.)', regex = False)])

# check that all values have the unit of K/thousand
salary[~salary.str.contains('K')].shape[0]
```

```
index
0    $137K-$171K (Glassdoor est.)
1    $137K-$171K (Glassdoor est.)
2    $137K-$171K (Glassdoor est.)
3    $137K-$171K (Glassdoor est.)
4    $137K-$171K (Glassdoor est.)
Name: Salary Estimate, dtype: object


index
303    $145K-$225K(Employer est.)
304    $145K-$225K(Employer est.)
305    $145K-$225K(Employer est.)
306    $145K-$225K(Employer est.)
307    $145K-$225K(Employer est.)
308    $145K-$225K(Employer est.)
309    $145K-$225K(Employer est.)
310    $145K-$225K(Employer est.)
311    $145K-$225K(Employer est.)
312    $145K-$225K(Employer est.)
313    $145K-$225K(Employer est.)
314    $145K-$225K(Employer est.)
315    $145K-$225K(Employer est.)
316    $145K-$225K(Employer est.)
317    $145K-$225K(Employer est.)
318    $145K-$225K(Employer est.)
319    $145K-$225K(Employer est.)
320    $145K-$225K(Employer est.)
321    $145K-$225K(Employer est.)
322    $145K-$225K(Employer est.)
Name: Salary Estimate, dtype: object
```

Out[21]:
```
0
```

In [22]:
```python
# Now we want to create three new columns, salary estimation's upper limit,
# lower limit, and midpoint. Also remove non-numerical characters
salary_info = (data['Salary Estimate'].str.replace(r'[^\d-]', '', regex=True
               .str.split('-'))
data['salary_lower'] = [int(value[0]) for value in salary_info]
#just practice another way to extract the values
data['salary_upper'] = list(map(lambda x:int(x[1]), salary_info))
data['salary_midpoint'] = (data.salary_lower.astype('int') +
                           data.salary_upper.astype('int')) / 2
```

3. Rating column

One thing needs attention for the 'Rating' column is that there are 50 negative values. It is possible that they are actualy missing values. In order to confirm my thought, I went to Glassdoor.com and searched some companies that have a -1 rating in this data set, and indeed all of them do not have a rating because few number of reviews. So we can replace all the -1 in the Rating column to NaN.

In [23]:
```
print(data.Rating.describe())
print('There are', sum(data.Rating == -1), '"-1" in the Rating column')
data.loc[data.Rating == -1, 'Rating'] = np.nan
```

```
count    672.000000
mean       3.518601
std        1.410329
min       -1.000000
25%        3.300000
50%        3.800000
75%        4.300000
max        5.000000
Name: Rating, dtype: float64
There are 50 "-1" in the Rating column
```

4. Company Name column

For the company name column, the biggest problem we can notice is that each name comes with its rating, which is already recorded in another column. So, we need to get rid of the unnessary postfix.

In [24]:
```
data['Company Name'] = data['Company Name'].str.split('\n').str[0]
```

5. Location column

The location of each job posting is consolidated into a single cell in the 'Location' column. To facilitate our future analysis, I will separate this information into 'City', 'State', and 'Country' columns. For the data consistency, I will use the state abbreviations in our newly created 'state' column. Additionally, I have observed that there are some exceptional values, such as 'remote' or location entries that do not conform to the primary wording format.

In [25]:
```
# create lists to identify the states and country name
US = ['United States', 'USA', 'united states', 'U.S.']
DC = ['Washington, D.C.', 'Washington, DC']

# Map of full state names to their abbreviations
state_name_to_abbreviation = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'Cal
    'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'FL'
    'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL', 'Indiana': 'IN', 'Iowa'
    'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA', 'Maine': 'ME', 'Mar
    'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'MN', 'Mississippi
    'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH'
    'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dak
    'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island':
    'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas'
    'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA', 'West Virginia':
    'Wyoming': 'WY'
}

# create a list that contains both full name and abbreviations
states = list(state_name_to_abbreviation.keys())
state_abbrs = list(state_name_to_abbreviation.values())
states.extend(state_abbrs)
```

In [26]:
```
# split the location into seperate city, state/province, and country column
data = data.assign(city = '', state = '', country = '')
```

```
location_splitted = data.Location.str.strip().str.split(', ')

# some entries in Location column only have state info, while some have
# city, state, and country. So, we need to check the number of element
# in the original location column and then assign the values into our
# new columns
for i in range(len(location_splitted)):
    location  = location_splitted[i]
    number_of_element = len(location)
    if data.Location[i] in DC:
        data.iloc[i, -3:] = ['DC', 'DC', 'United States']
    elif number_of_element == 1:
        if location[0] in ['remote', 'Remote']:
            data.iloc[i, -3:] = ['Remote'] * 3
        elif location[0] in US:
            data.iloc[i, -3:] = ['Unknown', 'Unknown', 'United States']
        elif location[0] in states:
            data.iloc[i, -3:] = ['Unknown', location[0], 'United States']
        else:
            data.iloc[i, -3:] = [np.nan] * 3
    elif number_of_element == 2:
        if location[1] in states:
            data.iloc[i, -3:] = [location[0], location[1], 'United States']
        elif location[0] in states and location[1] in US:
            data.iloc[i, -3:] = ['Unknown', location[0], 'United States']
        else:
            data.iloc[i, -3:] = [np.nan] * 3
    elif number_of_element == 3:
        data.iloc[i, -3:] = [location[0], location[1], location[2]]
    else:
        data.iloc[i, -3:] = [np.nan] * 3


# Then we replace the state full names with their abbreviations.
data['state'] = data['state'].replace(state_name_to_abbreviation)
# Specific replacements, it is a city in MD
data['state'] = data['state'].replace({'Anne Arundel': 'MD'})
```

6. Size column

After examining the unique values in the 'Size' column, I noticed that there are '-1' values. Upon further investigation into the companies that are listed with a size of '-1', I observed that their location, founding date, and other information are also missing. Therefore, it is reasonable to replace the '-1' values in the 'Size' column with 'Unknown'.

In [27]:
```
data.loc[data.Size == '-1', 'Size'] = 'Unknown'
data.Size.value_counts().sort_index()
```

Out[27]:
```
1 to 50 employees            86
10000+ employees             80
1001 to 5000 employees      104
201 to 500 employees         85
5001 to 10000 employees      61
501 to 1000 employees        77
51 to 200 employees         135
Unknown                      44
Name: Size, dtype: int64
```

7. Founded column

In [28]:
```
# Founded column also has -1 value, change them to 'Unknown'
data.loc[data['Founded'] == -1, 'Founded'] = 'Unknown'
data.Founded.value_counts()
```

Out[28]:
```
Unknown    118
2012        34
2011        25
2015        22
2010        22
          ...
1820         1
1952         1
1959         1
1894         1
1962         1
Name: Founded, Length: 103, dtype: int64
```

8. Type of ownership column

For the 'Ownership' column, We can combine the similar types while transfering -1 to Unknown.

In [29]:
```
public_ownership = ['Government']
private_ownership = ['Subsidiary or Business Segment', 'Private Practice / F
non_profit = ['College / University', 'Hospital']
other_ownership = ['Self-employed', 'Contract']

for index, row in data.iterrows():
    ownership = row['Type of ownership']
    if ownership == '-1':
        data.loc[index, 'Type of ownership'] = 'Unknown'
    elif ownership in public_ownership:
        data.loc[index, 'Type of ownership'] = 'Company - Public'
    elif ownership in private_ownership:
        data.loc[index, 'Type of ownership'] = 'Company - Private'
    elif ownership in non_profit:
        data.loc[index, 'Type of ownership'] = 'Nonprofit Organization'
    elif ownership in other_ownership:
        data.loc[index, 'Type of ownership'] = 'Other Organization'

data['Type of ownership'].value_counts()
```

Out[29]:
```
Company - Private         429
Company - Public          163
Nonprofit Organization     40
Unknown                    31
Other Organization          9
Name: Type of ownership, dtype: int64
```

9. Industry column

The 'Industry' column in our dataset records the detailed operational fields of the companies, and it encompasses many unique types. Given the limited size of our dataset, I believe it would be more effective to use the 'Sector' column for our analysis, which is more general and has fewer categories compared to the 'Industry' column.

In [30]:
```
data.loc[data['Industry'] == '-1', 'Industry'] = 'Unknown'
data.Industry.value_counts()
```

```
Out[30]: Unknown                                       71
         Biotech & Pharmaceuticals                     66
         IT Services                                   61
         Computer Hardware & Software                  57
         Aerospace & Defense                           46
         Enterprise Software & Network Solutions       43
         Consulting                                    38
         Staffing & Outsourcing                        36
         Insurance Carriers                            28
         Internet                                      27
         Advertising & Marketing                       23
         Health Care Services & Hospitals              21
         Research & Development                        17
         Federal Agencies                              16
         Investment Banking & Asset Management         13
         Banks & Credit Unions                          8
         Lending                                        8
         Energy                                         5
         Consumer Products Manufacturing                5
         Telecommunications Services                    5
         Insurance Agencies & Brokerages                4
         Food & Beverage Manufacturing                  4
         Utilities                                      3
         Electrical & Electronic Manufacturing          3
         Colleges & Universities                        3
         Other Retail Stores                            3
         Architectural & Engineering Services           3
         Chemical Manufacturing                         3
         Real Estate                                    3
         Miscellaneous Manufacturing                    3
         Accounting                                     3
         Wholesale                                      3
         Industrial Manufacturing                       3
         Video Games                                    3
         Travel Agencies                                2
         Express Delivery Services                      2
         Timber Operations                              2
         Financial Transaction Processing               2
         Construction                                   2
         Health, Beauty, & Fitness                      2
         Transportation Equipment Manufacturing         2
         Oil & Gas Services                             2
         Venture Capital & Private Equity               2
         Consumer Electronics & Appliances Stores       2
         Department, Clothing, & Shoe Stores            1
         Publishing                                     1
         Social Assistance                              1
         Farm Support Services                          1
         Logistics & Supply Chain                       1
         Transportation Management                      1
         State & Regional Agencies                      1
         Hotels, Motels, & Resorts                      1
         Food & Beverage Stores                         1
         News Outlet                                    1
         Telecommunications Manufacturing               1
         Cable, Internet & Telephone Providers          1
         Shipping                                       1
         Rail                                           1
         Name: Industry, dtype: int64
```

10. Revenue column

For the "Revenue" column, a challenge arises from the data being formatted as strings, with values represented in both billions and millions. To sort these values effectively, we will create a new column to record the revenue lower bond in numerical format, and we will use 'million' as the standard unit.

In [31]:
```python
# transfer -1 values to 'Unknown'
data.loc[data['Revenue'] == '-1', 'Revenue'] = 'Unknown / Non-Applicable'
data['revenue_lower_bond'] = 0

digits_in_rev = data['Revenue'].apply(lambda x: re.findall(r'\d+', x))

for index, row in data.iterrows():
    if 'billion' in row['Revenue'] and 'million' not in row['Revenue']:
        data.loc[index, 'revenue_lower_bond'] = (int(digits_in_rev[index][0]
                                                  * 1000)
    # the only special case
    elif row['Revenue'] == 'Less than $1 million (USD)':
        data.loc[index, 'revenue_lower_bond'] = 0.5
    elif 'million' in row['Revenue']:
        data.loc[index, 'revenue_lower_bond'] = int(digits_in_rev[index][0])

# as we can see, the Revenue column now can be sorted using the
# revenue_lower_bond column
(data[['Revenue', 'revenue_lower_bond']]
.loc[~data.duplicated('Revenue')].sort_values('revenue_lower_bond'))
```

Out[31]:

index	Revenue	revenue_lower_bond
0	Unknown / Non-Applicable	0.0
100	Less than $1 million (USD)	0.5
38	$1to5$ million (USD)	1.0
91	$5to10$ million (USD)	5.0
24	$10to25$ million (USD)	10.0
25	$25to50$ million (USD)	25.0
34	$50to100$ million (USD)	50.0
2	$100to500$ million (USD)	100.0
12	$500millionto1$ billion (USD)	500.0
1	$1to2$ billion (USD)	1000.0
8	$2to5$ billion (USD)	2000.0
18	$5to10$ billion (USD)	5000.0
6	$10+ billion (USD)	10000.0

11. Sector column

In [32]:
```python
data.loc[data['Sector'] == '-1', 'Sector'] = 'Unknown'
data.Sector.value_counts()
```

```
Out[32]: Information Technology                    188
         Business Services                         120
         Unknown                                    71
         Biotech & Pharmaceuticals                  66
         Aerospace & Defense                        46
         Finance                                    33
         Insurance                                  32
         Manufacturing                              23
         Health Care                                21
         Government                                 17
         Oil, Gas, Energy & Utilities               10
         Retail                                      7
         Telecommunications                          7
         Transportation & Logistics                  6
         Media                                       5
         Real Estate                                 3
         Travel & Tourism                            3
         Agriculture & Forestry                      3
         Education                                   3
         Accounting & Legal                          3
         Construction, Repair & Maintenance          2
         Consumer Services                           2
         Non-Profit                                  1
         Name: Sector, dtype: int64
```

12. Competitors column

Since there are way too many invalid values, the -1, in the Competitors column, we will just drop it.

```
In [33]: print(data.Competitors.value_counts())
         data.drop(['Competitors'], axis = 1, inplace = True)
```

```
-1                                                            501
Roche, GlaxoSmithKline, Novartis                               10
Los Alamos National Laboratory, Battelle, SRI International     6
Leidos, CACI International, Booz Allen Hamilton                 6
MIT Lincoln Laboratory, Lockheed Martin, Northrop Grumman       3
                                                              ...
Pfizer, GlaxoSmithKline                                         1
Square, Amazon, Apple                                           1
Lumentum Operations, Keysight Technologies, O-Net Technologies  1
Munich Re, Hannover RE, SCOR                                    1
Genomic Health, Myriad Genetics, The Broad Institute            1
Name: Competitors, Length: 108, dtype: int64
```

Now, we have the raw data cleaned, and the data is ready for the exploratory data analysis.

Exploratory Data Analysis

It is always a good practice to understand the data first and try to gather as many insights from it, and Exploratory Data Analysis (EDA) is a very helpful tool for that. EDA analyzes the data and discover trends, patterns, or check assumptions in data with the help of statistical summaries and graphical representations.

In this section, I will begin by posing questions and trying to approach them by various visualizations. Then, with the help of our visualizations, we can identify the hidden patterns, relationships, and trends in the data and use them to facilitate further analysis or formulating hypotheses.

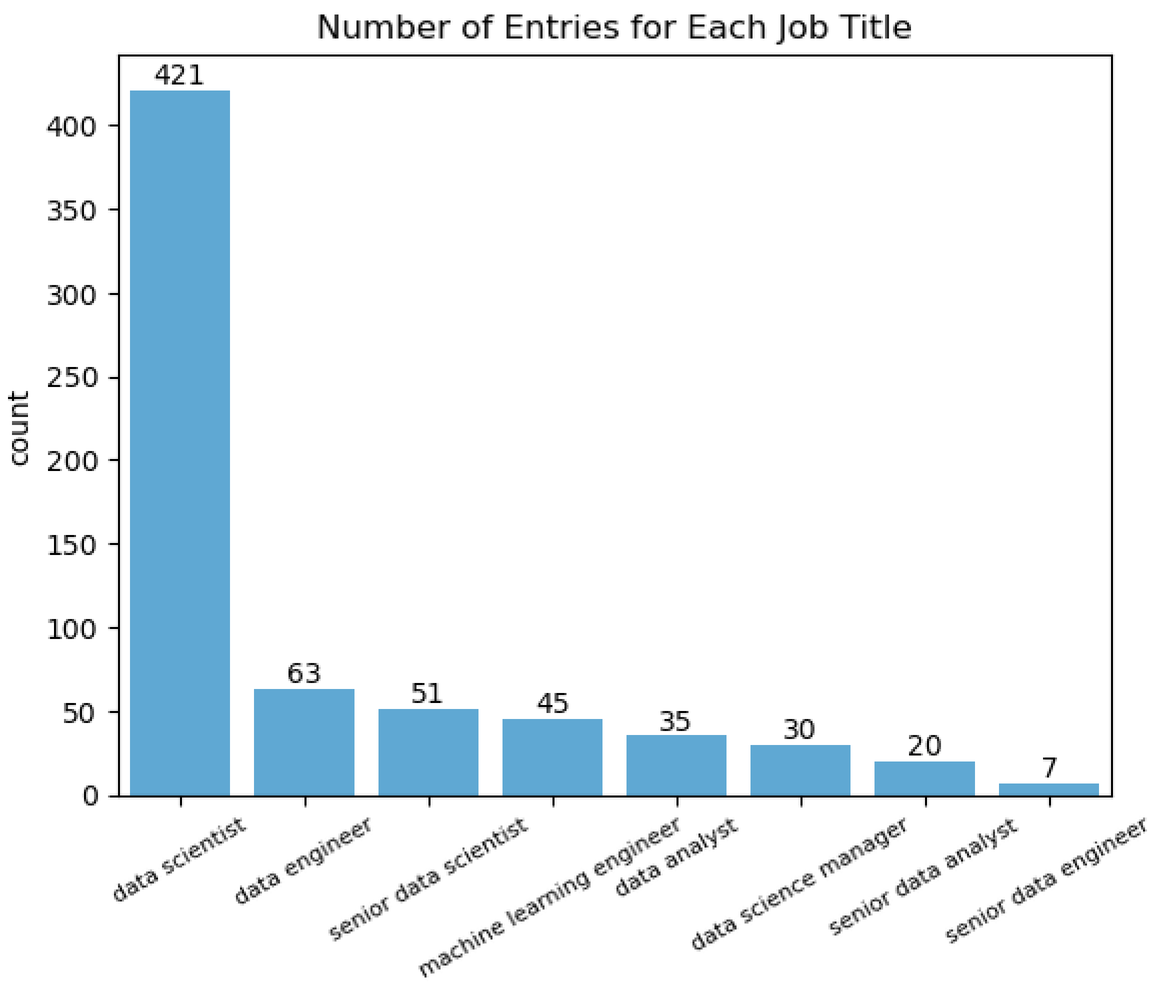
1.How many entries are there for each job title?

```
In [34]: # data preparation
title_count = data['Job Title'].value_counts()

# plot the count of each title
ax = sns.barplot(x = title_count.index,y = title_count, color = '#5fa8d3',
                 saturation = 1)

for bar in ax.patches:
    ax.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height(),
            round(bar.get_height()),
            va = 'bottom', ha = 'center')


ax.set_title('Number of Entries for Each Job Title')
ax.set_ylabel('count')
plt.xticks(rotation = 30, size = 8)
plt.show()
```



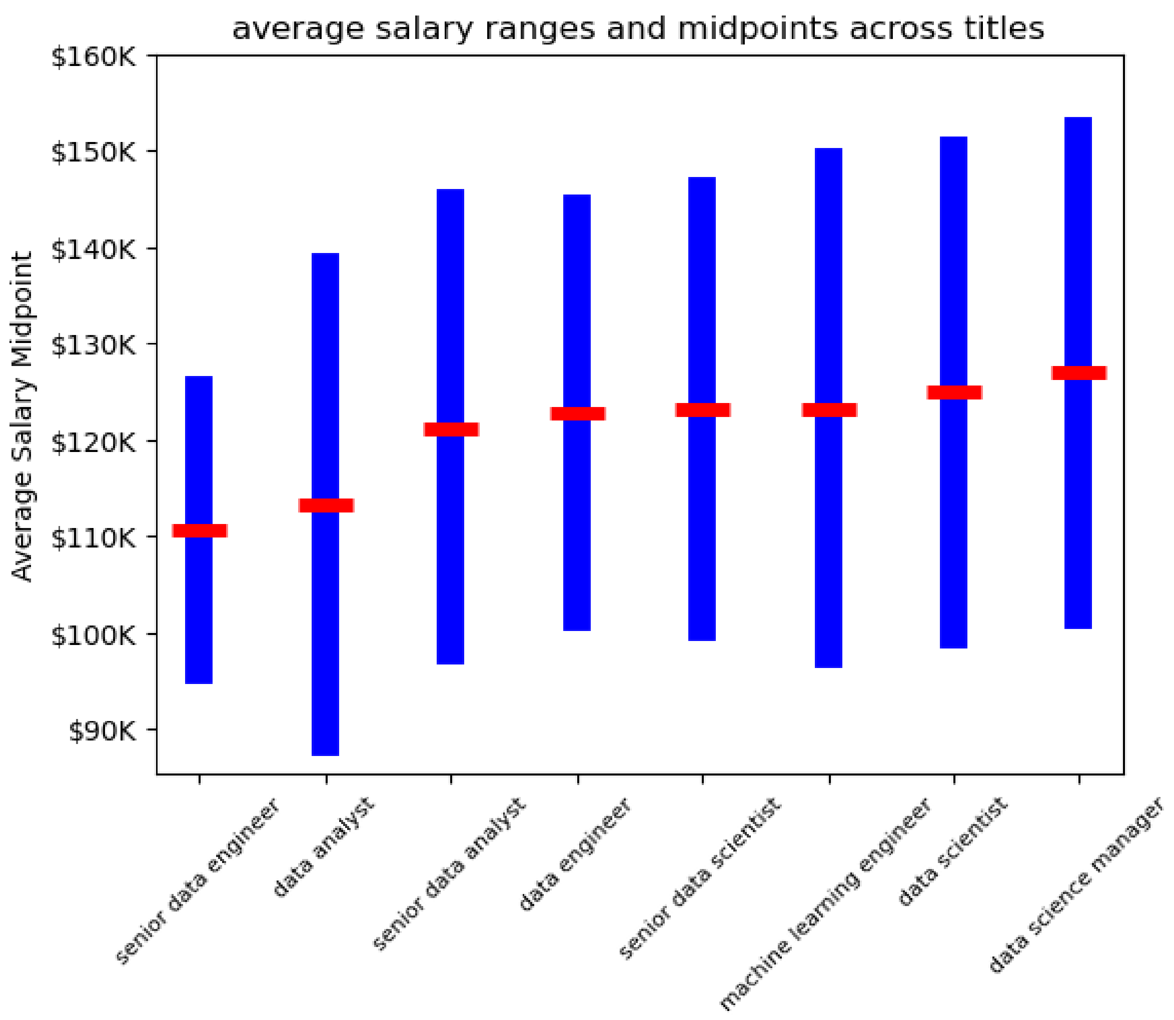
In the bar plot above, it's evident that the Data Scientist role is the most prevalent, with all other titles registering no more than 70 entries. Interestingly, senior positions are relatively rare across all titles, except for the Data Scientist role. It's worth noting that the term 'Data Scientist' is often used broadly, as the responsibilities associated with data roles can vary significantly depending on the company and project.

2.What is the salary distribution across different titles? Which titles have the highest/lowest salary in general?

In [35]:
```
# we can show the average salary midpoint as well as the average lower and u
# salary limits for each job title
salary_info = (data.loc[:, ['Job Title', 'salary_lower',
                            'salary_midpoint', 'salary_upper']]
                   .groupby('Job Title', as_index = False)
                   .agg(np.mean)
                   .round(1)
                   .sort_values(by = 'salary_midpoint')
                   .reset_index(drop = True))

long_salary_info = salary_info.melt(id_vars = ['Job Title'],
                                    var_name = 'type',
                                    value_name = 'value')

# plot the data
fig, ax = plt.subplots()
for title in salary_info['Job Title'].unique():
    current_title_data = long_salary_info[long_salary_info['Job Title'] == t
    plt.plot('Job Title', 'value', data = current_title_data, lw = 10,
             color = 'blue')
    plt.plot(title, np.median(current_title_data['value']),
             color = 'red', marker = '_', ms = 20, mew = 5)
plt.xticks(rotation = 45, size = 8)
ax.set_title('average salary ranges and midpoints across titles')
ax.set_ylabel('Average Salary Midpoint')
ax.set_yticks(range(90, 170, 10))
ax.set_yticklabels(['$' + str(label) + 'K' for label in range(90, 170, 10)])
plt.show()
```



In the displayed graph above, we represent the salary range for each job title, with the salary midpoint highlighted in red. And we can notice some interesting things:

- The Data Analyst position has a notably lower salary range and midpoint compared with other general titles.
- The salary midpoint for the Senior Data Engineer position is lower than that of the Data Engineer. And The salary midpoint and range for the Senior Data Scientist position is really close to that of the Data Scientist. These seem counterintuitive given the usual expectations for senior roles to command higher salaries, as what is shown between the senior data analyst and data analyst. First possible reason is that the sample sizes of these senior positions are too small, as there are only 7 pieces of salary data for the senior data engineer. Secondly, Glassdoor provides salary estimates as ranges, and without insight into the distribution within these ranges, we are not able to access the true salary dynamics.
- The salary midpoint and range for the Data Manager position stand out as they are noticeably higher compared to the other seven titles.

3.How is the distribution of job positions across various regions? Are there any states that stand out with a significantly larger number of data science roles?

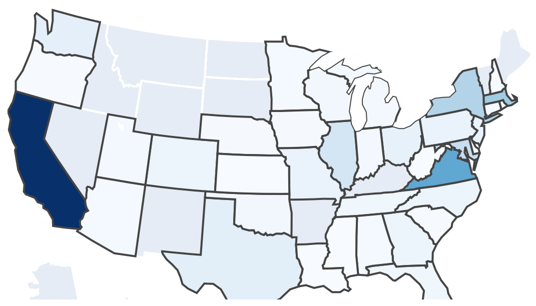
In [36]:
```
# Get the number of job posting in each state
state_counts = data.state.value_counts()
```

```python
# Convert the state counts to a DataFrame
state_counts_df = state_counts.reset_index()
state_counts_df.columns = ['state', 'count']

# Create a choropleth map
fig = px.choropleth(state_counts_df,
                    locations='state',
                    color='count',
                    locationmode='USA-states',
                    scope='usa',
                    title='Number of Data Science Job Postings by State',
                    color_continuous_scale='Blues',
                    labels={'count':'Number of Job Postings'})

fig.show()
```

Number of Data Science Job Postings by State



In the choropleth map above, it's evident that California stands out with a significantly higher number of job postings compared to other states. A moderate number of postings can also be seen in a few East Coast states such as Virginia, New York, and Massachusetts. In contrast, the states in the central region appear to have relatively few job postings or even no job postings.

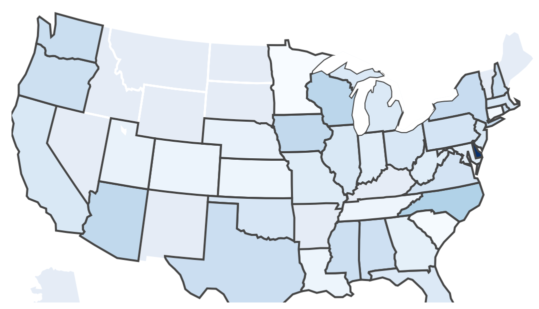
4.What are the average salary midpoints in each state? Which part of the country offers a higher salary on average?

In [37]:
```
state_salary = (data.groupby('state', as_index = False)
                [['state', 'salary_midpoint']]
                .agg({'salary_midpoint':'mean'})).round(1)

fig = px.choropleth(state_salary, locations = 'state',
                    color = 'salary_midpoint',
                    locationmode = 'USA-states',
                    scope = 'usa',
                    title = 'Average Salary Midpoint of Data Science Jobs in
                    color_continuous_scale = 'Blues',
                    labels={'salary_midpoint':
                            'Average Salary Midpoint (thousand dollar)'})
fig.show()

# Delaware is an outlier which is built on a single data point
data[data['state'] == 'DE']
```

Average Salary Midpoint of Data Science Jobs in Each State



Out[37]:

	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters
index							
509	data scientist	$212K-$331K (Glassdoor est.)	Title: Real World Science, Data Scientist\nLoc...	4.0	AstraZeneca	Wilmington, DE	Cambridge, United Kingdom

From the map, it's evident that most central states have comparatively lower average salary midpoints compared with most costal states. For example, New York ,North

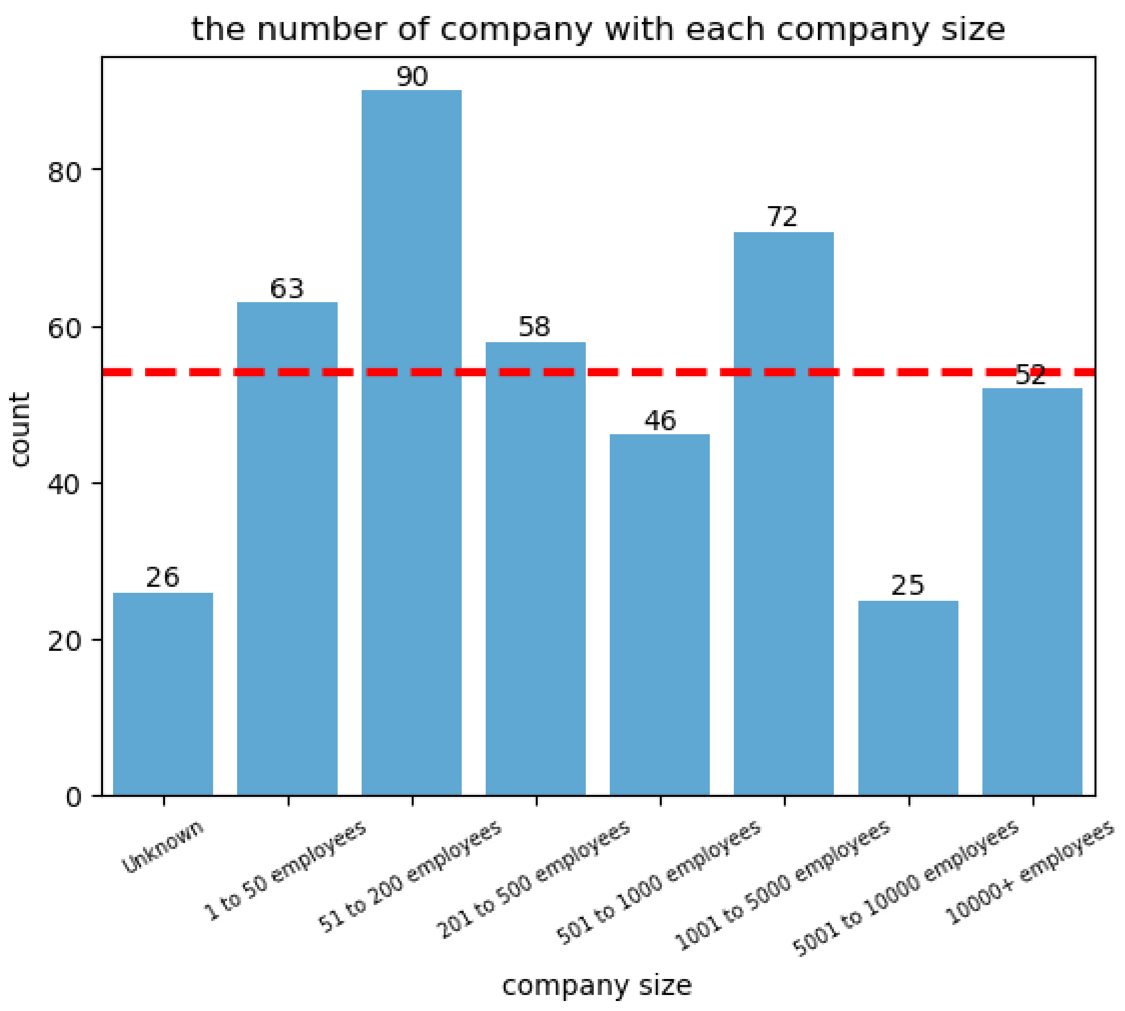
Carolina, Texas, and Washington all have an average salary midpoint over 133K. Delaware is an interesting outlier; despite being based on a single data point, it has the highest average salary midpoint at $271.5K.

5.How is company size distributed in our dataset? Are the majority large or small companies?

In [38]:
```python
# create a helper function to locate the lower bound of each category
def size_sorter(index_values):
    sizes = []
    for size in index_values:
        numbers = re.findall(r"(\d+\.\d|\d+)", size)
        if len(numbers) == 0:
            minimum = 0
        else:
            minimum = min(map(int, numbers))
        sizes.append(minimum)
    return pd.Series(sizes)


# create the company size count df
company_sizes = (data.loc[~data.duplicated('Company Name'), 'Size']
                .value_counts().reset_index())
company_sizes['lower_limit'] = size_sorter(company_sizes['index'])
company_sizes = company_sizes.sort_values('lower_limit')


# plot the bar plot
ax = sns.barplot(data = company_sizes, x = 'index', y = 'Size',
                 color = '#5fa8d3', saturation = 1)
for bar in ax.patches:
    ax.text(bar.get_x() + bar.get_width()/2,
            bar.get_height(),
            f'{round(bar.get_height())}',
            ha = 'center', va = 'bottom')
ax.set_title('the number of company with each company size')
ax.set_xlabel("company size")
ax.set_ylabel("count")
plt.xticks(rotation = 30, size = 7)
# also draw a horizontal line for the average count of company size
ax.axhline(y = company_sizes.Size.mean(), color = 'red',
           linewidth = 3, linestyle = '--')
plt.show()
```

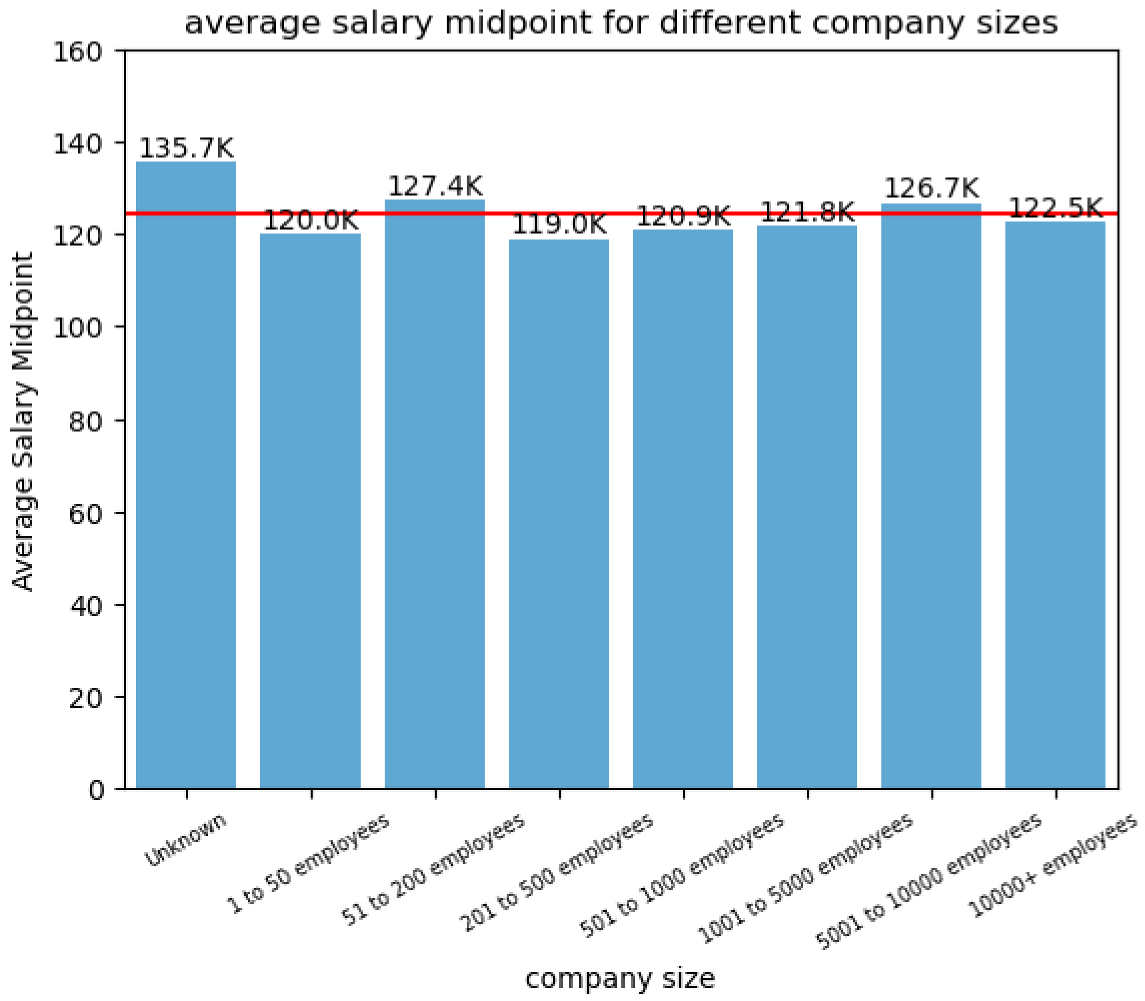


From the graph, we can tell although the distribution of company sizes is not well balanced, companies of all sizes have a demand for data professionals. Notably, 91 out of the 432 companies in this data set employ between 51 to 200 individuals, while a mere 25 companies in this data set have a workforce ranging from 5,001 to 10,000. No clear trend indicates a predominance of larger firms. The limited sample size might account for the absence of a distinct pattern.

6.How do average salary midpoints vary across different company sizes in our dataset? Which type of company offers the highest salary?

In [39]:
```
# prepare the data
company_size_salary = (data.groupby('Size', as_index = False).salary_midpoin
                      .apply(np.mean))
company_size_salary['lower_bound'] = size_sorter(company_size_salary.Size)
company_size_salary.sort_values('lower_bound', inplace = True)

# plot bar chart
ax = sns.barplot(data = company_size_salary, x = 'Size',
                 y = 'salary_midpoint', color = '#5fa8d3',
                 alpha = 1, saturation = 1)
for bar in ax.patches:
    ax.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height(),
            f'{round(bar.get_height(), 1)}K',
            ha = 'center', va = 'bottom')
plt.xticks(rotation = 30, size = 7)
ax.set_title('average salary midpoint for different company sizes')
ax.set_xlabel('company size')
ax.set_ylim([0, 160])
# denote the overall average salary midpoint using the red line
ax.axhline(y = company_size_salary.salary_midpoint.mean(), color = 'red',
           linestyle = '-')
ax.set_ylabel('Average Salary Midpoint')
plt.show()
```



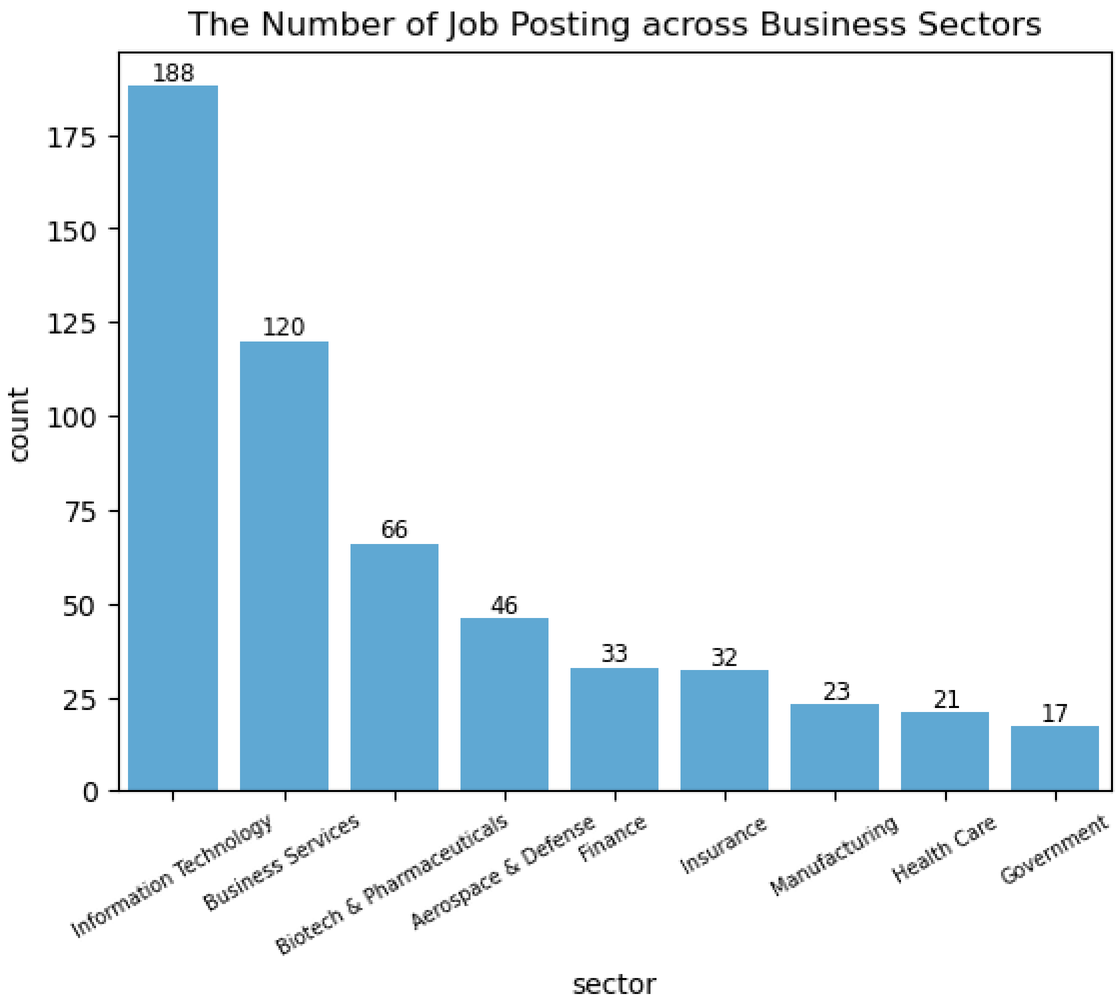
From the graph presented, there is no substantial variation in the average salary midpoints across different company sizes; the figures closely align with the overall average salary midpoint. Consequently, the data does not provide strong evidence to support the hypothesis that larger companies typically offer higher salaries.

7.How many job postings are there in each business sector? Which one on average offer the highest salary?

In [40]:
```
# prepare the data
sector_salary = (data.groupby('Sector', as_index = False)
                     .agg({'salary_midpoint':['mean', 'size']}).round(1))

sector_salary.columns = ['sector', 'salary_midpoint', 'number_of_data']
sector_salary = (sector_salary.query('(number_of_data > 10) & (sector != "Un
                             .sort_values('number_of_data', ascending = Fals


# plot the number of data in each business sector
plt.figure(1)
ax1 = sns.barplot(data = sector_salary, x = 'sector', y = 'number_of_data',
                  color = '#5fa8d3', alpha = 1, saturation = 1)
for bar in ax1.patches:
    ax1.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height(),
            f'{round(bar.get_height())}',
            va = 'bottom', ha = 'center', size = 8.5)
ax1.set_title('The Number of Job Posting across Business Sectors')
ax1.set_ylabel('count')
plt.xticks(rotation = 30, size = 7)
plt.show()
```

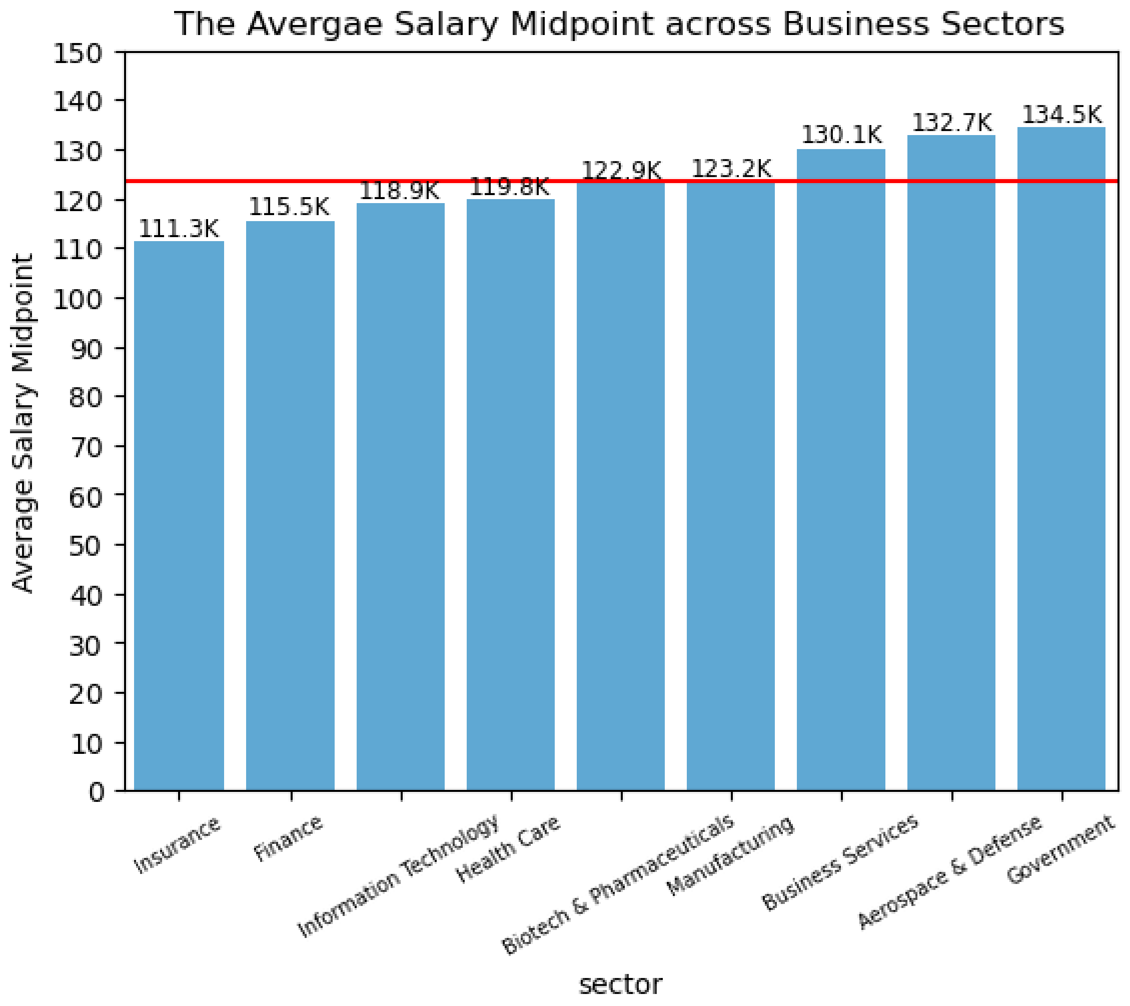


The graph illustrates that the Information and Technology sector is at the forefront, offering a substantial number of job opportunities, followed by the Business Services and Biotech & Pharmaceuticals sectors. Other sectors are shown to have a moderate level of data science job postings, with none exceeding 50 positions.

In [41]:
```
# plot the salary midpoint of each sector
sector_salary = sector_salary.sort_values('salary_midpoint')
plt.figure(2)
ax2 = sns.barplot(data = sector_salary, x = 'sector', y = 'salary_midpoint',
            color = '#5fa8d3', alpha = 1, saturation = 1)

for bar in ax2.patches:
    ax2.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height(),
            f'{bar.get_height()}K',
            va = 'bottom', ha = 'center', size = 8.5)
ax2.axhline(y = data.salary_midpoint.mean(), color = 'red')
plt.xticks(rotation = 30, size = 7)
ax2.set_ylim([0,150])
ax2.set_yticks(range(0, 160, 10))
ax2.set_title('The Avergae Salary Midpoint across Business Sectors')
```

```
ax2.set_ylabel('Average Salary Midpoint')
plt.show()
```



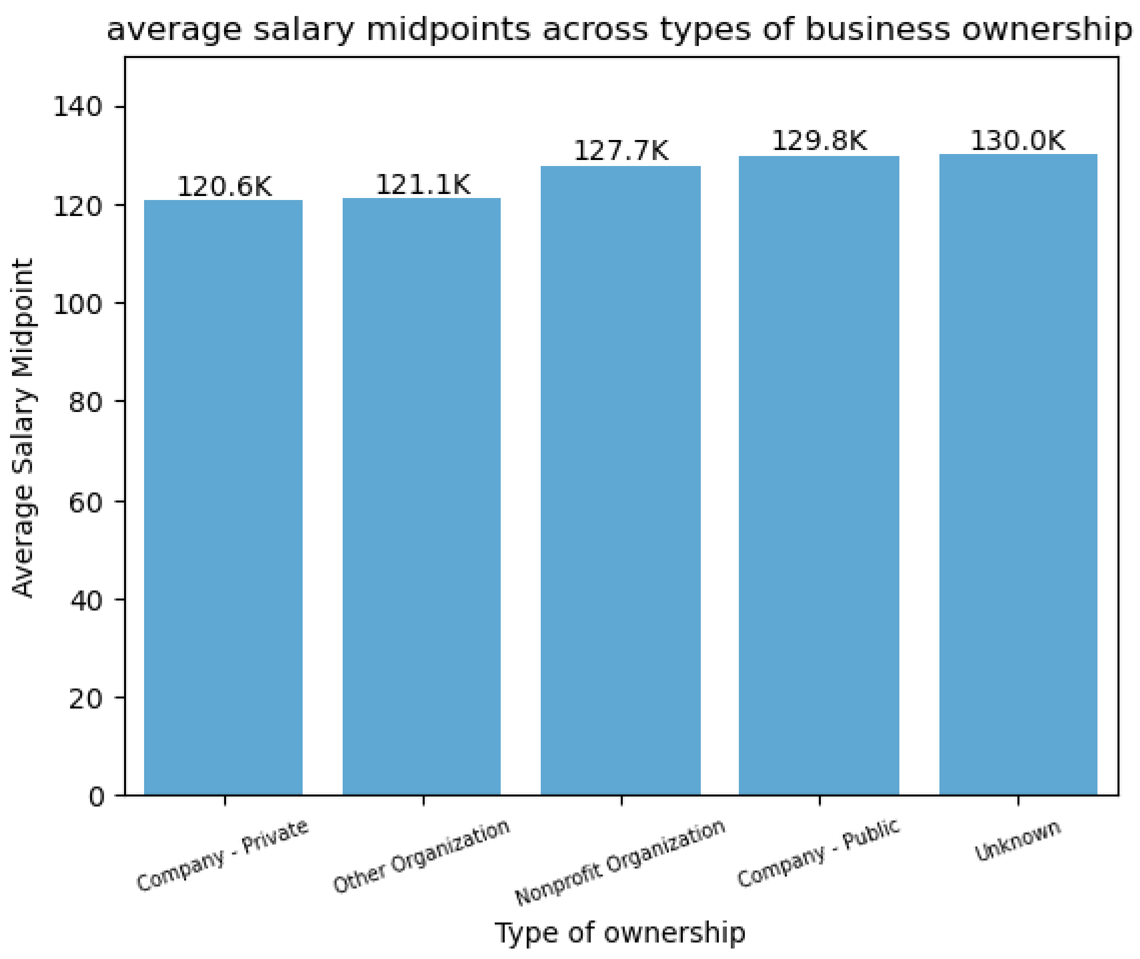
In the second bar plot, the average salary midpoints are displayed exclusively for sectors with more than 10 data entries. Notably, the Government and Aerospace & Defense sectors emerge as offering the highest salary midpoints, while the Finance and Insurance sectors appear to lag, falling on the lower end of the spectrum. The difference between these extremes is $23.2K, which, considering the range of salaries in the field, is not a substantial gap.

8.What are the average salary midpoints in different types of business ownership?

In [42]:
```
salary_onwership = (data.groupby('Type of ownership', as_index = False)
                    [['Type of ownership', 'salary_midpoint']]
                    .agg({'salary_midpoint':'mean'})
                    .round(1).sort_values('salary_midpoint'))

# plot the graph
ax = sns.barplot(data = salary_onwership, x = 'Type of ownership',
                 y = 'salary_midpoint', color = '#5fa8d3',
                 alpha = 1, saturation = 1)

for bar in ax.patches:
    ax.text(bar.get_x() + bar.get_width() / 2,
            bar.get_height(),
            f'{bar.get_height()}K',
            va = 'bottom', ha = 'center')
plt.xticks(rotation = 20, size = 7)
ax.set_title('average salary midpoints across types of business ownership')
ax.set_ylim([0, 150])
ax.set_ylabel('Average Salary Midpoint')
plt.show()
```

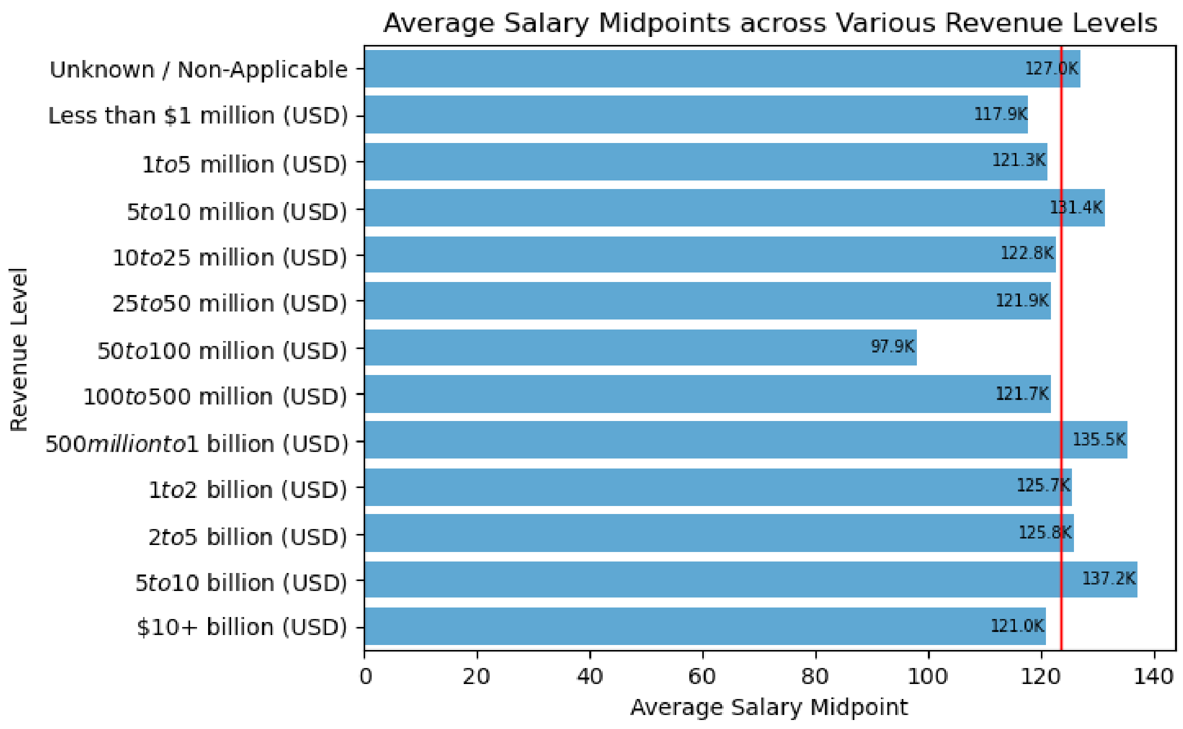


We didn't observe a substantial difference in average salary midpoints across various ownership types.

9.Do companies with greater revenues tend to offer higher average salary midpoints?

In [43]:
```
salary_revenue = (data.groupby('Revenue', as_index = False)
                    [['Revenue', 'salary_midpoint', 'revenue_lower_bond']]
                    .agg({'salary_midpoint':'mean',
                          'revenue_lower_bond':'mean'})
                    .round(1).sort_values('revenue_lower_bond'))

#plot
ax = sns.barplot(data = salary_revenue, y = 'Revenue',
                 x = 'salary_midpoint', color = '#5fa8d3',
                 alpha = 1, saturation = 1)
for bar in ax.patches:
    ax.text(bar.get_x() + bar.get_width(),
            bar.get_y() + bar.get_height() / 2,
            f'{bar.get_width()}K',
            va = 'center', ha = 'right', size = 7)
ax.axvline(x = data['salary_midpoint'].mean(), color = 'red', lw = 1)
ax.set_title('Average Salary Midpoints across Various Revenue Levels')
ax.set_ylabel('Revenue Level')
ax.set_xlabel('Average Salary Midpoint')
plt.show()
```



In the horizontal bar plot above, we compare average salary midpoints across different company revenue levels. While most revenue categories closely align with the overall salary midpoint—represented by the red vertical line—four categories deviate noticeably. The categories of 5 to 10 million, 500 million to 1 billion, and 5 to 10 billion all have average salary midpoints exceeding $130K. Interestingly, the 10+ billion category stands out as the sole group with revenues exceeding 1 billion that registers below the overall average salary midpoint. Conversely, the 5 to 10 million category is the only group with revenues under 1 billion but with a salary midpoint higher than average. This pattern to some extent suggests that the companies with a higher revenue tend to provide a higher salary midpoint.

Another thing I have to mention is that the average salary midpoint for the 50 to 100 million revenue category is only $97.9K, notably lower than other categories. This led me to question the representativeness of the data for this category. Upon closer examination, I found that the 50 to 100 million category comprises 32 data entries. Evaluating its diversity across various dimensions, the data is robust in key areas: it encompasses 6 of the 8 unique titles, represents 23 distinct companies across 10 different sectors, and spans 14 states.

However, despite these diverse data points, there's a level of uncertainty due to the limited sample size, and we should be wary of generalizing this finding without additional data or external validation.

In [44]:
```
# extract the number of unique value in each column of the data of
# the 50-100million group
diversity = (data.loc[data['revenue_lower_bond'] == 50]
.agg([lambda col: data[col.name].unique().shape[0],
      lambda col: col.unique().shape[0]])
.transpose())
diversity.columns = ['total number of unique values in dataset',
                     'number of unique values in 50-100 million group']
diversity
```

Out[44]:

	total number of unique values in dataset	number of unique values in 50-100 million group
Job Title	8	6
Salary Estimate	30	18
Job Description	489	22
Rating	32	15
Company Name	432	23
Location	207	22
Headquarters	229	20
Size	8	5
Founded	103	16
Type of ownership	5	3
Industry	58	13
Sector	23	10
Revenue	13	1
old title	172	11
integrated title	2	2
salary_lower	26	14
salary_upper	26	15
salary_midpoint	26	15
city	201	22
state	40	14
country	3	1
revenue_lower_bond	13	1

Conclusion

1. Based on our dataset, the 'Data Analyst' role has the lowest average salary midpoint, while the 'Data Science Manager' role commands the highest average salary midpoint. The 'Data Engineer,' 'Data Scientist,' and 'Machine Learning Scientist' roles exhibit relatively similar salary midpoints and ranges. As for the senior positions, we were unable to find clear evidence to support our hypothesis that senior titles, on average, command higher salaries.

2. The choropleth maps in our analysis reveal the significant geographical variations in the data science job market across the United States, with coastal states generally offering more opportunities and higher salaries compared to central states. California stands as a significant hotspot, boasting a high number of job postings, although its average salary midpoints are not exceptionally high when compared to certain other states. The East Coast, particularly New York, Virginia, and

Massachusetts, also emerges as a key region with a substantial number of job postings and competitive salary midpoints.

3. The analysis also shows interesting insights at the company level. The Information and Technology sector stands out as a leader in job opportunities, indicating a robust demand for data professionals in technology-centric industries. However, the data suggsted that a high demand for data science roles in certain sectors does not necessarily correlate with higher salaries.

4. While the data doesn't strongly support the hypothesis that larger companies invariably offer higher salaries, it does indicate that companies with a revenue exceeding 1 billion dollars tend to have higher salary midpoints.

Discussion

1. Analysis Implications

The insights from this analysis can serve as a valuable guide for data science job seekers as they craft their career strategies.

- Firstly, while the Information and Technology sector is a hotbed for data science opportunities, it is advisable for professionals to explore alternative sectors as well. Some sectors, such as Government and Aerospace & Defense, may have fewer positions available, but they tend to offer higher compensation. This scenario underscores the importance of cultivating specialized skills that align with the unique demands of these sectors.

- Secondly, location matters; coastal states like California and New York not only offer a high number of job opportunities but also competitive salaries. However, individuals must weigh these prospects against the cost of living in these regions.

- Lastly, the data does not support the common assumption that larger companies invariably offer higher salaries. As such, job seekers should cast a wider net, considering positions in smaller or medium-sized companies, which might offer competitive salaries, potentially lower stress levels, and a more intimate work environment.

2. Limitations:

While these insights offer valuable guidance for data science job seekers, it is important to recognize that this analysis has its limitations. The following are some of the limitations of this study that should be considered when interpreting its findings:

- Outdated Data: The dataset was recorded in 2021, and as the analysis is being conducted in 2023, there might have been significant changes in the data science job market. The two-year gap could have resulted in shifts in salary ranges, job demand in various locations, and industry hiring trends. This time lag may affect the current relevance and applicability of the findings.

- Limited Sample Size: The dataset contains only 672 job postings, which may not be sufficiently large to accurately represent the entire data science job market. For example, there is only one data entry for Delaware, and only 3 job postings for the Education sector. This limited sample size could potentially skew the analysis and may not capture the full scope of the job market.

- Source Bias and Data Integrity: The data was web scraped from Glassdoor. Relying on a single source for data can introduce bias, as Glassdoor's listings may not be fully representative of the broader job market. Meanwhile, because we could not validate the integrity of the scraped data, this raises questions about the accuracy and completeness of the information used in this analysis.

- Salary Estimates: The dataset does not provide fixed salary values, but rather a wide range estimated by Glassdoor. Additionally, the analysis is based on salary midpoints, which are calculated as the middle value between the given upper and lower limits. Since the actual distribution of salaries within this range is unclear, this approach may not accurately reflect the actual salary offers. This could, in turn, affect the reliability of the salary insights generated by the analysis.

- Title Consolidation: In the analysis, we consolidated the job titles into 8 general categories. However, the responsibilities of a data science professional can vary significantly based on their main field. This title aggregation process may introduce potential bias.

Acknowledgement

1. I would like to extend my gratitude to Rashik Rahman for providing the data used in this analysis. The dataset has been instrumental in allowing this comprehensive exploration of the data science job market. Data Source Link: https://www.kaggle.com/datasets/rashikrahmanpritom/data-science-job-posting-on-glassdoor