IBM Data Analyst Professional Certificate Capstone Project

Data Analysis on Survey for Future Skill Requirements in Technology

YU, ZHU

alexyuzhu@gmail.com

2021-10-10

IBM Developer

SKILLS NETWORK

OUTLINE

- Part I. Executive Summary
- Part II. Introduction
- Part III. Methodology
- Part IV. Results
  - Visualization – Charts
  - Dashboard
- Part V. Discussion
  - Findings & Implications
- Part VI. Conclusion
- Part VII. Appendix

EXECUTIVE SUMMARY

- Summary of methodologies
  - Data Collection via API, SQL and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with NumPy, Pandas and SQL
  - Data Visualization with matplotlib and seaborn
  - Building an interactive dashboard with IBM Cognos
- Summary of results
  - Exploratory Data Analysis Result
    - Data Distribution
    - Outlier
    - Correlation
  - Interactive Visualization
    - Relationship
    - Composition
    - Comparison

INTRODUCTION

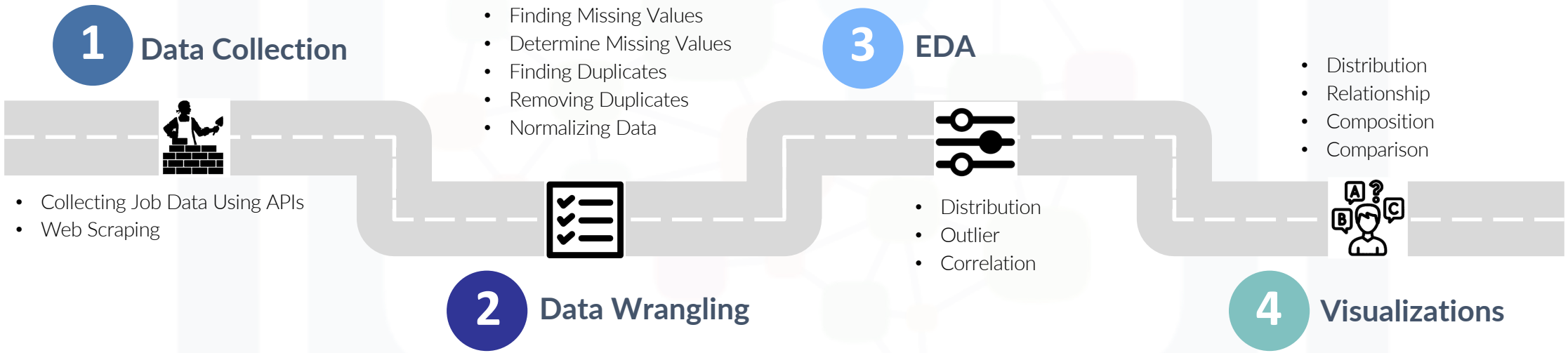
- **Project Background/Context**

  The world is changing rapidly with the huge evolution of storage power, computing and analytics techniques. As a global IT and business consulting services firm who is known for their expertise in IT solutions and their team of highly experienced IT consultants, it is necessary and valuable to regularly **analyze data to help identify future skill requirements**, in order to keep pace with changing technologies and remain flexible and competitive.

- **Insights and Trends to Drill**

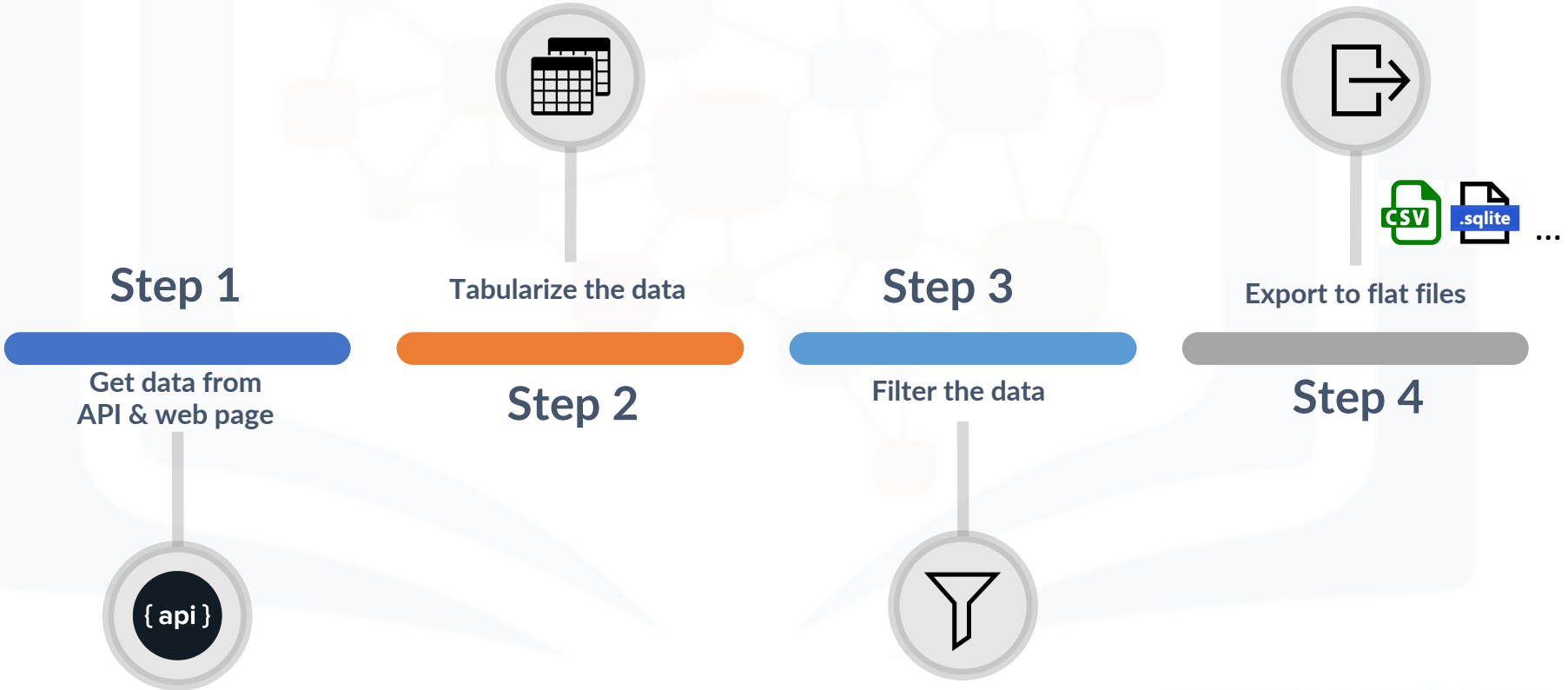
  - What are the top programming languages in demand?
  - What are the top database skills in demand?
  - What are the popular platforms?
  - What are the popular Web Frameworks?
  - …

Methodology – Bird Review



Methodology - Data Collection

Data collection is a process of gathering relevant information for answering the questions of interest. Usually, it is the first step for data analysis. Data can be collected in many ways according to the location and method they're stored. For this project, we collect the data by REST APIs and web scraping from Wikipedia page.



Data Collection – GitHub Jobs APIs

```
import requests

baseurl = "https://cf-courses-data.s3.us.cloud-object-storage.app
response = requests.get(baseurl)
```

```
def get_number_of_jobs(technology):
    number_of_jobs = 0
    #your code goes here
    response = requests.get(baseurl)
    data = response.json()
    for datum in data:
        if datum.get('A').lower() == technology.lower():
            number_of_jobs = datum.get('B')
            return technology,number_of_jobs
    if not number_of_jobs:
        return "WARNING! Enter right technology name!"
```

```
# Save the excel workbook
wb.save('github_job_postings.xlsx')
```

	technology	number of job posting
0	java	92
1	C	184
2	C#	14
3	C++	24
4	Java	92

Get response from API | **Convert response to .json file** | **Apply customized functions to read data** | **Create Workbook Object and store data in .xlsx** | **Filtering and export to flat files**

```
# Parse the response in .json format
data = response.json()

# Check
data
```

```
[{'A': 'technology', 'B': 'number of job posting'},
 {'A': 'java', 'B': '92'},
 {'A': 'C', 'B': '184'},
 {'A': 'C#', 'B': '14'},
 {'A': 'C++', 'B': '24'},
 {'A': 'Java', 'B': '92'},
 {'A': 'JavaScript', 'B': '65'},
 {'A': 'Python', 'B': '51'},
 {'A': 'Scala', 'B': '47'},
 {'A': 'Oracle', 'B': '6'},
 {'A': 'SQL Server', 'B': '16'},
 {'A': 'MySQL Server', 'B': '5'},
 {'A': 'PostgreSQL', 'B': '17'},
 {'A': 'MongoDB', 'B': '4'}]
```

```
from openpyxl import Workbook

# Create lists containing all technologies we need to find the number of postings
technology_list = ['java', 'C', 'C#', 'C++', 'Java', \
                   'JavaScript', 'Python', 'Scala', \
                   'Oracle', 'SQL Server', 'MySQL Server', \
                   'PostgreSQL', 'MongoDB']

# create a workbook object
wb=Workbook()
ws=wb.active

# Write in data in excel worksheet
ws.append(['technology', 'number of job posting'])
for tech in technology_list:
    ws.append(list(get_number_of_jobs(tech)))
```


Link to Github

Data Collection – Scraping Data from Websites

Screenshot from the web page:

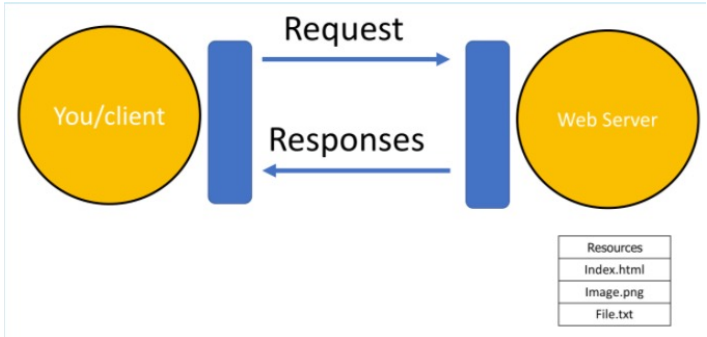
Popular Programming Languages

Finding out which is the best language is a tough task. A programming language is created to solve a specific problem. A language which is good for task A may not be able to properly handle task B. Comparing programming language is never easy. What we can do, however, is find which is popular in the industry.

There are many ways to find the popularity of a programming languages. Counting the number of google searchs for each language is a simple way to find the popularity. GitHub and StackOverflow also can give some good pointers.

Salary surveys are a way to find out the programmings languages that are most in demand in the industry. Below table is the result of one such survey. When using any survey keep in mind that the results vary year on year.

No.	Language	Created By	Average Annual Salary	Learning Difficulty
1	Python	Guido van Rossum	$114,383	Easy
2	Java	James Gosling	$101,013	Easy
3	R	Robert Gentleman, Ross Ihaka	$92,037	Hard
4	Javascript	Netscape	$110,981	Easy
5	Swift	Apple	$130,801	Easy
6	C++	Bjarne Stroustrup	$113,865	Hard
7	C#	Microsoft	$88,726	Hard
8	PHP	Rasmus Lerdorf	$84,727	Easy
9	SQL	Donald D. Chamberlin, Raymond F. Boyce.	$84,793	Easy
10	Go	Robert Griesemer, Ken Thompson, Rob Pike.	$94,082	Difficult



Data Collection – Scraping Data from Websites

Workflow

- Import Required Libraries
- Get response from HTML
- Create Beautiful Soup Object
- Find tables
- Get table cell data and store in lists
- Create DataFrame and store in data
- Export to flat files (.csv)

Code Sippets

```
# Import required libraries
import requests
from bs4 import BeautifulSoup
```

```
#this url contains the data you need to scrape
url = "https://cf-courses-data.s3.us.cloud-object-storage.appd

# Download web page
web_dt = requests.get(url).text
```

```
# Create soup object
soup = BeautifulSoup(web_dt, 'html5lib')
```

```
# Find table target
tb = soup.find('table')
tb
```

```
# Find language names and annual average salaries
# in the table and store them in list objects
langs = []
salaries = []
for row in tb.find_all('tr'):
    cells = row.find_all('td')
    langs.append(cells[1].getText())
    salaries.append(cells[3].getText())
```

```
# Store and export to .csv format
import pandas as pd
df = pd.DataFrame({'Language_Name':langs[1:], "Annual_Average_Salaries": salaries[1:]})
df.to_csv('lang_salary.csv', index=False)
```

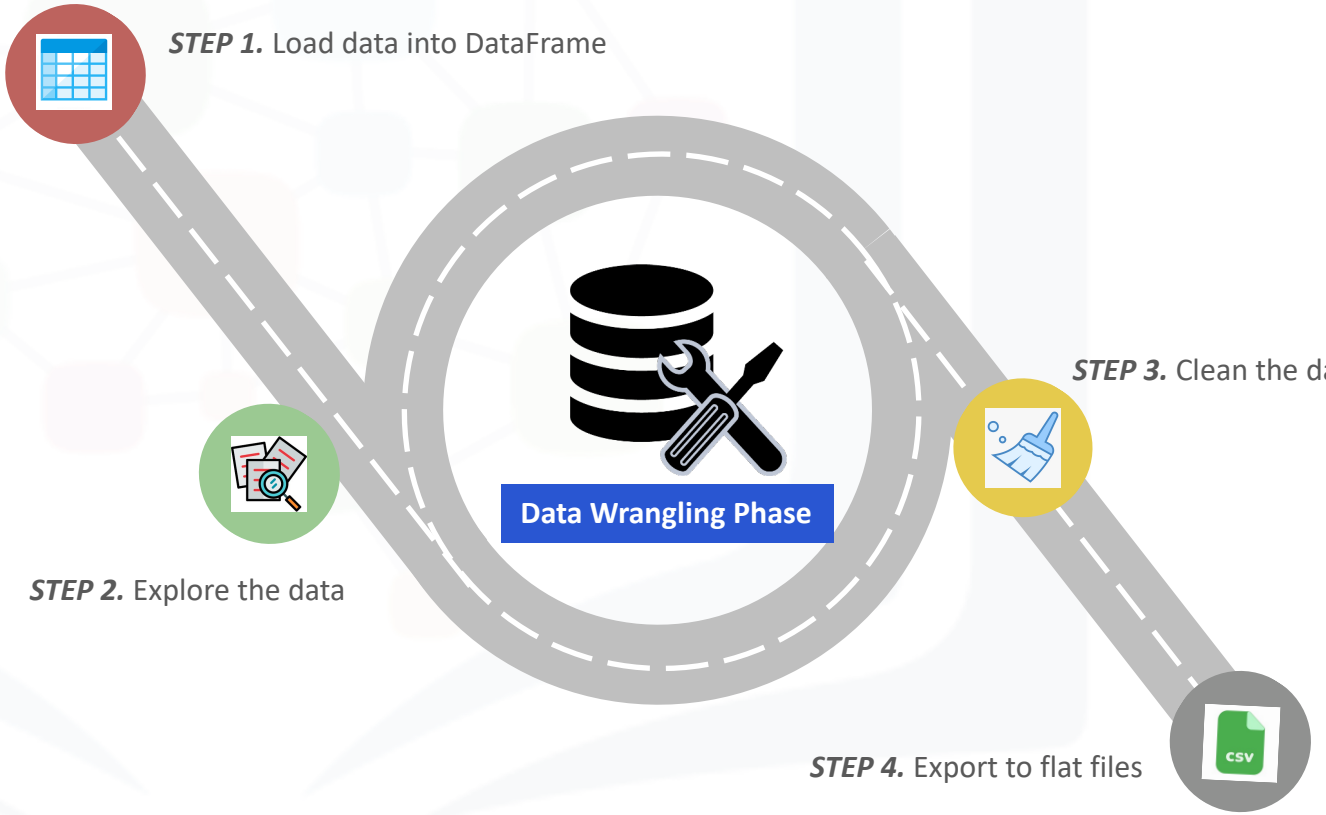
Result

	Language_Name	Annual_Average_Salaries
0	Python	$114,383
1	Java	$101,013
2	R	$92,037
3	Javascript	$110,981
4	Swift	$130,801
5	C++	$113,865
6	C#	$88,726
7	PHP	$84,727
8	SQL	$84,793
9	Go	$94,082

Data Wrangling

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.

- Identify duplicate values in the dataset.
- Remove duplicate values from the dataset.
- Identify missing values in the dataset.
- Impute the missing values in the dataset.
- Normalize data in the dataset.



Data Wrangling

Explore the dataset

- **Shape of Data:**
  11,552 rows and 85 columns
- **Data Types:**
  int64, object, float64
- **Avg. Age of Respondents:**
  31 y/o
- **Origin of Respondents:**
  135 countries

Finding Duplicates

- **Number of duplicates:**
  154 rows in the dataset
- **Shape after duplicates removed:**
  11,398 rows and 85 columns
- **Unique Respondent(ID):**
  11,398

Finding Missing Values
(Duplicate rows removed)

- **Missing Values under EdLevel:**
  112
- **Missing Values under Country:**
  0
- **Missing Values under WorkLoc:**
  32

Normalizing Data

- **Types of CompFreq:**
  3 (Yearly, Monthly, Weekly)
- **Number of respondents paid yearly:**
  6,073

Data Wrangling

Normalizing Data

There are two columns in the dataset that talk about compensation. One is **CompFreq**. This column shows how often a developer is paid (Yearly, Monthly, Weekly).The other is **CompTotal**. This column talks about how much the developer is paid per Year, Month, or Week depending upon his/her CompFreq. This makes it difficult to compare the total compensation of the developers. So we created a new column called **NormalizedAnnualCompensation** which contains the **Annual Compensation** irrespective of the 'CompFreq'.

- What is the median **NormalizedAnnualCompensation**?

```
# your code goes here
df.loc[df['CompFreq'] == 'Yearly', 'NormalizedAnnualCompensation'] = 1 * df['CompTotal']
df.loc[df['CompFreq'] == 'Monthly', 'NormalizedAnnualCompensation'] = 12 * df['CompTotal']
df.loc[df['CompFreq'] == 'Weekly', 'NormalizedAnnualCompensation'] = 52 * df['CompTotal']
```

After normalizing, we find the median value for **NormalizedAnnualCompensation** is **$100,000.**

Data Visualization

Scatter Plots

Scatter plots can visualize dependencies of attributes with each other. This is usually called correlation. Once a pattern is determined, it will help us to predict which factor would lead to the success of landing outcome.

Bar Graphs

Bar graph is a great way to visualize categorical data type. Each bar represents a member under that categorical feature, and it is easy to compare which member has the largest amount and which the least.

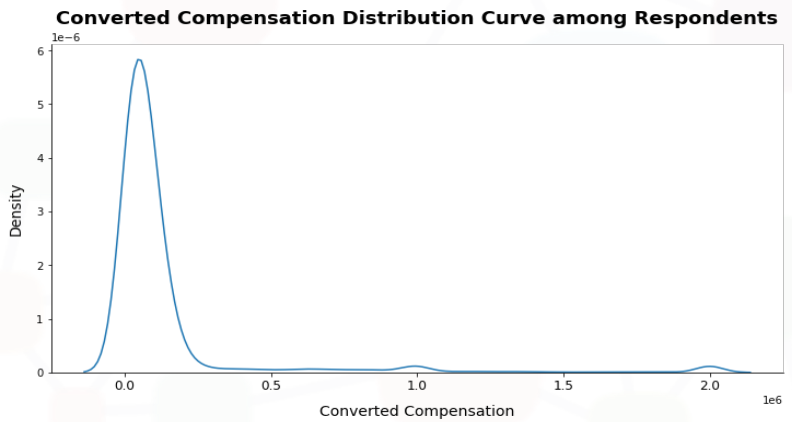
Line Plot

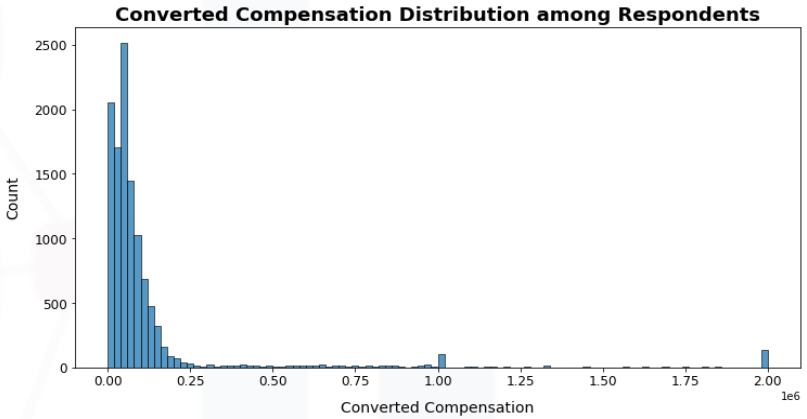
Line Plot is similar to scatter plot while the former is often used to visualize continuous numeric data type. From line plot, we can easily tell the trend between two variables thus helping us to make prediction about the future.

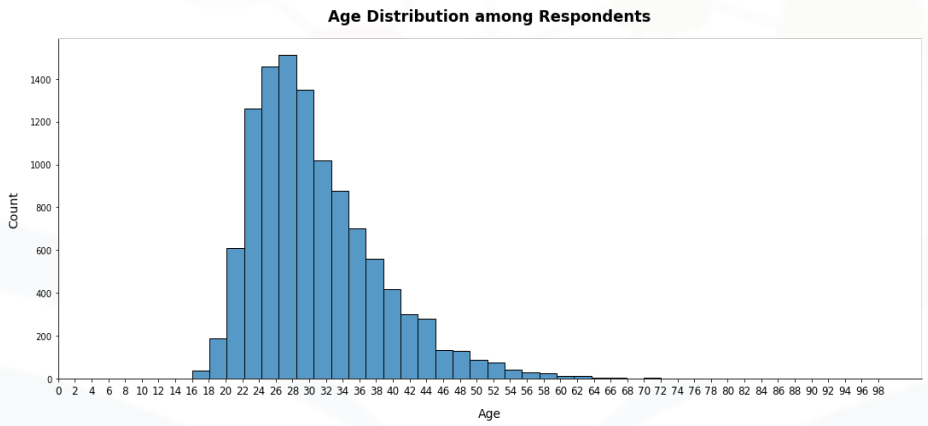
RESULTS - Distribution

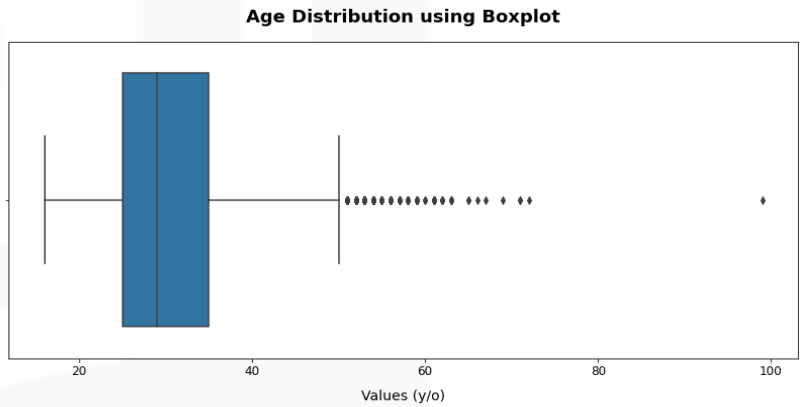
Explanation

- Both the converted compensation and age distribution have serious right-skewed shape from the histogram plots on the right.
- The majority of respondents have their converted compensation between 0 and $250,000
- The majority of respondents are aged between 24 and 32 y/o. And ages beyond 50 are regarded as outlier by convention.







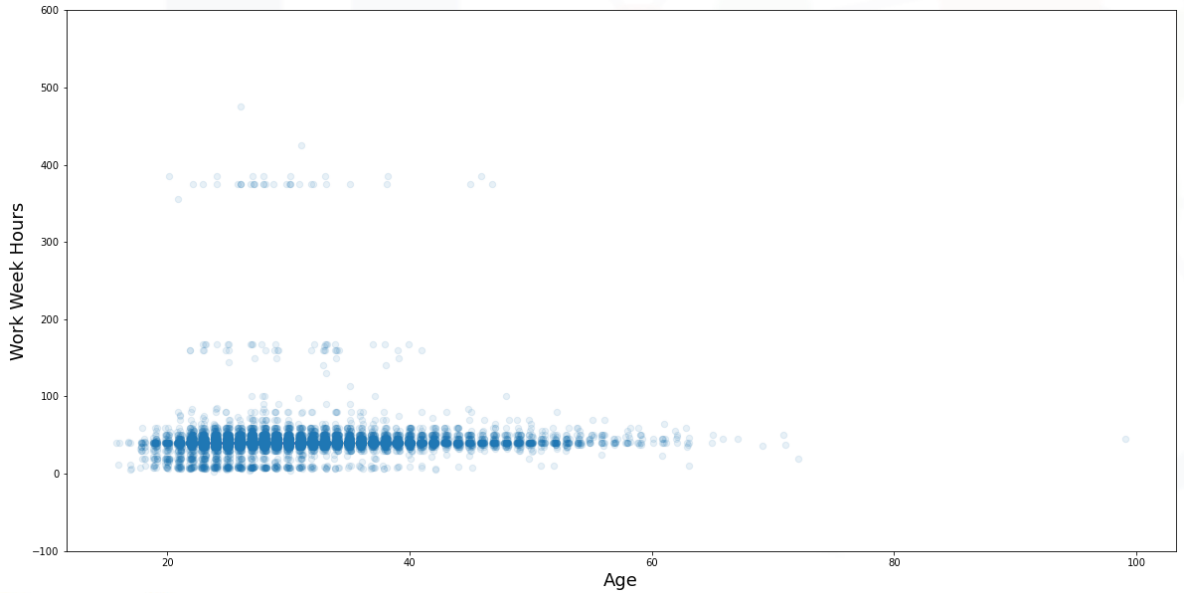


RESULTS - Relationship

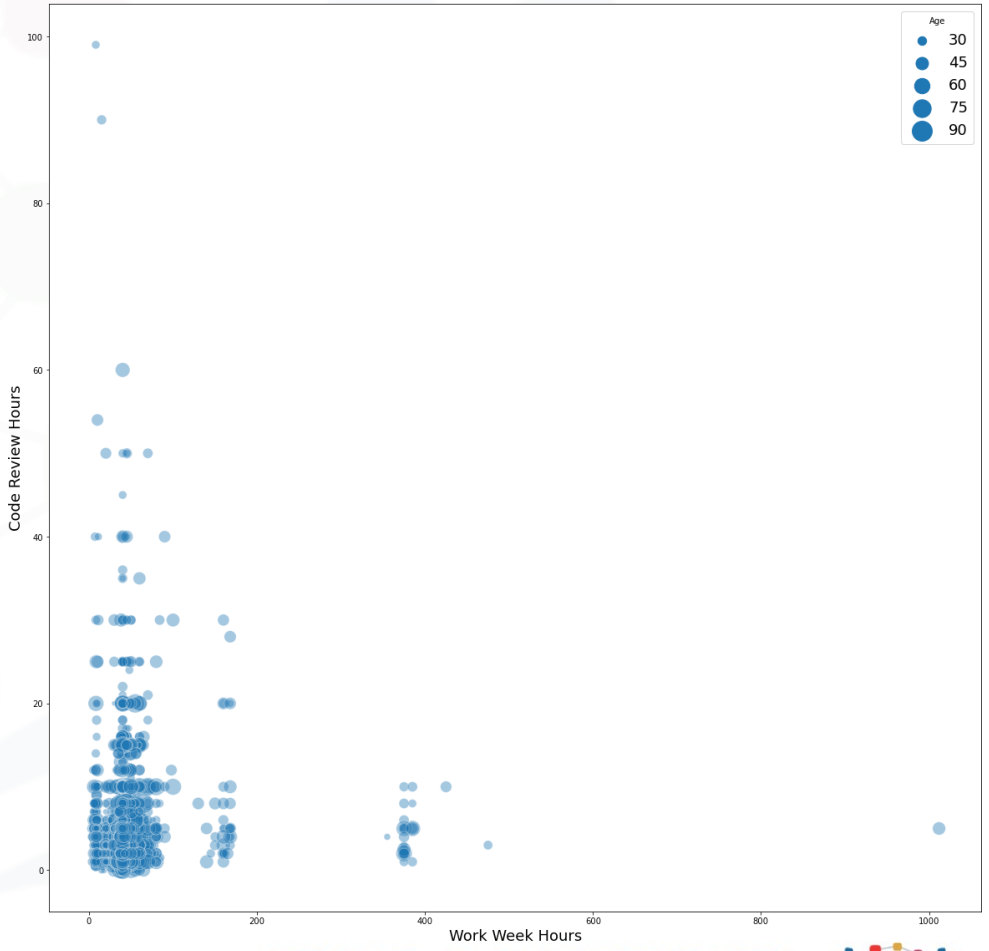
Explanation

- Workweek hours, above 100 can be regarded as outliers as this make little sense in reality. So, from the scatter plot below, workweek hours mainly concentrated around 50-70/w, throughout the age between 20 and 60 y/o, although a small group of respondents aged between 20 and 40 y/o claims to have workweek hours around 20.
- Code review hours may vary from 0 up to 60 hours/w, but mainly focus around 0-10 hours/w. In addition, the age does not seem to have any impact on the difference of workweek hours or hours for code review.

WorkWeekHours v.s. Age



WorkWeekHours v.s. CodeReviewHours

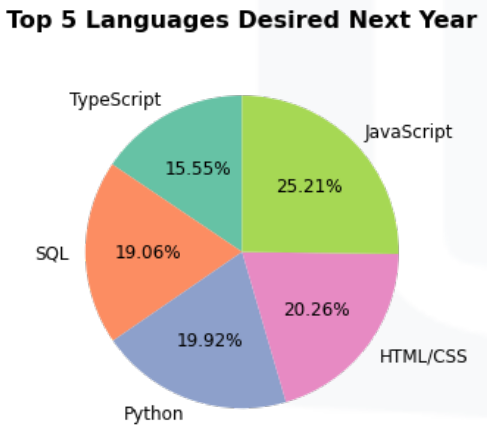


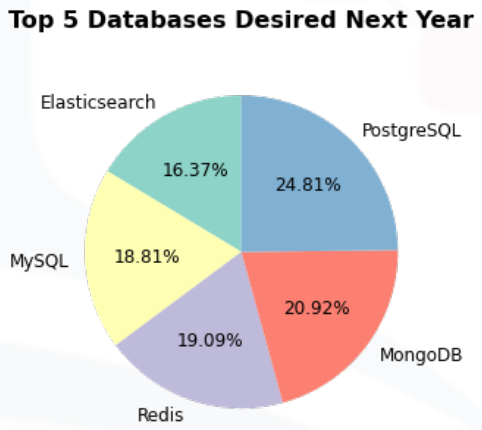
IBM Developer

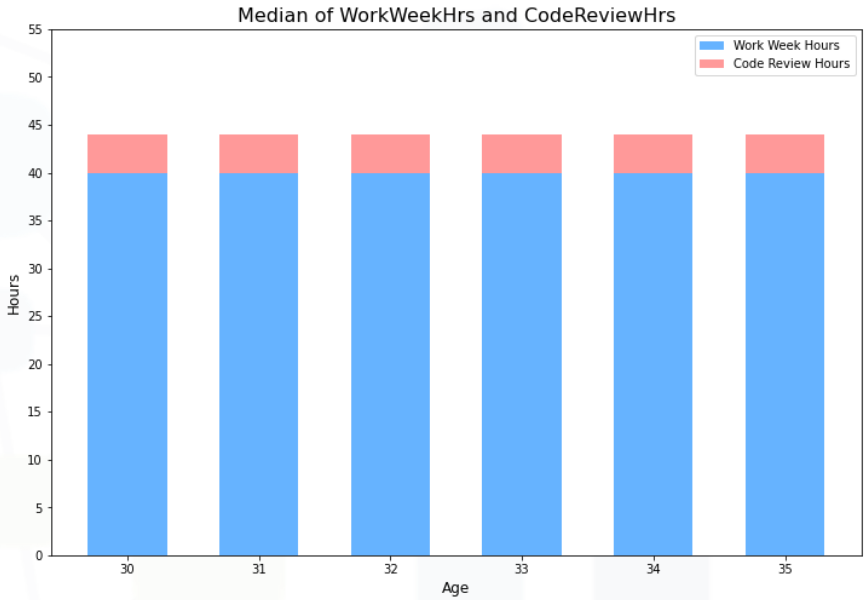
SKILLS NETWORK

RESULTS - Composition

- **JavaScript**, **HTML/CSS** and **Python** are the most desired programming languages for the next year, covering over 60% of the top 5.
- **PostgreSQL, MongoDB and Redis** are the most desired databases for the next year







The total of **work week hour** and **code review hours** among respondents aged between **30 to 35** are almost the same.

RESULTS - Comparison

Explanation

- **Fig.1** : The majority of the respondents regard coding as their professions, while less than 1/10 of them just code as part of their work;
- **Fig.2** : Respondents aged between **25 to 30** have a positive trend between their converted compensation and age;
- **Fig.3** : Respondents aged between **45 to 60** generally have a positive trend with more fluctuations between their converted compensation and age; however, their compensation range is much larger compared with that of the group aged between 25 to 30 y/o. It makes sense that their experiences and capabilities can contribute a lot to some extent.

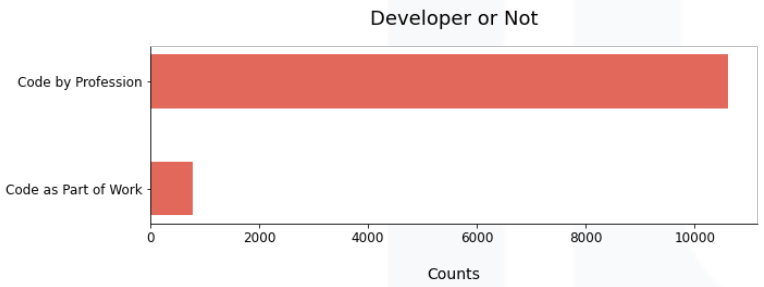


Fig.1

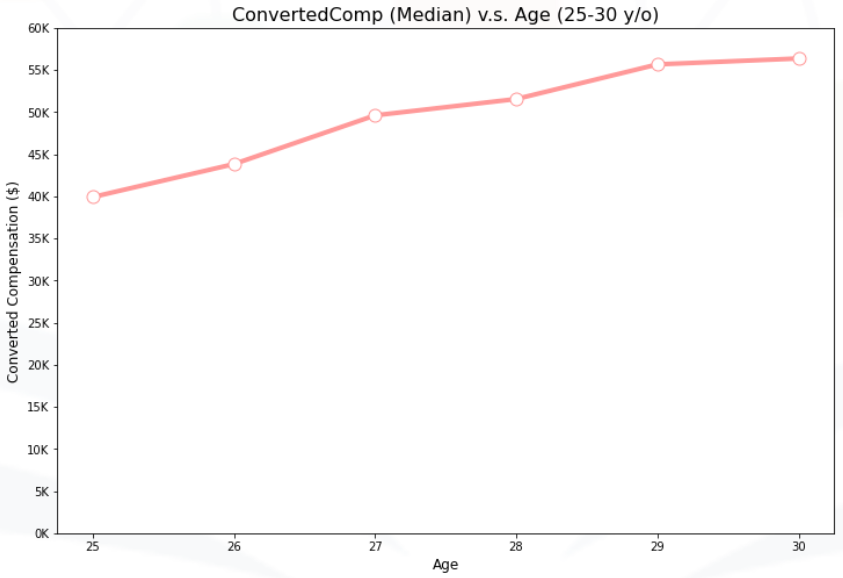


Fig.2

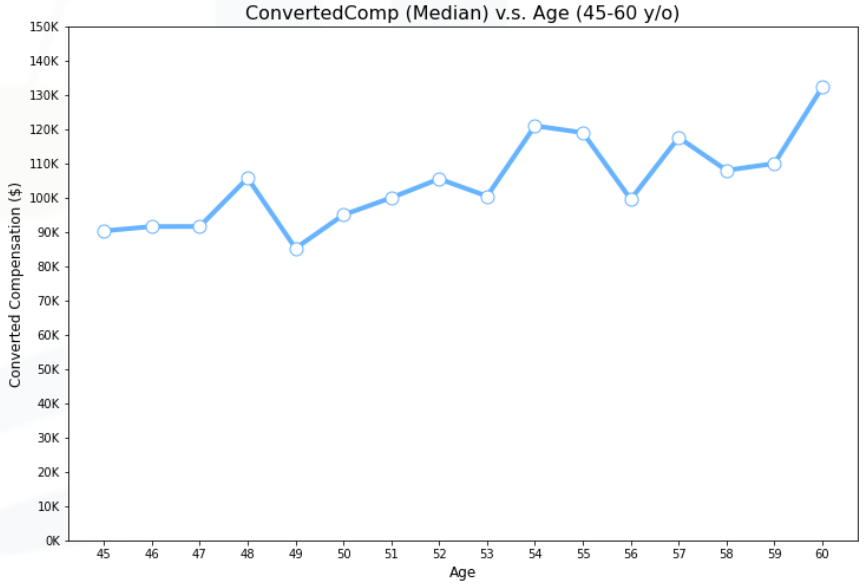
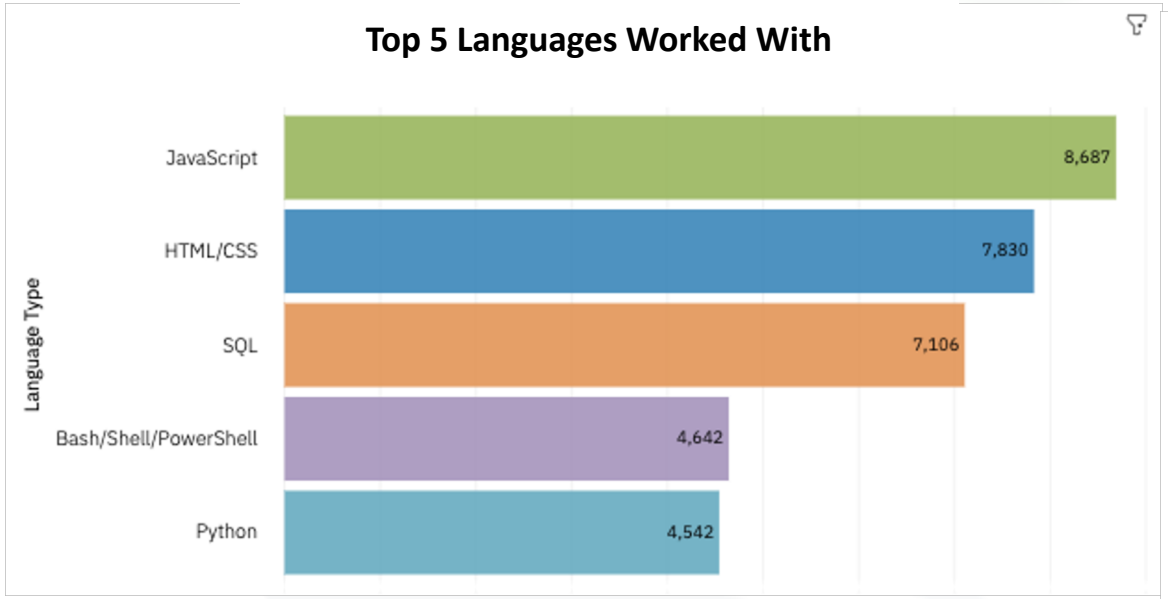


Fig.3

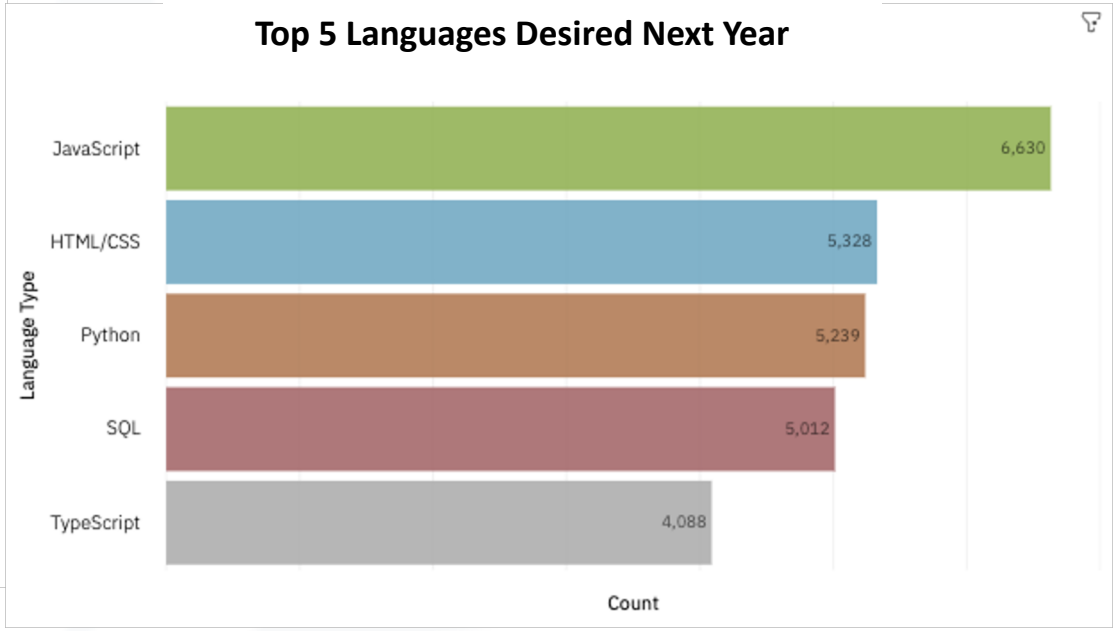
RESULTS – Features of Interest

PROGRAMMING LANGUAGE TRENDS

Current Year



Next Year



PROGRAMMING LANGUAGE TRENDS

Findings

- ***JavaScript***, ***HTML/CSS*** and ***SQL*** are among the most popular languages to work with at present.
- Currently, the usage of ***JavaScript*** is almost 2 times as large as that of Bash/Shell/Power Shell and Python.
- ***JavaScript***, ***HTML/CSS*** are still regarded as the most popular languages desired to work with next year.
- ***Python*** seems to be highly anticipated next year as it jumps up rapidly just close to ***JavaScript*** and ***HTML/CSS***. The differences between them is decreased sharply, compared to the present.
- For next year, ***TypeScript*** is an emerging language ranked among the top 5.

Implications

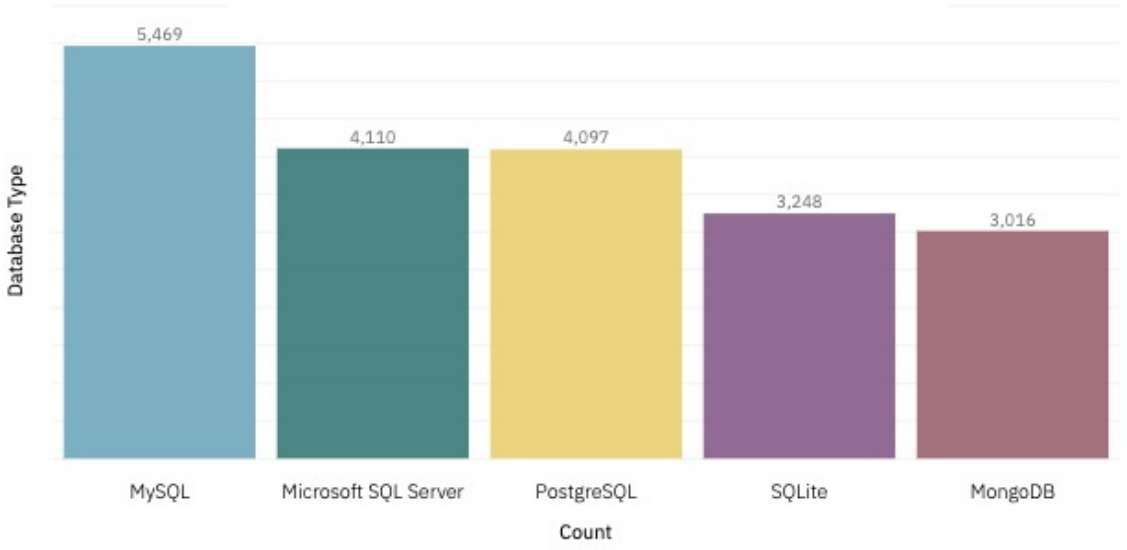
- ***JavaScript***, ***HTML/CSS*** and ***SQL*** seem to be the must-have language skillset in the market. But whether they will be oversupplied or not in future needs more investigation.
- The increasing speed for ***Python*** popularity is incredible, maybe out of its flexibility, versatile capabilities of multiple scenarios, and the power for data processing and data science.
- ***TypeScript*** is taken notice by the market, and it deserves a tight and further follow-up.

RESULTS – Features of Interest

DATABASE TRENDS

Current Year

Top 5 Database Worked With



Next Year

Top 5 Database Desired Next Year

DATABASE TRENDS

Findings

- Traditional databases like ***MySQL*** and ***Microsoft SQL Server*** dominate at present, covering almost half of the market.
- Currently ***PostgreSQL*** ranks the third, but it seems to be trying to track down the Microsoft SQL Server as the gap between the two is quite small.
- Unexpectedly, ***PostgreSQL*** jumps to the **No.1** database desired to work with next year.
- ***MongoDB***, ***Redis*** and ***Elasticsearch*** come into sight among the desired databases to work with.

Implications

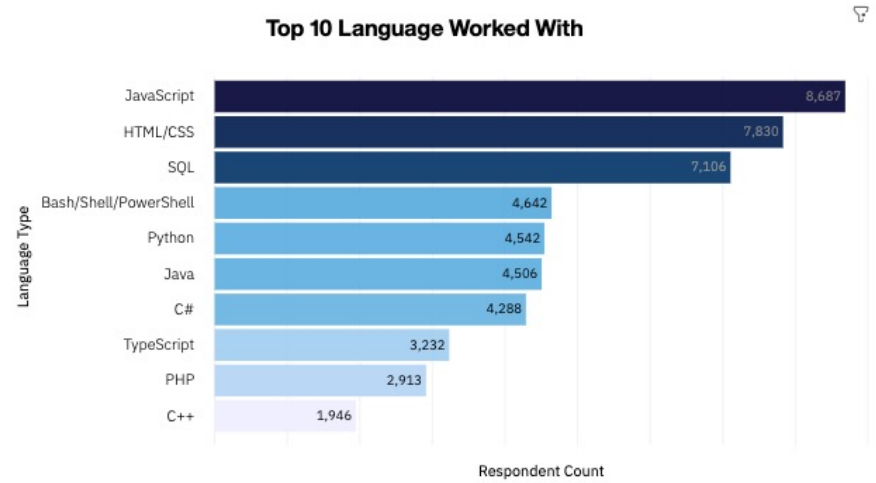
- Traditional databases are losing market. On one hand, it may imply that the outer environment is changing and making some significant transformation. On the other hand, traditional database vendors seem to be in urgent need for close follow-up to the business demand and openness for more innovation and trials.
- ***NoSQL*** databases are becoming popular in near future out of the fast evolution of big data technology and application.

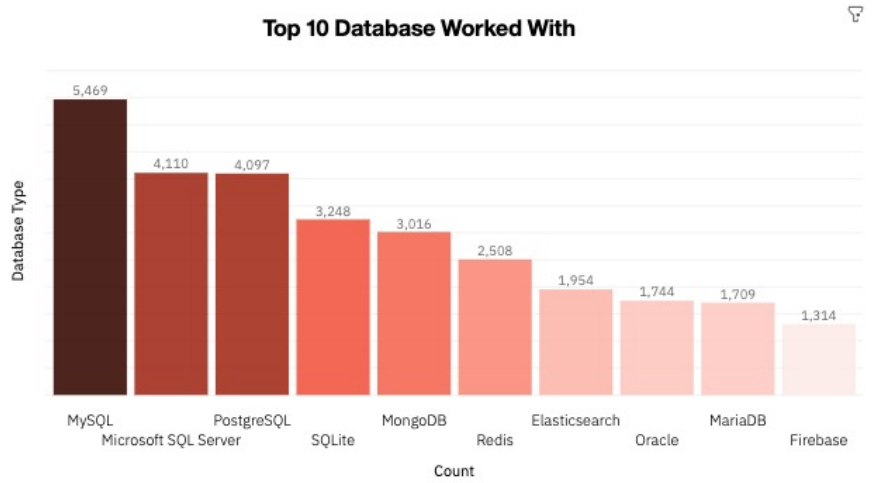
DASHBOARD

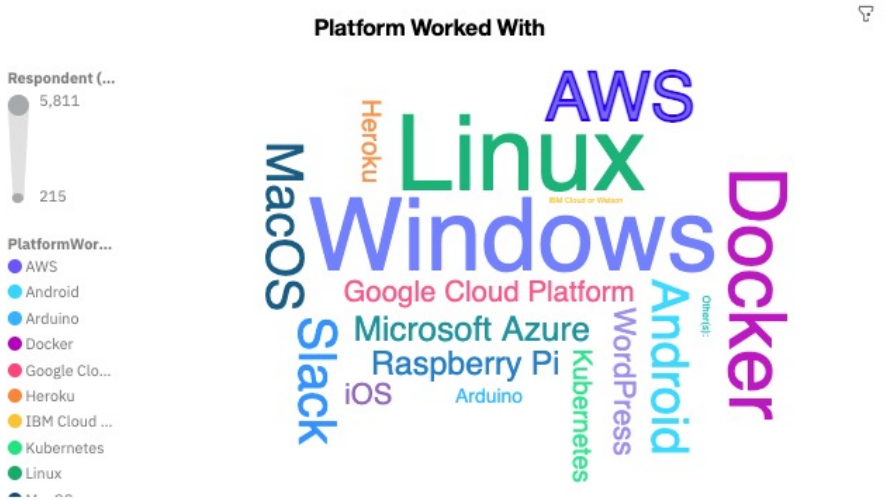
Additional visualization analytics is enhanced by IBM Cognos.

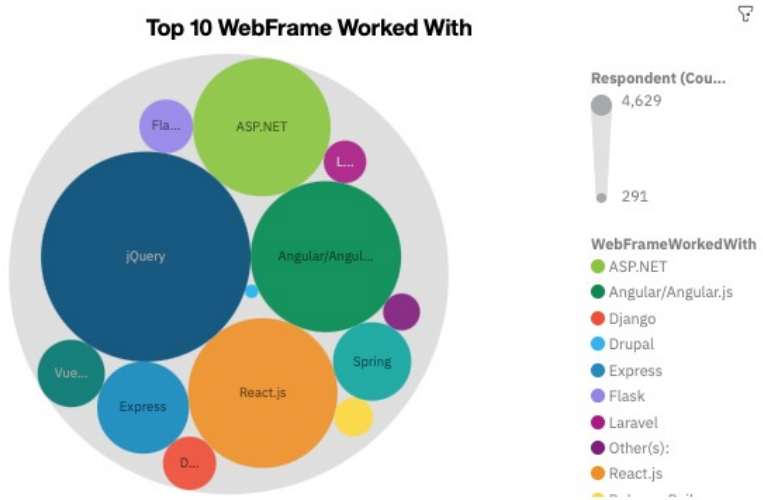
Click this **link** for the Dashboard on-live.

DASHBOARD TAB 1 – *Current Technology Usage*





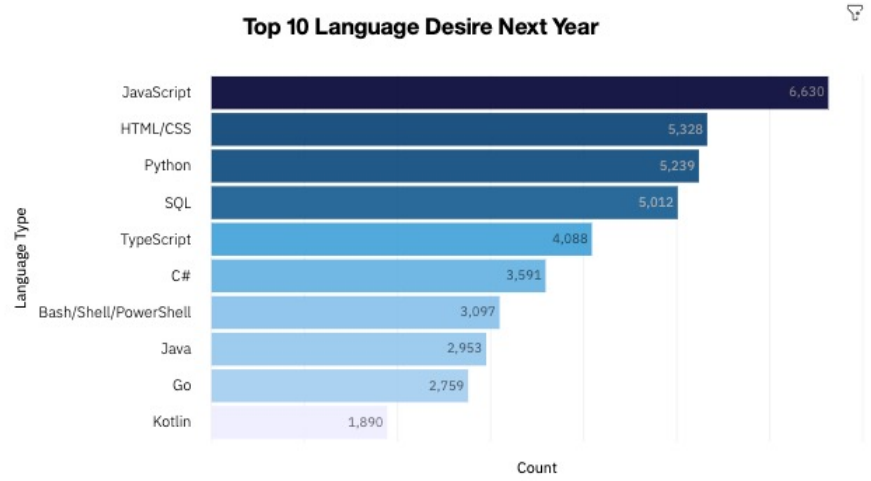


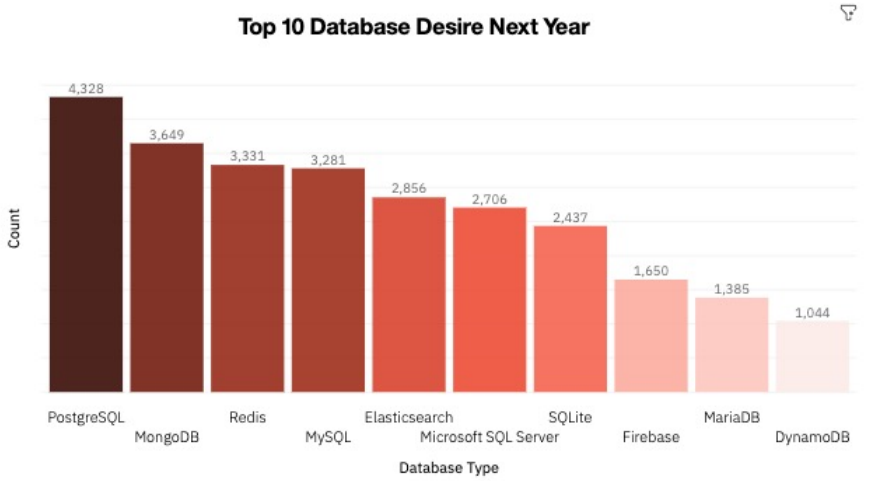


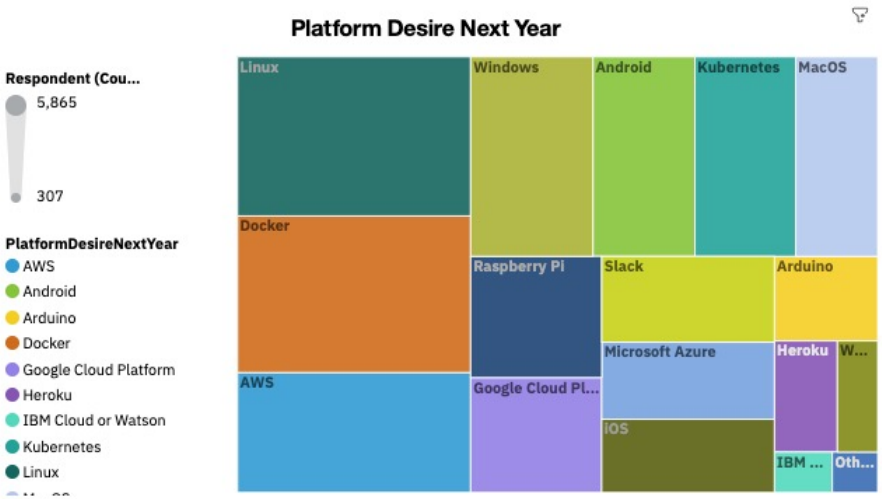
IBM Developer

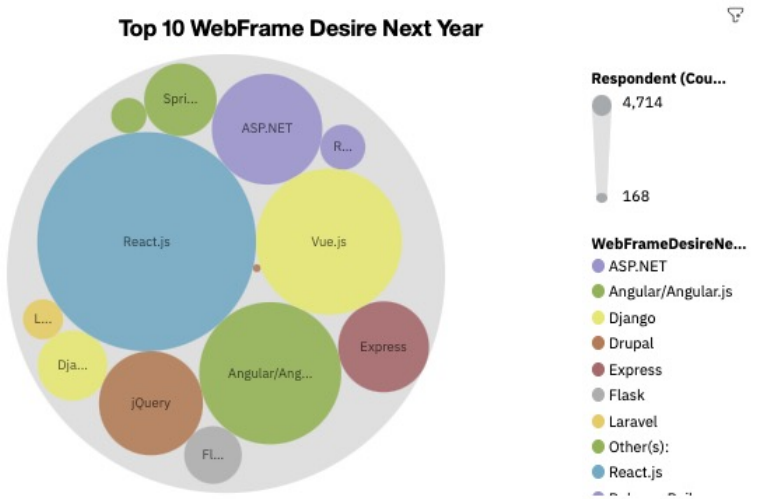
SKILLS NETWORK

DASHBOARD TAB 2 – *Future Technology Trend*





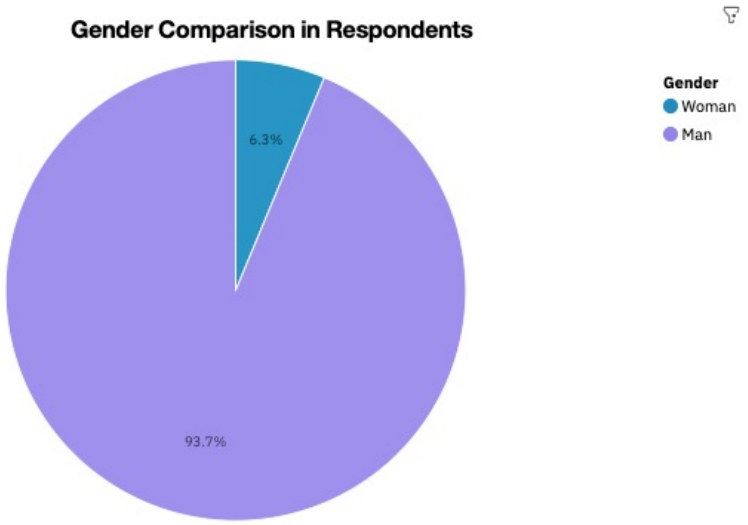


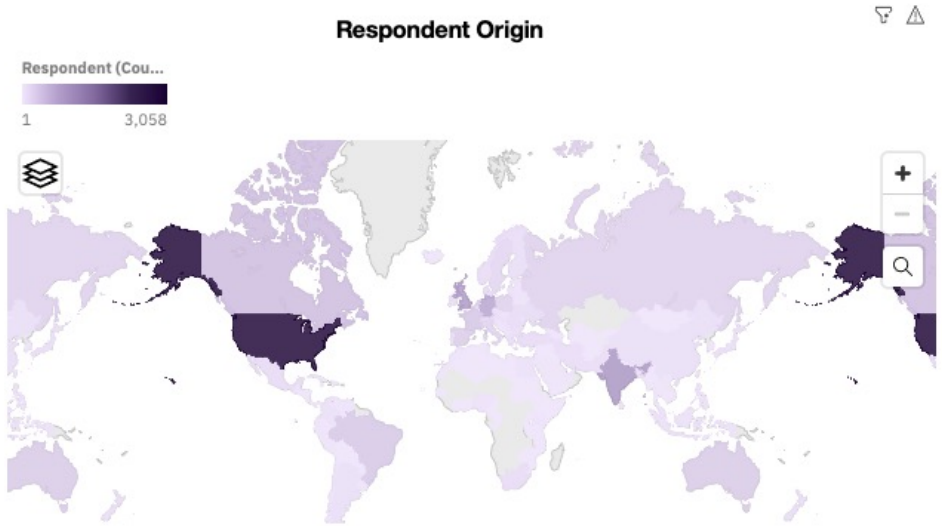


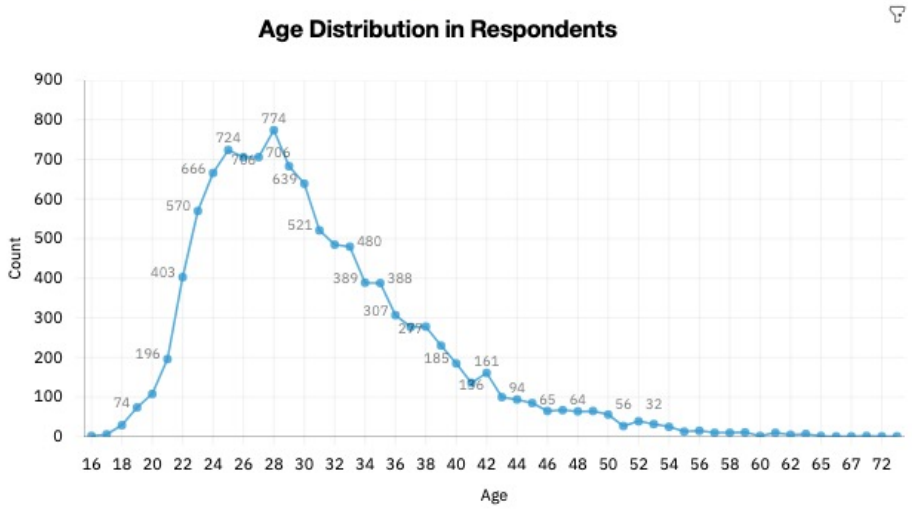
IBM Developer

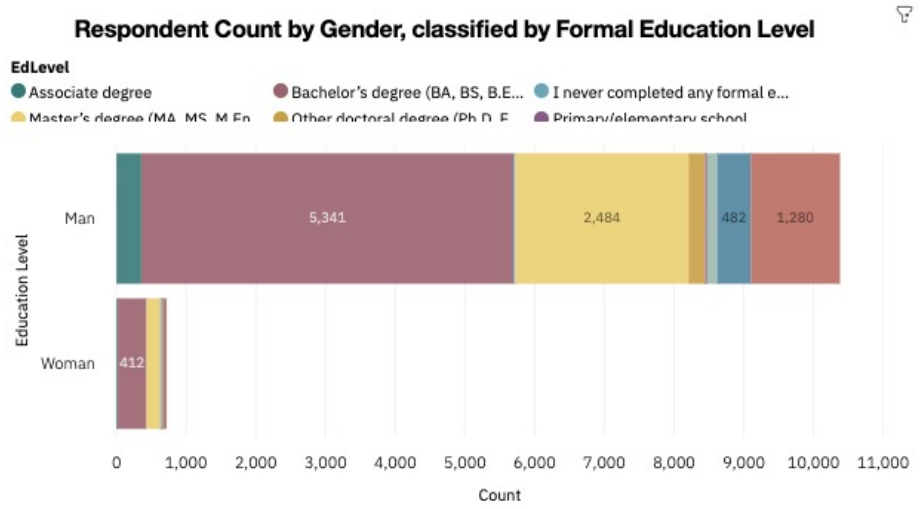
SKILLS NETWORK

DASHBOARD TAB 3 – *Demographics*









IBM Developer

SKILLS NETWORK

DISCUSSION

- What **attributes** should a programmer must have to stay competitive and flexible in a fast-changing technological environment?
- What **technical skillset** must a programmer have in a big-data dominated world, in terms of programming languages, databases, operating system platforms, and even web frameworks?

OVERALL FINDINGS & IMPLICATIONS/CONCLUSIONS

Findings

- Among programming languages, JavaScript, HTML/CSS will remain the sought-after in the market, no matter it's at present or in near future.
- Among databases, PostgreSQL is gaining more and more popular, compared with its competitors like MySQL, Microsoft SQL Server, etc.
- Among OS platforms, Linux is still vibrant in the market, but Docker and AWS are closely catching up with.
- Among Web Frameworks, jQuery, ASP.NET, React.js are the most popular in programmers.

Implications

- Python will be a hot sought-after programming language skillset in near future， benefited by the big-data and AI-powered age.
- NoSQL databases will share the market with the traditional ones as they are more adaptable to some certain complex scenarios.
- On-Cloud OS stay competitive and indispensable in future out of its features such as low cost, easy maintenance and fast technological evolution.
- Master of multiple web frameworks is highly necessary in future in order to face with more complex and flexible web using scenarios.

APPENDIX: References

Data Source Being Used:

- Githubposting
- Programming Languages
- Module 1 Survey Data
- Module 2 Survey Data
- Module 4 Survey Data
- Module 5 Survey Data – demographics
- Module 5 Survey Data – technologies normalized

Link to files and codes on GitHub:

- Project Link

APPENDIX: Data Wrangling

- **Find the value counts for the column WorkLoc. (Office)**

```
# your code goes here
df.WorkLoc.value_counts()
```

```
]: Office                                          6806
   Home                                            3589
   Other place, such as a coworking space or cafe   971
   Name: WorkLoc, dtype: int64
```

- **What is the majority category under the column Employment? (employed full-time)**

```
# Find the majority category under `Employment`
df.Employment.value_counts()
```

```
]: Employed full-time    10968
   Employed part-time      430
   Name: Employment, dtype: int64
```

- **Under the column " UndergradMajor", which category has the minimum number of rows? (health science)**

```
# Find the majority category under `UndergradMajor`
df.UndergradMajor.value_counts()
```

```
]: Computer science, computer engineering, or software engineering          6953
   Information systems, information technology, or system administration     794
   Another engineering discipline (ex. civil, electrical, mechanical)        759
   Web development or web design                                             410
   A natural science (ex. biology, chemistry, physics)                       403
   Mathematics or statistics                                                 372
   A business discipline (ex. accounting, finance, marketing)                244
   A social science (ex. anthropology, psychology, political science)        210
   A humanities discipline (ex. literature, history, philosophy)             207
   Fine arts or performing arts (ex. graphic design, music, studio art)      161
   I never declared a major                                                  124
   A health science (ex. nursing, pharmacy, radiology)                        24
   Name: UndergradMajor, dtype: int64
```

APPENDIX: Data Wrangling

- **What is the median of `ConvertedComp`?** ($57,745)

- **What is the median ConvertedComp of respondents who have identified themselves as 'Woman'?** ($57,708)

```
# your code goes here
df[df.Gender=='Woman'].ConvertedComp.median()
```

```
]: 57708.0
```

- **What is the upper and lower limit for ConvertedComp outliers?**

Calculation Criteria:

Large Outliers: Q3 + 1.5*IQR
Smaller Outliers: Q1 - 1.5*IQR

```
# your code goes here
max_limit = q3 + 1.5*q_range
min_limit = q1 - 1.5*q_range
max_limit, min_limit
```

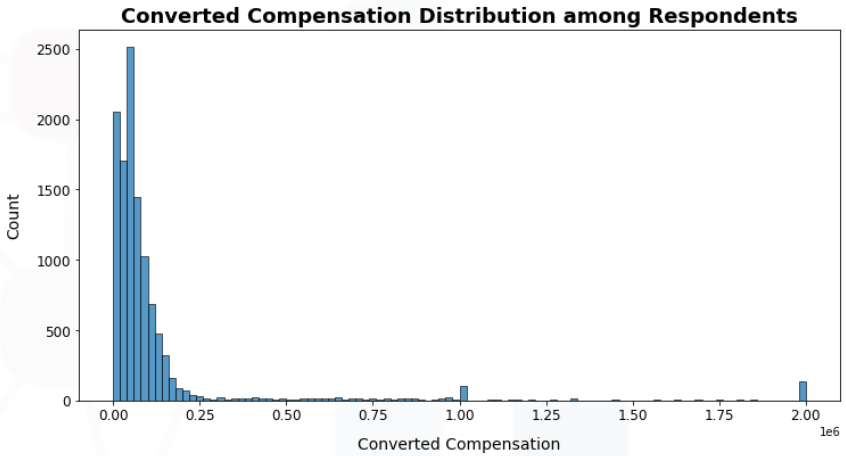
```
']: (209698.0, -82830.0)
```

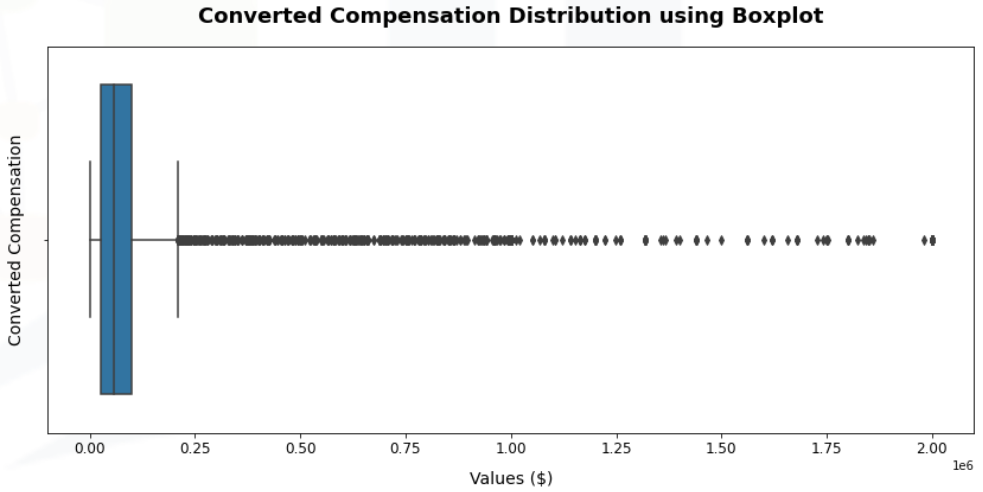
```
count_low_outlier = df.ConvertedComp.lt(min_limit).sum()
count_high_outlier = df.ConvertedComp.gt(max_limit).sum()
print("Total count of outliers is: {}".format(count_high_outlier+count_low_outlier))
```

```
Total count of outliers is: 879
```

```
# Filter out outliers in ConvertedComp
df1 = df[(df.ConvertedComp>=min_limit) & (df.ConvertedComp<=max_limit)]
df1.shape
```

```
]: (9703, 85)
```





IBM Developer

SKILLS NETWORK

APPENDIX: Data Wrangling

- **What is the median ConvertedComp before removing outliers?** ($57,745)
- **What is the median ConvertedComp after removing outliers?** ($52,704)
- **What is the mean ConvertedComp after removing outliers?** ($59,883)

Before removing outliers:

```
df.ConvertedComp.describe().apply(lambda s: format(s, 'f'))
```

```
]: count      10582.000000
   mean      131596.731620
   std       294786.515775
   min            0.000000
   25%        26868.000000
   50%        57745.000000
   75%       100000.000000
   max      2000000.000000
```

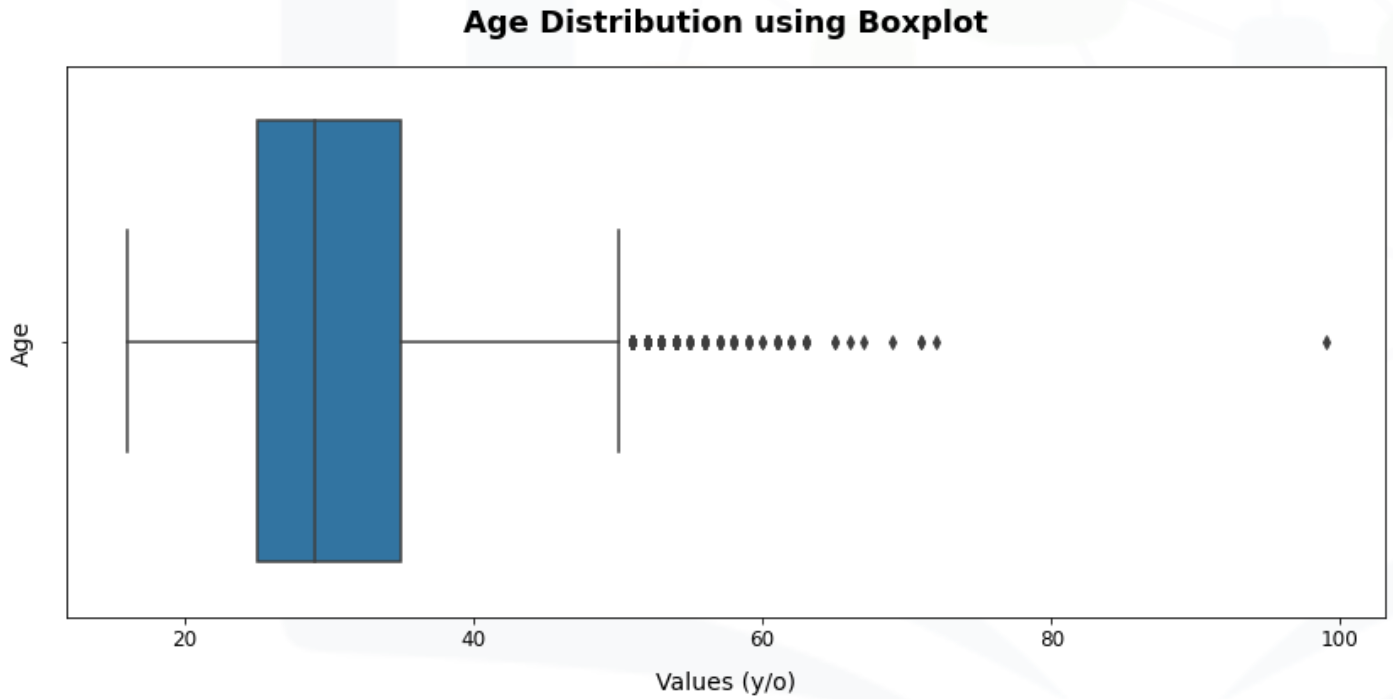
After removing outliers:

```
df1.ConvertedComp.describe()
```

```
]: count      9703.000000
   mean      59883.208389
   std       43394.336755
   min           0.000000
   25%       24060.000000
   50%       52704.000000
   75%       85574.500000
   max      209356.000000
```

APPENDIX: Data Wrangling

- **Based on the boxplot of the column “Age”, where do you see the outliers?** (Above Q3)
- **What is the median Age of survey respondents?** (29 y/o)



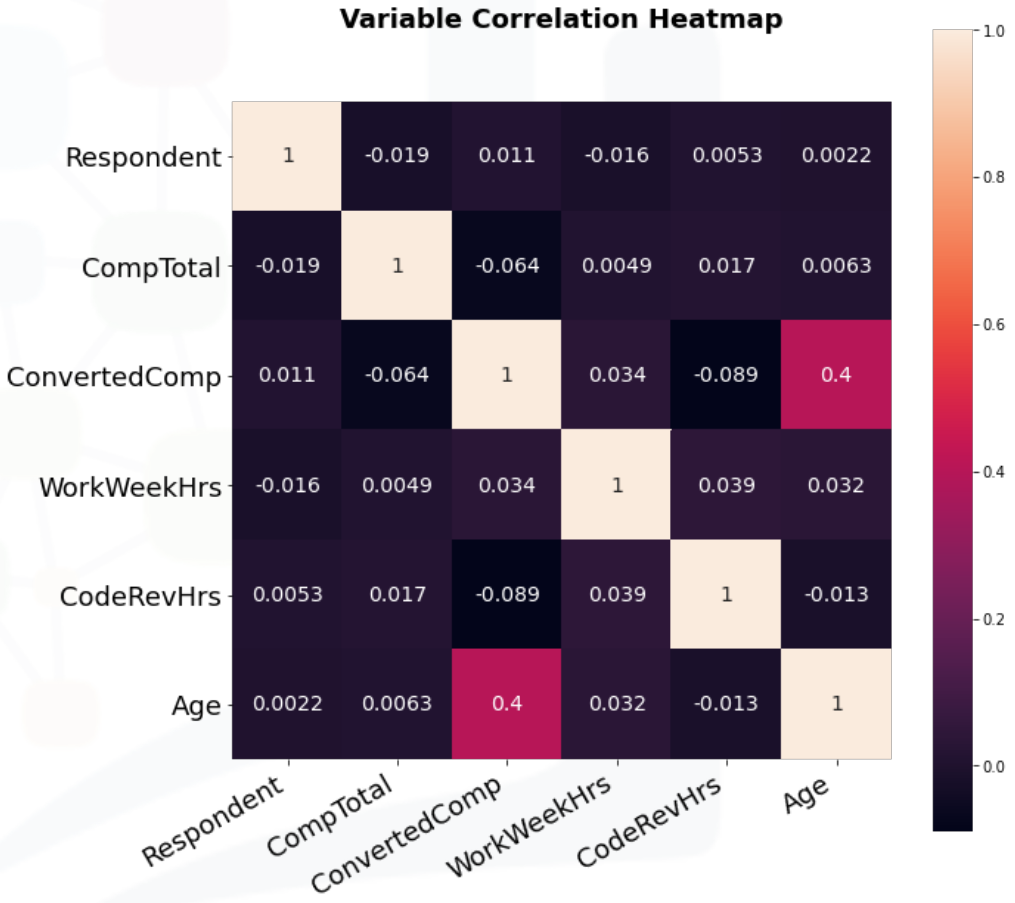

```
df.Age.describe()
```

```
count    11111.000000
mean        30.778895
std          7.393686
min         16.000000
25%         25.000000
50%         29.000000
75%         35.000000
max         99.000000
Name: Age, dtype: float64
```

APPENDIX: Data Wrangling

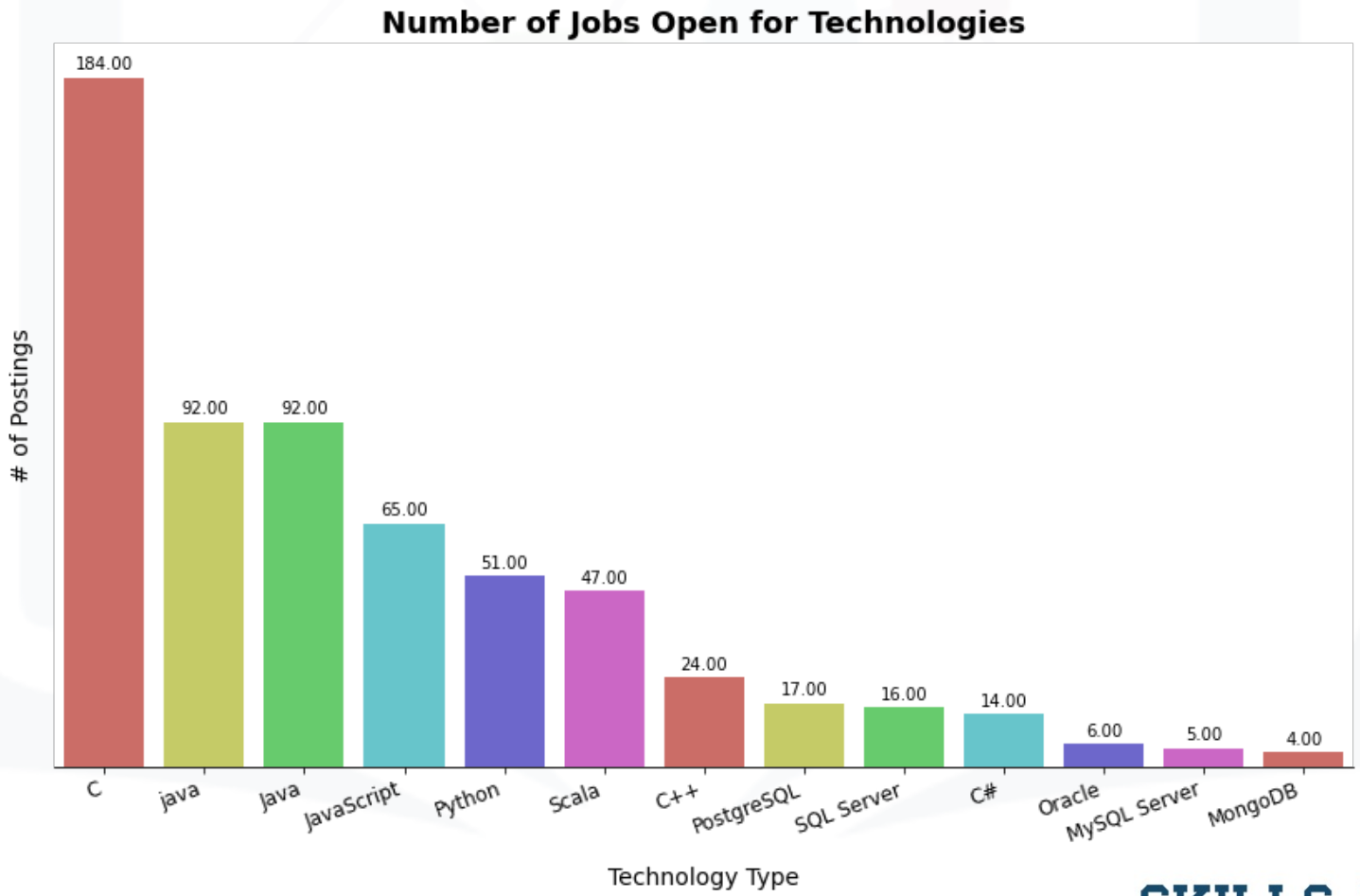
- **Which column has a negative correlation with “Age”?**
  CodeRevHrs

- **Which column has the highest correlation with “Age”?**
  ConvertedComp



GitHub Job Postings

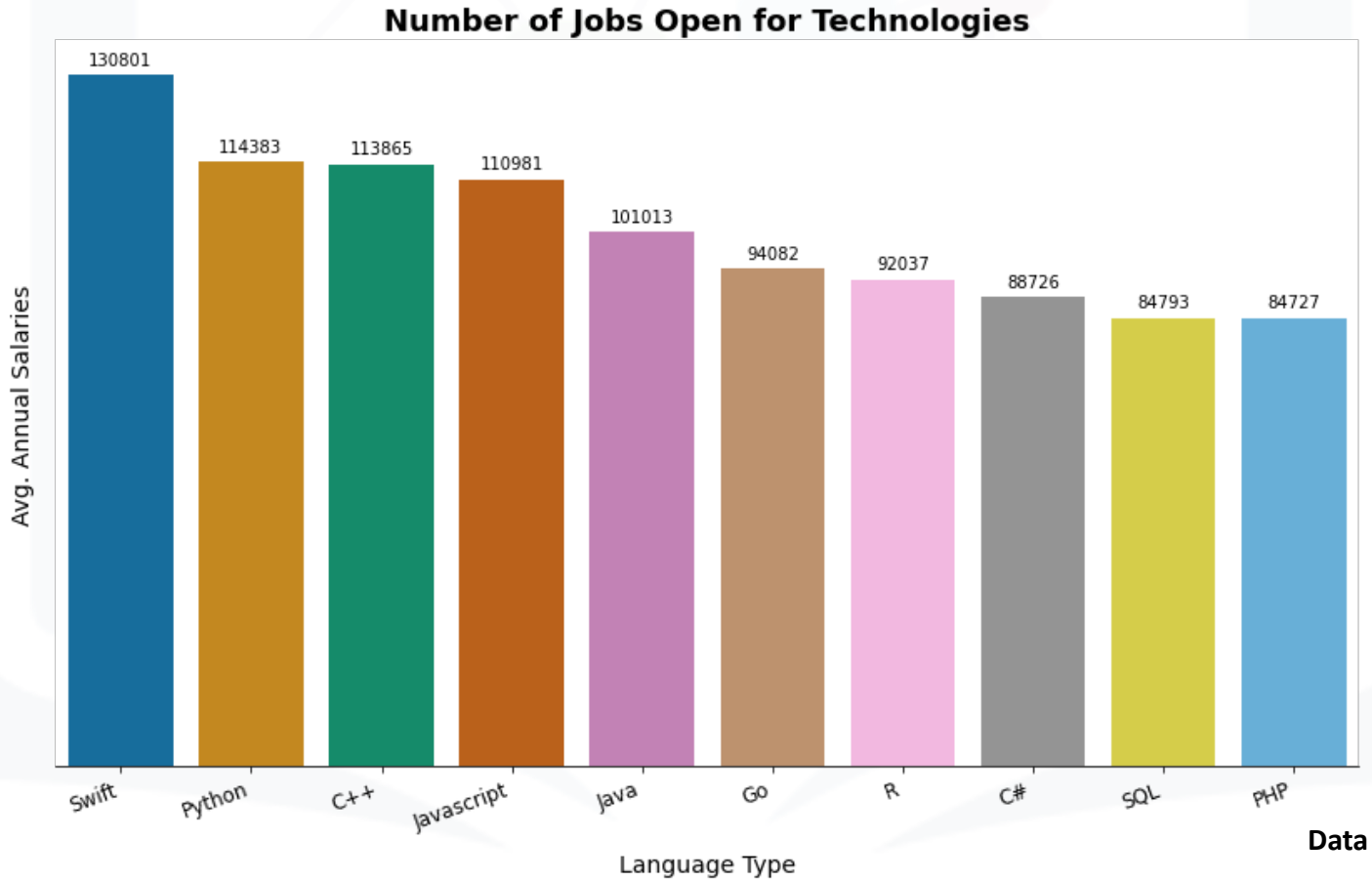
Determine the number of jobs currently open for various technologies.



Data Source: [githubposting.json]

Popular Programming Languages

Which language are developers paid the most according to the output of the web scraping lab?



Data Source: Programming_Languages.html