

# 프로젝트 진행 순서

1. 프로젝트 개요
2. 연관 규칙 분석
3. 데이터 수집 및 가공
4. 워드클라우드와 트리맵
5. 데이터 학습 및 시각화

# Section 01

## 프로젝트 개요

# 1. 프로젝트의 목표

## ■ 프로그래밍 언어 간의 연관 관계를 추출하기

- 기계 학습 라이브러리를 사용해 연관 규칙 분석을 수행
- 파이썬을 사용할 수 있는 데이터 분석가가 사용할 수 있는 다른 프로그래밍 언어의 종류 살펴보기
  - 데이터 분석가들이 주로 사용하는 프로그래밍 언어는 무엇인지 알아보기
  - 각 프로그래밍 언어 간 연관 관계를 시각화

## 2. 문제 정의

### ■ 데이터 분석가들이 다루는 프로그래밍 언어 분석

저는 미래의 데이터 분석가, 신후입니다. 현재 파이썬을 열심히 배우고 있는데, 추가로 어떤 프로그래밍 언어를 배워야 취업 시장에서 경쟁력이 있을지 고민 중입니다. 데이터 분석가가 다루는 언어로 파이썬과 R을 많이 언급하던데, 그 두 언어를 익히면 충분할 것인지, 아니면 또 다른 프로그래밍 언어를 학습해야 할지 궁금합니다.

그래서, 전 세계의 데이터 분석가들이 다루는 프로그래밍 언어가 무엇인지 데이터를 통해 확인하고 싶습니다. 그리고 파이썬을 다루는 데이터 분석가들이 함께 다루는 대표적인 프로그래밍 언어가 무엇인지도 알고 싶습니다. 또 파이썬뿐 아니라 어떤 언어를 다루는 것이 좋을지 그래프로 표현하고 싶습니다.

제가 배운 파이썬 데이터 분석 기술로 항목 간의 연관 관계를 계산해 그래프로 그리고 싶은데, 어떻게 해야 하나요?



## Section 02

### 연관 규칙 분석

# 1. 연관 규칙 분석의 개념

## ■ 연관 규칙 분석(Association Rule Analysis)

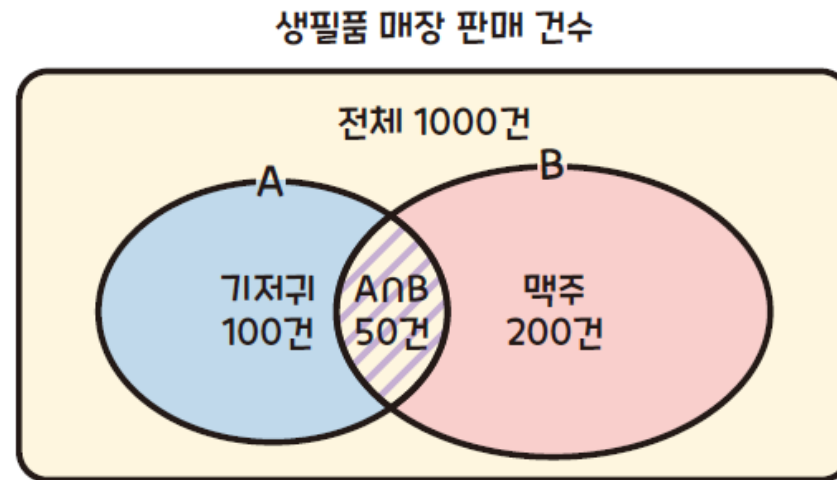
- 일련의 거래나 사건 안에 존재하는 항목 간의 연관 규칙을 발견하는 분석 방식
- 흔히 장바구니 분석(Market Basket Analysis)이라고도 함
  - 고객의 구매 기록을 분석해 상품 간의 연관성을 측정
  - 고객이 상품 구매 시 함께 구매할 가능성이 높은 상품을 특정할 수 있는 데서 붙여진 이름임

# 1. 연관 규칙 분석의 개념

## ■ 생필품 매장에서 찾을 수 있는 연관 규칙의 형태

A를 구매하면, B를 구매할 가능성이 높다.

- EX) 기저귀를 구매하는 고객이 맥주를 함께 구매하는 확률이 높다는 것을 발견
- 기저귀와 맥주를 가깝게 진열 → 매출이 상승한 사례




연관 규칙 개념 설명을 위한 벤 다이어그램

## 2. 지지도

### ■ 지지도(Support)

- 연관 규칙을 측정하는 지표
- 전체 거래 중 A(조건Antecedents)와 B(결과Consequent)가 동시에 판매되는 거래의 비율을 의미
  - EX매장의 전체 거래 : 1,000건
  - 기저귀와 맥주를 동시에 산 건수 : 50건 → 지지도는 5%

$$\text{지지도} = P(A \cap B) = \frac{\text{A와 B를 포함하는 거래 수}}{\text{전체 거래 수}}$$


연관 규칙 측정 지표 중 지지도 이해



### 3. 신뢰도

#### ■ 신뢰도(Confidence)

- 연관 규칙을 측정하는 지표
- A를 구매한 거래 중 B도 함께 구매한 거래의 비율을 의미
  - EX) 기저귀를 구매한 100건 중 맥주도 함께 구매한 건수 : 50건
  - 신뢰도는 50%

$$\text{신뢰도} = \frac{P(A \cap B)}{P(A)} = \frac{\text{A와 B를 포함하는 거래 수}}{\text{A를 포함하는 거래 수}}$$

연관 규칙 측정 지표 중 신뢰도 이해

## 4. 향상도

### ■ 향상도(Lift)

- 연관 규칙을 측정하는 지표
- 전체 건수에서 B를 구매한 비율 대비 A를 구매했을 때 신뢰도의 증가 비율
  - EX) 기저귀 구매 유무와 상관 없이 맥주를 구매한 비율은 전체 건수 1,000건 중 200건 → 20%
  - 기저귀를 구매했을 때 맥주를 구매하는 비율은 신뢰도를 의미 → 50%
  - 향상도는  $50\%/20\% = 2.5$
  - ✓ 기저귀를 구매한 경우에 맥주를 구매할 확률이 2.5배 높다는 의미

$$\text{향상도} = \frac{P(A \cap B)}{P(A)P(B)} = \frac{\text{A와 B를 포함하는 거래 수} \times \text{전체 거래 수}}{\text{A 포함 거래 수} \times \text{B 포함 거래 수}} = \frac{\text{신뢰도}}{P(B)}$$

연관 규칙 측정 지표 중 향상도 이해

## 5. 어프라이어리

### ■ 어프라이어리(Apriori)

- 연관 규칙을 추출 하기 위한 규칙
  - 각 항목이 항목 집합 안에서 어떤 빈도로 출현했고 어떤 항목과 함께 출현했는지 파악
  - 하지만, 데이터 셋이 큰 경우 모든 후보 항목 집합을 하나씩 검사하는 것은 비효율적임
- 어프라이어리 알고리즘의 원리 → 빈발 항목 집합을 추출
  - 빈발 항목 집합이란? 지지도가 최소 이상인 항목 집합
  - 모든 항목 집합에 대한 복잡한 계산량을 줄이기 위해 최소 지지도를 정하고 그 이상의 값만 찾아 연관 규칙을 생성
  - 만약, 항목 집합의 지지도가 사용자가 정한 최소 지지도를 넘지 못하면 연산 대상에서 제외하는 방식으로 계산량을 줄임
  - ✓ 장점 : 원리가 간단함
  - ✓ 단점 : 데이터가 클수록 속도가 느려지고 연산량이 많아짐

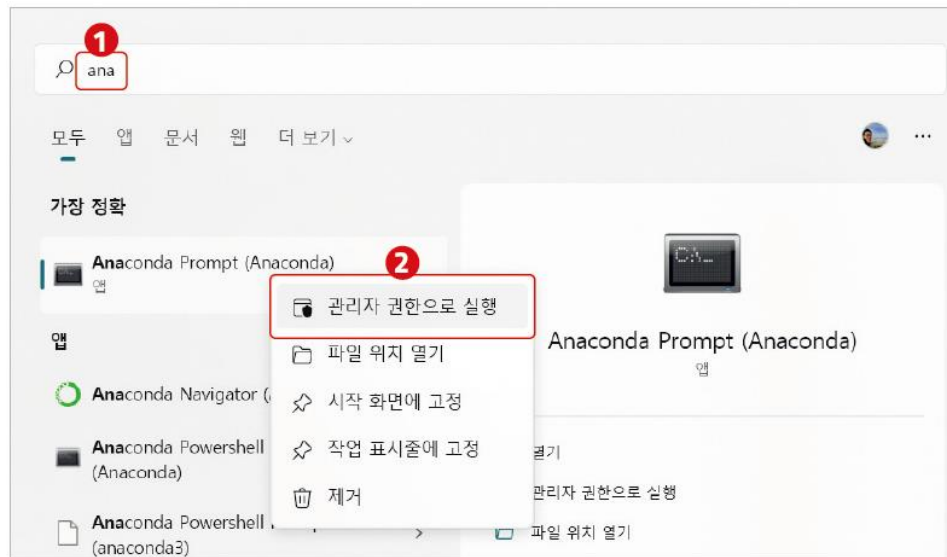
## Section 03

### 데이터 수집 및 가공

# 1. 어프라이어리 설치하기

## ■ 외부 라이브러리 설치

- <윈도우 키>를 누른 뒤 검색창에 'anaconda prompt'를 입력
- 검색한 앱→ 마우스 오른쪽 버튼 클릭→ [관리자 권한으로 실행]을 클릭



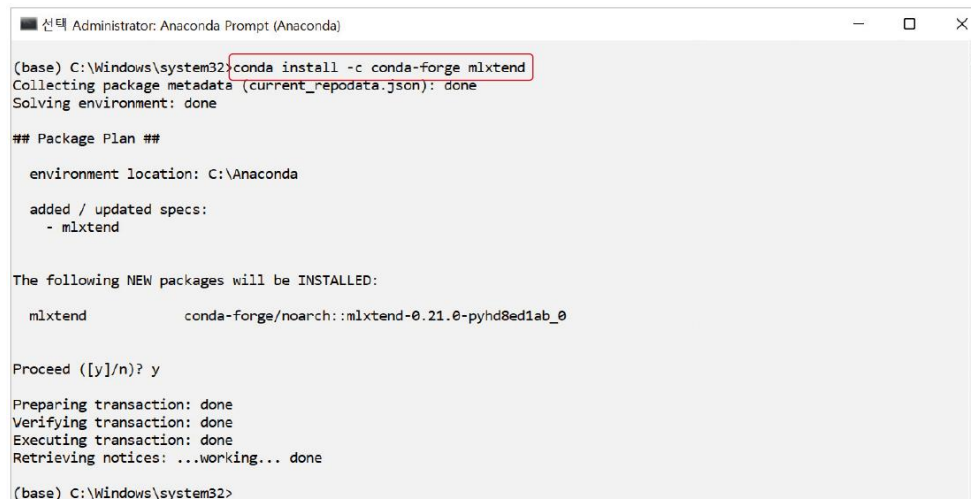
아나콘다 프롬프트(Anaconda Prompt) 열기

# 1. 어프라이어리 설치하기

## ■ 설치 명령어 실행

- Mlxtend 라이브러리
  - 어프라이어리 알고리즘을 사용할 수 있는 기능을 제공
  - 알고리즘 적용을 하기 위해 수행해야 하는 데이터 전처리 기능도 함께 제공
- 명령어 실행 후 Proceed ([y]/n)? 문장 → y를 입력해 설치를 진행

```
conda install -c conda-forge mlxtend
```



```
Administrator: Anaconda Prompt (Anaconda)

(base) C:\Windows\system32>conda install -c conda-forge mlxtend
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Anaconda

  added / updated specs:
    - mlxtend

The following NEW packages will be INSTALLED:

  mlxtend                conda-forge/noarch::mlxtend-0.21.0-pyhd8ed1ab_0

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done

(base) C:\Windows\system32>
```

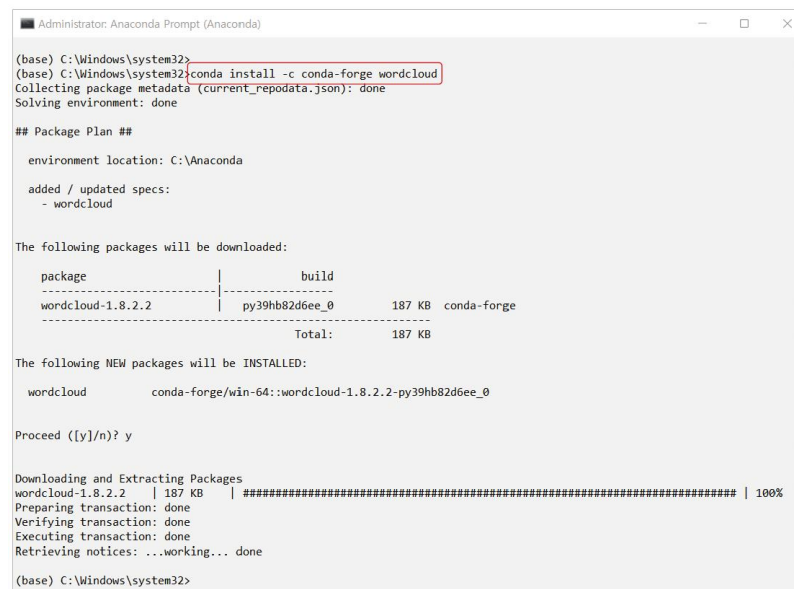
아나콘다 프롬프트에서 Mlxtend 라이브러리 설치하기

# 1. 어프라이어리 설치하기

## ■ 워드클라우드 생성

- 프로그래밍 언어 빈도를 쉽게 확인
- 다음 명령어를 실행 → Wordcloud 라이브러리 설치
- 명령어 실행 후 Proceed ([y]/n)? 문장 → y를 입력해 설치를 진행

```
conda install -c conda-forge wordcloud
```



```
Administrator: Anaconda Prompt (Anaconda)

(base) C:\Windows\system32>
(base) C:\Windows\system32>conda install -c conda-forge wordcloud
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Anaconda

added / updated specs:
- wordcloud

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
wordcloud-1.8.2.2 | py39hb82d6ee_0 | 187 KB | conda-forge
Total: 187 KB

The following NEW packages will be INSTALLED:

wordcloud | conda-forge/win-64::wordcloud-1.8.2.2-py39hb82d6ee_0

Proceed ([y]/n)? y

Downloading and Extracting Packages:
wordcloud-1.8.2.2 | 187 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done

(base) C:\Windows\system32>
```

아나콘다 프롬프트에서 Wordcloud 라이브러리 설치하기

## 2. 데이터 적재하기

### ■ 개발자 설문 조사 데이터 불러오기

✓신규 노트북 파일을 생성 : '9장. 프로그래밍 언어 사이의 연관 규칙 분석'

- 다음 소스 코드 실행
- 데이터 정보를 확인하여 정상적으로 적재가 되었는지 확인함

→ 79개 열이 있는 7,368건의 데이터가 데이터프레임 raw\_data에 저장되어 있음

```
In [1]: # 판다스 라이브러리 탑재
import pandas as pd

# CSV 파일 읽어오기
raw_data =
pd.read_csv("C:\data\survey_results_public.csv")

# 데이터프레임 정보 확인하기
raw_data.info()
```



## 2. 데이터 적재하기

### ■ In [1]: 코드 실행 결과

```
Out [1]: <class 'pandas.core.frame.DataFrame'>
RangeIndex: 73268 entries, 0 to 73267
Data columns (total 79 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ResponseId                           73268 non-null  int64
1   MainBranch                           73268 non-null  object
2   Employment                           71709 non-null  object
3   RemoteWork                           58958 non-null  object
4   CodingActivities                     58899 non-null  object
5   EdLevel 71571 non-null object
----- ( 중 략 ) -----
74  TrueFalse_2                           35715 non-null  object
75  TrueFalse_3                           35749 non-null  object
76  SurveyLength                          70444 non-null  object
77  SurveyEase                            70508 non-null  object
78  ConvertedCompYearly                   38071 non-null  float64
dtypes: float64(5), int64(1), object(73)
memory usage: 44.2+ MB
```

## 2. 데이터 적재하기

### ■ 설문 조사 결과 일부 데이터 확인하기

In [2]:

# 일부 데이터 확인하기  
raw\_data.head()

Out [2]:

	ResponseId	MainBranch	Employment	RemoteWork	CodingActivities	EdLevel	LearnCode	LearnCor
0	1	None of these	NaN	NaN	NaN	NaN	NaN	
1	2	I am a developer by profession	Employed, full-time	Fully remote	Hobby;Contribute to open-source projects	NaN	NaN	
2	3	I am not primarily a developer, but I write co...	Employed, full-time	Hybrid (some remote, some in-person)	Hobby	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	Books / Physical media;Friend or family member...	documentation;Blogs;Prog
3	4	I am a developer by profession	Employed, full-time	Fully remote	I don't code outside of work	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Books / Physical media;School (I.e., Universit...	
4	5	I am a developer by profession	Employed, full-time	Hybrid (some remote, some in-person)	Hobby	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Other online resources (e.g., videos, blogs, f...	documentation;Blic Ove

5 rows × 79 columns

### 3. 데이터 분석가 데이터만 추출하기

#### ■ 개발자 타입(Developer type) 세션 확인하기

- Stack OverFlow의 설문 조사 결과 페이지 확인
- 개발자 타입 세션→ 해당 값은 열 이름 중 DevType과 매핑됨



설문 조사 결과에서 개발자 타입 확인하기

### 3. 데이터 분석가 데이터만 추출하기

#### ■ 데이터프레임에서 DevType의 값 확인하기

- 각 개발자 타입의 값이 세미콜론(;)으로 구분되어 있음

In [3]:	# 개발자 타입 열 데이터 확인하기 raw_data['DevType']
Out [3]:	0 NaN 1 NaN 2 Data scientist or machine learning specialist;... 3 Developer, full-stack 4 Developer, front-end;Developer, full-stack;Dev... ... 73263 Developer, back-end 73264 Data scientist or machine learning specialist 73265 Developer, full-stack;Developer, desktop or en... 73266 Developer, front-end;Developer, desktop or ent... 73267 Developer, front-end;Engineer, data;Engineer, ... Name: DevType, Length: 73268, dtype: object

### 3. 데이터 분석가 데이터만 추출하기

#### ■ 개발자 타입 문자열을 리스트로 변경하기

- 데이터프레임의 열을 선택한 후 str 객체를 호출
  - split() 메서드
- 각 개발자 타입 문자열을 구분자인 세미콜론으로 나눈 리스트로 변경

In [4]:	# 문자열을 리스트로 변환 dev_type = raw_data['DevType'].str.split(';') dev_type	
Out [4]:	0	NaN
	1	NaN
	2	[Data scientist or machine learning specialist...
	3	[Developer, full-stack]
	4	[Developer, front-end, Developer, full-stack, ...
		...
	73263	[Developer, back-end]
	73264	[Data scientist or machine learning specialist]
	73265	[Developer, full-stack, Developer, desktop or ...
	73266	[Developer, front-end, Developer, desktop or e...
	73267	[Developer, front-end, Engineer, data, Enginee...
	Name: DevType, Length: 73268, dtype: object	

### 3. 데이터 분석가 데이터만 추출하기

#### ■ 결손치(NaN) 제거

- dropna() 메서드

→ 73,268건의 데이터가 61,302건으로 줄어들었음

In [5]:	# 결손치 제거 dev_type.dropna(inplace=True) dev_type	
Out [5]:	2	[Data scientist or machine learning specialist...
	3	[Developer, full-stack]
	4	[Developer, front-end, Developer, full-stack, ...
	7	[Developer, full-stack, Student]
	8	[Developer, back-end]
		...
	73263	[Developer, back-end]
	73264	[Data scientist or machine learning specialist]
	73265	[Developer, full-stack, Developer, desktop or ...
	73266	[Developer, front-end, Developer, desktop or e...
	73267	[Developer, front-end, Engineer, data, Enginee...
	Name: DevType, Length: 61302, dtype: object	



### 3. 데이터 분석가 데이터만 추출하기

## ■ 리스트 항목 열로 나누기

- explode() 메서드

In [6]:	# 리스트 항목을 각 열로 나누기 exploded_dev_type = dev_type.explode() exploded_dev_type
Out [6]:	<div> <div>2</div> <div>Data scientist or machine learning specialist</div> </div> <div> <div>2</div> <div>Developer, front-end</div> </div> <div> <div>2</div> <div>Engineer, data</div> </div> <div> <div>2</div> <div>Engineer, site reliability</div> </div> <div> <div>3</div> <div>Developer, full-stack</div> </div> <div> <div>73267</div> <div>...</div> <div>Data or business analyst</div> </div> <div> <div>73267</div> <div>Designer</div> </div> <div> <div>73267</div> <div>Scientist</div> </div> <div> <div>73267</div> <div>Product manager</div> </div> <div> <div>73267</div> <div>System administrator</div> </div> <div> <div>Name: DevType, Length: 164790, dtype: object</div> </div>



### 3. 데이터 분석가 데이터만 추출하기

#### ■ 유일한 값 리스트 구하기

- unique() 메서드
- 데이터 타입 중 데이터 분석가로 볼 수 있는 항목
  - Data scientist or machine learning specialist
  - Data or business analyst

In [7]:	# 유일한 값 확인하기 exploded_dev_type.unique()
Out [7]:	array(['Data scientist or machine learning specialist', 'Developer, front-end', 'Engineer, data', 'Engineer, site reliability', 'Developer, full-stack', 'Developer, back-end', ------(중략)----- 'Developer, game or graphics', 'Project manager', 'Cloud infrastructure engineer', 'Data or business analyst', 'Designer', 'Scientist', 'Product manager', 'Senior Executive (C-Suite, VP, etc.)', 'System administrator', 'Blockchain', 'Marketing or sales professional', 'Security professional'], dtype=object)

### 3. 데이터 분석가 데이터만 추출하기

#### ■ 데이터프레임 필터링 기능 사용

- 두 항목을 포함한 리스트를 항목으로 갖는 데이터를 추출  
→ 대괄호([]) 안에 조건을 넣는 방법을 사용함
- isin() 메서드  
→ 리스트 항목에 특정 항목이 있는지 확인하는 메서드

```
In [8]: # 데이터 분석가 데이터만 추출하기
data_analyst_data =
raw_data[raw_data['DevType'].isin(['Data scientist or
machine learning specialist',
'Data or business analyst'])]

# 데이터 타입 항목 값 확인하기
data_analyst_data['DevType']
```

### 3. 데이터 분석가 데이터만 추출하기

#### ■ In [8]: 코드 실행 결과

- 원하는 개발자 타입 정보만 추출 → 742건 출력

Out [8]:	240	Data or business analyst
	400	Data or business analyst
	463	Data scientist or machine learning specialist
	1089	Data scientist or machine learning specialist
		...
	73053	Data or business analyst
	73054	Data or business analyst
	73064	Data scientist or machine learning specialist
	73204	Data or business analyst
	73264	Data scientist or machine learning specialist
Name: DevType, Length: 742, dtype: object		

## 4. 프로그래밍 언어 데이터만 추출하기

### ■ 분석 대상 범위 좁히기

- 연관 관계 분석은 분석 대상과 학습 범위를 좁힐수록 좋음
- 프로그래밍 언어를 담고 있는 열의 이름 : LanguageHaveWorkedWith
- 다음 소스 코드를 실행

→ 프로그래밍 언어들이 하나의 문자열로 출력 → 구분자는 세미콜론(;)

In [9]:	# 프로그래밍 언어 데이터 추출 languages = data_analyst_data['LanguageHaveWorkedWith'] languages	
Out [9]:	240	HTML/CSS;Python;Ruby;SQL;VBA
	400	Bash/Shell;C++;Python;SQL
	------(중략)-----	
		...
	73204	Bash/Shell;HTML/CSS;Julia;R;SQL;VBA
	73264	Bash/Shell;HTML/CSS;JavaScript;Python;SQL
	Name: LanguageHaveWorkedWith, Length: 742, dtype: object	

## 4. 프로그래밍 언어 데이터만 추출하기

### ■ 문자열을 리스트로 변환

- `str.split()` 메서드

In [10]:	# 데이터 문자열 변환 후 구분자(;)로 구분 languages = languages.str.split(';') languages	
Out [10]:	240	[HTML/CSS, Python, Ruby, SQL, VBA]
	400	[Bash/Shell, C++, Python, SQL]
	463	[Bash/Shell, Python, SQL]
	1089	[Python]
	1704	[Elixir, Python, Rust, SQL, TypeScript]
		...
	73053	[PowerShell, Python, SQL]
	73054	[R]
	73064	[Java, Python, Scala, SQL]
	73204	[Bash/Shell, HTML/CSS, Julia, R, SQL, VBA]
	73264	[Bash/Shell, HTML/CSS, JavaScript, Python, SQL]
	Name: LanguageHaveWorkedWith, Length: 742, dtype: object	

## 4. 프로그래밍 언어 데이터만 추출하기

### ■ 리스트 항목을 각각 행으로 변경하기

- explode() 메서드

In [10]:	# 리스트 항목을 행으로 나누기 exploded_languages = languages.explode() exploded_languages
Out [10]:	240           HTML/CSS 240           Python 240           Ruby 240           SQL 240           VBA  ... 73264        Bash/Shell 73264        HTML/CSS 73264        JavaScript 73264        Python 73264        SQL Name: LanguageHaveWorkedWith, Length: 2566, dtype: object

## 4. 프로그래밍 언어 데이터만 추출하기

### ■ 프로그래밍 언어의 빈도 구하기

- groupby() 메서드
- 집계한 데이터에서 size()메서드로 각 그룹의 크기를 확인
  - 값은 그룹 크기의 역순으로 정렬

In [12]:	<pre># 프로그래밍별 응답 수 구하기 size_by_languages = exploded_languages.groupby(exploded_languages).size()  # 데이터 빈도 역순으로 정렬 size_by_languages.sort_values(ascending=False, inplace=True) size_by_languages</pre>
Out [12]:	<pre>LanguageHaveWorkedWith Python                624 SQL                   440 ----- ( 중 략 ) ----- COBOL                   1 Erlang                  1 Name: LanguageHaveWorkedWith, dtype: int64</pre>

## 4. 프로그래밍 언어 데이터만 추출하기

### ■ 시각화 준비하기

- 현재 데이터를 데이터프레임으로 변환

In [13]:	<pre># 데이터프레임 만들기 위한 딕셔너리 만들기 frame = {'language': size_by_languages.index, 'count': size_by_languages. values}  # 데이터프레임 만들기 size_by_languages_df = pd.DataFrame(frame) size_by_languages_df.head()</pre>																																	
Out [13]:	<table><tr><th></th><th>language</th><th>count</th></tr><tr><td>0</td><td>Python</td><td>624</td></tr><tr><td>1</td><td>SQL</td><td>440</td></tr><tr><td>2</td><td>R</td><td>213</td></tr><tr><td>3</td><td>Bash/Shell</td><td>174</td></tr><tr><td>4</td><td>JavaScript</td><td>169</td></tr><tr><td>5</td><td>HTML/CSS</td><td>147</td></tr><tr><td>6</td><td>C++</td><td>113</td></tr><tr><td>7</td><td>Java</td><td>85</td></tr><tr><td>8</td><td>VBA</td><td>70</td></tr><tr><td>9</td><td>C</td><td>66</td></tr></table>		language	count	0	Python	624	1	SQL	440	2	R	213	3	Bash/Shell	174	4	JavaScript	169	5	HTML/CSS	147	6	C++	113	7	Java	85	8	VBA	70	9	C	66
	language	count																																
0	Python	624																																
1	SQL	440																																
2	R	213																																
3	Bash/Shell	174																																
4	JavaScript	169																																
5	HTML/CSS	147																																
6	C++	113																																
7	Java	85																																
8	VBA	70																																
9	C	66																																



## Section 04

### 워드클라우드와 트리맵

# 1. 워드클라우드

## ■ 워드클라우드(Wordcloud)

- 태그 클라우드(Tag cloud)라고도 불리는 그래프
  - 메타 데이터에서 얻어진 태그들을 분석
  - 중요도나 인기도 등을 고려해 시각적으로 늘어놓아 웹 사이트에 표시하는 용도로 주로 사용
  - ✓ 직관적으로 알아볼 수 있도록 태그의 글자 형태를 측정 기준에 따라 다르게 적용함



워드클라우드의 예

# 1. 워드클라우드

## ■ 라이브러리 탑재 및 예시 확인

- 매프랏립 라이브러리의 pyplot 모듈을 탑재
- Wordcloud 라이브러리 탑재

In [14]:	<pre># 매프랏립 라이브러리 탑재 import matplotlib.pyplot as plt  # Wordcloud 라이브러리 탑재 from wordcloud import WordCloud</pre>
----------	--

# 1. 워드클라우드

## ■ 라이브러리 탑재 및 예시 확인

- Wordcloud의 공식 레퍼런스 문서 방문
- ([https://amueller.github.io/word\\_cloud/auto\\_examples/index.html](https://amueller.github.io/word_cloud/auto_examples/index.html))
  - 여러 사례를 예제 코드와 함께 확인 가능
  - 레퍼런스 문서 하단에서 참고할 수 있는 소스 코드를 다운로드할 수도 있음

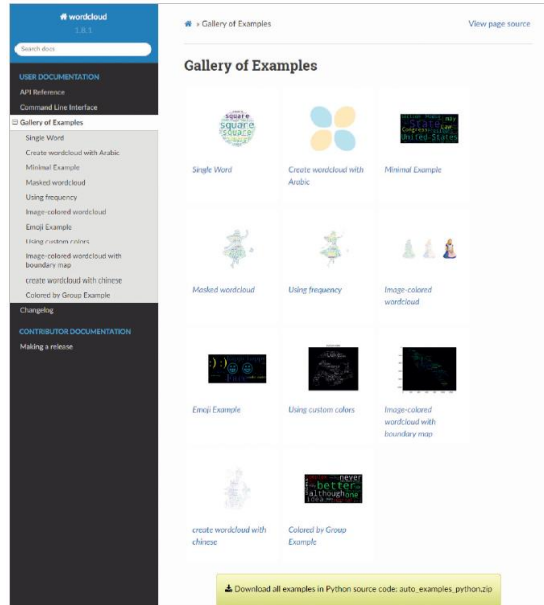


그림 9-10 워드클라우드 예시와 소스 코드

# 1. 워드클라우드

## ■ 워드클라우드 만들기

- Wordcloud 라이브러리에 데이터를 넣는 규칙
  - 프로그래밍 언어를 모두 나열한 전체 텍스트를 넣기
  - 항목이 프로그래밍 언어와 크기의 쌍으로 이루어진 딕셔너리를 준비
- to\_dict() 메서드 → 워드클라우드 생성에 필요한 형태로 변경

In [15]:	# 데이터프레임을 딕셔너리로 변경하기 size_by_languages.to_dict()
Out [15]:	{'Python': 624, 'SQL': 440, 'R': 213, 'Bash/Shell': 174, 'JavaScript': 169, ---- ( 중 략 ) ---- 'Elixir': 2, 'Crystal': 1, 'OCaml': 1, 'COBOL': 1, 'Erlang': 1}

# 1. 워드클라우드

## ■ 워드클라우드 만들기

- 워드 클라우드 배경색 흰색(white)
- generate\_from\_frequencies() 메서드 실행 → wordcloud 객체가 생성됨

In [16]:	<pre># 워드클라우드 만들기 wordcloud = WordCloud(background_color = 'white').generate_from_ frequencies(size_by_languages.to_dict()) wordcloud</pre>
Out [16]:	<pre>&lt;wordcloud.wordcloud.WordCloud at 0x167a11fe1c0&gt;</pre>



# 1. 워드클라우드

## ■ In [17]: 코드와 결과 분석하기

- **02** 워드클라우드를 그릴 코드와 실행 그래프의 가로, 세로 크기를 지정  
→ 그래프를 그릴 공간이 제한되어 있어 무한대로 커지지 않는으며, 작게 표현할 때 효과적임
- **03** wordcloud 객체를 이미지로 표현합니다.
- **04** 축 표기를 제거→ 워드클라우드에서는 좌표가 중요한 지표가 아님
- **05** 그래프를 출력
- **Out [17]:**의 워드클라우드를 보면 중앙의 Python 글자가 가장 큼  
→ 데이터 분석가로 분류된 개발자들이 파이썬을 가장 많이 사용한다는 의미임  
→ 다음으로 SQL, Bash/Shell, R 등이 크게 보이고 있음



## 2. 트리맵

### ■ 트리맵(Treemap)

- 큰 직사각형을 여러 개의 작은 직사각형으로 분할한 그래프
- 큰 범주에서 작은 범주로 나뉘지는 것을 효과적으로 보여줄 수 있음
  - 계층 구조나 계층(트리) 구조 데이터를 표시하는 데 적합함
  - 가장 큰 사각형은 왼쪽 상단에, 가장 작은 사각형은 오른쪽 하단에 표시됨
- ✓ 워드클라우드는 대략적인 빈도를 확인 가능하나 정확한 수치를 가늠하기엔 무리가 있음
- ✓ 정량적인 방법으로 빈도를 확인하려면 트리맵을 사용하는 것이 좋음

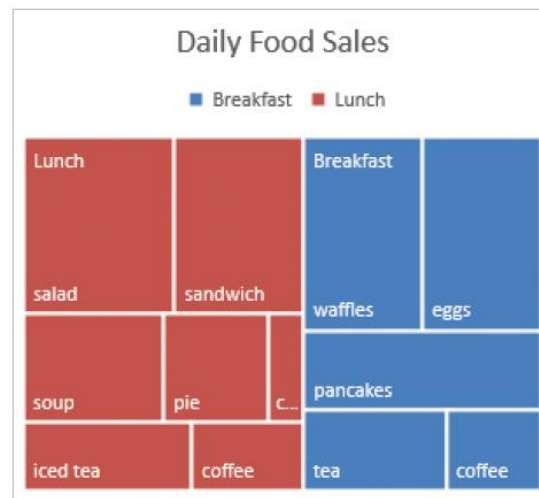



그림 9-11 트리맵의 예시

## 2. 트리맵

### ■ 트리맵 만들기

- 트리맵을 그리기 위해 plotly.express 모듈을 탑재
- `treemap()` 메서드 → 다음 소스 코드 실행

In [18]:	<pre># plotly.express 모듈 탑재 import plotly.express as px  # 트리맵 그리기 fig = px.treemap(size_by_languages_df, path=['language'], values='count') fig.show()</pre>
Out [18]:	

## 2. 트리맵

### ■ 트리맵 생성

- 마우스 포인터를 위치→ 각 상자의 이름과 크기를 확인할 수 있음



각 상자의 상세 값 확인하기

# **Section 05**

## **데이터 학습 및 시각화**

# 1. 데이터 2차 가공

## ■ 데이터 분석가들이 사용하는 프로그래밍 언어 추출하기

- 데이터 분석가들이 응답한 프로그래밍 언어에서 결손치를 제거
- 필터링을 하기 위해 리스트로 변환함

In [19]:	<pre># 결손치 제거 languages.dropna(inplace=True)  # 필터링을 위한 리스트로 변환 lang_list = languages.to_list()  # 일부 값 확인 lang_list[:10]</pre>
Out [19]:	<pre>[['HTML/CSS', 'Python', 'Ruby', 'SQL', 'VBA'], ['Bash/Shell', 'C++', 'Python', 'SQL'], ['Bash/Shell', 'Python', 'SQL'], ['Python'], ['Elixir', 'Python', 'Rust', 'SQL', 'TypeScript'], ['Python'], ['Bash/Shell', 'C++', 'HTML/CSS',</pre>

# 1. 데이터 2차 가공

## ■ 데이터 분석가들이 사용하는 프로그래밍 언어 추출하기

```
Out [19]: 'JavaScript',  
          'Python',  
          'Rust',  
          'SQL',  
          'TypeScript'],  
          ['Python', 'SQL'],  
          ['Bash/Shell', 'Go', 'HTML/CSS', 'Python'],  
          ['Bash/Shell', 'Python', 'Rust']]
```

## ■ 추출 대상 프로그래밍 언어 선정

- 데이터 분석가가 되기 위해 배워야 할 프로그래밍 언어 → 구글에서 검색  
→ 그 중 가장 빈번하게 나오는 언어 10개를 선정

```
In [20]: # 추출 대상 프로그래밍 언어 선정  
target_langs = ['Python', 'R', 'SQL', 'MATLAB', 'Go', 'SAS',  
                'Scala', 'Julia', 'Java', 'JavaScript']
```

# 1. 데이터 2차 가공

## ■ 리스트 추출하기

- 전체 프로그래밍 언어 리스트 → 선택한 10개 프로그래밍 언어만 남기기
- 추출한 리스트에 파이썬이 포함되는 경우에만 학습을 진행
- 다음 소스 코드 실행

In [21]:	<pre>#필터링 완료된 프로그래밍 언어를 담기 위한 리스트 생성 revised_lang_list = []  # 필터링을 위해 전체 리스트 순회 for lang in lang_list:     filtered = [x for x in lang if x in target_langs]     revised_lang_list.append(filtered)  # 필터링 결과 확인 revised_lang_list</pre>	<p>← 추출 대상 프로그래밍 언어만 필터링</p> <p>→ 학습 대상 리스트에 추가</p>
----------	---	---

# 1. 데이터 2차 가공

## ■ In [21]: 코드 실행 결과

```
Out [21]: [['Python', 'SQL'],  
           ['Python', 'SQL'],  
           ['Python', 'SQL'],  
           ['Python'],  
           ['Python', 'SQL'],  
  
           -----( 중 략 )-----  
  
           ['Python'],  
           ['Python', 'SQL'],  
           ['R'],  
           ['Java', 'Python', 'Scala', 'SQL'],  
           ['Julia', 'R', 'SQL'],  
           ['JavaScript', 'Python', 'SQL']]
```



# 1. 데이터 2차 가공

## ■ 데이터 전처리

- 현재 리스트로 되어 있는 정보를 2차원 행렬로 변경  
→ 2차원 행렬에는 각 프로그래밍 언어의 출현 여부를 표기함

In [22]:	<pre># 전처리 라이브러리 탑재 from mlxtend.preprocessing import TransactionEncoder  # 전처리기 생성 te = TransactionEncoder()  # 전처리 수행 te_ary = te.fit(revised_lang_list).transform(revised_lang_list) te_ary</pre>
Out [22]:	<pre>array([[False, False, False, ..., False, True, False],        [False, False, False, ..., False, True, False],        [False, False, False, ..., False, True, False],        ...,        [False, False, False, ..., False, True, False],        [False, True, False, ..., False, True, True],        [False, False, True, ..., False, True, False]])</pre>

# 1. 데이터 2차 가공

## ■ 전처리 결과를 데이터프레임으로 변환하기

- 실행 결과의 첫 번째 행

→ Python과 SQL의 값은 True이고 나머지는 False임

→ 이 정보는 In [21]:에서 리스트 항목으로 있었던 ['Python', 'SQL']과 일치하는 값임

✓ 앞으로 데이터분석 알고리즘을 수행할 때 이런 식의 데이터 변환을 자주 하게 될 것임

In [23]:	<pre># 전처리 결과를 데이터프레임으로 변환 new_languages = pd.DataFrame(te_ary,                               columns=te.columns_) new_languages</pre>																																																																																																																																				
Out [23]:	<table><tr><th></th><th>Go</th><th>Java</th><th>JavaScript</th><th>Julia</th><th>MATLAB</th><th>Python</th><th>R</th><th>SAS</th><th>SQL</th><th>Scala</th></tr><tr><td>0</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr><tr><td>1</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr><tr><td>2</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr><tr><td>3</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>False</td><td>False</td></tr><tr><td>4</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>724</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr><tr><td>725</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>False</td></tr><tr><td>726</td><td>False</td><td>True</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>True</td></tr><tr><td>727</td><td>False</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td><td>True</td><td>False</td></tr><tr><td>728</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td><td>False</td><td>True</td><td>False</td></tr></table> <p>729 rows × 10 columns</p>		Go	Java	JavaScript	Julia	MATLAB	Python	R	SAS	SQL	Scala	0	False	False	False	False	False	True	False	False	True	False	1	False	False	False	False	False	True	False	False	True	False	2	False	False	False	False	False	True	False	False	True	False	3	False	False	False	False	False	True	False	False	False	False	4	False	False	False	False	False	True	False	False	True	False	...	...	...	...	...	...	...	...	...	...	...	724	False	False	False	False	False	True	False	False	True	False	725	False	False	False	False	False	False	True	False	False	False	726	False	True	False	False	False	True	False	False	True	True	727	False	False	False	True	False	False	True	False	True	False	728	False	False	True	False	False	True	False	False	True	False
	Go	Java	JavaScript	Julia	MATLAB	Python	R	SAS	SQL	Scala																																																																																																																											
0	False	False	False	False	False	True	False	False	True	False																																																																																																																											
1	False	False	False	False	False	True	False	False	True	False																																																																																																																											
2	False	False	False	False	False	True	False	False	True	False																																																																																																																											
3	False	False	False	False	False	True	False	False	False	False																																																																																																																											
4	False	False	False	False	False	True	False	False	True	False																																																																																																																											
...	...	...	...	...	...	...	...	...	...	...																																																																																																																											
724	False	False	False	False	False	True	False	False	True	False																																																																																																																											
725	False	False	False	False	False	False	True	False	False	False																																																																																																																											
726	False	True	False	False	False	True	False	False	True	True																																																																																																																											
727	False	False	False	True	False	False	True	False	True	False																																																																																																																											
728	False	False	True	False	False	True	False	False	True	False																																																																																																																											

## 2. 데이터 학습하기

### ■ 지지도 구하기

- 학습을 위해 필요한 라이브러리를 탑재
  - 어프라이어리 알고리즘을 사용해 각 항목의 출현 빈도를 의미하는 지지도 구하기
- 인수 min\_support → 지지를 구할 최소값을 지정
- use\_colnames → 데이터 프레임에 있는 열 이름을 그대로 사용 → True로 설정
- verbose=1은 조합의 건수를 출력함

In [24]:	<pre># 학습 알고리즘 탑재 from mlxtend.frequent_patterns import apriori, association_rules  # 지지도 구하기 freq_items = apriori(new_languages, min_support=0.01, use_colnames=True, verbose=1)</pre>
	Processing 55 combinations   Sampling itemset size 54

## 2. 데이터 학습하기

### ■ 지지도 구하기

- 학습 결과를 지지도 역순으로 상위 20건 확인하기

→ Python이 포함된 결과만 추출했기 때문에 제일 윗줄의 지지도는 1이 됨

In [25]:	# 학습 결과 확인하기 freq_items.sort_values(['support'], ascending=False).head(20)																																																															
Out [25]:	<table><tr><th></th><th>support</th><th>itemsets</th></tr><tr><td>5</td><td>0.855967</td><td>(Python)</td></tr><tr><td>8</td><td>0.603567</td><td>(SQL)</td></tr><tr><td>32</td><td>0.517147</td><td>(Python, SQL)</td></tr><tr><td>6</td><td>0.292181</td><td>(R)</td></tr><tr><td>30</td><td>0.237311</td><td>(R, Python)</td></tr><tr><td>2</td><td>0.231824</td><td>(JavaScript)</td></tr><tr><td>35</td><td>0.213992</td><td>(R, SQL)</td></tr><tr><td>20</td><td>0.211248</td><td>(JavaScript, Python)</td></tr><tr><td>64</td><td>0.182442</td><td>(R, Python, SQL)</td></tr><tr><td>22</td><td>0.160494</td><td>(JavaScript, SQL)</td></tr><tr><td>53</td><td>0.148148</td><td>(JavaScript, Python, SQL)</td></tr><tr><td>1</td><td>0.116598</td><td>(Java)</td></tr><tr><td>14</td><td>0.098765</td><td>(Java, Python)</td></tr><tr><td>16</td><td>0.076818</td><td>(Java, SQL)</td></tr><tr><td>45</td><td>0.065844</td><td>(Java, Python, SQL)</td></tr><tr><td>13</td><td>0.064472</td><td>(Java, JavaScript)</td></tr><tr><td>41</td><td>0.060357</td><td>(Java, JavaScript, Python)</td></tr><tr><td>4</td><td>0.056241</td><td>(MATLAB)</td></tr><tr><td>27</td><td>0.056241</td><td>(MATLAB, Python)</td></tr><tr><td>3</td><td>0.053498</td><td>(Julia)</td></tr></table>		support	itemsets	5	0.855967	(Python)	8	0.603567	(SQL)	32	0.517147	(Python, SQL)	6	0.292181	(R)	30	0.237311	(R, Python)	2	0.231824	(JavaScript)	35	0.213992	(R, SQL)	20	0.211248	(JavaScript, Python)	64	0.182442	(R, Python, SQL)	22	0.160494	(JavaScript, SQL)	53	0.148148	(JavaScript, Python, SQL)	1	0.116598	(Java)	14	0.098765	(Java, Python)	16	0.076818	(Java, SQL)	45	0.065844	(Java, Python, SQL)	13	0.064472	(Java, JavaScript)	41	0.060357	(Java, JavaScript, Python)	4	0.056241	(MATLAB)	27	0.056241	(MATLAB, Python)	3	0.053498	(Julia)
	support	itemsets																																																														
5	0.855967	(Python)																																																														
8	0.603567	(SQL)																																																														
32	0.517147	(Python, SQL)																																																														
6	0.292181	(R)																																																														
30	0.237311	(R, Python)																																																														
2	0.231824	(JavaScript)																																																														
35	0.213992	(R, SQL)																																																														
20	0.211248	(JavaScript, Python)																																																														
64	0.182442	(R, Python, SQL)																																																														
22	0.160494	(JavaScript, SQL)																																																														
53	0.148148	(JavaScript, Python, SQL)																																																														
1	0.116598	(Java)																																																														
14	0.098765	(Java, Python)																																																														
16	0.076818	(Java, SQL)																																																														
45	0.065844	(Java, Python, SQL)																																																														
13	0.064472	(Java, JavaScript)																																																														
41	0.060357	(Java, JavaScript, Python)																																																														
4	0.056241	(MATLAB)																																																														
27	0.056241	(MATLAB, Python)																																																														
3	0.053498	(Julia)																																																														

## 2. 데이터 학습하기

### ■ 신뢰도와 향상도 구하기

- 지지도 데이터프레임에 각 항목 집합의 항목 개수를 포함한 데이터프레임이 필요함
- 다음 소스 코드를 실행

→ `apply()` 메서드와 람다식을 사용 해 구한 값을 `length` 열에 추가한 것을 확인할 수 있음

In [26]:

# 항목 집합의 항목 개수 추가하기  
freq\_items['length'] = freq\_items['itemsets'].apply(lambda  
x: len(x))  
freq\_items

Out [26]:

	support	itemsets	length
0	0.030178	(Go)	1
1	0.116598	(Java)	1
2	0.231824	(JavaScript)	1
3	0.053498	(Julia)	1
4	0.056241	(MATLAB)	1
...	...	...	...
78	0.017833	(R, Python, SQL, Julia)	4
79	0.019204	(R, MATLAB, Python, SQL)	4
80	0.021948	(R, SAS, Python, SQL)	4
81	0.012346	(R, Scala, Python, SQL)	4
82	0.013717	(Java, SQL, JavaScript, Python, R)	5

83 rows × 3 columns

## 2. 데이터 학습하기

### ■ 신뢰도와 향상도 구하기

- `association_rules()` 메서드를 호출
- `min_threshold(최솟값 임계치)`는 지표가 포함할 수 있는 최솟값을 지정함  
→ 최솟값보다 낮아지는 순간부터는 계산을 더 진행하지 않음

In [27]:	# 신뢰도와 향상도 구하기 rules = association_rules(freq_items, min_threshold=0.01) rules																																																																																																																																
Out [27]:	<table><tr><th></th><th>antecedents</th><th>consequents</th><th>antecedent support</th><th>consequent support</th><th>support</th><th>confidence</th><th>lift</th><th>leverage</th><th>conviction</th></tr><tr><td>0</td><td>(JavaScript)</td><td>(Go)</td><td>0.231824</td><td>0.030178</td><td>0.013717</td><td>0.059172</td><td>1.960732</td><td>0.006721</td><td>1.030817</td></tr><tr><td>1</td><td>(Go)</td><td>(JavaScript)</td><td>0.030178</td><td>0.231824</td><td>0.013717</td><td>0.454545</td><td>1.960732</td><td>0.006721</td><td>1.408322</td></tr><tr><td>2</td><td>(Python)</td><td>(Go)</td><td>0.855967</td><td>0.030178</td><td>0.028807</td><td>0.033654</td><td>1.115166</td><td>0.002975</td><td>1.003597</td></tr><tr><td>3</td><td>(Go)</td><td>(Python)</td><td>0.030178</td><td>0.855967</td><td>0.028807</td><td>0.954545</td><td>1.115166</td><td>0.002975</td><td>3.168724</td></tr><tr><td>4</td><td>(SQL)</td><td>(Go)</td><td>0.603567</td><td>0.030178</td><td>0.017833</td><td>0.029545</td><td>0.979029</td><td>-0.000382</td><td>0.999348</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>437</td><td>(Java)</td><td>(R, JavaScript, Python, SQL)</td><td>0.116598</td><td>0.048011</td><td>0.013717</td><td>0.117647</td><td>2.450420</td><td>0.008119</td><td>1.078921</td></tr><tr><td>438</td><td>(SQL)</td><td>(R, Java, JavaScript, Python)</td><td>0.603567</td><td>0.013717</td><td>0.013717</td><td>0.022727</td><td>1.656818</td><td>0.005438</td><td>1.009219</td></tr><tr><td>439</td><td>(JavaScript)</td><td>(R, Java, Python, SQL)</td><td>0.231824</td><td>0.016461</td><td>0.013717</td><td>0.059172</td><td>3.594675</td><td>0.009901</td><td>1.045397</td></tr><tr><td>440</td><td>(Python)</td><td>(Java, JavaScript, R, SQL)</td><td>0.855967</td><td>0.013717</td><td>0.013717</td><td>0.016026</td><td>1.168269</td><td>0.001976</td><td>1.002346</td></tr><tr><td>441</td><td>(R)</td><td>(Java, JavaScript, Python, SQL)</td><td>0.292181</td><td>0.043896</td><td>0.013717</td><td>0.046948</td><td>1.069542</td><td>0.000892</td><td>1.003203</td></tr></table> 442 rows × 9 columns										antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	0	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817	1	(Go)	(JavaScript)	0.030178	0.231824	0.013717	0.454545	1.960732	0.006721	1.408322	2	(Python)	(Go)	0.855967	0.030178	0.028807	0.033654	1.115166	0.002975	1.003597	3	(Go)	(Python)	0.030178	0.855967	0.028807	0.954545	1.115166	0.002975	3.168724	4	(SQL)	(Go)	0.603567	0.030178	0.017833	0.029545	0.979029	-0.000382	0.999348	...	...	...	...	...	...	...	...	...	...	437	(Java)	(R, JavaScript, Python, SQL)	0.116598	0.048011	0.013717	0.117647	2.450420	0.008119	1.078921	438	(SQL)	(R, Java, JavaScript, Python)	0.603567	0.013717	0.013717	0.022727	1.656818	0.005438	1.009219	439	(JavaScript)	(R, Java, Python, SQL)	0.231824	0.016461	0.013717	0.059172	3.594675	0.009901	1.045397	440	(Python)	(Java, JavaScript, R, SQL)	0.855967	0.013717	0.013717	0.016026	1.168269	0.001976	1.002346	441	(R)	(Java, JavaScript, Python, SQL)	0.292181	0.043896	0.013717	0.046948	1.069542	0.000892	1.003203
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction																																																																																																																								
0	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817																																																																																																																								
1	(Go)	(JavaScript)	0.030178	0.231824	0.013717	0.454545	1.960732	0.006721	1.408322																																																																																																																								
2	(Python)	(Go)	0.855967	0.030178	0.028807	0.033654	1.115166	0.002975	1.003597																																																																																																																								
3	(Go)	(Python)	0.030178	0.855967	0.028807	0.954545	1.115166	0.002975	3.168724																																																																																																																								
4	(SQL)	(Go)	0.603567	0.030178	0.017833	0.029545	0.979029	-0.000382	0.999348																																																																																																																								
...	...	...	...	...	...	...	...	...	...																																																																																																																								
437	(Java)	(R, JavaScript, Python, SQL)	0.116598	0.048011	0.013717	0.117647	2.450420	0.008119	1.078921																																																																																																																								
438	(SQL)	(R, Java, JavaScript, Python)	0.603567	0.013717	0.013717	0.022727	1.656818	0.005438	1.009219																																																																																																																								
439	(JavaScript)	(R, Java, Python, SQL)	0.231824	0.016461	0.013717	0.059172	3.594675	0.009901	1.045397																																																																																																																								
440	(Python)	(Java, JavaScript, R, SQL)	0.855967	0.013717	0.013717	0.016026	1.168269	0.001976	1.002346																																																																																																																								
441	(R)	(Java, JavaScript, Python, SQL)	0.292181	0.043896	0.013717	0.046948	1.069542	0.000892	1.003203																																																																																																																								

## 2. 데이터 학습하기

### ■ In [27]: 코드 실행 결과 확인하기

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817

- 열의 값
  - antecedents: 조건이 되는 프로그래밍 언어 리스트
  - consequents: 결과가 되는 프로그래밍 언어 리스트
  - support: 지지도
  - confidence: 신뢰도
  - lift: 향상도

## 2. 데이터 학습하기

### ■ In [27]: 코드 실행 결과 확인하기

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817

- 실행 결과의 첫 번째 행을 해석한 결과
  - JavaScript를 다룰 줄 아는 데이터 분석가가 Go를 함께 다룰 가능성 즉, 전체 데이터에서 JavaScript와 Go를 함께 다룰 가능성(지지도)은 약 1%(0.013717)
  - JavaScript를 다루는 데이터 분석가가 Go를 함께 다룰 가능성(신뢰도)은 6%(0.059172)
  - 전체 데이터 분석가가 Go를 다룰 가능성에 비해서, JavaScript를 다루는 데이터 분석가가 Go를 다룰 가능성(향상도)은 약 1.96배(1.960732) 높음



### 3. 산점도 그래프

#### ■ 산점도(산포도) 그래프

- 두 변수의 관계를 알아보는 데 유용한 그래프
- x축에 한 변수를, y 축에 다른 변수를 설정
- 각 변수의 값을 나타내는 점을 찍어 두 변수의 관계를 파악함

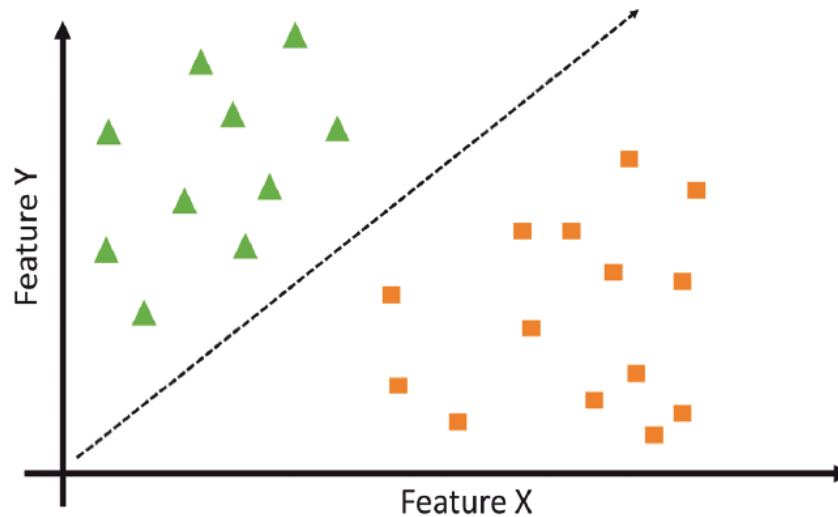


그림 9-13 산점도 그래프의 예시

### 3. 산점도 그래프

#### ■ 산점도 그래프 그리기

- `px.scatter()` 메서드 : 그래프를 그리기 위한 데이터를 설정
  - x축에는 지지도를, y축에는 신뢰도를 설정
  - 향상도는 점의 크기와 색으로 구분
- `update_layout()` 메서드
  - 산점도 그래프의 데이터를 설정하고, 반환한 fig 객체에 사용
  - 그래프에 표기할 다양한 값을 설정
    - ✓ x축과 y축에 표기할 라벨, 제목에 보일 내용, 글꼴 이름과 색 등

### 3. 산점도 그래프

#### ■ 산점도 그래프 그리기

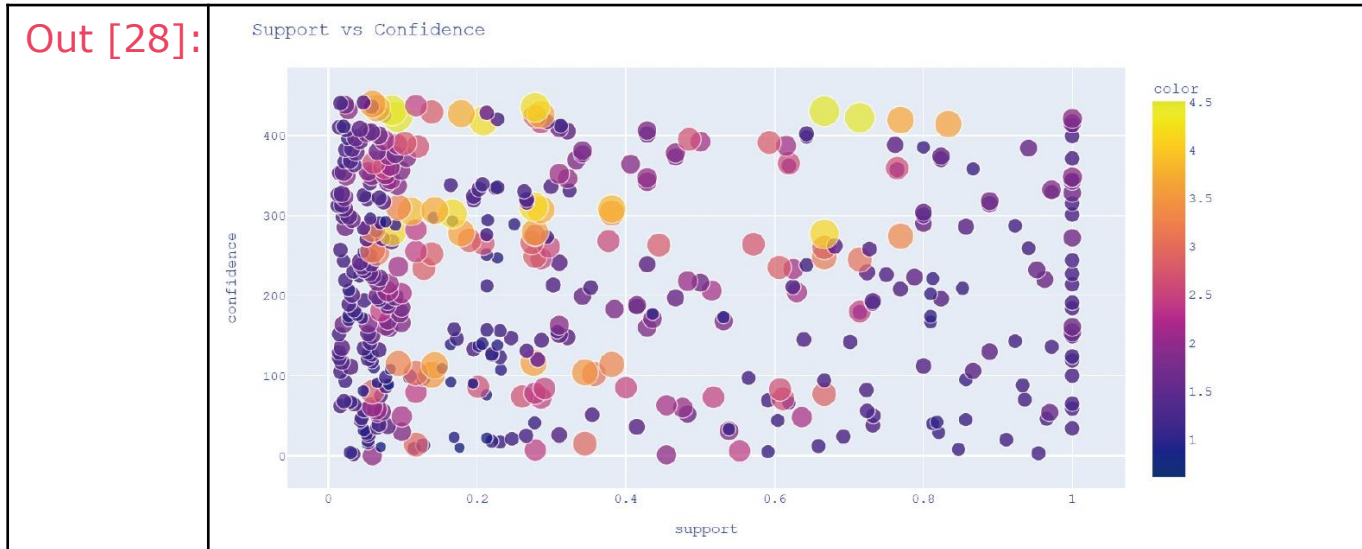
```
In [28]: # 산점도 그래프 그리기
fig=px.scatter(
    rules['support'],
    rules['confidence'],
    size=rules['lift'],
    color=rules['lift']
)

# 그래프 레이아웃 설정
fig.update_layout(
    xaxis_title='support',
    yaxis_title='confidence',
    font_family='Courier New',
    font_color='blue',
    title_font_family='Times New Roman',
    title_font_color='red',
    title=('Support vs Confidence')
)

# 그래프 출력
fig.show()
```

### 3. 산점도 그래프

#### ■ In [28]: 코드 실행 결과

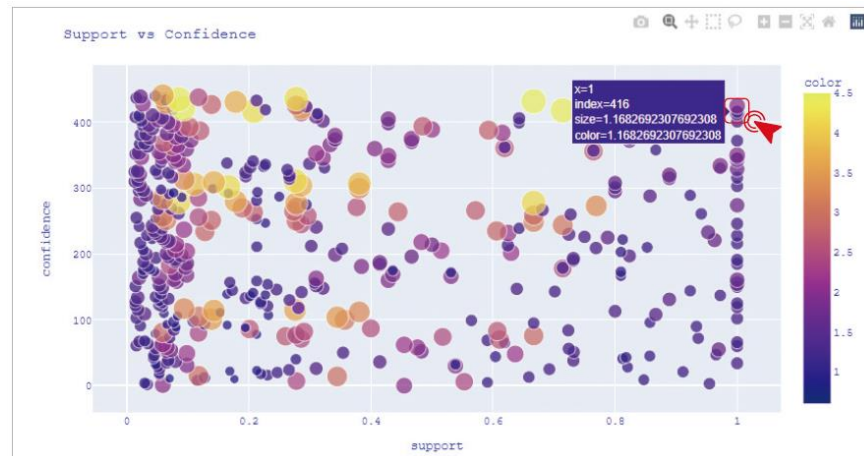


- 점의 위치는 지지도와 신뢰도에 따라 결정됨
- 왼쪽에 몰려 있는 값이 많음 → 지지도가 낮은 값이 많다는 것을 알 수 있음  
→ 프로그래밍 언어가 조합으로 동시에 나타나는 경우가 적다는 의미임
- 색이 밝고 큰 원 → 높은 향상도를 의미함  
→ 전반적으로 신뢰도가 높은 값의 향상도가 높은 경향을 보이고 있음

### 3. 산점도 그래프

#### ■ In [28]: 코드 실행 결과

- 그래프 오른쪽에 한 줄로 늘어선 데이터 → 지지도가 1인 데이터
  - 전체 건수에 조건과 결과 프로그래밍 언어가 모두 나타난다는 의미임
  - scatter 함수 안에서 값이 너무 물리는 것을 방지하기 위해 자체적으로 x축 데이터를 0과 1 사이에 정규화(Normalization)한 것으로 보임
- [그림] 같이 마우스를 위치시키면 조건과 결과 프로그래밍 언어를 확인할 수 있도록 학습 결과 데이터프레임에 값을 추가



산점도 그래프에서 특정 데이터의 값을 마우스 호버로 확인하기

### 3. 산점도 그래프

#### ■ 산점도 그래프 수정하기

- Out [29]: antecedents 열과 consequents 열의 데이터가 괄호로 묶여 있음
  - 일반 문자열이 아닌 파이썬의 기본 데이터 타입인 frozenset임
  - 그래프 출력을 위해 두 열의 frozenset을 문자열로 변경한 열을 추가

```
In [29]: # frozenset 데이터를 문자열로 변환하여 열 추가
rules['antecedents_str'] = rules['antecedents'].apply(lambda x: ','.join(list(x)))
rules['consequents_str'] = rules['consequents'].apply(lambda x: ','.join(list(x)))
rules
```

```
Out [29]:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents_str	consequents_str
0	(Go)	(JavaScript)	0.030178	0.231824	0.013717	0.454545	1.960732	0.006721	1.408322	Go	JavaScript
1	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817	JavaScript	Go
2	(Python)	(Go)	0.855967	0.030178	0.028807	0.033654	1.115166	0.002975	1.003597	Python	Go
3	(Go)	(Python)	0.030178	0.855967	0.028807	0.954545	1.115166	0.002975	3.168724	Go	Python
4	(SQL)	(Go)	0.603567	0.030178	0.017833	0.029545	0.979029	-0.000382	0.999348	SQL	Go
...	...	...	...	...	...	...	...	...	...	...	...
437	(Python)	(R, Java, SQL, JavaScript)	0.855967	0.013717	0.013717	0.016026	1.168269	0.001976	1.002346	Python	R,Java,SQL,JavaScript
438	(Java)	(Python, SQL, R, JavaScript)	0.116596	0.048011	0.013717	0.117647	2.450420	0.008119	1.078921	Java	Python,SQL,R,JavaScript
439	(SQL)	(Python, Java, R, JavaScript)	0.603567	0.013717	0.013717	0.022727	1.656818	0.005438	1.009219	SQL	Python,Java,R,JavaScript
440	(R)	(Python, Java, SQL, JavaScript)	0.292181	0.043896	0.013717	0.046948	1.069542	0.000892	1.003203	R	Python,Java,SQL,JavaScript
441	(JavaScript)	(R, Python, Java, SQL)	0.231824	0.016461	0.013717	0.059172	3.594675	0.009901	1.045397	JavaScript	R,Python,Java,SQL

42 rows × 11 columns

### 3. 산점도 그래프

#### ■ 산점도 그래프 수정하기

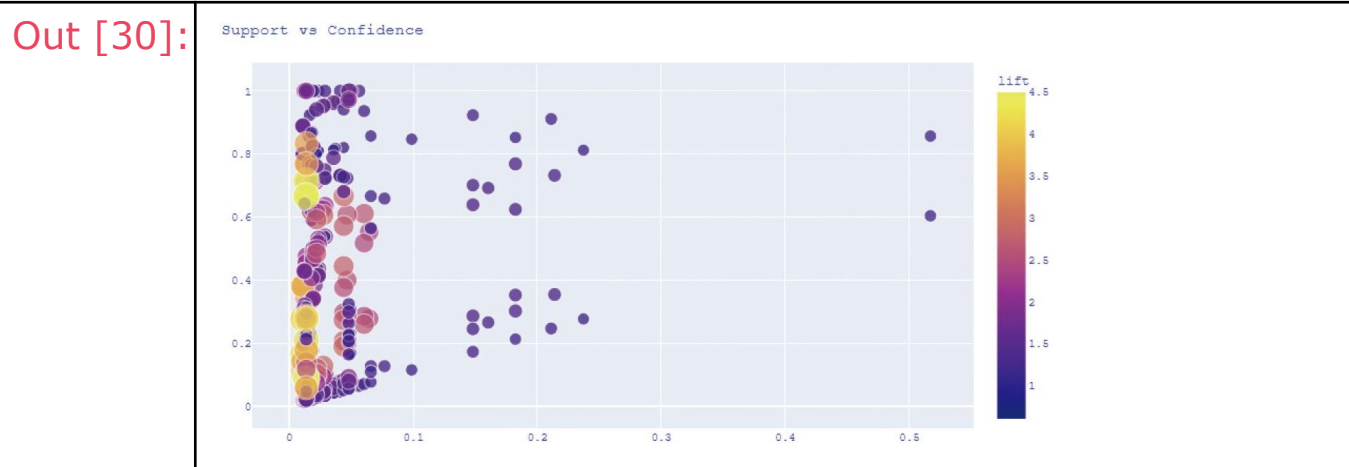
```
In [30]: # 산점도 그래프 그리기
fig=px.scatter(
    rules,
    x='support',
    y='confidence',
    size='lift',
    color='lift',
    hover_data=['antecedents_str', 'consequents_str']
)

# 그래프 레이아웃 설정
fig.update_layout(
    xaxis_title='support',
    yaxis_title='confidence',
    font_family='Courier New',
    font_color='blue',
    title_font_family='Times New Roman',
    title_font_color='red',
    title=('Support vs Confidence')
)

# 그래프 출력
fig.show()
```

### 3. 산점도 그래프

#### ■ In [30]: 코드 실행 결과



- x축에 데이터가 변경 없이 제대로 출력되는 것을 확인 할 수 있음
- 점의 분포가 사뭇 달라진 것이 확인 됨

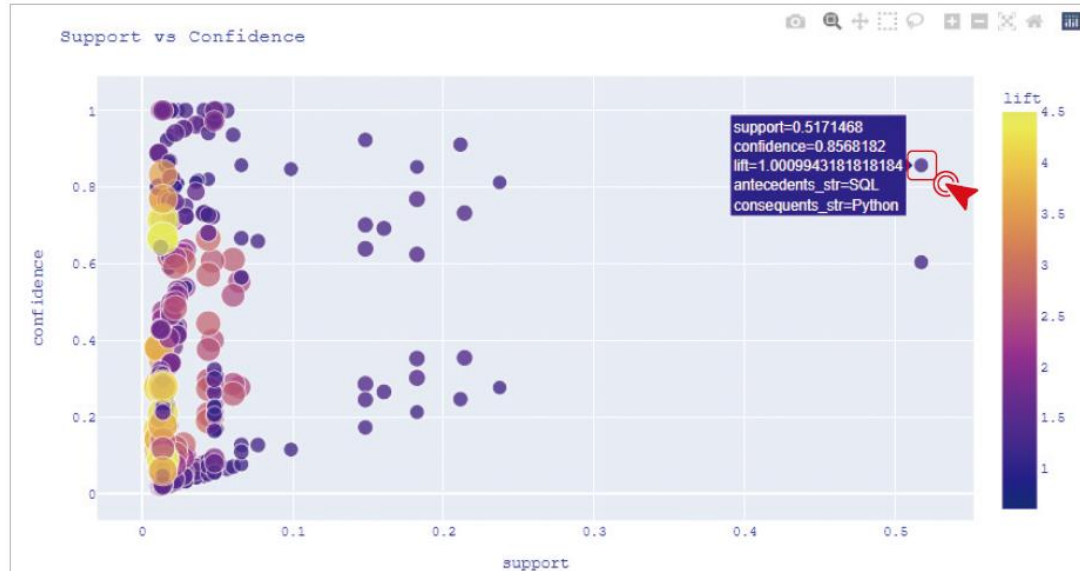


### 3. 산점도 그래프

#### ■ 산점도 그래프 데이터 확인하기

- [그림] 오른쪽 상단의 점

- SQL을 다룰 줄 아는 데이터 분석가는 파이썬을 다루는 경우가 가장 많다는 정보가 보임
- 지지도와 신뢰도는 높지만 향상도는 낮기 때문에 원의 크기가 작고 색이 어두움
- SQL을 다룰 줄 알기 때문에 파이썬을 다루게 될 확률은 그리 높지 않다고 해석할 수 있음

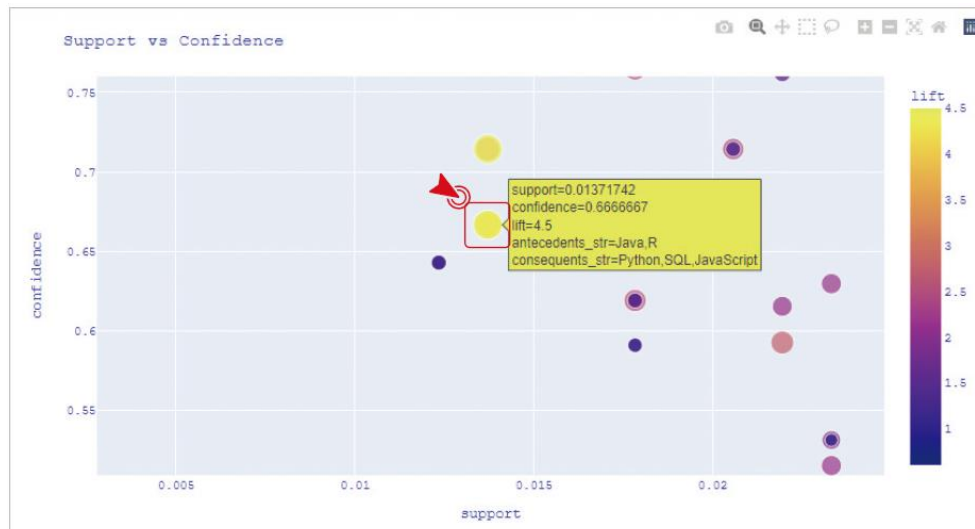


오른쪽 상단의 점 데이터 확인하기

### 3. 산점도 그래프

#### ■ 향상도가 높은 값 확인하기

- 향상도가 높은 값 왼쪽에 분포 → 지지도가 낮다는 의미
  - 전체 데이터 대비 나타날 확률이 낮지만, 조건 프로그래밍 언어를 다룰 줄 안다면 결과 프로그래밍 언어를 다루는 경향이 더 짙다는 것을 알 수 있음
- 가장 밝은 노란색 원이 있는 영역을 드래그 → 그래프 확대 후 정보 확인
  - Java와 R을 함께 다루는 사람은 파이썬, SQL, JavaScript를 모두 다룰 가능성이 높지만, 그런 데이터 분석가는 드물다는 뜻으로 해석 가능



향상도가 가장 높은 값 확대하여 확인하기

## 4. 히트맵

### ■ 산점도 그래프와 히트맵

- 파이썬 외에 어떤 언어를 배우면 좋을지 선택하는 데 도움이 될 정보
  - 산점도 그래프 : 전체 데이터의 경향을 파악하는 데 도움이 됨
  - 히트맵 : 파이썬을 다룰 줄 아는 데이터 분석가가 가장 많이 사용할 줄 아는 언어 값을 파악하기 쉬움

### ■ 히트맵 그리기

- 모든 언어를 x축과 y축에 나열하고 신뢰도 값을 설정
  - 그래프를 2차원 표에 생성

## 4. 히트맵

### ■ 항목이 하나인 값만 추출하기

- 데이터 프레임의 apply 메서드와 람다식을 활용

→ 하나의 프로그래밍 언어로 이루어진 연관 관계 분석 결과를 추출

In [31]:

# 항목이 하나인 값만 필터링하기  
rules\_for\_single = rules[rules.apply(lambda x: (len(x['antecedents']) ==  
1 and len(x['consequents']) == 1), axis=1)]  
rules\_for\_single

Out [31]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(JavaScript)	(Go)	0.231824	0.030178	0.013717	0.059172	1.960732	0.006721	1.030817
1	(Go)	(JavaScript)	0.030178	0.231824	0.013717	0.454545	1.960732	0.006721	1.408322
2	(Python)	(Go)	0.855967	0.030178	0.028807	0.033654	1.115166	0.002975	1.003597
3	(Go)	(Python)	0.030178	0.855967	0.028807	0.954545	1.115166	0.002975	3.168724
4	(SQL)	(Go)	0.603567	0.030178	0.017833	0.029545	0.979029	-0.000382	0.999348
5	(Go)	(SQL)	0.030178	0.603567	0.017833	0.590909	0.979029	-0.000382	0.969060
6	(Java)	(JavaScript)	0.116598	0.231824	0.064472	0.552941	2.385172	0.037442	1.718287
7	(JavaScript)	(Java)	0.231824	0.116598	0.064472	0.278107	2.385172	0.037442	1.223729
8	(Java)	(Python)	0.116598	0.855967	0.098765	0.847059	0.989593	-0.001039	0.941754
9	(Python)	(Java)	0.855967	0.116598	0.098765	0.115385	0.989593	-0.001039	0.998628
10	(Java)	(R)	0.116598	0.292181	0.020576	0.176471	0.603977	-0.013492	0.859494
11	(R)	(Java)	0.292181	0.116598	0.020576	0.070423	0.603977	-0.013492	0.950326
12	(Java)	(SQL)	0.116598	0.603567	0.076818	0.658824	1.091551	0.006443	1.161960
13	(SQL)	(Java)	0.603567	0.116598	0.076818	0.127273	1.091551	0.006443	1.012231
14	(Java)	(Scala)	0.116598	0.039781	0.013717	0.117647	2.957404	0.009079	1.088249
15	(Scala)	(Java)	0.039781	0.116598	0.013717	0.344828	2.957404	0.009079	1.348350
16	(JavaScript)	(Julia)	0.231824	0.053498	0.012346	0.053254	0.995448	-0.000056	0.999743
17	(Julia)	(JavaScript)	0.053498	0.231824	0.012346	0.230769	0.995448	-0.000056	0.998628
18	(MATLAB)	(JavaScript)	0.056241	0.231824	0.012346	0.219512	0.946890	-0.000692	0.984225
19	(JavaScript)	(MATLAB)	0.231824	0.056241	0.012346	0.053254	0.946890	-0.000692	0.996845

## 4. 히트맵

### ■ 신뢰도로 구성된 데이터프레임 생성하기

- frozenset을 리스트로 변환→ 각 프로그래밍 언어를 문자열로 변경
- 다음 소스 코드를 실행

```
In [32]: # 프로그래밍 언어 문자열로 추출
rules_for_single_df = pd.DataFrame(rules_for_single['antecedents'].
                                     apply(lambda x: list(x)[0]))
rules_for_single_df['consequents'] = rules_for_single['consequents'].
                                     apply(lambda x: list(x)[0])
rules_for_single_df['confidence'] = rules_for_single['confidence']

rules_for_single_df
```

```
Out [32]:
```

	antecedents	consequents	confidence
0	JavaScript	Go	0.069172
1	Go	JavaScript	0.454545
2	Python	Go	0.033654
3	Go	Python	0.954545
4	SQL	Go	0.029545
5	Go	SQL	0.590909
6	Java	JavaScript	0.552941
7	JavaScript	Java	0.278107
8	Java	Python	0.847059
9	Python	Java	0.115385
10	Java	R	0.176471
11	R	Java	0.070423
12	Java	SQL	0.658824
13	SQL	Java	0.127273
14	Java	Scala	0.117647
15	Scala	Java	0.344828
16	JavaScript	Julia	0.053254
17	Julia	JavaScript	0.230769
18	MATLAB	JavaScript	0.219512
19	JavaScript	MATLAB	0.053254

## 4. 히트맵

### ■ 히트맵에서 사용할 데이터프레임 생성하기

- 다음 소스 코드를 실행

```
In [33]: # 히트맵용 데이터프레임 생성, 색인은 대상 프로그래밍 언어로 설정
revised_rules_df = pd.DataFrame(index=target_langs)

# 대상 프로그래밍 언어 순회
for col in target_langs:

    # 열 설정을 위한 빈 리스트 생성
    col_conf = []
    for row in target_langs:
        confidence = rules_for_single_df[rules_for_single_
            df['antecedents'] == row][rules_for_single_
            df['consequents'] == col]['confidence']

        if len(confidence.values) > 0:
            col_conf.append(confidence.values[0])
        else:
            col_conf.append(0)
```

신뢰도 확보. 조건 언어는 줄,  
결과 언어는 행으로 설정

신뢰도 값이 존재하는  
경우에만 행에 값 추가

그렇지 않은 경우 0 설정

## 4. 히트맵

### ■ 히트맵에서 사용할 데이터프레임 생성하기

```
# 완성된 열을 데이터프레임에 추가
revised_rules_df[col] = col_conf

# 최종 데이터프레임 값 확인
revised_rules_df
```

Out [33]:

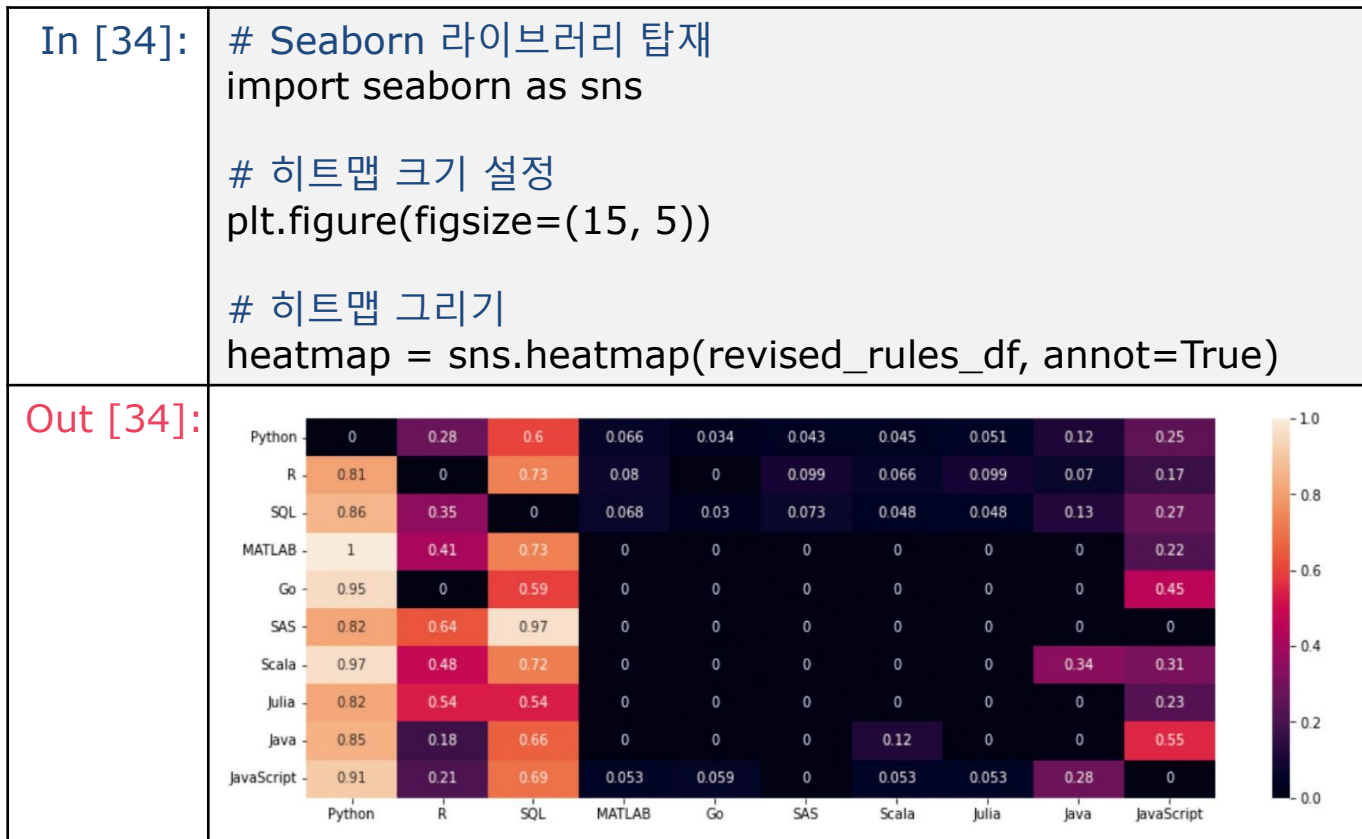
	Python	R	SQL	MATLAB	Go	SAS	Scala	Julia	Java	JavaScript
Python	0.000000	0.277244	0.604167	0.065705	0.033654	0.043269	0.044872	0.051282	0.115385	0.246795
R	0.812207	0.000000	0.732394	0.079812	0.000000	0.098592	0.065728	0.098592	0.070423	0.169014
SQL	0.856818	0.354545	0.000000	0.068182	0.029545	0.072727	0.047727	0.047727	0.127273	0.265909
MATLAB	1.000000	0.414634	0.731707	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.219512
Go	0.954545	0.000000	0.590909	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.454545
SAS	0.818182	0.636364	0.969697	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Scala	0.965517	0.482759	0.724138	0.000000	0.000000	0.000000	0.000000	0.000000	0.344828	0.310345
Julia	0.820513	0.538462	0.538462	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.230769
Java	0.847059	0.176471	0.658824	0.000000	0.000000	0.000000	0.117647	0.000000	0.000000	0.552941
JavaScript	0.911243	0.213018	0.692308	0.053254	0.059172	0.000000	0.053254	0.053254	0.278107	0.000000

## 4. 히트맵

### ■ 히트맵 그리기

- Seaborn 라이브러리의 heatmap() 메서드

→ 인수를 annot=True로 설정해 해당 값을 출력





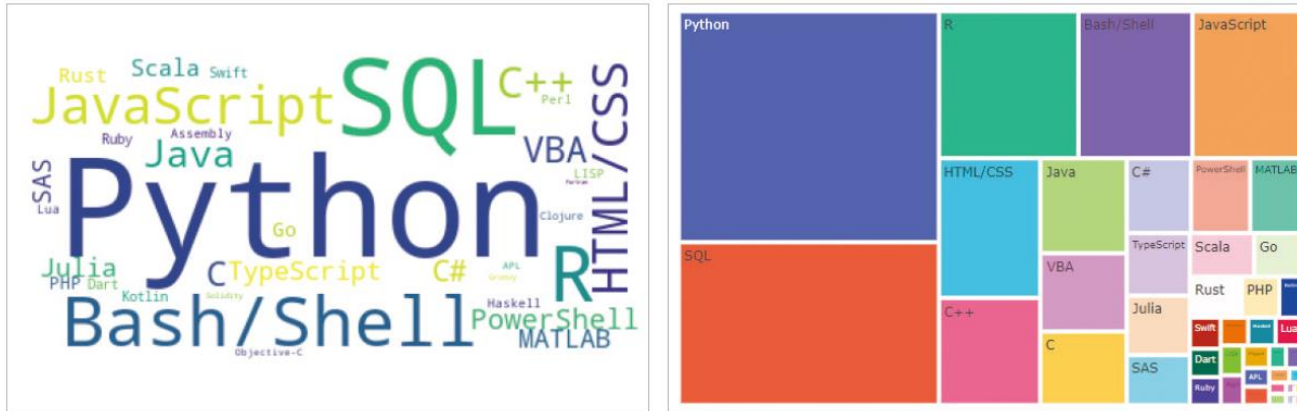
## 4. 히트맵

### ■ In [34]: 코드 실행 결과 확인하기

- 각 프로그래밍 언어가 x축과 y축에 설정, 신뢰도가 높을수록 밝은색으로 표기
  - 첫 번째 줄 → 파이썬을 다룰 줄 아는 사람이 가장 많이 다루는 언어는 SQL
  - 60%의 개발자가 파이썬과 SQL을 동시에 다루고 있음
  - 네 번째 줄 → MATLAB Python의 신뢰도는 100%
  - 설문 조사에 응답한 데이터 분석가 중 MATLAB을 다룰 줄 아는 사람은 모두 Python도 다룰 수 있다는 의미임
  - SQL도 무려 75%의 데이터 분석가가 다룰 줄 아는 것을 알 수 있음
  - ✓ 분석 결과를 바탕으로 신후에게 SQL을 추천할 수 있음

## 5. 마무리

### ■ 프로젝트 진행 결과



프로그래밍 언어 출현 빈도 출력 결과

- 데이터 분석가와 관련 있는 개발자 타입을 추출
  - 이들이 주로 사용하는 프로그래밍 언어를 워드클라우드와 트리맵으로 표현했음

## 5. 마무리

### 프로젝트 진행 결과



산점도와 히트맵 그래프 실행 결과

- 프로그래밍 언어 간의 연관 관계를 분석해 산점도 그래프로 표현
- 프로그래밍 언어의 신뢰도를 히트맵 그래프로 시각화
- 신호가 파이썬 외에 배워야 할 프로그래밍 언어 → SQL이라는 결과 도출