COMPETITOR_DIGITIAL_ADVERTISING_INTELLIGENCE_ANALYSI

April 4, 2025

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

0.0.1 DATA EXTRACTION

[ ]:
```
import requests

API_KEY = 'a73e9612bb5cdd0625570da6ce7bea8f5518e98bd47e5f237ac8ea638fb699e5'
query = "zoho.com"
region = '2566'

params = {
    "engine": "google_ads_transparency_center",
    "api_key": API_KEY,
    "text": query,
    "region": region  # Nigeria's code in this example
}

def fetch_all_ads(params):
    url = "https://serpapi.com/search"
    all_ads = []

    while True:
        response = requests.get(url, params=params)
        if response.status_code != 200:
            print("Error:", response.text)
            break

        data = response.json()
```

```
        if "ad_creatives" in data:
            all_ads.extend(data["ad_creatives"])

        # Check if there's a next_page_token for more results
        pagination = data.get("serpapi_pagination", {})
        next_token = pagination.get("next_page_token")
        if next_token:
            params["next_page_token"] = next_token
        else:
            break
    return all_ads

ads_data = fetch_all_ads(params)
print("Total ads fetched:", len(ads_data))
```

```
Total ads fetched: 352
```

[ ]:
```
# Create a DataFrame using only the ad_creatives data
zoho_df = pd.DataFrame(ads_data)

# first 5 rows of the dataframe
zoho_df.head()
```

[ ]:

	advertiser_id	advertiser	ad_creative_id	format	target_domain	image	width	height	total_days_shown	first_shown	last_shown	details_link	link
0	AR07034216898162065409	Zoho Corporation Pvt. Ltd.	CR05256056336893870081	text	zoho.com	https://tpc.googlesyndication.com/archive/simg…	380.0	199.0	46	1739782436	1743673892	https://adstransparency.google.com/advertiser/…	NaN
1	AR07034216898162065409	Zoho Corporation Pvt. Ltd.	CR10240843776322961409	text	zoho.com	https://tpc.googlesyndication.com/archive/simg…	380.0	199.0	536	1697522348	1743673613	https://adstransparency.google.com/advertiser/…	NaN
2	AR07034216898162065409	Zoho Corporation Pvt. Ltd.	CR03269618964468072449	text	zoho.com	https://tpc.googlesyndication.com/archive/simg…	380.0	132.0	1194	1635145200	1743673523		
3	AR07034216898162065409	Zoho Corporation Pvt. Ltd.	CR02979166578395840513	image	zoho.com	https://tpc.googlesyndication.com/archive/simg…	320.0	480.0	320	1716185667	1743673303		
4	AR07034216898162065409	Zoho Corporation Pvt. Ltd.	CR08183638897775869953	image	zoho.com	https://tpc.googlesyndication.com/archive/simg…	300.0	250.0	320	1716185418	1743672150		

```
2  https://adstransparency.google.com/advertiser/…  NaN
3  https://adstransparency.google.com/advertiser/…  NaN
4  https://adstransparency.google.com/advertiser/…  NaN
```

[ ]:
```
zoho_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 352 entries, 0 to 351
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     352 non-null    object
 1   advertiser        352 non-null    object
 2   ad_creative_id    352 non-null    object
 3   format            352 non-null    object
 4   target_domain     352 non-null    object
 5   image             327 non-null    object
 6   width             327 non-null    float64
 7   height            327 non-null    float64
 8   total_days_shown  352 non-null    int64
 9   first_shown       352 non-null    int64
 10  last_shown        352 non-null    int64
 11  details_link      352 non-null    object
 12  link              25 non-null     object
dtypes: float64(2), int64(3), object(8)
memory usage: 35.9+ KB
```

[ ]:
```
cond = (zoho_df.width.isna())
check = zoho_df[cond]
```

[ ]:
```
check
```

[ ]:
```
              advertiser_id                    advertiser  \
8    AR07034216898162065409    Zoho Corporation Pvt. Ltd.
41   AR07034216898162065409    Zoho Corporation Pvt. Ltd.
51   AR07034216898162065409    Zoho Corporation Pvt. Ltd.
55   AR07034216898162065409    Zoho Corporation Pvt. Ltd.
68   AR07034216898162065409    Zoho Corporation Pvt. Ltd.
73   AR07034216898162065409    Zoho Corporation Pvt. Ltd.
181  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
182  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
184  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
185  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
186  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
187  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
188  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
195  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
```

```
196  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
209  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
250  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
261  AR07758168442317832193  Grey Couch Conversations LLC
262  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
270  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
271  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
272  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
273  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
278  AR07034216898162065409    Zoho Corporation Pvt. Ltd.
309  AR07034216898162065409    Zoho Corporation Pvt. Ltd.

              ad_creative_id format target_domain image  width  height  \
8    CR13202282344628617217  video      zoho.com   NaN    NaN     NaN
41   CR15158580213551464449  video      zoho.com   NaN    NaN     NaN
51   CR11415666710741516289  video      zoho.com   NaN    NaN     NaN
55   CR15090753883854602241  video      zoho.com   NaN    NaN     NaN
68   CR16000023425412235265  video      zoho.com   NaN    NaN     NaN
73   CR17736434909053976577  video      zoho.com   NaN    NaN     NaN
181  CR14250198066421301249  video      zoho.com   NaN    NaN     NaN
182  CR14151737624787681281  video      zoho.com   NaN    NaN     NaN
184  CR04743064206109048833  video      zoho.com   NaN    NaN     NaN
185  CR14666869130322247681  video      zoho.com   NaN    NaN     NaN
186  CR08919722264253956097  video      zoho.com   NaN    NaN     NaN
187  CR01838884973562560513  video      zoho.com   NaN    NaN     NaN
188  CR07248834025941368833  video      zoho.com   NaN    NaN     NaN
195  CR03227499250988351489  video      zoho.com   NaN    NaN     NaN
196  CR01138272676830248961  video      zoho.com   NaN    NaN     NaN
209  CR04917321079531241473  video      zoho.com   NaN    NaN     NaN
250  CR09141607782368149505  video      zoho.com   NaN    NaN     NaN
261  CR17770527895593156609  video      zoho.com   NaN    NaN     NaN
262  CR13344450984806449153  video      zoho.com   NaN    NaN     NaN
270  CR15502844811810963457  video      zoho.com   NaN    NaN     NaN
271  CR04755549590139174913  video      zoho.com   NaN    NaN     NaN
272  CR03271182504362508289  video      zoho.com   NaN    NaN     NaN
273  CR17768606705182048257  video      zoho.com   NaN    NaN     NaN
278  CR18430381794500018177  image      zoho.com   NaN    NaN     NaN
309  CR02035235433633284097  image      zoho.com   NaN    NaN     NaN

     total_days_shown  first_shown  last_shown  \
8                  12   1715584622  1743671040
41                 55   1738941430  1743653299
51                 55   1738945763  1743645912
55                 28   1741107610  1743644331
68                 30   1741107894  1743633614
73                 29   1741108124  1743632259
181                30   1738699965  1741199495
```

182	30	1738698681	1741196243
184	56	1734074595	1738823712
185	56	1734074665	1738823625
186	56	1734074820	1738823594
187	56	1734074325	1738823043
188	56	1734074663	1738821967
195	150	1721113200	1734014071
196	150	1721130089	1734014016
209	136	1722257130	1733980828
250	9	1715584622	1724867702
261	4	1723329329	1723589178
262	12	1722264031	1723199873
270	29	1700738855	1719915708
271	223	1700739154	1719915705
272	223	1700737758	1719915665
273	223	1700739577	1719915586
278	76	1710839466	1717413565
309	7	1715666105	1716181737

	details_link	\
8	https://adstransparency.google.com/advertiser/…	
41	https://adstransparency.google.com/advertiser/…	
51	https://adstransparency.google.com/advertiser/…	
55	https://adstransparency.google.com/advertiser/…	
68	https://adstransparency.google.com/advertiser/…	
73	https://adstransparency.google.com/advertiser/…	
181	https://adstransparency.google.com/advertiser/…	
182	https://adstransparency.google.com/advertiser/…	
184	https://adstransparency.google.com/advertiser/…	
185	https://adstransparency.google.com/advertiser/…	
186	https://adstransparency.google.com/advertiser/…	
187	https://adstransparency.google.com/advertiser/…	
188	https://adstransparency.google.com/advertiser/…	
195	https://adstransparency.google.com/advertiser/…	
196	https://adstransparency.google.com/advertiser/…	
209	https://adstransparency.google.com/advertiser/…	
250	https://adstransparency.google.com/advertiser/…	
261	https://adstransparency.google.com/advertiser/…	
262	https://adstransparency.google.com/advertiser/…	
270	https://adstransparency.google.com/advertiser/…	
271	https://adstransparency.google.com/advertiser/…	
272	https://adstransparency.google.com/advertiser/…	
273	https://adstransparency.google.com/advertiser/…	
278	https://adstransparency.google.com/advertiser/…	
309	https://adstransparency.google.com/advertiser/…	

	link

```
8    https://displayads-formats.googleusercontent.c…
41   https://displayads-formats.googleusercontent.c…
51   https://displayads-formats.googleusercontent.c…
55   https://displayads-formats.googleusercontent.c…
68   https://displayads-formats.googleusercontent.c…
73   https://displayads-formats.googleusercontent.c…
181  https://displayads-formats.googleusercontent.c…
182  https://displayads-formats.googleusercontent.c…
184  https://displayads-formats.googleusercontent.c…
185  https://displayads-formats.googleusercontent.c…
186  https://displayads-formats.googleusercontent.c…
187  https://displayads-formats.googleusercontent.c…
188  https://displayads-formats.googleusercontent.c…
195  https://displayads-formats.googleusercontent.c…
196  https://displayads-formats.googleusercontent.c…
209  https://displayads-formats.googleusercontent.c…
250  https://displayads-formats.googleusercontent.c…
261  https://displayads-formats.googleusercontent.c…
262  https://displayads-formats.googleusercontent.c…
270  https://displayads-formats.googleusercontent.c…
271  https://displayads-formats.googleusercontent.c…
272  https://displayads-formats.googleusercontent.c…
273  https://displayads-formats.googleusercontent.c…
278  https://displayads-formats.googleusercontent.c…
309  https://displayads-formats.googleusercontent.c…
```

[ ]:
```
# filling missing values
zoho_df.fillna({"link": "not_available", "image": "not_available", "width": 0,␣
  ↪"height": 0}, inplace=True)
```

[ ]:
```
zoho_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 352 entries, 0 to 351
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     352 non-null    object
 1   advertiser        352 non-null    object
 2   ad_creative_id    352 non-null    object
 3   format            352 non-null    object
 4   target_domain     352 non-null    object
 5   image             352 non-null    object
 6   width             352 non-null    float64
 7   height            352 non-null    float64
 8   total_days_shown  352 non-null    int64
 9   first_shown       352 non-null    int64
```

```
 10  last_shown        352 non-null    int64
 11  details_link      352 non-null    object
 12  link              352 non-null    object
dtypes: float64(2), int64(3), object(8)
memory usage: 35.9+ KB
```

0.0.2 Vend(Lightspeed)

[ ]:
```python
import requests

API_KEY = 'a73e9612bb5cdd0625570da6ce7bea8f5518e98bd47e5f237ac8ea638fb699e5'
query = "lightspeedhq.com"
region = '2566'

params = {
    "engine": "google_ads_transparency_center",
    "api_key": API_KEY,
    "text": query,
    "region": region  # Nigeria's code in this example
}

def fetch_all_ads(params):
    url = "https://serpapi.com/search"
    all_ads = []

    while True:
        response = requests.get(url, params=params)
        if response.status_code != 200:
            print("Error:", response.text)
            break

        data = response.json()
        if "ad_creatives" in data:
            all_ads.extend(data["ad_creatives"])

        # Check if there's a next_page_token for more results
        pagination = data.get("serpapi_pagination", {})
        next_token = pagination.get("next_page_token")
        if next_token:
            params["next_page_token"] = next_token
        else:
            break
    return all_ads

ads_data = fetch_all_ads(params)
print("Total ads fetched:", len(ads_data))
```

```
Total ads fetched: 1
```

[ ]:
```
# Create a DataFrame using only the ad_creatives data
vend_df = pd.DataFrame(ads_data)

# first 5 rows of the dataframe
vend_df.head()
```

[ ]:
```
             advertiser_id                advertiser          ad_creative_id  \
0  AR10603491951300837377  Lightspeed Commerce Inc.  CR06543312581589729281

  format     target_domain                                                image  \
0   text  lightspeedhq.com  https://tpc.googlesyndication.com/archive/simg…

   width  height  total_days_shown  first_shown  last_shown  \
0    380     199               843   1660374000  1743617603

                                       details_link
0  https://adstransparency.google.com/advertiser/…
```

[ ]:
```
vend_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     1 non-null      object
 1   advertiser        1 non-null      object
 2   ad_creative_id    1 non-null      object
 3   format            1 non-null      object
 4   target_domain     1 non-null      object
 5   image             1 non-null      object
 6   width             1 non-null      int64
 7   height            1 non-null      int64
 8   total_days_shown  1 non-null      int64
 9   first_shown       1 non-null      int64
 10  last_shown        1 non-null      int64
 11  details_link      1 non-null      object
dtypes: int64(5), object(7)
memory usage: 228.0+ bytes
```

0.0.3 Square Pos

[ ]:
```
import requests

API_KEY = 'a73e9612bb5cdd0625570da6ce7bea8f5518e98bd47e5f237ac8ea638fb699e5'
query = "squareup.com"
region = '2566'
```

```
params = {
    "engine": "google_ads_transparency_center",
    "api_key": API_KEY,
    "text": query,
    "region": region  # Nigeria's code in this example
}

def fetch_all_ads(params):
    url = "https://serpapi.com/search"
    all_ads = []

    while True:
        response = requests.get(url, params=params)
        if response.status_code != 200:
            print("Error:", response.text)
            break

        data = response.json()
        if "ad_creatives" in data:
            all_ads.extend(data["ad_creatives"])

        # Check if there's a next_page_token for more results
        pagination = data.get("serpapi_pagination", {})
        next_token = pagination.get("next_page_token")
        if next_token:
            params["next_page_token"] = next_token
        else:
            break
    return all_ads

ads_data = fetch_all_ads(params)
print("Total ads fetched:", len(ads_data))
```

```
Total ads fetched: 86
```

[ ]:
```
# Create a DataFrame using only the ad_creatives data
square_df = pd.DataFrame(ads_data)

# first 5 rows of the dataframe
square_df.head()
```

[ ]:
```
            advertiser_id          advertiser          ad_creative_id format  \
0  AR04722834171410513921    MightyHive, INC.  CR06611493727124848641  video
1  AR14896030700992987137              Square  CR07106127312260694017  image
2  AR14896030700992987137              Square  CR17054888556264161281   text
3  AR05698947849719382017  Rocketgenius, Inc.  CR05922805754254852097   text
```

```
4  AR14896030700992987137              Square  CR07355796578675720193  image

  target_domain                                                image  width  \
0  squareup.com  https://tpc.googlesyndication.com/archive/simg…  648.0
1  squareup.com  https://s0.2mdn.net/9598779/RST_USEN_Programma…  300.0
2  squareup.com  https://tpc.googlesyndication.com/archive/simg…  380.0
3  squareup.com  https://tpc.googlesyndication.com/archive/simg…  380.0
4  squareup.com                                                  NaN    NaN

   height  total_days_shown  first_shown  last_shown  \
0   478.0               207   1725895800  1743663600
1   250.0                92   1735858800  1743663600
2   199.0               546   1696563220  1743659906
3   239.0                17   1742221830  1743654920
4     NaN               204   1725946200  1743647400

                                       details_link  \
0  https://adstransparency.google.com/advertiser/…
1  https://adstransparency.google.com/advertiser/…
2  https://adstransparency.google.com/advertiser/…
3  https://adstransparency.google.com/advertiser/…
4  https://adstransparency.google.com/advertiser/…

                                                 link
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                                 NaN
4  https://displayads-formats.googleusercontent.c…
```

[ ]:
```
square_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86 entries, 0 to 85
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     86 non-null     object
 1   advertiser        86 non-null     object
 2   ad_creative_id    86 non-null     object
 3   format            86 non-null     object
 4   target_domain     86 non-null     object
 5   image             14 non-null     object
 6   width             14 non-null     float64
 7   height            14 non-null     float64
 8   total_days_shown  86 non-null     int64
 9   first_shown       86 non-null     int64
```

```
 10  last_shown        86 non-null     int64
 11  details_link      86 non-null     object
 12  link              72 non-null     object
dtypes: float64(2), int64(3), object(8)
memory usage: 8.9+ KB
```

[ ]:
```
cond = (square_df.link.isna())
check = square_df[cond]
```

[ ]:
```
check
```

[ ]:

	advertiser_id	advertiser	ad_creative_id	\
0	AR04722834171410513921	MightyHive, INC.	CR06611493727124848641	
1	AR14896030700992987137	Square	CR07106127312260694017	
2	AR14896030700992987137	Square	CR17054888556264161281	
3	AR05698947849719382017	Rocketgenius, Inc.	CR05922805754254852097	
12	AR14896030700992987137	Square	CR13803824383090753537	
13	AR14896030700992987137	Square	CR13820994356670103553	
20	AR13692503641141805057	Mr Carpet Cleaner Ltd	CR13973191642404880385	
68	AR14896030700992987137	Square	CR02645102682083164161	
80	AR14896030700992987137	Square	CR10191278891654447105	
81	AR13353888419525165057	Brad A Paige	CR13382379583299387393	
82	AR16951853287265533953	Brad Allen Paige	CR04455288220317384705	
83	AR16951853287265533953	Brad Allen Paige	CR00861195506112528385	
84	AR16951853287265533953	Brad Allen Paige	CR06172561899268341761	
85	AR16951853287265533953	Brad Allen Paige	CR17913500278770892801	

	format	target_domain	image	\
0	video	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
1	image	squareup.com	https://s0.2mdn.net/9598779/RST_USEN_Programma…	
2	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
3	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
12	image	squareup.com	https://s0.2mdn.net/9598779/RST_USEN_Programma…	
13	image	squareup.com	https://s0.2mdn.net/9598779/RST_USEN_Programma…	
20	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
68	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
80	image	squareup.com	https://s0.2mdn.net//8701942/87da3f9c-9b8e-4fb…	
81	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
82	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
83	text	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
84	image	squareup.com	https://tpc.googlesyndication.com/archive/simg…	
85	image	squareup.com	https://tpc.googlesyndication.com/archive/simg…	

	width	height	total_days_shown	first_shown	last_shown	\
0	648.0	478.0	207	1725895800	1743663600	
1	300.0	250.0	92	1735858800	1743663600	
2	380.0	199.0	546	1696563220	1743659906	

```
3   380.0   239.0                17   1742221830  1743654920
12  300.0   250.0                91   1735858800  1743593400
13  300.0   250.0                91   1735858800  1743589800
20  380.0   147.0                67   1733089418  1739090826
68  380.0   167.0               812   1654905026  1727782795
80  333.0   333.0               238   1699309800  1719813600
81  380.0   147.0                 5   1713279244  1713652615
82  380.0   147.0                75   1689974767  1713221336
83  380.0   153.0               281   1635145200  1713055440
84  300.0   600.0                 3   1712263031  1712430972
85  300.0   600.0                 5   1711053193  1712263771

                                          details_link link
0   https://adstransparency.google.com/advertiser/…  NaN
1   https://adstransparency.google.com/advertiser/…  NaN
2   https://adstransparency.google.com/advertiser/…  NaN
3   https://adstransparency.google.com/advertiser/…  NaN
12  https://adstransparency.google.com/advertiser/…  NaN
13  https://adstransparency.google.com/advertiser/…  NaN
20  https://adstransparency.google.com/advertiser/…  NaN
68  https://adstransparency.google.com/advertiser/…  NaN
80  https://adstransparency.google.com/advertiser/…  NaN
81  https://adstransparency.google.com/advertiser/…  NaN
82  https://adstransparency.google.com/advertiser/…  NaN
83  https://adstransparency.google.com/advertiser/…  NaN
84  https://adstransparency.google.com/advertiser/…  NaN
85  https://adstransparency.google.com/advertiser/…  NaN
```

[ ]:
```
# filling missing values
square_df.fillna({"link": "not_available", "image": "not_available", "width":␣
↪0, "height": 0}, inplace=True)
```

[ ]:
```
square_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86 entries, 0 to 85
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     86 non-null     object
 1   advertiser        86 non-null     object
 2   ad_creative_id    86 non-null     object
 3   format            86 non-null     object
 4   target_domain     86 non-null     object
 5   image             86 non-null     object
 6   width             86 non-null     float64
 7   height            86 non-null     float64
```

```
 8   total_days_shown  86 non-null     int64
 9   first_shown       86 non-null     int64
 10  last_shown        86 non-null     int64
 11  details_link      86 non-null     object
 12  link              86 non-null     object
dtypes: float64(2), int64(3), object(8)
memory usage: 8.9+ KB
```

[ ]:

0.0.4 INFLOW

[ ]:
```python
import requests

API_KEY = 'a73e9612bb5cdd0625570da6ce7bea8f5518e98bd47e5f237ac8ea638fb699e5'
query = "inflowinventory.com"
region = '2566'

params = {
    "engine": "google_ads_transparency_center",
    "api_key": API_KEY,
    "text": query,
    "region": region  # Nigeria's code in this example
}

def fetch_all_ads(params):
    url = "https://serpapi.com/search"
    all_ads = []

    while True:
        response = requests.get(url, params=params)
        if response.status_code != 200:
            print("Error:", response.text)
            break

        data = response.json()
        if "ad_creatives" in data:
            all_ads.extend(data["ad_creatives"])

        # Check if there's a next_page_token for more results
        pagination = data.get("serpapi_pagination", {})
        next_token = pagination.get("next_page_token")
        if next_token:
            params["next_page_token"] = next_token
        else:
            break
    return all_ads
```

```
ads_data = fetch_all_ads(params)
print("Total ads fetched:", len(ads_data))
```

```
Total ads fetched: 2
```

[ ]:
```
# Create a DataFrame using only the ad_creatives data
inflow_df = pd.DataFrame(ads_data)

# first 5 rows of the dataframe
inflow_df.head()
```

[ ]:
```
            advertiser_id          advertiser          ad_creative_id format  \
0  AR00936138356100694017  Archon Systems Inc  CR15129661666171551745   text
1  AR00188580400375791617  Archon Systems Inc  CR10089892340341669889  image

         target_domain                                                image  \
0  inflowinventory.com  https://tpc.googlesyndication.com/archive/simg…
1  inflowinventory.com  https://tpc.googlesyndication.com/archive/simg…

   width  height  total_days_shown  first_shown  last_shown  \
0    380     239               856   1660460400  1743620350
1    300     600               461   1660374000  1738593541

                                      details_link
0  https://adstransparency.google.com/advertiser/…
1  https://adstransparency.google.com/advertiser/…
```

[ ]:
```
inflow_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     2 non-null      object
 1   advertiser        2 non-null      object
 2   ad_creative_id    2 non-null      object
 3   format            2 non-null      object
 4   target_domain     2 non-null      object
 5   image             2 non-null      object
 6   width             2 non-null      int64
 7   height            2 non-null      int64
 8   total_days_shown  2 non-null      int64
 9   first_shown       2 non-null      int64
 10  last_shown        2 non-null      int64
 11  details_link      2 non-null      object
dtypes: int64(5), object(7)
```

```
memory usage: 324.0+ bytes
```

[ ]:

0.0.5 odoo

[ ]:
```python
import requests

API_KEY = 'a73e9612bb5cdd0625570da6ce7bea8f5518e98bd47e5f237ac8ea638fb699e5'
query = "odoo.com"
region = '2566'

params = {
    "engine": "google_ads_transparency_center",
    "api_key": API_KEY,
    "text": query,
    "region": region  # Nigeria's code in this example
}

def fetch_all_ads(params):
    url = "https://serpapi.com/search"
    all_ads = []

    while True:
        response = requests.get(url, params=params)
        if response.status_code != 200:
            print("Error:", response.text)
            break

        data = response.json()
        if "ad_creatives" in data:
            all_ads.extend(data["ad_creatives"])

        # Check if there's a next_page_token for more results
        pagination = data.get("serpapi_pagination", {})
        next_token = pagination.get("next_page_token")
        if next_token:
            params["next_page_token"] = next_token
        else:
            break
    return all_ads

ads_data = fetch_all_ads(params)
print("Total ads fetched:", len(ads_data))
```

```
Total ads fetched: 145
```

[ ]:
```
# Create a DataFrame using only the ad_creatives data
odoo_df = pd.DataFrame(ads_data)

# first 5 rows of the dataframe
odoo_df.head()
```

[ ]:

	advertiser_id	advertiser	ad_creative_id	format	target_domain	image	width	height	total_days_shown	first_shown	last_shown	details_link	link
0	AR13734029429363965953	OpenERP SA	CR04640816221795647489	text	odoo.com	https://tpc.googlesyndication.com/archive/simg…	380.0	173.0	119	1733415312	1743674085	https://adstransparency.google.com/advertiser/…	NaN
1	AR13734029429363965953	OpenERP SA	CR01233947969611366401	text	odoo.com	https://tpc.googlesyndication.com/archive/simg…	380.0	265.0	119	1733415465	1743673956	https://adstransparency.google.com/advertiser/…	NaN
2	AR13734029429363965953	OpenERP SA	CR01402260230338772993	text	odoo.com	https://tpc.googlesyndication.com/archive/simg…	380.0	173.0	1256	1635145200	1743673929	https://adstransparency.google.com/advertiser/…	NaN
3	AR13353130495236374529	e-atlete bvba	CR04728542286026113025	image	odoo.com	https://tpc.googlesyndication.com/archive/simg…	300.0	600.0	478	1697791711	1743673228	https://adstransparency.google.com/advertiser/…	NaN
4	AR13734029429363965953	OpenERP SA	CR17412179670073344001	text	odoo.com	https://tpc.googlesyndication.com/archive/simg…	380.0	199.0	500	1700517813	1743672408	https://adstransparency.google.com/advertiser/…	NaN

[ ]:
```
odoo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     145 non-null    object
 1   advertiser        145 non-null    object
 2   ad_creative_id    145 non-null    object
 3   format            145 non-null    object
 4   target_domain     145 non-null    object
 5   image             128 non-null    object
```

```
 6   width             128 non-null    float64
 7   height            128 non-null    float64
 8   total_days_shown  145 non-null    int64
 9   first_shown       145 non-null    int64
 10  last_shown        145 non-null    int64
 11  details_link      145 non-null    object
 12  link              17 non-null     object
dtypes: float64(2), int64(3), object(8)
memory usage: 14.9+ KB
```

[ ]:
```
# filling missing values
odoo_df.fillna({"link": "not_available", "image": "not_available", "width": 0,␣
 ↪"height": 0}, inplace=True)
```

[ ]:
```
odoo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     145 non-null    object
 1   advertiser        145 non-null    object
 2   ad_creative_id    145 non-null    object
 3   format            145 non-null    object
 4   target_domain     145 non-null    object
 5   image             145 non-null    object
 6   width             145 non-null    float64
 7   height            145 non-null    float64
 8   total_days_shown  145 non-null    int64
 9   first_shown       145 non-null    int64
 10  last_shown        145 non-null    int64
 11  details_link      145 non-null    object
 12  link              145 non-null    object
dtypes: float64(2), int64(3), object(8)
memory usage: 14.9+ KB
```

0.0.6 saving to csv files

[ ]:
```
odoo_df.to_csv("odoo.csv")
inflow_df.to_csv("inflow.csv")
square_df.to_csv("square.csv")
vend_df.to_csv("vend.csv")
zoho_df.to_csv("zoho.csv")
```

0.0.7 INITIAL DATA EXPLORATION

[ ]:
```
import pandas as pd

# Assuming these DataFrames already exist
dfs = [odoo_df, inflow_df, square_df, vend_df, zoho_df]

# Concatenating all DataFrames
combined_df = pd.concat(dfs, ignore_index=True)

# Save to CSV if needed
combined_df.to_csv("combined_data.csv", index=False)

# Display first few rows to confirm
combined_df.head()
```

[ ]:
```
            advertiser_id     advertiser          ad_creative_id format  \
0  AR13734029429363965953     OpenERP SA  CR04640816221795647489   text
1  AR13734029429363965953     OpenERP SA  CR01233947969611366401   text
2  AR13734029429363965953     OpenERP SA  CR01402260230338772993   text
3  AR13353130495236374529  e-atlete bvba  CR04728542286026113025  image
4  AR13734029429363965953     OpenERP SA  CR17412179670073344001   text

  target_domain                                              image  width  \
0      odoo.com  https://tpc.googlesyndication.com/archive/simg…  380.0
1      odoo.com  https://tpc.googlesyndication.com/archive/simg…  380.0
2      odoo.com  https://tpc.googlesyndication.com/archive/simg…  380.0
3      odoo.com  https://tpc.googlesyndication.com/archive/simg…  300.0
4      odoo.com  https://tpc.googlesyndication.com/archive/simg…  380.0

   height  total_days_shown  first_shown  last_shown  \
0   173.0               119   1733415312  1743674085
1   265.0               119   1733415465  1743673956
2   173.0              1256   1635145200  1743673929
3   600.0               478   1697791711  1743673228
4   199.0               500   1700517813  1743672408

                                        details_link           link
0  https://adstransparency.google.com/advertiser/…  not_available
1  https://adstransparency.google.com/advertiser/…  not_available
2  https://adstransparency.google.com/advertiser/…  not_available
3  https://adstransparency.google.com/advertiser/…  not_available
4  https://adstransparency.google.com/advertiser/…  not_available
```

[ ]:
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586 entries, 0 to 585
```

```
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     586 non-null    object
 1   advertiser        586 non-null    object
 2   ad_creative_id    586 non-null    object
 3   format            586 non-null    object
 4   target_domain     586 non-null    object
 5   image             586 non-null    object
 6   width             586 non-null    float64
 7   height            586 non-null    float64
 8   total_days_shown  586 non-null    int64
 9   first_shown       586 non-null    int64
 10  last_shown        586 non-null    int64
 11  details_link      586 non-null    object
 12  link              583 non-null    object
dtypes: float64(2), int64(3), object(8)
memory usage: 59.6+ KB
```

[ ]:
```
# filling missing values
combined_df.fillna({"link": "not_available"}, inplace=True)
```

[ ]:
```
# Convert Unix epoch to datetime for first_shown and last_shown
combined_df['first_shown'] = pd.to_datetime(combined_df['first_shown'],␣
 ↪unit='s')
combined_df['last_shown'] = pd.to_datetime(combined_df['last_shown'], unit='s')
```

[ ]:
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586 entries, 0 to 585
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   advertiser_id     586 non-null    object
 1   advertiser        586 non-null    object
 2   ad_creative_id    586 non-null    object
 3   format            586 non-null    object
 4   target_domain     586 non-null    object
 5   image             586 non-null    object
 6   width             586 non-null    float64
 7   height            586 non-null    float64
 8   total_days_shown  586 non-null    int64
 9   first_shown       586 non-null    datetime64[ns]
 10  last_shown        586 non-null    datetime64[ns]
 11  details_link      586 non-null    object
 12  link              586 non-null    object
dtypes: datetime64[ns](2), float64(2), int64(1), object(8)
```

```
memory usage: 59.6+ KB
```

[ ]:
```
combined_df.describe()
```

[ ]:
```
              width       height  total_days_shown  \
count   586.000000   586.000000        586.000000
mean    352.216724   222.354949        411.061433
min       0.000000     0.000000          3.000000
25%     300.000000    92.500000         77.000000
50%     380.000000   173.000000        223.000000
75%     380.000000   250.000000        711.750000
max    1940.000000  1200.000000       1256.000000
std     280.394769   201.881374        409.830792

                          first_shown                     last_shown
count                             586                            586
mean   2023-10-10 17:53:40.645051136  2024-12-26 03:22:40.752559616
min              2021-10-25 07:00:00            2024-04-04 20:49:31
25%    2022-10-15 07:02:02.750000128            2024-10-04 00:00:00
50%       2024-03-19 09:10:04.500000            2025-03-27 13:05:47
75%    2024-06-03 08:57:57.750000128     2025-04-02 18:43:26.500000
max              2025-03-17 14:30:30            2025-04-03 09:54:45
std                              NaN                            NaN
```

[ ]:
```
combined_df.shape
```

[ ]:
```
(586, 13)
```

0.0.8 target_domain

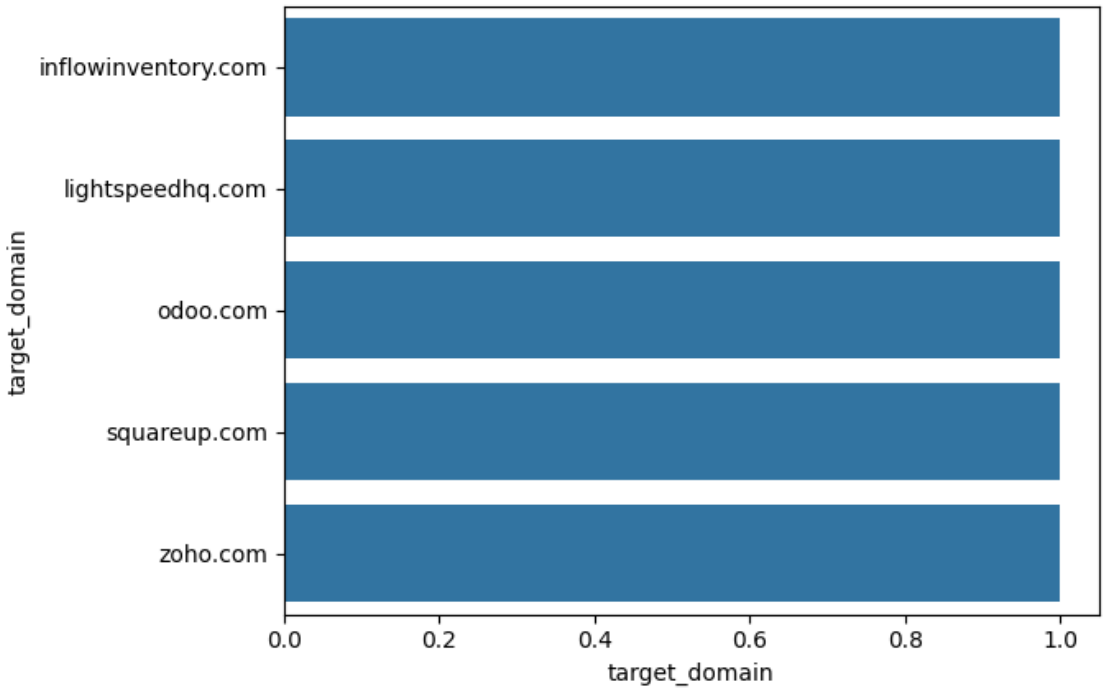
[ ]:
```
domain_counts = combined_df.groupby('target_domain')['target_domain'].nunique()
```

[ ]:
```
domain_counts
```

[ ]:
```
target_domain
inflowinventory.com    1
lightspeedhq.com       1
odoo.com               1
squareup.com           1
zoho.com               1
Name: target_domain, dtype: int64
```

[ ]:
```
sns.barplot(domain_counts, orient="h")
```

[ ]:
```
<Axes: xlabel='target_domain', ylabel='target_domain'>
```



0.0.9 AD CREATIVES

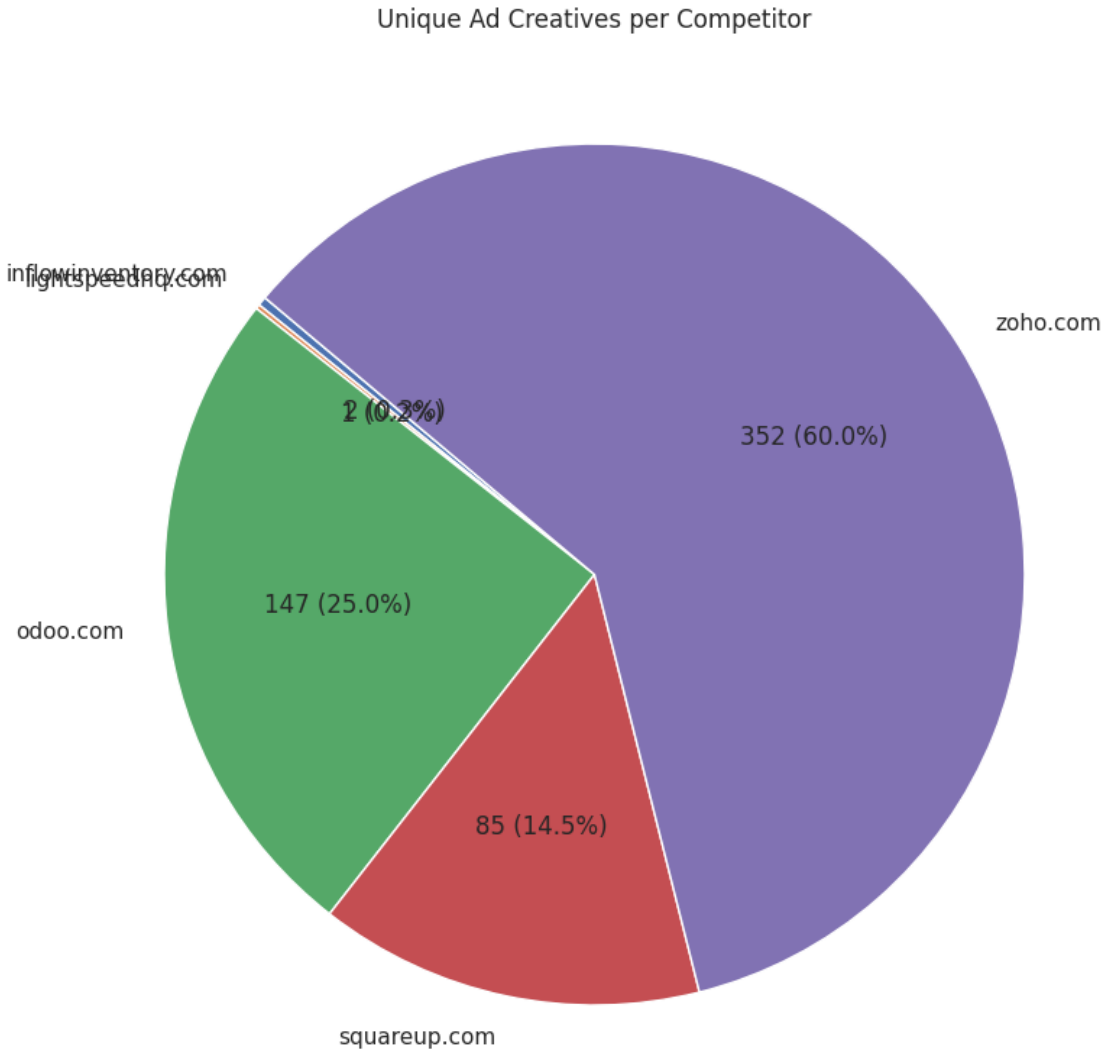
[ ]:
```
# Unique ad creatives count by competitor
creative_counts = combined_df.groupby('target_domain')['ad_creative_id'].
 ↪nunique().reset_index()
creative_counts.rename(columns={'ad_creative_id': 'unique_creatives'},␣
 ↪inplace=True)
print("\nUnique Ad Creatives per Competitor:")
print(creative_counts)
```

```
Unique Ad Creatives per Competitor:
         target_domain  unique_creatives
0  inflowinventory.com                 2
1     lightspeedhq.com                 1
2             odoo.com               145
3          squareup.com                86
4             zoho.com               352
```

[ ]:
```
```

[ ]:
```
import matplotlib.pyplot as plt

# Data from the unique creatives count
```

```
labels = creative_counts['target_domain']
sizes = creative_counts['unique_creatives']

# Create the pie chart
fig, ax = plt.subplots(figsize=(8, 8))

# Function to format the autopct with both percentage and exact figure
def func(pct, allsizes):
    absolute = int(pct/100.*sum(allsizes))  # calculate absolute number
    return f"{absolute} ({pct:.1f}%)"

# Create pie chart with proper autopct formatting
ax.pie(sizes, labels=labels, autopct=lambda pct: func(pct, sizes),␣
 ↪startangle=140)

# Equal aspect ratio ensures that pie is drawn as a circle
ax.axis('equal')

# Title for the pie chart
plt.title("Unique Ad Creatives per Competitor")

# Show the plot
plt.tight_layout()
plt.show()
```



0.0.10 TOTAL EXPOSURE

[ ]:
```
# Create an empty list to collect distribution results for each competitor␣
↪(target_domain)
distribution_results = []

# Get list of unique competitors based on target_domain
competitors = combined_df['target_domain'].unique()

# Loop through each competitor for individual analysis and store aggregated␣
↪distribution metrics
for comp in competitors:
```

```
    df_comp = combined_df[combined_df['target_domain'] == comp]

    # Aggregate distribution metrics for both width and height by ad format
    dist_data = df_comp.groupby('format').agg(
        count=('total_days_shown', 'count'),
        days_shown_avg=('total_days_shown',lambda x: x[x != 0].mean() if not␣
 ↪x[x != 0].empty else np.nan),
        days_shown_min=('total_days_shown', lambda x: x[x != 0].min() if not␣
 ↪x[x != 0].empty else np.nan),
        days_shown_q1=('total_days_shown', lambda x: x[x != 0].quantile(0.25)␣
 ↪if not x[x != 0].empty else np.nan),
        days_shown_median=('total_days_shown', lambda x: x[x != 0].median() if␣
 ↪not x[x != 0].empty else np.nan),
        days_shown_q3=('total_days_shown', lambda x:  x[x != 0].quantile(0.75)␣
 ↪if not x[x != 0].empty else np.nan),
        days_shown_max=('total_days_shown', lambda x: x[x != 0].max() if not␣
 ↪x[x != 0].empty else np.nan),
        days_shown_total=('total_days_shown', lambda x: x[x != 0].sum() if not␣
 ↪x[x != 0].empty else np.nan)
    ).reset_index()

    # Add the target_domain column to the aggregated data
    dist_data['target_domain'] = comp

    # Append the results for this competitor to the list
    distribution_results.append(dist_data)

# Concatenate all competitors' distribution data into a single DataFrame
distribution_days_df = pd.concat(distribution_results, ignore_index=True)

# Reorder columns (if needed)
distribution_days_df = distribution_days_df[['target_domain', 'format',␣
 ↪'count', 'days_shown_avg', 'days_shown_total',
                                    'days_shown_min', 'days_shown_q1',␣
 ↪'days_shown_median', 'days_shown_q3', 'days_shown_max'
                                    ]]

# Display the resulting DataFrame
distribution_days_df.head()

# (Optional) Save the aggregated distribution metrics to CSV
distribution_days_df.to_csv("ad_creatives_distribution_by_format.csv",␣
 ↪index=False)
```

[ ]:
```
distribution_days_df
```

[ ]:

	target_domain	format	count	days_shown_avg	days_shown_total	days_shown_min	days_shown_q1	days_shown_median	days_shown_q3	days_shown_max
0	odoo.com	image	9	138.888889	1250	4	15.0	19.0	165.0	519
1	odoo.com	text	137	811.080292	111118	22	457.0	953.0	1243.0	1255
2	odoo.com	video	1	26.000000	26	26	26.0	26.0	26.0	26
3	inflowinventory.com	image	1	461.000000	461	461	461.0	461.0	461.0	461
4	inflowinventory.com	text	1	855.000000	855	855	855.0	855.0	855.0	855
5	squareup.com	image	77	195.987013	15091	3	151.0	203.0	241.0	451
6	squareup.com	text	7	257.285714	1801	5	41.5	75.0	413.0	812
7	squareup.com	video	1	205.000000	205	205	205.0	205.0	205.0	205
8	lightspeedhq.com	text	1	843.000000	843	843	843.0	843.0	843.0	843
9	zoho.com	image	190	136.200000	25878	7	12.0	100.0	192.0	645
10	zoho.com	text	137	605.335766	82931	45	389.0	525.0	900.0	1193
11	zoho.com	video	25	71.040000	1776	4	29.0	54.0	56.0	223

[ ]:
```
df_pivot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 6 columns):
```

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   target_domain   84 non-null     object
 1   format          84 non-null     object
 2   count           84 non-null     int64
 3   statistic       84 non-null     object
 4   value           84 non-null     float64
 5   metric          84 non-null     object
dtypes: float64(1), int64(1), object(4)
memory usage: 4.1+ KB
```

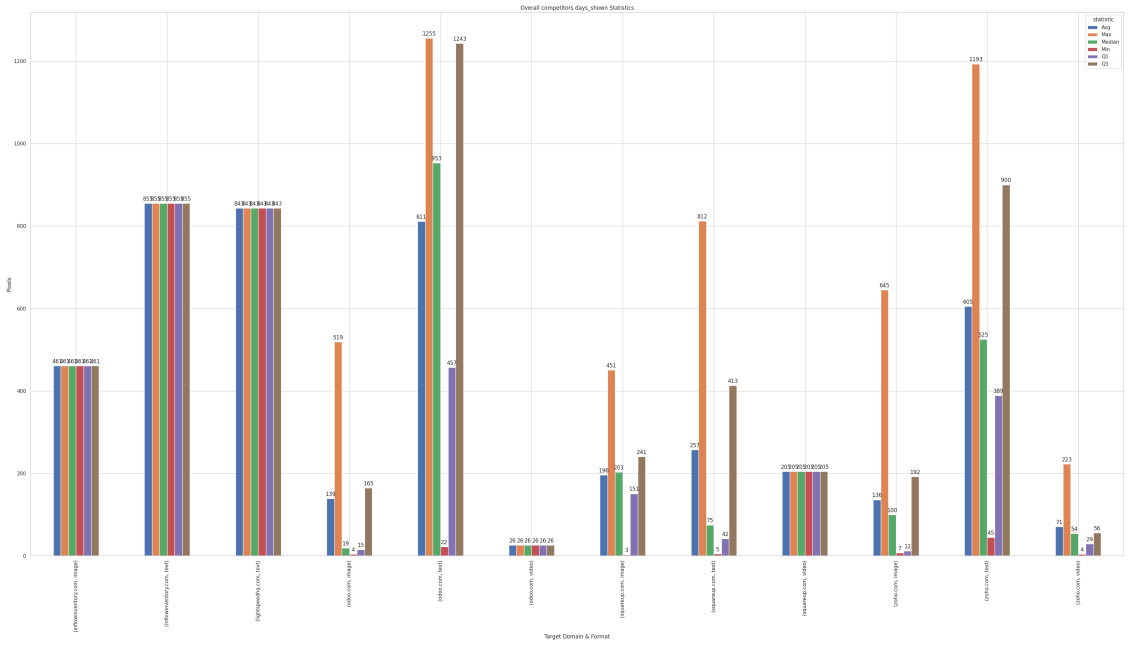
[ ]:
```
df_pivot.head()
```

[ ]:

	target_domain	format	count	statistic	value	metric
0	odoo.com	image	9	Avg	138.888889	Height
1	odoo.com	text	137	Avg	811.080292	Height
2	odoo.com	video	1	Avg	26.000000	Height
3	inflowinventory.com	image	1	Avg	461.000000	Height
4	inflowinventory.com	text	1	Avg	855.000000	Height

[ ]:
```
# Pivoting the DataFrame
df_pivot = distribution_days_df.melt(id_vars=["target_domain", "format",␣
 ↪"count"],
                     value_vars=['days_shown_avg', 'days_shown_min',␣
 ↪'days_shown_q1', 'days_shown_median', 'days_shown_q3', 'days_shown_max'],
                     var_name="statistic", value_name="value")

# Extracting metric type (width or height)
df_pivot["metric"] = "days_shown"
df_pivot["statistic"] = df_pivot["statistic"].apply(lambda x: x.split("_")[-1].
 ↪capitalize())

# Plotting
fig, axes = plt.subplots(1, 1, figsize=(35, 20), sharey=False)
sns.set(style="whitegrid")

for i, metric in enumerate(["days_shown"]):
    pivot_subset = df_pivot[df_pivot["metric"] == metric]
    ax = axes

    plot_data = pivot_subset.pivot(index=["target_domain", "format"],␣
 ↪columns="statistic", values="value")
    plot_data.plot(kind="bar", ax=ax)
    ax.set_title(f"Overall competitors {metric} Statistics")
    ax.set_xlabel("Target Domain & Format")
    ax.set_ylabel("Pixels")
```

```
    # Annotate bars with values
    for container in ax.containers:
        ax.bar_label(container, fmt='%.0f', padding=3)

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[ ]:
```
# Group the distribution_df by 'format', but ignore zero values in the␣
 ↪aggregation
grouped_summary = combined_df.groupby('format').agg(
    count=('format', 'count'),
    overall_days_shown_mean = ('total_days_shown', lambda x: x[x != 0].mean()␣
 ↪if not x[x != 0].empty else np.nan),
    overall_days_shown_min = ('total_days_shown', lambda x: x[x != 0].min() if␣
 ↪not x[x != 0].empty else np.nan),
    overall_days_shown_median = ('total_days_shown', lambda x: x[x != 0].
 ↪median() if not x[x != 0].empty else np.nan),
    overall_days_shown_max = ('total_days_shown', lambda x: x[x != 0].max() if␣
 ↪not x[x != 0].empty else np.nan),
    overall_days_shown_total = ('total_days_shown', lambda x: x[x != 0].sum()␣
 ↪if not x[x != 0].empty else np.nan)
).reset_index()


# (Optional) Save the summary to a CSV file
```

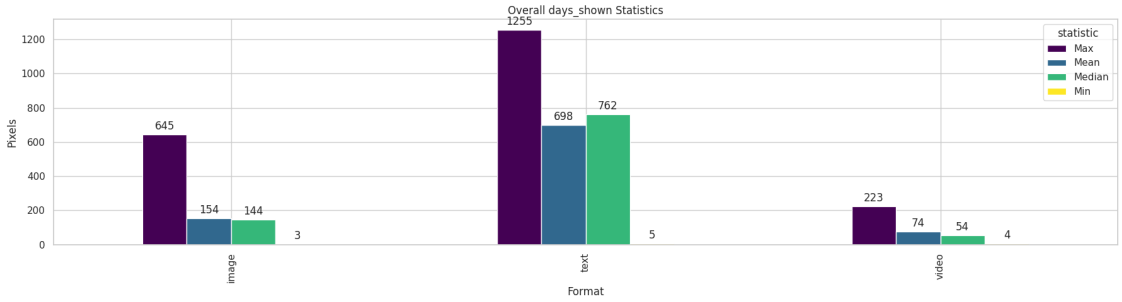
```
grouped_summary.to_csv("grouped_ad_dimension_summary_nonzero_3.csv",␣
 ↪index=False)
```

[ ]:
```
grouped_summary
```

[ ]:
```
   format  count  overall_days_shown_mean  overall_days_shown_min  \
0   image    277                154.079422                       3
1    text    283                698.049470                       5
2   video     27                 74.333333                       4

   overall_days_shown_median  overall_days_shown_max  overall_days_shown_total
0                      144.0                     645                     42680
1                      762.0                    1255                    197548
2                       54.0                     223                      2007
```

[ ]:
```
# @title Pivoting the DataFrame Visualization
df_pivot = grouped_summary.melt(id_vars=["format", "count"],
                    value_vars=["overall_days_shown_min",␣
 ↪"overall_days_shown_mean", "overall_days_shown_median",␣
 ↪"overall_days_shown_max",
                                ],
                    var_name="statistic", value_name="value")

# Extracting metric type (width or height)
df_pivot["metric"] = "days_shown"
df_pivot["statistic"] = df_pivot["statistic"].apply(lambda x: x.split("_")[-1].
 ↪capitalize())

# Plotting
fig, axes = plt.subplots(1, 1, figsize=(18, 5), sharey=False)
sns.set(style="whitegrid")

for i, metric in enumerate(["days_shown"]):
    pivot_subset = df_pivot[df_pivot["metric"] == metric]
    ax = axes

    plot_data = pivot_subset.pivot(index="format", columns="statistic",␣
 ↪values="value")
    plot_data.plot(kind="bar", ax=ax, cmap="viridis")
    ax.set_title(f"Overall {metric} Statistics")
    ax.set_xlabel("Format")
    ax.set_ylabel("Pixels")

    # Annotate bars with values
    for container in ax.containers:
        ax.bar_label(container, fmt='%.0f', padding=3)
```

```
plt.tight_layout()
plt.show()
```



0.0.11 FORMAT; WIDTH; HEIGHT

[ ]:
```
# Create an empty list to collect distribution results for each competitor
 ↪(target_domain)
distribution_results = []

# Get list of unique competitors based on target_domain
competitors = combined_df['target_domain'].unique()

# Loop through each competitor for individual analysis and store aggregated
 ↪distribution metrics
for comp in competitors:
    df_comp = combined_df[combined_df['target_domain'] == comp]

    # Aggregate distribution metrics for both width and height by ad format
    dist_data = df_comp.groupby('format').agg(
        count=('width', 'count'),
        avg_width=('width',lambda x: x[x != 0].mean() if not x[x != 0].empty
 ↪else np.nan),
        width_min=('width', lambda x: x[x != 0].min() if not x[x != 0].empty
 ↪else np.nan),
        width_q1=('width', lambda x: x[x != 0].quantile(0.25) if not x[x != 0].
 ↪empty else np.nan),
        width_median=('width', lambda x: x[x != 0].median() if not x[x != 0].
 ↪empty else np.nan),
        width_q3=('width', lambda x:  x[x != 0].quantile(0.75) if not x[x != 0].
 ↪empty else np.nan),
        width_max=('width', lambda x: x[x != 0].max() if not x[x != 0].empty
 ↪else np.nan),
        avg_height=('height',lambda x: x[x != 0].mean() if not x[x != 0].empty
 ↪else np.nan),
```

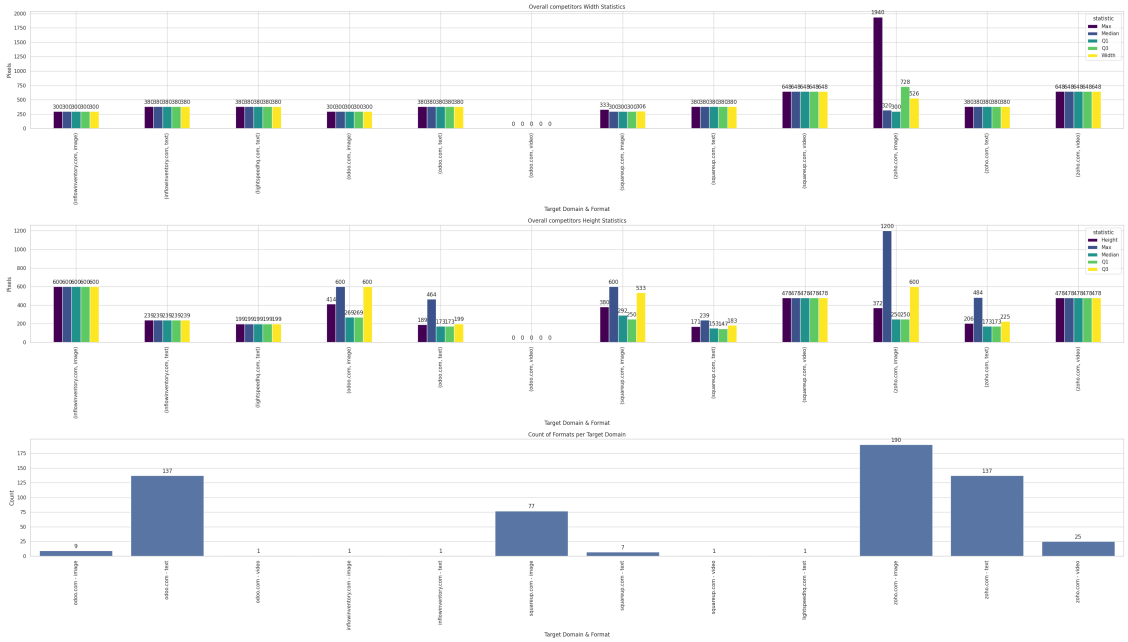
```
        height_min=('height', lambda x: x[x != 0].min() if not x[x != 0].empty␣
 ↪else np.nan),
        height_q1=('height', lambda x:  x[x != 0].quantile(0.25) if not x[x !=␣
 ↪0].empty else np.nan),
        height_median=('height',lambda x: x[x != 0].median() if not x[x != 0].
 ↪empty else np.nan),
        height_q3=('height', lambda x:  x[x != 0].quantile(0.75) if not x[x !=␣
 ↪0].empty else np.nan),
        height_max=('height', lambda x: x[x != 0].max() if not x[x != 0].empty␣
 ↪else np.nan)
    ).reset_index()

    # Add the target_domain column to the aggregated data
    dist_data['target_domain'] = comp

    # Append the results for this competitor to the list
    distribution_results.append(dist_data)

# Concatenate all competitors' distribution data into a single DataFrame
distribution_df = pd.concat(distribution_results, ignore_index=True)

# Reorder columns (if needed)
distribution_df = distribution_df[['target_domain', 'format', 'count',␣
 ↪'avg_width', 'avg_height',
                                   'width_min', 'width_q1', 'width_median',␣
 ↪'width_q3', 'width_max',
                                   'height_min', 'height_q1', 'height_median',␣
 ↪'height_q3', 'height_max']]

# Display the resulting DataFrame
print(distribution_df.head())

# (Optional) Save the aggregated distribution metrics to CSV
distribution_df.to_csv("ad_dimension_distribution_by_format.csv", index=False)
```

```
         target_domain format  count  avg_width  avg_height  width_min  \
0              odoo.com  image      9      300.0   414.00000      300.0
1              odoo.com   text    137      380.0   189.22314      380.0
2              odoo.com  video      1        NaN         NaN        NaN
3  inflowinventory.com  image      1      300.0   600.00000      300.0
4  inflowinventory.com   text      1      380.0   239.00000      380.0

   width_q1  width_median  width_q3  width_max  height_min  height_q1  \
0     300.0         300.0     300.0      300.0       250.0      269.0
1     380.0         380.0     380.0      380.0       147.0      173.0
2       NaN           NaN       NaN        NaN         NaN        NaN
3     300.0         300.0     300.0      300.0       600.0      600.0
```

```
4     380.0         380.0     380.0       380.0        239.0       239.0

   height_median  height_q3  height_max
0          269.0      600.0       600.0
1          173.0      199.0       464.0
2            NaN        NaN         NaN
3          600.0      600.0       600.0
4          239.0      239.0       239.0
```

[ ]:
```
# Pivoting the DataFrame
df_pivot = distribution_df.melt(id_vars=["target_domain", "format", "count"],
                    value_vars=["avg_width", "width_q1", "width_median",␣
 ↪"width_q3", "width_max",
                                "avg_height", "height_q1", "height_median",␣
 ↪"height_q3", "height_max"],
                    var_name="statistic", value_name="value")

# Extracting metric type (width or height)
df_pivot["metric"] = df_pivot["statistic"].apply(lambda x: "Width" if "width"␣
 ↪in x else "Height")
df_pivot["statistic"] = df_pivot["statistic"].apply(lambda x: x.split("_")[-1].
 ↪capitalize())

# Plotting
fig, axes = plt.subplots(3, 1, figsize=(35, 20), sharey=False)
sns.set(style="whitegrid")

for i, metric in enumerate(["Width", "Height"]):
    pivot_subset = df_pivot[df_pivot["metric"] == metric]
    ax = axes[i]

    plot_data = pivot_subset.pivot(index=["target_domain", "format"],␣
 ↪columns="statistic", values="value")
    plot_data.plot(kind="bar", ax=ax, cmap="viridis")
    ax.set_title(f"Overall competitors {metric} Statistics")
    ax.set_xlabel("Target Domain & Format")
    ax.set_ylabel("Pixels")

    # Annotate bars with values
    for container in ax.containers:
        ax.bar_label(container, fmt='%.0f', padding=3)

# Plot count separately
ax = axes[2]
sns.barplot(x=distribution_df["target_domain"] + " - " +␣
 ↪distribution_df["format"], y=distribution_df["count"], ax=ax)
ax.set_title("Count of Formats per Target Domain")
```

```
ax.set_xlabel("Target Domain & Format")
ax.set_ylabel("Count")

# Annotate count bars
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f', padding=3)

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[ ]:
```
distribution_df
```

[ ]:

	target_domain	format	count	avg_width	avg_height	width_min	\
0	odoo.com	image	9	300.000000	414.000000	300.0	
1	odoo.com	text	137	380.000000	189.223140	380.0	
2	odoo.com	video	1	NaN	NaN	NaN	
3	inflowinventory.com	image	1	300.000000	600.000000	300.0	
4	inflowinventory.com	text	1	380.000000	239.000000	380.0	
5	squareup.com	image	77	305.500000	380.500000	300.0	
6	squareup.com	text	7	380.000000	171.285714	380.0	
7	squareup.com	video	1	648.000000	478.000000	648.0	
8	lightspeedhq.com	text	1	380.000000	199.000000	380.0	
9	zoho.com	image	190	526.180851	371.787234	160.0	
10	zoho.com	text	137	380.000000	205.510949	380.0	
11	zoho.com	video	25	648.000000	478.000000	648.0	

```
    width_q1  width_median  width_q3  width_max  height_min  height_q1  \
0      300.0         300.0     300.0      300.0       250.0      269.0
1      380.0         380.0     380.0      380.0       147.0      173.0
2        NaN           NaN       NaN        NaN         NaN        NaN
3      300.0         300.0     300.0      300.0       600.0      600.0
4      380.0         380.0     380.0      380.0       239.0      239.0
5      300.0         300.0     300.0      333.0       250.0      250.0
6      380.0         380.0     380.0      380.0       147.0      147.0
7      648.0         648.0     648.0      648.0       478.0      478.0
8      380.0         380.0     380.0      380.0       199.0      199.0
9      300.0         320.0     728.0     1940.0        50.0      250.0
10     380.0         380.0     380.0      380.0       131.0      173.0
11     648.0         648.0     648.0      648.0       478.0      478.0

    height_median  height_q3  height_max
0           269.0     600.00       600.0
1           173.0     199.00       464.0
2             NaN        NaN         NaN
3           600.0     600.00       600.0
4           239.0     239.00       239.0
5           291.5     533.25       600.0
6           153.0     183.00       239.0
7           478.0     478.00       478.0
8           199.0     199.00       199.0
9           250.0     600.00      1200.0
10          173.0     225.00       484.0
11          478.0     478.00       478.0
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785724059ed0>
```

```
from matplotlib import pyplot as plt
distribution_df['count'].plot(kind='hist', bins=20, title='count')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['avg_width'].plot(kind='hist', bins=20, title='avg_width')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['avg_height'].plot(kind='hist', bins=20, title='avg_height')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['width_min'].plot(kind='hist', bins=20, title='width_min')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785723fef950>
```

```
from matplotlib import pyplot as plt
```

```
import seaborn as sns
distribution_df.groupby('target_domain').size().plot(kind='barh', color=sns.
 ↪palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
distribution_df.groupby('format').size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785724175fd0>
```

```
from matplotlib import pyplot as plt
distribution_df.plot(kind='scatter', x='count', y='avg_width', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df.plot(kind='scatter', x='avg_width', y='avg_height', s=32, alpha=.
 ↪8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df.plot(kind='scatter', x='avg_height', y='width_min', s=32, alpha=.
 ↪8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df.plot(kind='scatter', x='width_min', y='width_q1', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78572400ca90>
```

```
from matplotlib import pyplot as plt
distribution_df['count'].plot(kind='line', figsize=(8, 4), title='count')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['avg_width'].plot(kind='line', figsize=(8, 4), title='avg_width')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['avg_height'].plot(kind='line', figsize=(8, 4),␣
 ↪title='avg_height')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
distribution_df['width_min'].plot(kind='line', figsize=(8, 4), title='width_min')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785724193e90>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
```

```
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['format'].value_counts()
    for x_label, grp in distribution_df.groupby('target_domain')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('target_domain')
_ = plt.ylabel('format')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785724193490>
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(distribution_df['target_domain'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(distribution_df, x='count', y='target_domain', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(distribution_df['format'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(distribution_df, x='count', y='format', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
```

```
figsize = (12, 1.2 * len(distribution_df['target_domain'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(distribution_df, x='avg_width', y='target_domain', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(distribution_df['format'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(distribution_df, x='avg_width', y='format', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
WARNING: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
```

[ ]:
```
distribution_df.describe()
```

[ ]:

	count	avg_width	avg_height	width_min	width_q1	\
count	12.000000	11.000000	11.000000	11.000000	11.000000	
mean	48.916667	420.698259	338.755185	386.909091	399.636364	
std	68.498783	128.509634	146.001750	145.162981	128.454874	
min	1.000000	300.000000	171.285714	160.000000	300.000000	
25%	1.000000	342.750000	202.255474	300.000000	300.000000	
50%	8.000000	380.000000	371.787234	380.000000	380.000000	
75%	92.000000	453.090426	446.000000	380.000000	380.000000	
max	190.000000	648.000000	600.000000	648.000000	648.000000	

	width_median	width_q3	width_max	height_min	height_q1	\
count	11.000000	11.000000	11.000000	11.000000	11.00000	
mean	401.454545	438.545455	551.727273	269.909091	296.00000	
std	127.037289	156.923143	476.220976	173.265377	151.24219	
min	300.000000	300.000000	300.000000	50.000000	147.00000	
25%	310.000000	340.000000	356.500000	147.000000	186.00000	
50%	380.000000	380.000000	380.000000	239.000000	250.00000	
75%	380.000000	514.000000	514.000000	364.000000	373.50000	
max	648.000000	728.000000	1940.000000	600.000000	600.00000	

	height_median	height_q3	height_max
count	11.000000	11.000000	11.000000
mean	300.318182	394.022727	507.363636

```
std       149.896343  182.747793   273.635989
min       153.000000  183.000000   199.000000
25%       186.000000  212.000000   351.500000
50%       250.000000  478.000000   478.000000
75%       384.750000  566.625000   600.000000
max       600.000000  600.000000  1200.000000
```

[ ]:
```
# Group the distribution_df by 'format', but ignore zero values in the␣
 ↪aggregation
grouped_summary = combined_df.groupby('format').agg(
    count=('format', 'count'),
    overall_width_mean = ('width', lambda x: x[x != 0].mean() if not x[x != 0].
 ↪empty else np.nan),
    overall_width_min = ('width', lambda x: x[x != 0].min() if not x[x != 0].
 ↪empty else np.nan),
    overall_width_median = ('width', lambda x: x[x != 0].median() if not x[x !=␣
 ↪0].empty else np.nan),
    overall_width_max = ('width', lambda x: x[x != 0].max() if not x[x != 0].
 ↪empty else np.nan),
    overall_height_mean = ('height', lambda x: x[x != 0].mean() if not x[x !=␣
 ↪0].empty else np.nan),
    overall_height_min = ('height', lambda x: x[x != 0].min() if not x[x != 0].
 ↪empty else np.nan),
    overall_height_median = ('height', lambda x: x[x != 0].median() if not x[x !
 ↪= 0].empty else np.nan),
    overall_height_max = ('height', lambda x: x[x != 0].max() if not x[x != 0].
 ↪empty else np.nan)
).reset_index()


# (Optional) Save the summary to a CSV file
grouped_summary.to_csv("grouped_ad_dimension_summary_nonzero_2.csv",␣
 ↪index=False)
```

[ ]:
```
grouped_summary
```

[ ]:
```
   format  count  overall_width_mean  overall_width_min  overall_width_median  \
0   image    277          508.602941              160.0                 300.0
1    text    283          380.000000              380.0                 380.0
2   video     27          648.000000              648.0                 648.0

   overall_width_max  overall_height_mean  overall_height_min  \
0             1940.0           375.024510                50.0
1              380.0           197.333333               131.0
2              648.0           478.000000               478.0
```

```
   overall_height_median  overall_height_max
0                  250.0              1200.0
1                  173.0               484.0
2                  478.0               478.0
```

[ ]:
```
# @title Pivoting the DataFrame Visualization
df_pivot = grouped_summary.melt(id_vars=["format", "count"],
                    value_vars=["overall_width_mean", "overall_width_median",␣
 ↪"overall_width_max",
                                "overall_height_mean", "overall_height_median",␣
 ↪"overall_height_max"],
                    var_name="statistic", value_name="value")

# Extracting metric type (width or height)
df_pivot["metric"] = df_pivot["statistic"].apply(lambda x: "Width" if "width"␣
 ↪in x else "Height")
df_pivot["statistic"] = df_pivot["statistic"].apply(lambda x: x.split("_")[-1].
 ↪capitalize())

# Plotting
fig, axes = plt.subplots(1, 3, figsize=(18, 5), sharey=False)
sns.set(style="whitegrid")

for i, metric in enumerate(["Width", "Height"]):
    pivot_subset = df_pivot[df_pivot["metric"] == metric]
    ax = axes[i]

    plot_data = pivot_subset.pivot(index="format", columns="statistic",␣
 ↪values="value")
    plot_data.plot(kind="bar", ax=ax, cmap="viridis")
    ax.set_title(f"Overall {metric} Statistics")
    ax.set_xlabel("Format")
    ax.set_ylabel("Pixels")

    # Annotate bars with values
    for container in ax.containers:
        ax.bar_label(container, fmt='%.0f', padding=3)

# Plot count separately
ax = axes[2]
sns.barplot(x=grouped_summary["format"], y=grouped_summary["count"], ax=ax)
ax.set_title("Count of Formats")
ax.set_xlabel("Format")
ax.set_ylabel("Count")

# Annotate count bars
for container in ax.containers:
```

```
    ax.bar_label(container, fmt='%.0f', padding=3)

plt.tight_layout()
plt.show()
```