STACK OVERFLOW DEVELOPER SURVEY

ROSEWAL SIMON ALMEIDA

02nd June 2025

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- **Analysed developer trends** using the Stack Overflow survey data, focusing on compensation, satisfaction, and technology preferences.
- **Used Python (Pandas, Matplotlib) and SQL** to clean, transform, and analyse survey data through filtering, grouping, aggregation, and visualization techniques.
- **Cleaned and transformed data** by mapping age, satisfaction scores, and multi-select columns into usable formats.
- **Visualized patterns** through bubble plots, pie charts, line graphs, and stacked charts to uncover insights by age, experience, and roles.
- **Identified key insights** on how tech preferences, job satisfaction, and pay vary across developer types, age groups, and countries.

INTRODUCTION

- **Objective:** Understand developer trends in tools, compensation, job satisfaction, and preferences.
- **Dataset:** Stack Overflow Developer Survey 2023 loaded into SQLite and analysed using Pandas and SQL with over 65000 respondents.
- **Approach:** Transformed categorical and multi-select data, cleaned missing values, and used aggregation for analysis.
- **Focus areas:** Technology preferences, compensation trends, job satisfaction by age and experience.
- **Importance:** Helps job seekers, educators, and recruiters understand tech market shifts.

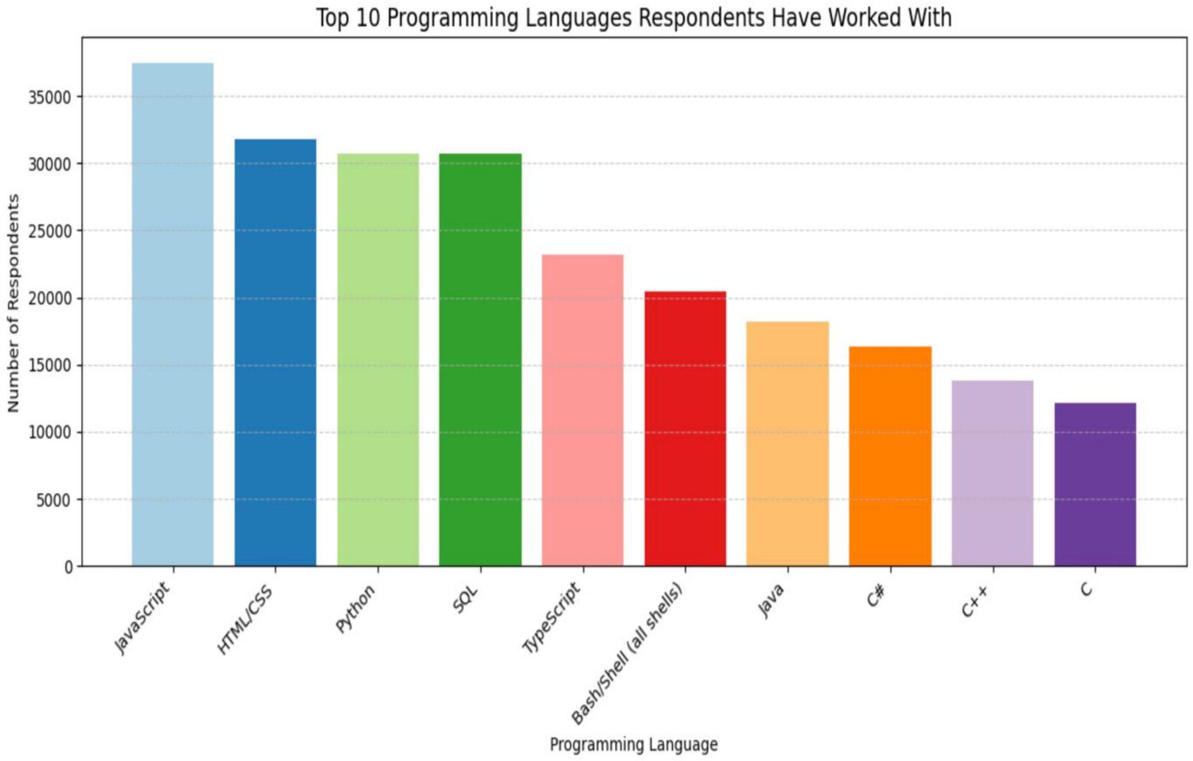
METHODOLOGY

- **Survey Data Collection & Exploration :** Loaded Stack Overflow Developer Survey data and explored structure using Python and SQL.
- **Data Wrangling:** Cleaned missing values, mapped categorical variables and processed multi-select fields using .explode().
- **Exploratory Data Analysis (EDA):** Analysed data distributions, handled outliers, and explored relationships through correlation and group-by operations.
- **SQL Integration:** Executed aggregation queries, filtered subsets using conditions, and binned data to extract targeted insights.
- **Data Visualization:** Used histograms, box plots, line charts, bubble plots, and stacked bar charts to highlight trends, relationships, and comparisons.

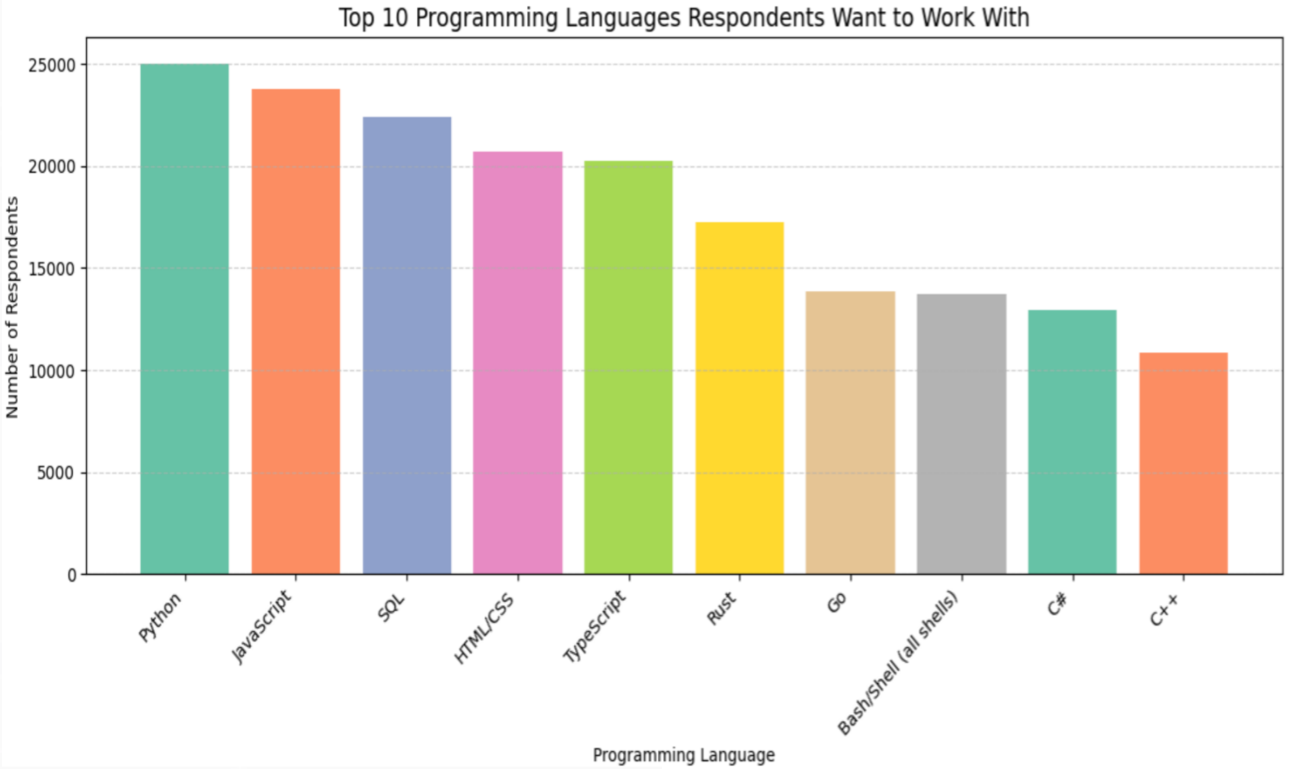
RESULTS

PROGRAMMING LANGUAGE TRENDS

Current Year



Next Year



PROGRAMMING LANGUAGE TRENDS - FINDINGS

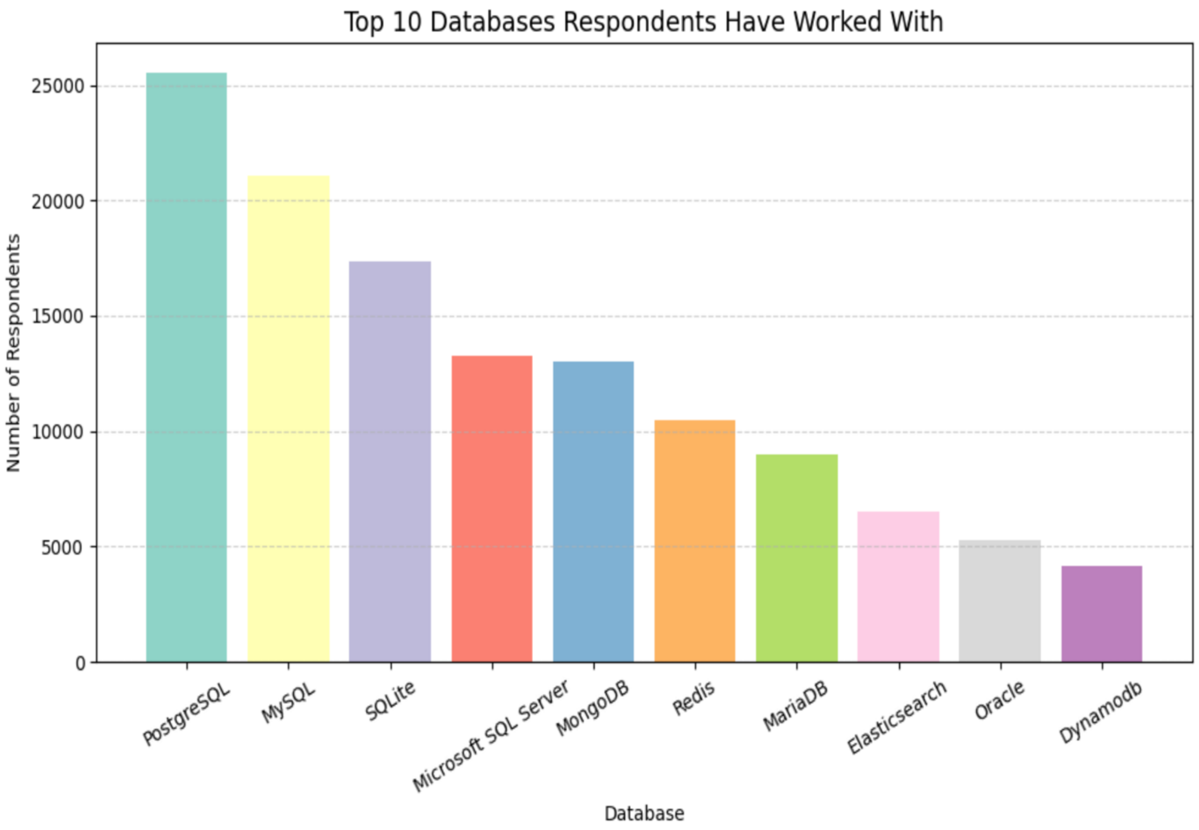
- **JavaScript and Python dominate** both current usage and future preference — indicating continued relevance across development roles.
- **SQL and HTML/CSS** are widely used now, but are slightly less favored in future preference — possibly signalling a shift towards backend and systems programming.
- **Emerging languages like Rust and Go** appear in the "want to work with" list, showing growing developer interest despite lower current usage.
- **TypeScript holds strong** in both charts — indicating its increasing adoption in modern web development.
- **Traditional languages like C, C++, and Java** still have significant current use but are less desired for future work, reflecting a shift towards newer or more flexible languages.

PROGRAMMING LANGUAGE TRENDS - IMPLICATIONS

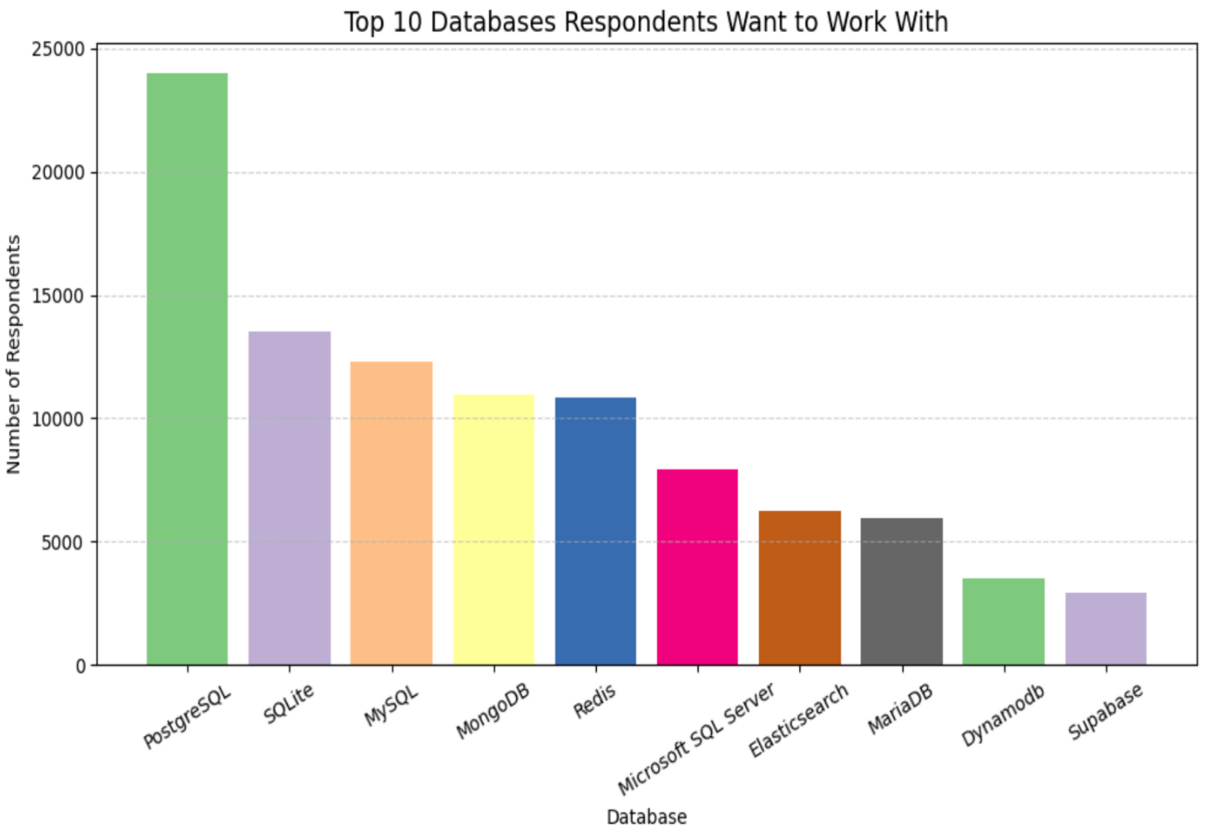
- **Training programs** and upskilling efforts should emphasize Python, TypeScript, and Rust to align with evolving developer interest.
- **Recruiters and tech companies** should monitor shifts toward newer languages (like Rust/Go) when planning future tech stacks.
- Developers may be **migrating away from legacy stacks**, creating opportunities for modernization in existing codebases.
- **Language preference trends** offer insight into evolving roles (e.g., data science, DevOps, web3), helping shape targeted hiring and curriculum planning.

DATABASE TRENDS

Current Year



Next Year



DATABASE TRENDS - FINDINGS

- **PostgreSQL leads** both in current usage and future preference, reinforcing its position as the most widely trusted and adopted open-source RDBMS.
- **MySQL and SQLite** remain highly used, but interest in working with them is declining slightly, possibly due to rising alternatives.
- **MongoDB and Redis** show strong presence in both charts, reflecting ongoing popularity in NoSQL and in-memory storage use cases.
- **DynamoDB and MariaDB** are lower in current usage but maintain a foothold in future interest, indicating consistent niche use.
- **Supabase appears only in future interest**, suggesting it’s a growing trend among developers looking for modern, serverless alternatives.

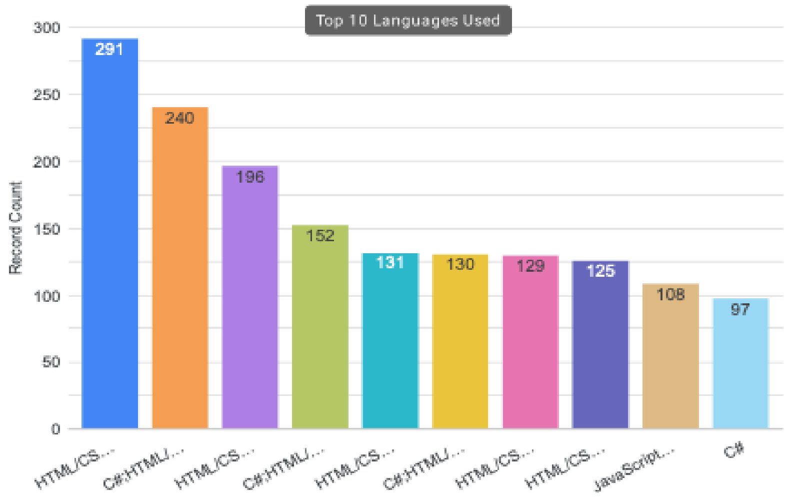
DATABASE TRENDS - IMPLICATIONS

- **PostgreSQL skills are in high demand**, making it a smart choice for training, upskilling, and hiring.
- Organizations relying on **legacy databases like Oracle or Microsoft SQL Server** should consider modern alternatives that align with developer preference to attract top talent.
- The presence of **Redis and MongoDB** in both categories indicates the continued importance of hybrid stacks (SQL + NoSQL).
- Newer platforms like **Supabase** reflect a growing demand for developer-friendly, open-source BaaS (Backend-as-a-Service) solutions — companies exploring startup-style development can leverage this shift.
- Companies should **adapt database strategies** not just based on stability and security, but also on talent availability and community momentum.

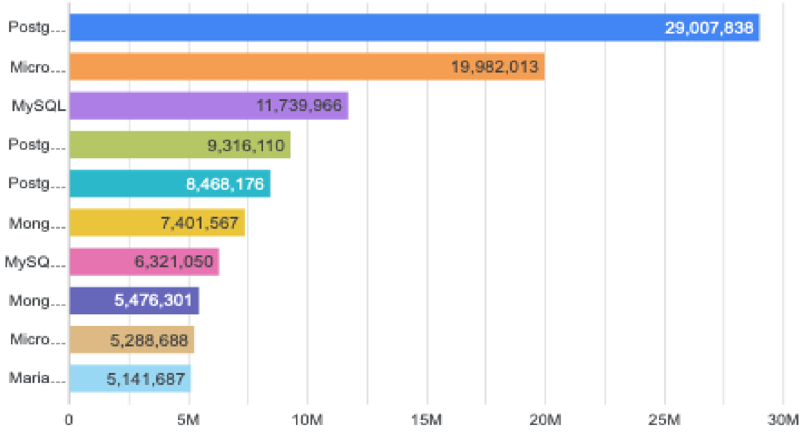
DASHBOARD

CURRENT TECHNOLOGY TREND

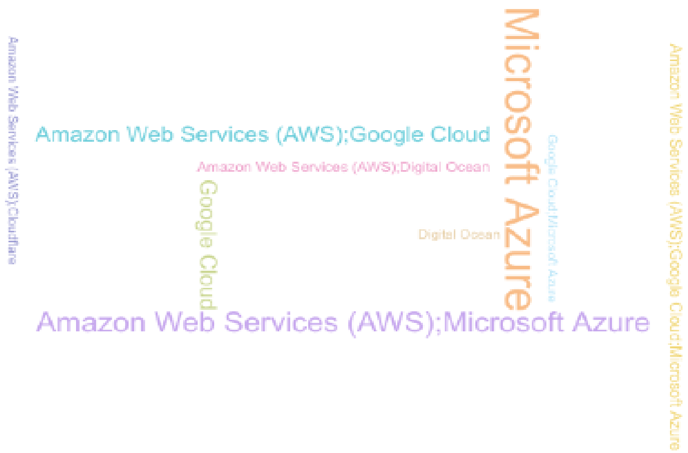
Top 10 Languages Used



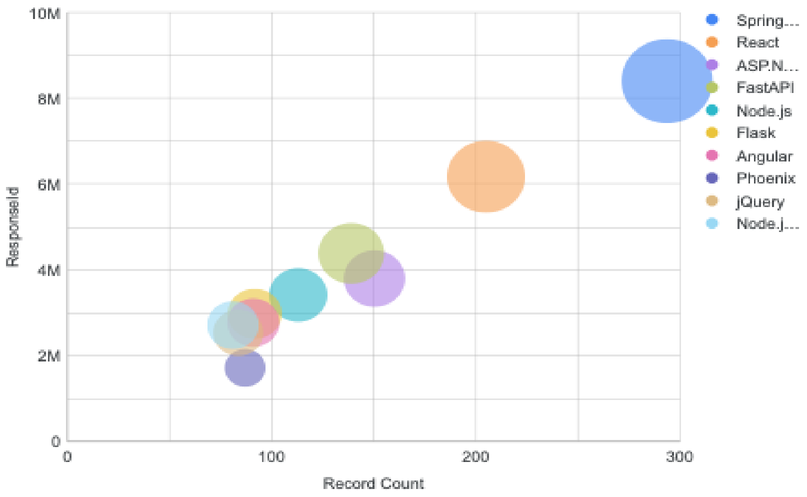
Top 10 Databases Used



Top 10 Current Platforms

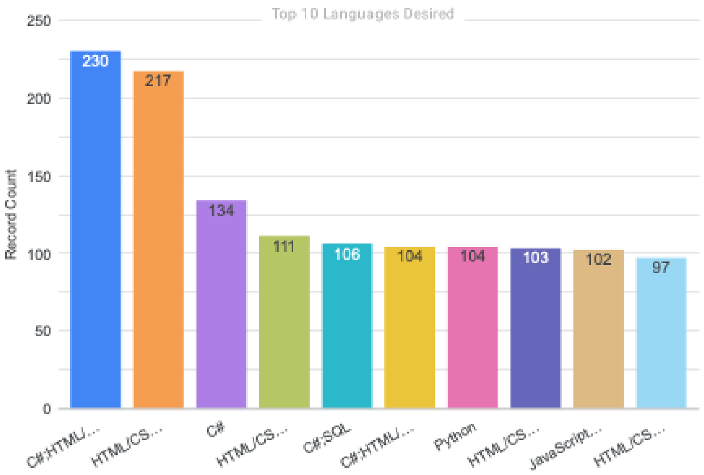


Top 10 Web Frameworks Used

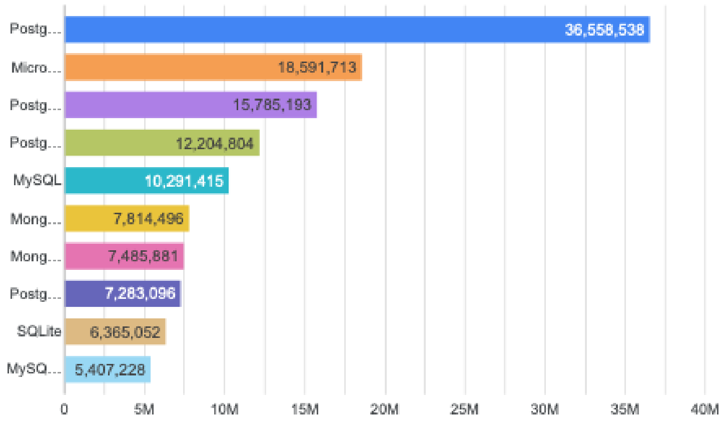


FUTURE TECHNOLOGY TREND

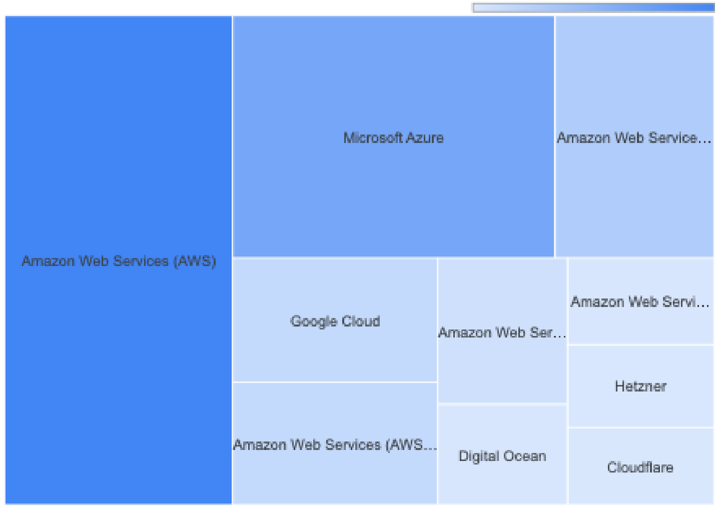
Top 10 Languages Desired



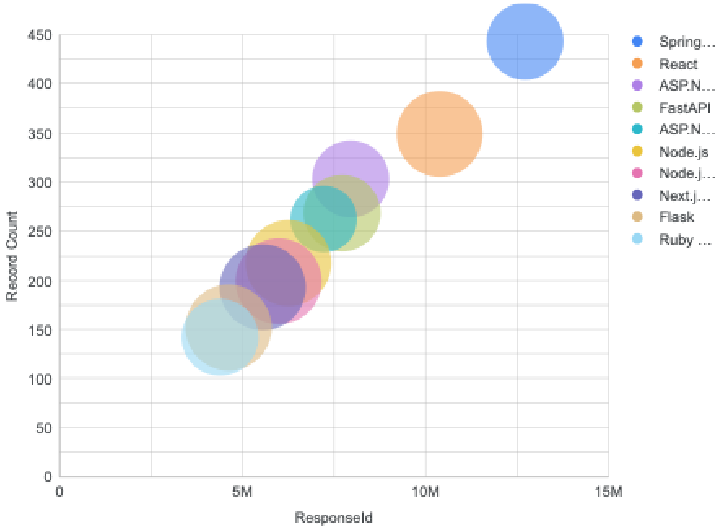
Top 10 Databases Desired



Top 10 Desired Platforms

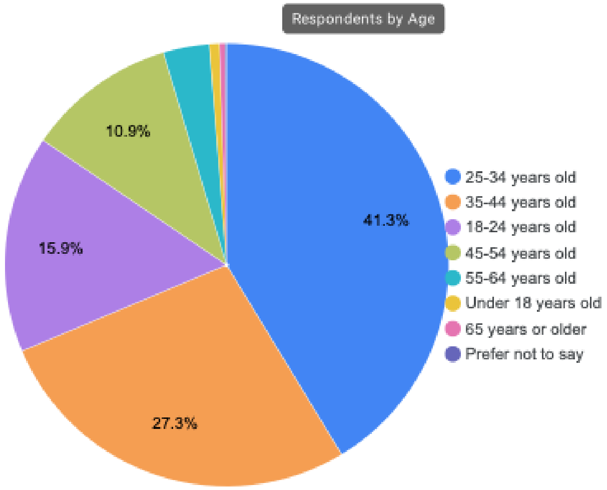


Top 10 Desired Web Frameworks

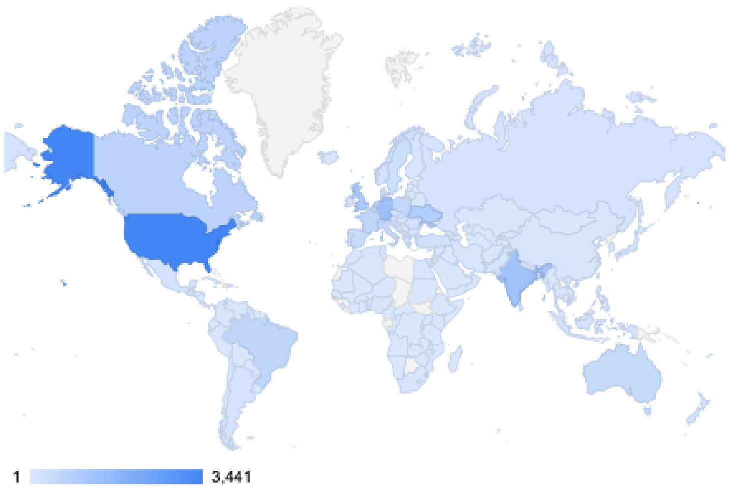


DEMOGRAPHICS

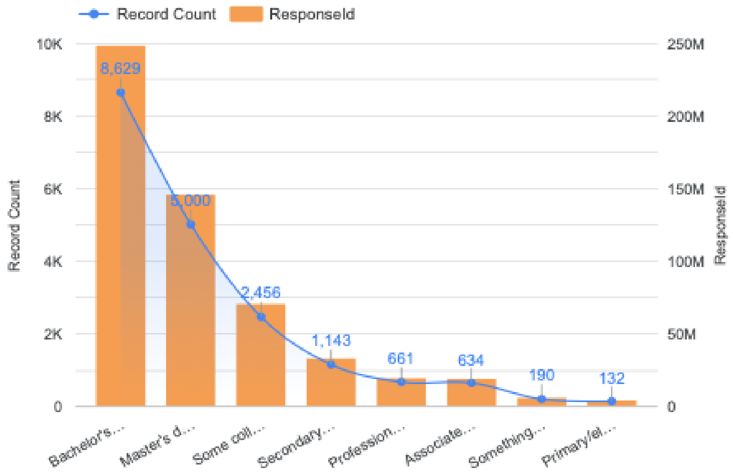
Respondents by Age



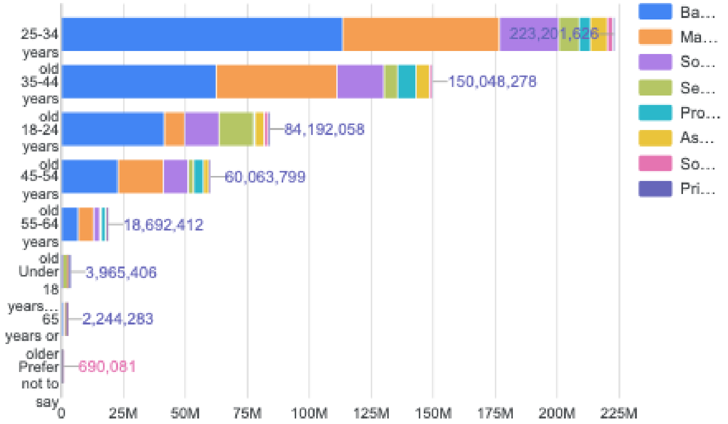
Respondent Count by Country



Respondent Distribution by Education Level



Respondent Count by Age, Classified by Education Level



Skills Network

IBM

DISCUSSION

OVERALL FINDINGS

1. Respondent Demographics

- The **majority of developers are aged 25–34** (41.3%), followed by 35–44 (27.3%) and 18–24 (15.9%) .
- Most respondents hold a **Bachelor’s degree (8,629)** or a **Master’s degree (5,000)**, reflecting a highly educated sample population .

2. Current Technology Usage

- **Languages**: HTML/CSS, C#, and JavaScript are the most commonly used languages .
- **Databases**: PostgreSQL leads in usage, followed by Microsoft SQL Server and MySQL.
- **Web Frameworks**: React, Spring Boot, and ASP.NET are top current choices.
- **Platforms**: Major platforms include AWS, Azure, and various cloud services.

3. Future Technology Preference

- **Languages**: HTML/CSS and C# still dominate, but there’s a **notable rise in Python** and JavaScript as desired skills .
- **Databases**: PostgreSQL sees even **higher interest for future use**, further solidifying its popularity.
- **Frameworks**: React, FastAPI, and Next.js are **rising stars** in future preference.
- **Platforms**: AWS and Azure remain in high demand, with **Google Cloud and Digital Ocean** gaining momentum.

OVERALL IMPLICATIONS

1. For Educators & Curriculum Designers:

- Emphasize **PostgreSQL, Python, React**, and **cloud platforms** like AWS/Azure in course offerings.
- Develop **transition pathways from commonly used tools** (e.g., C# or SQL Server) to more in-demand, future-oriented stacks (e.g., Python + PostgreSQL + React).

2. For Employers & Recruiters:

- Align **hiring and training programs** with technologies developers want to use—especially PostgreSQL, Python, and modern frameworks.
- Offering projects with **cloud-native architecture** (AWS, Next.js, FastAPI) could attract top developer talent.

3. For Developers:

- Gaining hands-on experience with **PostgreSQL, Python, React, and cloud tools** can provide a competitive edge.
- While traditional stacks like C# and SQL Server remain strong, **investing in newer frameworks and databases** may offer better long-term career growth.

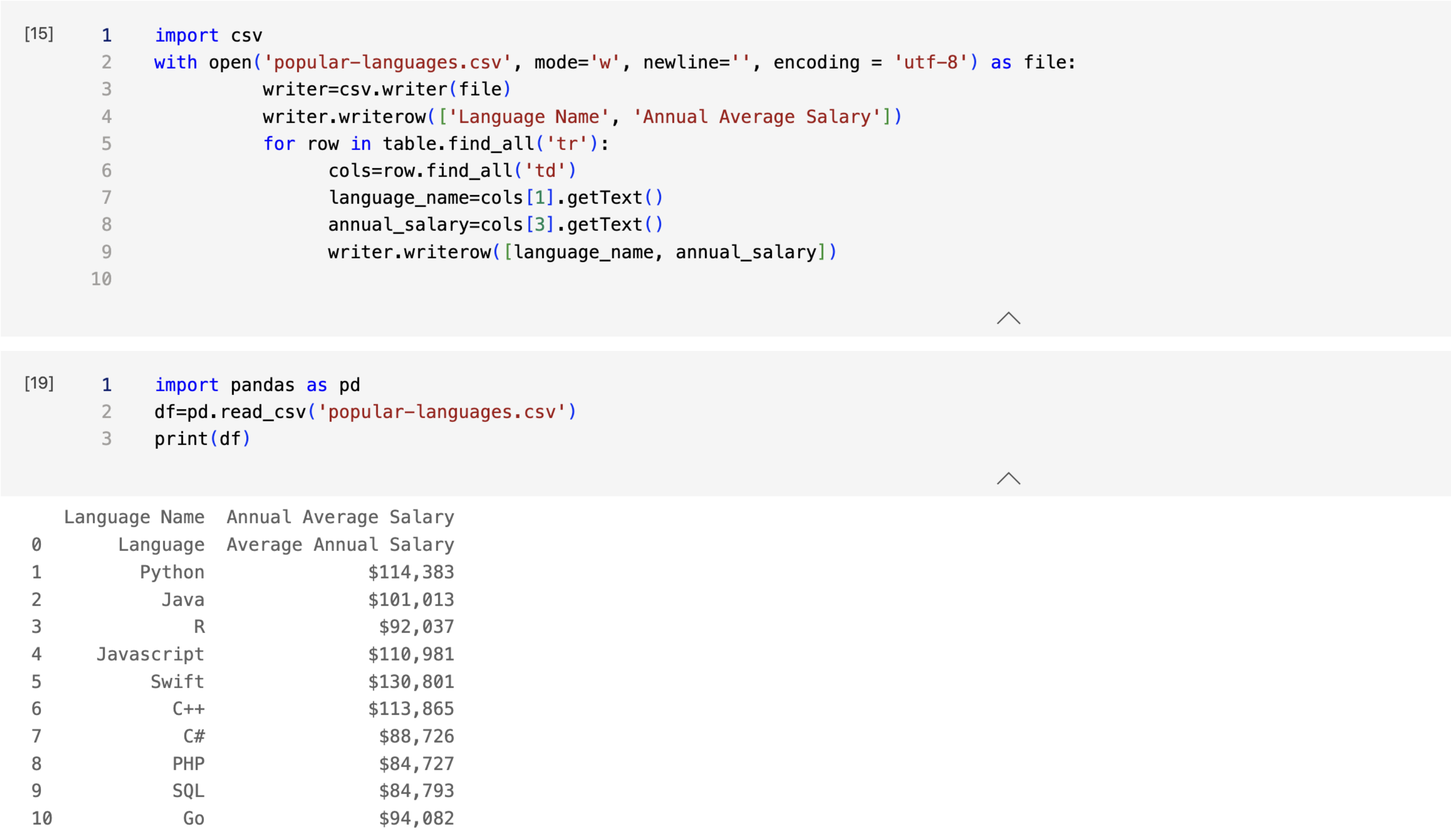
CONCLUSION

CONCLUSION

- The analysis identified key trends in both **current usage** and **future interest** across languages, databases, frameworks, and platforms.
- **PostgreSQL, Python, and React** emerged as consistent leaders in both present use and developer preference for future projects.
- There’s a growing shift toward **cloud-based, open-source, and developer-friendly technologies** (e.g., FastAPI, Next.js, Supabase).
- **Demographic insights** showed most developers are highly educated and between the ages of 25–44, a group likely to influence future tech adoption.
- These insights help guide **educators, employers, and developers** in aligning learning paths, hiring strategies, and tech stack decisions with emerging trends.

APPENDIX

WEB SCRAPING JOBS IN DEMAND



[15]

```
import csv
with open('popular-languages.csv', mode='w', newline='', encoding = 'utf-8') as file:
          writer=csv.writer(file)
          writer.writerow(['Language Name', 'Annual Average Salary'])
          for row in table.find_all('tr'):
                cols=row.find_all('td')
                language_name=cols[1].getText()
                annual_salary=cols[3].getText()
                writer.writerow([language_name, annual_salary])
```

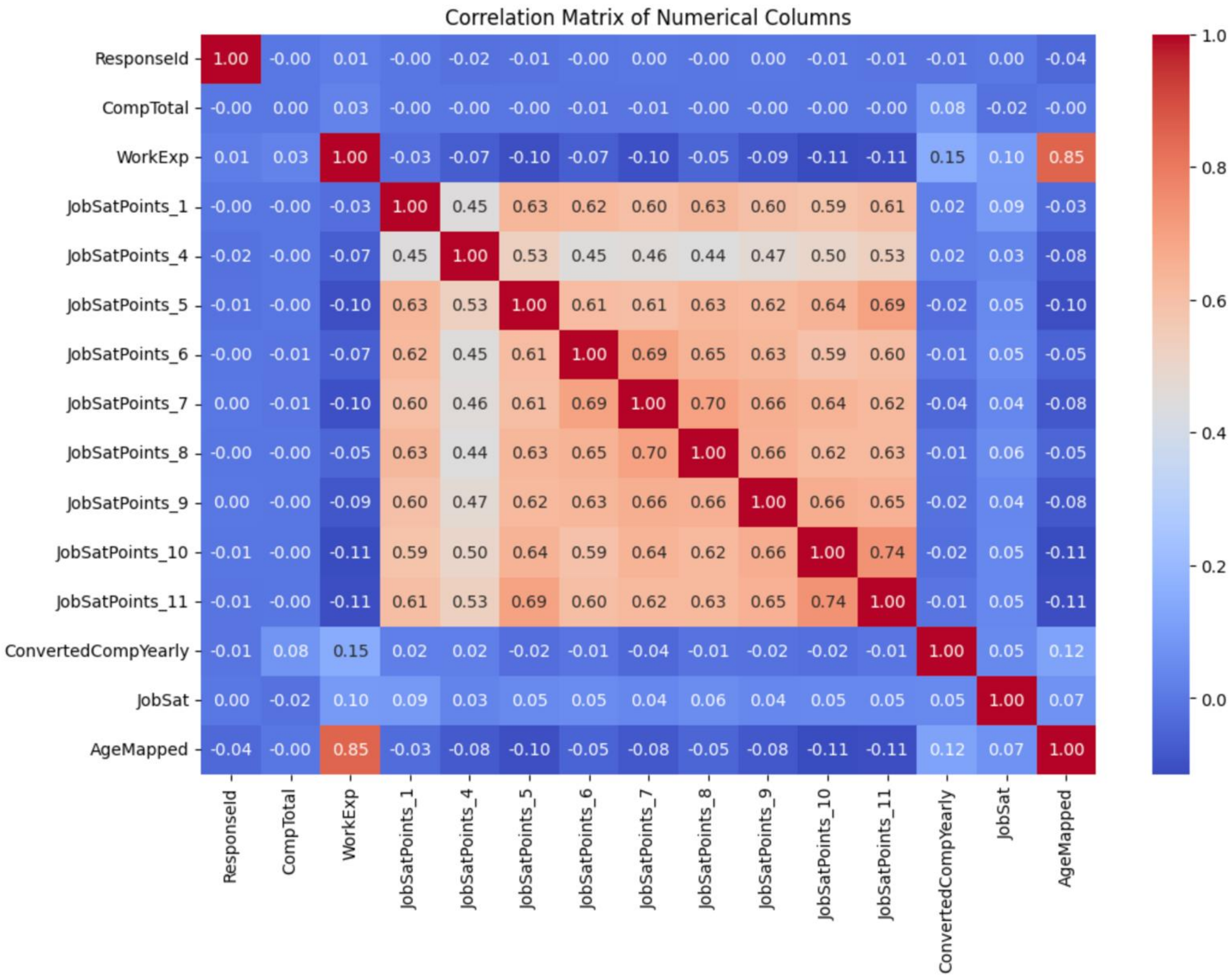
[19]

```
import pandas as pd
df=pd.read_csv('popular-languages.csv')
print(df)
```

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

Skills Network

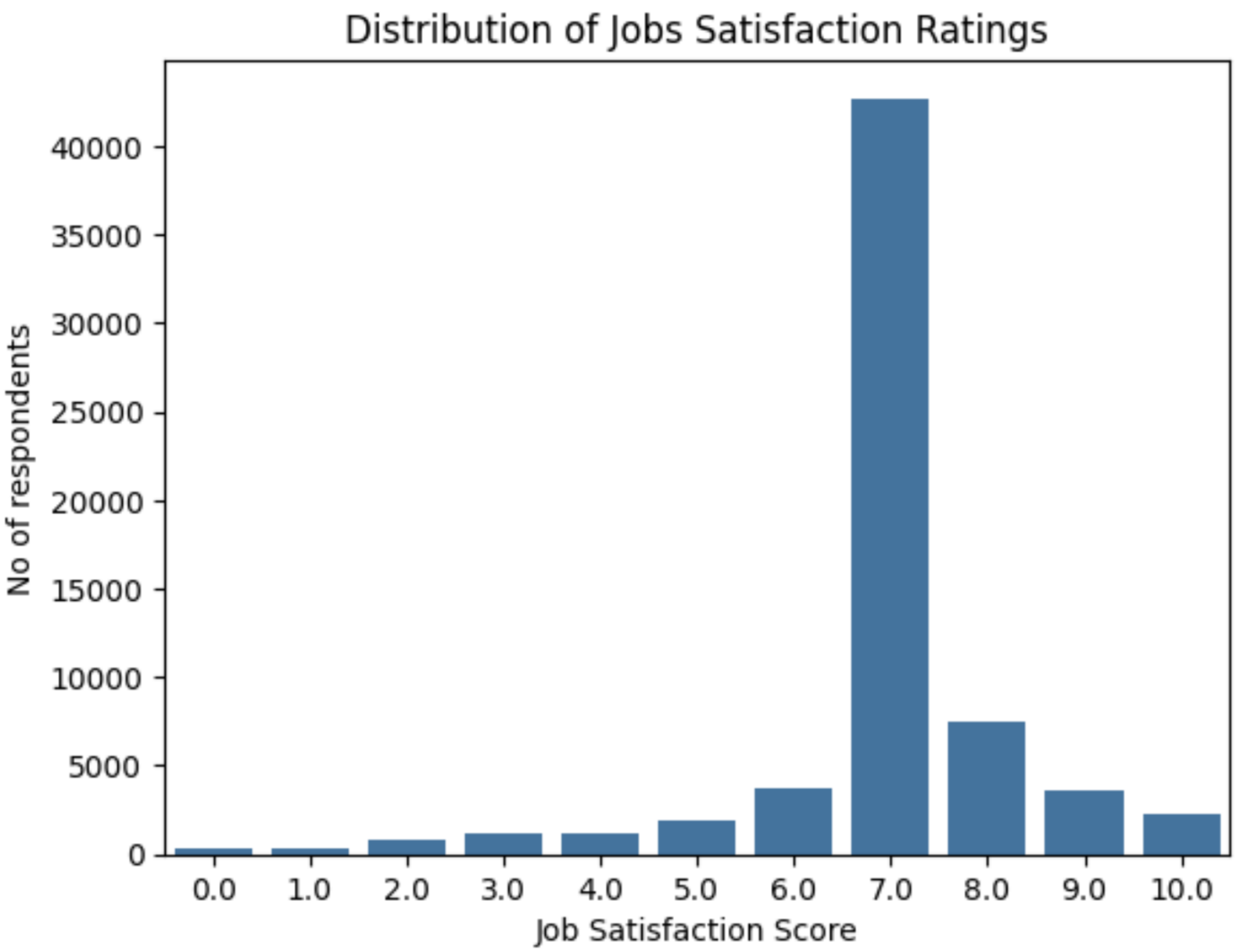
CORRELATON BETWEEN NUMERIC COLUMNS



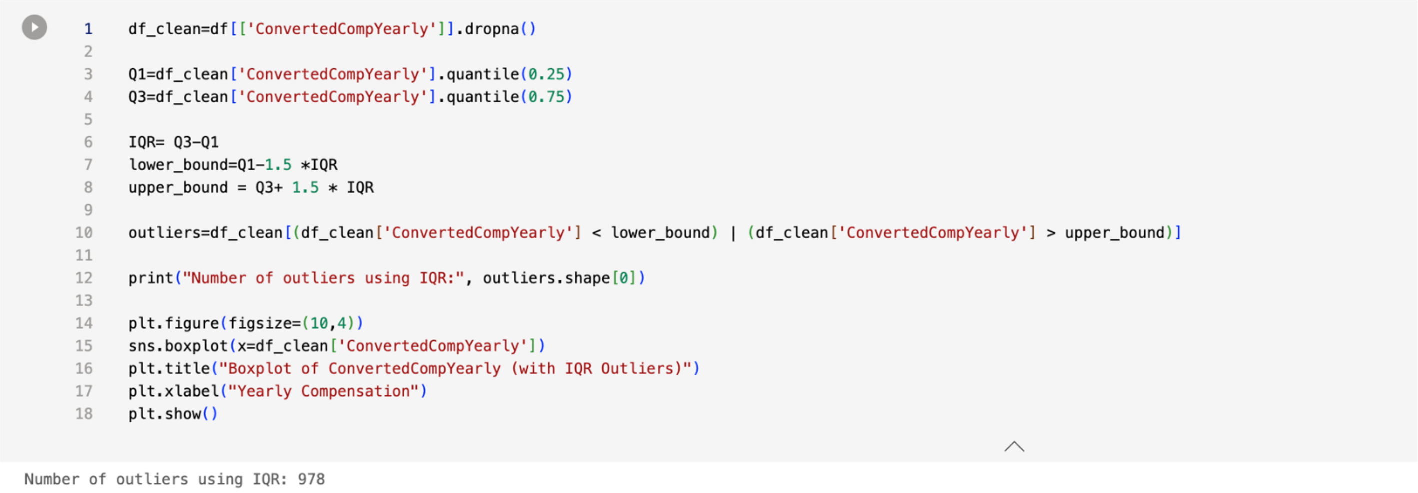
Correlation Matrix of Numerical Columns

	ResponseId	CompTotal	WorkExp	JobSatPoints_1	JobSatPoints_4	JobSatPoints_5	JobSatPoints_6	JobSatPoints_7	JobSatPoints_8	JobSatPoints_9	JobSatPoints_10	JobSatPoints_11	ConvertedCompYearly	JobSat	AgeMapped
ResponseId	1.00	-0.00	0.01	-0.00	-0.02	-0.01	-0.00	0.00	-0.00	0.00	-0.01	-0.01	-0.01	0.00	-0.04
CompTotal	-0.00	0.00	0.03	-0.00	-0.00	-0.00	-0.01	-0.01	-0.00	-0.00	-0.00	-0.00	0.08	-0.02	-0.00
WorkExp	0.01	0.03	1.00	-0.03	-0.07	-0.10	-0.07	-0.10	-0.05	-0.09	-0.11	-0.11	0.15	0.10	0.85
JobSatPoints_1	-0.00	-0.00	-0.03	1.00	0.45	0.63	0.62	0.60	0.63	0.60	0.59	0.61	0.02	0.09	-0.03
JobSatPoints_4	-0.02	-0.00	-0.07	0.45	1.00	0.53	0.45	0.46	0.44	0.47	0.50	0.53	0.02	0.03	-0.08
JobSatPoints_5	-0.01	-0.00	-0.10	0.63	0.53	1.00	0.61	0.61	0.63	0.62	0.64	0.69	-0.02	0.05	-0.10
JobSatPoints_6	-0.00	-0.01	-0.07	0.62	0.45	0.61	1.00	0.69	0.65	0.63	0.59	0.60	-0.01	0.05	-0.05
JobSatPoints_7	0.00	-0.01	-0.10	0.60	0.46	0.61	0.69	1.00	0.70	0.66	0.64	0.62	-0.04	0.04	-0.08
JobSatPoints_8	-0.00	-0.00	-0.05	0.63	0.44	0.63	0.65	0.70	1.00	0.66	0.62	0.63	-0.01	0.06	-0.05
JobSatPoints_9	0.00	-0.00	-0.09	0.60	0.47	0.62	0.63	0.66	0.66	1.00	0.66	0.65	-0.02	0.04	-0.08
JobSatPoints_10	-0.01	-0.00	-0.11	0.59	0.50	0.64	0.59	0.64	0.62	0.66	1.00	0.74	-0.02	0.05	-0.11
JobSatPoints_11	-0.01	-0.00	-0.11	0.61	0.53	0.69	0.60	0.62	0.63	0.65	0.74	1.00	-0.01	0.05	-0.11
ConvertedCompYearly	-0.01	0.08	0.15	0.02	0.02	-0.02	-0.01	-0.04	-0.01	-0.02	-0.02	-0.01	1.00	0.05	0.12
JobSat	0.00	-0.02	0.10	0.09	0.03	0.05	0.05	0.04	0.06	0.04	0.05	0.05	0.05	1.00	0.07
AgeMapped	-0.04	-0.00	0.85	-0.03	-0.08	-0.10	-0.05	-0.08	-0.05	-0.08	-0.11	-0.11	0.12	0.07	1.00

JOBS SATISFACTION RATINGS

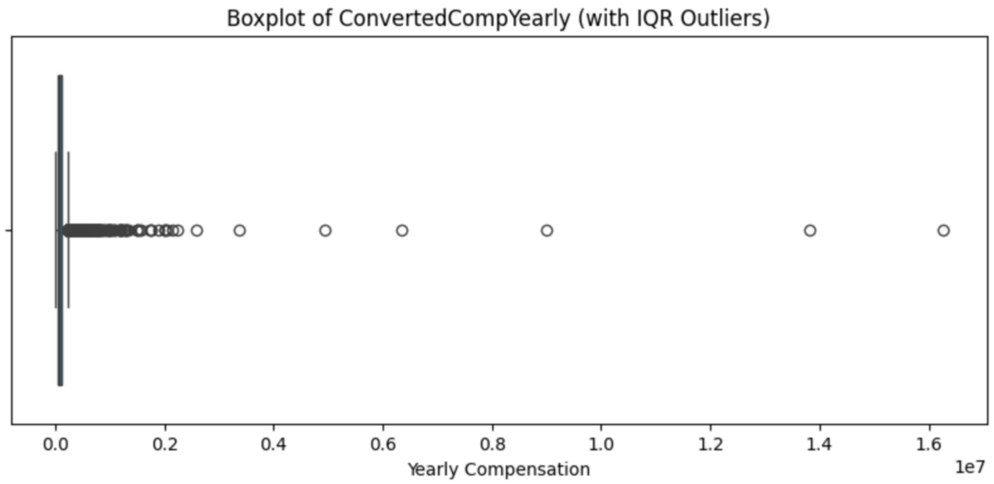


CHECKING OUTLIERS FOR COMPENSATION

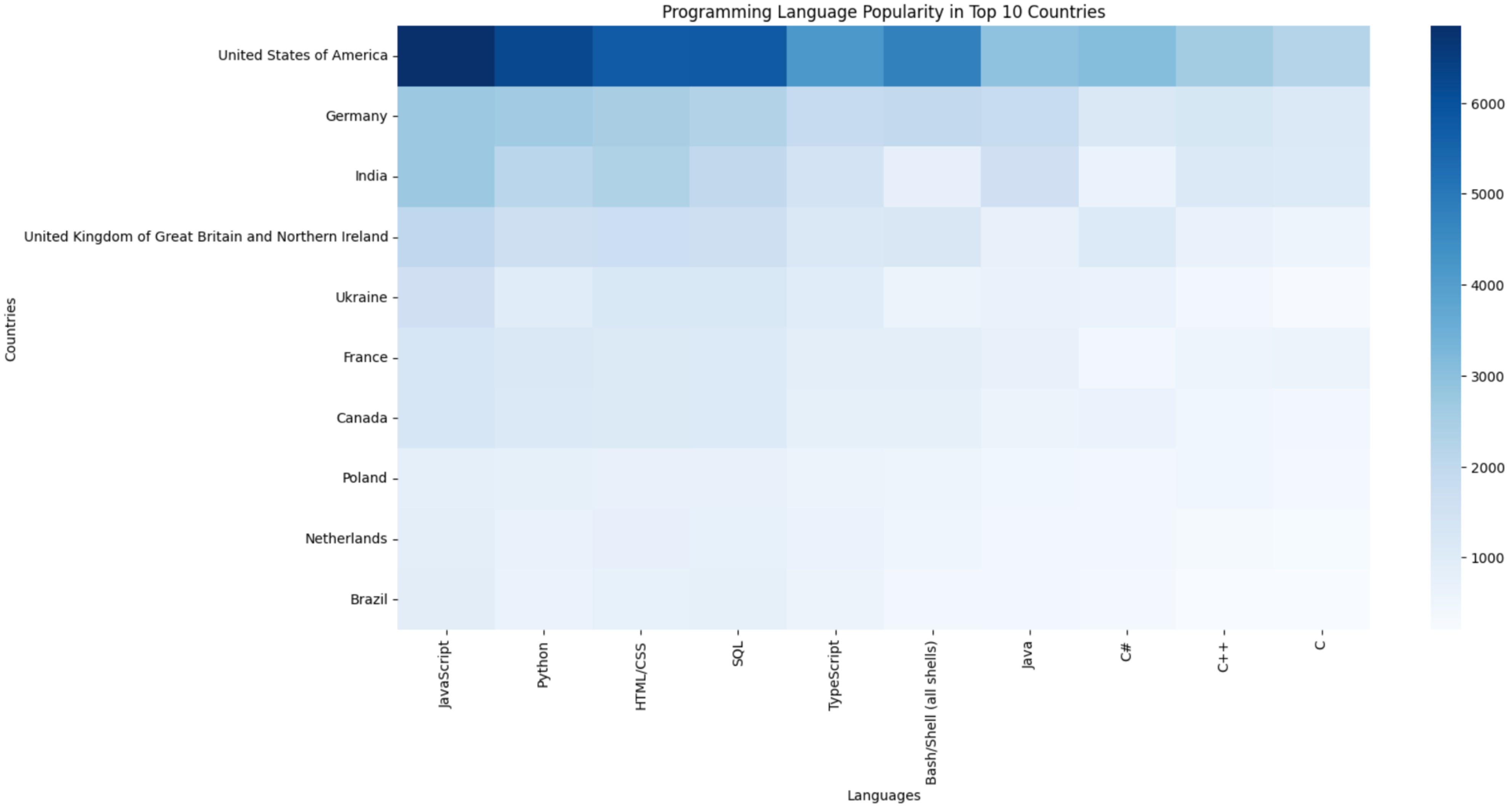


```python
df_clean=df[['ConvertedCompYearly']].dropna()

Q1=df_clean['ConvertedCompYearly'].quantile(0.25)
Q3=df_clean['ConvertedCompYearly'].quantile(0.75)

IQR= Q3-Q1
lower_bound=Q1-1.5 *IQR
upper_bound = Q3+ 1.5 * IQR

outliers=df_clean[(df_clean['ConvertedCompYearly'] < lower_bound) | (df_clean['ConvertedCompYearly'] > upper_bound)]

print("Number of outliers using IQR:", outliers.shape[0])

plt.figure(figsize=(10,4))
sns.boxplot(x=df_clean['ConvertedCompYearly'])
plt.title("Boxplot of ConvertedCompYearly (with IQR Outliers)")
plt.xlabel("Yearly Compensation")
plt.show()
```

```
Number of outliers using IQR: 978
```



POPULAR PROGRAMMING LANGUAGES IN TOP 10 COUNTRIES


Programming Language Popularity in Top 10 Countries
Countries
United States of America
Germany
India
United Kingdom of Great Britain and Northern Ireland
Ukraine
France
Canada
Poland
Netherlands
Brazil
JavaScript
Python
HTML/CSS
SQL
TypeScript
Bash/Shell (all shells)
Java
C#
C++
C
Languages