How accessible is the job market for Junior Data Analysts?

Navigating the tech job market can be challenging, especially for junior professionals. What are the most in demand skills that a Junior Data Analyst should have? What are the top industries that hire juniors? How is salary influenced by industry, skills or company's rating?

With these driving questions in mind, I looked for valuable insights that can benefit individuals who, like me, are seeking employment in this field.

Using Tableau (Public version) and Excel, I analysed a public dataset that provides information on the job market for Data Analysts in the USA. I have cleaned up the data using Python and Jupyter Notebook, ensuring it is ready and consistent for analysis. A data clean up version using Pandas is coming up soon.

The insights that I have discovered through this analysis are shared in the Conclusions section.

The dataset

The dataset was taken from Kaggel (https://www.kaggle.com/datasets/andrewmvd/data-analyst-jobs, by @picklesueat).

It is a CSV file containing 2253 job postings and it shows the USA data analyst job market as of June 2020. There are 16 columns included: #, Job Title, Salary Estimate, Job Description, Rating, Company Name, Location, Headquarters, Size, Founded, Type of Ownership, Industry, Sector, Revenue, Competitors and Easy Apply.

The driving questions

Throughout this analysis, I would like to find answers for the following questions:

- What kind of companies (by size and by age) are looking for data analysts?
- Which industries are looking for juniors?
- What are the most in demand skills? What skills are required for a junior vs more senior roles?
- How does the salary distribution look?
- What are the top 10 industries by salary range?
- Are seniority levels and skills influencing the min and the max salary?
- How is the average salary influenced by company age and rating ?

Reviewing the dataset

Load the data

The dataset can be downloaded as a ZIP from Kaggle. The format is CSV, one row representing one job posting. Once the file has been obtained, it can be loaded for further processing via Jupyter Notebook, using Python.

Sample the data

To see some data samples, I wrote a function to print the first 10 job postings, excluding the ‘Job Description’ column, which is too verbose.

```
def print_no_job_description(data):
    for i in range(0, 10):
        row = data[i]
        for key in row:
            if key != 'Job Description':
                print(str(key) + ':' + str(row[key]))
        print('\n')
```

```
print_no_job_description(data)
```

```
 :0
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
3.2
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:-1
Easy Apply:True


 :1
Job Title:Quality Data Analyst
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.8
Company Name:Visiting Nurse Service of New York
3.8
Location:New York, NY
Headquarters:New York, NY
Size:10000+ employees
Founded:1893
Type of ownership:Nonprofit Organization
Industry:Health Care Services & Hospitals
Sector:Health Care
Revenue:$2 to $5 billion (USD)
Competitors:-1
Easy Apply:-1


 :2
Job Title:Senior Data Analyst, Insights & Analytics Team [Customer Operations]
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.4
Company Name:Squarespace
3.4
Location:New York, NY
Headquarters:New York, NY
Size:1001 to 5000 employees
Founded:2003
Type of ownership:Company - Private
Industry:Internet
Sector:Information Technology
Revenue:Unknown / Non-Applicable
Competitors:GoDaddy
Easy Apply:-1


 :3
Job Title:Data Analyst
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:4.1
Company Name:Celerity
4.1
Location:New York, NY
Headquarters:McLean, VA
Size:201 to 500 employees
Founded:2002
Type of ownership:Subsidiary or Business Segment
Industry:IT Services
Sector:Information Technology
Revenue:$50 to $100 million (USD)
Competitors:-1
Easy Apply:-1


 :4
Job Title:Reporting Data Analyst
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.9
Company Name:FanDuel
3.9
Location:New York, NY
Headquarters:New York, NY
Size:501 to 1000 employees
Founded:2009
Type of ownership:Company - Private
Industry:Sports & Recreation
Sector:Arts, Entertainment & Recreation
Revenue:$100 to $500 million (USD)
Competitors:DraftKings
Easy Apply:True
```

```
 :5
Job Title:Data Analyst
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.9
Company Name:Point72
3.9
Location:New York, NY
Headquarters:Stamford, CT
Size:1001 to 5000 employees
Founded:2014
Type of ownership:Company - Private
Industry:Investment Banking & Asset Management
Sector:Finance
Revenue:Unknown / Non-Applicable
Competitors:-1
Easy Apply:-1


 :6
Job Title:Business/Data Analyst (FP&A)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:4.4
Company Name:Two Sigma
4.4
Location:New York, NY
Headquarters:New York, NY
Size:1001 to 5000 employees
Founded:2001
Type of ownership:Company - Private
Industry:Investment Banking & Asset Management
Sector:Finance
Revenue:Unknown / Non-Applicable
Competitors:-1
Easy Apply:-1


  :7
 Job Title:Data Science Analyst
 Salary Estimate:$37K-$66K (Glassdoor est.)
 Rating:3.7
 Company Name:GNY Insurance Companies
 3.7
 Location:New York, NY
 Headquarters:New York, NY
 Size:201 to 500 employees
 Founded:1914
 Type of ownership:Company - Private
 Industry:Insurance Carriers
 Sector:Insurance
 Revenue:$100 to $500 million (USD)
 Competitors:Travelers, Chubb, Crum & Forster
 Easy Apply:True


  :8
 Job Title:Data Analyst
 Salary Estimate:$37K-$66K (Glassdoor est.)
 Rating:4
 Company Name:DMGT
 4.0
 Location:New York, NY
 Headquarters:London, United Kingdom
 Size:5001 to 10000 employees
 Founded:1896
 Type of ownership:Company - Public
 Industry:Venture Capital & Private Equity
 Sector:Finance
 Revenue:$1 to $2 billion (USD)
 Competitors:Thomson Reuters, Hearst, Pearson
 Easy Apply:-1

 :9
Job Title:Data Analyst, Merchant Health
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:4.4
Company Name:Riskified
4.4
Location:New York, NY
Headquarters:New York, NY
Size:501 to 1000 employees
Founded:2013
Type of ownership:Company - Private
Industry:Research & Development
Sector:Business Services
Revenue:Unknown / Non-Applicable
Competitors:Signifyd, Forter
Easy Apply:-1
```

Review the available data series

Displaying all headers helps see that 16 columns are used to describe this dataset.

[4]:
```
def get_all_headers(data):
    list_of_keys = set()
    for dictionary in data:
        list_of_keys.update(dictionary.keys())
    return list_of_keys
```

[5]:
```
get_all_headers(data)
```

[5]:
```
{' ',
 'Company Name',
 'Competitors',
 'Easy Apply',
 'Founded',
 'Headquarters',
 'Industry',
 'Job Description',
 'Job Title',
 'Location',
 'Rating',
 'Revenue',
 'Salary Estimate',
 'Sector',
 'Size',
 'Type of ownership'}
```

Review the feasibility of the dataset

A fundamental question is whether this dataset can be used to find answers to the driving questions.

Sampling the data and examining the name of the series it can be observed that the dataset comprises a wide array of valuable information. It includes data about job title, salary, rating, company name, size, age, industry, job description, and more, which offer a good foundation for our analysis.

After taking a closer look at the original data, it becomes clear that the data within the intended scope is not yet ready for analysis. It needs to be enriched by extracting important information from the existing data. For example, the original dataset doesn't have a column that directly provides salary information for each job posting. However, this information can be obtained from the 'Salary Estimate' column. The same goes for company size, technical skills, seniority levels, and other relevant details.

The dataset also lacks consistency. Entries in some series use different ways to represent missing information, such as '-1', 'Unknown', 'NA', or leaving the field empty. Job title column offer plenty of examples in which postings for similar roles use different wording: ‘ Jr. Data Analyst’, ‘Junior Data Analyst’, etc.

However, by using proper data cleaning techniques, this dataset can be shaped into a usable and consistent form for analysis. This involves making the content relevant and ensuring that it follows a consistent pattern.

Cleaning and transforming the dataset

Remove duplicates

After using a function that finds and removes any duplicates from the original dataset, it turns out that there aren't any duplicates in the data. Despite this removal, the number of rows remains the same, with a total of 2253 rows, indicating that each row in the dataset seems to be unique.

[2]:
```
def remove_duplicates(data):
    unique = []
    for row in data:
        if row not in unique:
            unique.append(row)
    return unique
```

[3]:
```
print(len(data))
data = remove_duplicates(data)
print(len(data))
```

```
2253
2253
```

Cleaning the headers

When listing all the headers, it becomes clear that one of them is missing a title. Examining a sample of the data, it can be observed that this column represents an index for each job posting. Therefore, I created a function that replaces the empty title with a suitable one: 'Index'.

[6]:
```
def rename_empty_header(data, old_header, replacement_header):
    for row in data:
        row[replacement_header] = row[old_header]
        del row[old_header]
```

[7]:
```
rename_empty_header(data, ' ', 'Index')
```

[8]:
```
get_all_headers(data)
```

Checking the invalid values

It's important to determine the number of entries with invalid data for each series, as this could affect the reliability of certain series for the analysis. To start, I checked the number of invalid values in each series. By invalid data, I mean values that are either null, empty strings, or -1.

```
# invalid values means None, empty string or -1
def count_invalid_values_for_column(data, col):
    count_invalid_data = 0
    for row in data:
        if col not in row:
            continue
        value = row[col]
        if value == None or value == '' or value == '-1':
            count_invalid_data += 1
    return count_invalid_data
```

```
def count_invalid_data(data):
    all_headers = get_all_headers(data)
    invalid_values_per_column = {}
    for header in all_headers:
        invalid_values_per_column[header] = count_invalid_values_for_column(data, header)
    return invalid_values_per_column
```

```
count_invalid_data(data)
```

```
{'Location': 0,
 'Easy Apply': 2173,
 'Company Name': 1,
 'Type of ownership': 163,
 'Competitors': 1732,
 'Industry': 353,
 'Job Description': 0,
 'Size': 163,
 'Salary Estimate': 1,
 'Job Title': 0,
 'Sector': 353,
 'Index': 0,
 'Rating': 272,
 'Revenue': 163,
 'Founded': 660,
 'Headquarters': 172}
```

When looking at the unique values in a specific series, I noticed that these invalid values were used inconsistently. This inconsistency makes it difficult to determine whether this data is usable or not.

```
def find_unique_values_for_column(data, col):
    list_of_values = set()
    for row in data:
        if col not in row:
            continue
        list_of_values.add(row[col])
    return list_of_values
```

```
find_unique_values_for_column(data, 'Easy Apply')
```

```
{'-1', 'True'}
```

```
def replace_specific_value_with_specified(data, col, old_value, new_value):
    cleaned_data = data.copy()
    for dictionary in cleaned_data:
        if col not in row:
            continue
        if dictionary[col] == old_value:
            dictionary[col] = new_value
    return cleaned_data
```

```
data = replace_specific_value_with_specified(data, 'Easy Apply', '-1', 'NA')
```

```
data = replace_specific_value_with_specified(data, 'Revenue', 'Unknown / Non-Applicable', 'NA')
```

```
data = replace_specific_value_with_specified(data, 'Size', 'Unknown', 'NA')
```

For example, the 'Easy Apply' series has '-1' and 'True' as distinct values. However, other columns in the dataset use different ways to indicate missing or invalid data, by using 'Unknown' or 'Non-Applicable'. To make things consistent, I used a function called 'replace_specific_values_with_specified' to replace those inconsistent values with 'NA' throughout the dataset (seen above).

As a final step, when examining the dataset to determine the number of 'NA' values in each series, here is what I found:

```
def count_NA_values_for_specified_column(data, col):
    count_invalid_data = 0
    for row in data:
        if col not in row:
            continue
        if row[col] == 'NA':
            count_invalid_data += 1
    return count_invalid_data
```

```
def count_NA_values(data):
    all_headers = get_all_headers(data)
    NA_values_per_column = {}
    for header in all_headers:
        NA_values_per_column[header] = count_NA_values_for_specified_column(data, header)
    return NA_values_per_column
```

```
count_NA_values(data)
```

```
{'Easy Apply': 2173,
 'Industry': 353,
 'Revenue': 778,
 'Company Name': 1,
 'Location': 0,
 'Size': 205,
 'Sector': 353,
 'Competitors': 1732,
 'Index': 0,
 'Type of ownership': 179,
 'Job Title': 0,
 'Founded': 660,
 'Salary Estimate': 1,
 'Rating': 272,
 'Job Description': 0,
 'Headquarters': 172}
```

It can be observed that series such as ‘Easy Apply’ and ‘Competitors’ have a high number of invalid/missing data, which might render them unusable for any analysis

Cleaning the ‘Company Name’ column

When looking at the data in the 'Company Name' column, it can be noticed that it is a combination of both the company name and the rating. This can create difficulties when working with the data. The dataset already has a separate column called 'Rating', containing values matching those in the 'Company Name' column. To address this issue, I decided to remove the rating information from the 'Company Name' column. I achieved this by running the function listed below.

```
# remove rating from clumn 'Company Name'
def clean_column_company_name(data):
    cleaned_data = data.copy()
    key = 'Company Name'
    for row in cleaned_data:
        if key not in row:
            continue
        parts = str(row[key]).split('\n')
        row[key] = parts[0].strip()
    return cleaned_data
```

```
data = clean_column_company_name(data)
```

As seen below, now the ‘Company Name’ column contains only the needed information.

```
print_no_job_description(data)
```

```
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:NA
Easy Apply:True
Index:0
Max size:500
Min salary:37
Max salary:66
Average salary:51.5


Job Title:Quality Data Analyst
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.8
Company Name:Visiting Nurse Service of New York
Location:New York, NY
Headquarters:New York, NY
Size:10000+ employees
Founded:1893
Type of ownership:Nonprofit Organization
Industry:Health Care Services & Hospitals
Sector:Health Care
Revenue:$2 to $5 billion (USD)
Competitors:NA
Easy Apply:NA
Index:1
Max size:10000
Min salary:37
Max salary:66
Average salary:51.5


Job Title:Senior Data Analyst, Insights & Analytics Team [Customer Operations]
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.4
Company Name:Squarespace
Location:New York, NY
```

Transforming the ‘Size’ column

This column contains a mix of numerical and string values, describing the number of employees as a range. My goal was to extract and keep only the maximum number of employees for each entry. To achieve this, I created a new column called 'Max size' to store these values.

```
find_unique_values_for_column(data, 'Size')
```

```
{'1 to 50 employees',
 '10000+ employees',
 '1001 to 5000 employees',
 '201 to 500 employees',
 '5001 to 10000 employees',
 '501 to 1000 employees',
 '51 to 200 employees',
 'NA'}
```

```
import re
def extract_max_no_of_employees(data):
    cleaned_data = data.copy()
    key = 'Size'
    new_key = 'Max size'
    for row in cleaned_data:
        if key not in row:
            continue
        if '10000+' in row[key]:
            row[new_key] = 10000
        else:
            value = re.search(r'(\d+)(\sto\s)*(\d*)(.*)', row[key], re.IGNORECASE)
            if value:
                row[new_key] = int(value.group(3))
            else:
                row[new_key] = 'NA'
    return cleaned_data
```

```
data = extract_max_no_of_employees(data)
```

After this transformation the data looks like:

```
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:NA
Easy Apply:True
Index:0
Max size:500
```

Transforming the ‘Salary’ column into min, max and average salary

Similar to the ‘Size’ column, the ‘Salary Estimate’ column has numbers mixed with strings, in a specific pattern, describing the salary range. To make this column useful I extracted min and max salary as new columns and converted them to numbers. Then I calculated the average salary and stored it as a separate column as well.

```
import re
def extract_min_and_max_salary_as_new_columns(data):
    key = 'Salary Estimate'
    for row in data:
        if key not in row:
            continue
        value = re.search(r"\$(\d+)(.*)\$(\d+)(.*)", row[key], re.IGNORECASE)
        if value:
            row['Min salary'] = int(value.group(1))
            row['Max salary'] = int(value.group(3))
    return data
```

```
data = extract_min_and_max_salary_as_new_columns(data)
```

```
def calculate_average_salary_as_new_column(data):
    cleaned_data = data.copy()
    for row in cleaned_data:
        if 'Min salary' not in row or 'Max salary' not in row:
            continue
        row['Average salary'] = (row['Min salary'] + row['Max salary'])/2
    return cleaned_data
```

```
data = calculate_average_salary_as_new_column(data)
```

The effects can be seen in the below picture, as 3 more columns have been added.

```
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:NA
Easy Apply:True
Index:0
Max size:500
Min salary:37
Max salary:66
Average salary:51.5
```

Rechecking for missing information

I reviewed the data once more to identify any missing information, and I discovered three columns:

max salary, min salary, and average salary. To address this, I replaced the missing values with "NA" to ensure consistency throughout the dataset. By doing so, we can maintain the integrity of the data and handle any gaps effectively.

```
def count_missing_keys(data, col):
    count_missing_keys = 0
    for row in data:
        if col not in row:
            count_missing_keys += 1
    return count_missing_keys
```

```
def count_missing_keys_info(data):
    all_headers = get_all_headers(data)
    missing_info = {}
    for header in all_headers:
        missing_info[header] = count_missing_keys(data, header)
    return missing_info
```

```
count_missing_keys_info(data)
```

```
{'Location': 0,
 'PowerBI required': 0,
 'Max salary': 1,
 'Company Name': 0,
 'Senior level': 0,
 'Salary Estimate': 0,
 'Type of ownership': 0,
 'Job Title': 0,
 'Lead level': 0,
 'Revenue': 0,
 'Min salary': 1,
 'Founded': 0,
 'Headquarters': 0,
 'Max size': 0,
 'Competitors': 0,
 'Master': 0,
 'Excel required': 0,
 'Tableau required': 0,
 'Mid level': 0,
 'Easy Apply': 0,
 'Python required': 0,
 'Industry': 0,
 'Job Description': 0,
 'Sector': 0,
 'Average salary': 1,
 'Index': 0,
 'Rating': 0,
 'R required': 0,
 'SQL required': 0,
 'Bachelor': 0,
 'Doctorate': 0,
 'Size': 0,
```

```
def replace_missing_keys_with_NA(data, col):
    for row in data:
        if col not in row:
            row[col] = "NA"
    return data
```

```
data = replace_missing_keys_with_NA(data, 'Min salary')
```

```
data = replace_missing_keys_with_NA(data, 'Max salary')
```

```
data = replace_missing_keys_with_NA(data, 'Average salary')
```

Cleaning and transforming the ‘Job title’ column

Many job titles use different wording to describe similar positions. Taking as example the position of Junior Data Analyst, I identified multiple patterns like: abbreviations (Jr, Jr., etc), different languages (Data Analyst â Junior), different wording (Entry Level, Intern, etc). In order to make job titles more consistent, I created the function below and replaced the above patterns with “Junior Data Analyst”. Similar changes have been done for Senior Data Analyst positions.

```
def replace_job_title_with_consistent_naming(data, col, value_to_replace, new_value):
    cleaned_data = data.copy()
    for dictionary in cleaned_data:
        for key, value in list(dictionary.items()):
            if key == col:
                value = str(value).replace(value_to_replace, new_value)
                dictionary[key] = value
```

Next I mapped the data to 4 levels of seniority: junior, mid, senior and lead, and extracted each level of seniority as a new column.

```
def extract_seniority_as_column(data, col):
    seniority_keywords = {'Junior level':'[\W*]Junior|Jr|Entry[\s-]level|Intern[\W*]', 'Senior level':'[\W*]Senior|Sr[\W*]','Lead level':'[\W*]Lead|Principal[\W*]'}
    for new_header, pattern in seniority_keywords.items():
        data = extract_keyword_as_column(data, col, pattern, new_header)
    return data
```

```
data = extract_seniority_as_column(data, 'Job Title')
```

```
def extract_mid_seniority(data):
    for row in data:
        if row['Junior level'] == False and row['Senior level'] == False and row['Lead level'] == False:
            row['Mid level'] = True
        else:
            row['Mid level'] = False
    return data
```

```
data = extract_mid_seniority(data)
```

This is how data is looking by now:

```
print_no_job_description(data)
```

```
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:NA
Easy Apply:True
Index:0
Max size:500
Min salary:37
Max salary:66
Average salary:51.5
Junior level:False
Senior level:False
Lead level:False
Mid level:True
```

Transforming the ‘Job Description’ column

The ‘Job Description’ column is too verbose and too unstructured to be usable for analysis. However, it does contain a lot of useful information, such as the skills required by the employee, education requirements and others.

For this specific analysis, I focused on identifying whether a job posting required certain technical skills. These were the selected skills in scope: SQL, Excel, Python, R, Tableau and PowerBI. I extracted this information into separate flag columns.

```
def extract_skills_as_columns(data, col):
    skill_keywords = {'Excel required':'[\W+]Excel[\W+]', 'SQL required':'[\W+]SQL[\W+]','Python required':'[\W+]Python[\W+]','R required':'[\W+]R[\W+]','Tableau required':'[\W+]Tableau[\W+]',
                      'PowerBI required':'[\W+]Power[\s*]BI[\W+]'}
    for new_header, pattern in skill_keywords.items():
        data = extract_keyword_as_column(data, col, pattern, new_header)
    return data
```

```
data = extract_skills_as_columns(data, 'Job Description')
```

As seen below, now the data has been enriched with 6 new columns containing information about which skills are required for each advertised position..

```
print_no_job_description(data)
```

```
Job Title:Data Analyst, Center on Immigration and Justice (CIJ)
Salary Estimate:$37K-$66K (Glassdoor est.)
Rating:3.2
Company Name:Vera Institute of Justice
Location:New York, NY
Headquarters:New York, NY
Size:201 to 500 employees
Founded:1961
Type of ownership:Nonprofit Organization
Industry:Social Assistance
Sector:Non-Profit
Revenue:$100 to $500 million (USD)
Competitors:NA
Easy Apply:True
Index:0
Max size:500
Min salary:37
Max salary:66
Average salary:51.5
Junior level:False
Senior level:False
Lead level:False
Mid level:True
Excel required:False
SQL required:True
Python required:True
R required:True
Tableau required:False
PowerBI required:False
```

Checking the relevance of each column

After all the data cleaning and transformation steps are completed, it is important to check again if this dataset has relevant and valuable data series for this analysis. By counting the number of ‘NA’ values for each column, I can evaluate if the in-scope series are relevant.

```
def count_NA_values_for_specified_column(data, col):
    count_invalid_data = 0
    for row in data:
        if col not in row:
            continue
        if row[col] == 'NA':
            count_invalid_data += 1
    return count_invalid_data
```

```
def count_NA_values(data):
    all_headers = get_all_headers(data)
    NA_values_per_column = {}
    for header in all_headers:
        NA_values_per_column[header] = count_NA_values_for_specified_column(data, header)
    return NA_values_per_column
```

```
count_NA_values(data)
```

```
{'Excel required': 0,
 'R required': 0,
 'Industry': 353,
 'Revenue': 778,
 'Size': 205,
 'Sector': 353,
 'Max salary': 1,
 'Competitors': 1732,
 'PowerBI required': 0,
 'Headquarters': 172,
 'Max size': 205,
 'Mid level': 0,
 'Founded': 660,
 'Python required': 0,
 'Rating': 272,
 'Lead level': 0,
 'Average salary': 1,
 'Company Name': 1,
 'Senior level': 0,
 'Index': 0,
 'Salary Estimate': 1,
 'Job Description': 0,
 'Easy Apply': 2173,
 'Min salary': 1,
 'Tableau required': 0,
 'Location': 0,
 'Junior level': 0,
 'Type of ownership': 179,
 'SQL required': 0,
 'Job Title': 0}
```

The higher the number, the less usable the column. It seems that almost all of the in-scope series have a very small number of missing values. The in-scope columns with a higher number of missing data are: ‘Max size’ (205 out of 2253), ‘Rating’ (272), ‘Industry’ (353), ‘Founded’ (660). However, this is not critical for this analysis.

Exporting the clean dataset

Once the data has been cleaned and transformed into a useful and reliable dataset, I have exported the dataset as a CSV file, keeping only the columns of potential interest.

```
def create_file(data):
    result = []
    for row in data:

            new_info = {}
            new_info['Job title'] = row['Job Title']
            new_info['Company name'] = row['Company Name']
            new_info['Rating'] = row['Rating']
            new_info['Location'] = row['Location']
            new_info['Max size'] = row['Max size']
            new_info['Founded'] = row['Founded']
            new_info['Type of ownership'] = row['Type of ownership']
            new_info['Industry'] = row['Industry']
            new_info['Minimum estimated salary'] = row['Min salary']
            new_info['Maximum estimated salary'] = row['Max salary']
            new_info['Average estimated salary'] = row['Average salary']
            new_info['Excel required'] = row['Excel required']
            new_info['SQL required'] = row['SQL required']
            new_info['R required'] = row['R required']
            new_info['Python required'] = row['Python required']
            new_info['Tableau required'] = row['Tableau required']
            new_info['PowerBI required'] = row['PowerBI required']
            new_info['Junior level'] = row['Junior level']
            new_info['Mid level'] = row['Mid level']
            new_info['Senior level'] = row['Senior level']
            new_info['Lead level'] = row['Lead level']
            new_info['Bachelor'] = row['Bachelor']
            new_info['Master'] = row['Master']
            new_info['Doctorate'] = row['Doctorate']
            result.append(new_info)
    return result
```

```
new_data = create_file(data)
```

```
# write output to csv file
def write_as_csv(new_data):

    file = open('data_analyst_modified.csv', 'w', newline='', encoding="utf8")
    dict_writer = csv.DictWriter(file, fieldnames=['Job title','Company name', 'Max size','Rating', 'Location', 'Founded','Type of ownership','Industry', 'Minimum estimated salary',
                                                   'Maximum estimated salary','Average estimated salary', 'Excel required', 'SQL required', 'R required', 'Python required','Tableau required',
                                                   'PowerBI required', 'Junior level', 'Mid level','Senior level','Lead level','Bachelor', 'Master','Doctorate'])

    dict_writer.writeheader()
    for row in new_data:
        dict_writer.writerow(row)
    file.close()
```

```
write_as_csv(new_data)
```

Understanding the quality of the data using Excel

I utilised basic statistical methods in Excel to gain insights into the distribution of data in specific columns containing numerical values.

	Rating	Minimum estimated salary	Maximum estimated salary	Average estimated salary
Median	3.7	50	87	69
Average	3.73	54.27	89.98	72.12
Mode		37	76	59.5
Min	1	24	38	33.5
Q1	3.3	41	70	58
Q3	4.1	64	104	80.5
Max	5	113	190	150
Count	1981	2252	2252	2252

By examining these columns, I was able to observe and understand how the data was distributed, which helped me uncover patterns and trends within the dataset.

- **Rating**: Mean is very close to median (3.73 and 3.70). There is a slightly right skewed distribution.
- **Minimum Salary**: Average minimum salary is around 54K and median value for minimum salary is around 50K. There is a right skewed distribution. In this case, since the average is higher than the median, it suggests that there are some outliers or higher values that are pulling the average upwards. Therefore, it can be expected to have outliers on the right side of the distribution.
- **Maximum Salary**: Mean is very close to median (89.97 and 87), which means that it has a right skewed distribution. As in the case of min salary, outliers on the right side of the distribution are expected.
- **Average Salary**: Average salary is around 72K with a mean of 69, so outliers can be expected for this variable (min= 33.5K, max= 150K)

Data insights using Excel

Pivot tables in Excel are a great way to summarise large amounts of data and quickly uncover insights. I used them to compare the average estimated salary across different in-scope tech skills, gaining some valuable information about salary variations based on these skills.

Job postings that require Excel or SQL skills offer similar average salary.

Excel	SQL	Average - Average estimated salary
False	False	72.7
	True	72.8
True	False	72.8
	True	70.5

Jobs requiring Python pay better than R, with a noticeable difference of 8K.

Python	R	Average - Average estimated salary
False	False	71.4
	True	65.1
True	False	73.9
	True	75.4

For data visualisation tools, the jobs requiring Tableau pay about 10K more, compared to those asking for PowerBI.

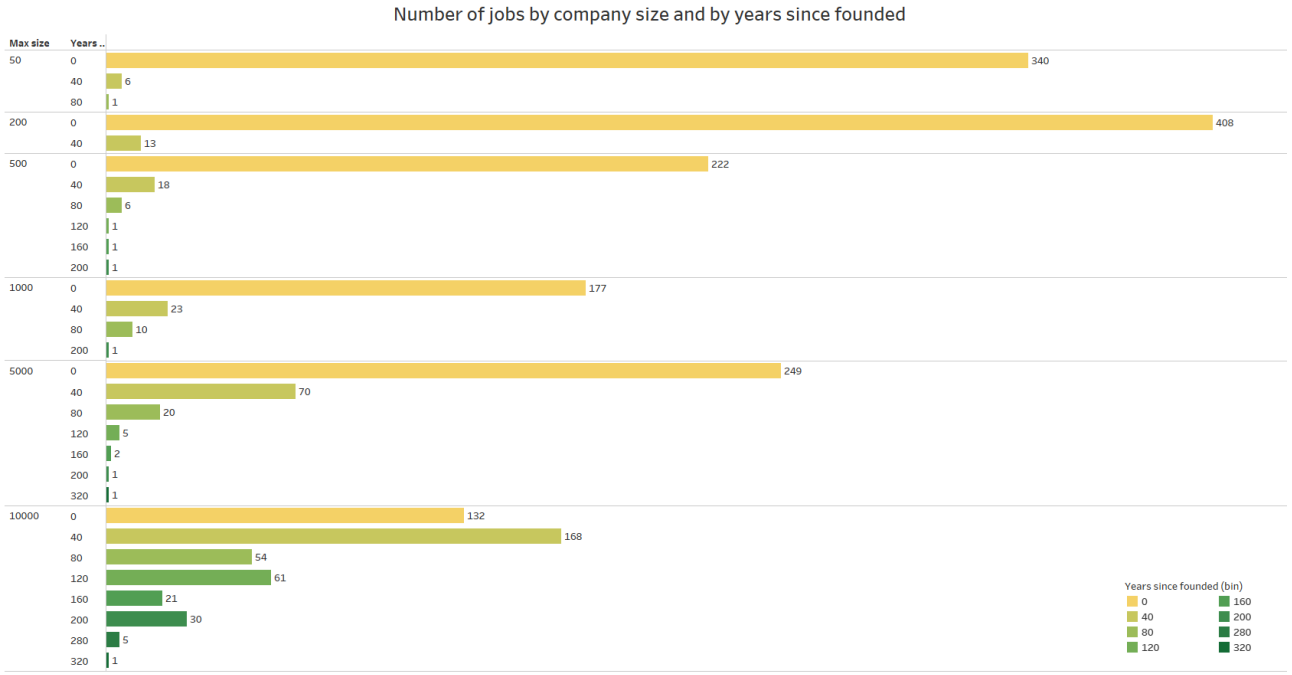
Tableau	PowerBI	Average - Average estimated salary
False	False	71.6
	True	63.1
True	False	74.5
	True	73.9

Data insights using Tableau

Tableau is a great tool for visualising data because it is easy to use and offers powerful features for creating visualisations. I chose Tableau because it is user-friendly, has advanced functions and provides many options for creating engaging visuals.

What kind of companies (by size and by age) are looking for data analysts?

The graph below shows the number of jobs by company size and by years since founded.

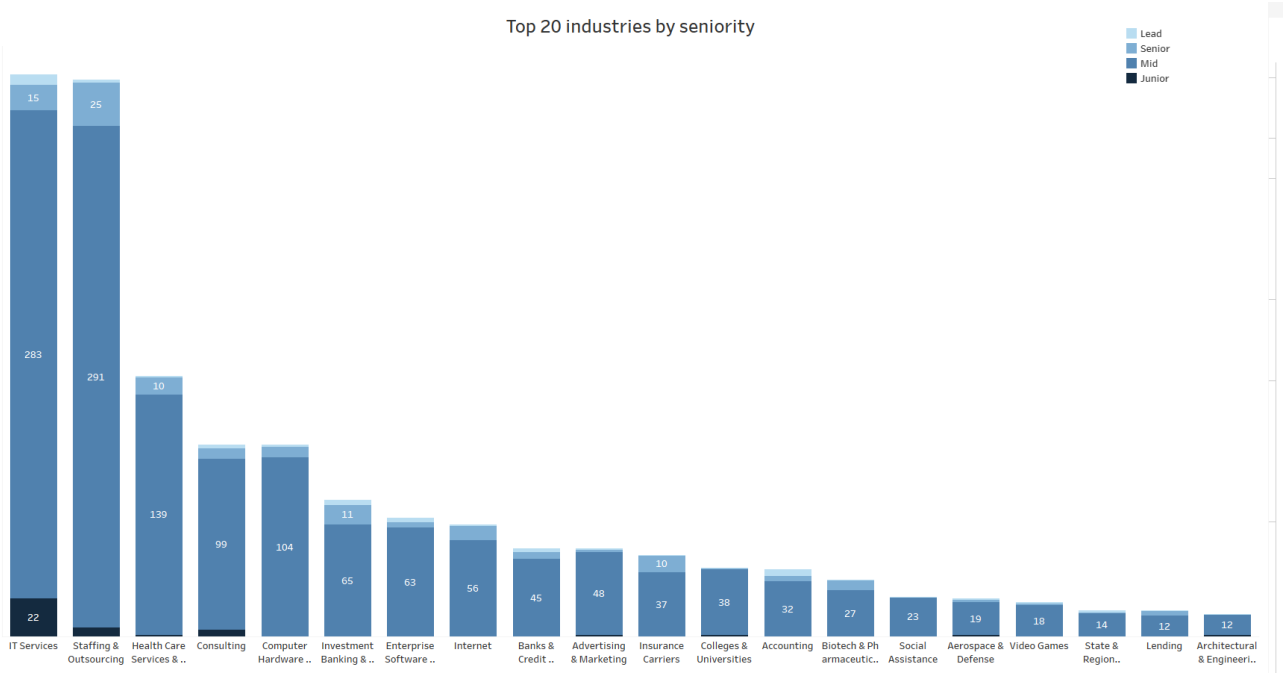


Young and small companies are leading the job market for data analysis roles. More than half of the job offerings are from companies with a maximum of 200 employees and less than 40 years since their establishment. Even for bigger companies, it's still the young ones that offer the bulk of job postings.

The trend of increased demand for data analysts in smaller and newer companies may be due to the fact that these companies are in the early stages of establishing their data analytics departments. Could be that old companies already have their data analytics departments in top shape and also, their retention rate could be higher.

Which industries are looking for juniors?

Top 5 industries hiring data analysts are IT Services, Staffing & Outsourcing, Health Care Services & Hospitals, Consulting and Computer Hardware & Software. Most sought after seniority seems to be Mid level, across all industries.

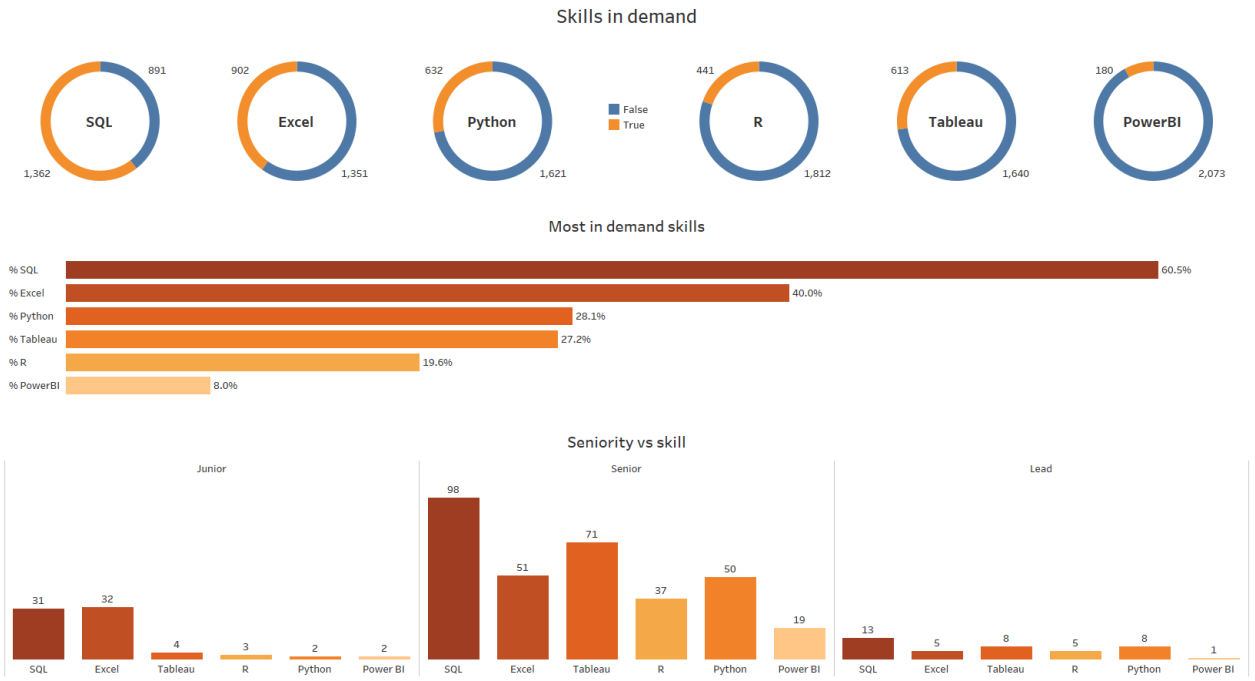


Looking closer at the Junior positions, the number of openings seems to be very low. This dataset shows 22 openings in IT Services, 5 openings in Staffing & Outsourcing, followed by 4 openings in Consulting. Only 1 job opening in each of the following industries: Health Care Services & Hospitals, Computer Hardware & Software, Advertising & Marketing, Colleges & Universities, Aerospace & Defence and Architectural & Engineering Services.

Based on this information, someone looking for a junior position, like me, is more likely to find one in the IT Services, Staffing & Outsourcing or Consulting industries, at a small, young company.

What are the most in demand skills? What skills are required for a junior vs more senior roles?

For the scope of this project I have selected 6 tech skills to focus on. I would like to understand which are required and which are tagged as nice to have. To gain insights into the demand for these skills and how it varies based on seniority level, I created the below dashboard.



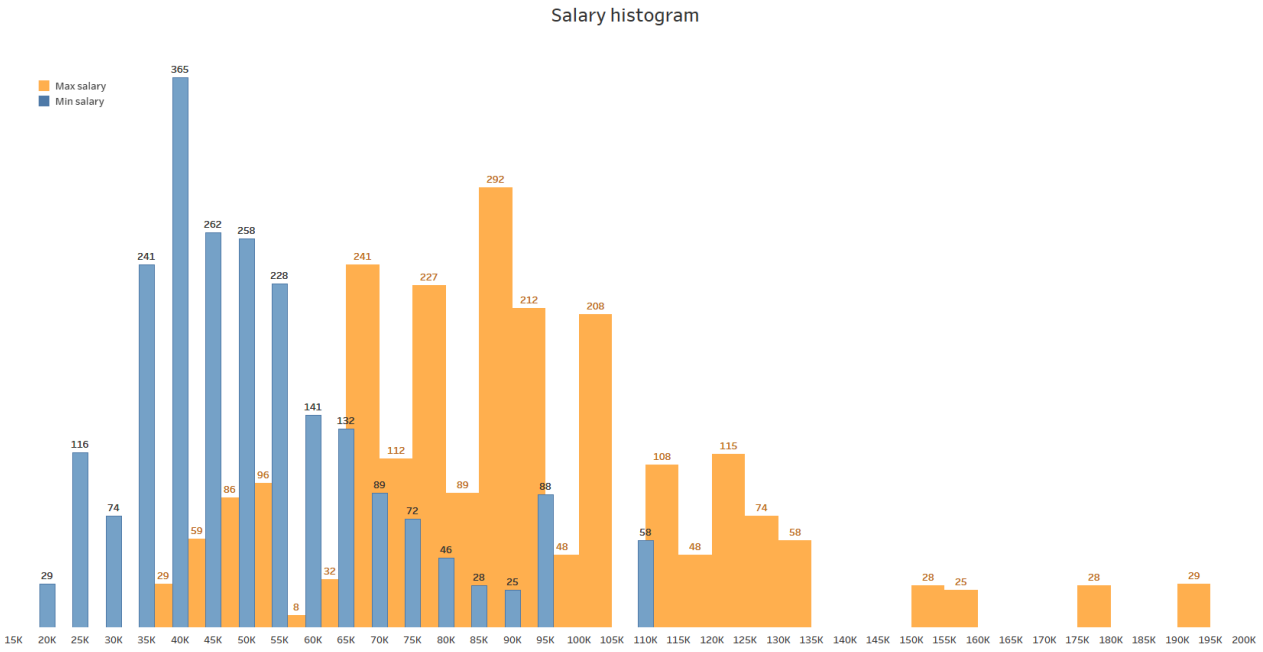
Top of the dashboard shows how frequent each of the in-scope tech skills is mentioned as required in the job openings. SQL seems to be required in 1362 job openings, out of 2253. Between the visualisation tools, Tableau seems to be required in 613 jobs, while Power BI only in 180 jobs.

The most required skills by employers are SQL and Excel, as they can be found in 60% and respectively 40% of the job descriptions, followed by Python and Tableau. R and PowerBI are the least demanded skills.

Seniority levels play a significant role in determining the specific tech skills that are required or considered optional. For junior positions, SQL and Excel are commonly required, while Tableau and R/Python are more like nice-to-have skills. In senior roles, SQL remains essential, followed by Tableau and Python. As a senior, there is an expectation to have a broader skill set. Similarly, for lead positions, SQL remains crucial, but proficiency in multiple skills is expected.

How does the salary distribution look?

The graph below shows the range for min salary (in blue colour) and max salary (in orange colour), including any overlap between them.



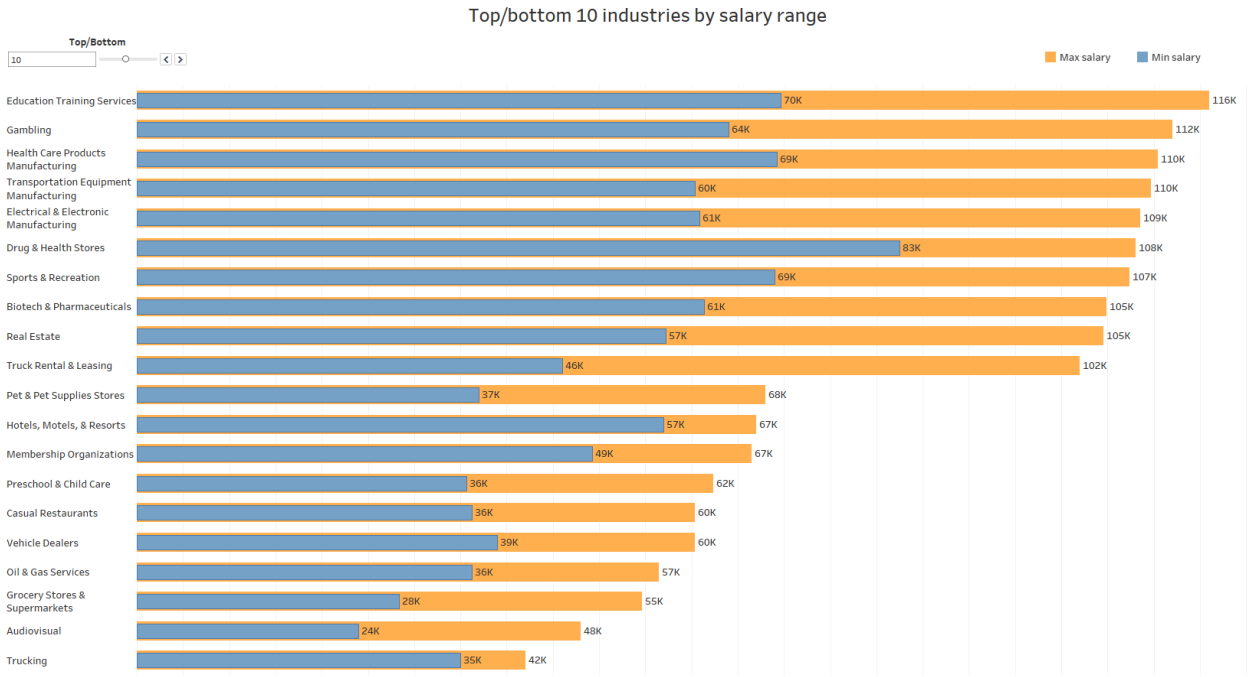
Min salary distribution is right skewed, with mode at 37k, with min value at around 20k and max value at around 110K (outlier).

Max salary distribution is slightly right skewed, with mode at 76K, min value at around 38k and max value at around 190k, multiple outliers on the higher side.

This graph would suggest that the salary ranges for job postings vary, with a concentration around the mode for both minimum and maximum salaries. However, there are outliers on both ends, indicating the presence of lower-paying and higher-paying positions in the dataset.

What are the top 10 industries by salary range?

Choosing 10 on the parameter slider, the graph shows top and bottom 10 industries by salary.



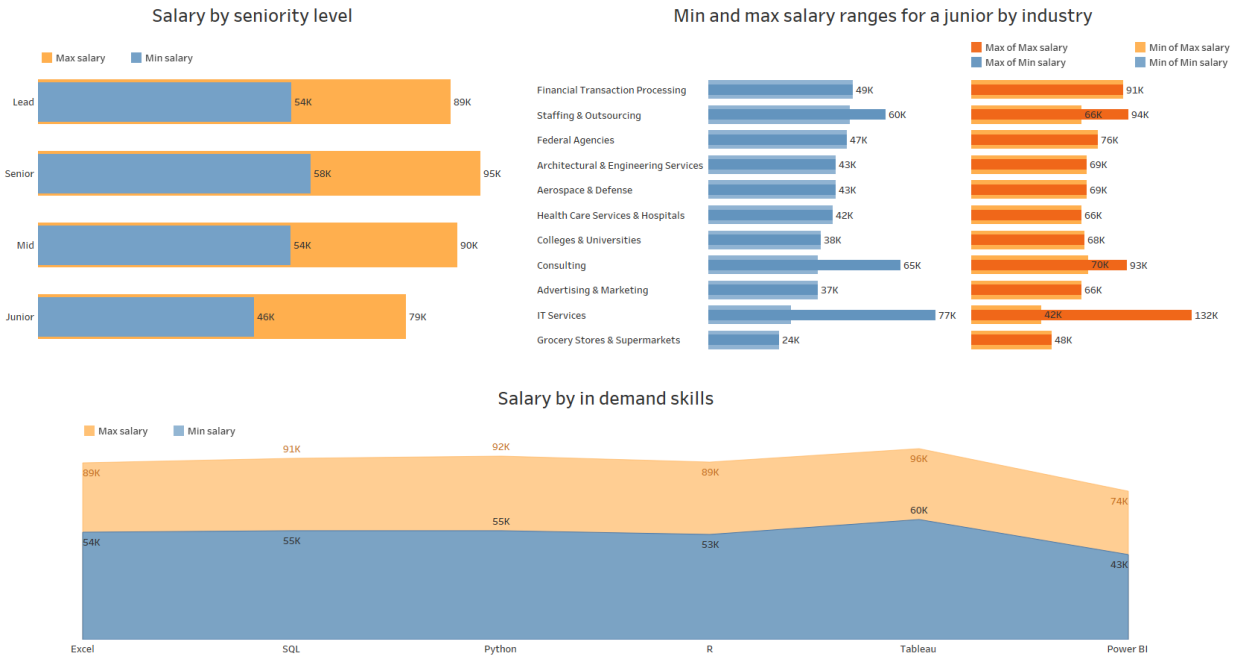
The top 3 industries by max salary are Education Training Services, Gambling and Health Care Products Manufacturing and the bottom 3 are Grocery Stores & Supermarkets, Audiovisual and Trucking. By min salary, the top 3 industries are Drug & Health Stores, Education Training Services and Health Care Products Manufacturing, while the bottom 3 are Grocery Stores & Supermarkets, Audiovisual and Trucking.

Drug & Health Stores, Hotels, Motels & Resorts and Trucking have the smallest gap between min and max salaries.

None of the top 5 hiring industries by number of job postings are in this top/bottom 10 industries by salary range. Extending the search to top/bottom 20, Computer Hardware & Software can be found on the 14th place, with a min salary of 58K and a max salary of 98K. This, combined with the fact that there are 88 unique industries means that the top 4 industries that are hiring are somewhere in the middle of the salary range.

Are seniority levels and skills influencing the min and the max salary?

The dashboard below shows how the salary range is influenced by seniority, industry (in the case of junior positions) and most in-demand skills.



As expected, there is a gradual increase in salary from the Junior level to the Senior level. For Junior positions, the average minimum salary is around 46K, while the average maximum salary is approximately 79K. In contrast, for Senior positions, the salary range is between 58K and 95K, representing an increase of around 12K to 16K compared to the Junior level.

It is surprising to observe that the salary range for Lead positions falls between the Junior and Senior roles, similar to that of Mid-level positions. Given that Lead positions typically require more skills and higher responsibilities, it is counterintuitive for the salary to not reflect this. This inconsistency may be due to the dataset containing a small number of jobs for Lead positions, which may not be representative enough to draw accurate conclusions.

On average, the maximum salary across all levels is around 85K, with Junior-level positions earning below this average. Conversely, all other seniority levels earn above this average. The same trend is observed for the average minimum salary, which is around 53K.

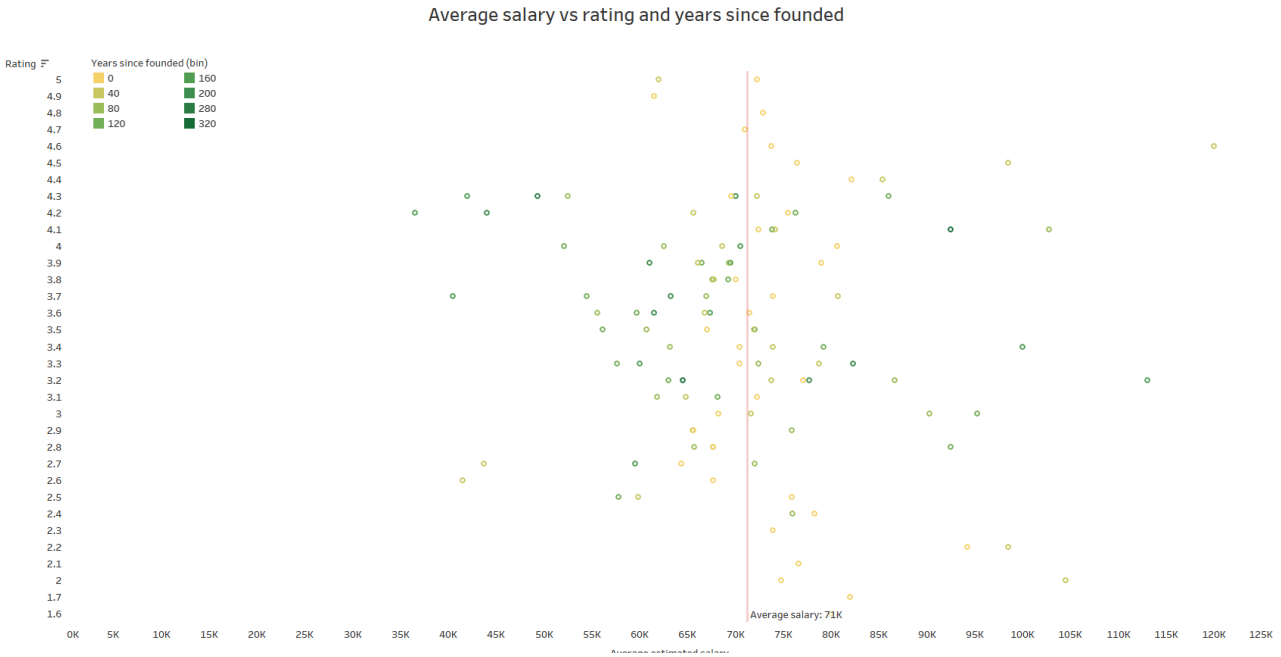
Taking a closer look at junior positions, I wanted to examine the range of minimum and maximum salaries, as averages tend to smooth out the data. Junior employees can expect a minimum salary of 24K, with the maximum capped at 132K. This difference is quite big.

In most industries, the range of minimum salaries and maximum salaries for juniors tends to overlap. However, there are a few exceptions in industries such as Staffing & Outsourcing, Consulting, and IT Services, where the ranges do not overlap. The same observation can be made for the ranges of maximum salaries. It is worth noting that IT Services stands out as the industry offering the highest salaries, both in terms of minimum and maximum salaries for junior positions.

The choice of tech skills has a noticeable impact on salary ranges. Job openings that require Tableau tend to offer the highest salary range, ranging from 60K to 96K. On the other hand, positions that require PowerBI offer the lowest salary range, with a range of 43K to 74K. When comparing the influence of Excel and SQL, it appears that jobs requesting SQL tend to have slightly higher salaries. Similarly, between Python and R, Python takes a slight lead in terms of salary. Based on these findings, it can be concluded that having knowledge of SQL, Excel, Python, and Tableau is more likely to lead to job opportunities with potential higher salary.

How is the average salary influenced by company age and rating?

To observe the influences of both company rating and age on the average salary, I have made the scatter graph below.



Older companies tend to cluster around ratings of 3 up to 4.5 (with 3.7 as an average) and an average salary below 71K.

Younger companies have ratings values quite spread out, ranging from 1.7 to 5 and their average salary seems to group more tightly around the 71K average.

The lowest paying companies (37K) are the ones in the 160 years bin, with a rating of 4.2. The highest paying companies (120K) are in the 40 years bin, with a rating of 4.6.

Conclusions

Navigating the data analyst job market as a junior can be a daunting task due to the multitude of factors involved, from determining the right industry and company size to understanding the specific skills in demand. The competition for junior positions is often fierce, with many candidates vying for limited opportunities.

Even though the dataset focuses on data analyst positions advertised specifically in the USA job market, it still provides valuable insights. While the findings may not directly apply to other regions, they can serve as a reference point or basis for comparison.

In the end, with so many factors to consider, finding the right fit becomes a balancing act, where one must weigh personal preferences, career aspirations and available opportunities.

This dataset shows that a Junior Data Analyst is more likely to find an open position within a small and young company that operates in IT Services, Staffing & Outsourcing, Health Care Services & Hospitals, Consulting or Computer Hardware & Software.

The minimum required tech skills are SQL and Excel. Any knowledge of Tableau and R/Python is beneficial and could increase the chances of landing a job.

While the top industries by salary range would be Education Training Services, Gambling, Drug & Health Stores and Health Care Products Manufacturing, unfortunately none of them have open junior positions. The lowest paying industries are Grocery Stores & Supermarkets, Preschool & Child Care, Audiovisual and Trucking, with barely any advertised jobs.

In terms of pay, a junior can expect to be paid on average between 46K and 79K, which is slightly below the average range for the job market: 53K-85K. The min salary ranges between 24K (Grocery Stores & Supermarkets) and 77K (IT Services) and the max salary ranges between 42K and 132K (both in IT Services).

When rating is taken into consideration, most older companies would be a good choice, as their rating seems to be clustered around the 3.7 average, with the salary on the lower side of the average. Some young companies have excellent ratings (5 or near) and they also tend to offer salaries above the average. If rating is weighing more than salary, then favouring older companies over younger ones could lead to great work experiences, in well proven environments.