```
# 3. 사드 배치의 영향으로 중국인 관광객이 얼마나 줄었을까?
```

데이터 불러오기 및 전처리

```
# 불러올 데이터의 형태 파악
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
kto_201901=pd.read_excel('/content/kto_201901.xlsx',header=1, usecols='A:G', skipfooter=4)
kto_201901

# header=1 : 국적/관광/상용/공용/유학,연수/기타 관련 내용 출력
# usecols='A:G'국적 ~ 계까지의 값 출력
# skipfooter=4 : 계, 전년동기, 성장률, 구성비 생략
```

	국적	관광	상용	공용	유학/연수	기타	계
0	아시아주	**765082**	**10837**	**1423**	**14087**	**125521**	**916950**
1	일본	**198805**	**2233**	**127**	**785**	**4576**	**206526**
2	대만	**86393**	**74**	**22**	**180**	**1285**	**87954**
3	홍콩	**34653**	**59**	**2**	**90**	**1092**	**35896**
4	마카오	**2506**	**2**	**0**	**17**	**45**	**2570**
...	...	...	...	...	...	...	...
62	아프리카 기타	**768**	**718**	**90**	**206**	**908**	**2690**
63	기타대륙	**33**	**4**	**0**	**1**	**16**	**54**
64	국적미상	**33**	**4**	**0**	**1**	**16**	**54**
65	교포소계	**0**	**0**	**0**	**0**	**15526**	**15526**
66	교포	**0**	**0**	**0**	**0**	**15526**	**15526**

67 rows × 7 columns

```
# tail() 함수 활용
kto_201901.tail()
```

	국적	관광	상용	공용	유학/연수	기타	계
62	아프리카 기타	**768**	**718**	**90**	**206**	**908**	**2690**
63	기타대륙	**33**	**4**	**0**	**1**	**16**	**54**
64	국적미상	**33**	**4**	**0**	**1**	**16**	**54**
65	교포소계	**0**	**0**	**0**	**0**	**15526**	**15526**
66	교포	**0**	**0**	**0**	**0**	**15526**	**15526**

데이터 전처리

```
# 데이터 전처리(Data Preprocessing): "분석에 적합하도록"이라는 표현에는 어디서나 통용되는 공통적인 프로세스가 있지는 않지만,
# 여기에는 데이터 변수별로 값에 이상이 없는지 확인, 결측값 처리, 이상치 처리, 변수 정규화, 파생 변수 생성 등의 과정이 포함됩니다.

# 이러한 데이터 전처리는 분석하려는 데이터에 대한 이해가 선행됐을 대, 그 기준을 정하기가 수월하기 때문에 해당 데이터 분야에 대한
# 도메인 지식이 필요합니다.
```

데이터 탐색

```
kto_201901.info()

# <class 'pandas.core.frame.DataFrame'>: 데이터는 pandas의 데이터프레임 클래스로 구성.
# dtypes: int64(6), object(1): 1개의 문자형 변수(국적)와 6개의 정수형 변수(관광,상용, 공용, 유학/연수, 기타, 계)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67 entries, 0 to 66
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   국적       67 non-null     object
 1   관광       67 non-null     int64
 2   상용       67 non-null     int64
 3   공용       67 non-null     int64
 4   유학/연수    67 non-null     int64
 5   기타       67 non-null     int64
 6   계        67 non-null     int64
dtypes: int64(6), object(1)
memory usage: 3.8+ KB
```

```
# describe()함수 활용: 정수형 변수의 특징
kto_201901.describe()
```

	관광	상용	공용	유학/연수	기타	
count	67.00000	67.000000	67.000000	67.000000	67.000000	67.00
mean	26396.80597	408.208955	132.507463	477.462687	5564.208955	32979.19
std	102954.04969	1416.040302	474.406339	2009.484800	17209.438418	122821.36
min	0.00000	0.000000	0.000000	0.000000	16.000000	54.00
25%	505.00000	14.500000	2.500000	17.500000	260.000000	927.00
50%	1304.00000	45.000000	14.000000	43.000000	912.000000	2695.00
75%	8365.00000	176.500000	38.000000	182.000000	2824.500000	14905.50
max	765082.00000	10837.000000	2657.000000	14087.000000	125521.000000	916950.00

```
# 각 칼럼에서 0인 부분을 필터링
condition=(kto_201901['관광']==0) | (kto_201901['상용']==0) |(kto_201901['공용']==0) | (kto_201901['유학/연수']==0)
kto_201901[condition]

# |(or) 조건을 이용해 4가지 목적 중 한 가지라도 입국객수가 0인 국적으로 필터링.
```

	국적	관광	상용	공용	유학/연수	기타	계
4	마카오	2506	2	0	17	45	2570
20	이스라엘	727	12	0	9	57	805
22	우즈베키스탄	1958	561	0	407	2828	5754
38	스위스	613	18	0	19	97	747
45	그리스	481	17	4	0	273	775
46	포르투갈	416	14	0	13	121	564
51	크로아티아	226	12	0	3	250	491
54	폴란드	713	10	0	27	574	1324
59	대양주 기타	555	3	4	0	52	614
63	기타대륙	33	4	0	1	16	54
64	국적미상	33	4	0	1	16	54
65	교포소계	0	0	0	0	15526	15526
66	교포	0	0	0	0	15526	15526

```
# 데이터프레임에 기준년월 추가
kto_201901['기준년월']='2019-01'
kto_201901.head()

# 2010년 1월부터 2019년 8월 데이터를 활용할 예정이기 때문에, 각 데이터마다 기준년월 정보가 필요합니다.
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월
0	아시아주	765082	10837	1423	14087	125521	916950	2019-01
1	일본	198805	2233	127	785	4576	206526	2019-01
2	대만	86393	74	22	180	1285	87954	2019-01
3	홍콩	34653	59	2	90	1092	35896	2019-01
4	마카오	2506	2	0	17	45	2570	2019-01

```
# 국적 데이터만 남기기
kto_201901['국적'].unique()

# .unique()함수: 칼럼 내 중복을 제거한 값들을 보여주는 함수.
# 하나의 칼럼에는 하나의 특징을 가진 값들이 들어있어야 데이터 분석에 활용하기 용이합니다.
# 국적 칼럼에는 '아시아주','일본','대만',...과 같이 대륙과 국가가 혼용된 것을 확인할 수 있습니다.
```

```
array(['아시아주', '일본', '대만', '홍콩', '마카오', '태국', '말레이시아', '필리핀', '인도네시아',
       '싱가포르', '미얀마', '베트남', '인도', '스리랑카', '파키스탄', '방글라데시', '캄보디아', '몽골',
       '중국', '이란', '이스라엘', '터키', '우즈베키스탄', '카자흐스탄', 'GCC', '아시아 기타', '미주',
       '미국', '캐나다', '멕시코', '브라질', '미주 기타', '구주', '영국', '독일', '프랑스',
       '네덜란드', '스웨덴', '스위스', '이탈리아', '덴마크', '노르웨이', '벨기에', '오스트리아', '스페인',
       '그리스', '포르투갈', '핀란드', '아일랜드', '우크라이나', '러시아', '크로아티아', '루마니아',
       '불가리아', '폴란드', '구주 기타', '대양주', '오스트레일리아', '뉴질랜드', '대양주 기타',
       '아프리카주', '남아프리카공화국', '아프리카 기타', '기타대륙', '국적미상', '교포소계', '교포'],
      dtype=object)
```

```
# 대륙 목록 만들기
continents_list=['아시아주','미주','구주','대양주','아프리카주','기타대륙','교포소계']
continents_list
```

```
['아시아주', '미주', '구주', '대양주', '아프리카주', '기타대륙', '교포소계']
```

```
# 대륙목록에 해당하는 값 제외
condition=(kto_201901.국적.isin(continents_list)==False)
kto_201901_country=kto_201901[condition]
kto_201901_country['국적'].unique()
```

```
array(['일본', '대만', '홍콩', '마카오', '태국', '말레이시아', '필리핀', '인도네시아', '싱가포르',
       '미얀마', '베트남', '인도', '스리랑카', '파키스탄', '방글라데시', '캄보디아', '몽골', '중국',
       '이란', '이스라엘', '터키', '우즈베키스탄', '카자흐스탄', 'GCC', '아시아 기타', '미국',
       '캐나다', '멕시코', '브라질', '미주 기타', '영국', '독일', '프랑스', '네덜란드', '스웨덴',
       '스위스', '이탈리아', '덴마크', '노르웨이', '벨기에', '오스트리아', '스페인', '그리스', '포르투갈',
       '핀란드', '아일랜드', '우크라이나', '러시아', '크로아티아', '루마니아', '불가리아', '폴란드',
       '구주 기타', '오스트레일리아', '뉴질랜드', '대양주 기타', '남아프리카공화국', '아프리카 기타',
       '국적미상', '교포'], dtype=object)
```

```
# kto_201901_country 데이터를 head()함수로 확인
kto_201901_country.head()

# kto_201901_country 데이터에서는 1번부터 시작합니다. 그 이유는 kto_201901_country 데이터가 kto_201901에서
# 필터링한 결과를 저장한 데이터이기 때문입니다.-> kto_201901_country의 인덱스 값에는 기존 데이터에서
# 대륙에 해당하는 값이 누락돼 있습니다.
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월
1	일본	**198805**	**2233**	**127**	**785**	**4576**	**206526**	**2019-01**
2	대만	**86393**	**74**	**22**	**180**	**1285**	**87954**	**2019-01**
3	홍콩	**34653**	**59**	**2**	**90**	**1092**	**35896**	**2019-01**
4	마카오	**2506**	**2**	**0**	**17**	**45**	**2570**	**2019-01**
5	태국	**34004**	**37**	**199**	**96**	**6998**	**41334**	**2019-01**

```
# 인덱스 재설정: 데이터 필터링하다 보면, 인덱스 값이 누락되는 경우가 발생하는 데,
# 이런 경우에는 인덱스 값을 초기화하는 것이 좋습니다.

kto_201901_country_newindex=kto_201901_country.reset_index(drop=True)
kto_201901_country_newindex.head()

# 첫번째 로우의 인덱스 값이 0부터 시작하는 것을 확인할 수 있습니다.
# reset_index() 함수를 사용하면 인덱스 값을 0부터 순차적으로 다시 초기화합니다.
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월
0	일본	**198805**	**2233**	**127**	**785**	**4576**	**206526**	**2019-01**
1	대만	**86393**	**74**	**22**	**180**	**1285**	**87954**	**2019-01**
2	홍콩	**34653**	**59**	**2**	**90**	**1092**	**35896**	**2019-01**
3	마카오	**2506**	**2**	**0**	**17**	**45**	**2570**	**2019-01**
4	태국	**34004**	**37**	**199**	**96**	**6998**	**41334**	**2019-01**

```
# 대륙 칼럼 만들기
# 지금까지의 작업을 통해 국적 칼럼에 국가명만 남겨놓았으니 각 국가들이 어떤 대륙에 속하는지에 대한
# 정보도 가지고 있으면 데이터 분석에 도움이 될 것이다.

# 대륙 칼럼값 만들기
continents=['아시아']*25+['아메리카']*5+['유럽']*23+['오세아니아']*3₩
+['아프리카']*2+['기타대륙']+['교포']
print('continents값:',continents)
```

```
continents값: ['아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아', '아시아',
```

	2019년 01월 외래객 입국-목적별/국적별								
2	국적	관광	상용	공용	유학/연수	기타	계	전년동기	성장률(%)	구성비(%)
3	아시아주	765,082	10,837	1,423	14,087	125,521	916,950	775,023	18.3	83
4	일본	198,805	2,233	127	785	4,576	206,526	167,083	23.6	18.7
5	대만	86,393	74	22	180	1,285	87,954	75,820	16	8
6	홍콩	34,653	59	2	90	1,092	35,896	34,671	3.5	3.2
7	마카오	2,506	2	0	17	45	2,570	2,933	-12.4	0.2
8	태국	34,004	37	199	96	6,998	41,334	44,941	-8	3.7
9	말레이시아	19,043	95	7	99	2,821	22,065	22,113	-0.2	2
10	필리핀	14,279	211	161	184	15,638	30,473	30,454	0.1	2.8
11	인도네시아	14,183	136	38	187	4,298	18,842	17,034	10.6	1.7
12	싱가포르	8,372	94	8	48	1,333	9,855	9,664	2	0.9
13	미얀마	1,304	10	31	67	3,877	5,289	5,988	-11.7	0.5
14	베트남	10,739	763	110	1,667	6,904	20,183	16,285	23.9	1.8
15	인도	2,318	2,656	46	177	3,474	8,671	7,421	16.8	0.8
16	스리랑카	157	54	5	28	1,043	1,287	1,500	-14.2	0.1
17	파키스탄	238	178	10	193	413	1,032	1,156	-10.7	0.1
18	방글라데시	149	126	27	97	848	1,247	1,231	1.3	0.1
19	캄보디아	635	39	55	51	1,915	2,695	2,431	10.9	0.2
20	몽골	8,358	77	304	484	562	9,785	10,793	-9.3	0.9
21	중국	320,113	2,993	138	8,793	60,777	392,814	305,127	28.7	35.6
22	이란	60	45	10	23	46	184	554	-66.8	0
23	이스라엘	727	12	0	9	57	805	809	-0.5	0.1
24	터키	792	13	32	36	912	1,785	1,934	-7.7	0.2
25	우즈베키스탄	1,958	561	0	407	2,828	5,754	5,431	5.9	0.5
26	카자흐스탄	2,185	24	7	92	1,266	3,574	3,942	-9.3	0.3
27	GCC	1,550	37	14	72	103	1,776	1,805	-1.6	0.2
28	아시아 기타	1,561	308	70	205	2,410	4,554	3,903	16.7	0.4
29	미주	54,982	526	2,657	479	22,277	80,921	77,166	4.9	7.3
30	미국	42,989	418	2,578	229	16,523	62,737	59,895	4.7	5.7
31	캐나다	8,034	57	30	43	4,296	12,460	12,417	0.3	1.1

```
# 대륙 칼럼 추가
kto_201901_country_newindex.head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월
0	일본	198805	2233	127	785	4576	206526	2019-01
1	대만	86393	74	22	180	1285	87954	2019-01
2	홍콩	34653	59	2	90	1092	35896	2019-01
3	마카오	2506	2	0	17	45	2570	2019-01
4	태국	34004	37	199	96	6998	41334	2019-01

```
# 대륙 칼럼 생성
kto_201901_country_newindex['대륙']=continents
kto_201901_country_newindex.head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙
0	일본	198805	2233	127	785	4576	206526	2019-01	아시아
1	대만	86393	74	22	180	1285	87954	2019-01	아시아
2	홍콩	34653	59	2	90	1092	35896	2019-01	아시아
3	마카오	2506	2	0	17	45	2570	2019-01	아시아
4	태국	34004	37	199	96	6998	41334	2019-01	아시아

```
kto_201901_country_newindex.tail()
# tail을 통해 결과값을 보면, 대륙: 교포(1), 기타대륙(1), 아프리카(2), 오세아니아(3) 등을 확인할 수 잇다.
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙
55	대양주 기타	**555**	**3**	**4**	**0**	**52**	**614**	**2019-01**	오세아니아
56	남아프리카공화국	**368**	**9**	**1**	**6**	**616**	**1000**	**2019-01**	아프리카
57	아프리카 기타	**768**	**718**	**90**	**206**	**908**	**2690**	**2019-01**	아프리카
58	국적미상	**33**	**4**	**0**	**1**	**16**	**54**	**2019-01**	기타대륙
59	교포	**0**	**0**	**0**	**0**	**15526**	**15526**	**2019-01**	교포

```
# 국적별 관광객 비율 살펴보기
kto_201901_country_newindex['관광객비율(%)']=round(kto_201901_country_newindex['관광']/kto_201901_country_newindex['계']*100,1)
kto_201901_country_newindex

# 관광객 비율을 추가해서 생성한 다음, 관광객으로 들어온 사람/총합을 통해 몇%인지 구한다.
# 관광객 비율: (관광객수/전체 입국객 수)*100
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)
0	일본	**198805**	**2233**	**127**	**785**	**4576**	**206526**	**2019-01**	아시아	**96.3**
1	대만	**86393**	**74**	**22**	**180**	**1285**	**87954**	**2019-01**	아시아	**98.2**
2	홍콩	**34653**	**59**	**2**	**90**	**1092**	**35896**	**2019-01**	아시아	**96.5**
3	마카오	**2506**	**2**	**0**	**17**	**45**	**2570**	**2019-01**	아시아	**97.5**
4	태국	**34004**	**37**	**199**	**96**	**6998**	**41334**	**2019-01**	아시아	**82.3**
5	말레이시아	**19043**	**95**	**7**	**99**	**2821**	**22065**	**2019-01**	아시아	**86.3**
6	필리핀	**14279**	**211**	**161**	**184**	**15638**	**30473**	**2019-01**	아시아	**46.9**
7	인도네시아	**14183**	**136**	**38**	**187**	**4298**	**18842**	**2019-01**	아시아	**75.3**
8	싱가포르	**8372**	**94**	**8**	**48**	**1333**	**9855**	**2019-01**	아시아	**85.0**
9	미얀마	**1304**	**10**	**31**	**67**	**3877**	**5289**	**2019-01**	아시아	**24.7**
10	베트남	**10739**	**763**	**110**	**1667**	**6904**	**20183**	**2019-01**	아시아	**53.2**
11	인도	**2318**	**2656**	**46**	**177**	**3474**	**8671**	**2019-01**	아시아	**26.7**
12	스리랑카	**157**	**54**	**5**	**28**	**1043**	**1287**	**2019-01**	아시아	**12.2**
13	파키스탄	**238**	**178**	**10**	**193**	**413**	**1032**	**2019-01**	아시아	**23.1**
14	방글라데시	**149**	**126**	**27**	**97**	**848**	**1247**	**2019-01**	아시아	**11.9**
15	캄보디아	**635**	**39**	**55**	**51**	**1915**	**2695**	**2019-01**	아시아	**23.6**
16	몽골	**8358**	**77**	**304**	**484**	**562**	**9785**	**2019-01**	아시아	**85.4**
17	중국	**320113**	**2993**	**138**	**8793**	**60777**	**392814**	**2019-01**	아시아	**81.5**
18	이란	**60**	**45**	**10**	**23**	**46**	**184**	**2019-01**	아시아	**32.6**
								2019-	아시	

```
# 관광객비율(%) 칼럼으로 내림차순 정렬
kto_201901_country_newindex.sort_values(by='관광객비율(%)',ascending=False).head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)
1	대만	86393	74	22	180	1285	87954	2019-01	아시아	98.2
3	마카오	2506	2	0	17	45	2570	2019-01	아시아	97.5
2	홍콩	34653	59	2	90	1092	35896	2019-01	아시아	96.5

```
# 관광객비율(%) 칼럼으로 오름차순 정렬
kto_201901_country_newindex.sort_values(by='관광객비율(%)',ascending=True).head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)
59	교포	0	0	0	0	15526	15526	2019-01	교포	0.0
14	방글라데시	149	126	27	97	848	1247	2019-01	아시아	11.9
12	스리랑카	157	54	5	28	1043	1287	2019-01	아시아	12.2

```
# pivot_table() 함수 활용
kto_201901_country_newindex.pivot_table(values='관광객비율(%)',index='대륙',aggfunc='mean') # 대륙별로 관광객 비율의 평균
```

대륙	관광객비율(%)
교포	0.000000
기타대륙	61.100000
아메리카	68.200000
아시아	59.624000
아프리카	32.700000
오세아니아	84.833333
유럽	63.826087

```
# 중국 국적만 필터링
condition=(kto_201901_country_newindex.국적=='중국')
kto_201901_country_newindex[condition]
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)

```
# 기준년월별로 전체 외국인 관광객 대비 국적별 관광객 비율 살펴보기
tourist_sum=sum(kto_201901_country_newindex['관광'])
tourist_sum # 2019년 1월에 한국을 방문한 총 외국인 관광객수: 884,293명
```

884293

```
# 전체 비율 칼럼 생성
kto_201901_country_newindex['전체비율(%)']=round(kto_201901_country_newindex['관광']/tourist_sum*100,1)
kto_201901_country_newindex.head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)	전체비율(%)
0	일본	198805	2233	127	785	4576	206526	2019-01	아시아	96.3	22.5
1	대만	86393	74	22	180	1285	87954	2019-01	아시아	98.2	9.8
2	홍콩	34653	59	2	90	1092	35896	2019-01	아시아	96.5	3.9

```
# 전체비율(%) 칼럼 기준으로 내림차순 정렬
kto_201901_country_newindex.sort_values('전체비율(%)',ascending=False).head()

# 중국: 36.2%: 전체 외국인 관광객 중 가장 높은 비율 차지
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	대륙	관광객비율(%)	전체비율(%)
17	중국	320113	2993	138	8793	60777	392814	2019-01	아시아	81.5	36.2
0	일본	198805	2233	127	785	4576	206526	2019-01	아시아	96.3	22.5
1	대만	86393	74	22	180	1285	87954	2019-01	아시아	98.2	9.8

```
# 데이터 전처리: 데이터를 분석에 적합한 형태로 만드는 과정이기도 하지만, 데이터를 미시적으로 살펴봐야 하는 만큼
# 데이터에 대한 이해도를 높이는 과정이기도 합니다.
```

데이터 전처리 과정을 함수로 만들기

```
# 1. 불러올 데이터의 형태 파악
# 2. 엑셀 파일 파이썬으로 불러오기(pd.read_excel())
# 3. 데이터 탐색(info(), describe())
# 4. 기준 년월 칼럼 추가
# 5. 국적 데이터만 남기기(대륙 데이터 제거): 결측치 확인, 데이터 정리
# 6. 대륙 칼럼 만들기
# 7. 국적별 관광객 비율(%) 살펴보기
# 8. 전체 외국인 관광객 대비 국적별 관광객 비율 살펴보기

# 반복되는 작업을 진행할 때는 작업 단위별로 함수를 만든 후에 반복문을 실행하는 것이 유용합니다.


def create_kto_data(yy,mm):
  # 1. 불러올 데이터의 형태 파악
  # python 버전: file_path='./files/kto_{}{}.xlsx'.format(yy,mm)
  from google.colab import drive
  drive.mount('/content/drive')
  file_path='/content/kto_{}{}.xlsx'.format(yy,mm)

  # 2. 엑셀 파일 불러오기
  df=pd.read_excel(file_path,header=1, skipfooter=4, usecols='A:G')

  # 3. 기준년월 칼럼 추가
  df['기준년월']='{}-{}'.format(yy,mm)

  # 4. 국적 칼럼에서 대륙 제거하고 국가만 남기기
  ignore_list=['아시아주','미주','구주','대양주','아프리카주','기타대륙','교포'] # 제거할 대륙명 선정하기
  condition=(df['국적'].isin(ignore_list)==False) # 대륙 미포함 조건
  df_country=df[condition].reset_index(drop=True)

  # 5. 대륙 칼럼 추가
  continents=['아시아']*25+['아메리카']*5+['유럽']*23+['대양주']*3+['아프리카']*2+['기타대륙']+['교포']

  # 6. 국가별 관광객비율(%) 칼럼 추가
  df_country['관광객비율(%)']=round(df_country.관광/df_country.계 * 100, 1)

  # 7. 전체 비율(%) 칼럼 추가
  tourist_sum=sum(df_country['관광'])
  df_country['전체비율(%)']=round(df_country['관광']/tourist_sum *100,1)

  # 8. 결과 출력
  return(df_country)


# {}: 괄호 안의 yy(기준 년), mm(기준 월)-각 자리에 입력받는 값을 함수 내에서 사용하겠다는 의미.


# create_kto_data()함수를 활용해 2018년 12월 데이터 불러오기
kto_test=create_kto_data(2018,12)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
kto_test.head()
```

	국적	관광	상용	공용	유학/연수	기타	계	기준년월	관광객비율(%)	전체비율(%)
0	일본	252461	1698	161	608	3593	258521	2018-12	97.7	22.7
1	대만	85697	71	22	266	1252	87308	2018-12	98.2	7.7
2	홍콩	58355	41	3	208	939	59546	2018-12	98.0	5.2
3	마카오	6766	0	1	20	36	6823	2018-12	99.2	0.6

반복문을 통해 다수의 엑셀 데이터를 불러와서 합치기

```
# 이중 반복문으로 기준년월 출력
for yy in range(2010,2020):
  for mm in range(1,13):
    yymm='{}{}'.format(yy,mm)
    print(yymm)

# 이중 반복문을 통해 연도와 월을 결합한 결과가 출력되는 것을 확인할 수 있습니다.
```

```
20101
20102
20103
20104
20105
20106
20107
20108
20109
201010
201011
201012
20111
20112
20113
20114
20115
20116
20117
20118
20119
201110
201111
201112
20121
20122
20123
20124
20125
20126
20127
20128
20129
201210
201211
201212
20131
20132
20133
20134
20135
20136
20137
20138
20139
201310
201311
201312
20141
20142
20143
20144
20145
20146
20147
20148
20149
201410
```

```
# zfil()함수 이용
mm=1
print('mm값:',mm)
print('mm값:',str(mm).zfill(2))
```

```
mm값: 1
mm값: 01
```

```
for yy in range(2010,2020):
  for mm in range(1,13):
    mm_str=str(mm).zfill(2)
    yymm='{}{}'.format(yy,mm_str)
    print(yymm)
```

```
201001
201002
201003
201004
201005
201006
201007
201008
201009
201010
201011
201012
201101
201102
201103
201104
201105
201106
201107
201108
201109
201110
201111
201112
201201
201202
201203
201204
201205
201206
201207
201208
201209
201210
201211
201212
201301
201302
201303
201304
201305
201306
201307
201308
201309
201310
201311
201312
201401
201402
201403
201404
201405
201406
201407
201408
201409
201410
```

```
# 데이터를 담을 빈 데이터 프레임 만들기
df=pd.DataFrame()
```

```
# import pandas as pd

for yy in range(2010,2020):
  for mm in range(1,13):
    temp=create_kto_data(str(yy),str(mm).zfill(2))
    mm_str=str(mm).zfill(2)
    yymm='{}{}'.format(yy,mm_str)
    print('년도',yymm)

    df=pd.concat([df,temp],ignore_index=True)
# df.append -> pd.concat
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201001
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201002
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201003
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201004
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201005
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201006
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201007
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201008
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201009
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201010
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
년도 201011
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-38-052408fbb09d> in <cell line: 3>()
      3 for yy in range(2010,2020):
      4   for mm in range(1,13):
----> 5     temp=create_kto_data(str(yy),str(mm).zfill(2))
      6     mm_str=str(mm).zfill(2)
      7     yymm='{}{}'.format(yy,mm_str)

                                4 frames
/usr/local/lib/python3.10/dist-packages/google/colab/_message.py in
read_reply_from_input(message_id, timeout_sec)
     94     reply = _read_next_input_message()
     95     if reply == _NOT_READY or not isinstance(reply, dict):
---> 96       time.sleep(0.025)
     97       continue
     98     if (

KeyboardInterrupt:
```

```
# error: 2019년 9월 데이터가 없기 때문에 에러 발생
# 실제로는 문제가 없지만, 이렇게 오류가 나는 경우를 출력하지 않고 싶을 때: try~except문


#try~except문
# import pandas as pd

for yy in range(2010,2020):
  for mm in range(1,13):
    try:
      temp=create_kto_data(str(yy),str(mm).zfill(2))
      mm_str=str(mm).zfill(2)
      yymm='{}{}'.format(yy,mm_str)
      print('년도',yymm)
      df=pd.concat([df,temp],ignore_index=True)
    except:
      pass
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201001
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201002
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201003
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201004
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201005
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201006
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201007
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201008
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201009
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201010
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201011
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201012
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
년도 201101
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201102
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201103
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201104
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201105
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201106
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201107
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201108
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201109
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201110
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201111
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201112
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201201
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201202
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201203
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
년도 201204
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```