Pasakon Paingjai 622115028

Using the approach demonstrated up to page #22 of the slide and the data provided in the class, show that you can answer the following questions:

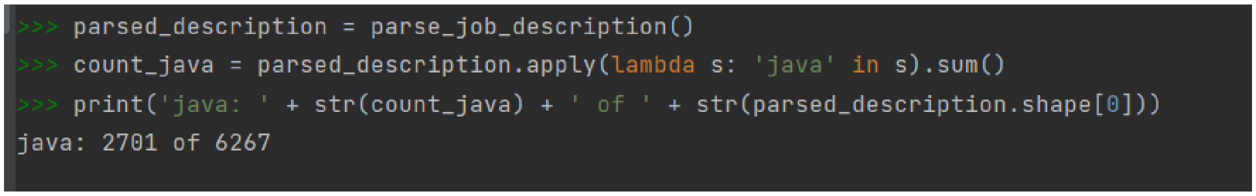
1. What DB should I learn after java?
2. Which DB is in demand alongside oracle?
3. What programing language is in demand alongside python?

- If possible, use the indexed matrices illustrated on page#21.

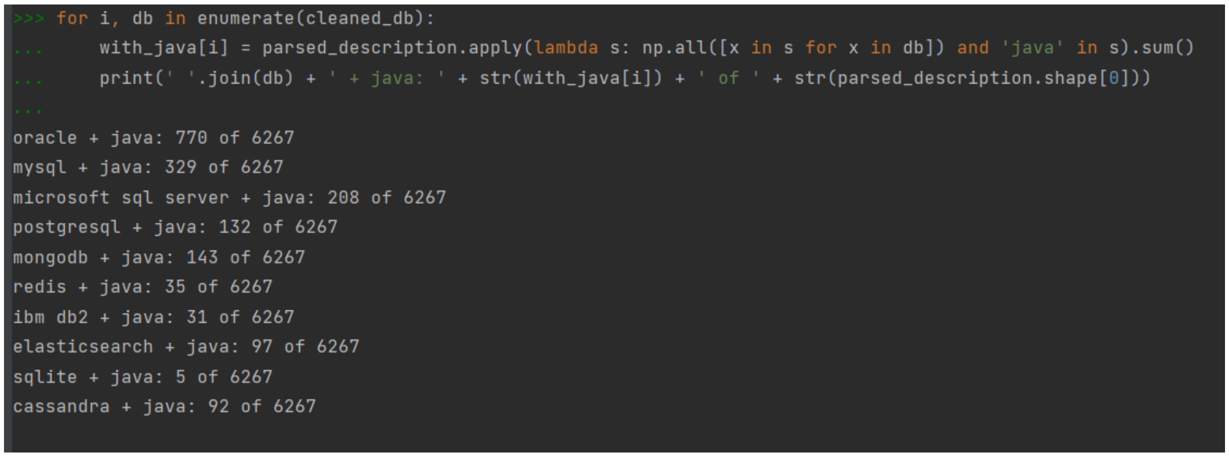
Please define the term "in demand" by yourself and explain why you defined it that way.

Demand: The top number of the list that is popular to search. The 5 of number that stays on the top, means higher demand. In any real-world, most of the total number appears on anything. It conveys as a highest of each category such as top 5 of highest views of YouTube. The 5 highest number of each database that is popular for each programing language that users want. That means users can predict if they use this database, it will be better than any database that is lower than the top of the list.

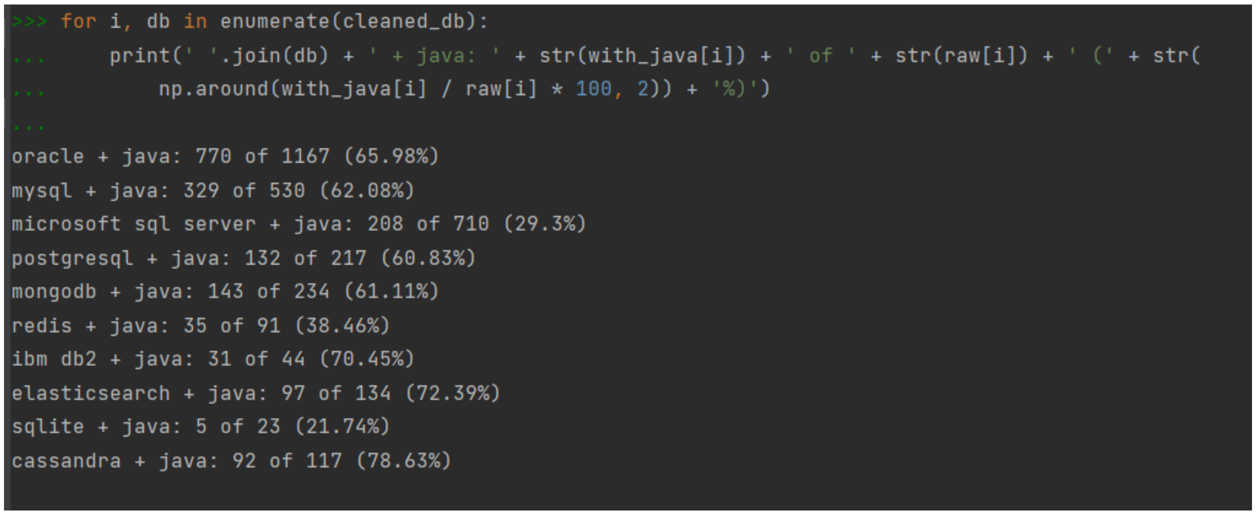
1.What DB should I learn after java?



```
>>> parsed_description = parse_job_description()
>>> count_java = parsed_description.apply(lambda s: 'java' in s).sum()
>>> print('java: ' + str(count_java) + ' of ' + str(parsed_description.shape[0]))
java: 2701 of 6267
```



```
>>> for i, db in enumerate(cleaned_db):
...     with_java[i] = parsed_description.apply(lambda s: np.all([x in s for x in db]) and 'java' in s).sum()
...     print(' '.join(db) + ' + java: ' + str(with_java[i]) + ' of ' + str(parsed_description.shape[0]))
...
oracle + java: 770 of 6267
mysql + java: 329 of 6267
microsoft sql server + java: 208 of 6267
postgresql + java: 132 of 6267
mongodb + java: 143 of 6267
redis + java: 35 of 6267
ibm db2 + java: 31 of 6267
elasticsearch + java: 97 of 6267
sqlite + java: 5 of 6267
cassandra + java: 92 of 6267
```



```
>>> for i, db in enumerate(cleaned_db):
...     print(' '.join(db) + ' + java: ' + str(with_java[i]) + ' of ' + str(raw[i]) + ' (' + str(
...         np.around(with_java[i] / raw[i] * 100, 2)) + '%)')
...
oracle + java: 770 of 1167 (65.98%)
mysql + java: 329 of 530 (62.08%)
microsoft sql server + java: 208 of 710 (29.3%)
postgresql + java: 132 of 217 (60.83%)
mongodb + java: 143 of 234 (61.11%)
redis + java: 35 of 91 (38.46%)
ibm db2 + java: 31 of 44 (70.45%)
elasticsearch + java: 97 of 134 (72.39%)
sqlite + java: 5 of 23 (21.74%)
cassandra + java: 92 of 117 (78.63%)
```

oracle + java: 770 of 1167 (65.98%)

mysql + java: 329 of 530 (62.08%)

microsoft sql server + java: 208 of 710 (29.3%)

postgresql + java: 132 of 217 (60.83%)

mongodb + java: 143 of 234 (61.11%)

redis + java: 35 of 91 (38.46%)

ibm db2 + java: 31 of 44 (70.45%)

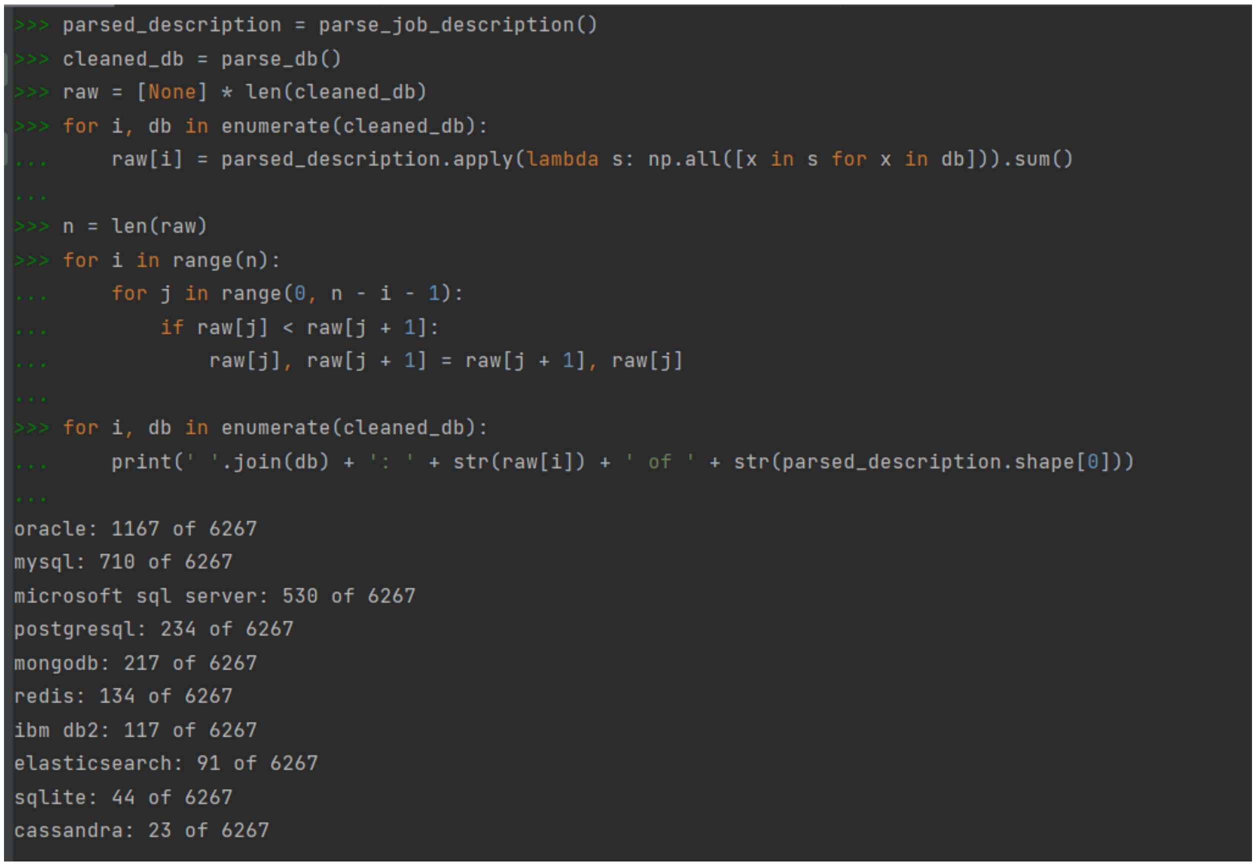
elasticsearch + java: 97 of 134 (72.39%)

sqlite + java: 5 of 23 (21.74%)

cassandra + java: 92 of 117 (78.63%)

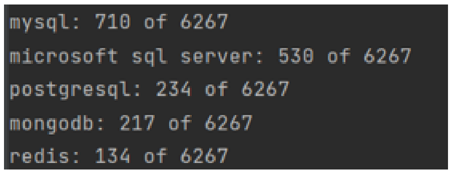
2.Which DB is in demand alongside oracle?

In the output sort the top 5 of the list that alongside oracle we will know the DB that should be alongside.



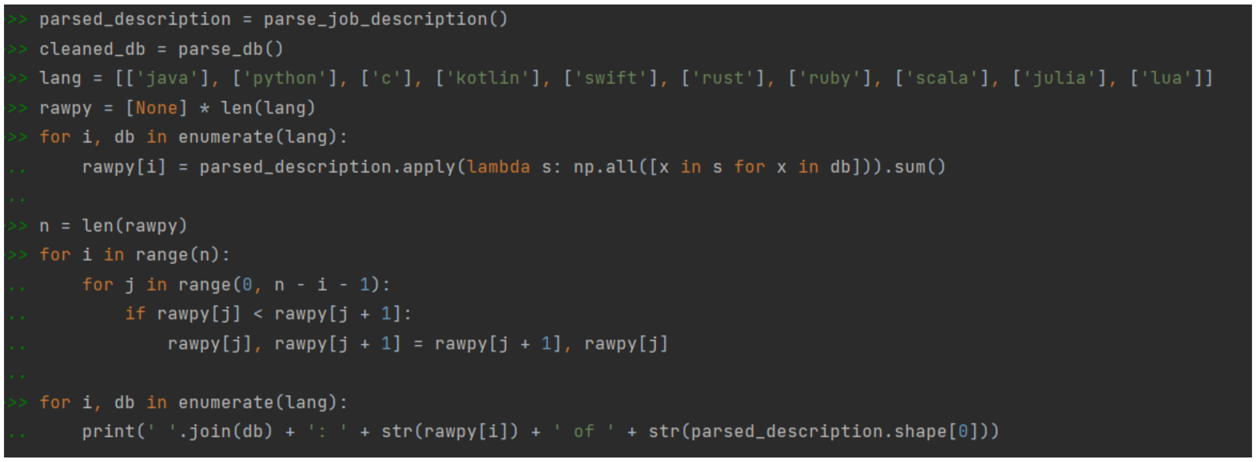
```
>>> parsed_description = parse_job_description()
>>> cleaned_db = parse_db()
>>> raw = [None] * len(cleaned_db)
>>> for i, db in enumerate(cleaned_db):
...     raw[i] = parsed_description.apply(lambda s: np.all([x in s for x in db])).sum()
...
>>> n = len(raw)
>>> for i in range(n):
...     for j in range(0, n - i - 1):
...         if raw[j] < raw[j + 1]:
...             raw[j], raw[j + 1] = raw[j + 1], raw[j]
...
>>> for i, db in enumerate(cleaned_db):
...     print(' '.join(db) + ': ' + str(raw[i]) + ' of ' + str(parsed_description.shape[0]))
...
oracle: 1167 of 6267
mysql: 710 of 6267
microsoft sql server: 530 of 6267
postgresql: 234 of 6267
mongodb: 217 of 6267
redis: 134 of 6267
ibm db2: 117 of 6267
elasticsearch: 91 of 6267
sqlite: 44 of 6267
cassandra: 23 of 6267
```

result

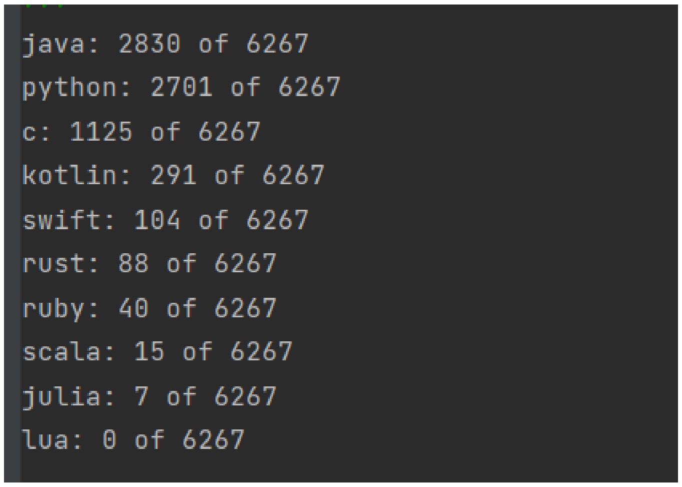


```
mysql: 710 of 6267
microsoft sql server: 530 of 6267
postgresql: 234 of 6267
mongodb: 217 of 6267
redis: 134 of 6267
```

3.What programing language is in demand alongside python?

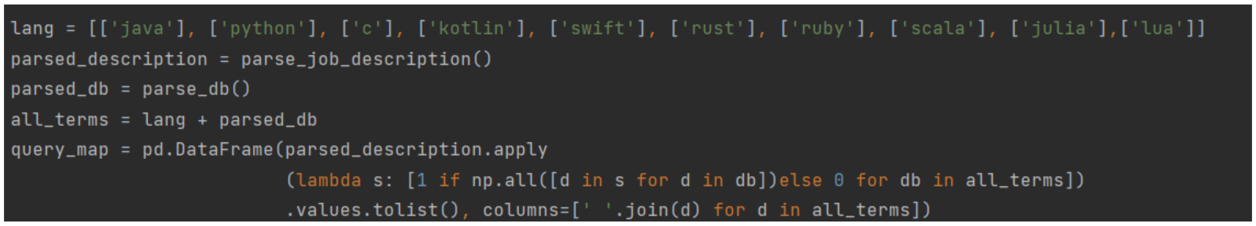


```
>> parsed_description = parse_job_description()
>> cleaned_db = parse_db()
>> lang = [['java'], ['python'], ['c'], ['kotlin'], ['swift'], ['rust'], ['ruby'], ['scala'], ['julia'], ['lua']]
>> rawpy = [None] * len(lang)
>> for i, db in enumerate(lang):
..     rawpy[i] = parsed_description.apply(lambda s: np.all([x in s for x in db])).sum()
..
>> n = len(rawpy)
>> for i in range(n):
..     for j in range(0, n - i - 1):
..         if rawpy[j] < rawpy[j + 1]:
..             rawpy[j], rawpy[j + 1] = rawpy[j + 1], rawpy[j]
..
>> for i, db in enumerate(lang):
..     print(' '.join(db) + ': ' + str(rawpy[i]) + ' of ' + str(parsed_description.shape[0]))
```



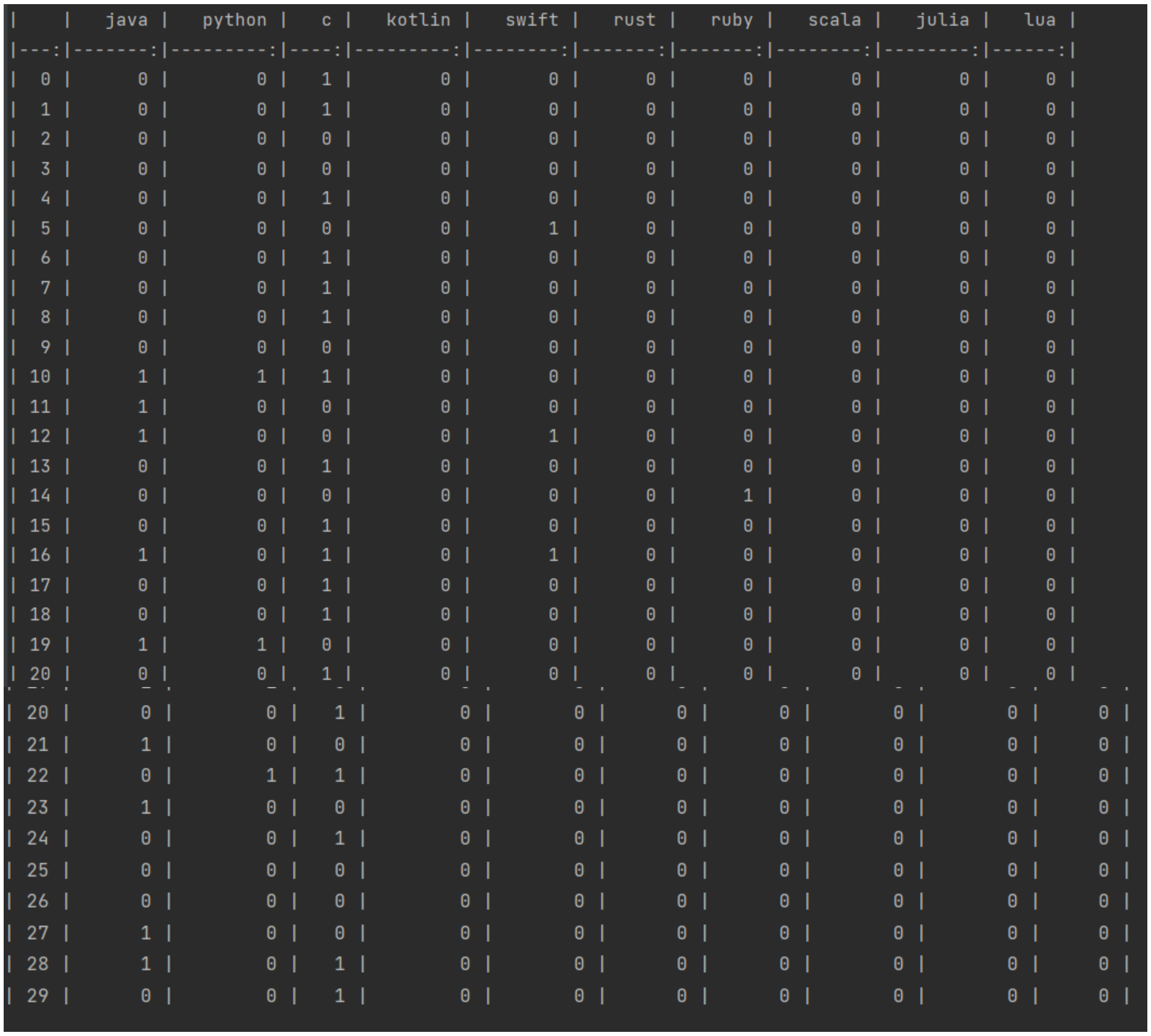
```
java: 2830 of 6267
python: 2701 of 6267
c: 1125 of 6267
kotlin: 291 of 6267
swift: 104 of 6267
rust: 88 of 6267
ruby: 40 of 6267
scala: 15 of 6267
julia: 7 of 6267
lua: 0 of 6267
```

try to show like matrix 0 1



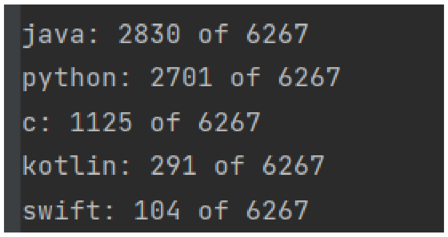
```
lang = [['java'], ['python'], ['c'], ['kotlin'], ['swift'], ['rust'], ['ruby'], ['scala'], ['julia'],['lua']]
parsed_description = parse_job_description()
parsed_db = parse_db()
all_terms = lang + parsed_db
query_map = pd.DataFrame(parsed_description.apply
                         (lambda s: [1 if np.all([d in s for d in db])else 0 for db in all_terms])
                         .values.tolist(), columns=[' '.join(d) for d in all_terms])
```

Pasakon Paingjai 622115028



```
|    |   java |   python |   c |   kotlin |   swift |   rust |   ruby |   scala |   julia |   lua |
|---:|-------:|---------:|----:|---------:|--------:|-------:|-------:|--------:|--------:|------:|
|  0 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  1 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  2 |      0 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  3 |      0 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  4 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  5 |      0 |        0 |   0 |        0 |       1 |      0 |      0 |       0 |       0 |     0 |
|  6 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  7 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  8 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
|  9 |      0 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 10 |      1 |        1 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 11 |      1 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 12 |      1 |        0 |   0 |        0 |       1 |      0 |      0 |       0 |       0 |     0 |
| 13 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 14 |      0 |        0 |   0 |        0 |       0 |      0 |      1 |       0 |       0 |     0 |
| 15 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 16 |      1 |        0 |   1 |        0 |       1 |      0 |      0 |       0 |       0 |     0 |
| 17 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 18 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 19 |      1 |        1 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 20 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 20 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 21 |      1 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 22 |      0 |        1 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 23 |      1 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 24 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 25 |      0 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 26 |      0 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 27 |      1 |        0 |   0 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 28 |      1 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
| 29 |      0 |        0 |   1 |        0 |       0 |      0 |      0 |       0 |       0 |     0 |
```

result



```
java: 2830 of 6267
python: 2701 of 6267
c: 1125 of 6267
kotlin: 291 of 6267
swift: 104 of 6267
```