

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: data = pd.read_csv('~/.data/trip.csv')
```

```
In [4]: data.head()
```

```
Out[4]:
```

	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
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1	3.70	20.5	6.39	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

```
In [5]: # 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
```

```
In [6]: # Q. describe() 메서드를 사용하여 데이터 컬럼별 통계량을 확인합니다.
```

```
data.describe()
```

```
Out[6]:
```

	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

```
In [7]: # Q. 중복 데이터를 확인합니다.
```

```
data[data.duplicated()]
```

```
Out[7]:
```

	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

```
In [9]: # Q. 중복 데이터를 확인합니다.
# 위에서 확인한 중복 데이터의 승객명을 [[PASSENGER_NAME]] 대신 넣어주세요.
```

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

```
Out[9]:
```

	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

```
In [10]: # Q. 중복 데이터를 제거합니다.
```

```
data = data.drop_duplicates()
```

```
In [11]: data
```

Out[11]:

	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	0.00
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	0.00
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	0.00
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1	3.70	20.5	6.39	0.00
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	0.00
...
22696	Austin Johnson	02/24/2017 5:37:23 PM	02/24/2017 5:40:39 PM	Cash	3	0.61	4.0	0.00	0.00
22697	Monique Williams	08/06/2017 4:43:59 PM	08/06/2017 5:24:47 PM	Cash	1	16.71	52.0	14.64	5.76
22698	Drew Graves	09/04/2017 2:54:14 PM	09/04/2017 2:58:22 PM	Debit Card	1	0.42	4.5	0.00	0.00
22699	Jonathan Copeland	07/15/2017 12:56:30 PM	07/15/2017 1:08:26 PM	Debit Card	1	2.36	10.5	1.70	0.00
22700	Benjamin Miller	03/02/2017 1:02:49 PM	03/02/2017 1:16:09 PM	Cash	1	2.10	11.0	2.35	0.00

22699 rows × 9 columns

In [12]: data.isna().sum()

Out[12]:

```
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
```

In [13]: # Q. 전체 데이터 대비 결측치의 비율을 확인합니다.

data.isna().mean()

Out[13]:

```
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
```

In [14]: # Q. 결측치를 제거합니다.

data = data.dropna()

In [15]: data.isna().mean()

Out[15]:

```
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
```

In [16]: # passenger_count 컬럼의 값을 기준으로 정렬합니다.

data['passenger_count'].sort_values()

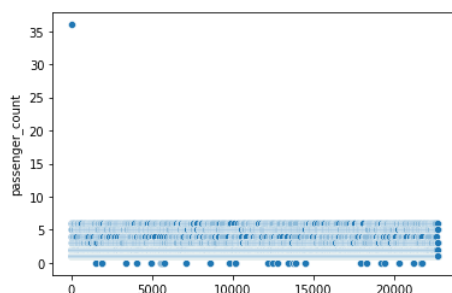
Out[16]:

```
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
```

In [17]: # passenger_count 값의 scatter plot을 그립니다.

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

Out[17]: <AxesSubplot: ylabel='passenger_count'>



```
In [18]: # passenger_count 컬럼의 이상치를 제거합니다.
# (passenger_count가 6을 초과하는 경우)

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

```
In [19]: # passenger_count 컬럼의 이상치를 확인합니다.
# (passenger_count가 0인 경우)

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

```
Out[19]: 33
```

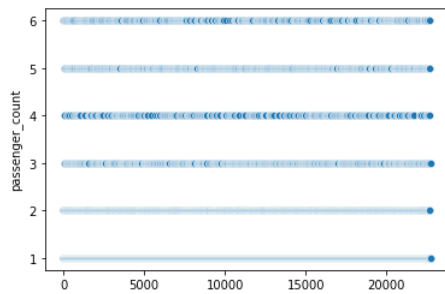
```
In [20]: # passenger_count 컬럼의 이상치를 제거합니다.

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

```
In [21]: # passenger_count의 scatter plot을 다시 그려봅니다.

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

```
Out[21]: <AxesSubplot:ylabel='passenger_count'>
```



```
In [22]: # Q. trip_distance의 이상치를 확인합니다.

data['trip_distance'] == 0
```

```
Out[22]: 0      False
1      False
2      False
3      False
4      False
...
22696  False
22697  False
22698  False
22699  False
22700  False
Name: trip_distance, Length: 22662, dtype: bool
```

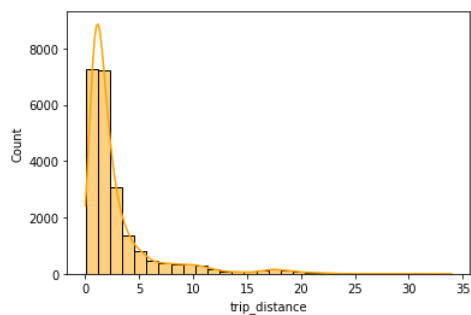
```
In [23]: # Q. trip_distance의 이상치를 제거합니다.

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

```
In [24]: # Q. trip_distance의 히스토그램을 그립니다.

sns.histplot(data['trip_distance'], bins=30, kde=True, color='orange')
```

```
Out[24]: <AxesSubplot:xlabel='trip_distance', ylabel='Count'>
```



```
In [25]: data.describe()
```

```
Out[25]:
```

	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
count	22515.000000	22515.000000	22515.000000	22515.000000	22515.000000
mean	1.645969	2.931924	12.958055	1.829513	0.309625
std	1.285783	3.657290	12.701799	2.767054	1.387300
min	1.000000	0.010000	-120.000000	0.000000	0.000000
25%	1.000000	1.000000	6.500000	0.000000	0.000000
50%	1.000000	1.630000	9.500000	1.360000	0.000000
75%	2.000000	3.090000	14.500000	2.450000	0.000000
max	6.000000	33.960000	999.990000	200.000000	19.100000

```
In [26]: # Q. fare_amount의 이상치 데이터 개수를 확인합니다.
# (fare_amount가 0 이하인 경우)

len(data[data['fare_amount'] <= 0])
```

Out[26]: 16

In [27]: # Q. fare_amount의 이상치를 제거합니다.

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

In [28]: 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
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	4	4.50	0.01	0.00	10.50
14470	Leah Carrillo	09/09/2017 1:29:37 PM	09/09/2017 1:29:57 PM	Credit Card	3	0.02	2.50	0.00	0.00
2987	Christine Harper	11/24/2017 4:32:18 AM	11/24/2017 4:32:23 AM	Credit Card	1	0.01	2.50	0.00	0.00
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	1	0.02	2.50	0.00	0.00
6702	Yvonne Brooks	08/26/2017 7:33:22 AM	08/26/2017 7:34:18 AM	Debit Card	1	0.10	2.50	0.00	0.00
...
16381	Erica Hernandez	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	Cash	1	25.50	140.00	0.00	16.26
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	2	33.96	150.00	0.00	0.00
3584	Matthew Chavez	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	Credit Card	1	7.30	152.00	0.00	0.00
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	1	33.92	200.01	51.64	5.76
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	1	2.60	999.99	200.00	0.00

22499 rows × 9 columns

In [29]: # Q. fare_amount의 이상치를 제거합니다.

#data = data[data['fare_amount'] <= 201]

data.sort_values('fare_amount')

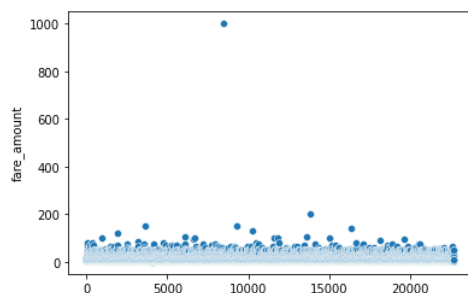
	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	4	4.50	0.01	0.00	10.50
14470	Leah Carrillo	09/09/2017 1:29:37 PM	09/09/2017 1:29:57 PM	Credit Card	3	0.02	2.50	0.00	0.00
2987	Christine Harper	11/24/2017 4:32:18 AM	11/24/2017 4:32:23 AM	Credit Card	1	0.01	2.50	0.00	0.00
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	1	0.02	2.50	0.00	0.00
6702	Yvonne Brooks	08/26/2017 7:33:22 AM	08/26/2017 7:34:18 AM	Debit Card	1	0.10	2.50	0.00	0.00
...
16381	Erica Hernandez	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	Cash	1	25.50	140.00	0.00	16.26
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	2	33.96	150.00	0.00	0.00
3584	Matthew Chavez	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	Credit Card	1	7.30	152.00	0.00	0.00
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	1	33.92	200.01	51.64	5.76
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	1	2.60	999.99	200.00	0.00

22499 rows × 9 columns

In [30]: # Q. fare_amount의 scatter plot을 그립니다.

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

Out[30]: <AxesSubplot:ylabel='fare_amount'>



In [31]: # fare_amount가 150을 초과한다면 150으로 변환합니다.

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

In [32]: data['fare_amount'].apply(fare_func)

```
Out[32]:
0      13.0
1      16.0
2       6.5
3      20.5
4      16.5
...
22696   4.0
22697  52.0
22698   4.5
22699  10.5
22700  11.0
Name: fare_amount, Length: 22499, dtype: float64
```

```
In [33]: data['fare_amount'] = data['fare_amount'].apply(lambda x: 150 if x > 150 else x)
```

```
In [34]: data.sort_values('fare_amount')
```

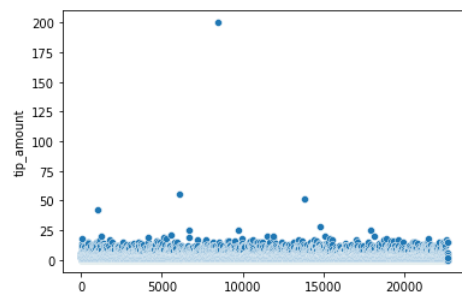
```
Out[34]:
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	4	4.50	0.01	0.00	10.50
14470	Leah Carrillo	09/09/2017 1:29:37 PM	09/09/2017 1:29:57 PM	Credit Card	3	0.02	2.50	0.00	0.00
2987	Christine Harper	11/24/2017 4:32:18 AM	11/24/2017 4:32:23 AM	Credit Card	1	0.01	2.50	0.00	0.00
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	1	0.02	2.50	0.00	0.00
6702	Yvonne Brooks	08/26/2017 7:33:22 AM	08/26/2017 7:34:18 AM	Debit Card	1	0.10	2.50	0.00	0.00
...
16381	Erica Hernandez	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	Cash	1	25.50	140.00	0.00	16.26
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	1	33.92	150.00	51.64	5.76
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	2	33.96	150.00	0.00	0.00
3584	Matthew Chavez	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	Credit Card	1	7.30	150.00	0.00	0.00
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	1	2.60	150.00	200.00	0.00

22499 rows × 9 columns

```
In [35]: # Q. tip_amount의 scatter plot을 그립니다.
sns.scatterplot(x = data.index, y = data['tip_amount'])
```

```
Out[35]: <AxesSubplot:ylabel='tip_amount'>
```



```
In [36]: # Q. tip_amount의 이상치를 확인합니다.
```

```
data[data['tip_amount'] >= 40]
```

```
Out[36]:
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
986	Elaine Horton	08/23/2017 6:23:26 PM	08/23/2017 7:18:29 PM	Cash	1	16.70	52.0	42.29	0.00
6066	Tina Knight	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	Debit Card	1	32.72	107.0	55.50	16.26
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	1	2.60	150.0	200.00	0.00
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	1	33.92	150.0	51.64	5.76

```
In [37]: # Q. tip_amount의 이상치를 제거합니다.
```

```
data = data[data['tip_amount'] <= 40]
```

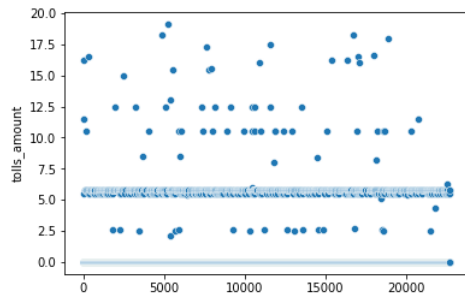
```
In [38]: len(data)
```

```
Out[38]: 22495
```

```
In [39]: # Q. tolls_amount의 scatter plot을 그립니다.
```

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

```
Out[39]: <AxesSubplot:ylabel='tolls_amount'>
```



In [40]: `data.head(30)`

Out[40]:

	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	13.00	2.76	0.00
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	1	1.80	16.00	4.00	0.00
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	1	1.00	6.50	1.45	0.00
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1	3.70	20.50	6.39	0.00
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	1	4.37	16.50	0.00	0.00
5	Justin Smith	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	Debit Card	6	2.30	9.00	2.06	0.00
6	Tonya Moreno	05/03/2017 7:04:09 PM	05/03/2017 8:03:47 PM	Cash	1	12.83	47.50	9.86	0.00
7	Hannah Foley	08/15/2017 5:41:06 PM	08/15/2017 6:03:05 PM	Debit Card	1	2.98	16.00	1.78	0.00
8	Katie Whitney	02/04/2017 4:17:07 PM	02/04/2017 4:29:14 PM	Cash	1	1.20	9.00	0.00	0.00
9	Amanda Jones	11/10/2017 3:20:29 PM	11/10/2017 3:40:55 PM	Cash	1	1.60	13.00	2.75	0.00
10	Cory Jensen	03/04/2017 11:58:00 AM	03/04/2017 12:13:12 PM	Cash	1	1.77	11.50	2.46	0.00
11	Jamie Brown	03/05/2017 7:15:30 PM	03/05/2017 7:52:18 PM	Debit Card	2	18.90	52.00	14.58	5.54
12	Ryan Reyes	06/09/2017 7:00:26 PM	06/09/2017 7:20:11 PM	Debit Card	1	3.00	15.00	3.35	0.00
13	Jessica Mooney	11/06/2017 11:35:05 PM	11/06/2017 11:42:57 PM	Credit Card	1	2.39	9.50	2.16	0.00
14	Heidi May	02/22/2017 3:18:31 PM	02/22/2017 3:42:50 PM	Cash	1	3.30	17.50	4.55	0.00
15	Anthony Richard	06/02/2017 6:41:39 AM	06/02/2017 6:57:47 AM	Credit Card	1	5.93	19.00	3.00	0.00
16	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	1	3.60	12.50	2.85	0.00
18	Susan Robinson	07/10/2017 1:36:31 PM	07/10/2017 1:48:43 PM	Cash	2	1.71	9.50	0.00	0.00
19	Cynthia Mendoza	04/10/2017 6:12:58 PM	04/10/2017 6:17:39 PM	Cash	2	0.63	5.00	0.00	0.00
20	Zachary James	03/05/2017 4:01:07 AM	03/05/2017 4:14:11 AM	Credit Card	2	2.77	11.50	3.20	0.00
21	Marissa Scott	12/30/2017 11:52:44 PM	12/30/2017 11:58:57 PM	Debit Card	1	1.10	6.50	0.00	0.00
22	Jacqueline Mclean DVM	10/11/2017 12:34:49 PM	10/11/2017 1:22:38 PM	Debit Card	1	12.30	68.25	12.00	16.26
23	Krista Stewart	01/06/2017 8:12:07 PM	01/06/2017 8:18:37 PM	Cash	1	0.52	5.50	1.00	0.00
24	Mike Taylor	06/27/2017 12:08:22 AM	06/27/2017 12:13:45 AM	Credit Card	1	1.70	7.00	2.05	0.00
25	Heather Johnson	02/13/2017 10:29:33 AM	02/13/2017 10:34:11 AM	Cash	1	0.90	5.50	1.25	0.00
26	Tiffany Ramirez	01/14/2017 7:58:42 PM	01/14/2017 8:05:59 PM	Debit Card	1	1.72	8.00	2.79	0.00
27	James Taylor	11/04/2017 1:27:59 AM	11/04/2017 1:44:05 AM	Credit Card	1	2.70	13.00	2.85	0.00
28	Gabriela Bryan	11/24/2017 10:48:13 AM	11/24/2017 10:52:57 AM	Cash	1	0.85	5.50	0.00	0.00
29	Janet Hogan MD	11/22/2017 10:24:17 AM	11/22/2017 10:38:52 AM	Cash	1	2.30	11.00	2.35	0.00
30	David Burton	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	Credit Card	1	30.83	80.00	18.56	11.52

In [41]: `# payment_method 컬럼에 어떤 값들이 있는지 살펴봅시다.`

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

Out[41]: `array(['Debit Card', 'Cash', 'Credit Card'], dtype=object)`

In [42]: `data['payment_method'].nunique()`

Out[42]: `3`

In [43]: `data['payment_method'].value_counts()`

```
Out[43]: Cash      11094
Debit Card  5729
Credit Card 5672
Name: payment_method, dtype: int64
```

In [44]: `# 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'})
```

In [45]: `data['payment_method'].value_counts()`

```
Out[45]: Card      11401
Cash      11094
Name: payment_method, dtype: int64
```

In [46]: `example = 'Susan Robinson'`

In [47]: `example.split()`

Out[47]: ['Susan', 'Robinson']

In [48]: # Q. passenger_name을 성 과 이름으로 분리하여 성 부분만 passenger_first_name 컬럼으로 저장합니다.
data['passenger_first_name'] = data['passenger_name'].str.split().apply(lambda x: x[0])

택시 탑승, 하차 시간을 활용해보시다.

In [49]: data.head()

Out[49]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount	passenger_first_name
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Card	6	3.34	13.0	2.76	0.0	Pamela
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Card	1	1.80	16.0	4.00	0.0	Michelle
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Card	1	1.00	6.5	1.45	0.0	Tina
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	1	3.70	20.5	6.39	0.0	Anthony
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Card	1	4.37	16.5	0.00	0.0	Brianna

In [50]: data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   passenger_name        22495 non-null object
1   tpep_pickup_datetime  22495 non-null object
2   tpep_dropoff_datetime 22495 non-null object
3   payment_method        22495 non-null object
4   passenger_count       22495 non-null int64
5   trip_distance         22495 non-null float64
6   fare_amount           22495 non-null float64
7   tip_amount            22495 non-null float64
8   tolls_amount          22495 non-null float64
9   passenger_first_name  22495 non-null object
dtypes: float64(4), int64(1), object(5)
memory usage: 1.9+ MB
```

In [51]: # Q. tpep_pickup_datetime 컬럼의 object 자료형을 datetime으로 변환합니다.

```
from datetime import datetime

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

In [52]: # Q. tpep_dropoff_datetime 컬럼의 object 자료형을 datetime으로 변환합니다.

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

In [53]: data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   passenger_name        22495 non-null object
1   tpep_pickup_datetime  22495 non-null datetime64[ns]
2   tpep_dropoff_datetime 22495 non-null datetime64[ns]
3   payment_method        22495 non-null object
4   passenger_count       22495 non-null int64
5   trip_distance         22495 non-null float64
6   fare_amount           22495 non-null float64
7   tip_amount            22495 non-null float64
8   tolls_amount          22495 non-null float64
9   passenger_first_name  22495 non-null object
dtypes: datetime64[ns](2), float64(4), int64(1), object(3)
memory usage: 1.9+ MB
```

In [54]: # Q. 하차 시각과 승차 시각의 차이를 travel_time 컬럼으로 저장합니다.

```
data['travel_time'] = data['tpep_dropoff_datetime'] - data['tpep_pickup_datetime']
```

In [55]: data.head()

Out[55]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount	passenger_first_name
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	6	3.34	13.0	2.76	0.0	Pamela
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	1	1.80	16.0	4.00	0.0	Michelle
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	1	1.00	6.5	1.45	0.0	Tina
3	Anthony Ray	2017-05-07 13:17:59	2017-05-07 13:48:14	Cash	1	3.70	20.5	6.39	0.0	Anthony
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	1	4.37	16.5	0.00	0.0	Brianna

In [56]: data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   passenger_name         22495 non-null  object
1   tpep_pickup_datetime   22495 non-null  datetime64[ns]
2   tpep_dropoff_datetime  22495 non-null  datetime64[ns]
3   payment_method         22495 non-null  object
4   passenger_count        22495 non-null  int64
5   trip_distance          22495 non-null  float64
6   fare_amount            22495 non-null  float64
7   tip_amount             22495 non-null  float64
8   tolls_amount           22495 non-null  float64
9   passenger_first_name   22495 non-null  object
10  travel_time            22495 non-null  timedelta64[ns]
dtypes: datetime64[ns](2), float64(4), int64(1), object(3), timedelta64[ns](1)
memory usage: 2.1+ MB
```

```
In [57]: # Q. travel_time 컬럼의 데이터를 초 단위로 변환합니다.

data['travel_time'] = data['travel_time'].dt.total_seconds()
```

보너스 (feature engineering 맛보기)

```
In [58]: data.head()
```

```
Out[58]:
```

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount	passenger_first_name
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	6	3.34	13.0	2.76	0.0	Pamela
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	1	1.80	16.0	4.00	0.0	Michelle
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	1	1.00	6.5	1.45	0.0	Tina
3	Anthony Ray	2017-05-07 13:17:59	2017-05-07 13:48:14	Cash	1	3.70	20.5	6.39	0.0	Anthony
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	1	4.37	16.5	0.00	0.0	Brianna

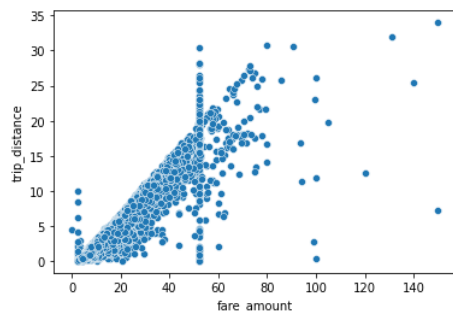
```
In [59]: # Q. 승객이 지불한 총 요금을 total_amount 컬럼으로 저장합니다.

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

```
In [60]: # Q. fare_amount와 trip_distance 사이의 관계를 scatter plot으로 표현합니다.

# sns.scatterplot(x = data['fare_amount'], y = data['trip_distance'])
sns.scatterplot(x='fare_amount', y='trip_distance', data=data)
```

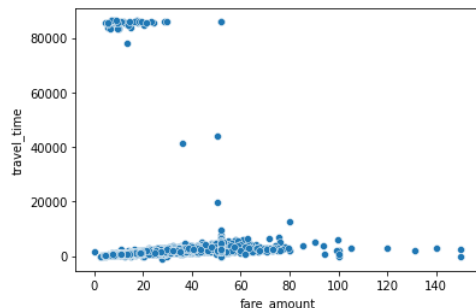
```
Out[60]: <AxesSubplot:xlabel='fare_amount', ylabel='trip_distance'>
```



```
In [61]: # Q. fare_amount와 travel_time 사이의 관계를 scatter plot으로 표현합니다.

sns.scatterplot(x='fare_amount', y='travel_time', data=data)
```

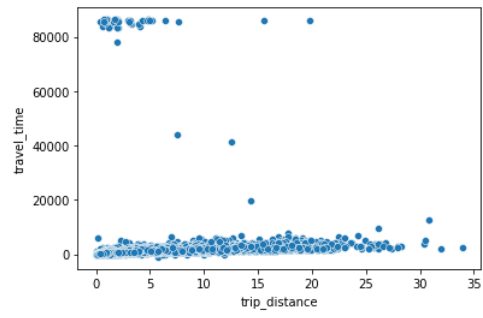
```
Out[61]: <AxesSubplot:xlabel='fare_amount', ylabel='travel_time'>
```



```
In [62]: # Q. trip_distance와 travel_time 사이의 관계를 scatter plot으로 표현합니다.

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

```
Out[62]: <AxesSubplot:xlabel='trip_distance', ylabel='travel_time'>
```

```
In [63]: # Q. scatter plot으로 관찰된 travel_time의 이상치를 제거합니다.
```

```
data = data[data['travel_time'] < 60000]
```

```
In [ ]:
```