DataScience Exam 01

20192883 강승민

데이터 명세

- 4669개의 행 / 9개의 열
- ID_USER: 유저의 id
- USER_STATE: 유저가 있는 주의 이름
- USER_TIMEZONE: 유저가 있는 곳의 time zone
- ID_HOTEL: 호텔의 id
- HOTEL_CITY: 호텔이 있는 도시의 이름
- HOTEL_STATE: 호텔이 있는 주의 이름
- HOTEL_TIMEZONE: 호텔이 있는 곳의 time zone
- Trip Type: 여행 유형
  - 1: 가족
  - 2: 연인
  - 3: 직장
  - 4: 혼자
  - 5: 친구
- Rating: 평가 점수

US Time Zone

데이터 분석

1. 데이터 불러오기

- Excel 형식 데이터 불러오기

2. 누락 데이터 확인

- 누락된 데이터 없음

3. 중복 데이터 확인 및 삭제

- 중복된 데이터 26개 삭제

4. Trip Type 속성 이름 바꾸기

- 속성의 값들을 숫자에서 문자열로 매핑

5. 데이터 요약

- 전체 데이터의 평가 점수 평균
- 유저별 호텔 방문 횟수

- 호텔별 방문 유저 수

6. 평점 분석

- 각 평균 평점대별 호텔의 수
- 호텔이 위치한 주별 평균 평점
- 호텔이 위치한 도시별 평균 평점

7. 상관관계 분석

- 호텔이 위치한 시간대와 여행 유형의 상관관계 분석
- 호텔이 위치한 주와 여행 유형의 상관관계 분석
- 유저 위치와 여행 유형의 상관관계 분석
- 호텔 위치와 이용자 거주지 간의 상관 관계 분석
- 여행 유형과 호텔 평가 점수 간의 상관 관계 분석

8. 유사성 분석

1. 모든 행에 대해 USER_STATE와 HOTEL_STATE가 같은 지에 대한 열을 새로 만든다.
    - IS_SAME_STATE

1. 전체 사용자 간의 유사성을 분석한다.
    - 유사성은 전체 유저 간 평균으로 계산한다.

1. 사는 곳과 다른 지역으로 여행을 가는 유저들의 유사성 분석
    - 같은 STATE == True인 ID_USER_SAME_STATE 리스트를 만든다.
    - ID_USER_SAME_STATE 리스트에 속한 유저들 간의 유사성을 분석한다.
        - 유사성은 해당되는 유저 간 평균으로 계산한다.

1. 사는 곳과 같은 지역으로 여행을 가는 유저들의 유사성 분석
    - 같은 STATE == False인 ID_USER_DIFF_STATE 리스트를 만든다.
    - ID_USER_DIFF_STATE 리스트에 속한 유저들 간의 유사성을 분석한다.
        - 유사성은 해당되는 유저 간 평균으로 계산한다.

- 평균 평점이 5점 만점인 호텔의 수
- 호텔 위치별(도시, 주) 평균 평가 점수
- 여행 유형별 호텔 id, 호텔 위치(도시, 주), 호텔 time zone
- 1인당 평균 방문 호텔 수
- 자신이 사는 주에 있는 호텔에 방문하는 유저는 주 내에 있는 호텔 위주로 방문한다.
- 반대로, 다른 주에 있는 호텔에 방문하는 유저는 주로 다른 주에 있는 호텔 위주로 방문한다.

1. 데이터 불러오기

In [1]:
```
# -*- coding: utf-8 -*-
import numpy as np
# pandas 불러오기
import pandas as pd

# 데이터 모든 행과 열 출력하기 (누락 데이터를 눈으로 확인하기 위함)
# pd.set_option('display.max_columns', None)
# pd.set_option('display.max_rows', None)

# 엑셀 파일 읽기 (같은 디렉토리 안에 위치)
```

```
df = pd.read_excel('./exam_data.xls')              # header=0 (default 옵션)

df.head(10)
```

Out[1]:

	ID_USER	USER_STATE	USER_TIMEZONE	ID_HOTEL	HOTEL_CITY	HOTEL_STATE	HOTEL_TIMEZONE	T
0	45	GA	Eastern	105170	Memphis	TN	Central	
1	45	GA	Eastern	223229	SanAntonio	TX	Central	
2	45	GA	Eastern	258688	Albuquerque	NM	Mountain	
3	45	GA	Eastern	98827	ELPaso	TX	Central	
4	45	GA	Eastern	99518	SanAntonio	TX	Central	
5	64	TX	Central	224427	Cleveland	OH	Eastern	
6	64	TX	Central	1751886	Austin	TX	Central	
7	64	TX	Central	99120	Houston	TX	Central	
8	100	NY	Eastern	120111	Jacksonville	FL	Eastern	
9	100	NY	Eastern	91428	Indianapolis	IN	Eastern	

Reason

- 데이터를 엑셀 형식으로 불러와서 제대로 업로드 되었는지 상위 10개 행만 출력하여 확인해본다.

2. 누락 데이터 확인

In [2]:
```
# isnull() 메소드를 통해 각 열의 누락 데이터 개수 세기
print(df.isnull().sum(axis=0))
```

```
ID_USER            0
USER_STATE         0
USER_TIMEZONE      0
ID_HOTEL           0
HOTEL_CITY         0
HOTEL_STATE        0
HOTEL_TIMEZONE     0
Trip Type          0
Rating             0
dtype: int64
```

Reason

- 비어있는 값은 데이터를 분석할 때 오류를 발생시킬 가능성이 높기 때문에 제거하는 것이 좋다.
- 해당 데이터에는 모든 속성에 비어있는 값이 없기 때문에 누락 데이터 처리 과정은 생략한다.

3. 중복 데이터 확인

In [3]:
```
# duplicated() 메소드를 활용하여 모든 속성에 대해 중복된 데이터 확인
dup = df.duplicated() # default: 모든 속성에 대해 확인
dup.head()
```

Out[3]:
```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

In [4]:
```
# 중복 데이터 제거 전 데이터 개수 확인: 4669개
len(df)
```

Out[4]:
```
4669
```

In [5]:
```
# drop_duplicates() 메소드를 활용하여 중복된 데이터 제거
df = df.drop_duplicates()
```

In [6]:
```
# 중복 데이터 제거 후 데이터 개수 확인: 4643개 (26개 삭제)
len(df)
```

Out[6]:
```
4643
```

Reason

- 해당 과제에서 중복된 데이터는 이상치라고 가정한다. (다시 말해, 한 사람이 같은 곳에 같은 사람들과 같은 평점을 남기지 않는다고 가정한다.)
- 중복된 데이터는 데이터 분석 시 데이터의 편향을 높일 수 있기 때문에 제거하는 것이 좋다.

4. Trip Type 속성 이름 바꾸기

In [7]:
```
# Trip Type 속성의 값들을 (숫자 -> 문자열)로 타입 변경
df['Trip Type'] = df['Trip Type'].astype(str)
```

In [8]:
```
# 바꿀 이름 설정 {기존 이름: 새 이름}
new_trip_name = {'1': 'Family', '2': 'Coouples', '3': 'Business', '4': 'Solo travel', '5
```

In [9]:
```
# 미리 설정한 딕셔너리를 통해 새로운 이름으로 매핑
df['Trip Type'] = df['Trip Type'].map(new_trip_name)
df.head(10)
```

Out[9]:

	ID_USER	USER_STATE	USER_TIMEZONE	ID_HOTEL	HOTEL_CITY	HOTEL_STATE	HOTEL_TIMEZONE	
0	45	GA	Eastern	105170	Memphis	TN	Central	E
1	45	GA	Eastern	223229	SanAntonio	TX	Central	E
2	45	GA	Eastern	258688	Albuquerque	NM	Mountain	E
3	45	GA	Eastern	98827	ELPaso	TX	Central	E
4	45	GA	Eastern	99518	SanAntonio	TX	Central	E
5	64	TX	Central	224427	Cleveland	OH	Eastern	
6	64	TX	Central	1751886	Austin	TX	Central	E
7	64	TX	Central	99120	Houston	TX	Central	E
8	100	NY	Eastern	120111	Jacksonville	FL	Eastern	C
9	100	NY	Eastern	91428	Indianapolis	IN	Eastern	C

Reason

- Trip Type 속성이 기존대로라면, 숫자들이 무엇을 의미하는 지 전혀 예측할 수 없다.
- 속성의 값들의 이름을 변경함으로써 각 여행 유형이 무엇을 나타내는 지 명확하게 알 수 있다.
- 속성의 값들이 나타내는 의미는 다음과 같다.
  - 1 : 가족과 여행

- 2 : 연인과 여행
- 3 : 출장
- 4 : 혼자 여행
- 5 : 친구들과 여행

5. 데이터 요약

In [10]:
```
# 데이터프레임의 기본 정보
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4643 entries, 0 to 4668
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   ID_USER          4643 non-null   int64 
 1   USER_STATE       4643 non-null   object
 2   USER_TIMEZONE    4643 non-null   object
 3   ID_HOTEL         4643 non-null   int64 
 4   HOTEL_CITY       4643 non-null   object
 5   HOTEL_STATE      4643 non-null   object
 6   HOTEL_TIMEZONE   4643 non-null   object
 7   Trip Type        4643 non-null   object
 8   Rating           4643 non-null   int64 
dtypes: int64(3), object(6)
memory usage: 362.7+ KB
```

In [11]:
```
# 데이터프레임의 기술 통계 정보 요약 (숫자 데이터만 해당)
df_des = df.describe()
df_des
```

Out[11]:

	ID_USER	ID_HOTEL	Rating
count	4643.000000	4.643000e+03	4643.000000
mean	32788.116519	3.008667e+05	3.628042
std	18862.912513	3.928865e+05	1.078623
min	45.000000	7.233900e+04	1.000000
25%	16292.000000	9.420600e+04	3.000000
50%	33018.000000	1.095150e+05	4.000000
75%	49609.500000	2.440030e+05	4.000000
max	65457.000000	2.151986e+06	5.000000

- 평가 점수 평균이 약 3.62점이라는 것을 알 수 있다.
- 나머지 데이터는 크게 의미가 없는 통계치이다.

In [12]:
```
# 유저별 호텔 방문 횟수
df["ID_USER"].value_counts().head()
```

Out[12]:
```
27783    18
13087    16
40423    16
17535    12
49098    12
Name: ID_USER, dtype: int64
```

- 27783이라는 id를 가진 유저가 18번의 호텔 방문으로 가장 많은 횟수를 방문하였다.

In [13]:
```
# 호텔별 방문 유저 수
df["ID_HOTEL"].value_counts().head()
```

Out[13]:
```
100603    13
86320     12
98117     12
224241    12
111855    11
Name: ID_HOTEL, dtype: int64
```

- 100603이라는 id를 가진 호텔이 13번의 방문으로 가장 많은 방문 횟수를 가진다.

Reason

- 데이터의 전체적인 정보를 알아보는 것은 중요하다. 정보의 유형에 따라 범주형/연속형 데이터로 분류할 수 있다.
  - 만약 Rating 속성이 숫자가 아닌 문자열로 되어있다면 숫자 계산을 할 수 있도록 숫자형으로 바꿔주어야 한다.
- 데이터 통계를 알아보면 전체적인 데이터의 흐름을 이해할 수 있다.
  - 한 명의 유저가 최대 몇 번이나 호텔을 방문했는 지, 하나의 호텔이 몇 명의 방문자가 있었는 지를 대략적으로 파악하면 데이터를 이해하기 수월해진다.

6. 평점 분석

In [14]:
```
# 필요한 라이브러리를 불러온다.
import matplotlib.pyplot as plt
import seaborn as sns
```

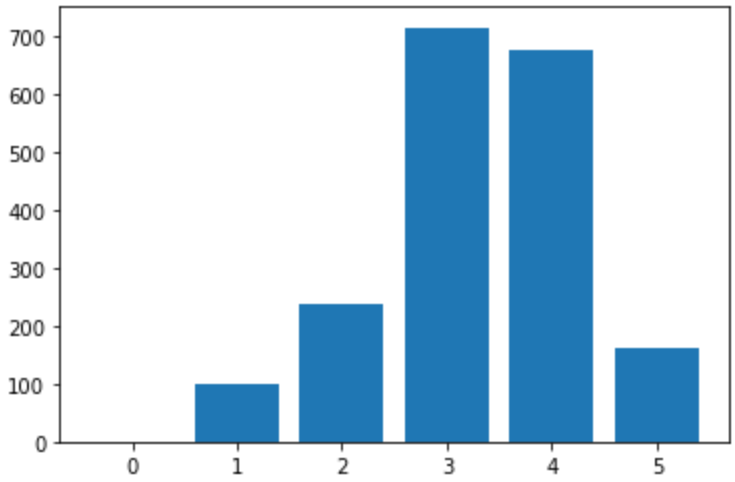
In [15]:
```
# 평균 평점이 5점인 호텔의 수
df_rating = df.groupby('ID_HOTEL').mean()
df_rating_5 = len(df_rating[df_rating['Rating'] == 5])
df_rating_4_to_5 = len(df_rating[df_rating['Rating'] < 5]) - len(df_rating[df_rating['Ra
df_rating_3_to_4 = len(df_rating[df_rating['Rating'] < 4]) - len(df_rating[df_rating['Ra
df_rating_2_to_3 = len(df_rating[df_rating['Rating'] < 3]) - len(df_rating[df_rating['Ra
df_rating_1_to_2 = len(df_rating[df_rating['Rating'] < 2]) - len(df_rating[df_rating['Ra
df_rating_0_to_1 = len(df_rating[df_rating['Rating'] < 1])

print("평점이 5점인 호텔의 수는 {} 곳이다.".format(df_rating_5))
print("평점이 4점대인 호텔의 수는 {} 곳이다.".format(df_rating_4_to_5))
print("평점이 3점대인 호텔의 수는 {} 곳이다.".format(df_rating_3_to_4))
print("평점이 2점대인 호텔의 수는 {} 곳이다.".format(df_rating_2_to_3))
print("평점이 1점대인 호텔의 수는 {} 곳이다.".format(df_rating_1_to_2))
print("평점이 0점대인 호텔의 수는 {} 곳이다.".format(df_rating_0_to_1))
```

```
평점이 5점인 호텔의 수는 163 곳이다.
평점이 4점대인 호텔의 수는 675 곳이다.
평점이 3점대인 호텔의 수는 715 곳이다.
평점이 2점대인 호텔의 수는 237 곳이다.
평점이 1점대인 호텔의 수는 100 곳이다.
평점이 0점대인 호텔의 수는 0 곳이다.
```

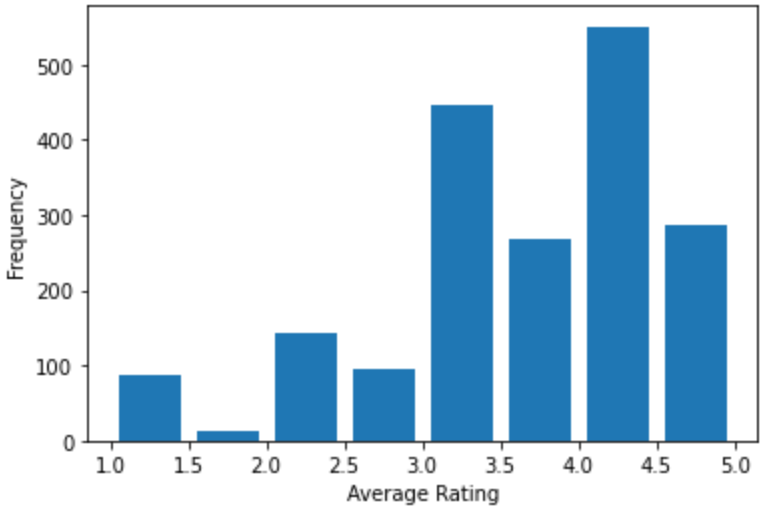
In [16]:
```
# 각 평점대별 호텔 수 막대 그래프
rating_row = ["0", "1", "2", "3", "4", "5"]
rating_col = [df_rating_0_to_1, df_rating_1_to_2, df_rating_2_to_3, df_rating_3_to_4, df
```

In [17]:
```
# x축: 평점대, y축: 호텔의 수
plt.bar(rating_row, rating_col)
plt.show()
```



In [18]:
```
# 각 호텔의 평균 평가 점수 구하기
hotel_rating_mean = df.groupby('ID_HOTEL')['Rating'].mean()

# 히스토그램으로 시각화
plt.hist(hotel_rating_mean, bins=8, rwidth=0.8)
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
plt.show()
```



Reason

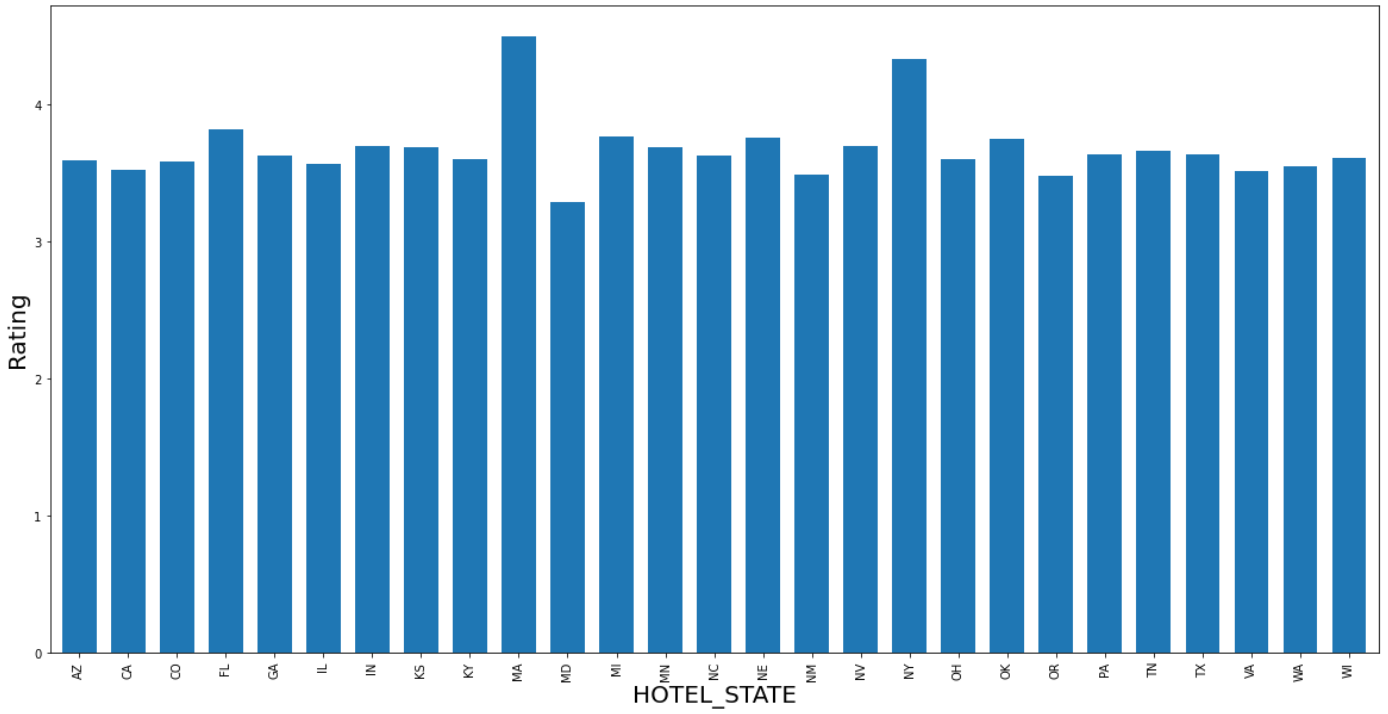
- 평균 평가 점수에 따른 호텔의 수를 파악하여 평점 데이터의 분포를 파악한다.
- 호텔 ID로 그룹화하여 각 호텔이 받은 평점의 평균을 기준으로 잡는다.
- 확인 결과, 데이터가 편향되어 있지 않음을 확인할 수 있다. (무분별한 평점 난발은 없다.)
- 3점대의 평가 점수를 가진 호텔이 가장 많았다.

In [19]:
```
# 호텔이 위치한 주별 평균 평점
df_rating = df.groupby('HOTEL_STATE').mean()
df_rating = df_rating['Rating']
df_rating
```

Out[19]:
```
HOTEL_STATE
AZ    3.591281
CA    3.523992
CO    3.582524
FL    3.817518
GA    3.628049
IL    3.571429
IN    3.698925
KS    3.692913
```

```
KY    3.601626
MA    4.500000
MD    3.285714
MI    3.764706
MN    3.690476
NC    3.625532
NE    3.761905
NM    3.492891
NV    3.696629
NY    4.333333
OH    3.599078
OK    3.745645
OR    3.482759
PA    3.636364
TN    3.660793
TX    3.636285
VA    3.518519
WA    3.550000
WI    3.608696
Name: Rating, dtype: float64
```

In [20]:
```
# 막대 그래프 그리기
df_rating.plot(kind="bar", figsize=(20, 10), width=0.7)
plt.ylabel("Rating", size=20)
plt.xlabel("HOTEL_STATE", size=20)
plt.show()
```



- 호텔이 위치한 주별 평균 평가 점수는 다음과 같이 나타났다.
- 호텔이 위치한 주 중 평균 평가 점수가 가장 높은 곳은 MA이다.

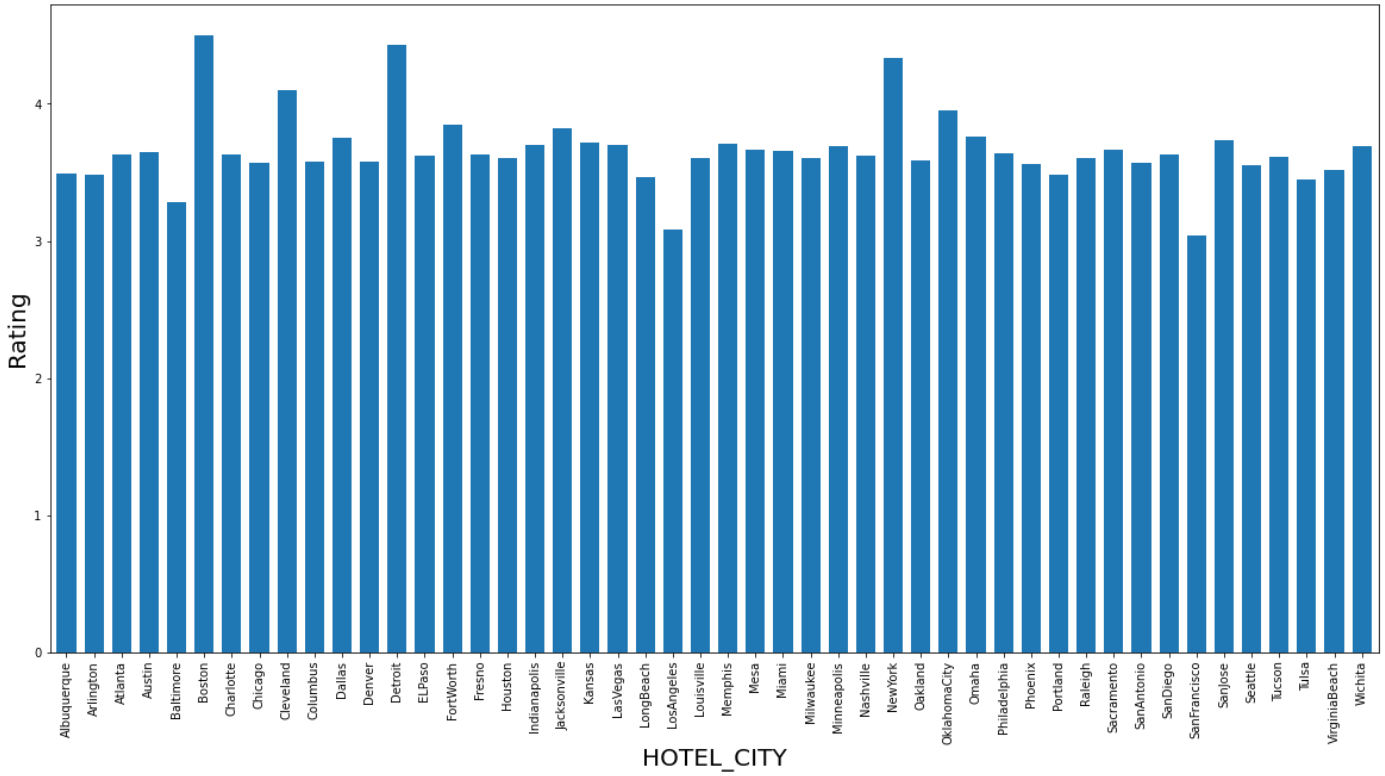
In [21]:
```
# 호텔이 위치한 도시별 평균 평점
df_rating = df.groupby('HOTEL_CITY').mean()
df_rating = df_rating['Rating']
df_rating
```

Out[21]:
```
HOTEL_CITY
Albuquerque      3.492891
Arlington        3.481481
Atlanta          3.628049
Austin           3.646809
Baltimore        3.285714
```

```
Boston          4.500000
Charlotte       3.634731
Chicago         3.571429
Cleveland       4.100000
Columbus        3.574879
Dallas          3.755556
Denver          3.582524
Detroit         4.428571
ELPaso          3.625000
FortWorth       3.844156
Fresno          3.630137
Houston         3.608553
Indianapolis    3.698925
Jacksonville    3.817518
Kansas          3.714286
LasVegas        3.696629
LongBeach       3.464286
LosAngeles      3.078947
Louisville      3.601626
Memphis         3.712871
Mesa            3.666667
Miami           3.659091
Milwaukee       3.608696
Minneapolis     3.690476
Nashville       3.619048
NewYork         4.333333
Oakland         3.583333
OklahomaCity    3.952941
Omaha           3.761905
Philadelphia    3.636364
Phoenix         3.559322
Portland        3.482759
Raleigh         3.602941
Sacramento      3.661654
SanAntonio      3.569106
SanDiego        3.626374
SanFrancisco    3.035714
SanJose         3.732143
Seattle         3.550000
Tucson          3.610390
Tulsa           3.444444
VirginiaBeach   3.518519
Wichita         3.691667
Name: Rating, dtype: float64
```

In [22]:

```
# 막대 그래프 그리기
df_rating.plot(kind="bar", figsize=(20, 10), width=0.7)
plt.ylabel("Rating", size=20)
plt.xlabel("HOTEL_CITY", size=20)
plt.show()
```



- 호텔이 위치한 도시별 평균 평가 점수는 다음과 같이 나타났다.
- 호텔이 위치한 도시 중 평균 평가 점수가 가장 높은 곳은 Boston이다.

7. 상관관계 분석

In [23]:
```
# 시간대별 호텔 이용 현황 분석
user_by_timezone = df.groupby('USER_TIMEZONE').size()
hotel_by_timezone = df.groupby('HOTEL_TIMEZONE').size()

# 데이터프레임으로 변환하여 출력
result = pd.DataFrame({'User': user_by_timezone, 'Hotel': hotel_by_timezone})
result
```

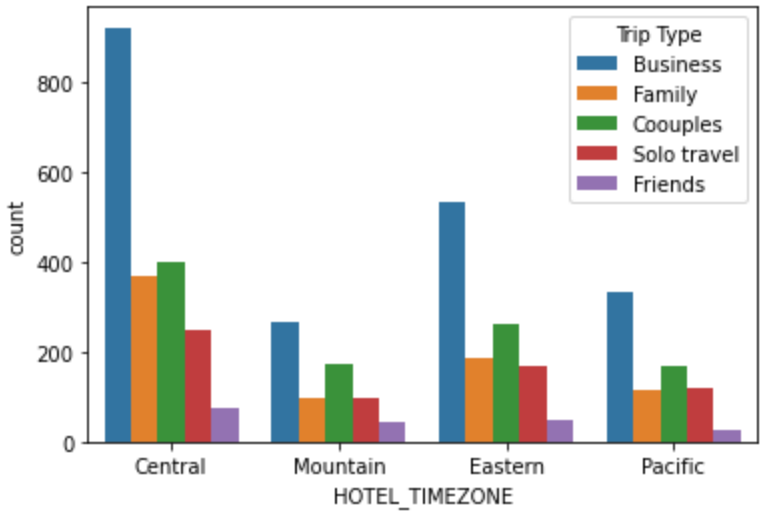
Out[23]:

	User	Hotel
AK	16	NaN
Central	1838	2013.0
Eastern	1581	1192.0
HI	16	NaN
Mountain	425	681.0
Pacific	767	757.0

- AK와 HI에는 호텔이 없는 것을 확인할 수 있다. (거주자도 많지 않다.)
- 나머지는 시간대별로 거주자와 호텔 수가 비슷하게 분포한다.

In [24]:
```
# 호텔이 위치한 시간대와 여행 유형 간의 상관 관계 분석
sns.countplot(x='HOTEL_TIMEZONE', hue='Trip Type', data=df)
```

Out[24]:
```
<AxesSubplot:xlabel='HOTEL_TIMEZONE', ylabel='count'>
```

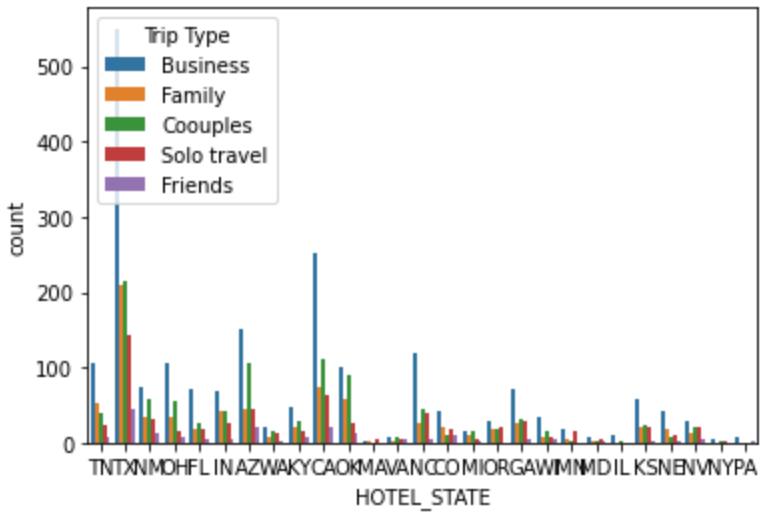


Reason

- 호텔이 위치한 시간대별로 여행 유형에 따른 수를 분석한다.
- 모든 여행 유형 중, 출장의 비율이 가장 높다.
- 사람들은 주로 Central로 출장을 간다. 두 번째는 Eastern으로 많이 간다.
- 가족, 연인, 혼자, 친구와의 연인도 가장 많이 방문하는 곳은 역시 Central이다.
- Central에 호텔이 가장 많거나, 좋다는 것을 대략적으로 알 수 있다.

In [25]:
```
# 호텔이 위치한 주와 여행 유형 간의 상관 관계 분석
sns.countplot(x='HOTEL_STATE', hue='Trip Type', data=df)
```

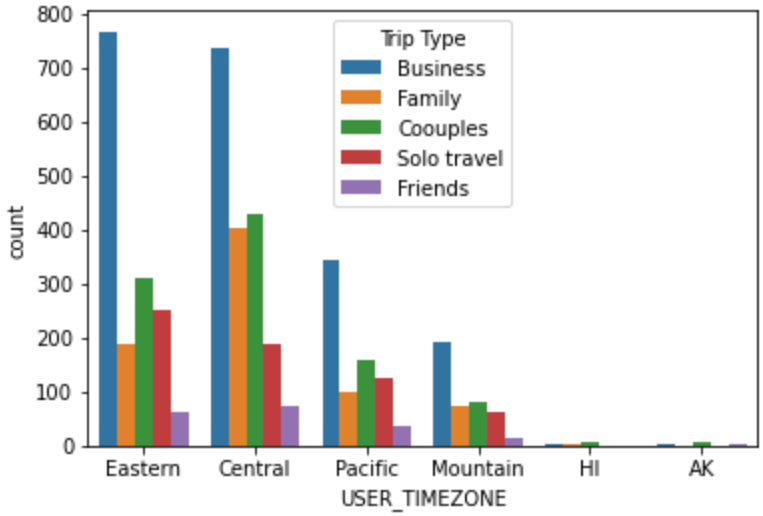
Out[25]:
```
<AxesSubplot:xlabel='HOTEL_STATE', ylabel='count'>
```



- TX 주에 가장 많은 호텔 방문이 있는 것을 알 수 있다.

In [26]:
```
# 유저가 위치한 시간대와 여행 유형 간의 상관 관계 분석
sns.countplot(x='USER_TIMEZONE', hue='Trip Type', data=df)
```

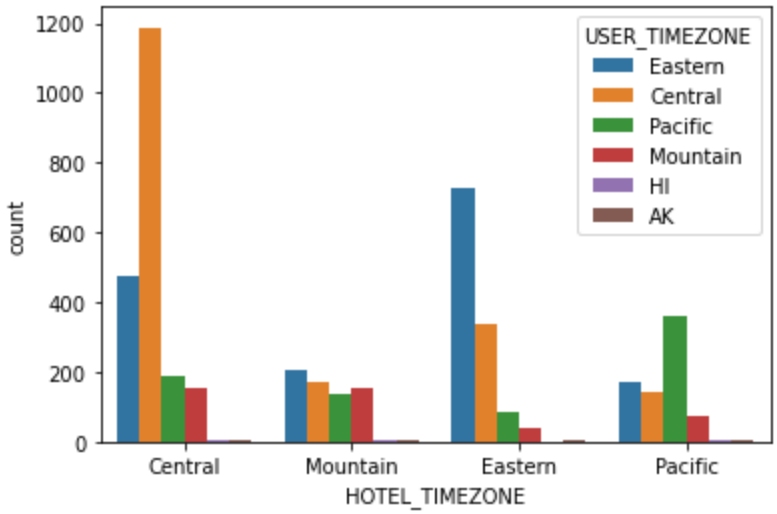
Out[26]:
```
<AxesSubplot:xlabel='USER_TIMEZONE', ylabel='count'>
```



- Eastern과 Central에 사는 사람들은 출장을 가장 많이 간다.
- 위에서 분석한 결과를 보았을 때, Central에 위치한 호텔로 출장을 갈 확률이 높아 보인다.

In [27]:
```
# 호텔 위치와 이용자 거주지 간의 상관 관계 분석 (countplot)
sns.countplot(x='HOTEL_TIMEZONE', hue='USER_TIMEZONE', data=df)
```

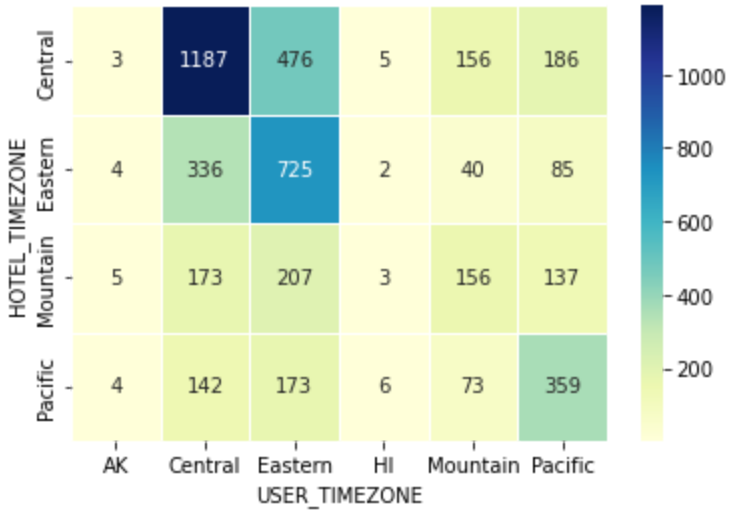
Out[27]:
```
<AxesSubplot:xlabel='HOTEL_TIMEZONE', ylabel='count'>
```



In [28]:
```
# 호텔 위치와 이용자 거주지 간의 상관 관계 분석 (heatmap)
table = df.pivot_table(index=['HOTEL_TIMEZONE'], columns=['USER_TIMEZONE'], aggfunc='siz

sns.heatmap(table,
            annot=True, fmt='d',
            cmap='YlGnBu',
            linewidth=.5)
```

Out[28]:
```
<AxesSubplot:xlabel='USER_TIMEZONE', ylabel='HOTEL_TIMEZONE'>
```

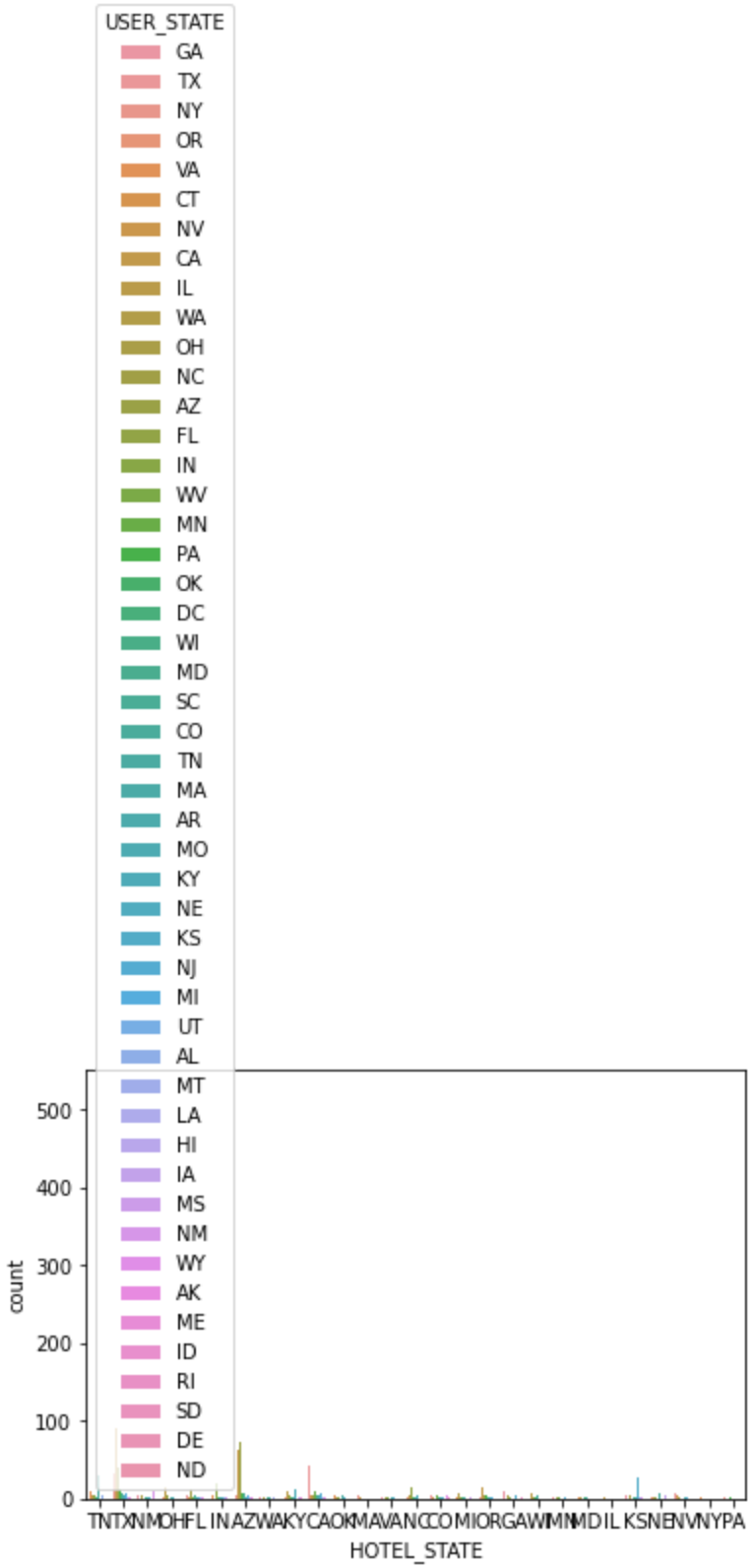


- Central에 위치한 호텔에 방문하는 사람들은 Central에 사는 사람들이 가장 많다.
- Eastern에 위치한 호텔에 방문하는 사람들은 Eastern에 사는 사람들이 가장 많다.
- Pacific에 위치한 호텔에 방문하는 사람들은 Pacific에 사는 사람들이 가장 많다.

-> 사람들은 자신이 살고 있는 시간대 근처 호텔에 방문할 가능성이 가장 높다.

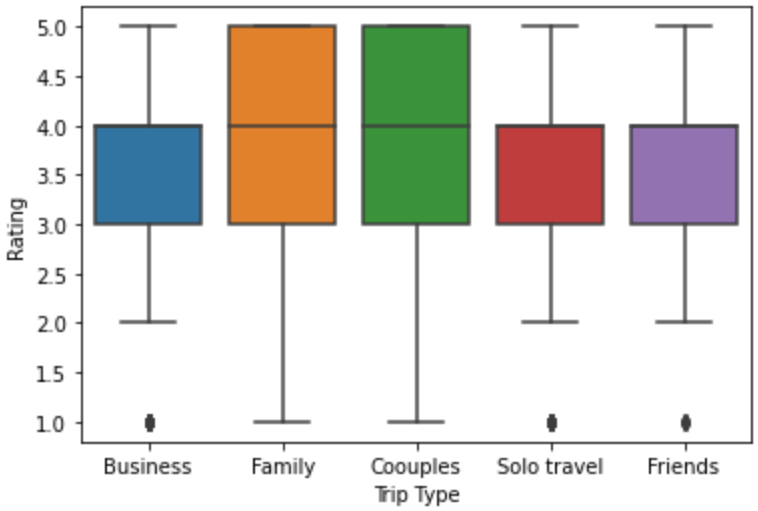
In [29]:
```
# 호텔 위치와 이용자 거주지 간의 상관 관계 분석 (countplot)
sns.countplot(x='HOTEL_STATE', hue='USER_STATE', data=df)
```

Out[29]:
```
<AxesSubplot:xlabel='HOTEL_STATE', ylabel='count'>
```



In [30]:
```
# 여행 유형과 호텔 평가 점수 간의 상관 관계 분석
sns.boxplot(x='Trip Type', y='Rating', data=df)
```

Out[30]:
```
<AxesSubplot:xlabel='Trip Type', ylabel='Rating'>
```



- 점수를 높게 준 여행 유형은 가족과 함께 간 사람들, 연인과 함께 간 사람들로 나타났다.
  - 가족이나 연인과 함께 여행을 간다면 평점을 높게 줄 확률이 높다는 것을 대략적으로 파악할 수 있다.

8. 유사성 분석

유저가 거주하는 주와 호텔이 위치한 주 간의 유사성 분석

In [31]:
```
# USER_STATE와 HOTEL_STATE 열에서 주에 대한 정보만 추출하여 새로운 열을 만들기
df['USER_STATE_ONLY'] = df['USER_STATE'].str.split(',').str[-1].str.strip()
df['HOTEL_STATE_ONLY'] = df['HOTEL_STATE'].str.split(',').str[-1].str.strip()


# 공통된 주의 개수 계산
num_common_states = np.sum(df['USER_STATE_ONLY'] == df['HOTEL_STATE_ONLY'])


# 공통된 주의 비율 계산
state_counts = df['USER_STATE_ONLY'].value_counts()
common_states = df[df['USER_STATE_ONLY'] == df['HOTEL_STATE_ONLY']]['USER_STATE_ONLY'].v
common_ratio = (common_states / state_counts).fillna(0)

print('공통된 주(State) 개수:', num_common_states)
print('공통된 주(State) 비율:\n', common_ratio)
```

```
공통된 주(State) 개수: 1246
공통된 주(State) 비율:
 AK    0.000000
AL    0.000000
AR    0.000000
AZ    0.366834
CA    0.410745
CO    0.084507
CT    0.000000
DC    0.000000
DE    0.000000
FL    0.167464
GA    0.058065
HI    0.000000
IA    0.000000
ID    0.000000
IL    0.000000
IN    0.338983
KS    0.296703
KY    0.200000
LA    0.000000
```

```
MA    0.000000
MD    0.016667
ME    0.000000
MI    0.011236
MN    0.018692
MO    0.000000
MS    0.000000
MT    0.000000
NC    0.352113
ND    0.000000
NE    0.290323
NJ    0.000000
NM    0.272727
NV    0.116279
NY    0.007519
OH    0.372881
OK    0.540984
OR    0.200000
PA    0.010989
RI    0.000000
SC    0.000000
SD    0.000000
TN    0.281818
TX    0.619385
UT    0.000000
VA    0.013245
WA    0.045977
WI    0.241935
WV    0.000000
WY    0.000000
Name: USER_STATE_ONLY, dtype: float64
```

- 공통된 주의 비율이 0에 가까운 주가 많은 것을 확인할 수 있다.
- 이 말은 즉, 사람들은 자신이 거주한 주에 위치한 호텔에 방문하기보다, 다른 주에 위치한 호텔에 방문하는 사람이 많다는 것이다.

In [32]:
```
# USER_STATE와 HOTEL_STATE 개수 확인
print("user_state 개수: ", len(df['USER_STATE'].unique()))
print("hotel_state 개수: ", len(df['HOTEL_STATE'].unique()))
```

```
user_state 개수:  49
hotel_state 개수:  27
```

In [33]:
```
pd.crosstab(df.HOTEL_STATE, df.USER_STATE)
```

Out[33]:

USER_STATE	AK	AL	AR	AZ	CA	CO	CT	DC	DE	FL	...	SC	SD	TN	TX	UT	VA	WA	WI	WV
HOTEL_STATE																				
AZ	3	2	3	73	64	13	0	3	0	8	...	8	0	3	28	2	13	14	1	0
CA	1	0	5	31	237	9	3	7	0	9	...	5	0	5	42	1	7	17	1	0
CO	1	0	2	1	13	12	1	2	0	3	...	2	2	1	6	1	4	0	1	0
FL	1	3	2	0	8	3	0	0	0	35	...	1	0	2	5	4	4	0	0	6
GA	0	8	1	5	10	0	0	2	0	15	...	4	0	13	14	0	6	2	0	2
IL	0	0	0	0	3	2	0	0	0	1	...	0	0	1	2	0	1	0	0	0
IN	0	3	0	1	10	7	0	2	1	8	...	1	0	3	12	0	4	0	7	1
KS	0	1	3	1	6	7	0	1	0	2	...	3	1	4	15	1	0	2	1	0
KY	0	2	0	0	9	1	0	2	0	5	...	1	0	13	11	0	2	0	4	0

MA	0	0	0	0	1	0	1	0	0	1	...	0	0	0	0	0	0	0	0	0
MD	0	0	1	0	1	0	0	1	0	0	...	0	0	1	1	0	3	2	0	0
MI	1	0	2	0	1	0	0	2	0	8	...	0	0	1	7	1	1	8	0	0
MN	0	0	0	0	3	4	0	2	0	4	...	0	0	0	2	0	0	1	3	0
NC	2	1	1	0	9	2	5	2	2	14	...	9	0	2	9	0	19	1	3	3
NE	0	1	1	1	7	8	0	0	0	3	...	1	4	0	9	0	2	0	1	0
NM	1	1	2	17	26	13	0	3	0	9	...	5	0	2	29	0	10	0	3	0
NV	0	0	1	5	21	11	0	1	0	5	...	2	0	1	7	2	1	1	2	0
NY	0	0	0	0	0	0	0	0	0	1	...	0	0	0	4	0	2	0	0	0
OH	0	1	0	11	10	1	0	1	0	6	...	0	0	4	10	0	8	2	1	4
OK	0	0	24	2	9	3	0	0	0	7	...	1	1	4	74	2	6	1	4	0
OR	0	1	0	4	15	1	0	0	0	3	...	2	0	0	7	0	2	11	0	0
PA	0	0	0	0	0	0	0	0	0	0	...	0	0	1	2	0	2	0	0	0
TN	0	5	10	4	8	9	1	3	0	11	...	1	0	31	14	0	11	4	5	1
TX	3	6	7	39	92	31	2	8	0	45	...	7	0	18	524	1	35	17	9	0
VA	0	0	1	1	0	0	0	3	0	0	...	0	0	0	3	0	2	0	0	1
WA	3	1	0	1	8	0	0	2	0	3	...	2	0	0	5	0	3	4	1	0
WI	0	1	1	2	6	5	1	0	0	3	...	0	0	0	4	0	3	0	15	0

27 rows × 49 columns

- USER_STATE와 HOTEL_STATE 교차표 생성

In [34]:
```
df["IS_SAME_STATE"] = df["USER_STATE"] == df["HOTEL_STATE"]
df.head(10)
```

Out[34]:

	ID_USER	USER_STATE	USER_TIMEZONE	ID_HOTEL	HOTEL_CITY	HOTEL_STATE	HOTEL_TIMEZONE	
0	45	GA	Eastern	105170	Memphis	TN	Central	E
1	45	GA	Eastern	223229	SanAntonio	TX	Central	E
2	45	GA	Eastern	258688	Albuquerque	NM	Mountain	E
3	45	GA	Eastern	98827	ELPaso	TX	Central	E
4	45	GA	Eastern	99518	SanAntonio	TX	Central	E
5	64	TX	Central	224427	Cleveland	OH	Eastern	
6	64	TX	Central	1751886	Austin	TX	Central	E
7	64	TX	Central	99120	Houston	TX	Central	E
8	100	NY	Eastern	120111	Jacksonville	FL	Eastern	C
9	100	NY	Eastern	91428	Indianapolis	IN	Eastern	C

- USER_STATE와 HOTEL_STATE의 지역이 같은 지에 대한 컬럼 생성

In [35]:
```
# df에서 IS_SAME_STATE 속성이 True인 것만 추출
```

```
ID_USER_SAME_STATE = df[df["IS_SAME_STATE"] == True]
```

- IS_SAME_STATE == True인 유저의 id가 담긴 ID_USER_SAME_STATE 리스트 생성

In [36]:
```
# df에서 IS_DIFF_STATE 속성이 False인 것만 추출
ID_USER_DIFF_STATE = df[df["IS_SAME_STATE"] == False]
```

- IS_SAME_STATE == False인 유저의 id가 담긴 ID_USER_DIFF_STATE 리스트 생성

In [37]:
```
# ID_USER_SAME_STATE와 ID_USER_DIFF_STATE에서 ID_USER 속성에 대해 중복된 데이터 제거
ID_USER_SAME_STATE = ID_USER_SAME_STATE.drop_duplicates(subset=["ID_USER"])
ID_USER_DIFF_STATE = ID_USER_DIFF_STATE.drop_duplicates(subset=["ID_USER"])

print(len(ID_USER_SAME_STATE['ID_USER']))
print(len(ID_USER_DIFF_STATE['ID_USER']))
```

```
521
1057
```

- 자신이 사는 주와 같은 곳으로 여행을 가본 사람은 521명이다.
- 자신이 사는 주와 다른 곳으로 여행을 가본 사람은 1057명이다.

-> 사람들은 자신이 거주한 주 이외의 주에 방문할 가능성이 높다.

<결론>

- 상관관계와 유사성을 둘 다 분석해 본 결과, 더 넓은 범위인 시간대에서는 자신이 거주한 시간대와 같은 시간대에 위치한 호텔에 방문하지만, 좁은 범위인 주에 대해서는 자신이 거주한 주와 다른 주에 위치한 호텔에 방문한다.

기타 분석

1) 호텔이 위치한 도시와 여행 유형 간의 분석

In [38]:
```
# 여행 유형별로 호텔 방문 건수 계산
hotel_trip = df.groupby(['Trip Type', 'HOTEL_CITY']).size().reset_index(name='visit_coun

# 가장 많이 방문한 여행 유형과 지역 추출
most_visited = hotel_trip.sort_values(by='visit_count', ascending=False).iloc[0]
print(f"가장 많은 방문을 기록한 여행 유형은 {most_visited['Trip Type']}이고, 그 지역은 {most_visited
```

가장 많은 방문을 기록한 여행 유형은 Business이고, 그 지역은 Houston이다.

2) 호텔이 위치한 도시와 평균 평가 점수 간의 분석

In [39]:
```
# 호텔 위치별 평균 평가 점수 계산
hotel_rating = df.groupby('HOTEL_CITY')['Rating'].mean().reset_index(name='mean_rating')

# 가장 높은 평가를 받은 지역 추출
best_rated = hotel_rating.sort_values(by='mean_rating', ascending=False).iloc[0]
print(f"가장 높은 평가를 받은 지역은 {best_rated['HOTEL_CITY']}이고, 평균 평가 점수는 {best_rated['m
```

가장 높은 평가를 받은 지역은 Boston이고, 평균 평가 점수는 4.5이다.

3) 유저 ID와 방문한 지역의 수 간의 분석

In [40]:
```
# 유저별로 방문한 지역의 종류 수 계산
user_city = df.groupby('ID_USER')['HOTEL_CITY'].nunique().reset_index(name='num_cities')
```

```
# 가장 많은 지역을 방문한 유저 추출
most_traveler = user_city.sort_values(by='num_cities', ascending=False).iloc[0]
print(f"가장 많은 지역을 방문한 유저의 ID는 {most_traveler['ID_USER']}이고, 방문한 지역의 수는 {most_t
```

가장 많은 지역을 방문한 유저의 ID는 40423이고, 방문한 지역의 수는 14이다.