Capstone Project rev2.0

Capstone Analysis

Using Stack Overflow Developer Survey 2024

Analysis Conducted by	: Danny Tang
Data	: Nov 14, 2025

Table of Contents

Executive Summary

- ❖ This report analyzes developer technology trends using the 2024 Stack Overflow Developer Survey.
- ❖ Dashboards were built in Google Looker Studio to visualize current usage, future preferences, and demographic patterns.
- ❖ Key findings include the prevalence of modern web frameworks and language, strong cloud platform consolidation, and a respondents skew younger and hold formal education credentials, especially Bachelor’s and Master’s degree.
- ❖ Insights are intended to guide strategic decisions in hiring, education, and tech adoption.

Introduction

This report presents a data-driven analysis of developer technology trends using the 2024 Stack Overflow Developer Survey. As the analyst leading this initiative, I designed and built interactive dashboards in Google Looker Studio to visualize current usage and future preferences across programming languages, cloud platforms, databases, and web frameworks.

The dataset used is a curated subset of the original survey, filtered for completeness, cleaned for outliers, and dashboard responsiveness. Visualizations were organized into three thematic panels: Current Technology Usage, Future Technology Trends, and Demographics. Each panel follows a structured 2×2 layout, featuring stacked bar charts, bubble plots, word clouds, treemaps, and map-based visuals.

Advanced visualizations using Vega-Lite and calculated fields enhanced interpretability. Filters were applied to exclude null values and ensure data accuracy. This report offers actionable insights for stakeholders interested in developer behavior, technology adoption, and emerging trends across industries and geographies.

Methodology

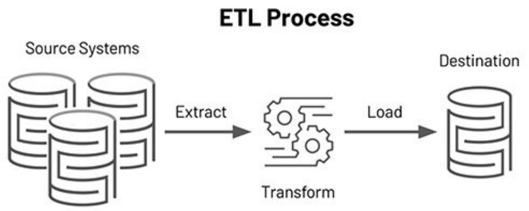
- Tools used:
  - Google CoLab [python coding for Data E.T.L]
  - Google Looker Studio [Dashboard]
  - Google Slides [Presentation]
  - Microsoft Excel & VBA [Data Analysis]

- Data wrangling:
  - Extract data from provided URL
  - Transform data [dropna, drop_duplicates, IQR, create new features]
  - Load cleaned data to dataframe [Output multiple csv files]

- Visualizations:
  - Built-in charts
  - Vega/Vega-Lite for word cloud
  - Sunburst

- Screenshots of setup steps included in the appendix

Data Process Pipeline (ETL)



1. Data Loading
2. Data Cleaning
3. Feature Engineering

- Loading dataset from URL using df = pd.read_csv(url)

- Perform initial verification
  - Check data size and structure
  - Inspect column names and data types
  - Confirm completeness and consistency

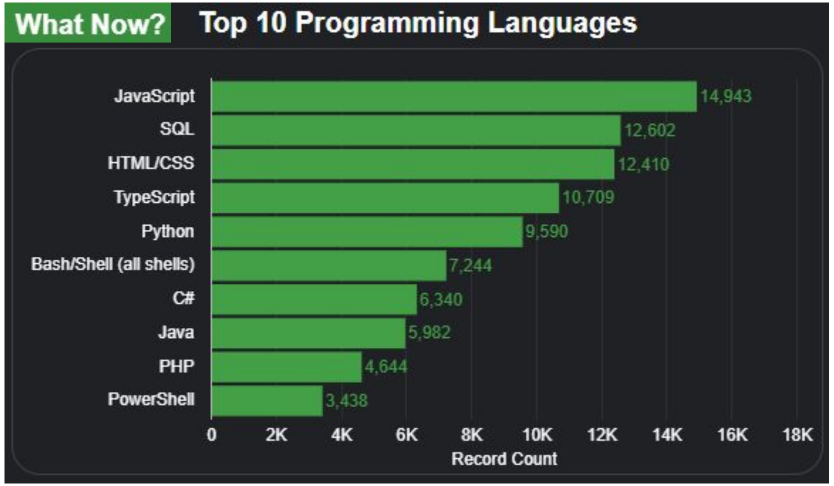
- Apply standard cleaning steps
  - Remove missing values: dropna()
  - Eliminate duplicates: drop_duplicates()
  - Filter outliers using IQR method

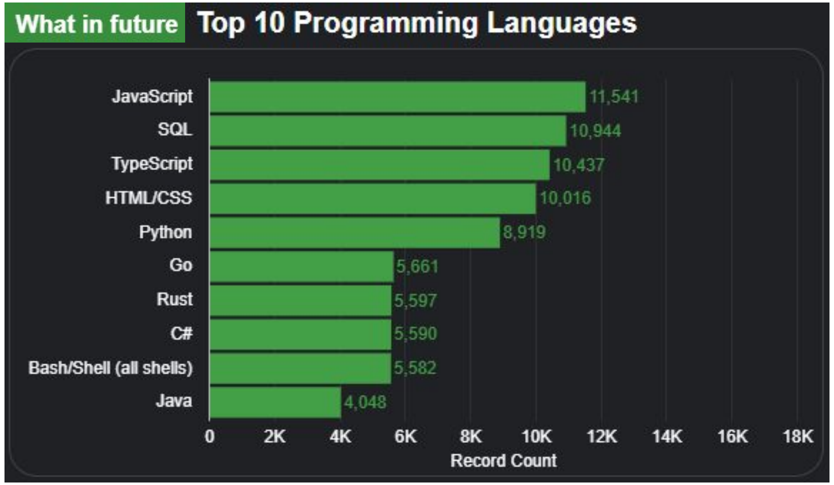
- Transform categorical age ranges into numeric values by calculating midpoints (eg., ‘10-20 years’ → 15)

- Simplified education levels for clarity:
  - 'Bachelor’s degree (B.A., B.S., B.Eng., etc.)' → 'Bachelor'
  - 'Some college/university study without earning a degree' → 'Non-deg college'

Final cleaned dataset stored in: df_clean

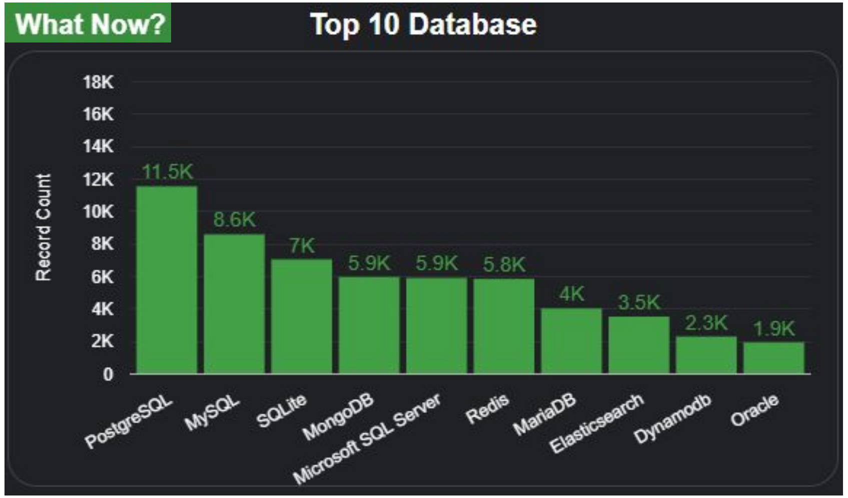
Observations – Programming Languages Trends

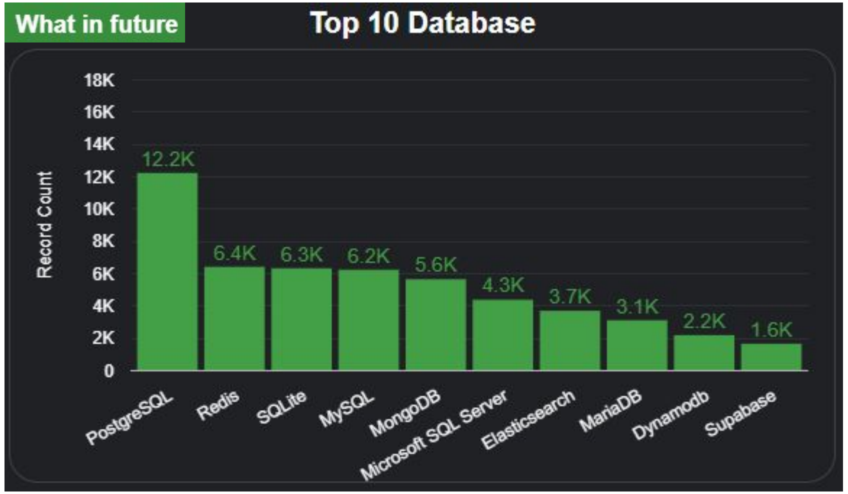




- Currently, JavaScript is the most famous programming language. It has ~15k respondents from replied survey. There are ~2.5k more than SQL & HTML/CSS.
- However, in next year, the gap between them are became minor.
- Top 5 languages remain consistent, though TypeScript overtakes HTML/CSS in future preference.
- Go & Rust were jumped into Top 10 while PHP & PowerShell were out.

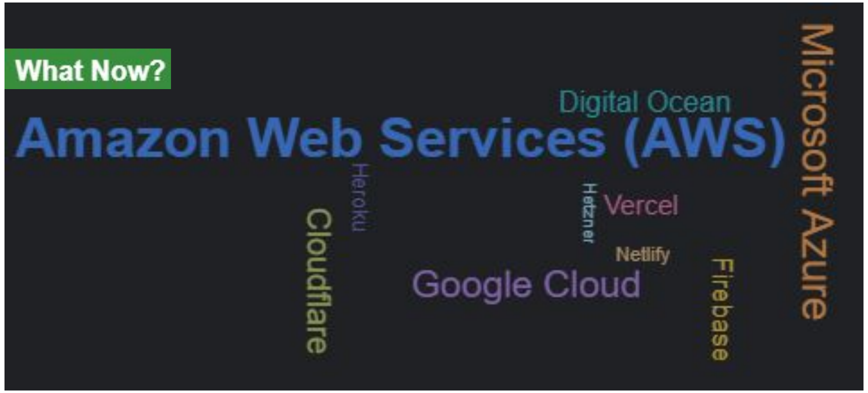
Observations – Database Trends



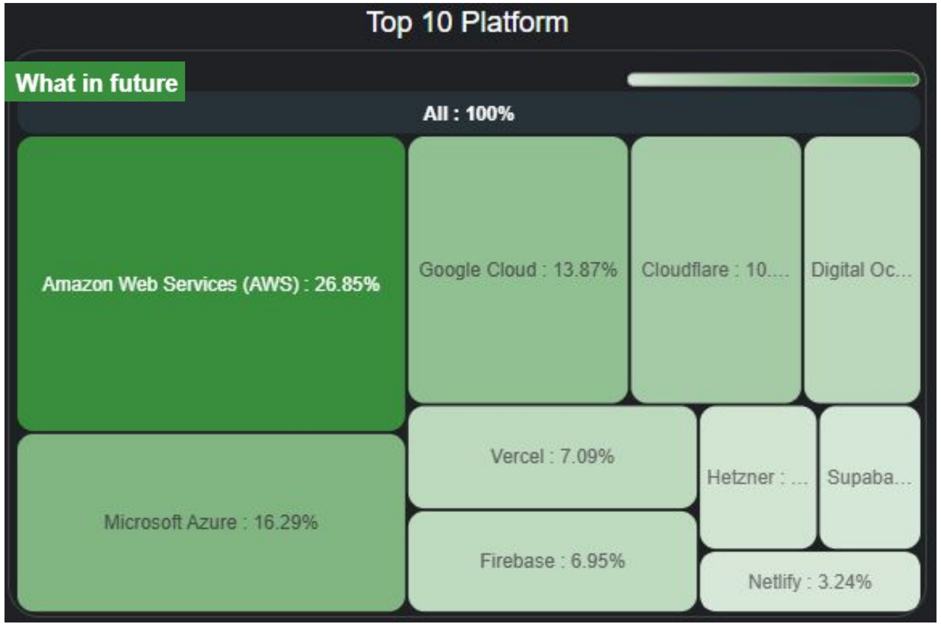


- Data PostgreSQL is still outstanding from today to next year. It achieved 11.5k to 12.2k.
- Rank 2 to Rank 5 will become comparable. Their differences are within 1k.
- In general, top 9 database can be kept but ranks switched.
- Oracle drops out of the top 10, replaced by Supabase.
- In next year, 6 of 10 are SQL-based while today is 7.

Observations – Platform Trends

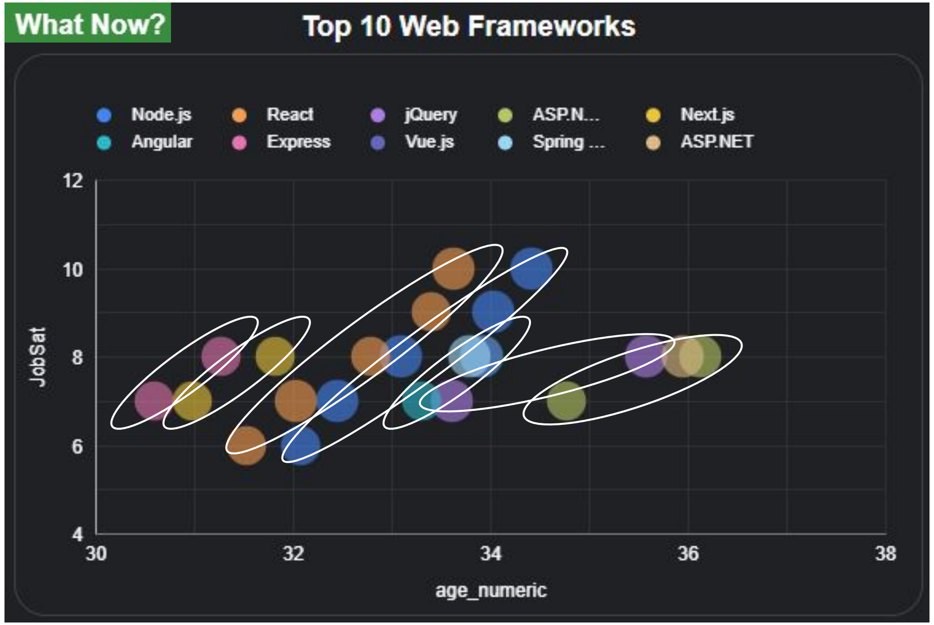


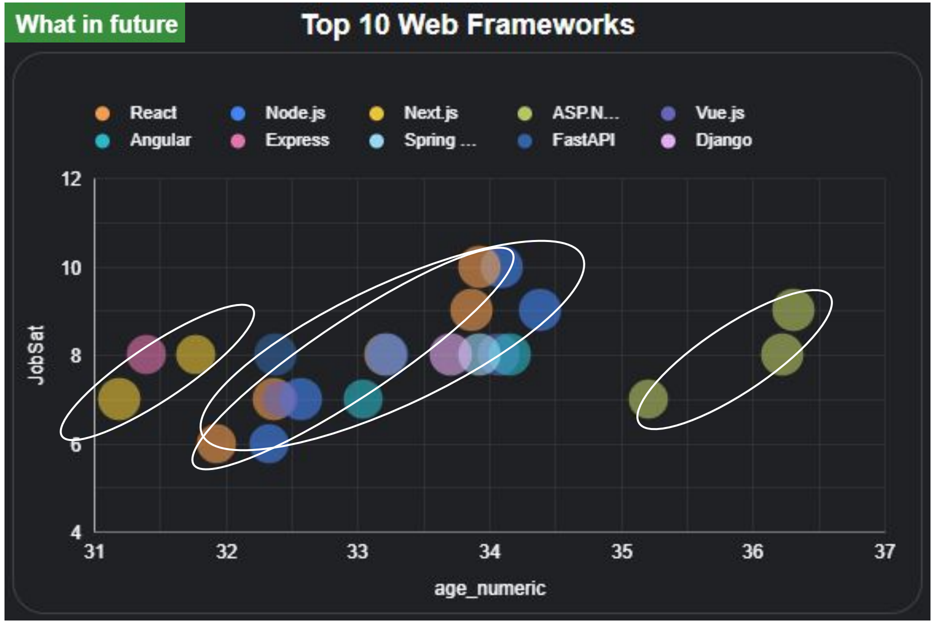
- Amazon Web Services (AWS) is the most prominent platform.
- Microsoft Azure and Google Cloud follow closely in usage frequency.
- Other platforms like Cloudflare, Firebase, Digital Ocean and Vercel appear with moderate adoption with niche appeal.



- Amazon Web Services (AWS) holds largest projected market share (26.85%). Microsoft Azure and Google Cloud follow with significant adoption.
- Smaller platforms like Digital Ocean and Cloudflare show niche but growing interest.
- The treemaps highlights a clear concentration of cloud usage among the top three providers

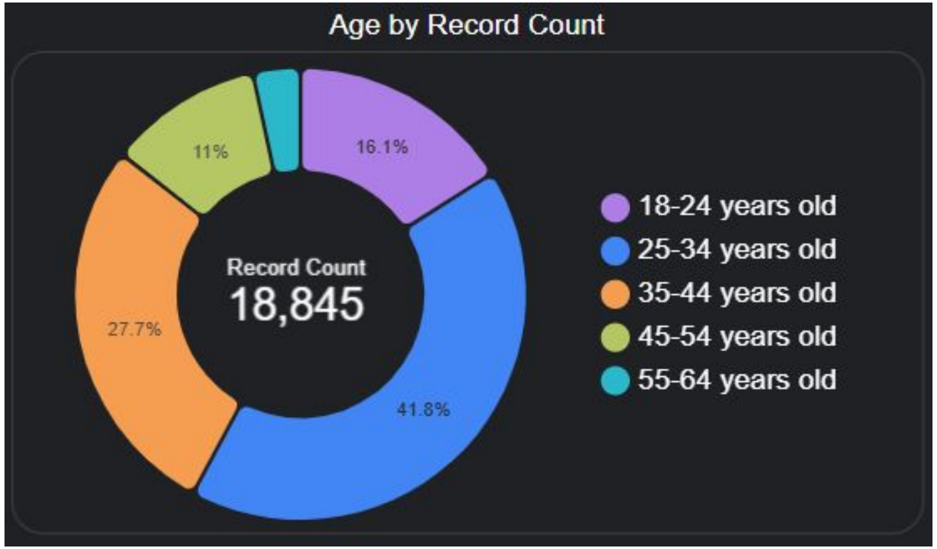
Observations – Web Frameworks Trends

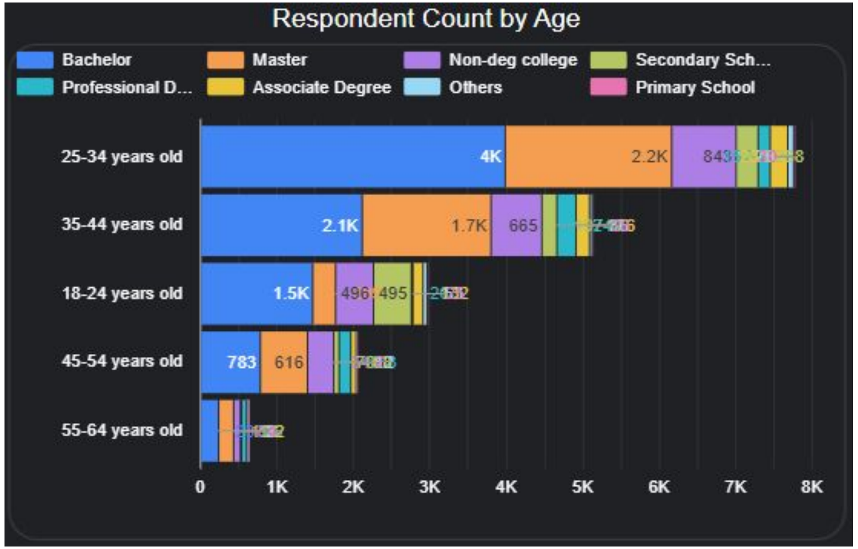




- For the correlation plot of Today Top 10 Web Frameworks, the individual trend is observed by Frameworks.
- However, the trends from React, Node.js and Spring Boot will be fixed together.

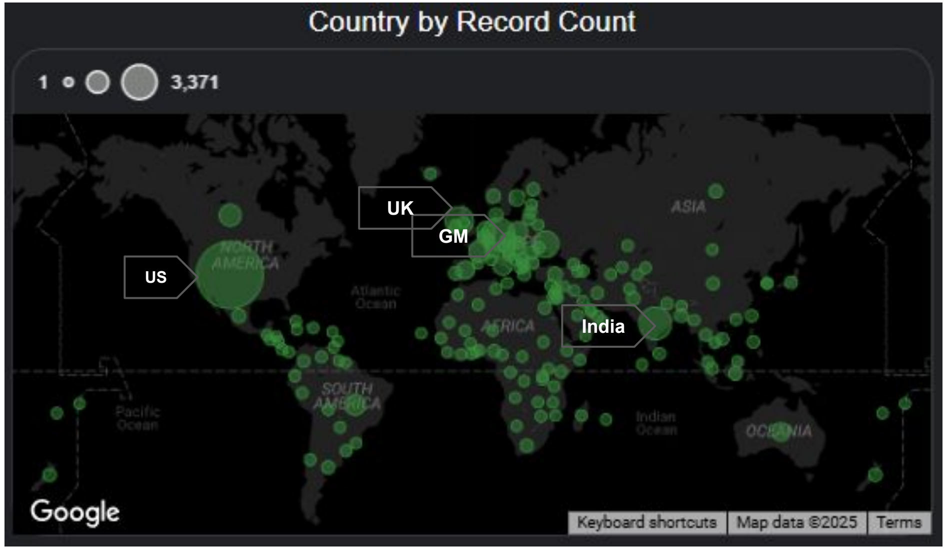
Observations – Respondents Distribution

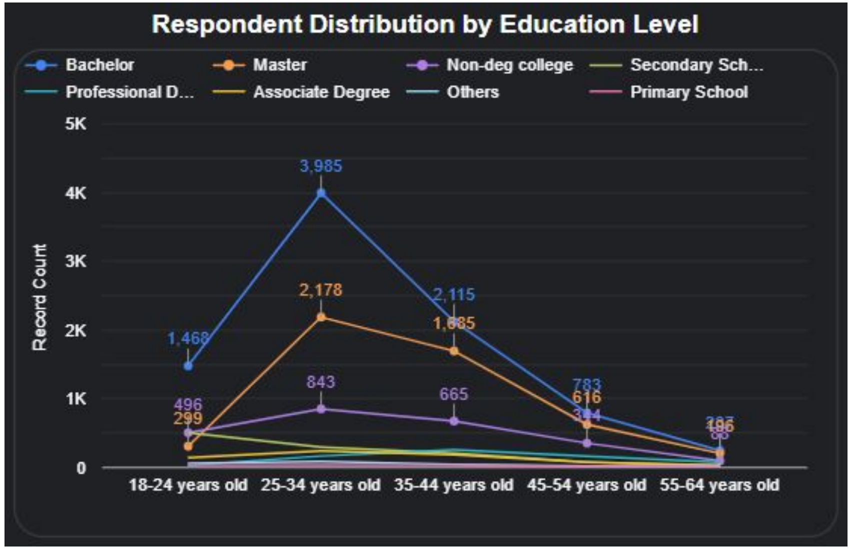


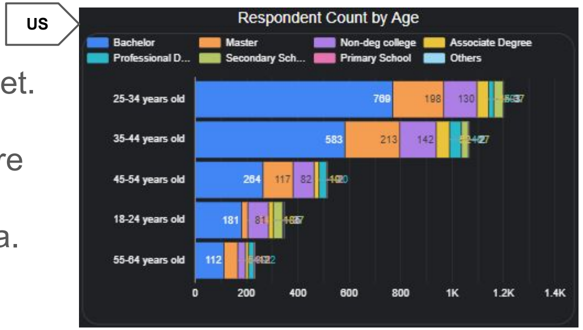


- Top 3 contributors provide over 85% of respondents.
- The Age group of 25-34 years that contributed 41.8%, where ~80% respondents with Bachelor or above levels.
- The group of 35-44 years gives 27.7% that is the 2nd high contributor, where ~75% with Bachelor or above levels.

Observations – Respondents Distribution

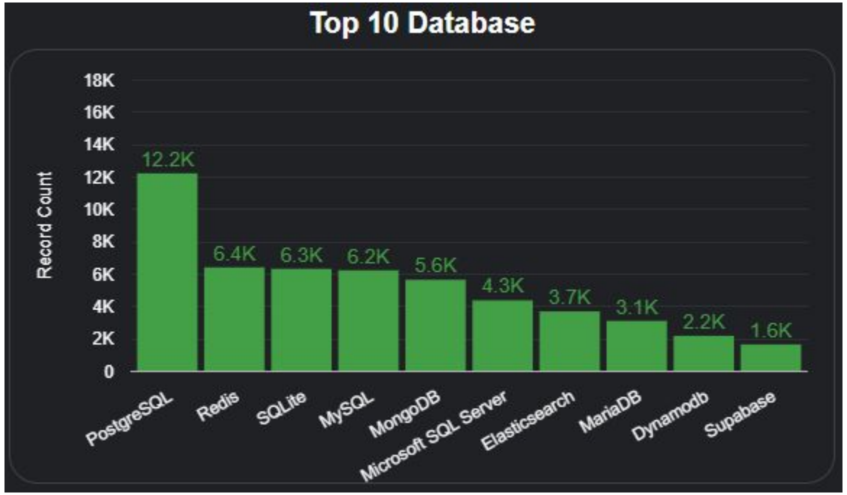


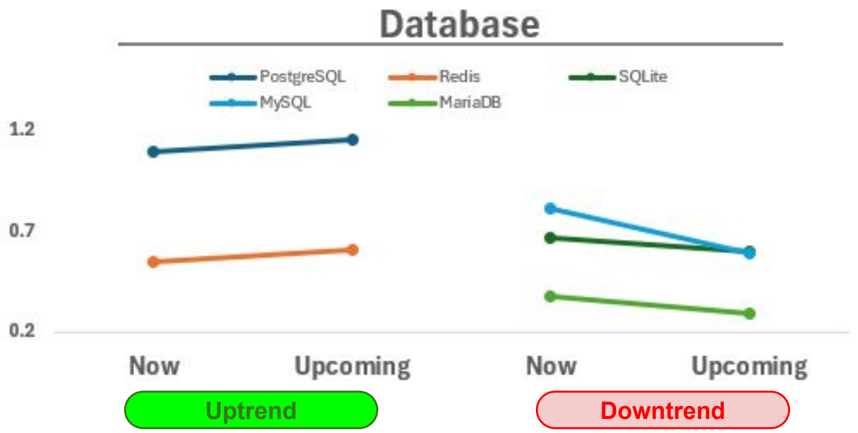




- The United States has the highest respondent count (3,371), dominating the dataset. There is ~54% respondents have bachelor level or above.
- Strong participation from countries like India, Germany, and UK. They are also more than 1000 respondents
- Global coverage includes North America, Europe, Asia, and parts of South America.
- Geographic diversity supports broad relevance of the survey findings.

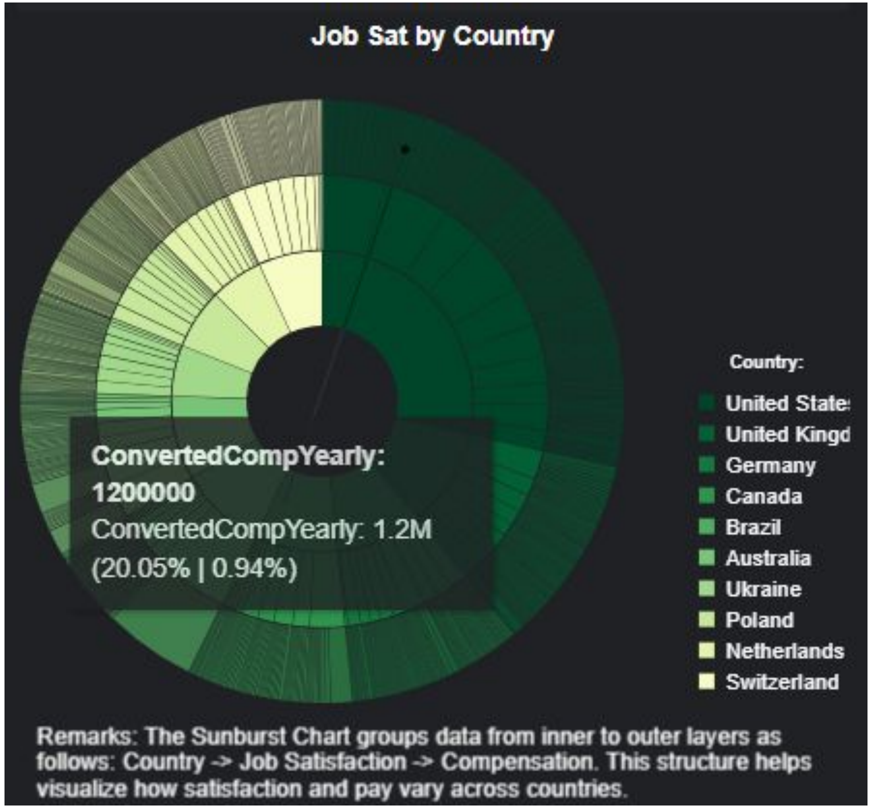
Discussion – Database Trend





- According to normalized analysis, PostgreSQL & Redis achieved stable growing. However, MySQL and MariaDB dropped significantly.

Discussion – Hidden Tendency of Job Satisfaction & Compensation

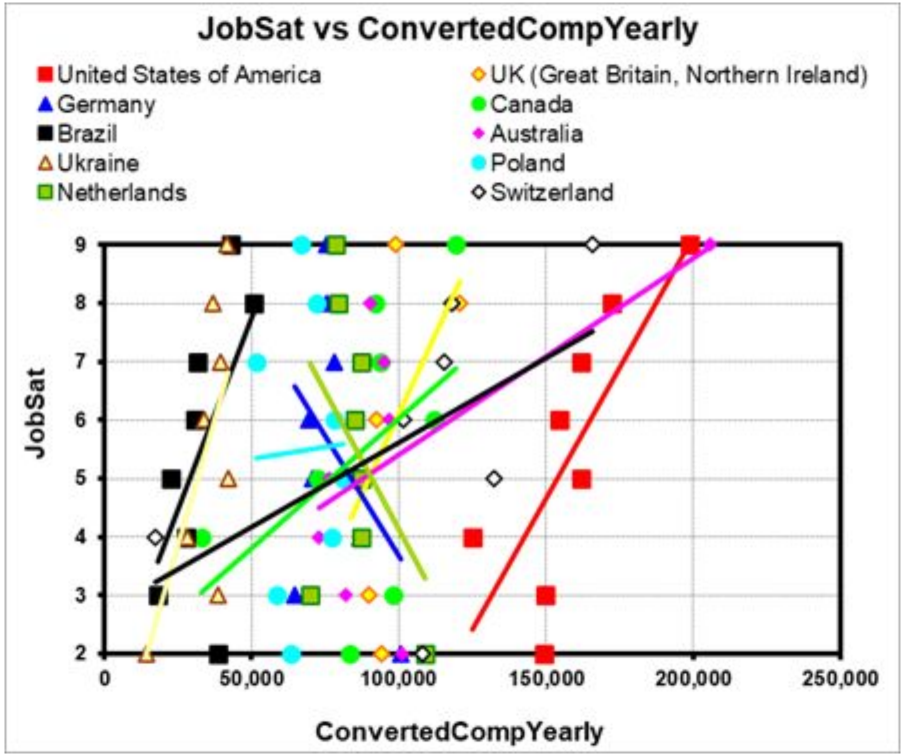


Remarks: The Sunburst Chart groups data from inner to outer layers as follows: Country -> Job Satisfaction -> Compensation. This structure helps visualize how satisfaction and pay vary across countries.

Observations

- Among the selected top 10 countries, there is a consistent pattern: Higher compensation levels tend to align with higher job satisfaction, regardless of absolute values. This correlation is visible across all compensation tiers.

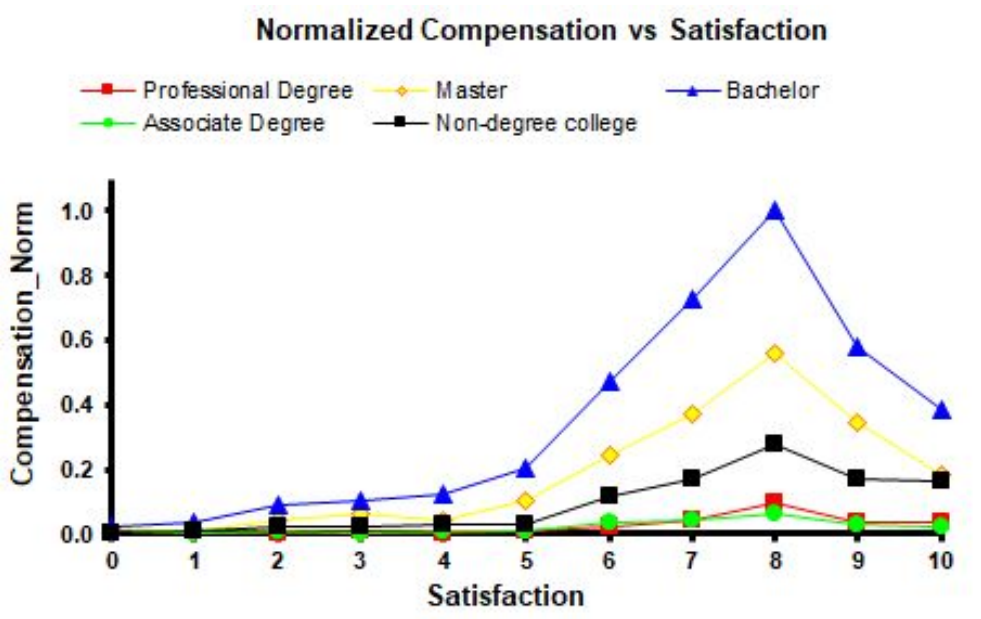
Discussion – Hidden Tendency



Verification

- However, upon deeper analysis, this pattern does not hold universally. In some countries, higher compensation does not consistently lead to higher job satisfaction, suggesting that other cultural or workplace factors may be influencing the outcome.

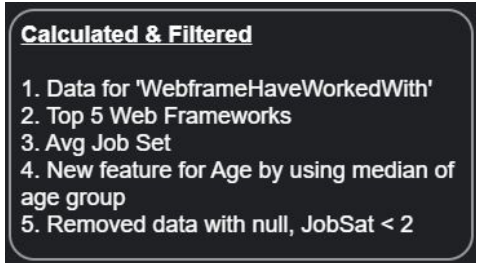
Discussion – Correlation between Normalized Compensation and Satisfaction



Pattern

- Observed the same pattern from Compensation (Normalized) vs Satisfaction chart, Satisfaction arise associated with Compensation from 5 to 8, and drop after that. This phenomena is consistent along these 5 of difference education groups.

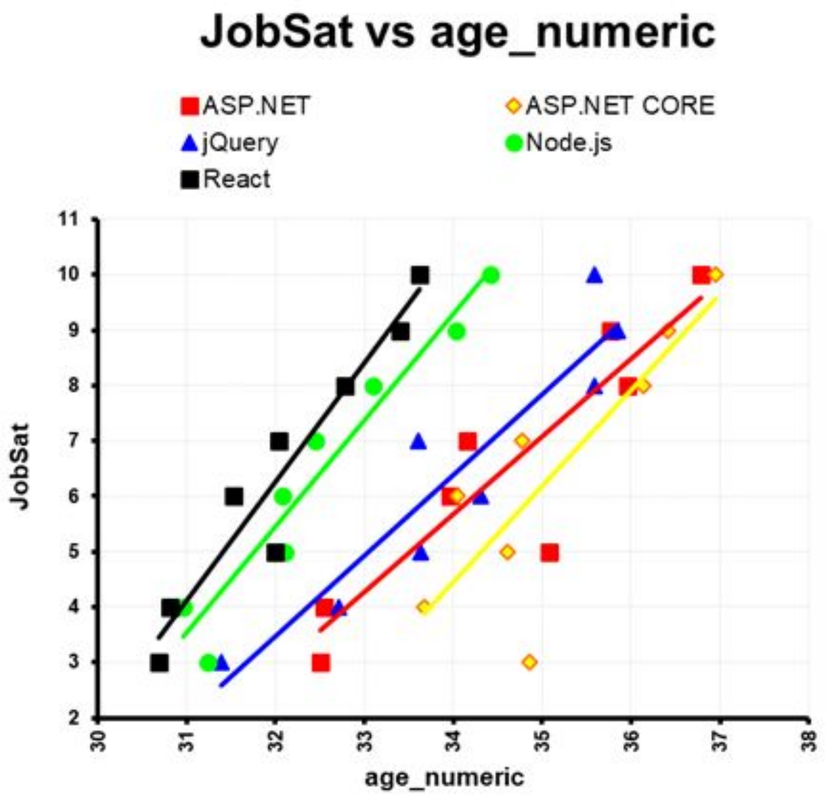
- Hypothesis: Satisfaction 9 & 10 are the critical points that respondents are not easy to mark. If the extra score is not related to the compensation, then another possibility may:
  - Intrinsic Motivation over extrinsic rewards
  - Improved Work Environment
  - Career advancement
  - Reduced Stress / Mental well-being improves



Calculated & Filtered

1. Data for 'WebframeHaveWorkedWith'
2. Top 5 Web Frameworks
3. Avg Job Set
4. New feature for Age by using median of age group
5. Removed data with null, JobSat < 2

Discussion – Correlation of Job Satisfaction & Age



Observations

Trends

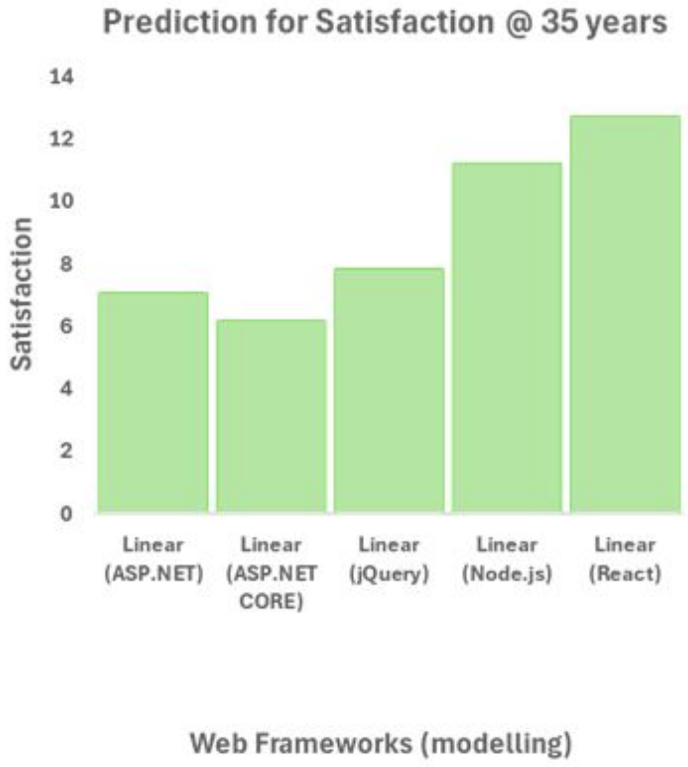
- React and Node.js show steeper upward trends, suggesting that older developers using these frameworks report significantly higher satisfaction
- jQuery has a flatter trend line, implying that satisfaction is relatively stable across age possibly due to its legacy status and limited modern use

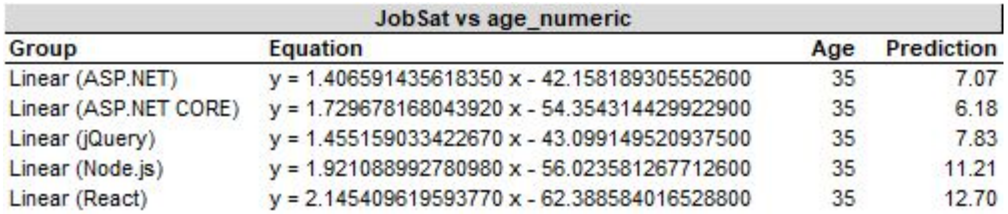
Sensitivity

- From the averaged age range, the slope of satisfaction varies by framework
- This could indicate that framework choice interacts with career maturity, not just age

⇒ *Prediction for the sensitivity*

Prediction – Normalized Age @ 35 and Predict the Satisfaction





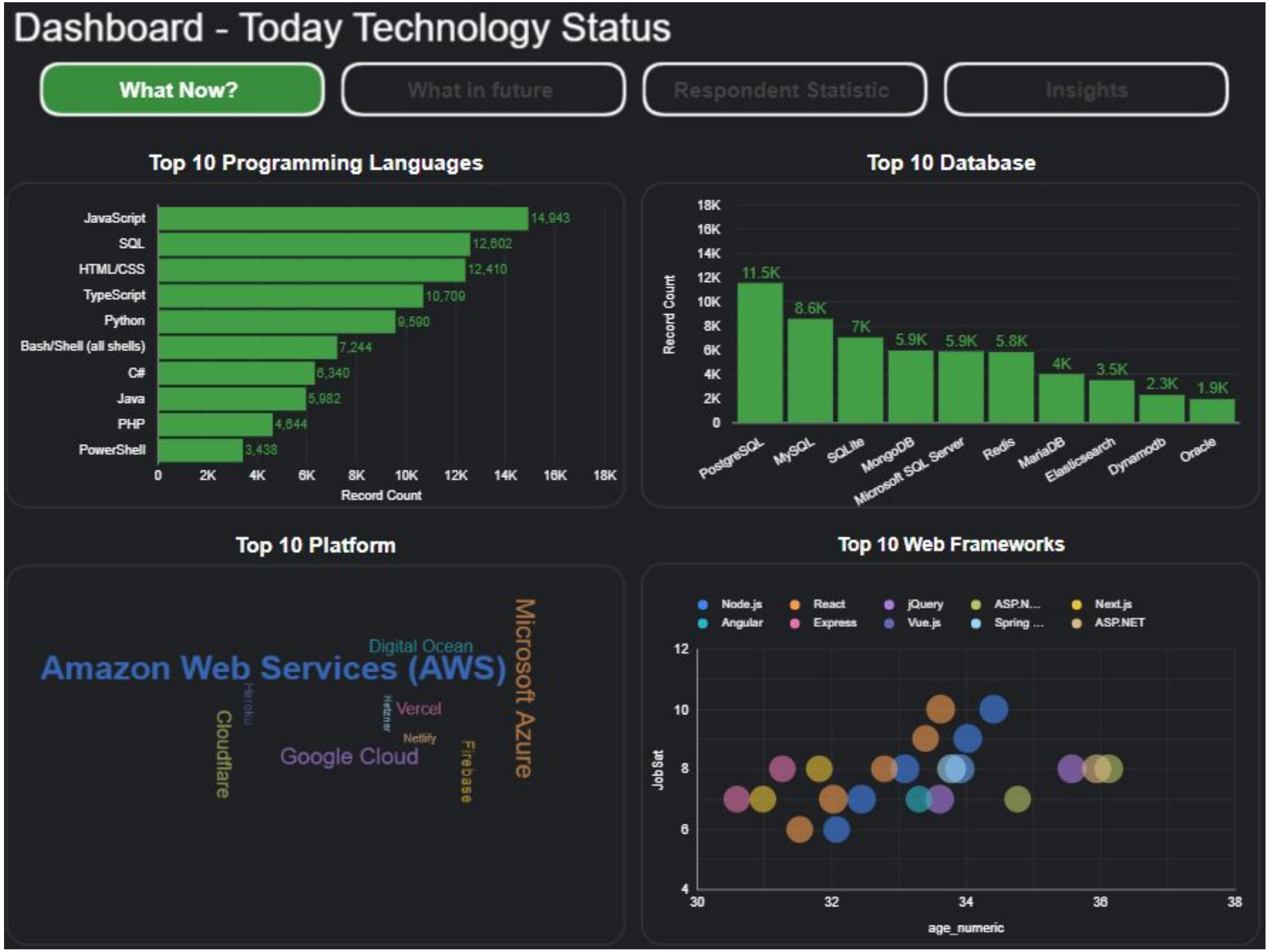
JobSat vs age_numeric			
Group	Equation	Age	Prediction
Linear (ASP.NET)	y = 1.406591435618350 x - 42.158189305552600	35	7.07
Linear (ASP.NET CORE)	y = 1.729678168043920 x - 54.354314429922900	35	6.18
Linear (jQuery)	y = 1.455159033422670 x - 43.099149520937500	35	7.83
Linear (Node.js)	y = 1.921088992780980 x - 56.023581267712600	35	11.21
Linear (React)	y = 2.145409619593770 x - 62.388584016528800	35	12.70

- Based on the sensitivity and align to the Age at 35 years, the predicted satisfaction shows React & Node.js achieved 11.21 and 12.7 respectively.

- One of the critical point is, how these two Web Frameworks can maintain the user and not switch to other competitors.

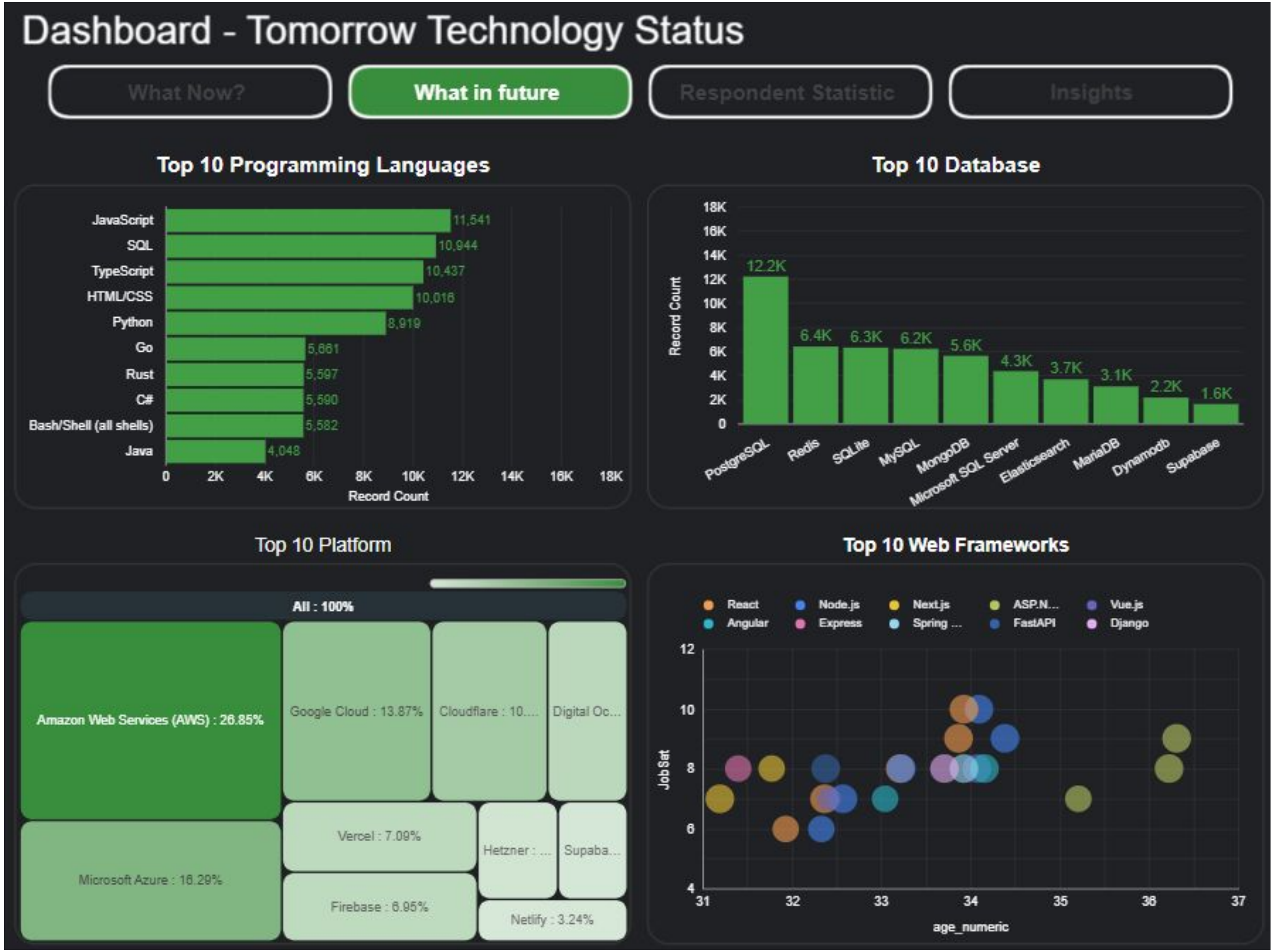
Dashboard

- Current Technology



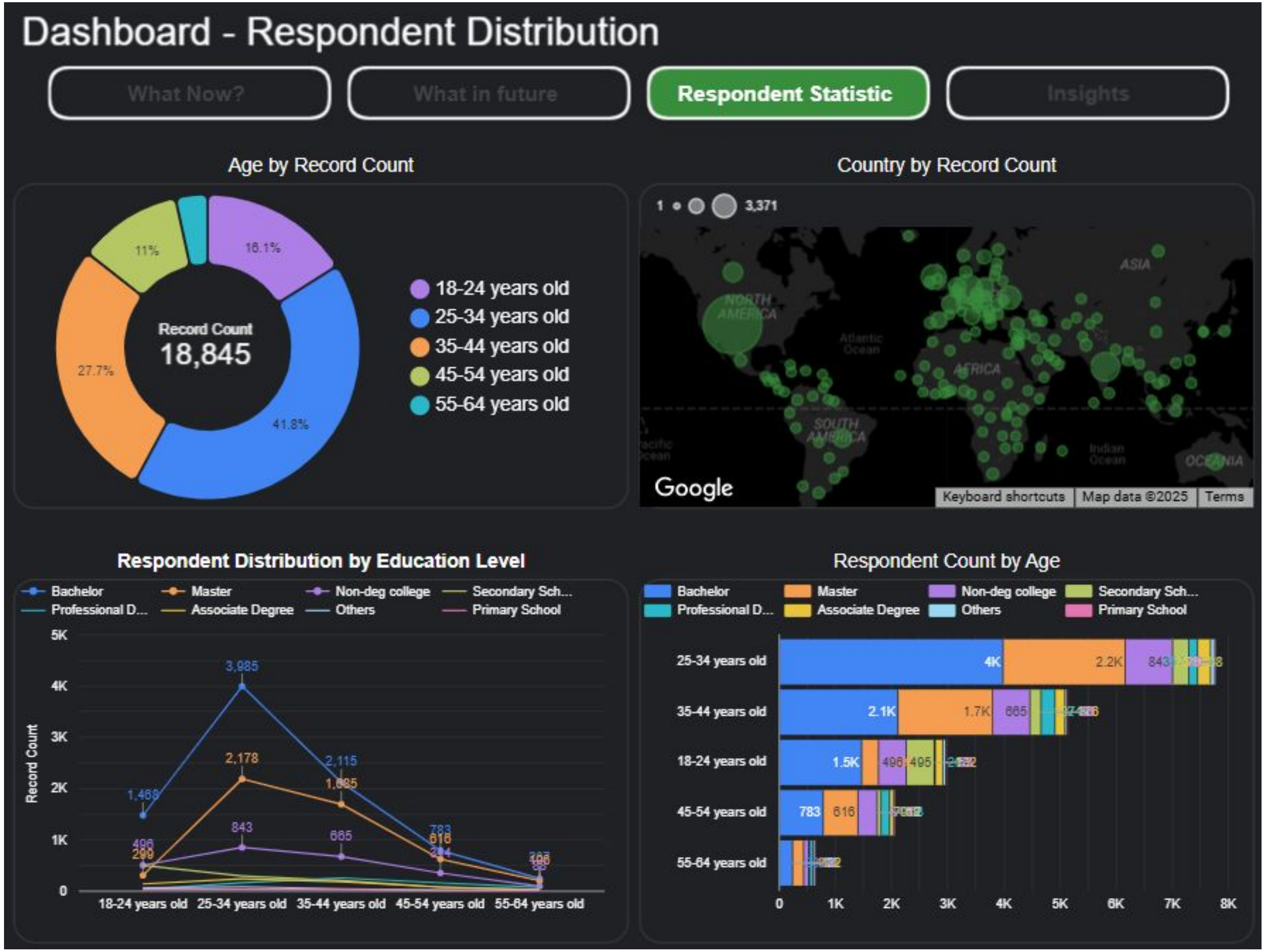
Dashboard

- Next Technology

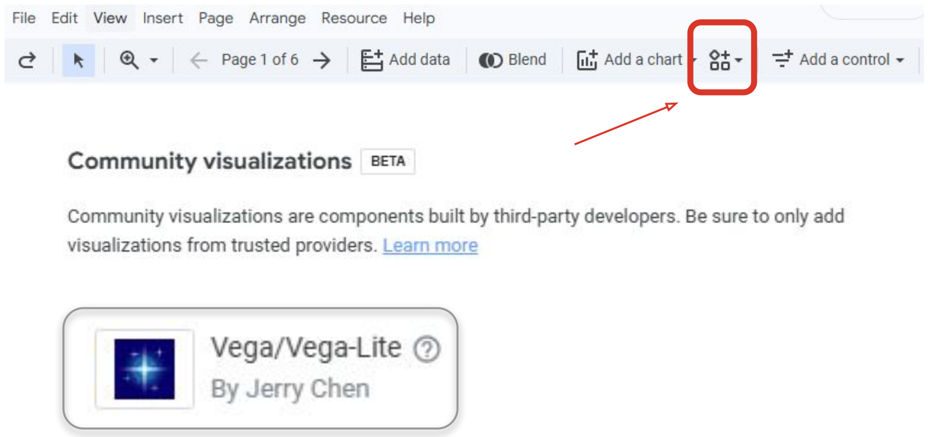


Dashboard

- Statistic of Respondents



Appendix – Word Cloud (Google Looker Studio)

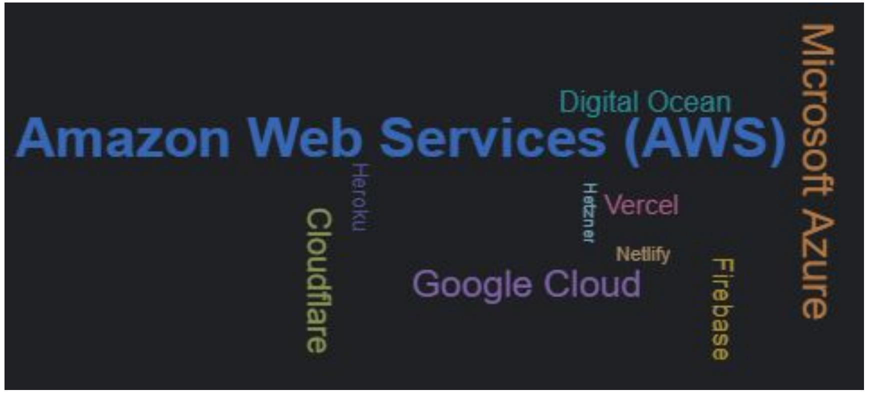


Community visualizations BETA

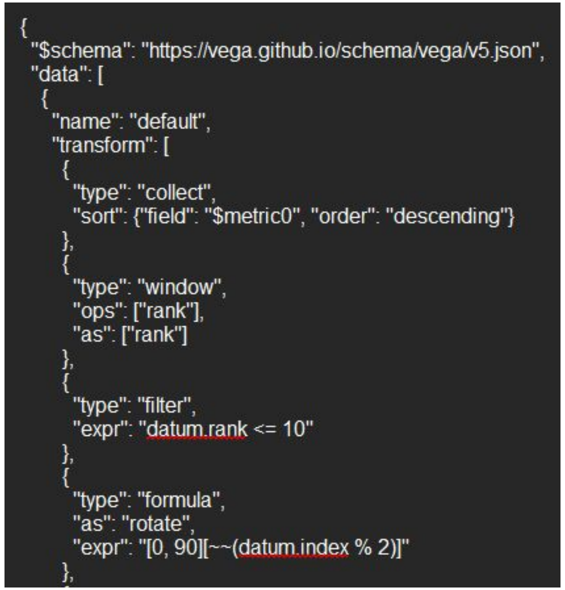
Community visualizations are components built by third-party developers. Be sure to only add visualizations from trusted providers. Learn more

Vega/Vega-Lite

By Jerry Chen

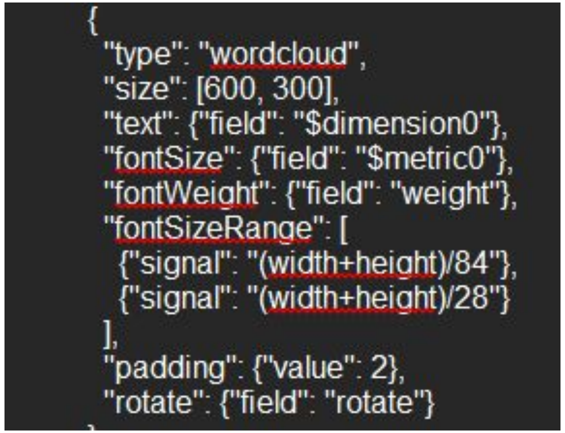


Configuration



```
{
  "$schema": "https://vega.github.io/schema/vega/v5.json",
  "data": [
    {
      "name": "default",
      "transform": [
        {
          "type": "collect",
          "sort": {"field": "$metric0", "order": "descending"}
        },
        {
          "type": "window",
          "ops": ["rank"],
          "as": ["rank"]
        },
        {
          "type": "filter",
          "expr": "datum.rank <= 10"
        },
        {
          "type": "formula",
          "as": "rotate",
          "expr": "[0, 90][~~(datum.index % 2)]"
        },
```

← Max of words



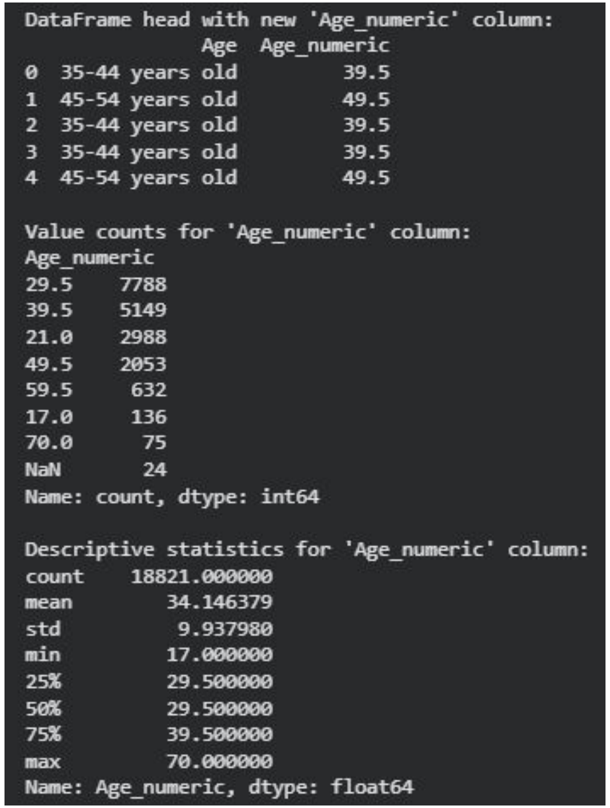
```
    {
      "type": "wordcloud",
      "size": [600, 300],
      "text": {"field": "$dimension0"},
      "fontSize": {"field": "$metric0"},
      "fontWeight": {"field": "weight"},
      "fontSizeRange": [
        {"signal": "(width+height)/84"},
        {"signal": "(width+height)/28"}
      ],
      "padding": {"value": 2},
      "rotate": {"field": "rotate"}
```

← Word cloud size

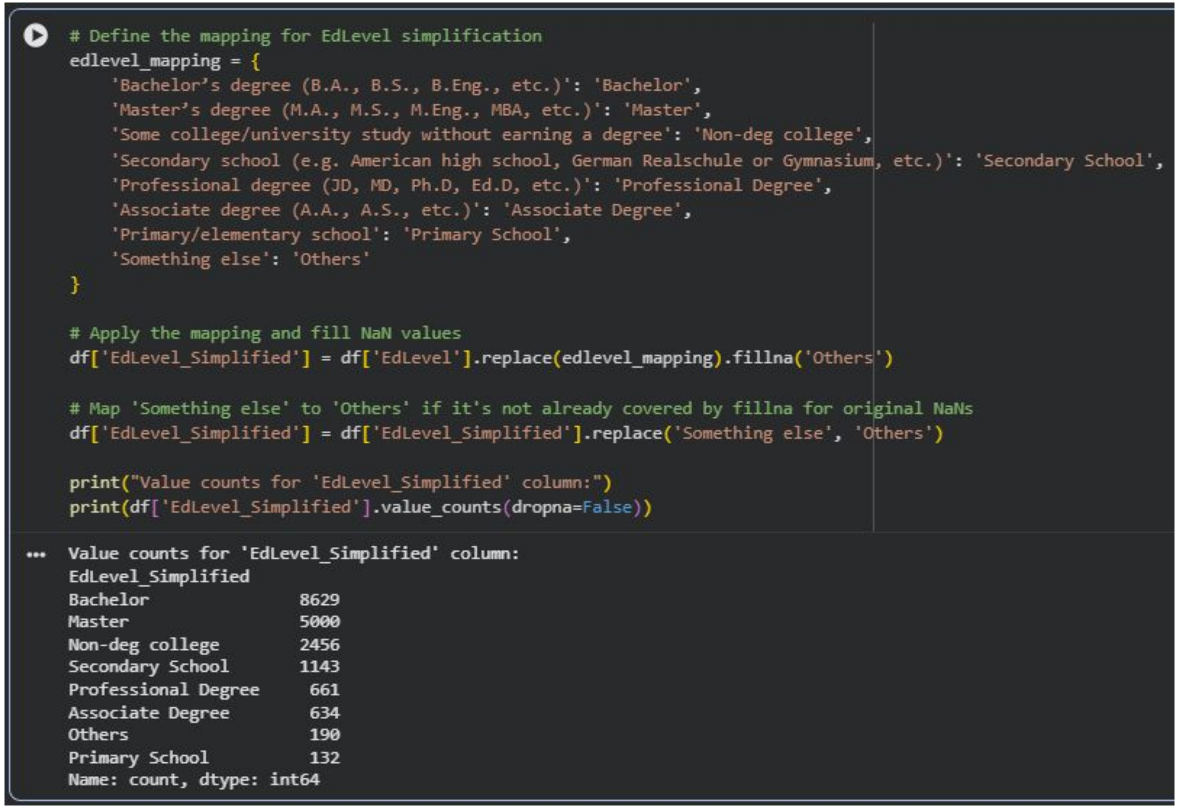
← Formula for calculating min/max word size

← Allow words rotation

Appendix



```
DataFrame head with new 'Age_numeric' column:
               Age  Age_numeric
0  35-44 years old         39.5
1  45-54 years old         49.5
2  35-44 years old         39.5
3  35-44 years old         39.5
4  45-54 years old         49.5

Value counts for 'Age_numeric' column:
Age_numeric
29.5    7788
39.5    5149
21.0    2988
49.5    2053
59.5     632
17.0     136
70.0      75
NaN       24
Name: count, dtype: int64

Descriptive statistics for 'Age_numeric' column:
count    18821.000000
mean        34.146379
std          9.937980
min         17.000000
25%         29.500000
50%         29.500000
75%         39.500000
max         70.000000
Name: Age_numeric, dtype: float64
```



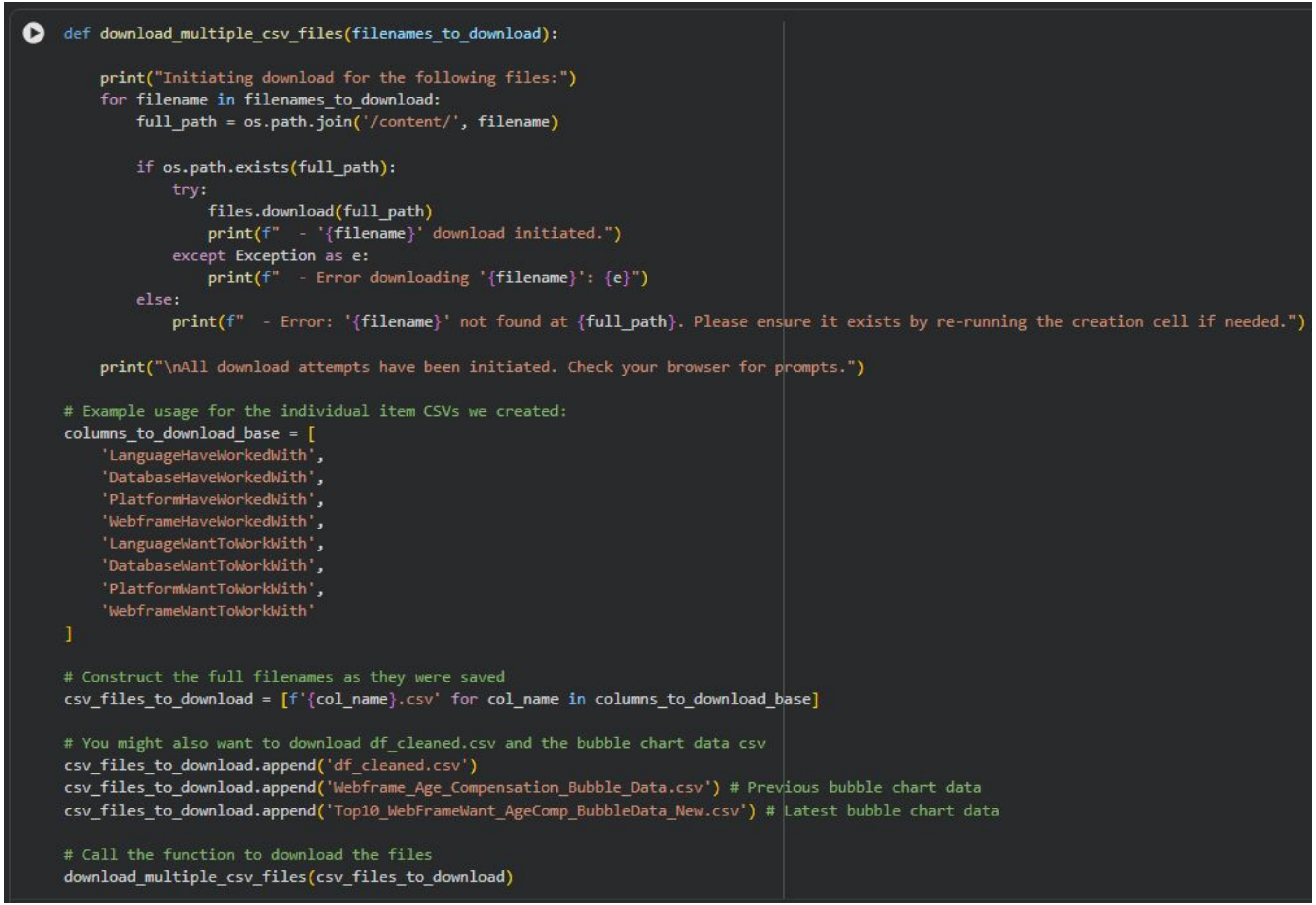
```python
# Define the mapping for EdLevel simplification
edlevel_mapping = {
    'Bachelor’s degree (B.A., B.S., B.Eng., etc.)': 'Bachelor',
    'Master’s degree (M.A., M.S., M.Eng., MBA, etc.)': 'Master',
    'Some college/university study without earning a degree': 'Non-deg college',
    'Secondary school (e.g. American high school, German Realschule or Gymnasium, etc.)': 'Secondary School',
    'Professional degree (JD, MD, Ph.D, Ed.D, etc.)': 'Professional Degree',
    'Associate degree (A.A., A.S., etc.)': 'Associate Degree',
    'Primary/elementary school': 'Primary School',
    'Something else': 'Others'
}

# Apply the mapping and fill NaN values
df['EdLevel_Simplified'] = df['EdLevel'].replace(edlevel_mapping).fillna('Others')

# Map 'Something else' to 'Others' if it's not already covered by fillna for original NaNs
df['EdLevel_Simplified'] = df['EdLevel_Simplified'].replace('Something else', 'Others')

print("Value counts for 'EdLevel_Simplified' column:")
print(df['EdLevel_Simplified'].value_counts(dropna=False))
```

```
Value counts for 'EdLevel_Simplified' column:
EdLevel_Simplified
Bachelor               8629
Master                 5000
Non-deg college        2456
Secondary School       1143
Professional Degree     661
Associate Degree        634
Others                  190
Primary School          132
Name: count, dtype: int64
```

Feathers Engineering

- Created columns ‘Age_numeric’ and ‘EdLevel_Simplified’

Appendix

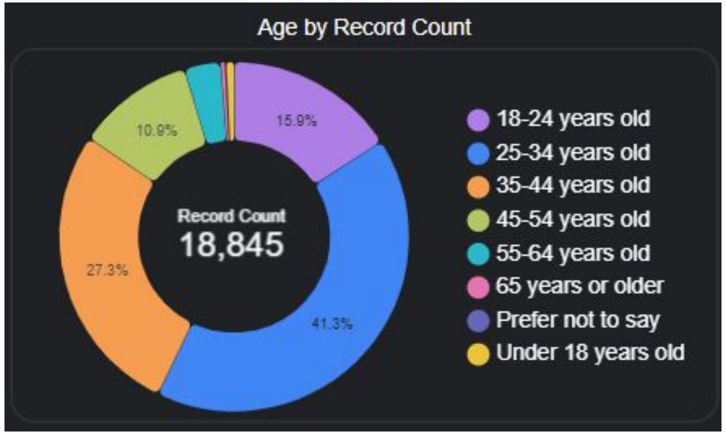


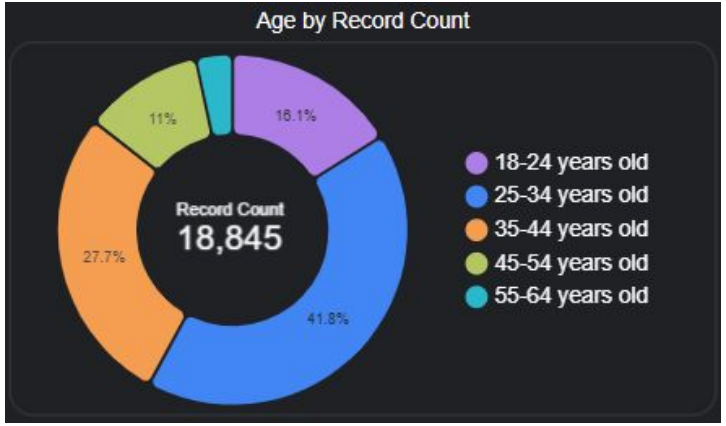
```
def download_multiple_csv_files(filenames_to_download):

    print("Initiating download for the following files:")
    for filename in filenames_to_download:
        full_path = os.path.join('/content/', filename)

        if os.path.exists(full_path):
            try:
                files.download(full_path)
                print(f"  - '{filename}' download initiated.")
            except Exception as e:
                print(f"  - Error downloading '{filename}': {e}")
        else:
            print(f"  - Error: '{filename}' not found at {full_path}. Please ensure it exists by re-running the creation cell if needed.")

    print("\nAll download attempts have been initiated. Check your browser for prompts.")

# Example usage for the individual item CSVs we created:
columns_to_download_base = [
    'LanguageHaveWorkedWith',
    'DatabaseHaveWorkedWith',
    'PlatformHaveWorkedWith',
    'WebframeHaveWorkedWith',
    'LanguageWantToWorkWith',
    'DatabaseWantToWorkWith',
    'PlatformWantToWorkWith',
    'WebframeWantToWorkWith'
]

# Construct the full filenames as they were saved
csv_files_to_download = [f'{col_name}.csv' for col_name in columns_to_download_base]

# You might also want to download df_cleaned.csv and the bubble chart data csv
csv_files_to_download.append('df_cleaned.csv')
csv_files_to_download.append('Webframe_Age_Compensation_Bubble_Data.csv') # Previous bubble chart data
csv_files_to_download.append('Top10_WebFrameWant_AgeComp_BubbleData_New.csv') # Latest bubble chart data

# Call the function to download the files
download_multiple_csv_files(csv_files_to_download)
```

Export csv files

- Backup those dataframe in csv files for further usage.

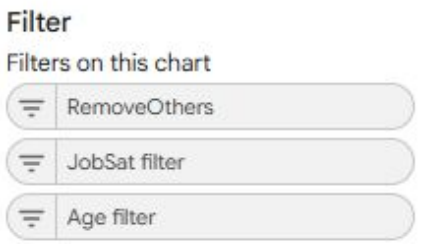
Appendix





Removed noise in Age

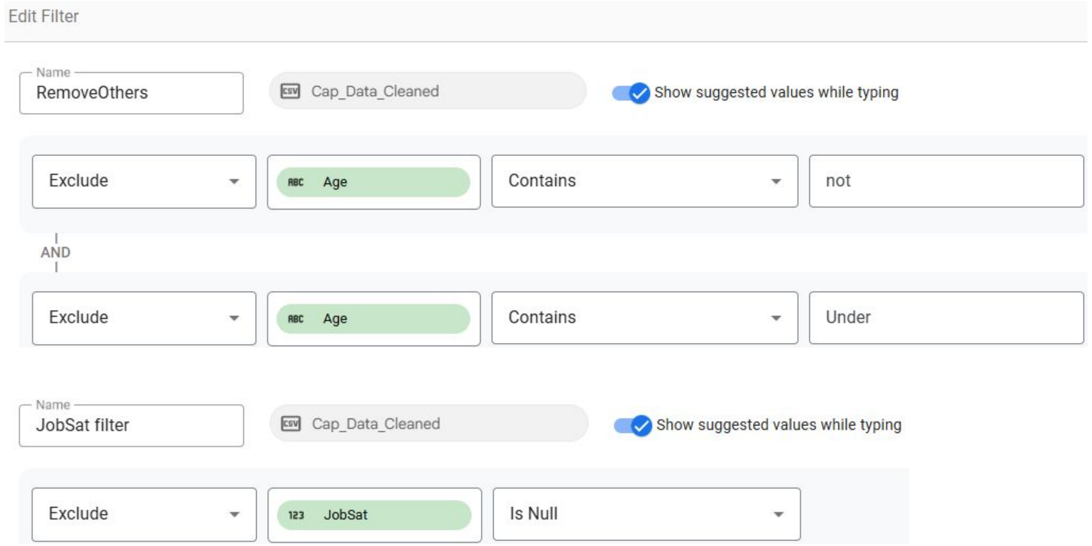
- Applied Filters in Looker Studio to exclude:
  - 65 years or older
  - Prefer not to say
  - Under 18 years old



Filter

Filters on this chart

- RemoveOthers
- JobSat filter
- Age filter



Edit Filter

Name: RemoveOthers — Cap_Data_Cleaned — Show suggested values while typing

Exclude	Age	Contains	not
AND			
Exclude	Age	Contains	Under

Name: JobSat filter — Cap_Data_Cleaned — Show suggested values while typing

Exclude	JobSat	Is Null

End of Presentation

Thank You

Let’s Connect

🔗 Portfolio: github.com/dannytang-analytics
🔗 Dashboards: lookerstudio.google.com/dannytang/CapProj2
📧 Email: dannypctang@gmail.com
💼 LinkedIn: www.linkedin.com/in/dannypctang