Criteo Dataset – Data Viz - Dauphine M2

Elaboree par CHERIF MONGIA

In [231]:
```python
# we present first of all the librairies that we need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Exploring data : campaigns_delivery

campaigns_delivery.tsv – Delivery metrics for the online advertising campaigns that the e-commerce website is running with Criteo

In [232]:
```python
# Read data
df=pd.read_table(r'C:\Users\monjia\Downloads\campaigns_delivery.tsv',se
p='\t')
```

In [233]:
```python
#Read the 5 first rows
df.head()
```

Out[233]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays	N
0	2019-10-13	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	11487	1

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays
1	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0
2	2019-11-08	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0
3	2019-10-11	web	Mac OS	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	2102
4	2019-10-16	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0

In [234]:

```
#Read the 5 latest rows
df.tail()
```

Out[234]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numbe display
8823	2019-11-19	web	Android	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
8824	2019-11-19	web	Windows	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numbe (display
8825	2019-10-17	other	Other	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	4
8826	2019-11-14	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
8827	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

In [235]:
```
#The shape and size of dataset
print('the shape of df:',df.shape)
print('the size of df:',df.size)
```

```
the shape of df: (8828, 11)
the size of df: 97108
```

In [236]:
```
#The columns
df.columns
```

Out[236]:
```
Index(['Day', 'Environment', 'Os', 'Campaign ID', 'Campaign Optimizatio
n',
       'Campaign Type', 'Context IDs Eligible', 'Number of displays',
       'Number of clicks', 'Criteo Revenue', 'Criteo Cost'],
      dtype='object')
```

In [237]:
```
#print train data columns
print('Columns of the campaigns_delivery dataset is', df.columns)
```

```
Columns of the campaigns_delivery dataset is Index(['Day', 'Environmen
t', 'Os', 'Campaign ID', 'Campaign Optimization',
```

```
       'Campaign Type', 'Context IDs Eligible', 'Number of displays',
       'Number of clicks', 'Criteo Revenue', 'Criteo Cost'],
      dtype='object')
```

In [238]:
```
df.describe()
```

Out[238]:

	Campaign ID	Number of displays	Number of clicks	Criteo Revenue	Criteo Cost
count	8828.000000	8.828000e+03	8828.000000	8828.000000	8828.000000
mean	164273.597191	2.141655e+05	1650.587789	300.494769	169.976270
std	36342.317697	5.201823e+05	4037.909541	979.402831	564.852979
min	113450.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	113890.000000	2.500000e+01	0.000000	0.030000	0.019295
50%	179971.000000	2.806500e+04	143.000000	24.075285	14.009600
75%	196251.000000	1.604248e+05	1070.000000	168.771213	92.444600
max	204638.000000	6.413055e+06	54592.000000	18055.210806	10587.617563

In [239]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8828 entries, 0 to 8827
Data columns (total 11 columns):
Day                         8828 non-null object
Environment                 8828 non-null object
Os                          8828 non-null object
Campaign ID                 8828 non-null int64
Campaign Optimization       8828 non-null object
Campaign Type               8828 non-null object
Context IDs Eligible        8828 non-null object
Number of displays          8828 non-null int64
Number of clicks            8828 non-null int64
Criteo Revenue              8828 non-null float64
```

```
Criteo Cost               8828 non-null float64
dtypes: float64(2), int64(3), object(6)
memory usage: 758.7+ KB
```

In [240]:
```
#count the nan value for each column
df.isnull().sum()
```

Out[240]:
```
Day                        0
Environment                0
Os                         0
Campaign ID                0
Campaign Optimization      0
Campaign Type              0
Context IDs Eligible       0
Number of displays         0
Number of clicks           0
Criteo Revenue             0
Criteo Cost                0
dtype: int64
```

2.1. Compute the average Criteo margin

In [241]:
```
#We add a column for Margin_criteo
df['Margin_criteo'] = df['Criteo Revenue']-df['Criteo Cost']
df.head()
```

Out[241]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays	N
0	2019-10-13	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	11487	1

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays	N
1	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	(
2	2019-11-08	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	(
3	2019-10-11	web	Mac OS	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	2102	2
4	2019-10-16	app	Android	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	(

In [242]:

```
#first method to check the average of the column Margin criteo
# To compute the average Margin_criteo we just write the describe
df.describe()
#so for the average of Margin_criteo we get 130.518499
```

Out[242]:

	Campaign ID	Number of displays	Number of clicks	Criteo Revenue	Criteo Cost	Margin_
count	8828.000000	8.828000e+03	8828.000000	8828.000000	8828.000000	8828.00(
mean	164273.597191	2.141655e+05	1650.587789	300.494769	169.976270	130.518
std	36342.317697	5.201823e+05	4037.909541	979.402831	564.852979	416.195
min	113450.000000	0.000000e+00	0.000000	0.000000	0.000000	-75.3854
25%	113890.000000	2.500000e+01	0.000000	0.030000	0.019295	0.00598

	Campaign ID	Number of displays	Number of clicks	Criteo Revenue	Criteo Cost	Margin_
50%	179971.000000	2.806500e+04	143.000000	24.075285	14.009600	10.0524
75%	196251.000000	1.604248e+05	1070.000000	168.771213	92.444600	72.4247
max	204638.000000	6.413055e+06	54592.000000	18055.210806	10587.617563	7467.59

In [243]:

```
# The second methode to check the average
df['Margin_criteo'].mean()
```

Out[243]: 130.51849934385925

2.2. Plot the margin over time (day and then week). Which graph is more insightful? How would you qualify Criteo margin?

In [244]:

```
#we first order the Day column
df.sort_values(by=['Day']).head()
```

Out[244]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	N di
4047	2019-10-07	web	iOS	113890	Conversion Optimization	LOWER FUNNEL CUSTOM	0-6,10	17
6460	2019-10-07	web	Windows	137914	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	39

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	N di
8341	2019-10-07	web	Mac OS	179971	Visit Optimization	PROSPECTING	0-6,10	66
6312	2019-10-07	web	iOS	137914	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	47
669	2019-10-07	app	iOS	182460	Visit Optimization	INAPP	2-10	13

In [245]:

```
# By ordering the column day, we show that we have same day appear mult
iple time so we group by day
#and sum the the Margin criteo per day
Groupby_Day=df.groupby(['Day'])['Margin_criteo'].sum()
Groupby_Day.head()
```

Out[245]:

```
Day
2019-10-07    8030.812709
2019-10-08    3550.634406
2019-10-09    4619.908384
2019-10-10    7172.925559
2019-10-11    6508.447351
Name: Margin_criteo, dtype: float64
```

In [246]:

```
# the shape become 116,
Groupby_Day.shape
```

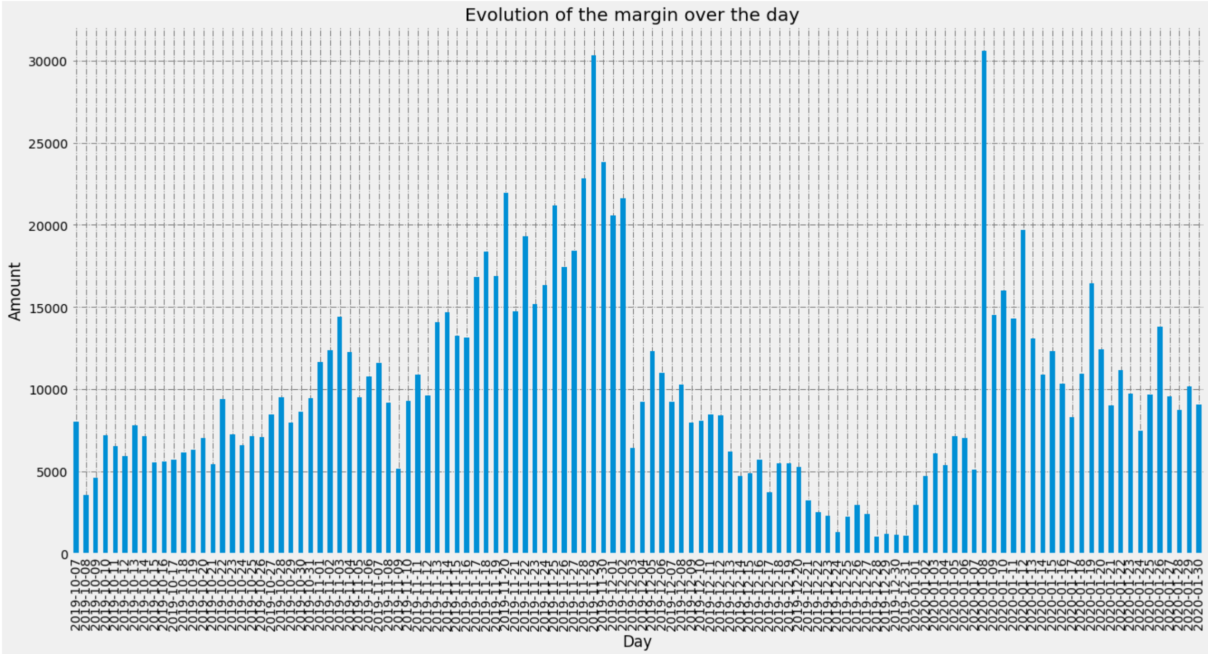
Out[246]:

```
(116,)
```

In [115]:

```
#We plot here the evolution of the Margin_criteo over the day
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
Groupby_Day.plot(kind='bar')
```

```python
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("Evolution of the margin over the day  " , fontsize = 20)
ax.set_xlabel("Day")
ax.set_ylabel("Amount")
plt.show()
```



We visualized the variation of the daily criteo margin, and we were able to identify days when the margin reached these maximums, this behavior for good reason, the days of the holidays (Christmas) and the days of the sales.

we do a general analysis for the margin criteo

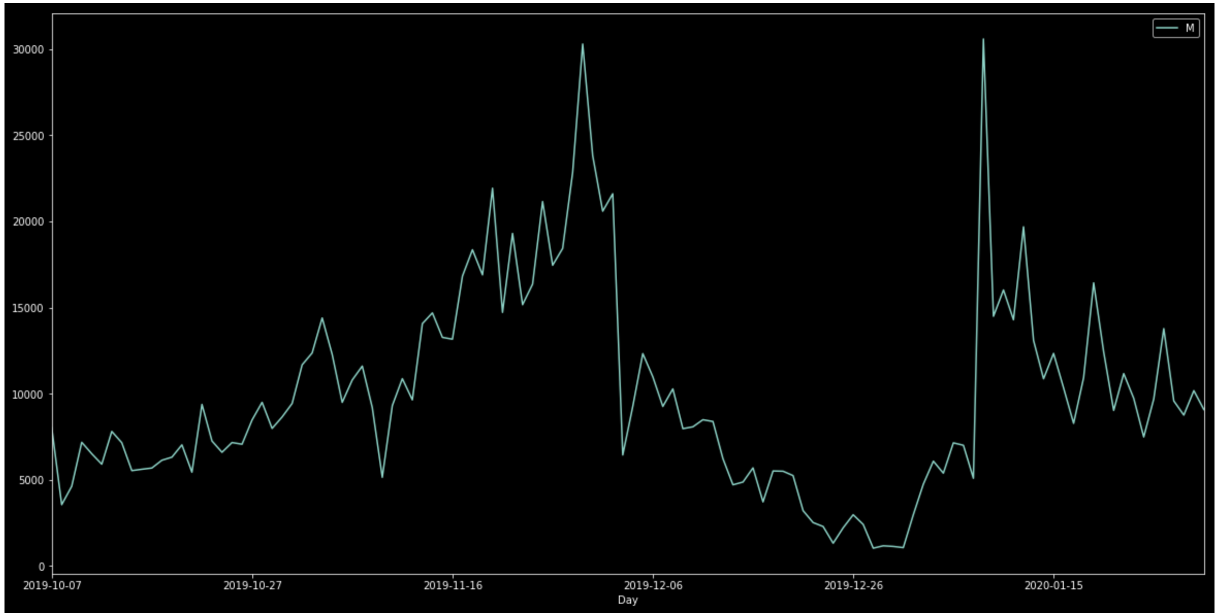
[2019/10/07 - 2019/10/31] the magin criteo approximately between [5000- 10000]

[2019/11/01 - 2019/11/16] the margin in evolution between [10000-15000]

[2019/11/17 - 2019/12/03] the most interest part where the graph reached the maximum that means represent the days of sales or maybe there are promotions for the end of the year

Then the graph decrease to reach less than 2000 in the end of year and return to the normal case

In [315]:
```
# check another view for the graph
fig, ax = plt.subplots(figsize=(20, 10))
plt.style.use("dark_background")
Groupby_Day.plot()
ax.legend('Margin_criteo')
plt.show()
```



To get more details on the peak days we zoom in, our previous explanations are confirmed by this plot, this peak coincides with the Christmas period.

In [247]:
```
df['date'] = pd.to_datetime(df['Day'])
```

In [248]:
```
#We count the 'Margin_Criteo ' by month
Groupby_Month=df.groupby(df['date'].dt.strftime('%B'))['Margin_criteo']
.sum().sort_values()
Groupby_Month
```

Out[248]:
```
date
October     173900.782808
December    196329.683308
January     326570.936044
November    455415.910048
Name: Margin_criteo, dtype: float64
```

In [20]:
```
Groupby_Month.shape
```

Out[20]:
```
(4,)
```

In [249]:
```
#We count the 'Margin_Criteo ' by week
Groupby_week = df.groupby(pd.Grouper(key='date', freq='1w'))['Margin_cr
iteo'].sum() # groupby each 1 month
Groupby_week.index = Groupby_week.index.strftime('%B')
Groupby_week
```

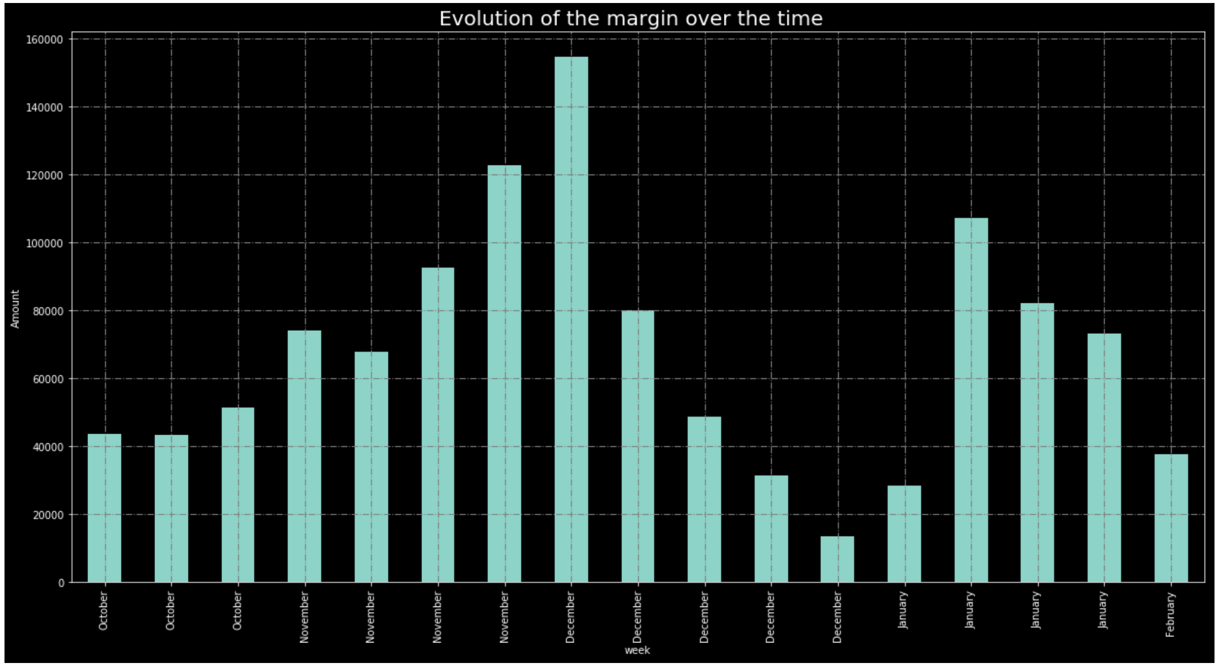
Out[249]:
```
October      43589.006068
October      43431.452893
October      51336.934737
November     73970.109082
November     67783.746074
November     92503.104001
November    122716.364956
December    154578.138384
December     80128.274951
December     48695.068910
December     31361.390754
December     13367.931904
January      28491.403962
January     107157.854273
January      82243.477033
January      73286.377216
```

```
February     37576.677009
Name: Margin_criteo, dtype: float64
```

In [119]:
```
Groupby_week.shape
```

Out[119]:
```
(17,)
```

In [320]:
```
#We plot here the evolution of the Margin_criteo over the week
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
Groupby_week.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("Evolution of the margin over the time  " , fontsize = 20)
ax.set_xlabel("week")
ax.set_ylabel("Amount")
plt.show()
```

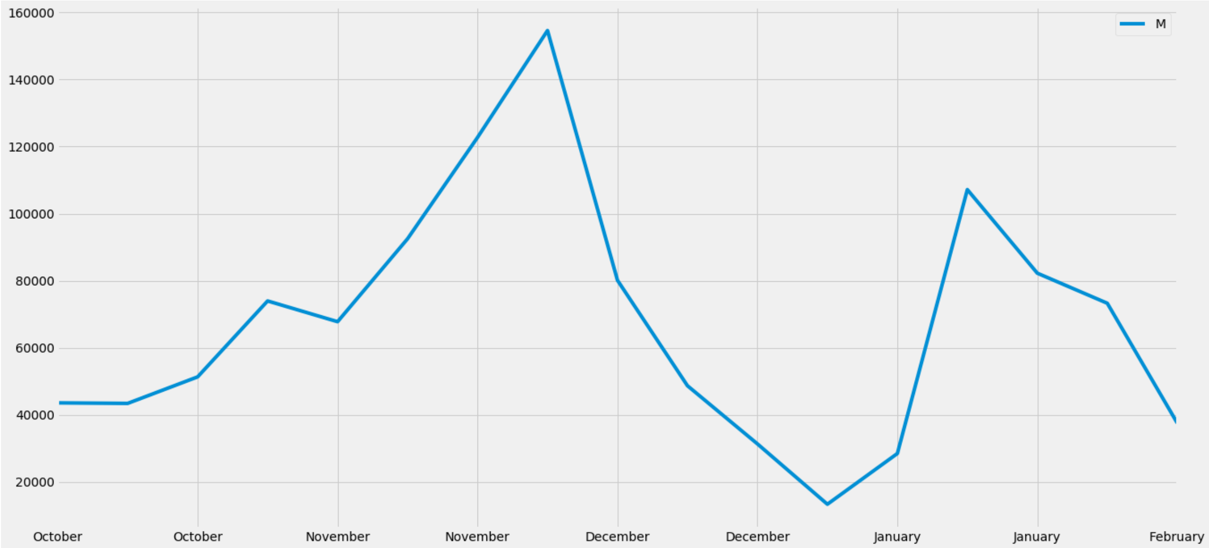


We study the variation of criteo margin per month. Criteo reached its peak in December, other remarkable margins reached the months of November and January.global peak in December, other remarkable margins are reached in November and January

In [25]:
```
#To check available style
plt.style.available
```

Out[25]:
```
['bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark-palette',
 'seaborn-dark',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'seaborn',
 'Solarize_Light2',
 '_classic_test']
```

In [33]:
```
# check another view for the graph we use here'fivethirtyeight'
fig, ax = plt.subplots(figsize=(20, 10))
plt.style.use("fivethirtyeight")
Groupby_week.plot()
ax.legend('Margin_criteo')
plt.show()
```



We have on the margin of December November and we have discovered that it is December we reached the maximum margin the first part of the month and the minimum margin the last part of the month

Note that the graph representing the margin criteo per day is more detailed than the second graph of margin criteo per week. we see here from the two graphs the margin criteo is alternative,is not stable but in general it is not bad

2.3. Plot the margin by Criteo product. Please comment/investigate the App Install margin.

In [250]:
```
#we groupby the compaign Id the sum of margin criteo
dg=df.groupby(['Campaign ID'])['Margin_criteo'].sum()
```

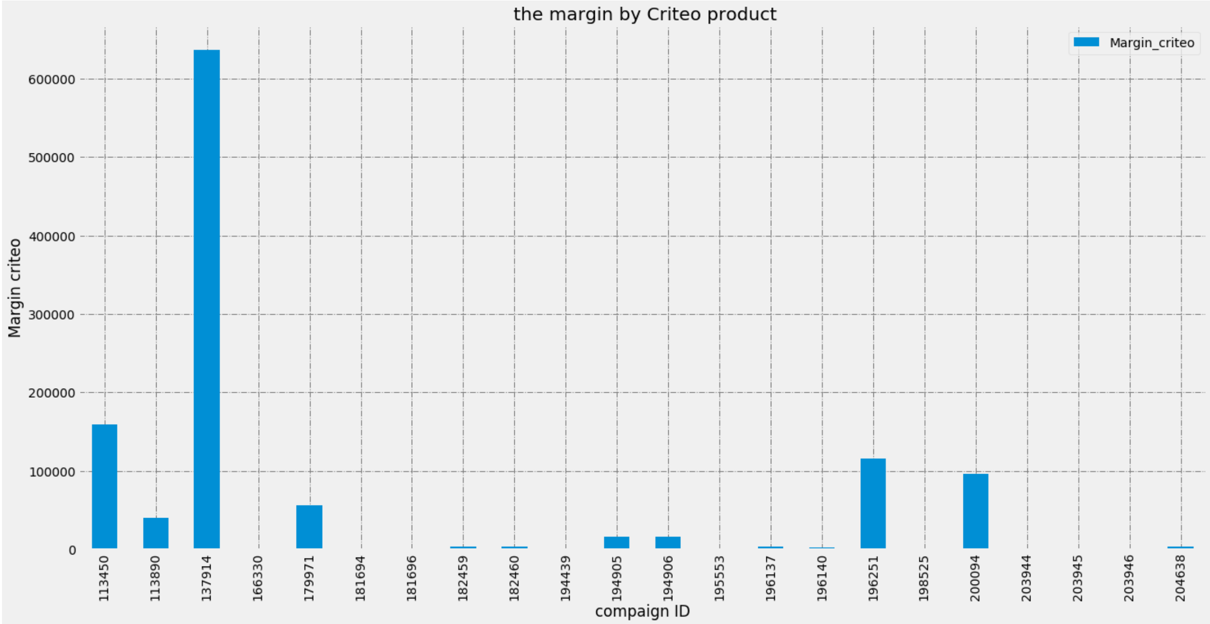
In [251]:
```
dg
```

Out[251]:
```
Campaign ID
113450     159478.455557
```

```
113890     39787.781498
137914    636310.196889
166330         0.000000
179971     55337.203695
181694       494.256597
181696       532.491244
182459      3108.481900
182460      2933.678806
194439      1423.389604
194905     15733.850053
194906     15914.905969
195553         1.085909
196137      2728.771397
196140      2598.219791
196251    115445.440253
198525       574.621791
200094     96495.489802
203944       187.583201
203945       120.411894
203946       152.612054
204638      2858.384303
Name: Margin_criteo, dtype: float64
```

In [180]:

```
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
dg.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title(" the margin by Criteo product" , fontsize = 20)
ax.set_xlabel("compaign ID")
ax.set_ylabel("Margin criteo")
plt.legend()
plt.show()
```



to plot the margin by criteo product:

the idea here to look for the compaign who allows criteo to have more margin, we found the compaign that had the ID = 137914 it's the most important one and we get more than 600000

Investigate the App install margin

In [252]:
```
#Check Campaign Type that have 'APP INSTALL '
APP_INSTALL =df[df['Campaign Type']=='APP INSTALL']
```

In [253]:
```
APP_INSTALL
```

Out[253]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb… display

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb[illegible] display
134	2019-10-16	web	Android	181696	Install Optimization	APP INSTALL	2-10	18546
135	2019-10-11	web	Unknown	181696	Install Optimization	APP INSTALL	2-10	58
136	2019-10-12	web	Android	181696	Install Optimization	APP INSTALL	2-10	20884
137	2019-10-27	web	Android	181696	Install Optimization	APP INSTALL	2-10	18835
138	2019-10-23	web	iOS	181696	Install Optimization	APP INSTALL	2-10	7
139	2019-10-23	web	Other	181696	Install Optimization	APP INSTALL	2-10	58
140	2019-10-25	web	Unknown	181696	Install Optimization	APP INSTALL	2-10	1
141	2019-10-08	web	iOS	181696	Install Optimization	APP INSTALL	2-10	9
142	2019-10-12	web	Unknown	181696	Install Optimization	APP INSTALL	2-10	1
143	2019-10-26	web	Other	181696	Install Optimization	APP INSTALL	2-10	45
144	2019-10-21	web	iOS	181696	Install Optimization	APP INSTALL	2-10	2
145	2019-10-14	web	iOS	181696	Install Optimization	APP INSTALL	2-10	1

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb… display
146	2019-10-17	web	iOS	181696	Install Optimization	APP INSTALL	2-10	3
147	2019-10-11	app	Android	181696	Install Optimization	APP INSTALL	2-10	64380
148	2019-11-22	web	Android	181696	Install Optimization	APP INSTALL	2-10	0
149	2019-11-24	app	Android	181696	Install Optimization	APP INSTALL	2-10	0
150	2019-10-29	app	Android	181696	Install Optimization	APP INSTALL	2-10	2
151	2019-10-08	web	Mac OS	181696	Install Optimization	APP INSTALL	2-10	2
152	2019-10-15	web	Other	181696	Install Optimization	APP INSTALL	2-10	27
153	2019-10-28	app	Android	181696	Install Optimization	APP INSTALL	2-10	10211
154	2019-10-24	web	Android	181696	Install Optimization	APP INSTALL	2-10	16426
155	2019-10-22	web	Windows	181696	Install Optimization	APP INSTALL	2-10	2
156	2019-10-07	web	Android	181696	Install Optimization	APP INSTALL	2-10	22401
157	2019-10-10	web	Android	181696	Install Optimization	APP INSTALL	2-10	11638

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb display
158	2019-10-25	app	Android	181696	Install Optimization	APP INSTALL	2-10	49293
159	2019-11-11	app	Android	181696	Install Optimization	APP INSTALL	2-10	0
160	2019-10-10	web	Other	181696	Install Optimization	APP INSTALL	2-10	11
161	2019-10-23	web	Android	181696	Install Optimization	APP INSTALL	2-10	19228
162	2019-10-07	web	Other	181696	Install Optimization	APP INSTALL	2-10	73
163	2019-10-18	web	Windows	181696	Install Optimization	APP INSTALL	2-10	1
...	...	...	...	...	...	...	...	...
969	2019-10-24	web	Other	181694	Install Optimization	APP INSTALL	2-10	8
970	2019-11-10	app	iOS	181694	Install Optimization	APP INSTALL	2-10	0
971	2019-10-20	app	iOS	181694	Install Optimization	APP INSTALL	2-10	100509
972	2019-10-08	app	iOS	181694	Install Optimization	APP INSTALL	2-10	85028
973	2019-10-25	web	Other	181694	Install Optimization	APP INSTALL	2-10	3

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb display
974	2019-10-28	app	iOS	181694	Install Optimization	APP INSTALL	2-10	23100
975	2019-10-16	web	Windows	181694	Install Optimization	APP INSTALL	2-10	1
976	2019-10-14	web	iOS	181694	Install Optimization	APP INSTALL	2-10	2782
977	2019-10-25	app	iOS	181694	Install Optimization	APP INSTALL	2-10	91300
978	2019-10-31	app	iOS	181694	Install Optimization	APP INSTALL	2-10	0
979	2019-10-12	app	iOS	181694	Install Optimization	APP INSTALL	2-10	140274
980	2019-10-11	web	Other	181694	Install Optimization	APP INSTALL	2-10	3
981	2019-10-16	web	Other	181694	Install Optimization	APP INSTALL	2-10	8
982	2019-11-01	app	iOS	181694	Install Optimization	APP INSTALL	2-10	0
983	2019-10-18	web	iOS	181694	Install Optimization	APP INSTALL	2-10	3021
984	2019-10-21	web	Other	181694	Install Optimization	APP INSTALL	2-10	2
985	2019-10-12	web	Mac OS	181694	Install Optimization	APP INSTALL	2-10	1

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb display
986	2019-10-15	web	Other	181694	Install Optimization	APP INSTALL	2-10	5
987	2019-10-10	web	Other	181694	Install Optimization	APP INSTALL	2-10	3
988	2019-10-17	web	Unknown	181694	Install Optimization	APP INSTALL	2-10	1
989	2019-10-18	web	Windows	181694	Install Optimization	APP INSTALL	2-10	2
990	2019-11-22	app	iOS	181694	Install Optimization	APP INSTALL	2-10	0
991	2019-10-26	web	Other	181694	Install Optimization	APP INSTALL	2-10	1
996	2019-10-07	web	iOS	181694	Install Optimization	APP INSTALL	2-10	5333
997	2019-10-07	app	iOS	181694	Install Optimization	APP INSTALL	2-10	153114
998	2019-10-08	web	Android	181694	Install Optimization	APP INSTALL	2-10	1
1007	2019-10-12	web	iOS	181694	Install Optimization	APP INSTALL	2-10	4570
1008	2019-10-08	web	Mac OS	181694	Install Optimization	APP INSTALL	2-10	6
1009	2019-10-26	app	iOS	181694	Install Optimization	APP INSTALL	2-10	77498

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Numb[...] display
1010	2019-10-07	web	Windows	181694	Install Optimization	APP INSTALL	2-10	2

217 rows × 13 columns

In [254]:
```
#We calculate the margin_criteo for the type compaign criteo APP_INSTAL
L
APP_INSTALL ['Margin_criteo'].sum()
```

Out[254]: 1026.7478413068789

we calculate the total Margin criteo and we found 8828 (we get it from the df.describe) and for this type compaign APP_INSTALL we get 1026 of the total 8828.

Exploring Data 2 : matched_sales

matched_sales.tsv – Sales of an e-commerce website that are attributed to Criteo, using a post-click 30D attribution rule

In [255]:
```
#Read table
import pandas as pd
df1=pd.read_table(r'C:\Users\monjia\Downloads\matched_sales.tsv',sep='
\t')
```

In [256]:
```
df1.head(10)
```

Out[256]:

	Timestamp	Environment	Os	Device	Campaign ID	
0	1569889040	web	Windows	Desktop	113450	2EEEE2185BB347E076531
1	1569890180	web	Android	Smartphone	137914	18F4B9C704A39843FE435
2	1569890704	web	Windows	Desktop	137914	513CD8C04005C7ED66F34
3	1569892066	web	Android	Smartphone	137914	1620E50E507DDF33489D8
4	1569892673	web	Windows	Desktop	137914	85AC7958028EFF2231B49
5	1569892689	web	Android	Smartphone	137914	8A5E21F9B4EF17BEA1001
6	1569892900	web	Mac OS	Desktop	137914	BF1BC289862CD559584BF
7	1569893490	web	Android	Smartphone	137914	D505600BB8926FE75C10E
8	1569893697	web	Android	Tablet	137914	2C7F0DD62A7F9AD330A0
9	1569893876	web	Android	Smartphone	137914	45F39915EF95F5F798C50

In [257]:
```
print('the shape of df1:',df1.shape)
print('the size of df1:',df1.size)
```

```
the shape of df1: (149809, 9)
the size of df1: 1348281
```

In [258]:
```
df1.columns
```

Out[258]:
```
Index(['Timestamp', 'Environment', 'Os', 'Device', 'Campaign ID', 'User
ID',
       'Context ID', 'Transaction ID', 'Order Value'],
      dtype='object')
```

In [259]:
```
#print train data columns
print('Columns of the matched_sales dataset is', df1.columns)
```

```
Columns of the matched_sales dataset is Index(['Timestamp', 'Environmen
```

```
t', 'Os', 'Device', 'Campaign ID', 'User ID',
       'Context ID', 'Transaction ID', 'Order Value'],
      dtype='object')
```

In [260]:
```
#Describe table
df1.describe()
```

Out[260]:

	Timestamp	Campaign ID	Context ID	Order Value
count	1.498090e+05	149809.000000	149809.000000	149808.000000
mean	1.572520e+09	141789.036753	6.652591	76.258191
std	1.430771e+06	20260.271847	2.274443	172.129753
min	1.569889e+09	113450.000000	0.000000	0.000000
25%	1.571261e+09	137914.000000	6.000000	29.140000
50%	1.572638e+09	137914.000000	7.000000	48.770000
75%	1.573824e+09	137914.000000	8.000000	85.590000
max	1.574674e+09	200094.000000	10.000000	51658.000000

In [261]:
```
#info table
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149809 entries, 0 to 149808
Data columns (total 9 columns):
Timestamp         149809 non-null int64
Environment       149809 non-null object
Os                149809 non-null object
Device            149809 non-null object
Campaign ID       149809 non-null int64
User ID           149809 non-null object
Context ID        149809 non-null int64
Transaction ID    149786 non-null object
Order Value       149808 non-null float64
```

```
dtypes: float64(1), int64(3), object(5)
memory usage: 10.3+ MB
```

In [262]:
```
#nan value
df1.isnull().sum()
```

Out[262]:
```
Timestamp          0
Environment        0
Os                 0
Device             0
Campaign ID        0
User ID            0
Context ID         0
Transaction ID    23
Order Value        1
dtype: int64
```

In [263]:
```
#Check nan valuer per column
df1['Order Value'].unique()
```

Out[263]:
```
array([  32.08,  149.92,  125.1 , ..., 1100.72,   93.78,      nan])
```

In [264]:
```
len(df1['Order Value'].unique())
```

Out[264]:
```
26904
```

In [265]:
```
#dropthe nan value
df1['Order Value'].nunique()
```

Out[265]:
```
26903
```

In [266]:
```
#zero value
zero_val = (df1 == 0.00).sum(axis=0)
zero_val
```

Out[266]:
```
Timestamp             0
Environment           0
Os                    0
Device                0
```

```
Campaign ID            0
User ID                0
Context ID         10717
Transaction ID         0
Order Value         1668
dtype: int64
```

1.1. Use data visualization techniques to explore the dataset and identify data issue. Clean the dataset accordingly.

Tips: Transactions with an Order Value of zero should be removed; Transactions with abnormally high Order Value should be removed; Duplicated Transactions should be removed.

In [267]:
```
#We check the box plot order value
fig,ax = plt.subplots()
df1.boxplot(column='Order Value')
plt.show()
```

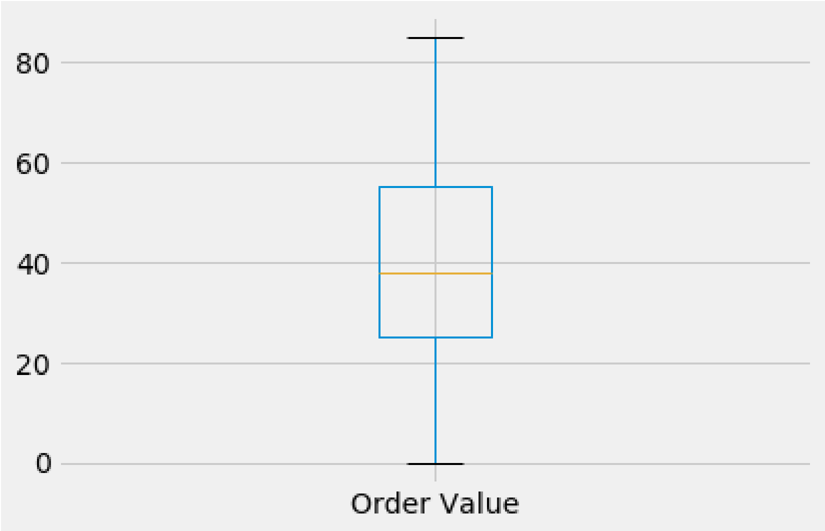
In [268]:
```
#1/ Remove zero value
df1.drop(df1[df1['Order Value']== 0 ].index, inplace = True )
```

```
#2/ transaction with abnormally value, to choose it i check 75% of orde
r values and it's equal to 85.59 i get it from the descibe table
#so for that i choose 85
df1.drop(df1[df1['Order Value'] > 85 ].index, inplace = True )
```

When i plot the box plot before removing the order value greater than 86 i can't get a clear visualization because there are a lot of outliers at the time I thought of eliminating these values using the mustache box (it comes down to eliminating the 3 rd quartile)

And know i re plot the graph after deleting the outliers values and here is the result

In [269]:
```
#fig = plt.figure(figsize=(8,6))
fig,ax = plt.subplots()
df1.boxplot(column='Order Value')
plt.show()
```



In [270]:
```
#to check that the zero value removed
df1[df1['Order Value']== 0 ].count()
```

Out[270]:
```
Timestamp          0
Environment        0
```

```
Os                 0
Device             0
Campaign ID        0
User ID            0
Context ID         0
Transaction ID     0
Order Value        0
dtype: int64
```

In [271]:
```
#We check duplicate Transaction id in the table
df1.duplicated(subset=['Transaction ID']).value_counts()
# we have 143 duplicate transaction id
```

Out[271]:
```
False    110280
True        150
dtype: int64
```

In [272]:
```
#to remove the duplicate value i kept the high value of order value
df1 = df1.sort_values(by='Order Value', ascending=False)
df1 = df1.drop_duplicates(subset='Transaction ID', keep="first")
```

In [273]:
```
#To check that we delete the duplicate transaction id
df1.duplicated(subset=['Transaction ID']).value_counts()
```

Out[273]:
```
False    110280
dtype: int64
```

Here we have the User ID is the Cookie-centric identifier of the user (for each browser or device used there is a different user ID)

so we check for each device we need different user ID

In [274]:
```
Device=df1.groupby(['Device'])['User ID'].count()
```

In [275]:
```
Device
```

Out[275]:
```
Device
```

```
Desktop       58489
Other             5
Smartphone    35072
Tablet        11925
Unknown        4789
Name: User ID, dtype: int64
```

Here for example for Desktop should the user have different user ID (58489 different id)

In [276]:
```
#we check that the user id be different
Device.duplicated()
```

Out[276]:
```
Device
Desktop       False
Other         False
Smartphone    False
Tablet        False
Unknown       False
Name: User ID, dtype: bool
```

In [277]:
```
#Same idea that i put it for the device should we have different user i
d using os
Os=df1.groupby(['Os'])['User ID'].count()
```

In [278]:
```
Os.duplicated()
```

Out[278]:
```
Os
Android    False
Mac OS     False
Other      False
Unknown    False
Windows    False
iOS        False
Name: User ID, dtype: bool
```

So here we checked that we have different user id

In [279]:
```
df1.shape
```
Out[279]: (110280, 9)

1.2. Can you estimate the time zone of the client?

In [280]:
```
#to convert the timestamp into date
df1 = df1.set_index(['Timestamp'])
df1.index = pd.to_datetime(df1.index, unit='s')
```

In [281]:
```
df1.head()
```
Out[281]:

	Environment	Os	Device	Campaign ID	User ID
Timestamp					
2019-10-12 16:12:46	web	Windows	Desktop	137914	B0E28FBAF7F6FB96321C2DE
2019-10-02 14:24:07	web	Windows	Desktop	137914	43BE93DCB6E54D266E71485I
2019-10-31 06:56:08	web	iOS	Smartphone	137914	5485E7A857ED699F72618EC8
2019-11-03 21:49:36	web	Windows	Desktop	137914	D15CB0B8488F4B0B9B3FCCD
2019-11-18 19:05:40	web	Android	Smartphone	137914	D3F6BADA8D110A0C13EB232

In [282]:
```
df1.shape
```
Out[282]: (110280, 8)

1.3. What is the evolution of the Criteo attributed revenue over time? Can you pinpoint and explain particular events during the period

Here for the evolution of the criteo attributed revenue i choose to plot it over the day and week

In [141]:
```
#Resampe per day
Resample_per_day=df1['Order Value'].resample('d').sum()
```

In [283]:
```
Resample_per_day
```

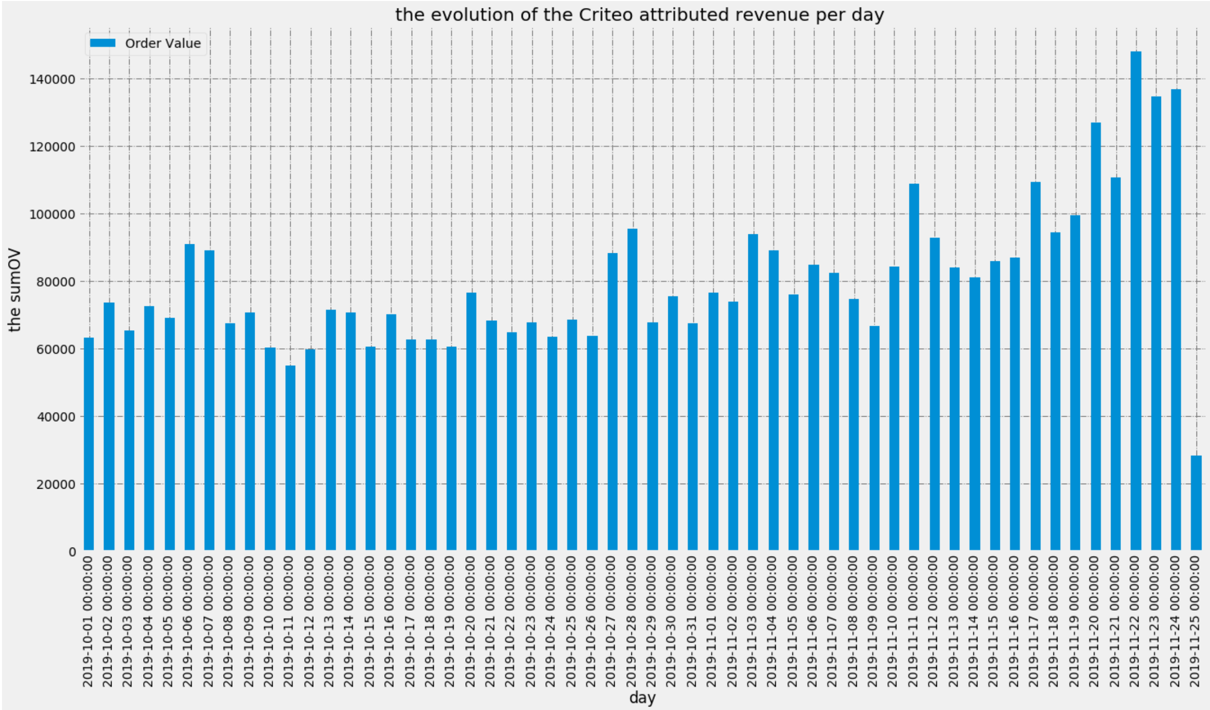
Out[283]:
```
Timestamp
2019-10-01     63156.60
2019-10-02     73615.05
2019-10-03     65497.07
2019-10-04     72726.27
2019-10-05     69024.29
2019-10-06     91084.52
2019-10-07     89267.04
2019-10-08     67498.88
2019-10-09     70863.00
2019-10-10     60277.65
2019-10-11     54976.92
2019-10-12     59816.62
2019-10-13     71497.08
2019-10-14     70788.08
2019-10-15     60482.68
2019-10-16     70096.95
2019-10-17     62690.79
2019-10-18     62671.81
2019-10-19     60527.80
2019-10-20     76671.65
2019-10-21     68379.19
2019-10-22     64780.12
2019-10-23     67899.43
2019-10-24     63461.79
2019-10-25     68638.14
```

```
2019-10-26     63718.03
2019-10-27     88417.88
2019-10-28     95633.61
2019-10-29     67814.57
2019-10-30     75530.49
2019-10-31     67530.79
2019-11-01     76511.10
2019-11-02     74062.07
2019-11-03     94039.30
2019-11-04     89243.62
2019-11-05     76208.83
2019-11-06     84832.88
2019-11-07     82610.26
2019-11-08     74750.08
2019-11-09     66768.37
2019-11-10     84335.37
2019-11-11    108963.43
2019-11-12     92775.56
2019-11-13     84161.62
2019-11-14     81085.21
2019-11-15     85888.03
2019-11-16     87127.50
2019-11-17    109493.32
2019-11-18     94575.35
2019-11-19     99513.10
2019-11-20    127085.59
2019-11-21    110799.13
2019-11-22    148239.77
2019-11-23    134703.87
2019-11-24    137025.12
2019-11-25     28400.16
Freq: D, Name: Order Value, dtype: float64
```

In [284]:

```
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
Resample_per_day.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the evolution of the Criteo attributed revenue per day "
, fontsize = 20)
```

```
ax.set_xlabel("day")
ax.set_ylabel("the sumOV")
ax.legend()
plt.show()
```



Here we plot the criteo revenu over the week

In [285]:
```
#Resample per week
Resample_per_week=df1['Order Value'].resample('w').sum()
```

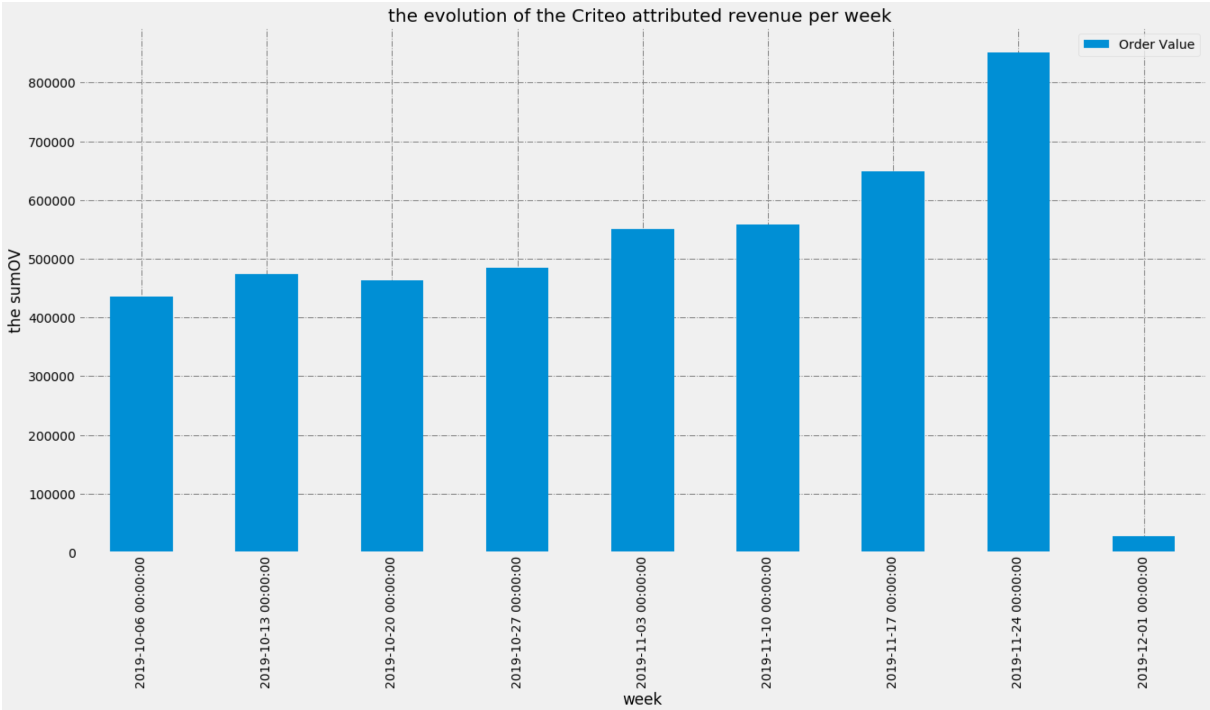
In [286]:
```
Resample_per_week
```

Out[286]:
```
Timestamp
2019-10-06    435103.80
2019-10-13    474197.19
2019-10-20    463929.76
```

```
2019-10-27    485294.58
2019-11-03    551121.93
2019-11-10    558749.41
2019-11-17    649494.67
2019-11-24    851941.93
2019-12-01     28400.16
Freq: W-SUN, Name: Order Value, dtype: float64
```

In [287]:

```python
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
Resample_per_week.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the evolution of the Criteo attributed revenue per week "
 , fontsize = 20)
ax.set_xlabel("week")
ax.set_ylabel("the sumOV")
ax.legend()
plt.show()
```



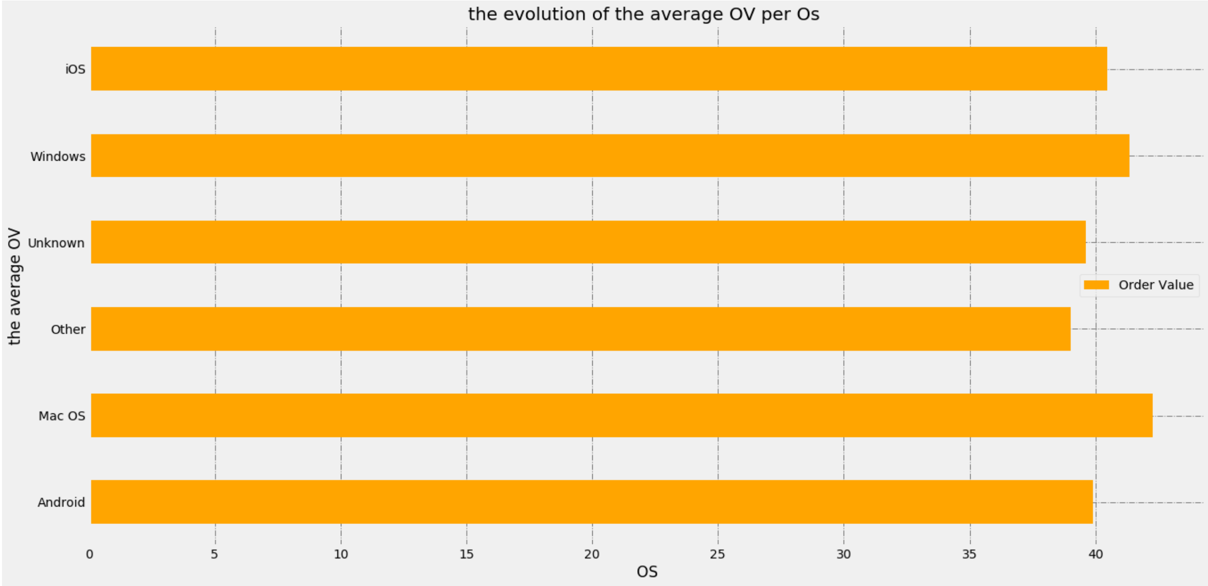
For the two graphs that we plot per week or per day we show an augmentation of the sum of the order value We got a slight increase during the 5 weeks and then and then a pick for the 6th week (2019/11/24) end of year this from the chrismas

1.4. Propose a vizualization for the evolution of the average OV per Os. Tips: Watch out for transactions with empty Transaction I

In [288]:
```
Groupby_average_OV=df1.groupby(['Os'])['Order Value'].mean()
Groupby_average_OV
```

Out[288]:
```
Os
Android     39.906754
Mac OS      42.265046
Other       39.018320
Unknown     39.623243
Windows     41.356928
iOS         40.470485
Name: Order Value, dtype: float64
```

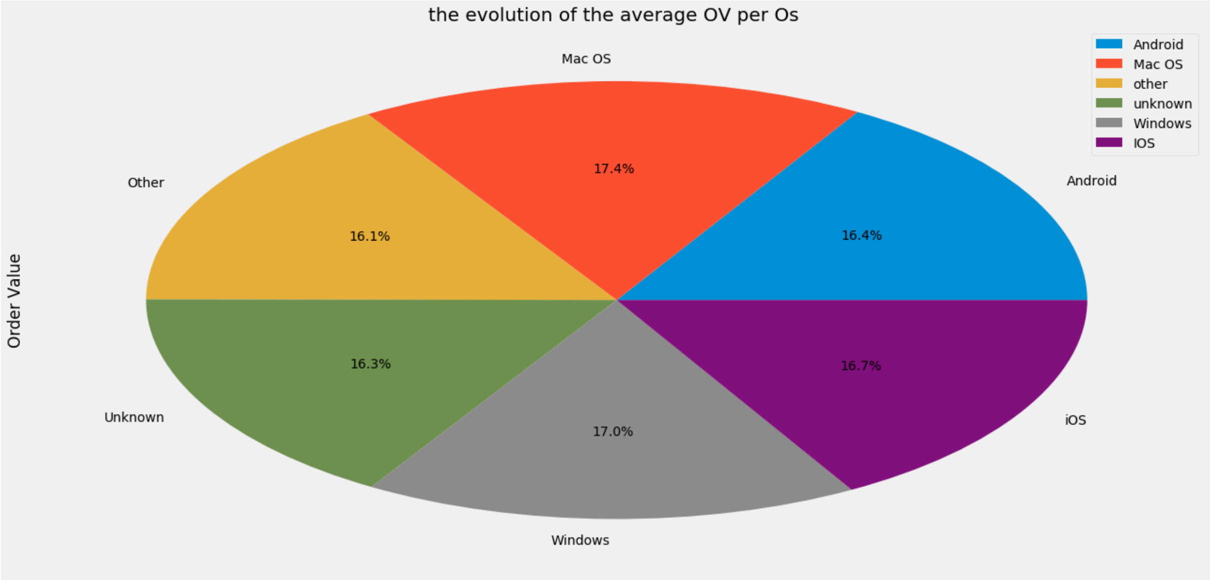
In [289]:
```
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
Groupby_average_OV.plot(kind='barh',color='orange')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the evolution of the average OV per Os " , fontsize = 20)
ax.set_xlabel("OS")
ax.set_ylabel("the average OV")
ax.legend()
plt.show()
```



The OS most used by users are mac OS first and windows in second positions. we notice that there is not a remarkable difference between all the beats

In [290]:
```
fig, ax = plt.subplots(figsize=(20, 10))

Groupby_average_OV.plot.pie(autopct='%.1f%%')
labels = [ "Android" , "Mac OS","other","unknown","Windows","IOS"]
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the evolution of the average OV per Os " , fontsize = 20)
ax.legend(labels)
plt.show()
```



We also present the variation in order valueby OS in the form of pie plot.

In [291]:
```
#To check the empty rows in the column 'Transaction ID'
DF_new_row=df1.loc[df1['Transaction ID']=='']
DF_new_row.sum()
```

Out[291]:
```
Environment       0.0
Os                0.0
Device            0.0
Campaign ID       0.0
User ID           0.0
Context ID        0.0
Transaction ID    0.0
Order Value       0.0
dtype: float64
```

Part 3

3. Join the 2 datasets

In [292]:
```
df.columns
```

Out[292]:
```
Index(['Day', 'Environment', 'Os', 'Campaign ID', 'Campaign Optimizatio
n',
       'Campaign Type', 'Context IDs Eligible', 'Number of displays',
       'Number of clicks', 'Criteo Revenue', 'Criteo Cost', 'Margin_cri
teo',
       'date'],
      dtype='object')
```

In [293]:
```
df1.columns
```

Out[293]:
```
Index(['Environment', 'Os', 'Device', 'Campaign ID', 'User ID', 'Contex
t ID',
       'Transaction ID', 'Order Value'],
      dtype='object')
```

In [294]:
```
# To join two table we need to the commun columns
df2=pd.merge(df, df1)
```

In [295]:
```
df2.head()
```

Out[295]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays	N
0	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	0
1	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Number of displays	N
2	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	[illegible]
3	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	[illegible]
4	2019-10-16	web	Windows	194439	Visit Optimization	MID FUNNEL CUSTOM	0-6,10	0	[illegible]

In [296]:
```
df2[df2['Campaign Type']=='LOWER FUNNEL CUSTOM']
```

Out[296]:

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
44047	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44048	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44049	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu dis
44050	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44051	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44052	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44053	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44054	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44055	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44056	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44057	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
44058	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44059	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44060	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44061	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44062	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44063	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44064	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44065	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
44066	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44067	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44068	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44069	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44070	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44071	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44072	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44073	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
44074	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44075	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
44076	2019-12-13	web	Android	113450	Conversion Optimization	LOWER FUNNEL CUSTOM	2-6,10	364
...	...	...	...	...	...	...	...	...
12045571	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045572	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045573	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045574	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045575	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
12045576	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045577	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045578	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045579	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045580	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045581	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045582	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045583	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu disp
12045584	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045585	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045586	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045587	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045588	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045589	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045590	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045591	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu dis
12045592	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045593	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045594	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045595	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045596	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045597	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045598	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0
12045599	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

	Day	Environment	Os	Campaign ID	Campaign Optimization	Campaign Type	Context IDs Eligible	Nu dis
12045600	2019-11-15	web	iOS	194906	Conversion Optimization	LOWER FUNNEL CUSTOM	0-10	0

11782366 rows × 18 columns

Statistic tools

In [297]:
```
df2.shape
```

Out[297]: (12060681, 18)

In [141]:
```
df2.describe()
```

Out[141]:

	Campaign ID	Number of displays	Number of clicks	Criteo Revenue	Criteo Cost	Margin_c
count	1.136067e+07	1.136067e+07	1.136067e+07	1.136067e+07	1.136067e+07	1.136067
mean	1.381802e+05	1.695578e+06	8.634799e+03	3.323853e+03	1.915434e+03	1.408419
std	1.525379e+04	1.429901e+06	6.418731e+03	3.503857e+03	2.059148e+03	1.448653
min	1.134500e+05	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-7.53855
25%	1.379140e+05	4.669170e+05	2.540000e+03	7.015284e+02	3.946768e+02	3.017487
50%	1.379140e+05	1.328849e+06	7.492000e+03	2.025184e+03	1.127593e+03	9.149666
75%	1.379140e+05	2.750505e+06	1.375000e+04	4.855827e+03	2.716917e+03	2.075248
max	2.000940e+05	6.413055e+06	3.485200e+04	1.805521e+04	1.058762e+04	7.467593

Part 4

4. Analyze Criteo Campaigns Performance from the advertiser’s perspective

4.1. Identify the most performing (using performance and scale criteria) campaigns using a viz.

To identifie the most performing campaigns there are many methods

Idea 1: we can plot the Margin Criteo per compaign Id

Idea 2: we can plot the Order value per compaign Id

Idea 3: We can check the number of clicks or displays for each compaign

Idea 4: We can check the number of context ID if we have the compaigns contain more number (df2['Context ID']>6) then it will be the most performing

(i choose >6 : because 7 The visitor has made one purchase (1 or several products bought)

8 The visitor has made 2 or 3 purchases (1 or several products bought per purchase)

9 The visitor has made 4 or more purchases (1 or several products bought per purchase))

In [298]:
```
#first idea
dg=df2.groupby(['Campaign ID'])['Margin_criteo'].sum()
```

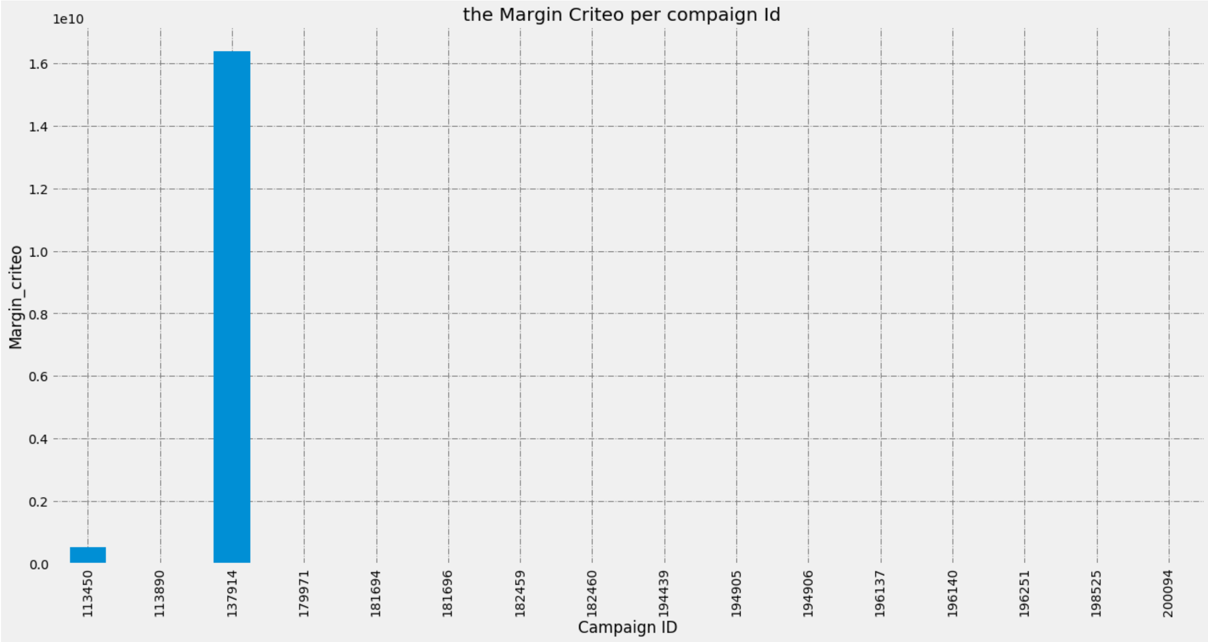
In [299]:
```
dg
```

Out[299]:
```
Campaign ID
113450    5.155450e+08
113890    2.156041e+07
```

```
137914    1.637404e+10
179971    9.338911e+06
181694    2.198270e+04
181696    1.485005e+04
182459    4.029755e+05
182460    7.297193e+05
194439    2.365378e+04
194905    1.023823e+07
194906    2.316891e+07
196137    3.807362e+04
196140    1.246386e+05
196251    2.041989e+07
198525    9.504636e+03
200094    9.001079e+06
Name: Margin_criteo, dtype: float64
```

In [300]:

```
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
dg.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the Margin Criteo per compaign Id  " , fontsize = 20)
ax.set_xlabel("Campaign ID")
ax.set_ylabel("Margin_criteo")
plt.show()
```



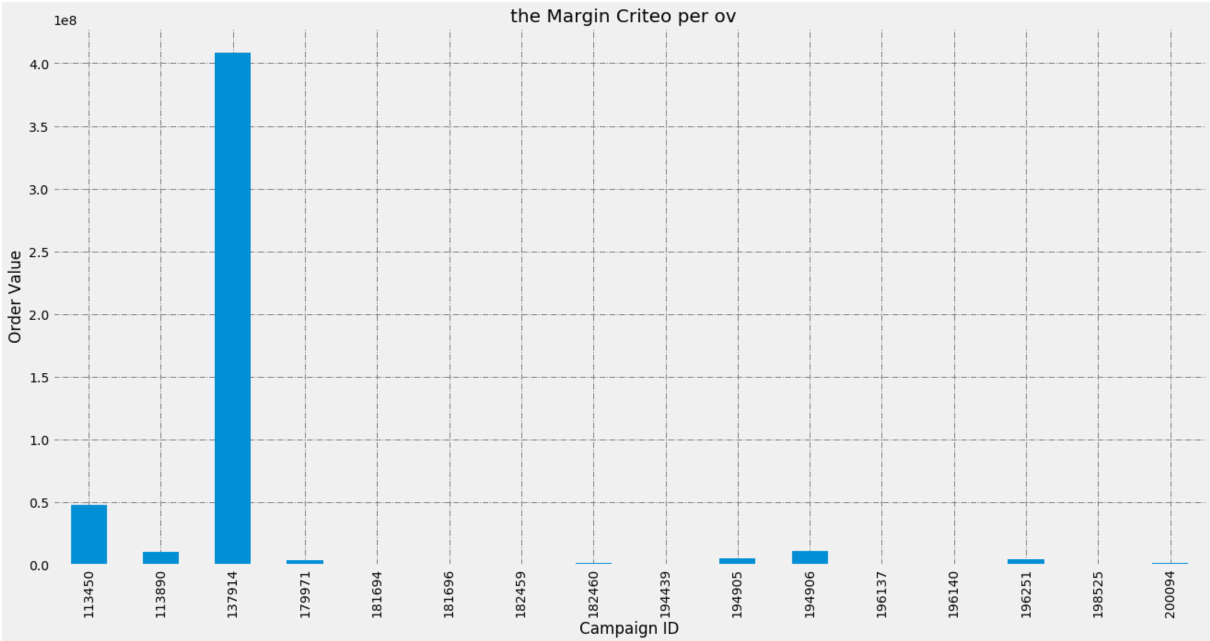
In [301]:
```
methode_2=df2.groupby(['Campaign ID'])['Order Value'].sum()
```

In [302]:
```
methode_2
```

Out[302]:
```
Campaign ID
113450    4.752088e+07
113890    9.845014e+06
137914    4.081528e+08
179971    3.326958e+06
181694    8.392262e+04
181696    6.590150e+04
182459    5.884344e+05
182460    1.159978e+06
194439    7.382039e+04
194905    4.815408e+06
194906    1.066331e+07
196137    6.227308e+04
196140    2.088360e+05
196251    3.940298e+06
```

```
198525    9.942250e+04
200094    1.502195e+06
Name: Order Value, dtype: float64
```

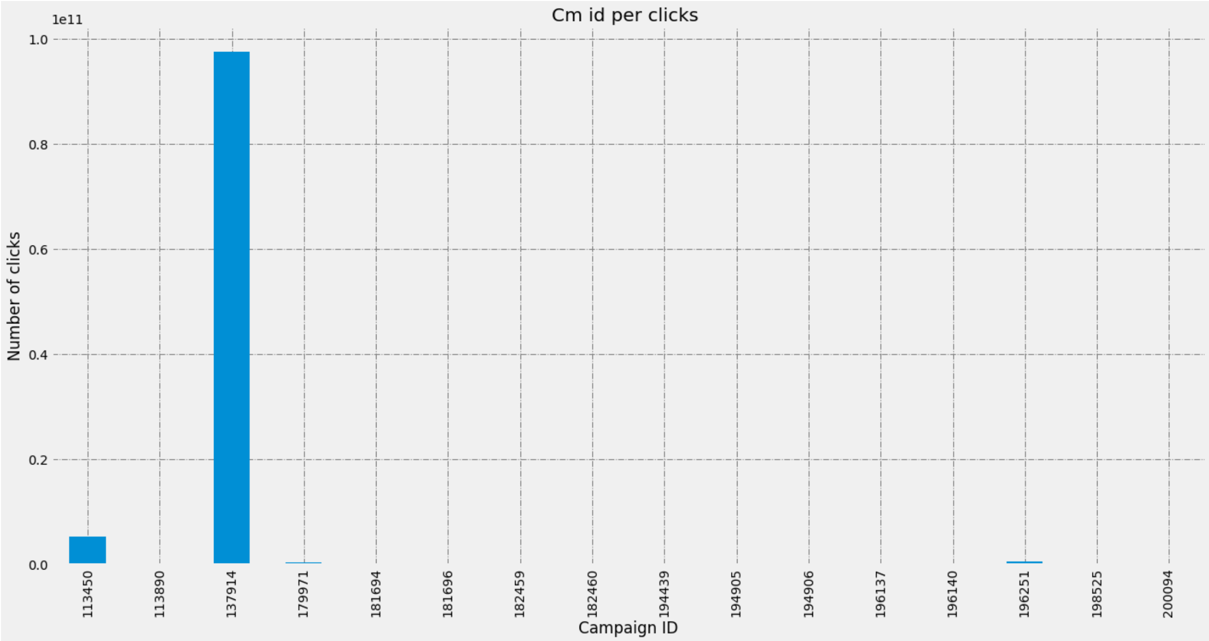
In [303]:
```python
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
methode_2.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("the Margin Criteo per ov  " , fontsize = 20)
ax.set_xlabel("Campaign ID")
ax.set_ylabel("Order Value")
plt.show()
```



In [304]:
```python
methode_3=df2.groupby(['Campaign ID'])['Number of clicks'].sum()
```

In [305]:
```python
fig, ax = plt.subplots(figsize=(20, 10))
# Add x-axis and y-axis
methode_3.plot(kind='bar')
ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
```

```
ax.set_title("Cm id per clicks " , fontsize = 20)
ax.set_xlabel("Campaign ID")
ax.set_ylabel("Number of clicks")
plt.show()
```



For each method that we treated here we get the same compaign Id but we have problem to see the other compaign

4.2. The previous viz made it challenging to see patterns. Can you adapt it? Which Criteo campaign types perform the best? Tips: use Campaign Type instead

In [306]:
```
campaign_types=df2.groupby(['Campaign Type'])['Margin_criteo'].sum()
```

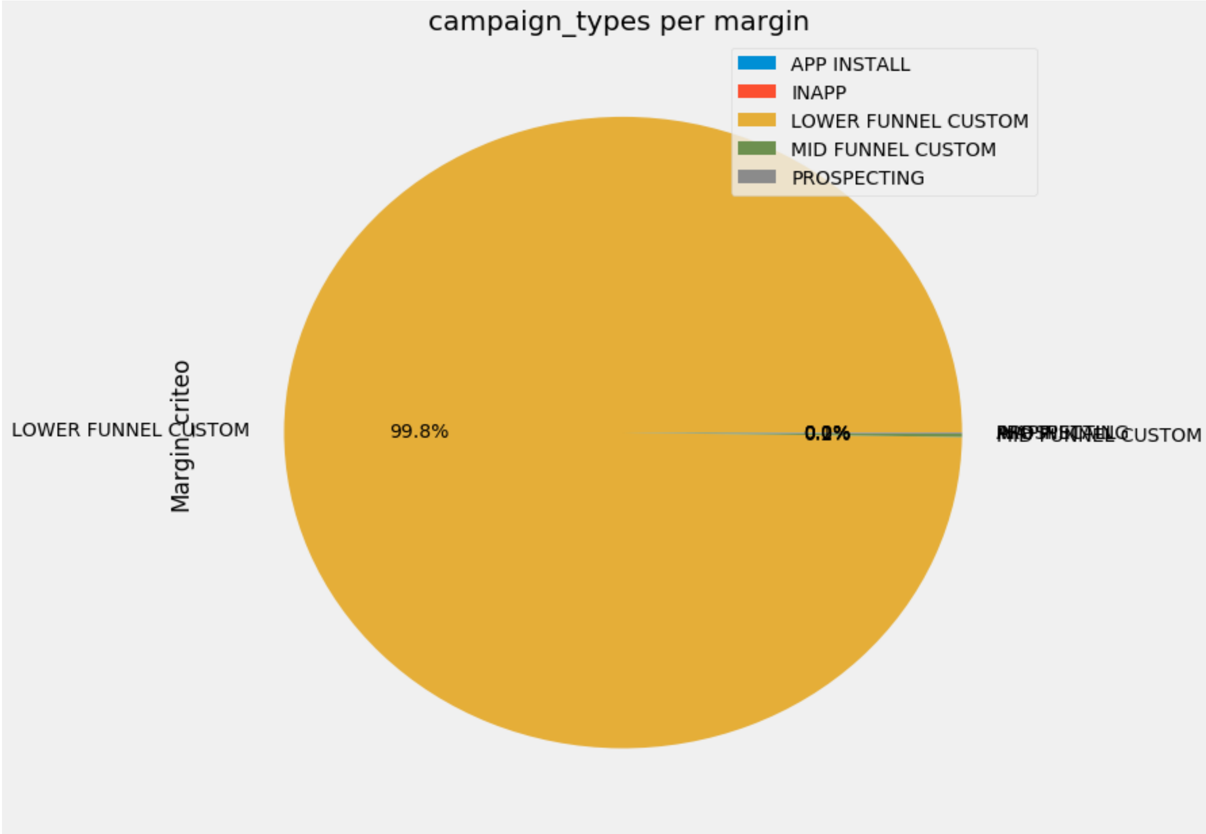
In [307]:
```
campaign_types
```

Out[307]:
```
Campaign Type
APP INSTALL            3.683275e+04
```

```
INAPP                       1.295407e+06
LOWER FUNNEL CUSTOM         1.694455e+10
MID FUNNEL CUSTOM           2.945413e+07
PROSPECTING                 9.338911e+06
Name: Margin_criteo, dtype: float64
```

In [311]:
```
fig, ax = plt.subplots(figsize=(10, 10))

campaign_types.plot.pie(autopct='%.1f%%')
labels = [ "APP INSTALL " , "INAPP","LOWER FUNNEL CUSTOM","MID FUNNEL C
USTOM","PROSPECTING"]
#ax.grid(linestyle = '-.' , linewidth = 1 , color = '.5')
ax.set_title("campaign_types per margin " , fontsize = 20)
ax.legend(labels)
plt.show()
```



The best compaign we get 99.8 percent for LOWER FUNNEL CUSTOM and 0.2 between the other

i check it with hist plot and i get same result

4.3. Is the COS/ROAS a relevant metric to look at for all Criteo campaign types? What would you recommend looking at instead?

COS (Cost Of Sales):The ratio between the total cost of the campaign and the sales amount that the campaign generated. COS provides a good indication of your campaign's ROI.

COS = Revenue / Sales post-click

ROAS (Return On Advertising Spend): Corresponds to the ratio between the Sales Amount the campaign generated and the total Cost of the campaign.

ROAS = 1/COS

To check the COS/ROAS is a relevant metric we just show the ROAS = 1/COS

we calculate the Roas for each Criteo campaign type

roas = sum(order value) / sum (criteo cost)

In [313]:
```
campaign_types=df2.groupby(['Campaign Type'])['Order Value'].sum()
```

In [314]:
```
campaign_cost=df2.groupby(['Campaign Type'])['Criteo Cost'].sum()
```

In [315]:
```
roas =campaign_types/campaign_cost
```

In [316]:
```
roas
```

Out[316]:
```
Campaign Type
APP INSTALL            2.834492
INAPP                  1.057712
LOWER FUNNEL CUSTOM    0.020868
MID FUNNEL CUSTOM      0.158348
PROSPECTING            0.206185
dtype: float64
```

In [317]:
```
cos= 1/roas
```

In [318]:
```
cos
```

Out[318]:
```
Campaign Type
```

```
APP INSTALL             0.352797
INAPP                   0.945437
LOWER FUNNEL CUSTOM    47.919984
MID FUNNEL CUSTOM       6.315216
PROSPECTING             4.850022
dtype: float64
```

In [ ]:
```
#! pip install seaborn
```

In this notebook i use different type plot bar,barh,box plot ,pie .. but i can't use the library seaborn so i used collab to plot some graphs