택시 요금 데이터 다루기 [미니 프로젝트]

1. 데이터 불러오기, 데이터 확인

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns`

data = pd.read_csv('/aiffel/data/trip.csv')

data.head()
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	6	3.34
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	1	1.80
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	1	1.00
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1	3.70
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	1	4.37

fare_amount	tip_amount	tolls_amount
13.0	2.76	0.0
16.0	4.00	0.0
6.5	1.45	0.0
20.5	6.39	0.0
16.5	0.00	0.0

```
# Q. info() 메서드를 사용하여 데이터 컬럼명과 자료형을 확인합니다.

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22701 entries, 0 to 22700
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   passenger_name         22701 non-null  object
 1   tpep_pickup_datetime   22701 non-null  object
 2   tpep_dropoff_datetime  22701 non-null  object
 3   payment_method         22701 non-null  object
 4   passenger_count        22701 non-null  int64
 5   trip_distance          22701 non-null  float64
 6   fare_amount            22698 non-null  float64
 7   tip_amount             22701 non-null  float64
 8   tolls_amount           22701 non-null  float64
dtypes: float64(4), int64(1), object(4)
memory usage: 1.6+ MB
```

Q. describe() 메서드를 사용하여 데이터 컬럼별 통계량을 확인합니다.

data.describe()

	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
count	22701.000000	22701.000000	22698.000000	22701.000000	22701.000000
mean	1.643584	2.913400	13.024009	1.835745	0.312514
std	1.304942	3.653023	13.240074	2.800537	1.399153
min	0.000000	0.000000	-120.000000	0.000000	0.000000
25%	1.000000	0.990000	6.500000	0.000000	0.000000
50%	1.000000	1.610000	9.500000	1.350000	0.000000
75%	2.000000	3.060000	14.500000	2.450000	0.000000
max	36.000000	33.960000	999.990000	200.000000	19.100000

2. 중복 데이터 확인

Q. 중복 데이터를 확인합니다.

data[data.duplicated()]

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
17	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	1	3.6	12.5	2.85	0.0
204	Lisa Bullock	02/13/2017 4:25:41 PM	02/13/2017 4:55:35 PM	Cash	1	4.2	21.0	0.00	0.0

Q. 중복 데이터를 확인합니다.
위에서 확인한 중복 데이터의 승객명을 [[PASSENGER_NAME]] 대신 넣어주세요.

data[data['passenger_name'] == 'Sarah Gross']

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
16	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	1	3.6	12.5	2.85	0.0
17	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	1	3.6	12.5	2.85	0.0

Q. 중복 데이터를 제거합니다.

data = data.drop_duplicates()

data

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	6
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	1
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	1
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	1
...	...	...	...	...	...
22696	Austin Johnson	02/24/2017 5:37:23 PM	02/24/2017 5:40:39 PM	Cash	3
22697	Monique Williams	08/06/2017 4:43:59 PM	08/06/2017 5:24:47 PM	Cash	1
22698	Drew Graves	09/04/2017 2:54:14 PM	09/04/2017 2:58:22 PM	Debit Card	1
22699	Jonathan Copeland	07/15/2017 12:56:30 PM	07/15/2017 1:08:26 PM	Debit Card	1
22700	Benjamin Miller	03/02/2017 1:02:49 PM	03/02/2017 1:16:09 PM	Cash	1

22699 rows × 9 columns

trip_distance	fare_amount	tip_amount	tolls_amount
3.34	13.0	2.76	0.00
1.80	16.0	4.00	0.00
1.00	6.5	1.45	0.00
3.70	20.5	6.39	0.00
4.37	16.5	0.00	0.00
...	...	...	...
0.61	4.0	0.00	0.00
16.71	52.0	14.64	5.76
0.42	4.5	0.00	0.00
2.36	10.5	1.70	0.00
2.10	11.0	2.35	0.00

3. 결측치 확인

data.isna().sum()

```
passenger_name          0
tpep_pickup_datetime    0
tpep_dropoff_datetime   0
payment_method          0
passenger_count         0
trip_distance           0
fare_amount             3
tip_amount              0
tolls_amount            0
dtype: int64
```

Q. 전체 데이터 대비 결측치의 비율을 확인합니다.

data.isna().mean()

```
passenger_name          0.000000
tpep_pickup_datetime    0.000000
tpep_dropoff_datetime   0.000000
payment_method          0.000000
passenger_count         0.000000
trip_distance           0.000000
fare_amount             0.000132
tip_amount              0.000000
tolls_amount            0.000000
dtype: float64
```

```
# Q. 결측치를 제거합니다.

data = data.dropna()  # 결측치 제거 후 data 업데이트 (안 하면 처음이랑 동일값 출력)
data.isna().mean()
```

```
passenger_name          0.0
tpep_pickup_datetime    0.0
tpep_dropoff_datetime   0.0
payment_method          0.0
passenger_count         0.0
trip_distance           0.0
fare_amount             0.0
tip_amount              0.0
tolls_amount            0.0
dtype: float64
```

4. passenger_count 컬럼의 이상치 제거

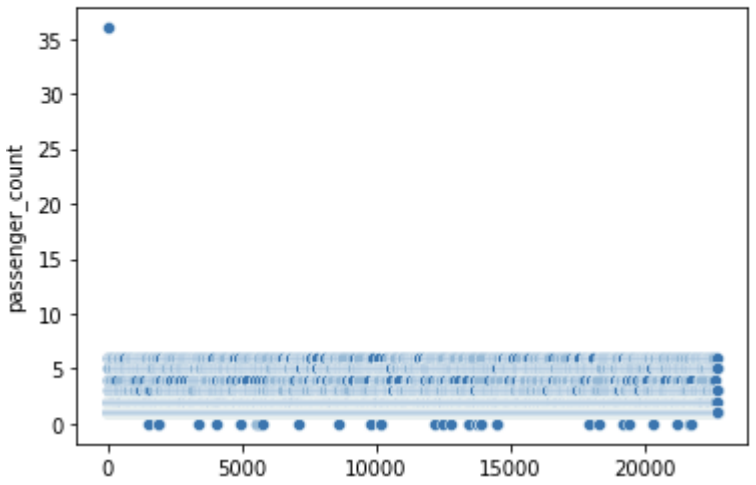
```
# passenger_count 컬럼의 값을 기준으로 정렬합니다.

data['passenger_count'].sort_values()
```

```
12804     0
19458     0
5565      0
5670      0
13718     0
          ..
416       6
4322      6
14500     6
0         6
64       36
Name: passenger_count, Length: 22696, dtype: int64
```

#passenger_count 값의 scatter plot을 그립니다.

sns.scatterplot(x = data.index, y = data['passenger_count'])

```
<AxesSubplot:ylabel='passenger_count'>
```



passenger_count 컬럼의 이상치를 제거합니다.
(passenger_count가 6을 초과하는 경우)

```
data = data[data['passenger_count'] <= 6]
```

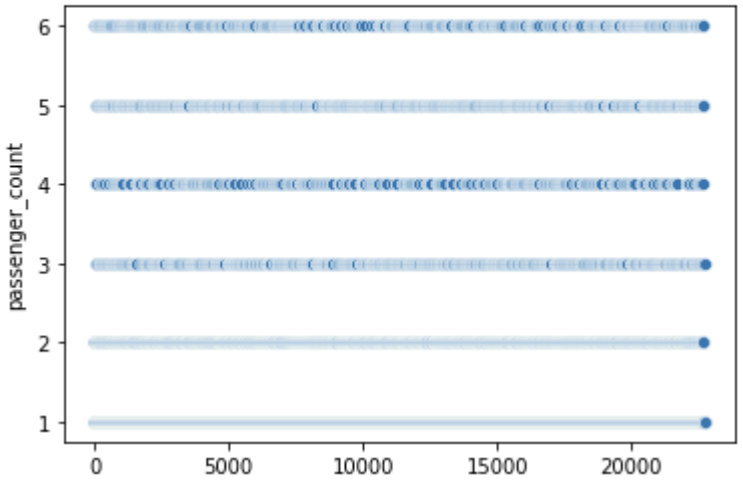
```
# passenger_count 컬럼의 이상치를 확인합니다.
# (passenger_count가 0인 경우)

len(data[data['passenger_count'] == 0])
```

```
33
```

```
# passenger_count 컬럼의 이상치를 제거합니다.

data = data[data['passenger_count'] != 0]
```

```
# passenger_count의 scatter plot을 다시 그려봅니다.

sns.scatterplot(x = data.index, y = data['passenger_count'])
```

```
<AxesSubplot:ylabel='passenger_count'>
```



5. 수치형 컬럼의 이상치 제거

1) trip_distance 이상치 제거

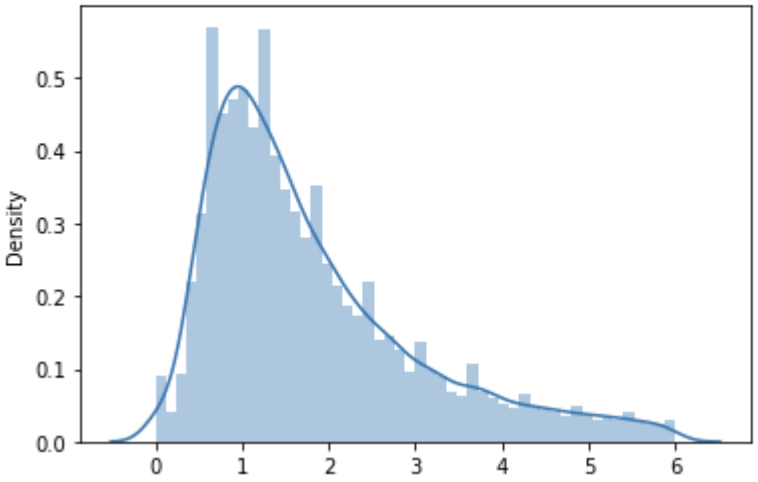
```
# Q. trip_distance의 이상치를 확인합니다.

data['trip_distance'].describe()
```

```
count    22662.000000
mean         2.912906
std          3.652999
min          0.000000
25%          0.990000
50%          1.610000
75%          3.060000
max         33.960000
Name: trip_distance, dtype: float64
```

```
# Q. trip_distance의 이상치를 제거합니다.

data = data[data['trip_distance'] <= 6]
```

```
# Q. trip_distance의 히스토그램을 그립니다.

sns.distplot(data['trip_distance'])
```

```
<AxesSubplot:xlabel='trip_distance', ylabel='Density'>
```




```
data.describe()
```

	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
count	20083.000000	20083.000000	20083.000000	20083.000000	20083.000000
mean	1.638899	1.810189	9.984990	1.401438	0.021759
std	1.280835	1.239392	9.623435	2.048460	0.391778
min	1.000000	0.000000	-120.000000	0.000000	0.000000
25%	1.000000	0.900000	6.000000	0.000000	0.000000
50%	1.000000	1.460000	8.500000	1.250000	0.000000
75%	2.000000	2.390000	12.000000	2.150000	0.000000
max	6.000000	6.000000	999.990000	200.000000	18.000000

2) fare_amount 이상치 제거

```
# Q. fare_amount의 이상치 데이터 개수를 확인합니다.
# (fare_amount가 0 이하인 경우)

data[data['fare_amount'] <= 0]['fare_amount'].count()
```

18

```
# Q. fare_amount의 이상치를 제거합니다.

data = data[data['fare_amount'] > 0]
```

```
data.sort_values('fare_amount')
```

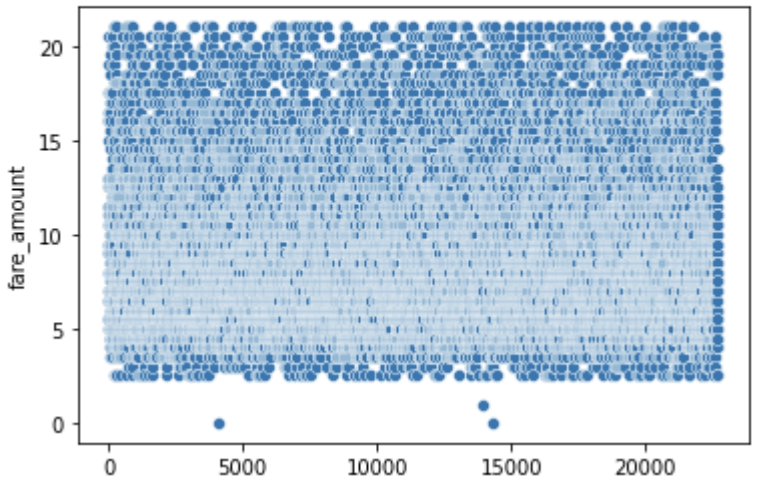
	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
14285	Mark Reed	05/03/2017 7:44:28 PM	05/03/2017 7:44:38 PM	Debit Card	1	0.0	0.01	0.00	0.00
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	4	4.5	0.01	0.00	10.50
13972	Matthew Blake	02/23/2017 9:21:25 AM	02/23/2017 9:21:57 AM	Cash	1	0.0	1.00	0.00	0.00
18699	Donna Silva	12/01/2017 12:03:08 PM	12/01/2017 12:03:13 PM	Cash	1	0.0	2.50	0.00	0.00
3564	Jeffrey Schneider	04/11/2017 12:16:28 PM	04/11/2017 12:16:34 PM	Cash	1	0.0	2.50	12.00	0.00
...	...	...	...	...	...	...	...	...	...
11271	Daniel Carrillo	06/19/2017 12:51:17 AM	06/19/2017 12:52:12 AM	Cash	2	0.0	120.00	20.00	11.52
12513	Mr. Wesley Reyes	12/17/2017 6:24:24 PM	12/17/2017 6:24:42 PM	Cash	1	0.0	175.00	46.69	11.75
15476	James Dyer MD	06/06/2017 8:55:01 PM	06/06/2017 8:55:06 PM	Debit Card	1	0.0	200.00	11.00	0.00
20314	Nicholas Thomas	12/19/2017 9:40:46 AM	12/19/2017 9:40:55 AM	Cash	2	0.0	450.00	0.00	0.00
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	1	2.6	999.99	200.00	0.00

20065 rows × 9 columns

```
# Q. fare_amount의 이상치를 제거합니다.

data = data[data['fare_amount'] <= 21]
```

```
# Q. fare_amount의 scatter plot을 그립니다.

sns.distplot(data['fare_amount'])
```

```
<AxesSubplot:ylabel='fare_amount'>
```



`# fare_amount가 150을 초과한다면 150으로 변환합니다.

```
def fare_func(x):
    if x > 150:
        return 150
    else:
        return x`
```

`data['fare_amount'].apply(fare_func)`

```
data['fare_amount'].apply(fare_func)
```

```
0        13.0
1        16.0
2         6.5
3        20.5
4        16.5
         ...
22695     7.5
22696     4.0
22698     4.5
22699    10.5
22700    11.0
Name: fare_amount, Length: 19485, dtype: float64
```

```
data['fare_amount'] = data['fare_amount'].apply(lambda x: 150 if x > 150 el
se x)
```

```
data.sort_values('fare_amount')
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	4
14285	Mark Reed	05/03/2017 7:44:28 PM	05/03/2017 7:44:38 PM	Debit Card	1
13972	Matthew Blake	02/23/2017 9:21:25 AM	02/23/2017 9:21:57 AM	Cash	1
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	1
19778	Lisa Cox	12/24/2017 4:29:28 PM	12/24/2017 4:30:23 PM	Cash	1
...	...	...	...	...	...
20775	Chris Sullivan	10/03/2017 7:05:48 AM	10/03/2017 7:28:37 AM	Credit Card	2
18843	Clayton Yang	06/09/2017 6:13:02 PM	06/09/2017 6:45:15 PM	Debit Card	1
21373	Susan Watts	09/08/2017 9:12:28 PM	09/08/2017 9:39:10 PM	Cash	1
18272	Tammie Palmer	09/09/2017 5:49:33 PM	09/09/2017 6:11:39 PM	Debit Card	1
10854	Catherine Scott	05/06/2017 4:52:41 PM	05/06/2017 5:21:27 PM	Debit Card	3

19485 rows × 9 columns

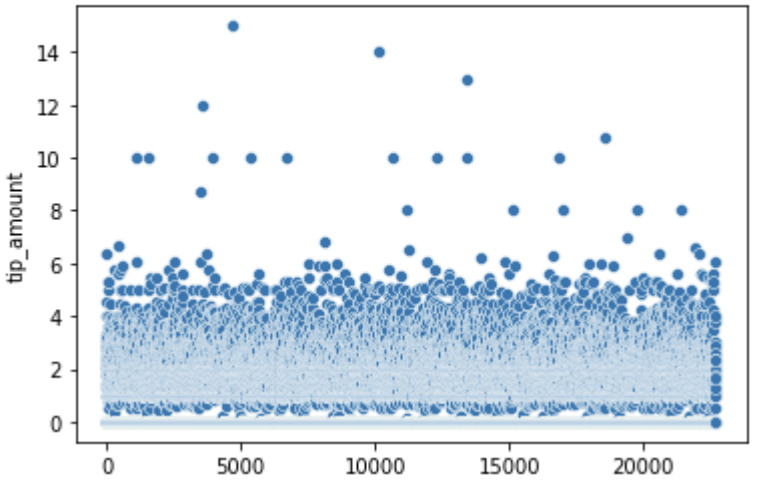
trip_distance	fare_amount	tip_amount	tolls_amount
4.50	0.01	0.00	10.5
0.00	0.01	0.00	0.0
0.00	1.00	0.00	0.0
0.02	2.50	0.00	0.0
0.07	2.50	0.00	0.0
...	...	...	...
5.74	21.00	4.36	0.0
2.60	21.00	1.50	0.0
5.20	21.00	8.00	0.0
5.77	21.00	0.00	0.0
4.69	21.00	0.00	0.0

3) tip_amount 이상치 제거

Q. tip_amount의 scatter plot을 그립니다.

sns.scatterplot(x = data.index, y = data['tip_amount'])

```
<AxesSubplot:ylabel='tip_amount'>
```



Q. tip_amount의 이상치를 확인합니다.

tip_amount 기초 통계 확인
data['tip_amount'].describe()

```
count    19485.000000
mean         1.328358
std          1.287980
min          0.000000
25%          0.000000
50%          1.250000
75%          2.080000
max         15.000000
Name: tip_amount, dtype: float64
```

```
# Q. tip_amount의 이상치를 제거합니다.

data = data[data['tip_amount'] <= 5]
```

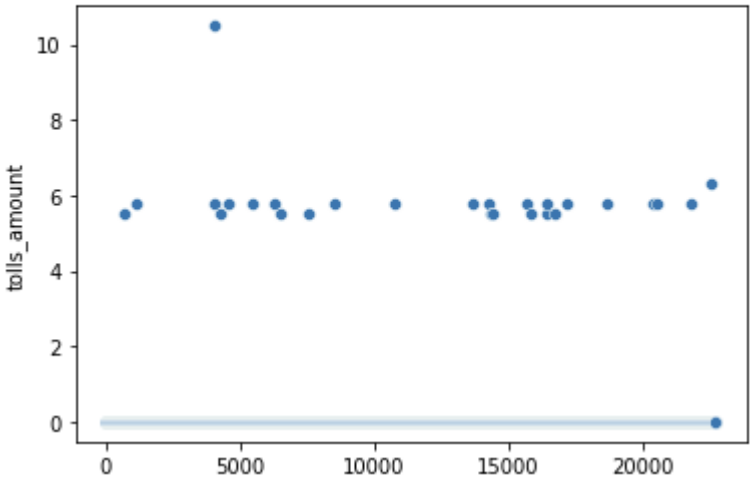
```
len(data)
```

```
19348
```

4) tolls_amount 이상치 제거

```
# Q. tolls_amount의 scatter plot을 그립니다.

sns.scatterplot(x = data.index, y = data['tolls_amount'])
```

```
<AxesSubplot:ylabel='tolls_amount'>
```



6. 범주형 데이터 전처리

1) 결제 방법: Debit Card와 Credit Card를 Card로 통합

```
data.head(30)
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	6	3.34	13.0	2.76	0.0
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	1	1.80	16.0	4.00	0.0
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	1	1.00	6.5	1.45	0.0
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	1	4.37	16.5	0.00	0.0
5	Justin Smith	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	Debit Card	6	2.30	9.0	2.06	0.0
7	Hannah Foley	08/15/2017 5:41:06 PM	08/15/2017 6:03:05 PM	Debit Card	1	2.98	16.0	1.78	0.0
8	Katie Whitney	02/04/2017 4:17:07 PM	02/04/2017 4:29:14 PM	Cash	1	1.20	9.0	0.00	0.0
9	Amanda Jones	11/10/2017 3:20:29 PM	11/10/2017 3:40:55 PM	Cash	1	1.60	13.0	2.75	0.0
10	Cory Jensen	03/04/2017 11:58:00 AM	03/04/2017 12:13:12 PM	Cash	1	1.77	11.5	2.46	0.0
12	Ryan Reyes	06/09/2017 7:00:26 PM	06/09/2017 7:20:11 PM	Debit Card	1	3.00	15.0	3.35	0.0
13	Jessica Mooney	11/06/2017 11:35:05 PM	11/06/2017 11:42:57 PM	Credit Card	1	2.39	9.5	2.16	0.0
14	Heidi May	02/22/2017 3:18:31 PM	02/22/2017 3:42:50 PM	Cash	1	3.30	17.5	4.55	0.0
15	Anthony Richard	06/02/2017 6:41:39 AM	06/02/2017 6:57:47 AM	Credit Card	1	5.93	19.0	3.00	0.0
16	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	1	3.60	12.5	2.85	0.0
18	Susan Robinson	07/10/2017 1:36:31 PM	07/10/2017 1:48:43 PM	Cash	2	1.71	9.5	0.00	0.0
19	Cynthia Mendoza	04/10/2017 6:12:58 PM	04/10/2017 6:17:39 PM	Cash	2	0.63	5.0	0.00	0.0
20	Zachary James	03/05/2017 4:01:07 AM	03/05/2017 4:14:11 AM	Credit Card	2	2.77	11.5	3.20	0.0
21	Marissa Scott	12/30/2017 11:52:44 PM	12/30/2017 11:58:57 PM	Debit Card	1	1.10	6.5	0.00	0.0
23	Krista Stewart	01/06/2017 8:12:07 PM	01/06/2017 8:18:37 PM	Cash	1	0.52	5.5	1.00	0.0
24	Mike Taylor	06/27/2017 12:08:22 AM	06/27/2017 12:13:45 AM	Credit Card	1	1.70	7.0	2.05	0.0
25	Heather Johnson	02/13/2017 10:29:33 AM	02/13/2017 10:34:11 AM	Cash	1	0.90	5.5	1.25	0.0
26	Tiffany Ramirez	01/14/2017 7:58:42 PM	01/14/2017 8:05:59 PM	Debit Card	1	1.72	8.0	2.79	0.0
27	James Taylor	11/04/2017 1:27:59 AM	11/04/2017 1:44:05 AM	Credit Card	1	2.70	13.0	2.85	0.0
28	Gabriela Bryan	11/24/2017 10:48:13 AM	11/24/2017 10:52:57 AM	Cash	1	0.85	5.5	0.00	0.0
29	Janet Hogan MD	11/22/2017 10:24:17 AM	11/22/2017 10:38:52 AM	Cash	1	2.30	11.0	2.35	0.0
31	Jessica Cohen	08/09/2017 9:01:50 PM	08/09/2017 9:14:28 PM	Cash	1	2.30	10.5	1.77	0.0
32	Katherine Martin	04/12/2017 11:07:56 AM	04/12/2017 11:19:29 AM	Cash	4	1.50	9.0	0.00	0.0
34	Adrienne Mitchell	10/26/2017 7:43:07 PM	10/26/2017 8:06:02 PM	Cash	1	2.90	15.5	3.45	0.0
35	Michael Murphy	03/04/2017 6:19:04 PM	03/04/2017 6:32:13 PM	Credit Card	1	1.70	10.5	1.70	0.0
36	Benjamin Walls	11/05/2017 1:35:25 AM	11/05/2017 1:43:39 AM	Cash	5	2.69	9.5	0.00	0.0

```
# payment_method 컬럼에 어떤 값들이 있는지 살펴봅시다.
```

```
data['payment_method'].unique()
```

```
array(['Debit Card', 'Cash', 'Credit Card'], dtype=object)
```

```
data['payment_method'].nunique()
```

```
3
```

```
data['payment_method'].value_counts()
```

```
Cash           9544
Debit Card     4931
Credit Card    4873
Name: payment_method, dtype: int64
```

```
# Q. 'Debit Card'와 'Credit Card' 항목을 'Card'로 변환합니다.
# (힌트: replace() 메서드를 사용합니다.)

data['payment_method'] = data['payment_method'].replace({'Debit Card': 'Card', 'Credit Card': 'Card'})
```

```
data['payment_method'].value_counts()
```

```
Card    9804
Cash    9544
Name: payment_method, dtype: int64
```

2) 승객명: 성과 이름을 분리하여 성 부분만 저장

```
example = 'Susan Robinson'
example.split()
```

```
['Susan', 'Robinson']
```

```
# Q. passenger_name을 성과 이름으로 분리하여 성 부분만 passenger_first_nam
e 컬럼으로 저장합니다.

data['passenger_first_name'] = data['passenger_name'].str.split().apply(la
mbda x: x[-1])

data
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Card	6
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Card	1
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Card	1
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Card	1
5	Justin Smith	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	Card	6
...	...	...	...	...	...
22695	Patrick Williams	08/10/2017 10:20:04 PM	08/10/2017 10:29:31 PM	Cash	1
22696	Austin Johnson	02/24/2017 5:37:23 PM	02/24/2017 5:40:39 PM	Cash	3
22698	Drew Graves	09/04/2017 2:54:14 PM	09/04/2017 2:58:22 PM	Card	1
22699	Jonathan Copeland	07/15/2017 12:56:30 PM	07/15/2017 1:08:26 PM	Card	1
22700	Benjamin Miller	03/02/2017 1:02:49 PM	03/02/2017 1:16:09 PM	Cash	1

19348 rows × 10 columns

trip_distance	fare_amount	tip_amount	tolls_amount	passenger_first_name
3.34	13.0	2.76	0.0	Duffy
1.80	16.0	4.00	0.0	Foster
1.00	6.5	1.45	0.0	Combs
4.37	16.5	0.00	0.0	Johnson
2.30	9.0	2.06	0.0	Smith
...	...	...	...	...
0.89	7.5	1.76	0.0	Williams
0.61	4.0	0.00	0.0	Johnson
0.42	4.5	0.00	0.0	Graves
2.36	10.5	1.70	0.0	Copeland
2.10	11.0	2.35	0.0	Miller

3) 택시 탑승, 하차 시간을 활용

```
data.head()
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Card	6	3.34
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Card	1	1.80
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Card	1	1.00
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Card	1	4.37
5	Justin Smith	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	Card	6	2.30

fare_amount	tip_amount	tolls_amount	passenger_first_name
13.0	2.76	0.0	Duffy
16.0	4.00	0.0	Foster
6.5	1.45	0.0	Combs
16.5	0.00	0.0	Johnson
9.0	2.06	0.0	Smith

data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19348 entries, 0 to 22700
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   passenger_name        19348 non-null  object
 1   tpep_pickup_datetime  19348 non-null  object
 2   tpep_dropoff_datetime 19348 non-null  object
 3   payment_method        19348 non-null  object
 4   passenger_count       19348 non-null  int64
 5   trip_distance         19348 non-null  float64
 6   fare_amount           19348 non-null  float64
 7   tip_amount            19348 non-null  float64
 8   tolls_amount          19348 non-null  float64
 9   passenger_first_name  19348 non-null  object
dtypes: float64(4), int64(1), object(5)
memory usage: 1.6+ MB
```

Q. tpep_pickup_datetime 컬럼의 object 자료형을 datetime으로 변환합니다.

data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])

Q. tpep_dropoff_datetime 컬럼의 object 자료형을 datetime으로 변환합니다.

```
data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19348 entries, 0 to 22700
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   passenger_name         19348 non-null  object
 1   tpep_pickup_datetime   19348 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  19348 non-null  datetime64[ns]
 3   payment_method         19348 non-null  object
 4   passenger_count        19348 non-null  int64
 5   trip_distance          19348 non-null  float64
 6   fare_amount            19348 non-null  float64
 7   tip_amount             19348 non-null  float64
 8   tolls_amount           19348 non-null  float64
 9   passenger_first_name   19348 non-null  object
dtypes: datetime64[ns](2), float64(4), int64(1), object(3)
memory usage: 1.6+ MB
```

```
`# Q. 하차 시각과 승차 시각의 차이를 travel_time 컬럼으로 저장합니다.

data['travel_time'] = (data['tpep_dropoff_datetime'] - data['tpep_pickup_datetime']).dt.total_seconds() / 60`
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	6	3.34
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	1	1.80
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	1	1.00
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	1	4.37
5	Justin Smith	2017-03-25 20:34:11	2017-03-25 20:42:11	Card	6	2.30

fare_amount	tip_amount	tolls_amount	passenger_first_name	travel_time
13.0	2.76	0.0	Duffy	14.066667
16.0	4.00	0.0	Foster	26.500000
6.5	1.45	0.0	Combs	7.200000
16.5	0.00	0.0	Johnson	16.716667
9.0	2.06	0.0	Smith	8.000000

data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19348 entries, 0 to 22700
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   passenger_name         19348 non-null  object
 1   tpep_pickup_datetime   19348 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  19348 non-null  datetime64[ns]
 3   payment_method         19348 non-null  object
 4   passenger_count        19348 non-null  int64
 5   trip_distance          19348 non-null  float64
 6   fare_amount            19348 non-null  float64
 7   tip_amount             19348 non-null  float64
 8   tolls_amount           19348 non-null  float64
 9   passenger_first_name   19348 non-null  object
 10  travel_time            19348 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(1), object(3)
memory usage: 1.8+ MB
```

Q. travel_time 컬럼의 데이터를 초 단위로 변환합니다.

data['travel_time_sec'] = data['travel_time'] * 60

4) feature engineering 맛보기

data.head()

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	6	3.34
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	1	1.80
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	1	1.00
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	1	4.37
5	Justin Smith	2017-03-25 20:34:11	2017-03-25 20:42:11	Card	6	2.30

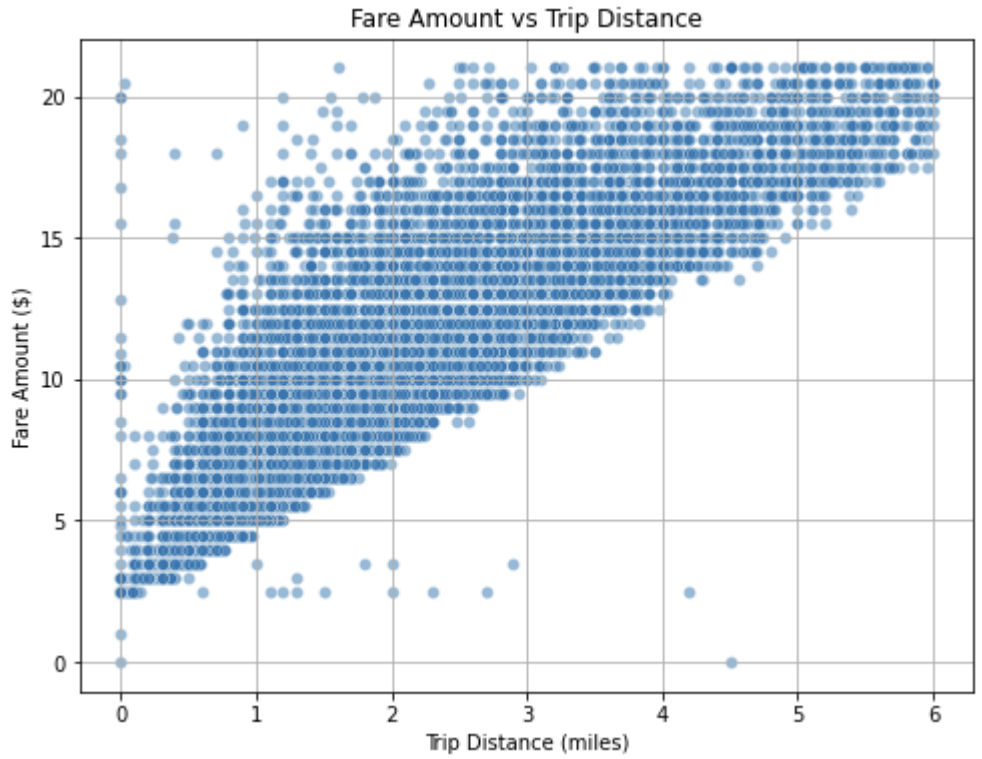
fare_amount	tip_amount	tolls_amount	passenger_first_name	travel_time	travel_time_sec
13.0	2.76	0.0	Duffy	14.066667	844.0
16.0	4.00	0.0	Foster	26.500000	1590.0
6.5	1.45	0.0	Combs	7.200000	432.0
16.5	0.00	0.0	Johnson	16.716667	1003.0
9.0	2.06	0.0	Smith	8.000000	480.0

`# Q. 승객이 지불한 총 요금을 total_amount 컬럼으로 저장합니다.

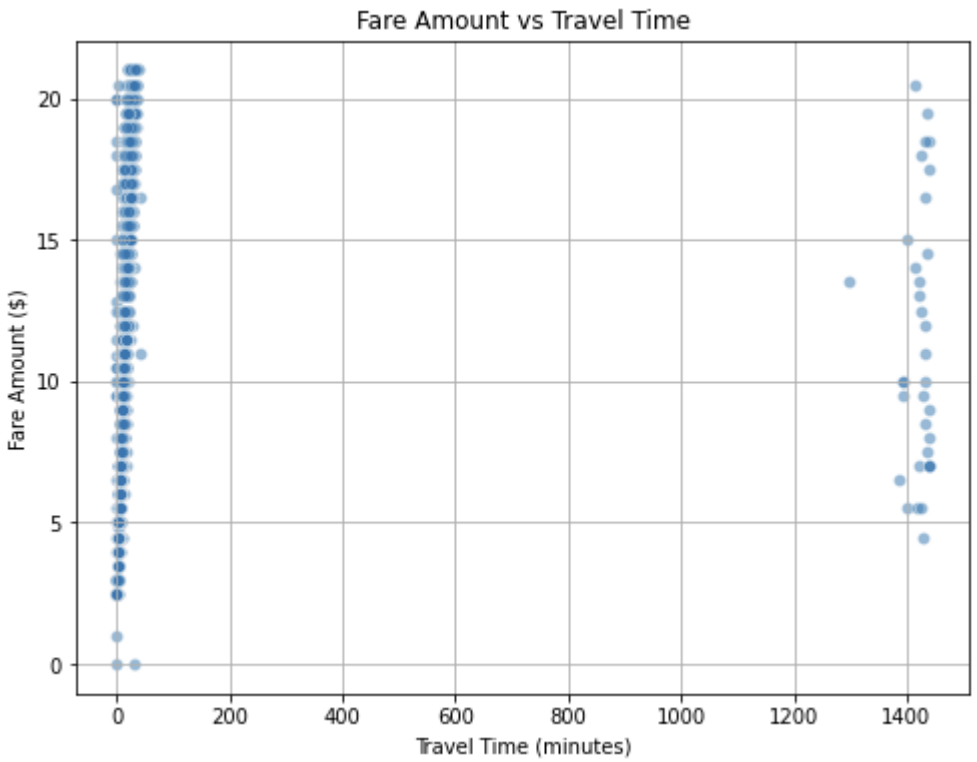
data['total_amount'] = data['fare_amount'] + data['tip_amount'] + data['tolls_amount']`

Q. fare_amount와 trip_distance 사이의 관계를 scatter plot으로 표현합니다.

```
# 산점도 생성
plt.figure(figsize=(8,6))
sns.scatterplot(x='trip_distance', y='fare_amount', data=data, alpha=0.5)

# 그래프 제목과 축 이름
plt.title('Fare Amount vs Trip Distance')
plt.xlabel('Trip Distance (miles)')
plt.ylabel('Fare Amount ($)')
plt.grid(True)
plt.show()
```

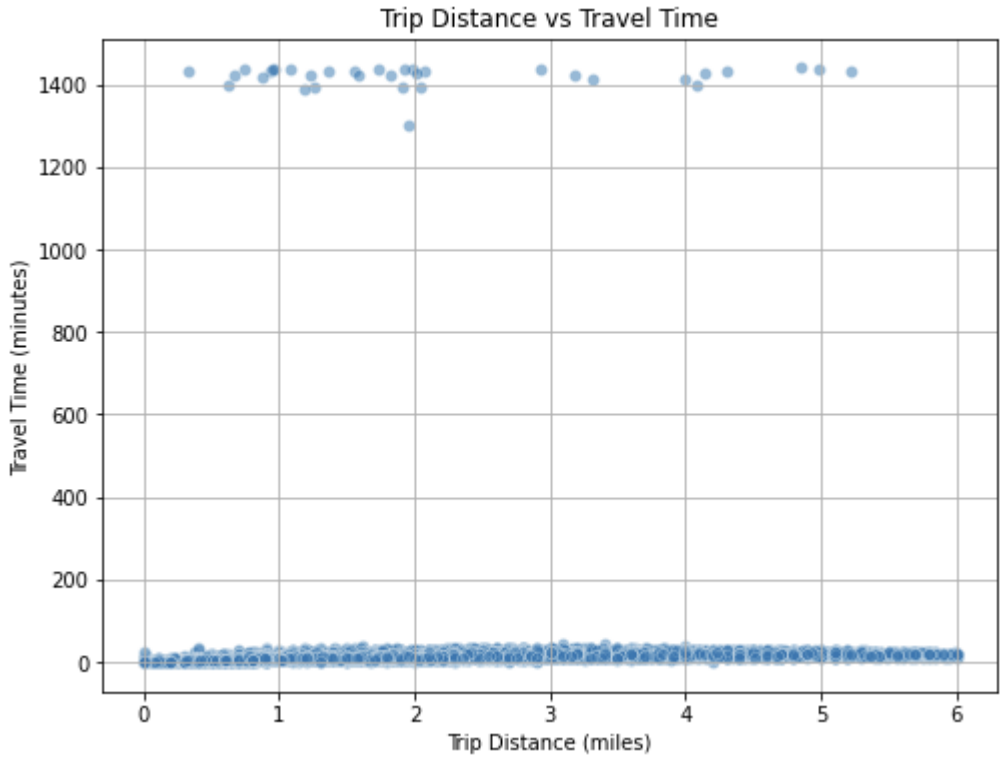


```
# Q. fare_amount와 travel_time 사이의 관계를 scatter plot으로 표현합니다.

# 산점도 생성
plt.figure(figsize=(8,6))
sns.scatterplot(x='travel_time', y='fare_amount', data=data, alpha=0.5)

# 그래프 제목과 축 이름
plt.title('Fare Amount vs Travel Time')
plt.xlabel('Travel Time (minutes)')
plt.ylabel('Fare Amount ($)')
plt.grid(True)
plt.show()
```




```
# Q. trip_distance와 travel_time 사이의 관계를 scatter plot으로 표현합니다.

# 산점도 생성
plt.figure(figsize=(8,6))
```

```
sns.scatterplot(x='trip_distance', y='travel_time', data=data, alpha=0.5)

# 그래프 제목과 축 이름
plt.title('Trip Distance vs Travel Time')
plt.xlabel('Trip Distance (miles)')
plt.ylabel('Travel Time (minutes)')
plt.grid(True)
plt.show()
```




```
# Q. scatter plot으로 관찰된 travel_time의 이상치를 제거합니다.

# IQR 계산
Q1 = data['travel_time'].quantile(0.25)
Q3 = data['travel_time'].quantile(0.75)
IQR = Q3 - Q1

# 상/하한 설정 (1.5 IQR 기준)
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
# 이상치 제거
data_clean = data[(data['travel_time'] >= lower_bound) & (data['travel_tim
e'] <= upper_bound)]

# 제거 후 산점도 확인
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,6))
sns.scatterplot(x='trip_distance', y='travel_time', data=data_clean, alpha=0.
5)
plt.title('Trip Distance vs Travel Time (Outliers Removed)')
plt.xlabel('Trip Distance (miles)')
plt.ylabel('Travel Time (minutes)')
plt.grid(True)
plt.show()
```