02장. 대리점 데이터를 가공하는 테크닉

> 💡
> **REMEMBER**

- '데이터의 정합성에 문제가 있다' = 데이터에 나타나는 **입력 오류**나 **표기 방법의 차이**가 부정합을 일으킨다
  - 같은 날짜라도 포맷이 다르면 다른 문자열 데이터
  - 공백 유무에 따라 다른 데이터로 인식
- 데이터의 오류를 해소하고 정합성을 보장하는 것은 데이터 분석의 기초
  - **데이터의 속성이나 의미를 이해**
  - **데이터의 오류를 파악**
- 엑셀 데이터
  - **서식이 다른 데이터**가 섞여 있을 수 있다는 점을 주의
  - 날짜 형식의 숫자를 단순히 파이썬으로 계산하면 이틀이 어긋나므로 , **엑셀 숫자값에서 2를 빼서 처리**
    (엑셀은 1에서 출발,파이썬은 0에서 출발 / 엑셀은 평년인 1900년의 2월 29일도 유효한 날짜로 계산하는 버그가 존재)
- 데이터를 가공 후 저장할 때 **칼럼의 배치를 조정한 후에 파일로 저장**하는 것이 직관적으로 이해하기 쉬움
  (예 - date와 month는 가까이 위치)

> 📌
> **Question**

Q1. groupby와 pivot_table을 적용하는 상황이 따로 정해져 있나 ?

Q2. str 을 굳이 사용해야 하나 ?

Q3 astype()은 왜 적용할 때도 있고 안할때도 있나 ?

Q4. 공백 제거시 왜 replace를 두 번 사용하나 ?

※ 제품 가격 전처리 코드 다시 짜보기 !

A1. 기본적으로 "~별 ~ " 분석이 필요할 때는 groupby를 사용하나, 가독성이 pivot_table이 더 좋으므로 **행/열을 만족하는 데이터가 하나 존재한다면 pivot_table 사용**하기

A2. upper() , replace()는 기본적으로 **문자열에 적용하는 함수**이므로 Series 객체에 적용하는 것이 아닌 str를 활용해 StringMethods로 **변환** 후 적용

문자열 메소드(String Methods) : 문자열에 특화되어 문자열 자료형을 다양하게 처리할 수 있는 함수

R, Python 분석과 프로그래밍의 친구 (by R Friend) :: [Python] 파이썬 문자열 처리를 위한 다양한 메소드 (Python string methods)

▼ A3.

```
# 의문을 가진 코드
kokyaku_daiho['등록일'].astype("str").str.isdigit().sum() :22
kokyaku_daiho['등록일'].str.isdigit().sum() : 0.0
kokyaku_daiho['등록일'].astype("str').isdigit().sum() : 오류

temp1 = kokyaku_daicho['등록일'].astype("str").str : StringMethods
temp2 = kokyaku_daicho['등록일'].astype("str") : Series
temp3 = kokyaku_daicho['등록일'].str : StringMethods

temp1 == temp3 : False
```

A4. 공백이 두 개가 있는 혹시 모르는 상황을 방지하기 위해서 .. ? 그렇다면 더 좋은 방법이 분명 있지 않을까 .. ?

> **Plus Topic**

🔶 문자열 형태 확인
🔶 데이터프레임 인덱싱

🔶 문자열 형태 확인

- Reference

  https://velog.io/@oaoong/python-문자열체크-isdigit-isalpha-isalnum-isnumeric-is-decimic-함수

  https://onthepressure.tistory.com/entry/Python-String-isdigit-Method-문자열-함수

- 조건에 만족하면 True , 만족하지 않으면 False를 리턴
- isalpha() : 문자열이 문자로만 되어 있는지 확인(공백문자, : , 숫자 등 안됨)
- isalnum() : 문자열이 영어,한글 혹은 숫자로만 되어있는지 확인 (공백문자 , : 등 안됨)
- isdigit() : 문자열이 숫자인지 확인 (거듭제곱 특수문자 확인)
  음수나 소수점이 있는 경우에도 False 리턴
- isnumeric() : 숫자처럼 생긴 모든 글자(제곱근,분수,거듭제곱 특수문자)
- isdecimal() : 문자열이 int타입으로 변환 가능한 문자로 구성되었는지 확인

🔶 데이터프레임 인덱싱

- Reference

  https://datascienceschool.net/01 python/04.03 데이터프레임 고급 인덱싱.html

  https://dojang.io/mod/page/view.php?id=2211

- loc 인덱서
  - df.loc[행 인덱싱값] df.loc[행 인덱싱값, 열 인덱싱값]
  - 행 인덱싱 값은 정수 또는 행 인덱스 데이터, 열 인덱싱값은 라벨 문자열
    - 인덱스데이터
    - 인덱스데이터 슬라이스
    - 인덱스데이터 리스트
    - 같은 행 인덱스를 가지는 불리언 시리즈(행 인덱싱)
    - 또는 위의 값들을 반환하는 함수

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

  - df.loc["b":"c"] = df["b":"c"] = df.loc[["b","c"]] ≠ df[["b","c"]]
  - 칼럼을 선택하고 싶으면 loc연산자 없이 사용
- iloc 인덱서

- 라벨이 아니라 순서를 나타내는 정수 인덱스만 사용 가능
- -붙이면 뒤에서부터 순서 세기
- loc 인덱서와 거의 비슷 - 인덱스가 하나만 들어가면 행을 선택 등

▼ 참고

다음 소스 코드를 완성하여 튜플 n에서 인덱스가 홀수인 요소들이 출력되게 만드세요.

practice_odd_index.py

```
n = -32, 75, 97, -10, 9, 32, 4, -15, 0, 76, 14, 2

print(_____________)
```

실행 결과

```
(75, -10, 32, -15, 76, 2)
```

```
n[1::2]
n[1:12:2]
n[1:len(n):2]
```