In [1]:

```
from sklearn.cluster import KMeans
from tqdm import tqdm

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

train = pd.read_csv('flight_train.csv')
test = pd.read_csv('flight_test.csv')
df = train.merge(test, how='left')

# Shuffling dataset
p = np.random.permutation(len(df))
df = df.iloc[p]

pd.set_option('display.max_rows', 100)

# Converting dataframe to lower
for col in df.select_dtypes(exclude='number').columns:
    df[col] = df[col].apply(lambda x: str(x).lower())

# Removing non-ascii characters
for col in df.select_dtypes(exclude='number').columns:
    df[col] = df[col].str.encode('ascii', 'ignore').str.decode('ascii')


df = df.replace('nan',np.nan)
df.sample(20).T
```

Out[1]:

	3032	16797	24906	3976	17583	43455	12294	8511	31576	30432	14356
MEMBER_NO	5956	2270	13339	9584	11390	43411	21615	57840	49668	16195	59119
FFP_DATE	10/24/2011	8/1/2005	12/12/2009	3/21/2006	8/9/2007	1/17/2013	10/31/2008	4/18/2008	5/6/2010	11/12/2012	3/21/2011
FIRST_FLIGHT_DATE	10/24/2011	6/7/2010	11/27/2012	2/20/2007	8/9/2007	1/17/2013	11/5/2008	4/18/2008	5/6/2010	11/12/2012	3/21/2011
GENDER	male	male	male	male	male	male	male	male	female	male	male
FFP_TIER	5	4	4	5	4	4	4	4	4	4	4
WORK_CITY	guangzhou	beijing	pishan	beijing	beijing	beijing	shanghai	chengdu	taibei	beijing	shenzhen
WORK_PROVINCE	guangdong	beijing	xinjiang	beijing	beijing	.	shanghai	sichuan	taiwan	beijing	guangdong
WORK_COUNTRY	cn	cn	cn	cn	cn	cn	cn	cn	cn	cn	cn
AGE	43.0	52.0	48.0	60.0	53.0	30.0	48.0	35.0	67.0	39.0	41.0
LOAD_TIME	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014	3/31/2014
FLIGHT_COUNT	43	6	13	19	11	5	11	28	8	6	14
BP_SUM	37827	14770	6015	36621	12599	3644	17827	22447	5956	5299	14422
SUM_YR_1	25213.0	776.0	1680.0	6255.0	8231.0	710.0	8175.0	15146.0	6652.0	603.0	5124.0
SUM_YR_2	16886.0	17772.0	1394.0	28227.0	3953.0	3368.0	10418.0	7014.0	383.0	6300.0	6341.0
SEG_KM_SUM	53121	18190	21551	27333	16849	4014	13540	34037	12776	12540	23152
LAST_FLIGHT_DATE	3/7/2014	3/15/2014	12/18/2013	3/11/2014	3/23/2014	9/30/2013	12/16/2013	1/12/2014	9/16/2013	12/20/2013	3/27/2014
LAST_TO_END	25	17	105	21	9	184	107	80	198	103	5
AVG_INTERVAL	16.190476	99.6	32.166667	33.888889	70.3	64.0	42.9	23.444444	70.285714	80.6	54.923077
MAX_INTERVAL	51	239	194	191	232	120	134	103	176	357	177
EXCHANGE_COUNT	0	0	0	0	0	0	0	1	0	0	1
avg_discount	0.821321	0.794778	0.444227	1.409157	0.823517	0.974131	1.384561	0.721401	0.54438	0.584404	0.713941
Points_Sum	38963	14770	6015	38305	13384	3644	24418	22447	5956	5299	14422
Point_NotFlight	0	0	0	0	2	0	11	1	0	0	1

In [2]:

```
df.drop(['MEMBER_NO','LOAD_TIME','EXCHANGE_COUNT','LAST_TO_END','SEG_KM_SUM','BP_SUM','Points_Sum','SUM_YR_1','SU
df.head().reset_index(drop=True)
```

Out[2]:

	GENDER	WORK_CITY	WORK_PROVINCE	WORK_COUNTRY	AGE	FLIGHT_COUNT	AVG_INTERVAL	MAX_INTERVAL
0	female	changchun	jilin	cn	33.0	6	7.400000	18
1	male	xian	shanxi	cn	31.0	3	126.500000	215
2	male	guangzhoushi	guangdong	cn	43.0	13	24.833333	54
3	male	ningbo	zhejiang	cn	46.0	4	212.666667	342
4	male	beijing	beijing	cn	36.0	2	160.000000	160

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55000 entries, 46524 to 43377
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   GENDER          54997 non-null  object 
 1   WORK_CITY       53034 non-null  object 
 2   WORK_PROVINCE   52216 non-null  object 
 3   WORK_COUNTRY    54980 non-null  object 
 4   AGE             54658 non-null  float64
 5   FLIGHT_COUNT    55000 non-null  int64  
 6   AVG_INTERVAL    55000 non-null  float64
 7   MAX_INTERVAL    55000 non-null  int64  
dtypes: float64(2), int64(2), object(4)
memory usage: 3.8+ MB
```

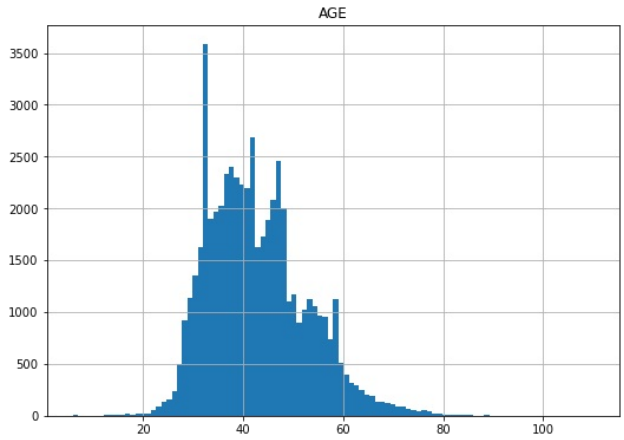
In [4]:
```
df.describe().T
```

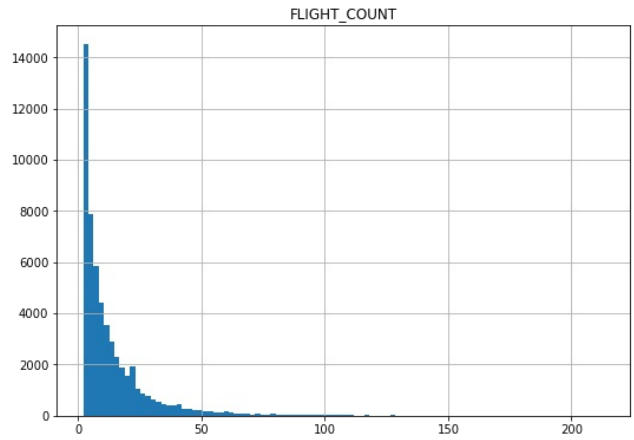
Out[4]:

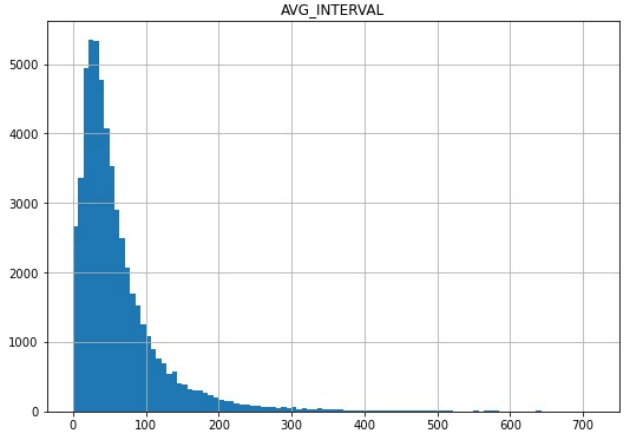
	count	mean	std	min	25%	50%	75%	max
AGE	54658.0	42.705789	9.803796	6.0	35.000000	42.0	48.0	110.0
FLIGHT_COUNT	55000.0	13.213527	14.528764	2.0	4.000000	8.0	16.0	213.0
AVG_INTERVAL	55000.0	62.979631	64.506386	0.0	25.166667	44.6	77.5	714.0
MAX_INTERVAL	55000.0	172.201745	117.382011	0.0	90.000000	149.0	230.0	719.0

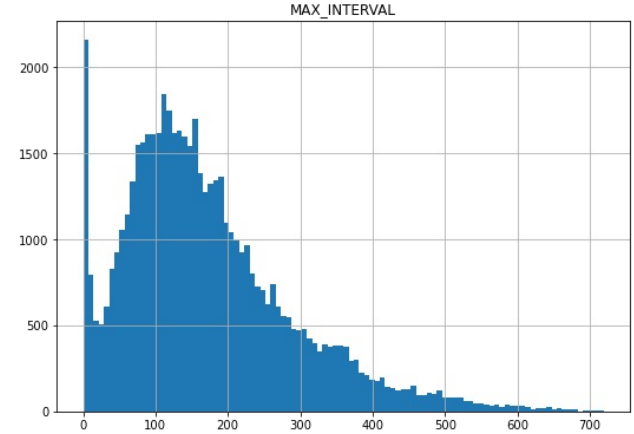
In [5]:
```
# Fillna
for col in df.select_dtypes('number').columns:
    df[col].fillna(np.median(df[col].dropna()),inplace=True)
```

In [6]:
```
df.hist(bins=100,figsize=(20,14));
```







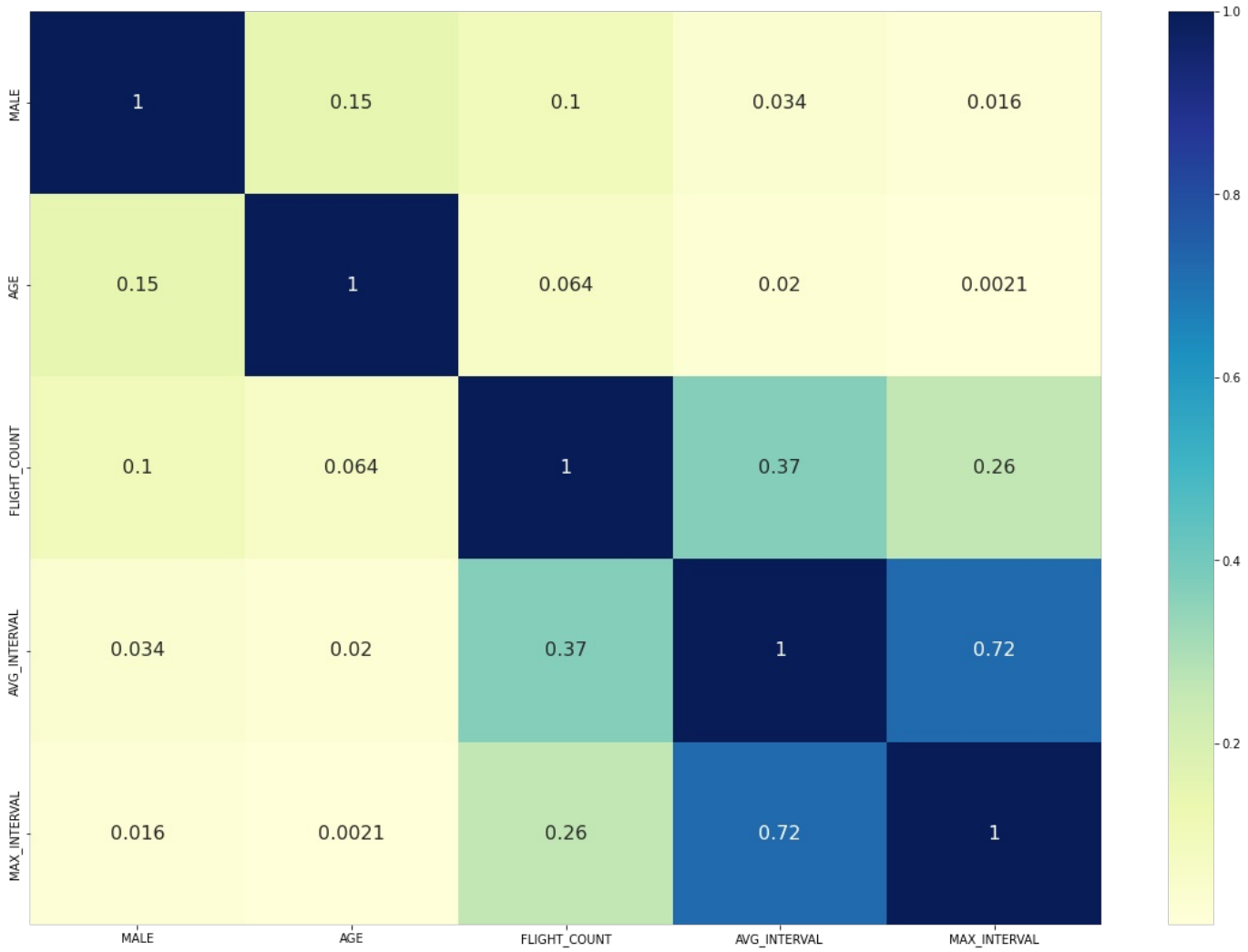


In [7]:
```
df['GENDER'] = df['GENDER'].replace({'male':1,'female':0})
df.rename({'GENDER':'MALE'},axis=1,inplace=True)
df['MALE'].fillna(df['MALE'].mode()[0],inplace=True)
```

In [8]:
```
df['MALE'].value_counts(normalize=True)
```

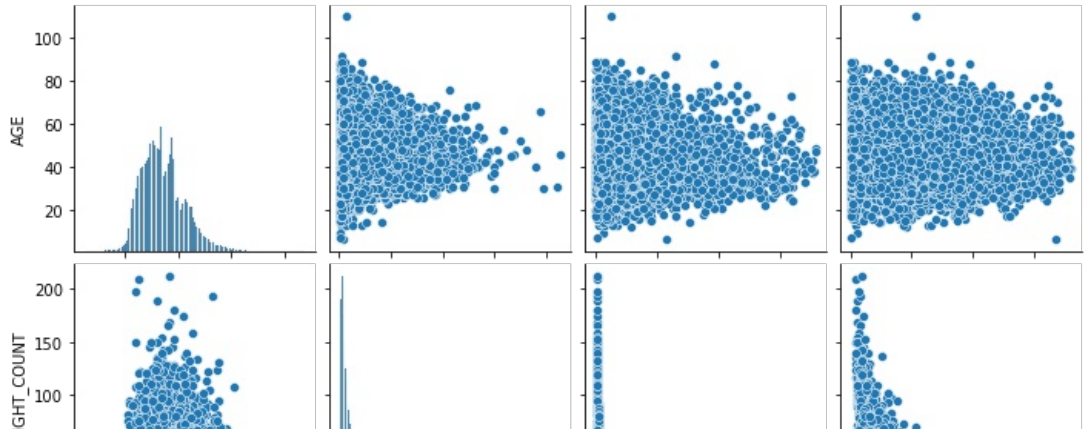
Out[8]:
```
1.0    0.772855
0.0    0.227145
Name: MALE, dtype: float64
```

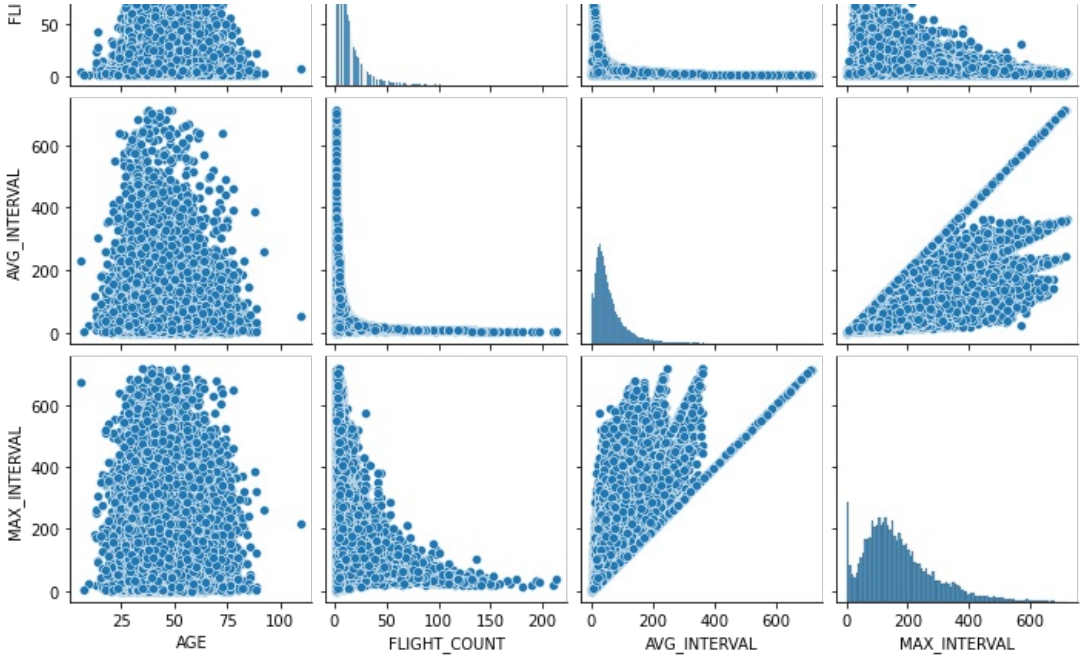
In [9]:
```
plt.figure(figsize=(20,14))
sns.heatmap(df.corr().abs(),annot=True,annot_kws={'size': 16},cmap="YlGnBu");
```



In [10]:
```
# Too weak to include
df.drop(['MALE'],axis=1,inplace=True)
```

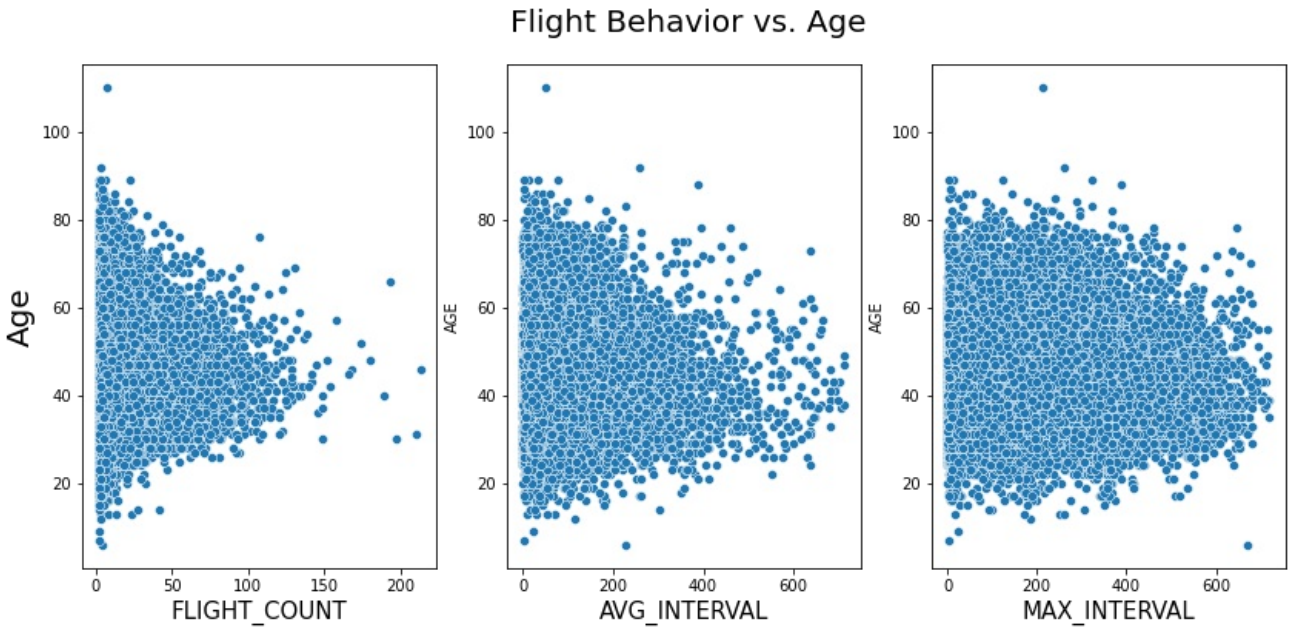
In [11]:
```
# It looks like Age has a non-linear relationship with the other features. We will explore that next.
sns.pairplot(df);
```





In [12]:

```
fig, (ax1,ax2,ax3) = plt.subplots(nrows=1,ncols=3,figsize=(14,6))
sns.scatterplot(data=df,y='AGE',x='FLIGHT_COUNT',ax=ax1)
sns.scatterplot(data=df,y='AGE',x='AVG_INTERVAL',ax=ax2)
sns.scatterplot(data=df,y='AGE',x='MAX_INTERVAL',ax=ax3);
ax1.set_ylabel(None)
ax1.set_ylabel('Age',fontsize=20)


ax1.xaxis.get_label().set_fontsize(15)
ax2.xaxis.get_label().set_fontsize(15)
ax3.xaxis.get_label().set_fontsize(15)

fig.text(0.40,0.93,'Flight Behavior vs. Age',fontsize=20);
```



In [13]:

```
# Cleaning with .replace()
df['WORK_CITY'] = df['WORK_CITY'].replace('.',np.nan).replace('-',np.nan)
df['WORK_PROVINCE'] = df['WORK_PROVINCE'].replace('.',np.nan).replace('-',np.nan).replace('`',np.nan).replace('0'
                                                 .replace('nan',np.nan).replace('hk','guangdong')\
                                                 .replace('hong kong','guangdong')
```

In [14]:

```
# Cleaning using vectorized splitting and indexing
df['WORK_PROVINCE'] = df['WORK_PROVINCE'].str.split('sheng').str[0].str.split('shi').str[0]
```

In [15]:

```
# Mapping word variations
missing_mask = df['WORK_PROVINCE'].isna()
mapping_dict = {'beijing':'beijing', 'shanghai':'shanghai','guangzhoushi':'guangdong','beijingshi':'beijing',
```

```
                        'shanghaishi':'shanghai','guangzhou':'guangdong','hong kong': 'guangdong','hongkong':'gua
                       'shenyangshi':'liaoning', 'taipei':'taiwan'}
df.loc[missing_mask, 'WORK_PROVINCE'] = df.loc[missing_mask, 'WORK_CITY'].map(mapping_dict)
```

In [16]:
```
# Keeping the top 95% most frequently occurring provinces
mask_idx = df['WORK_PROVINCE'].value_counts(normalize=True).index[df['WORK_PROVINCE'].value_counts(normalize=True
df['WORK_PROVINCE'].mask(df['WORK_PROVINCE'].isin(mask_idx), 'other', inplace=True)
```

In [17]:
```
df['WORK_PROVINCE'].fillna('other',inplace=True)
df['WORK_CITY'].fillna('other',inplace=True)
df.drop(['WORK_CITY', 'WORK_COUNTRY'],axis=1,inplace=True)
```

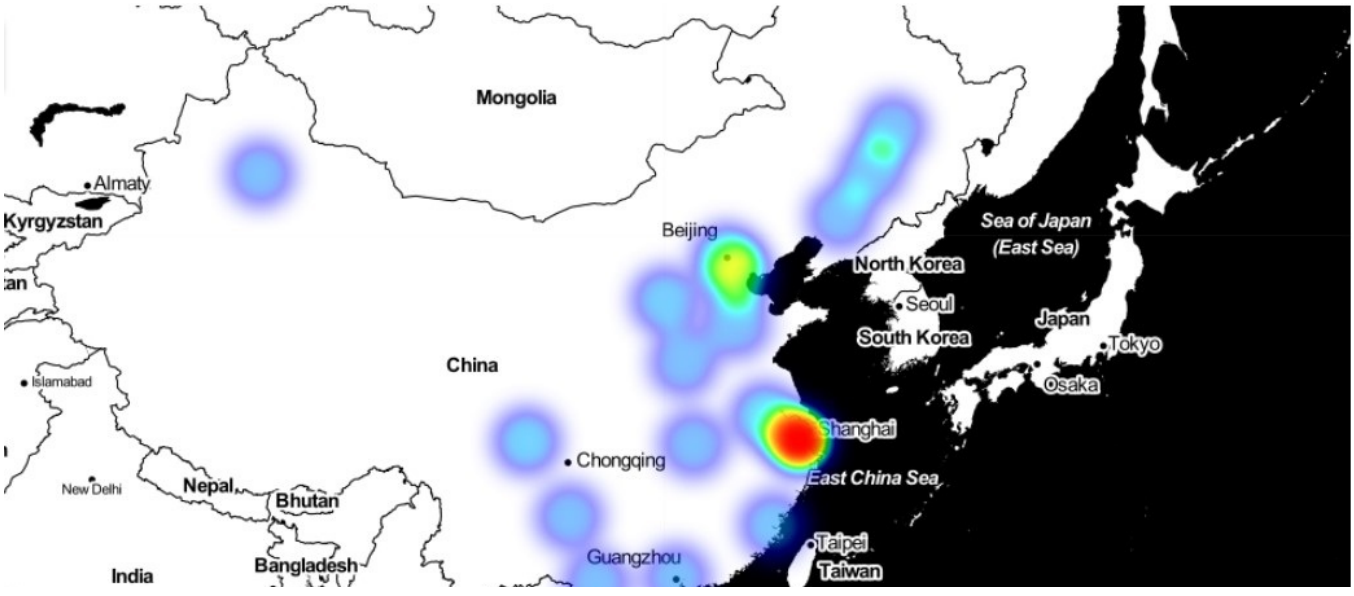
In [18]:
```
# Using requests and Google Geocode API to make a pretty graph
# config is a .py file that stores my Google API key as a string object.

import requests
import config

api_key = config.api_key
base_url = "https://maps.googleapis.com/maps/api/geocode/json"

lats = []
lngs = []

for address in df['WORK_PROVINCE'].unique():
    endpoint = f"{base_url}?address={address}&key={api_key}"
    r = requests.get(endpoint)
    if r.status_code not in range(200, 299):
        print("Error occured!")
    try:
        results = r.json()['results'][0]
        lats.append(results['geometry']['location']['lat'])
        lngs.append(results['geometry']['location']['lng'])
    except:
        pass
```

In [19]:
```
lat_map = dict(zip(df['WORK_PROVINCE'].unique(),lats))
lng_map = dict(zip(df['WORK_PROVINCE'].unique(),lngs))

df['lat'] = df['WORK_PROVINCE'].map(lat_map)
df['lng'] = df['WORK_PROVINCE'].map(lng_map)
```

In [20]:
```
heat_data = df.groupby(["lat","lng"])['AVG_INTERVAL'].mean()
```

In [21]:
```
import folium
from folium.plugins import HeatMap

lat = df.iloc[0]['lat']
lng = df.iloc[0]['lng']
map = folium.Map(location=[lat, lng], zoom_start=4,tiles='Stamen Toner')
heat_data = df.groupby(["lat","lng"])['AVG_INTERVAL'].mean().reset_index().values
HeatMap(heat_data).add_to(map)
```

Out[21]:
```
<folium.plugins.heat_map.HeatMap at 0x29b1e0b49d0>
```



In [22]:
```
df = df.drop(['WORK_PROVINCE','lat','lng'],axis=1)

x = (df - df.mean()) / df.std()
```

In [23]:
```
x.shape
```

Out[23]:
```
(55000, 4)
```

In [24]:
```
x.isnull().sum()
```

Out[24]:
```
AGE             0
FLIGHT_COUNT    0
AVG_INTERVAL    0
MAX_INTERVAL    0
dtype: int64
```

In [25]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55000 entries, 46524 to 43377
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   AGE            55000 non-null  float64
 1   FLIGHT_COUNT   55000 non-null  float64
 2   AVG_INTERVAL   55000 non-null  float64
 3   MAX_INTERVAL   55000 non-null  float64
dtypes: float64(4)
memory usage: 4.1 MB
```

In [26]:
```
x.describe().T
```

Out[26]:

	count	mean	std	min	25%	50%	75%	max
AGE	55000.0	7.106790e-16	1.0	-3.755224	-0.787994	-0.071766	0.542144	6.885877
FLIGHT_COUNT	55000.0	-3.642500e-16	1.0	-0.771816	-0.634158	-0.358842	0.191790	13.751099
AVG_INTERVAL	55000.0	-4.609297e-15	1.0	-0.976332	-0.586189	-0.284927	0.225100	10.092340
MAX_INTERVAL	55000.0	3.114680e-17	1.0	-1.467020	-0.700293	-0.197660	0.492394	4.658280

In [27]:
```
x.sample(20).T
```

Out[27]:

	32448	18533	26446	45784	13886	28151	5360	7518	91	16951	11923	9910
AGE	-1.913495	-0.378721	0.337507	0.542144	-1.094949	-0.378721	-1.094949	-0.174084	1.156053	0.030552	0.439825	0.951417
FLIGHT_COUNT	-0.771816	0.742422	-0.427671	-0.565329	0.122961	0.191790	0.811251	0.467106	1.981344	-0.496500	0.467106	0.948909
AVG_INTERVAL	-0.852313	-0.566531	-0.175377	0.833567	-0.349594	-0.541233	-0.526764	-0.424775	-0.733588	0.601807	-0.512077	-0.775994
MAX_INTERVAL	-1.398866	-0.393602	-0.206179	0.977988	-0.223218	-0.572505	-0.487313	0.134588	-0.308410	1.378391	-0.010238	-0.887715

Apply KMeans

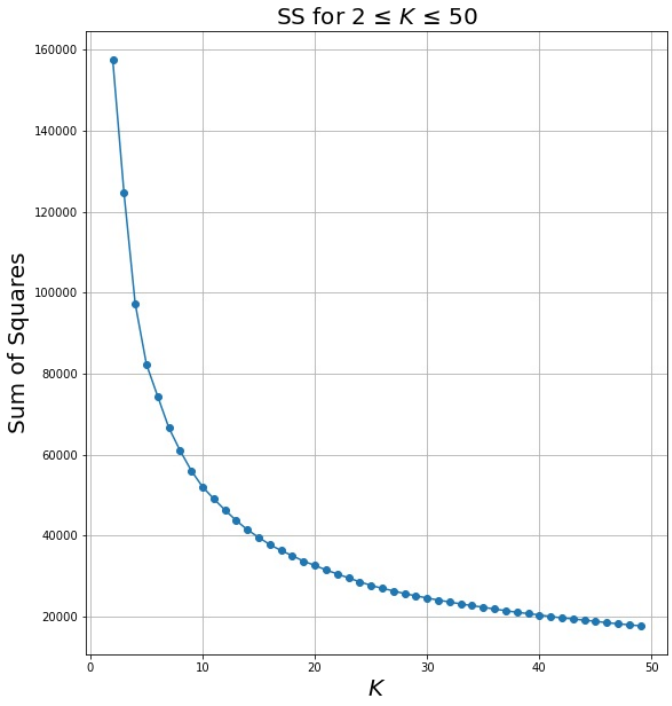
In [28]:
```
%%time
```

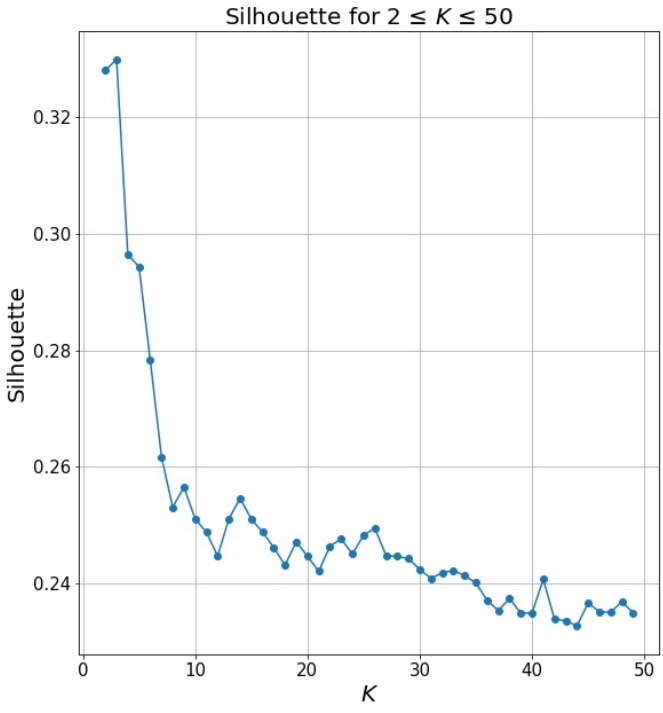
```
from sklearn.metrics import silhouette_score

x = x.to_numpy()

ss_list = []
sils = []


k_clusters = np.arange(2,50)
for k in tqdm(k_clusters):

    kmeans = KMeans(n_clusters = k, random_state=89)
    assigned_cluster = kmeans.fit_predict(x)
    silhouette_avg = silhouette_score(x, assigned_cluster)
    sils.append(silhouette_avg)
    clusters = kmeans.cluster_centers_
    ss = np.sum((x - clusters[assigned_cluster]) **2)
    ss_list.append(ss)

fig, (ax1, ax2) = plt.subplots(1,2, figsize=(20,10))

ax1 = plt.subplot(1,2,1)
ax1.plot(k_clusters, ss_list, marker='o')
ax1.set_title('SS for 2 ≤ $K$ ≤ 50', fontsize=20)
ax1.set_xlabel('$K$',fontsize=20)
ax1.set_ylabel('Sum of Squares',fontsize=20)
ax1.grid(True)

ax2 = plt.subplot(1,2,2)
ax2.plot(k_clusters, sils, marker='o')
ax2.set_title('Silhouette for 2 ≤ $K$ ≤ 50', fontsize=20)
ax2.set_xlabel('$K$',fontsize=20)
ax2.set_ylabel('Silhouette',fontsize=20);
ax2.grid(True)

plt.xticks(size=15)
plt.yticks(size=15);
```

```
100%|██████████| 48/48 [20:45<00:00, 25.95s/it]
Wall time: 20min 45s
```

Out[28]:

```
(array([0.22, 0.24, 0.26, 0.28, 0.3 , 0.32, 0.34]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```





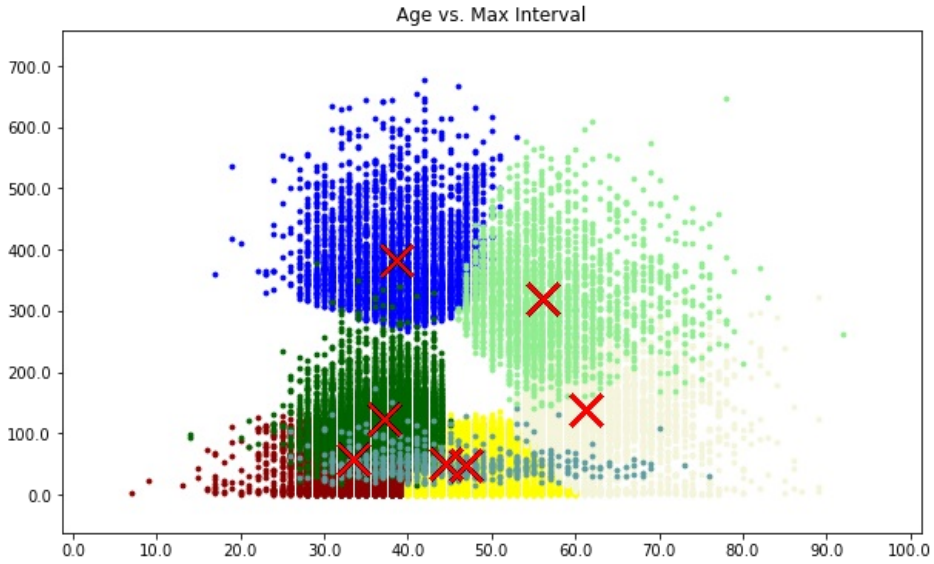
In [29]:

```
# Training our Model
Best_K = 14
```

```
kmeans = KMeans(n_clusters = Best_K, random_state = 89)
df_x = pd.DataFrame(x, columns=df.columns)
df_x['Cluster'] = kmeans.fit_predict(x)
df_x.head()

centers_map = dict(zip(range(Best_K),kmeans.cluster_centers_))

df_x['Cluster Centers'] = df_x['Cluster'].map(centers_map)

df_x['AGE_Centers'] = [x[0] for x in df_x['Cluster Centers']]
df_x['FlightCount_Centers'] = [x[1] for x in df_x['Cluster Centers']]
df_x['AVGInterval_Centers'] = [x[2] for x in df_x['Cluster Centers']]
df_x['MAXInterval_Centers'] = [x[3] for x in df_x['Cluster Centers']]
```

In [30]:

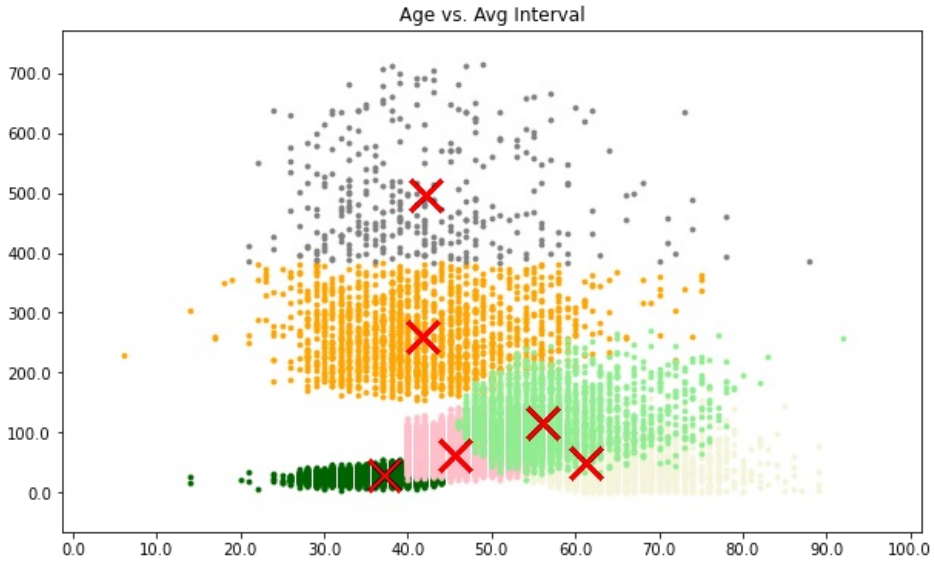
```
colors = ['yellow', 'blue', 'green', 'purple', 'orange', 'darkred', \
     'beige', 'darkblue', 'darkgreen', 'cadetblue', \
     'pink', 'lightblue', 'lightgreen', 'gray', \
     'black', 'lightgray', 'red', 'blue', 'green', 'purple', \
     'orange', 'darkred', 'lightred', 'beige', 'darkblue', \
     'darkgreen', 'cadetblue', 'pink', 'lightblue', \
     'lightgreen', 'gray', 'black', 'lightgray' ]

fig, ax = plt.subplots(figsize=(10,6))


ax.scatter(x = df_x.loc[df_x.Cluster == 0 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 0 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 1 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 1 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 5 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 5 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 6 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 6 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 8 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 8 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 9 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 9 , 'MAX_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 12 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 12 , 'MAX_INTERVAL'].va
ax.scatter(x = df_x.loc[df_x.Cluster.isin([0,1,5,6,8,9,12]), 'AGE_Centers'].values , y = df_x.loc[df_x.Cluster.is

# Transforming axes values to original
min_maxinterval_ylim = (0 - df['MAX_INTERVAL'].mean()) / df['MAX_INTERVAL'].std()
max_maxinterval_ylim = (800 - df['MAX_INTERVAL'].mean()) / df['MAX_INTERVAL'].std()
plt.yticks(np.linspace(min_maxinterval_ylim, max_maxinterval_ylim,9), np.linspace(0, 800,9))
ax.set_ylim([-2,5])

min_age_xlim = (0 - df['AGE'].mean()) / df['AGE'].std()
max_age_xlim = (110 - df['AGE'].mean()) / df['AGE'].std()
plt.xticks(np.linspace(min_age_xlim, max_age_xlim,12), np.linspace(0, 110,12))
ax.set_xlim([-4.5,6])

ax.set_title('Age vs. Max Interval');
```



In [31]:

```
fig, ax = plt.subplots(figsize=(10,6))

ax.scatter(x = df_x.loc[df_x.Cluster == 4 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 4 , 'AVG_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 6 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 6 , 'AVG_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 8 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 8 , 'AVG_INTERVAL'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 10 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 10 , 'AVG_INTERVAL'].va
ax.scatter(x = df_x.loc[df_x.Cluster == 12 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 12 , 'AVG_INTERVAL'].va
ax.scatter(x = df_x.loc[df_x.Cluster == 13 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 13 , 'AVG_INTERVAL'].va
ax.scatter(x = df_x.loc[df_x.Cluster.isin([4,6,8,10,12,13]), 'AGE_Centers'].values , y = df_x.loc[df_x.Cluster.is

min_avginterval_ylim = (0 - df['AVG_INTERVAL'].mean()) / df['AVG_INTERVAL'].std()
max_avginterval_ylim = (800 - df['AVG_INTERVAL'].mean()) / df['AVG_INTERVAL'].std()
```

```
min_avginterval_ylim = (0 - df['AVG_INTERVAL'].mean()) / df['AVG_INTERVAL'].std()
max_avginterval_ylim = (800 - df['AVG_INTERVAL'].mean()) / df['AVG_INTERVAL'].std()

plt.yticks(np.linspace(min_avginterval_ylim, max_avginterval_ylim,9), np.linspace(0, 800,9))
ax.set_ylim([-2,11])

max_age_xlim = (110 - df['AGE'].mean()) / df['AGE'].std()
min_age_xlim = (0 - df['AGE'].mean()) / df['AGE'].std()
plt.xticks(np.linspace(min_age_xlim, max_age_xlim,12), np.linspace(0, 110,12))
ax.set_xlim([-4.5,6])
ax.set_title('Age vs. Avg Interval');
```



In [32]:

```
fig, ax = plt.subplots(figsize=(10,6))

ax.scatter(x = df_x.loc[df_x.Cluster == 2 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 2 , 'FLIGHT_COUNT'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 4 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 4 , 'FLIGHT_COUNT'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 6 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 6 , 'FLIGHT_COUNT'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 7 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 7 , 'FLIGHT_COUNT'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 8 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 8 , 'FLIGHT_COUNT'].valu
ax.scatter(x = df_x.loc[df_x.Cluster == 11 , 'AGE'].values , y = df_x.loc[df_x.Cluster == 11 , 'FLIGHT_COUNT'].va
ax.scatter(x = df_x.loc[df_x.Cluster.isin([2,4,6,7,8,11]), 'AGE_Centers'].values , y = df_x.loc[df_x.Cluster.isin


min_flightcount_ylim = (0 - df['FLIGHT_COUNT'].mean()) / df['FLIGHT_COUNT'].std()
max_flightcount_ylim = (220 - df['FLIGHT_COUNT'].mean()) / df['FLIGHT_COUNT'].std()

min_flightcount_ylim = (0 - df['FLIGHT_COUNT'].mean()) / df['FLIGHT_COUNT'].std()
max_flightcount_ylim = (220 - df['FLIGHT_COUNT'].mean()) / df['FLIGHT_COUNT'].std()

plt.yticks(np.linspace(min_flightcount_ylim, max_flightcount_ylim,11), np.linspace(0, 220,11))
ax.set_ylim([-2,12])

max_age_xlim = (110 - df['AGE'].mean()) / df['AGE'].std()
min_age_xlim = (0 - df['AGE'].mean()) / df['AGE'].std()
plt.xticks(np.linspace(min_age_xlim, max_age_xlim,12), np.linspace(0, 110,12))
ax.set_xlim([-3.5,5.2])
ax.set_title('Age vs. Flight Count');
```