03장. 고객의 전체 모습을 파악하는 테크닉

> 💡
> **REMEMBER**

- 현황을 분석해서 **문제점을 파악**하고 더 좋은 미래로 바꾸기 위해 **최적의 정책을 실시할 수 있게 하는 것**이 데이터 분석의 묘미
- 데이터를 **적절히 가공해서 가시화**하는 것만으로도 많은 정보를 얻을 수 있음
- 데이터를 보고 든 의문점은 집계해서 확인하는 것 뿐만 아니라 현장 사람들에게 **적극적으로 의견을 청취**하는 것이 좋음

- 조인을 하면 자동으로 결측치가 들어가는 경우가 존재(키가 없거나 잘못된 조인)
- **조인 후에는 결측치를 확인**해서 조인이 잘 되었는지 확인하기

> 📚 **Plus Topic**

🔶 where()
🔶 relativedelta()
🔶 정규표현식

🔶 where()

- Reference

  https://pinkwink.kr/1236

- numpy의 함수
- 원하는 조건의 인덱스를 반환
- 슬라이싱에도 사용 가능
- if-else문 대체 가능
- for문보다 속도가 빠름

```
a = np.array([1,2,3,10,20,30,0.1,0.2])

np.where(a<1)
a[np.where(a<1)]
np.where(a>=10,0,a)
```

🔶 relativedelta()

- Reference

  https://yganalyst.github.io/data_handling/memo_8/

  https://jaeyung1001.tistory.com/108

  https://pydole.tistory.com/entry/Python-dateutil-모듈을-활용한-전년전월전일-구하기

  https://dateutil.readthedocs.io/en/stable/relativedelta.html

- dateutil.relativedelta 모듈의 relativedelta 함수
- 기준 날짜에 대한 연산이 필요할 때 사용
- timedelta에서 month의 계산이 불가능하므로 주로 월 계산을 할 때 많이 사용함

```
from datetime import datetime
from dateutil.relativedelta import relativedelta

my_date = datetime(2019,10,10)
new_date1 = my_date+relativedelta(months=4)
new_date2 = my_date+relatiedelta(months=-4)

new_data1 = my_date+relativedelta(days=100)
new_data2 = my_date+relativedelta(days=-100)
```

- 특정일로부터 얼마나 시간이 흘렀는지 , 특정일로부터 얼마나 시간이 남았는지 계산 가능

```
from datetime import *
from dateutil.relativedelta import *

now = datetime.now()
year, month, day = 1981, 1, 1
day = relativedelta(now, datetime(year, month, day))
print('%s years, %s months, %s days' %(day.years, day.months, day.days))
```

- 사용 가능한 인자
  - Singular argument : year,month,day,hour,minute,second,mirosecond
  - Plural argument : years,months,weeks,days,hours,minutes,seconds,microseconds
  - weekday
  - leapdays
  - yearday,nlyearday

- 추가 - Calender

```
from datetime import *
from dateutil.relativedelta import *
import calendar

NOW = datetime.now() # 시간까지 포함
TODAY = date.today() # 날짜만 포함

# Next Friday
TODAY+relativedelta(weekday=FR)
TODAY+relativedelta(weekday=calennder.FRIDAY)

# Last Friday in this month
TODAY+relativedelta(day=31,weekday=FR(-1))

# Next Wednesday
TODAY+relativedelta(weekday=WE(+1))

# Next Wednesday but Not Today
TODAY+relativedelta(days=+1,weekday=WE(+1))
```

🔶 정규표현식

- Reference

  https://wikidocs.net/4308

  https://bio-info.tistory.com/21

- 문자 클래스 [ ]
  - [ ] 사이의 문자들과 매치
  - 하이픈을 사용하면 두 문자 사이의 범위를 의미
  - 문자 클래스 안에는 어떤 문자나 메타 문자도 사용 가능
  - ^ 문자는 반대라는 의미를 가짐
  - 예시
    - `[abc]` : a,b,c 중 한 개의 문자와 매치
    - `[a-zA-Z]`: 알파벳 모두
    - `[0-9]` : 숫자

  ▼ 자주 사용하는 문자 클래스

[자주 사용하는 문자 클래스]

[0-9] 또는 [a-zA-Z] 등은 무척 자주 사용하는 정규 표현식이다. 이렇게 자주 사용하는 정규식은 별도의 표기법으로 표현할 수 있다. 다음을 기억해 두자.

- `\d` - 숫자와 매치, [0-9]와 동일한 표현식이다.
- `\D` - 숫자가 아닌 것과 매치, `[^0-9]`와 동일한 표현식이다.
- `\s` - whitespace 문자와 매치, `[ \t\n\r\f\v]`와 동일한 표현식이다. 맨 앞의 빈 칸은 공백문자(space)를 의미한다.
- `\S` - whitespace 문자가 아닌 것과 매치, `[^ \t\n\r\f\v]`와 동일한 표현식이다.
- `\w` - 문자+숫자(alphanumeric)와 매치, `[a-zA-Z0-9_]`와 동일한 표현식이다.
- `\W` - 문자+숫자(alphanumeric)가 아닌 문자와 매치, `[^a-zA-Z0-9_]`와 동일한 표현식이다.

대문자로 사용된 것은 소문자의 반대임을 추측할 수 있다.

- Dot(.)
  - 줄바꿈 문자 \n을 제외한 모든 문자와 매치
  - 정규식 작성시 re.DOTALL 옵션을 주면 \n 문자와도 매치됨
  - . 이 문자 클래스 내에 사용된다면 . 문자 그대로를 의미
  - 예시
    - `a.b` : a와 b 사이에 어떤 문자가 들어가도 매치
    - `a[.]b` : a.b 문자열과 매치

- 반복 *
  - * 바로 앞에 있는 문자가 무한대로 반복(메모리 제한으로 2억개 정도 가능)
  - 문자가 0번 반복되어도 매치 가능
  - 예시
    - `ca*t` : ct , cat , caaat 모두 매치 가능

- 반복 +
  - 최소 1번 이상 반복될 때 사용
  - 예시
    - `ca*t` : ct는 매치 불가능

- 반복 ({m,n},?)
  - { } 메타 문자를 사용하여 반복 횟수를 고정
  - 반복 횟수를 m부터 n까지 매치 가능
  - m을 생략하면 0을 의미 , n을 생략하면 무한대 의미
  - {m} : 반드시 m번 반복되어야만 매치
  - ? : {0,1}를 의미 , 있어도 되고 없어도 됨

- 그 외 메타 문자
  - ^ : 문자열의 처음
  - $: 문자열의 마지막

- 파이썬의 re 모듈
  - match() : 문자열의 처음부터 정규식과 매치되는지 조사
  - search() : 문자열 전체를 검색하여 정규식과 매치되는지 조사
  - findall() : 정규식과 매치되는 모든 문자열을 리스트로 반환
  - finditer() : 정규식과 매치되는 모든 문자열을 반복 가능한 객체로 반환

  - re.DOTALL 또는 re.S : . 메타 문자에 \n문자도 포함해여 매치
  - re.IGNORECASE 또는 re.I : 대소문자 구별 없이 매치를 수행
  - re.MULTILINE 또는 re.M : ^,$ 문자를 문자열의 각 줄마다 적용
  - re.VERBOSE 또는 re.X : 정규식을 주석 또는 줄 단위로 구분

> 💻 **Another Data**

Indian Companies Registration Data [1857 - 2020]

https://www.kaggle.com/rowhitswami/all-indian-companies-registration-data-1900-2019

▼ **Detail**

This file contains the details of all Indian companies ever registered in India from 1857 to 2020.**# of rows** - 1992170**# of columns** - 17

Columns

1. **CORPORATE_ IDENTIFICATION_NUMBER** - Corporate Identification Number sometimes referred to as CIN is a unique identification number which is assigned by the ROC (Registrar of Companies) of various states under the MCA (Ministry of Corporate Affairs).
2. **COMPANY_NAME** - Name of the company.
3. **COMPANY_STATUS** - The 'Status' tell the current state of the company. Whether it is active and operating or dormant or it has been struck off and closed. There are 13 such status that a company could be carrying.

- **ACTV** - Active
- **NAEF** - Not available for e-filing
- **ULQD** - Under liquidation
- **AMAL** - Amalgamated
- **STOF** - Strike off
- **DISD** - Dissolved
- **CLLD** - Converted to LLP and Dissolved
- **UPSO** - Under process of Striking Off
- **CLLP** - Converted to LLP
- **LIQD** - Liquidated
- **DRMT** - Dormant
- **MLIQ** - Vanished
- **D455** - Dormant under section 455

4. **COMPANY_CLASS** - Companies are primarily classified into private and public. Private companies or private limited companies are those companies that are closely-held and have less than 200 shareholders. Public companies are limited companies that have more than 200 shareholders and are listed on a stock exchange.
    - Public
    - Private
    - Private (One Person Company)
5. **COMPANY_CATEGORY** - The category of the company.
    - Company limited by Shares
    - Company Limited by Guarantee
    - Unlimited Company
6. **COMPANY_SUB _CATEGORY** - The sub-category of the company.
    - Non-govt company
    - State Govt company
    - Subsidiary of Foreign Company
    - Guarantee and Association comp
    - Union Govt company
7. **DATE*OF* REGISTRATION** - Date of registration of the company.
8. **REGISTERED_STATE** - State in which company was registered.
9. **AUTHORIZED_CAP** - Authorized Capital of the company (INR)
10. **PAIDUP_CAPITAL** - Paid Up Capital of the company (INR).
11. **INDUSTRIAL_CLASS** - Industrial class of the company as per NIC 2004.
12. **PRINCIPAL_ BUSINESS_ACTIVITY _AS_ PER_CIN** - Principal Business Activity of the company as per CIN.
13. **REGISTERED_OFFICE _ADDRESS** - Registered office address of the company.
14. **REGISTRAR_OF _COMPANIES** - Registrar office of the company.
15. **EMAIL_ADDR** - Email address of the companies owner/director.
16. **LATEST_YEAR _ANNUAL _RETURN** - Annual return of the last year.
17. **LATEST_YEAR _FINANCIAL _STATEMENT** - Financial Statement of the last year.

▼ **Memo**

- 데이터 전처리에서 데이터 값을 변경할 때 칼럼이름 지정 → 지정하지 않으면 칼럼 전체 변화 .. (혼합 칼럼 처리.. 할 뻔 😢) → loc 연산자를 사용해서 변경하기
- datetime 변환 시 range 오류 나면 파라미터 지정
- datetime에는 NaT가 있어서 dropna()로 제거되지 않는 것들이 있음
  → 처리해주지 않으면 year,month 등 구분시 실수로 처리됨
- 여러 개의 칼럼을 이름을 통해서 가져올 때는 [[]] 사용
- reset_index()로 인덱스 재할당 (drop=True,inplace=True)
- groupby 결과에서 barplot을 그리고 싶을 때는 라벨을 지정해줘야 하는데 , unique()를 사용하면 결과와 칼럼의 순서가 달라짐
- 데이터에서 여러 개의 조건을 필터링할 경우 and , or 대신 &,| 사용하고 조건들에 괄호 쳐주기

Step1. 데이터 전처리

※ NULL 값이 있는 칼럼 먼저 처리

1. COMPANY_CLASS 칼럼
   - 1992170 row 중 5078개 NULL
   - 값을 대체하는 것은 무리가 있다고 판단, NULL값에 Unknown값 할당
   - 편의를 위해 Private(One Person Company)이름 변경

2. COMPANY_CATEGORY 칼럼
   - 1992170 row 중 5085개 NULL
   - 마찬가지로 임의로 값을 대체하는데는 무리가 있다고 판단해, NULL 값에 Unknown값 할당
   - 값 이름 수정

3. COMPANY_SUB_CATEGORY 칼럼
   - 1992170 row 중 5090개 NULL
   - COMPANY_CATEGORY와 동일하게 처리

4. DATE_OF_REGISTRATION 칼럼
   - 1992170 row 중 2525개 NULL
   - 날짜를 위주로 데이터를 분석할 예정이므로 null값 제거
   - datetime형으로 변환
   - year,month,day 분리해서 사용

5. 그 외 전처리
   - 날짜끼리 뭉쳐있을 수 있도록 칼럼 순서 변화
   - COMPANY_STATUS,REGISTERED_STATE에 이상한 변수가 없는지 확인
   - 인덱스 값 다시 부여
   - 칼럼 이름 가공
   - 전처리 데이터 csv 파일로 저장

Step2. 데이터 분석

- 그룹 분석
  - STATUS 별

    - CATEGORY , SUB CATEGORY 별
    - REGISTERED 별
    - 날짜 별 - 연도 , 월 , 분기 ..

- 날짜 분석
    - YEAR,MONTH만 가지고는 특별한 결론을 내기 어려움
    - 분기, 계절 칼럼을 만들어서 시각화