Explanatory Data Analysis

Masalah/isu: Ketidaksesuaian biaya trip Uber yang ditagihkan ke customer.

Bukti masalah:

1. Relasi abnormal antara durasi trip dan jarak tempuh trip.
2. Waktu tunggu driver yang terlampau lama.
3. Kecepatan rata-rata trip yang terlampau cepat ataupun lambat.

In [1]:
```
import pandas as pd
import os
from datetime import date, datetime
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
sns.set()
```

In [2]:
```
# OS dependent path
# load bog, mex, and uio csv
bog_path = os.path.join("dataset", "bog_clean.csv")
mex_path = os.path.join("dataset", "mex_clean.csv")
equ_path = os.path.join("dataset", "uio_clean.csv")
```

In [3]:
```
# read dataset
bog_df = pd.read_csv(bog_path)
mex_df = pd.read_csv(mex_path)
equ_df = pd.read_csv(equ_path)
```

In [4]:
```
bog_df.tail(3)
```

Out[4]:

	id	vendor_id	pickup_datetime	dropoff_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fw
3060	3061	Bogotá	2016-10-27 12:28:22	2016-10-27 01:16:58	-74.061556	4.709213	-74.042396	4.708566	
3061	3062	Bogotá	2016-10-27 07:40:49	2016-10-27 09:08:09	-74.050934	4.752078	-74.050875	4.752123	
3062	3063	Bogotá	2016-10-26 04:27:39	2016-10-28 06:50:28	-74.052223	4.705252	-74.050725	4.714622	

In [5]:
```
mex_df.tail(3)
```

Out[5]:

	id	vendor_id	pickup_datetime	dropoff_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_
12691	12692	México DF Taxi Libre	2016-10-27 11:19:44	2016-10-27 11:38:35	-99.170637	19.283637	-99.178194	19.280982	
12692	12693	México DF Taxi de Sitio	2016-10-28 06:49:41	2016-10-28 06:51:25	-99.194384	19.396768	-99.194622	19.396717	
12693	12694	México DF Radio Taxi	2016-10-27 10:26:38	2016-10-28 07:10:21	-99.180135	19.369919	-99.180551	19.372276	

In [6]:
```
equ_df.tail(3)
```

Out[6]:

	id	vendor_id	pickup_datetime	dropoff_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_
32363	32364	Quito	2016-10-27 12:10:18	2016-10-27 06:33:53	-78.477247	-0.107514	-78.490093	-0.100859	
32364	32365	Quito	2016-10-25 04:58:55	2016-10-25 05:00:25	-78.550264	-0.256730	-78.550306	-0.256756	
32365	32366	Quito	2016-10-28 06:47:59	2016-10-28 06:58:31	-78.431986	-0.341538	-78.446296	-0.327428	

In [7]:
```
print(bog_df.shape)
print(mex_df.shape)
print(equ_df.shape)
```

```
(3063, 12)
(12694, 12)
(32366, 12)
```

In [8]:
```
# check missing data each files
bog_df.isna().sum(axis=0)
```

Out[8]:
```
id                   0
vendor_id            0
pickup_datetime      0
dropoff_datetime     0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
store_and_fwd_flag   0
trip_duration        0
dist_meters          0
wait_sec             0
dtype: int64
```

In [9]:
```
mex_df.isna().sum(axis=0)
```

Out[9]:
```
id                   0
vendor_id            0
pickup_datetime      0
dropoff_datetime     0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
store_and_fwd_flag   0
trip_duration        0
dist_meters          0
wait_sec             0
dtype: int64
```

In [10]:
```
equ_df.isna().sum(axis=0)
```

Out[10]:
```
id                   0
vendor_id            0
pickup_datetime      0
dropoff_datetime     0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
store_and_fwd_flag   0
trip_duration        0
dist_meters          0
wait_sec             0
dtype: int64
```

no NaN value for each file

In [11]:
```
# add a feature that contain the country name to distinguish 'em later
bog_df["country"] = "colombia"
mex_df["country"] = "mexico"
equ_df["country"] = "equador"
```

check longitude and latitude with marker for each country.

check map_checker.ipynb how to generate the picture yes, i've been thru that hell

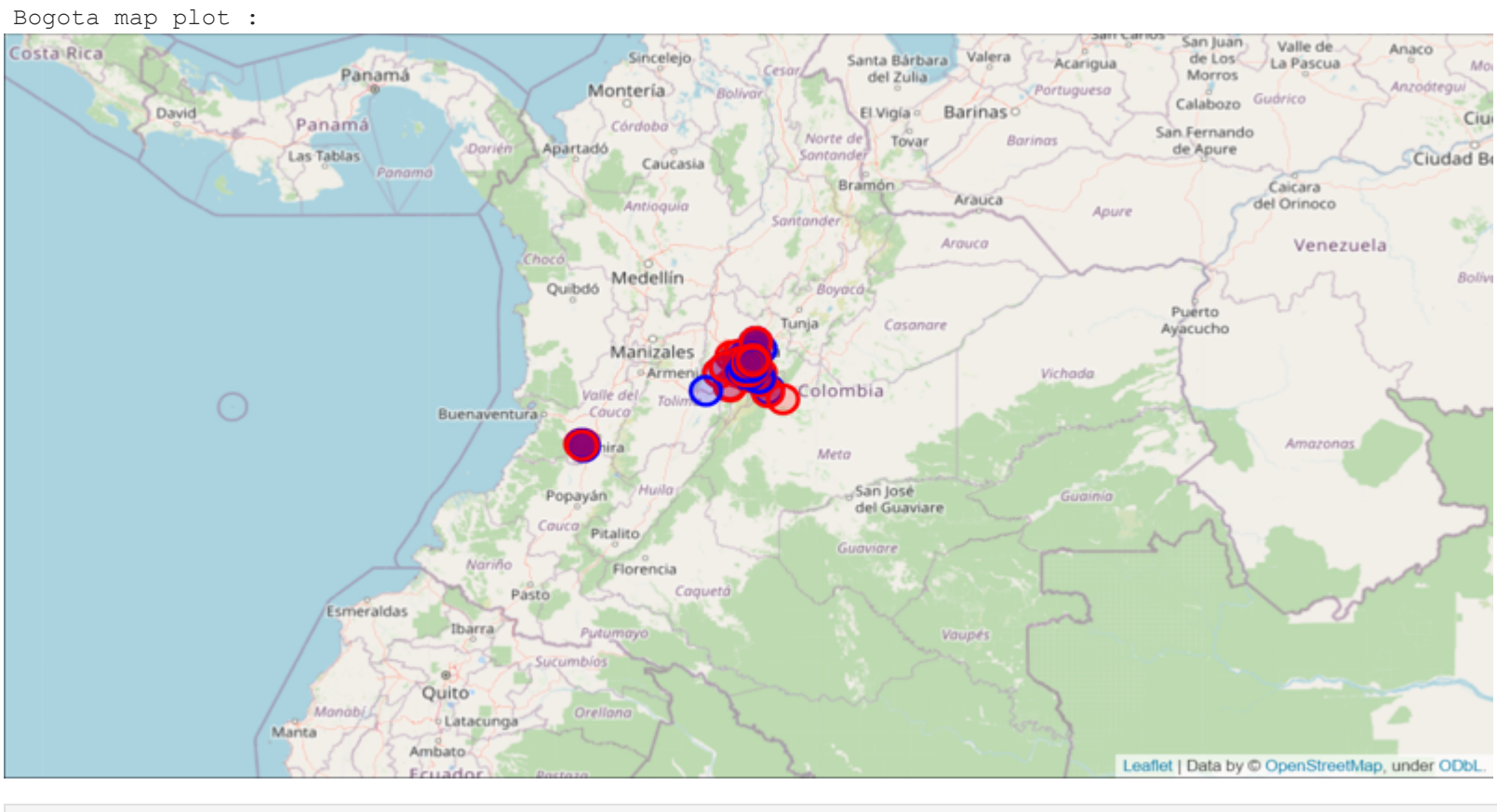
(its faster and lighter to load an image than folium with tons of marker)

In [12]:
```
# plot map with marker each country
IMG_WIDTH = 800
IMG_HEIGHT = 400

img = Image.open(os.path.join("assets", "bog_plot.png"))
img = img.resize((IMG_WIDTH, IMG_HEIGHT))
print("Bogota map plot : ")
img
```

```
Bogota map plot :
```

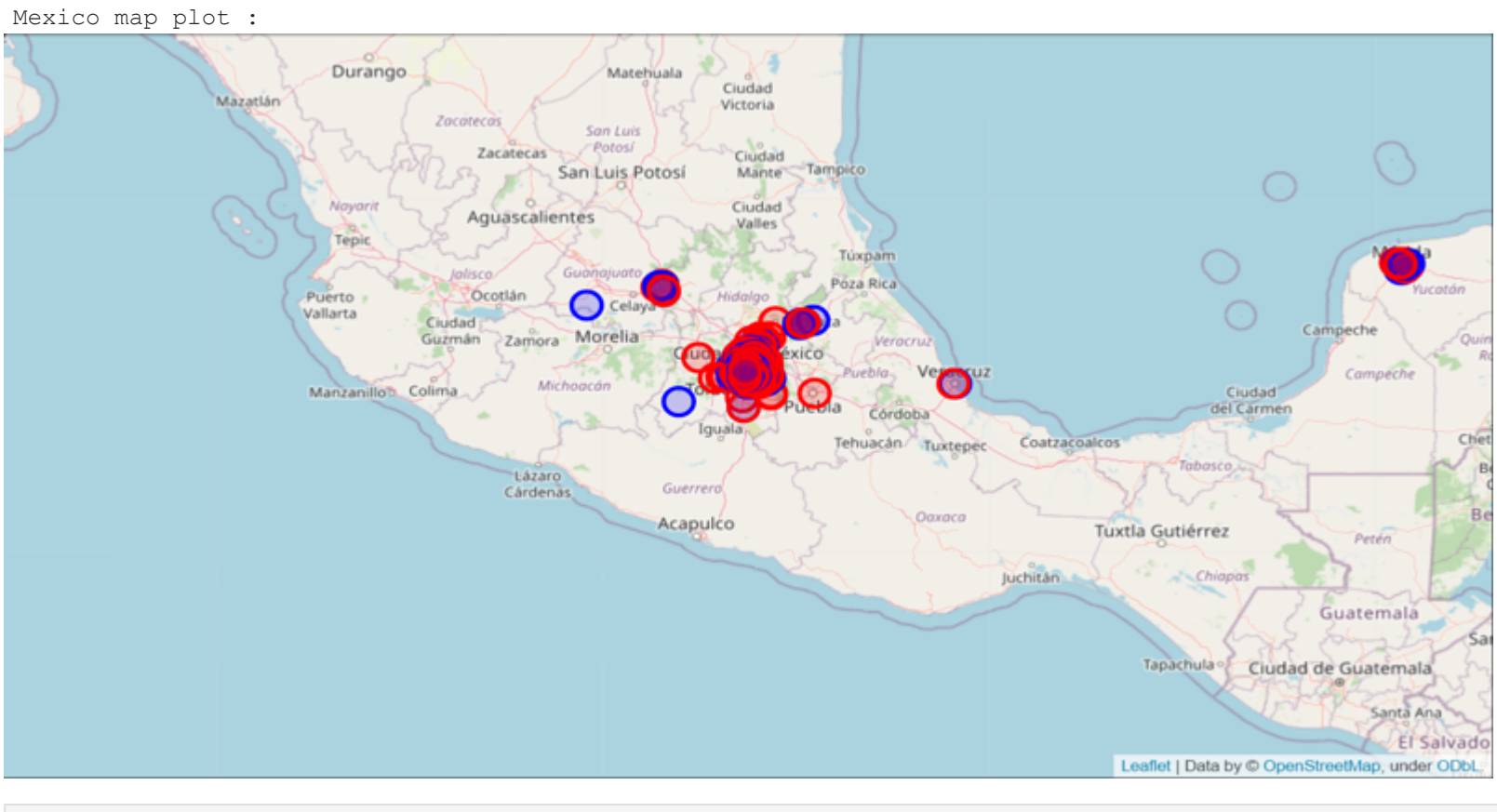
Out[12]:



In [13]:
```
img = Image.open(os.path.join("assets", "mex_plot.png"))
img = img.resize((IMG_WIDTH, IMG_HEIGHT))
print("Mexico map plot : ")
img
```

```
Mexico map plot :
```

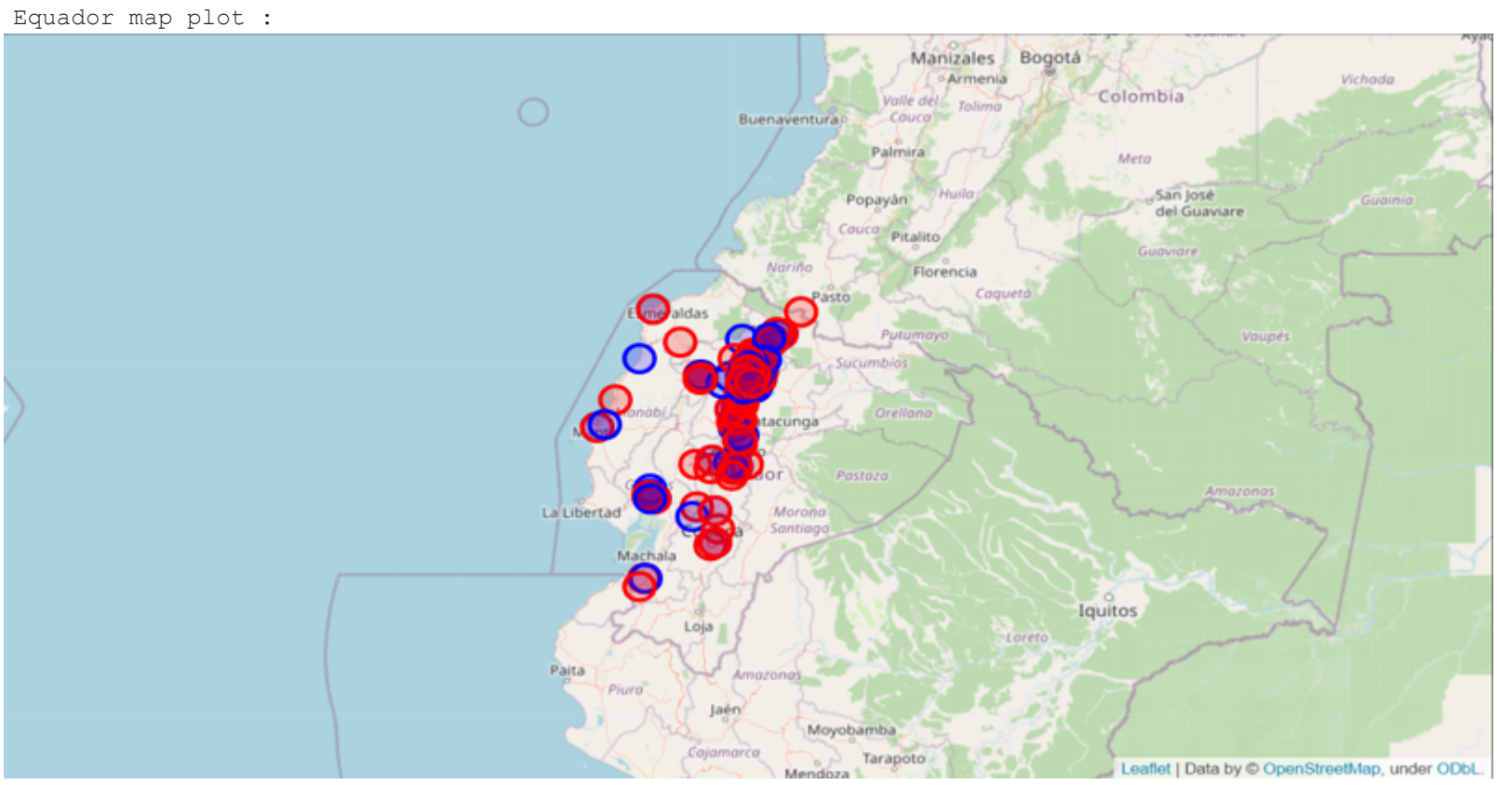
Out[13]:



In [14]:
```
img = Image.open(os.path.join("assets", "equ_plot.png"))
img = img.resize((IMG_WIDTH, IMG_HEIGHT))
print("Equador map plot : ")
img
```

```
Equador map plot :
```

Out[14]:



looks good no missing data, also the lat long is within the country, lets just concat the datasets from bogota, mexico, and equador

In [15]:
```
# concat csv
dataset_df = pd.concat([bog_df, mex_df, equ_df], ignore_index=True)
print(dataset_df.shape)
```

```
(48123, 13)
```

In [16]:
```
# check type data each column
dataset_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48123 entries, 0 to 48122
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  48123 non-null  int64
 1   vendor_id           48123 non-null  object
 2   pickup_datetime     48123 non-null  object
 3   dropoff_datetime    48123 non-null  object
 4   pickup_longitude    48123 non-null  float64
 5   pickup_latitude     48123 non-null  float64
 6   dropoff_longitude   48123 non-null  float64
 7   dropoff_latitude    48123 non-null  float64
 8   store_and_fwd_flag  48123 non-null  object
 9   trip_duration       48123 non-null  int64
 10  dist_meters         48123 non-null  int64
 11  wait_sec            48123 non-null  int64
 12  country             48123 non-null  object
dtypes: float64(4), int64(4), object(5)
memory usage: 4.8+ MB
```

Berdasarkan informasi di atas, dataset kita tidak terdapat nilai Null. Selain itu, semua tipe data fitur sesuai dengan peruntukkannya. Maka, tidak perlu dilakukan pengisian nilai Null dan konversi tipe data.

In [17]:
```
# check categorical features data
for i in dataset_df[["vendor_id", "store_and_fwd_flag", "country"]] :
    print(f"{i} : {dataset_df[i].unique()} \n")
```

```
vendor_id : ['Bogotá' 'Bogotá UberX' 'Bogota UberBlack' 'Bogotá UberVan'
 'Bogotá UberAngel' 'México DF Taxi de Sitio' 'México DF Taxi Libre'
 'México DF Radio Taxi' 'México DF UberX' 'México DF UberXL'
 'México DF UberSUV' 'México DF UberBlack' 'Quito' 'Quito Cabify Lite'
 'Quito Cabify Executive' 'Quito UberX']

store_and_fwd_flag : ['N']

country : ['colombia' 'mexico' 'equador']
```

so store_and_fwd_flag only have one unique value, just drop it coz not very useful, also drop id

In [18]:
```
# drop store_and_fwd_flag and id
dataset_df = dataset_df.drop(["store_and_fwd_flag", "id"], axis=1)
```

In [19]:
```
# check and drop if there is any duplicated data
dataset_df = dataset_df.drop_duplicates()
print(dataset_df.shape)
print("drop duplicated data : {}".format(48123-47388))
```

```
(47388, 11)
drop duplicated data : 735
```

In [20]:
```
# change date and time format
dataset_df["pickup_datetime"] = pd.to_datetime(dataset_df["pickup_datetime"], format="%Y/%m/%d %H:%M:%S")
dataset_df["dropoff_datetime"] = pd.to_datetime(dataset_df["dropoff_datetime"], format="%Y/%m/%d %H:%M:%S")
```

In [21]:
```
# check max and minimum time
print("pickup & dropoff maximal time is {} {}".format(
    dataset_df["pickup_datetime"].dt.time.max(),
    dataset_df["dropoff_datetime"].dt.time.max())
)

print("pickup & dropoff minimum time is {} {}".format(
    dataset_df["pickup_datetime"].dt.time.min(),
    dataset_df["dropoff_datetime"].dt.time.min())
)
```

```
pickup & dropoff maximal time is 12:59:59 12:59:59
pickup & dropoff minimum time is 01:00:01 01:00:01
```

there is no AM PM notation but the time is maxed at 12:59:59 and there is no 00:00:00 time range.

strip the time, because not useful and might lead to mis-calculation

In [22]:
```
# strip time from datetime
dataset_df["pickup_datetime"] = pd.to_datetime(dataset_df["pickup_datetime"]).dt.date
dataset_df["dropoff_datetime"] = pd.to_datetime(dataset_df["dropoff_datetime"]).dt.date
```

In [23]:
```
# sort by pickup date time
dataset_df = dataset_df.sort_values(by=["pickup_datetime"])
dataset_df = dataset_df.reset_index(drop=True)
```

In [24]:
```
dataset_df.describe()
```

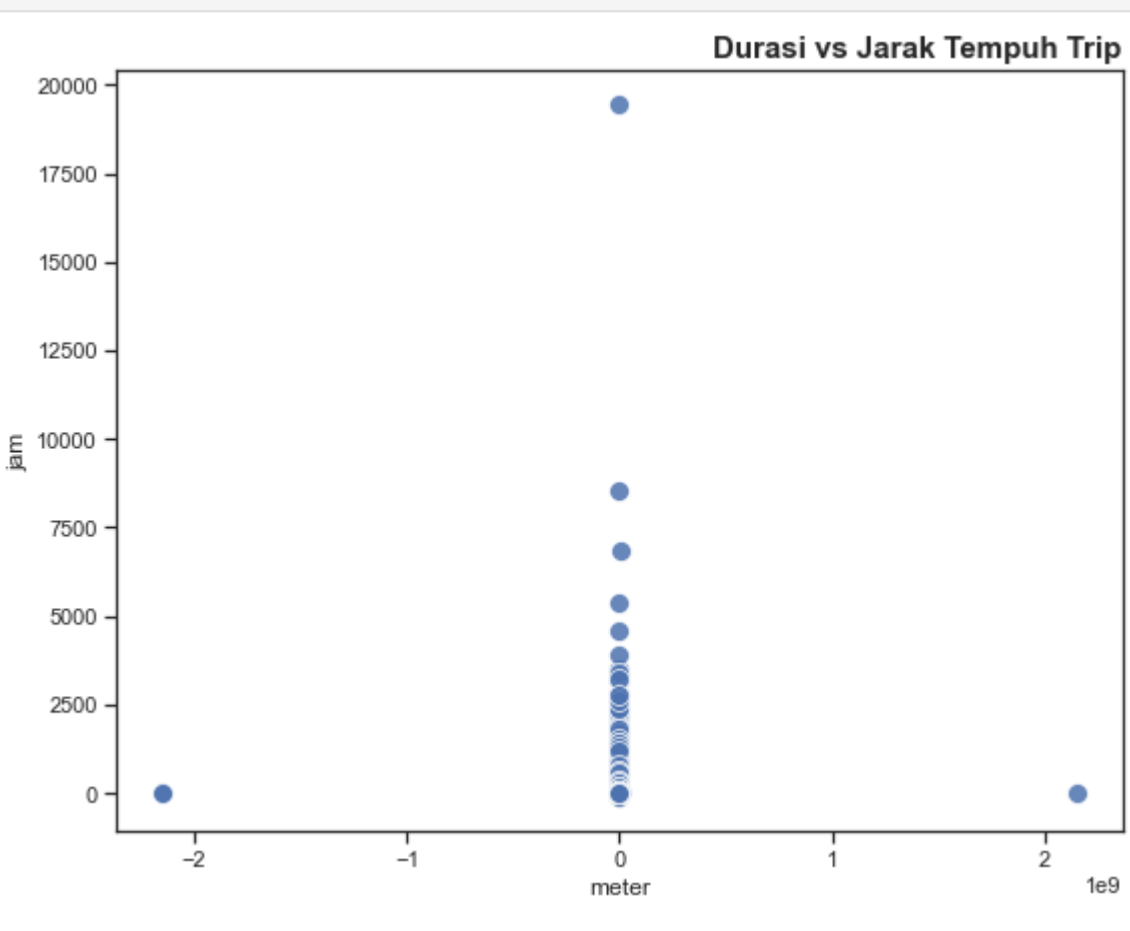
Out[24]:

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration	dist_meters	wait_sec
count	47388.000000	47388.000000	47388.000000	47388.000000	4.738800e+04	4.738800e+04	4.738800e+04
mean	-83.701066	5.321787	-83.700891	5.320583	3.854506e+04	-3.860076e+04	1.541136e+07
std	9.350035	8.542780	9.348693	8.544595	5.106782e+05	1.708677e+07	6.051557e+08
min	-108.985069	-3.454279	-108.987429	-3.574739	-3.887110e+05	-2.147484e+09	0.000000e+00
25%	-99.059911	-0.198614	-99.043489	-0.202007	4.960000e+02	2.021000e+03	1.010000e+02
50%	-78.499611	-0.149571	-78.498360	-0.150728	9.435000e+02	4.120500e+03	2.430000e+02
75%	-78.477705	19.283737	-78.479058	19.283099	1.951250e+03	7.855250e+03	5.340000e+02
max	-73.829723	25.752964	-73.615700	25.781058	7.002605e+07	2.147484e+09	9.516306e+10

Sebagai awalan, bahkan dari summary statistiknya saja dapat kita lihat terdapat kesalahan dalam recording data oleh apps. Sebagai contoh, fitur durasi trip (trip_duration) dan jarak tempuh trip (dist_meters) mempunyai nilai maksimum dan minimun yang terlampau jauh, bahkan bisa dikatakan sebagai data outlier/pencilan.

Bukti masalah 1: Relasi abnormal antara durasi trip dan jarak tempuh trip.

In [25]:
```
# plot dist_meter and trip_duratuion in hour
plt.figure(figsize=(9, 7))
with sns.axes_style(style="ticks"):
    sns.scatterplot(
        data=dataset_df,
        x="dist_meters",
        y=dataset_df.trip_duration/3600,
        s=100,
        alpha=0.85
    )
plt.title("Durasi vs Jarak Tempuh Trip",
    loc="right",
    fontweight="bold",
    size=15
)
plt.xlabel("meter")
plt.ylabel("jam");
```



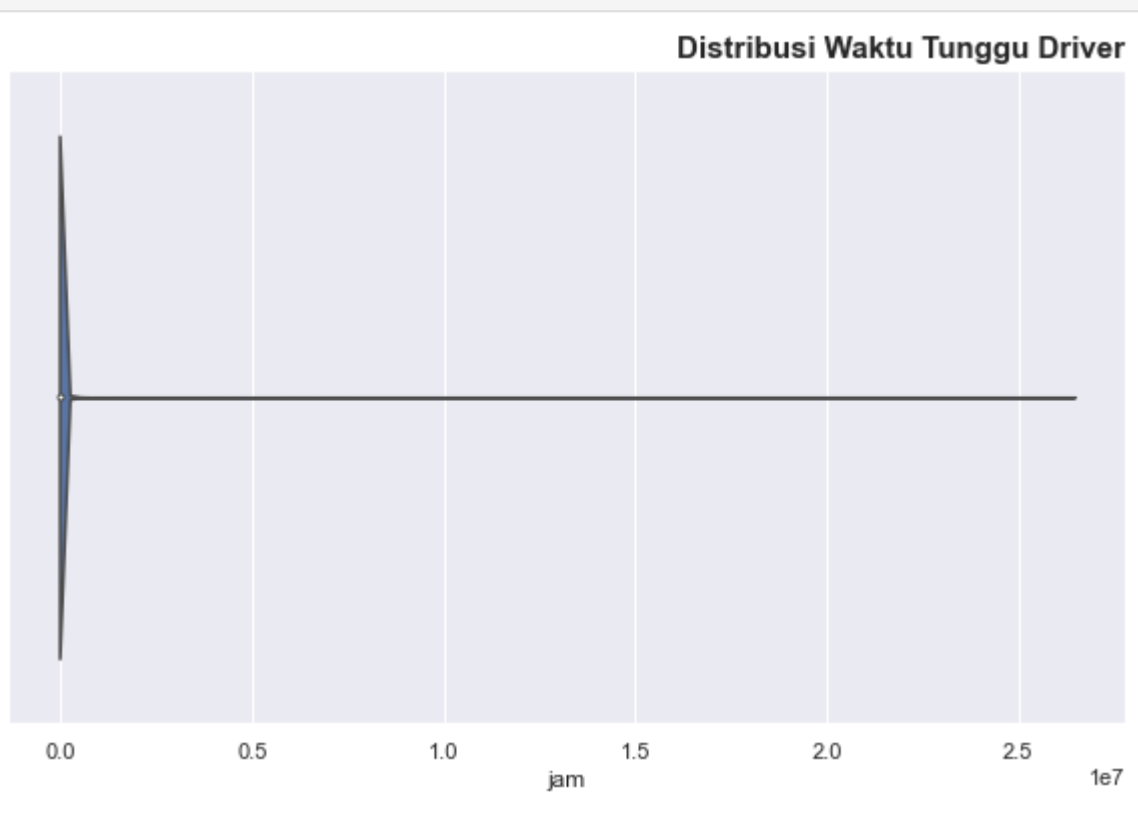
Grafik "Durasi vs Jarak Tempuh Trip" mendukung pernyataan di bagian summary statistik. Dapat dilihat bahwa terdapat beberapa data yang terletak sangat jauh dari sebaran data mayoritas/yang benar (tengah). Data yang sangat jauh tersebut merupakan data yang salah di-record oleh apps, sehingga merupakan trip yang False.

Bukti masalah 2: Waktu tunggu driver yang terlampau lama.

In [26]:
```
# plot wait_sec distribution
plt.figure(figsize=(10, 6))

with sns.axes_style(style="darkgrid"):
    sns.violinplot(x=dataset_df["wait_sec"]/3600)

plt.title(
    "Distribusi Waktu Tunggu Driver",
    loc="right",
    fontweight="bold",
    size=15
)
plt.xlabel("jam");
```



Berdasarkan grafik "Distribusi Waktu Tunggu Driver" dapat dilihat terdapat beberapa data yang bernilai sangat besar (jutaan jam). Hal ini sangat tidak masuk akal dan tentunya dapat dipastikan bahwa nilai yang sangat besar tersebut merupakan data yang salah di-record oleh apps, sehingga merupakan trip yang False.

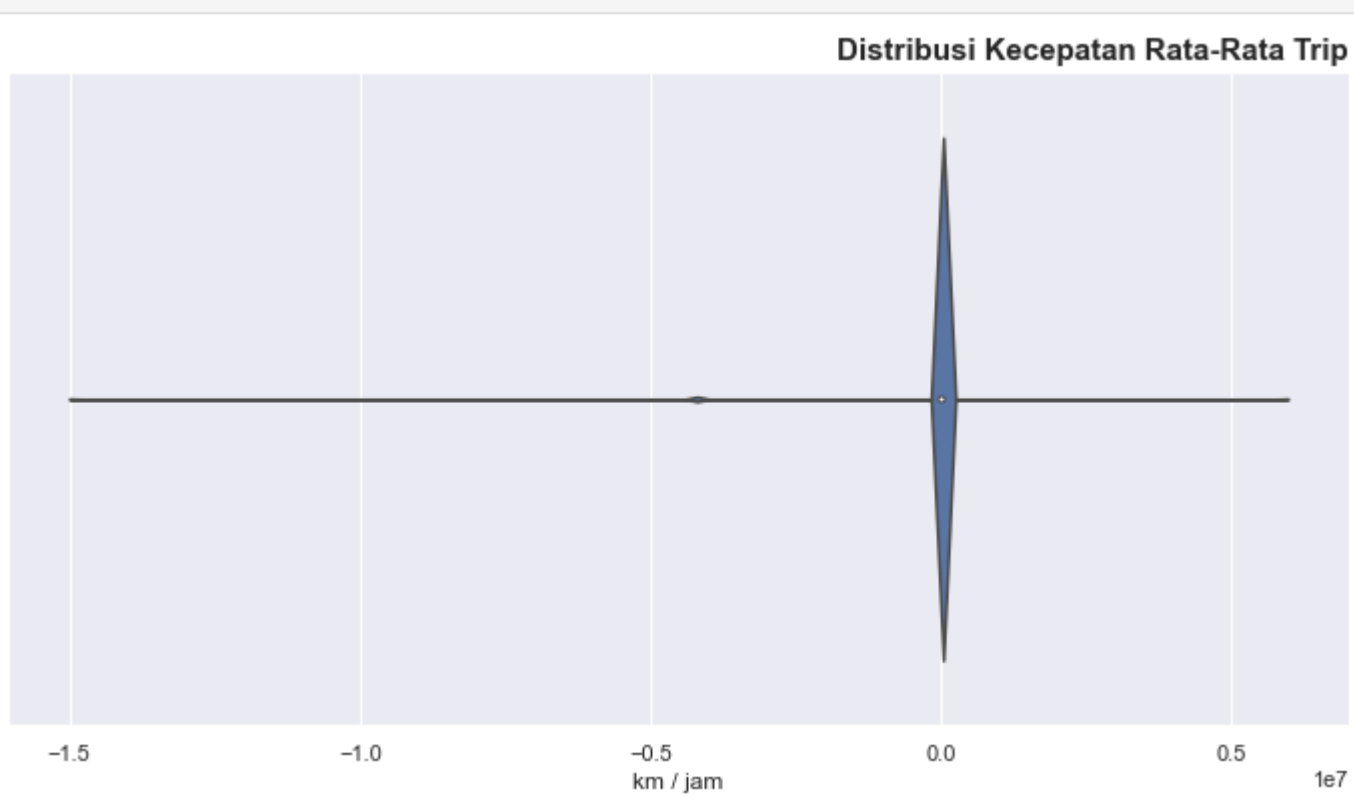
Bukti masalah 3: Kecepatan rata-rata trip yang terlampau cepat ataupun lambat.

In [27]:
```
# calculate speed in km/h
speed_kmph = (dataset_df["dist_meters"] / 1000) / ((dataset_df["trip_duration"] / 3600))
speed_kmph.describe()
```

Out[27]:
```
count    4.738800e+04
mean    -2.597491e+02
std      7.650799e+04
min     -1.498244e+07
25%      9.356612e+00
50%      1.638941e+01
75%      2.269694e+01
max      5.942307e+06
dtype: float64
```

In [28]:
```
# plot kmph distribution
plt.figure(figsize=(12, 6))
with sns.axes_style(style="darkgrid"):
    sns.violinplot(x=speed_kmph)

plt.title(
    "Distribusi Kecepatan Rata-Rata Trip",
    loc="right",
    fontweight="bold",
    size=15
)
plt.xlabel("km / jam");
```



Berdasarkan summary statistik fitur **speed_kmph** dan grafik "Distribusi Kecepatan Rata-Rata Trip" dapat dilihat bahwa terdapat beberapa data dengan nilai kecepatan rata-rata trip yang sangat besar (5.9 juta km/jam) dan bahkan negatif (-14 juta km/jam). Hal ini lagi-lagi sangat tidak masuk akal dan tentunya dapat dipastikan bahwa nilai yang sangat besar dan negatif tersebut merupakan data yang salah di-record oleh apps, sehingga merupakan trip yang False.

TODO :

1. make Label for T / F classification

- if the trip more than 1 day, false
- compare with est_dist
- compare with est_time

1. determine good feature for predicting the price

- extract the uber variant
  - use regex
  - make new column for extracted variant or rename the variant_id
- make uber uber prices for each country
- convert to dollar to make prices each country standarized

In [29]:
```
delta = dataset_df.dropoff_datetime[0] - dataset_df.pickup_datetime[0]
delta.days
```

Out[29]:
```
811
```

In [ ]: