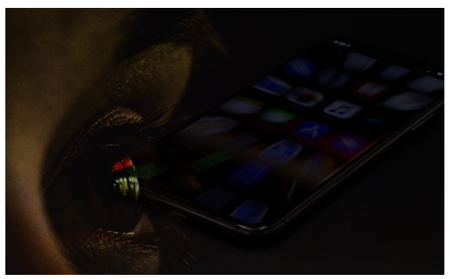
PRA2 - Limpieza y análisis de los datos

Alumno: Luis Manuel Molina Coca (luimoco)

1. Descripción del dataset

Dataset del tratamiento de la privacidad del usuario por parte de las aplicaciones móviles



Descripción
El dataset *Tratamiento de la privacidad del usuario por parte de aplicaciones móviles*, proporciona información de rastreadores que se han incluido en la aplicación y los permisos del dispositivo que ha de aceptar el usuario en el momento de su instalación.
Adicionalmente proporciona más características de la aplicación interesantes para el tratamiento analítico de aplicaciones móviles.

Como en toda revolución, la industria trata de sacar tajada del recurso que proporcione riqueza en cada momento. En esta era en la que el dato es tan valioso, han proliferado las herramientas para extraer la mayor cantidad de información posible de personas e instituciones mediante el uso de **rastreadores** de empresas analíticas y con un abuso poco justificado de los **permisos** necesarios para el funcionamiento de la aplicación con tal de recopilar toda la información posible.
En ocasiones esta información es utilizada de manera benévola para la mejora del servicio o interés de la ciudadanía, pero en la mayoría de las ocasiones se usa de manera **fraudulenta**, bien para la venta de información sin consentimiento ni beneficio del usuario o directamente secuestrando o dañando el dispositivo con fines de extorsión.
Es muy interesante que la población, en general gran productora de los datos, tome conciencia de este interés por su información y esté prevenida ante un uso fraudulento que ataque a su propio derecho de privacidad.

Con este dataset es posible generar conocimiento sobre la evolución de este mercado de los datos que maximiza la extracción de información minusvalorando la privacidad del usuario.
Bajo un punto de vista analítico, el dataset tiene una utilidad destacada en tareas de analítica descriptiva orientadas al ámbito de la situación actual y la evolución en la pérdida de privacidad:

- Métodos **no supervisados**
  - Clustering de aplicaciones benévolas o maliciosas orientadas a servicio o recopilación de datos.
- Métodos **supervisados**
  - Clasificación de aplicaciones en base a los conjuntos encontrados en el análisis anterior.

El dataset *Tratamiento de la privacidad del usuario por parte de aplicaciones móviles* puede encontrarse en los siguientes repositorios:

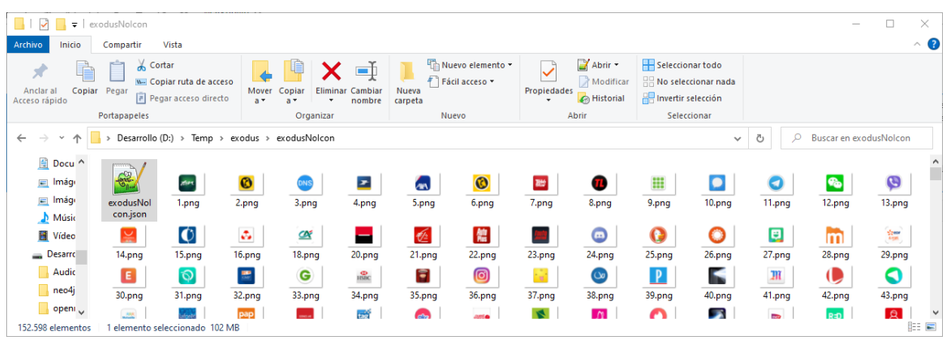
- **Zenodo**: DOI: 10.5281/zenodo.4261664
- **Google Drive**: Carpeta data

2. Integración y selección de los datos de interés a analizar

2.1. Obtención y exploración de los ficheros de datos

Procedemos a la descarga de los datos desde el repositorio Zenodo. En la descripción indica que existen dos versiones del dataset una *exodus.json* de 3GB en la que se integran los iconos de las aplicaciones en el propio fichero descompuestas en componentes [RGBA] y una versión ligera *exodusNoIcon.json* de 100MB donde las imagenes se ubican en formato PNG en una carpeta aparte.

Descargamos la versión ligera *exodusNoIcon.zip (506,7MB)* y descomprimimos el contenido. Tras la descompresión, se extraen a la carpeta de descarga 152.528 elementos. El dataset *exodusNoIcon.json*" en formato json de 102MB y 152.597 imágenes en formato PNG de iconos de aplicaciones que ocupan en total en disco 877MB.



Pasamos la información del dataset directamente a la carpeta *data* del directorio de trabajo de Python y el conjunto de imágenes en una nueva carpeta *img* dentro de este mismo directorio.

Cargamos el fichero json y echamos un primer vistazo a su contenido.

In [1]:
```
###Carga del fichero json exodusNoIcon.json y exploración del contenido.###

#Importar librerías necesarias
import json
import sys
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%pylab inline
import numpy as np
np.set_printoptions(suppress=True)

#Carga del fichero
with open('data/exodusNoIcon.json') as json_file:
    data_original = json.load(json_file)

#Exploración en la variable
print('Tipo de datos de la variable tras la carga del fichero: ' + str(type(data_original)))
print('\nRegistros contenidos en el objeto dictionary: ' + str(len(data_original)))
print('\nVisualizar los tres primeros registros:')
print(list(data_original.items())[:3])
print('\nVisualizar los tres últimos registros:')
print(list(data_original.items())[-3:])
print('\nPresentar el primer elemento de clave (key) \'1\':')
print(json.dumps(data_original['1'], indent = 2))
#Visualizar imagen del icono correspondiente
plt.figure(figsize = (3,3))
plt.imshow(mpimg.imread('data/img/1.png'))
plt.title('/data/img/1.png')
plt.axis('off')
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
Tipo de datos de la variable tras la carga del fichero: <class 'dict'>

Registros contenidos en el objeto dictionary: 153012

Visualizar los tres primeros registros:
```

[('1', {'Id': 1, 'Name': 'Tan', 'Tracker_count': 3, 'Permissions_count': 9, 'Version': '5.7.0', 'Downloads': '100,000+', 'Analysis_date': '12-11-2017', 'Trackers': [{'Google AdMob': ['advertisement']}, {'Google Analytics': ['analytics']}, {'Google Tag Manager': ['analytics']}], 'Permissions': ['ACCESS_FINE_LOCATION', 'ACCESS_NETWORK_STATE', 'INTERNET', 'READ_PHONE_STATE', 'VIBRATE', 'WAKE_LOCK', 'WRITE_EXTERNAL_STORAGE', 'RECEIVE', 'C2D_MESSAGE'], 'Permissions_warning_count': 2, 'Developer': 'SQLI', 'Country': 'FR'}), ('2', {'Id': 2, 'Name': 'AlloCine', 'Tracker_count': 2, 'Permissions_count': 7, 'Version': '3.3.5', 'Downloads': '5,000,000+', 'Analysis_date': '12-11-2017', 'Trackers': [{'Mobile Engagement': ['analytics']}, {'Smart': []}], 'Permissions': ['ACCESS_FINE_LOCATION', 'ACCESS_NETWORK_STATE', 'INTERNET', 'READ_PHONE_STATE', 'WRITE_EXTERNAL_STORAGE', 'C2D_MESSAGE', 'RECEIVE'], 'Permissions_warning_count': 2, 'Developer': 'Allocine', 'Country': 'FR'}), ('3', {'Id': 3, 'Name': 'DNS Changer (No Root - IPv6 - 3G/wifi)', 'Tracker_count': 0, 'Permissions_count': 5, 'Version': '1.15.2', 'Downloads': '100,000+', 'Analysis_date': '12-11-2017', 'Trackers': [], 'Permissions': ['ACCESS_NETWORK_STATE', 'INTERNET', 'RECEIVE_BOOT_COMPLETED', 'VIBRATE', 'INSTALL_SHORTCUT'], 'Permissions_warning_count': 0, 'Developer': '-', 'Country': '49'})]

Visualizar los tres últimos registros:

[('153371', {'Id': 153371, 'Name': 'Jelbi', 'Tracker_count': 2, 'Permissions_count': 13, 'Version': '3.21.0', 'Downloads': 'na', 'Analysis_date': '8-11-2020', 'Trackers': [{'Bugsnag': ['crash reporting']}, {'Google Firebase Analytics': ['analytics']}], 'Permissions': ['ACCESS_FINE_LOCATION', 'ACCESS_NETWORK_STATE', 'ACCESS_WIFI_STATE', 'CAMERA', 'FOREGROUND_SERVICE', 'INTERNET', 'READ_EXTERNAL_STORAGE', 'RECEIVE_BOOT_COMPLETED', 'RECORD_AUDIO', 'WAKE_LOCK', 'WRITE_EXTERNAL_STORAGE', 'RECEIVE', 'BIND_GET_INSTALL_REFERRER_SERVICE'], 'Permissions_warning_count': 2, 'Developer': 'Google Inc.', 'Country': 'US'}), ('153372', {'Id': 153372, 'Name': 'kvgOF Hopper', 'Tracker_count': 2, 'Permissions_count': 8, 'Version': '2.8.11', 'Downloads': 'na', 'Analysis_date': '8-11-2020', 'Trackers': [{'Google CrashLytics': ['crash reporting']}, {'Google Firebase Analytics': ['analytics']}], 'Permissions': ['ACCESS_COARSE_LOCATION', 'ACCESS_FINE_LOCATION', 'ACCESS_NETWORK_STATE', 'INTERNET', 'VIBRATE', 'WAKE_LOCK', 'RECEIVE', 'BIND_GET_INSTALL_REFERRER_SERVICE'], 'Permissions_warning_count': 0, 'Developer': 'Door2Door GmbH', 'Country': 'DE'}), ('153373', {'Id': 153373, 'Name': 'Microsoft SwiftKey Keyboard', 'Tracker_count': 3, 'Permissions_count': 11, 'Version': '7.6.7.5', 'Downloads': 'na', 'Analysis_date': '8-11-2020', 'Trackers': [{'Google Analytics': ['analytics']}, {'Google Tag Manager': ['analytics']}, {'Microsoft Visual Studio App Center Crashes': ['crash reporting']}], 'Permissions': ['ACCESS_NETWORK_STATE', 'ACCESS_WIFI_STATE', 'GET_ACCOUNTS', 'INTERNET', 'RECEIVE_BOOT_COMPLETED', 'VIBRATE', 'WAKE_LOCK', 'WRITE_EXTERNAL_STORAGE', 'RECEIVE', 'READLANG', 'READCONFIG'], 'Permissions_warning_count': 2, 'Developer': 'TouchType Limited', 'Country': 'GB'})]

Presentar el primer elemento de clave (key) '1':


```
{
  "Id": 1,
  "Name": "Tan",
  "Tracker_count": 3,
  "Permissions_count": 9,
  "Version": "5.7.0",
  "Downloads": "100,000+",
  "Analysis_date": "12-11-2017",
  "Trackers": [
    {
      "Google AdMob": [
        "advertisement"
      ]
    },
    {
      "Google Analytics": [
        "analytics"
      ]
    },
    {
      "Google Tag Manager": [
        "analytics"
      ]
    }
  ],
  "Permissions": [
    "ACCESS_FINE_LOCATION",
    "ACCESS_NETWORK_STATE",
    "INTERNET",
    "READ_PHONE_STATE",
    "VIBRATE",
    "WAKE_LOCK",
    "WRITE_EXTERNAL_STORAGE",
    "RECEIVE",
    "C2D_MESSAGE"
  ],
  "Permissions_warning_count": 2,
  "Developer": "SQLI",
  "Country": "FR"
}
```


/data/img/1.png

Con esta exploración, vemos:

- Que el fichero json se vuelca en una variable de tipo *Dictionary* de Python que tiene una estructura similar y es idónea para trabajar con este formato.
- Que la variable tiene **153.012** entradas lo que corresponde a la información de 153.012 aplicaciones.
  - Al descomprimir el fichero, habíamos contado la creación de **152.597** ficheros de imagen, lo que implica una primera información acerca de **elementos vacíos**. `Existen 415 aplicaciones donde no vamos a tener información acerca del icono.`
- La clave del diccionario (key) coincide con el atributo **Id** en su valor (value) correspondiente.
- La representación del primer registro de clave '*1*' como diccionario de los datos o **Id** *1* como registro del dataset.
- El nombre del fichero PNG que contiene la información del icono, coincide también con el valor del atributo **Id**

De esta manera, podemos contrastar la estructura de los datos (izquierda) con el esquema proporcionado en la información del dataset de Zenodo (derecha).

new 1


```
{
    "Id": 1,
    "Name": "Tan",
    "Tracker_count": 3,
    "Permissions_count": 9,
    "Version": "5.7.0",
    "Downloads": "100,000+",
    "Analysis_date": "12-11-2017",
    "Trackers": [
        {
            "Google AdMob": [
                "advertisement"
            ]
        },
        {
            "Google Analytics": [
                "analytics"
            ]
        },
        {
            "Google Tag Manager": [
                "analytics"
            ]
        }
    ],
    "Permissions": [
        "ACCESS_FINE_LOCATION",
        "ACCESS_NETWORK_STATE",
        "INTERNET",
        "READ_PHONE_STATE",
        "VIBRATE",
        "WAKE_LOCK",
        "WRITE_EXTERNAL_STORAGE",
        "RECEIVE",
        "C2D_MESSAGE"
    ],
    "Permissions_warning_count": 2,
    "Developer": "SQLI",
    "Country": "FR"
}
```


new 2


```
{
    "id": {
        "Id": id,
        "Name": "name",
        "Tracker_count": trackersCount,
        "Permissions_count": permissionsCount,
        "Version": "version",
        "Downloads": "downloads",
        "Analysis_date": "analysisDate",
        "Trackers": [
            {
                "Tracker Name": [
                    "trackerPurpose"
                ]
            }
        ],
        "Permissions": [
            "permission",
        ],
        "Permissions_warning_count": permissionWarningCount,
        "Developer": "developer",
        "Country": "country",
        "Icon": [
            [
                R,
                G,
                B,
                A
            ]
        ]
    }
}
```


Destacamos:

- En los datos cargados se rescata el valor de la key '1' por lo que éste no se muestra por ser un metadato interno del tipo diccionario. No obstante se ha constatado que este valor y el atributo Id van a coincidir.
- La aplicación identificada como *1*, de nombre *Tan*, tiene tres rastreadores *Trackers* cuya estructura de presentación coincide con la descripción esquemática del dataset aunque en ésta solo se representa una única aparición.
- Igual con la lista de *Permissions* en la que se observan 9 de ellos y su formación se ajusta a la descrita en el dataset.
- No tenemos información de la lista de píxeles en RGBA que indica la descripción del dataset, puesto que hemos escogido el dataset sin iconos integrados.
- **Claves desaparecidas**: Con esa primera aproximación a los datos, no pudimos constatar la existencia de registros en los cuales las claves *Developer* y *Country* podían no estar presentes. Al realizar el tratamiento de conversión de JSON a Data Frame y tratar cada registro individualmente, observamos la ausencia de estas claves en algunos registros y será un hecho de **elementos vacíos** que se deberá subsanar.

2.2. Generación del Data Frame

Para trabajar con el conjunto de datos, tendremos que realizar inicialmente una serie de operaciones para adaptarlo a una estructura tabular de registros y atributos o *Data Frame*.

Para esta labor tendremos en cuenta:

- El atributo *Trackers*, su subatributo *TrackerPurposse* y el atributo *Permissions* son listados de posibles valores y necesitaremos un tratamiento para "aplanarlo" en un registro.
  - 1 aplicación puede tener 0..n Trackers
    - 1 Tracker puede tener 0..n TrackerPurpose
  - 1 aplicación puede tener 0..1 Permissions
- El resto de atributos Id, Name, Tracker_count, Permissions_count, Version, Downloads, Analysis_date, Permissions_warning_count, Developer y Country son singulares para la misma aplicación.

2.2.1. Aplanar los atributos Trackers, TrackerPurpose y Permissions

La estrategia que vamos a seguir para aplanar los atributos *Trackers* y *Permissions*, será a través de una **integración horizontal** para generar tantos atributos binarios y singulares en el Data Frame como posibles rastreadores o permisos existan registrados en todo el conjunto de datos capturado.

En cuanto a la estrategia para el atributo *TrackerPurpose* al tratarse de un atributo descriptivo, invariable, dependiente y específico de cada *Tracker* podremos optar por una estrategia de **selección** y no considerarlos en el Data Frame final. Si el usuario quiere obtener la información de propósitos de un rastreador específico, bastaría con buscar una ocurrencia del rastreador en el conjunto de datos original y observar su listado de propósitos.

2.2.1.1. Integración horizontal de Trackers y Permissions (Paso previo)

Para resolver el problema, crearemos sendas listas que contengan todos los valores únicos de nombres de *Trackers* y nombres de *Permissions* para poder conocer cuántos atributos distintos tendremos que generar en el Data Frame final para poder albergar, como información binaria, si la aplicación en cuestión contiene ese rastreador *1* o no lo contiene *0* y si solicita *1* o prescinde *0* de cada permiso en cuestión.

In [2]:
```
### Obtención de las columnas para la información binaria de Trackers y Permissions ###

#Importar librerías necesarias
import pandas as pd
pd.options.display.float_format = '{:.4f}'.format

#El siguiente bucle, recorre todos los registros del dataset y para cada aplicación introduce los rastreadores y permisos que posée en su lista correspondiente.
lista_trackers = []
lista_permissions = []
for app in data_original.values():
    for rastreador in app['Trackers']:
        for key, value in rastreador.items():
            lista_trackers.append(key.upper())
    for permiso in app['Permissions']:
        lista_permissions.append(permiso.upper())

#Mostramos el tamaño de las listas
print('Total de rastreadores: ' + str(len(lista_trackers)))
print('Total de permisos: ' + str(len(lista_permissions)))

#Agrupamos los distintos permisos por su nombre y mostramos su número de ocurrencias
agrup_trackers = pd.Series(lista_trackers).value_counts()
print('\nAgrupación de los trackers obtenidos\n')
print(agrup_trackers.head(5))
print('...')
print(agrup_trackers.tail(5))
print('\nTotal de trackers únicos: ' + str(len(agrup_trackers)))

agrup_permissions = pd.Series(lista_permissions).value_counts()
print('\nAgrupación de los permisos obtenidos\n')
print(agrup_permissions.head(5))
print('...')
print(agrup_permissions.tail(5))
print('\nTotal de permisos únicos: ' + str(len(agrup_permissions)))
```

```
Total de rastreadores: 681869
Total de permisos: 2095467

Agrupación de los trackers obtenidos

GOOGLE FIREBASE ANALYTICS    81809
GOOGLE ADMOB                 61485
GOOGLE CRASHLYTICS           55929
GOOGLE ANALYTICS             39665
FACEBOOK LOGIN               37173
dtype: int64
...
TAGCOMMANDER (COMMANDERS ACT.)    4
SHOPKICK                          3
SENSORO                           2
BAIDU MAPS                        1
VPON                              1
dtype: int64

Total de trackers únicos: 312

Agrupación de los permisos obtenidos

INTERNET                  142029
ACCESS_NETWORK_STATE      137008
WAKE_LOCK                 115937
WRITE_EXTERNAL_STORAGE    105664
RECEIVE                    93793
dtype: int64
...
USERINFO_PROVIDER     1
SWITCH2AGENT          1
SMART_NOTIFY_FAULT    1
BDD                   1
MDM_DEVICE_MANAGER    1
dtype: int64

Total de permisos únicos: 4003
```

Observamos:

- Que el número de distintos rastreadores y permisos es muy elevado y resultaría un Data Frame con demasiados atributos.
- Que tenemos información sobre los rastreadores y permisos más utilizados y podemos establecer un filtro.

Atendiendo al número de rastreadores distintos *312* y de permisos *4.003* consideramos para nuestro Data Frame final, tomar únicamente de manera aproximada el 10% de los rastreadores y permisos más utilizados para realizar nuestros estudios. De tal manera que nos quedemos con 30 rastreadores y 40 permisos distintos.

In [3]:
```
### Acciones para seleccionar en las listas de trackers y permisos los 30 y 40 más utilizados ###

lista_trackers = list(agrup_trackers.head(30).keys())
lista_permissions = list(agrup_permissions.head(40).keys())

print('Nuevos atributos binarios para Trackers:\n' + str(lista_trackers))
print('\nNuevos atributos binarios para Permissions:\n' + str(lista_permissions))
```

```
Nuevos atributos binarios para Trackers:
['GOOGLE FIREBASE ANALYTICS', 'GOOGLE ADMOB', 'GOOGLE CRASHLYTICS', 'GOOGLE ANALYTICS', 'FACEBOOK LOGIN', 'FACEBOOK SHARE', 'FACEBOOK ANALYTICS', 'GOOGLE TAG MANAGER', 'FACEBOOK ADS', 'FACEBOOK PLACES', 'FLURRY', 'INMOBI', 'MOAT',
'APPSFLYER', 'TWITTER MOPUB', 'UNITY3D ADS', 'APPLOVIN (MAX AND SPARKLABS)', 'ADJUST', 'INTEGRAL AD SCIENCE', 'ADCOLONY', 'AMAZON ADVERTISEMENT', 'IRONSOURCE', 'CHARTBOOST', 'BRANCH', 'ONESIGNAL', 'MILLENNIAL MEDIA', 'HOCKEYAPP',
'TAPJOY', 'MIXPANEL', 'MYTARGET']

Nuevos atributos binarios para Permissions:
['INTERNET', 'ACCESS_NETWORK_STATE', 'WAKE_LOCK', 'WRITE_EXTERNAL_STORAGE', 'RECEIVE', 'READ_EXTERNAL_STORAGE', 'ACCESS_WIFI_STATE', 'VIBRATE', 'RECEIVE_BOOT_COMPLETED', 'BIND_GET_INSTALL_REFERRER_SERVICE', 'ACCESS_FINE_LOCATION',
'BILLING', 'C2D_MESSAGE', 'ACCESS_COARSE_LOCATION', 'CAMERA', 'READ_PHONE_STATE', 'GET_ACCOUNTS', 'FOREGROUND_SERVICE', 'WRITE_SETTINGS', 'BLUETOOTH', 'RECORD_AUDIO', 'READ_CONTACTS', 'READ_GSERVICES', 'SYSTEM_ALERT_WINDOW', 'CHANG
E_WIFI_STATE', 'READ_SETTINGS', 'READ', 'MODIFY_AUDIO_SETTINGS', 'WRITE', 'BROADCAST_BADGE', 'UPDATE_SHORTCUT', 'USE_CREDENTIALS', 'BLUETOOTH_ADMIN', 'UPDATE_COUNT', 'UPDATE_BADGE', 'USE_FINGERPRINT', 'CHANGE_BADGE', 'GET_TASKS',
'PROVIDER_INSERT_BADGE', 'READ_APP_BADGE']
```

2.2.2. Conversión del Json a un DataFrame tabular de Pandas

La estrategia a seguir será la siguiente:

- Navegar por la variable diccionario donde cargamos el dataset en formato Json registro a registro.
- Para cada registro:
  - Informar en una estructura de lista, donde cada elemento represente una columna del Data Frame final, cada uno de los atributos descritos en el dataset. A esta lista se añaden las listas de Rastreadores y Permisos que se van a considerar, obtenidas en el proceso anterior.:
    - Copia directa en el caso de atributos singulares: *Id, Name, Tracker_count, Permissions_count, Version, Downloads, Analysis_date, Permissions_warning_count, Developer y Country*
    - Para el atributo donde se listan los rastreadores:
      - Si la aplicación contiene alguno de los rastreadores a considerar, consignar 1 en éstos y 0 en el resto.
      - Si la aplicación no contiene rastreadores o éstos no se encuentran entre los considerados, consignar 0 en el total de rastreadores del registro.
    - Para el atributo donde se listan los permisos:
      - Si la aplicación contiene alguno de los permisos a considerar, consignar 1 en éstos y 0 en el resto.
      - Si la aplicación no contiene permisos o éstos no se encuentran entre los considerados, consignar 0 en el total de permisos del registro.
  - Ingresar el registro confeccionado en el Data Frame.

In [4]:
```
### Conversión del Json a DataFrame ###

#Configurar los nombres de los atributos o columnas del DataFrame
columnas = ['Id', 'Name', 'Tracker_count', 'Permissions_count', 'Version', 'Downloads', 'Analysis_date']
columnas.extend(lista_trackers)
columnas.extend(lista_permissions)
columnas.extend(['Permissions_warning_count','Developer','Country'])

#Crear una estructura de registros en listas para generar el dataframe de manera eficiente a partir de ésta.
lista_carga = []

#Iteramos el dataset cargado en la variable de tipo diccionario
for app in data_original.values():
    #Crear la estructura e inicializarla
    estr = {}
    estr['Id'] = np.nan
    estr['Name'] = np.nan
    estr['Tracker_count'] = np.nan
    estr['Permissions_count'] = np.nan
    estr['Version'] = np.nan
    estr['Downloads'] = np.nan
    estr['Analysis_date'] = np.nan
    estr['GOOGLE FIREBASE ANALYTICS'] = 0; estr['GOOGLE ADMOB'] = 0; estr['GOOGLE CRASHLYTICS'] = 0; estr['GOOGLE ANALYTICS'] = 0; estr['FACEBOOK LOGIN'] = 0;
    estr['FACEBOOK SHARE'] = 0; estr['FACEBOOK ANALYTICS'] = 0; estr['GOOGLE TAG MANAGER'] = 0; estr['FACEBOOK ADS'] = 0; estr['FACEBOOK PLACES'] = 0;
    estr['FLURRY'] = 0; estr['INMOBI'] = 0; estr['MOAT'] = 0; estr['APPSFLYER'] = 0; estr['TWITTER MOPUB'] = 0; estr['UNITY3D ADS'] = 0;
    estr['APPLOVIN (MAX AND SPARKLABS)'] = 0; estr['ADJUST'] = 0; estr['INTEGRAL AD SCIENCE'] = 0; estr['ADCOLONY'] = 0; estr['AMAZON ADVERTISEMENT'] = 0;
    estr['IRONSOURCE'] = 0; estr['CHARTBOOST'] = 0; estr['BRANCH'] = 0; estr['ONESIGNAL'] = 0; estr['MILLENNIAL MEDIA'] = 0; estr['HOCKEYAPP'] = 0;
    estr['TAPJOY'] = 0; estr['MIXPANEL'] = 0; estr['MYTARGET'] = 0
    estr['INTERNET'] = 0; estr['ACCESS_NETWORK_STATE'] = 0; estr['WAKE_LOCK'] = 0; estr['WRITE_EXTERNAL_STORAGE'] = 0; estr['RECEIVE'] = 0;
    estr['READ_EXTERNAL_STORAGE'] = 0; estr['ACCESS_WIFI_STATE'] = 0; estr['VIBRATE'] = 0; estr['RECEIVE_BOOT_COMPLETED'] = 0;
    estr['BIND_GET_INSTALL_REFERRER_SERVICE'] = 0; estr['ACCESS_FINE_LOCATION'] = 0; estr['BILLING'] = 0; estr['C2D_MESSAGE'] = 0; estr['ACCESS_COARSE_LOCATION'] = 0;
    estr['CAMERA'] = 0; estr['READ_PHONE_STATE'] = 0; estr['GET_ACCOUNTS'] = 0; estr['FOREGROUND_SERVICE'] = 0; estr['WRITE_SETTINGS'] = 0; estr['BLUETOOTH'] = 0;
    estr['RECORD_AUDIO'] = 0; estr['READ_CONTACTS'] = 0; estr['READ_GSERVICES'] = 0; estr['SYSTEM_ALERT_WINDOW'] = 0; estr['CHANGE_WIFI_STATE'] = 0;
    estr['READ_SETTINGS'] = 0; estr['READ'] = 0; estr['MODIFY_AUDIO_SETTINGS'] = 0; estr['WRITE'] = 0; estr['BROADCAST_BADGE'] = 0; estr['UPDATE_SHORTCUT'] = 0;
```

```
    estr['USE_CREDENTIALS'] = 0; estr['BLUETOOTH_ADMIN'] = 0; estr['UPDATE_COUNT'] = 0; estr['UPDATE_BADGE'] = 0; estr['USE_FINGERPRINT'] = 0; estr['CHANGE_BADGE'] = 0;
    estr['GET_TASKS'] = 0; estr['PROVIDER_INSERT_BADGE'] = 0; estr['READ_APP_BADGE'] = 0
    estr['Permissions_warning_count'] = np.nan
    estr['Developer'] = np.nan
    estr['Country'] = np.nan

    #Tratamiento de claves perdidas
    if 'Developer' not in app.keys(): app['Developer'] = 'na' #Se añade la clave con valor indeterminado.
    if 'Country' not in app.keys(): app['Country'] = 'na'

    #Si se ha informado, completar la información de cada atributo
    if app['Id'] != 'na': estr['Id'] = app['Id']
    if app['Name'] != 'na': estr['Name'] = app['Name']
    if app['Tracker_count'] != 'na': estr['Tracker_count'] = app['Tracker_count']
    if app['Permissions_count'] != 'na': estr['Permissions_count'] = app['Permissions_count']
    if app['Version'] != 'na': estr['Version'] = app['Version']
    if app['Downloads'] != 'na': estr['Downloads'] = app['Downloads']
    if app['Analysis_date'] != 'na': estr['Analysis_date'] = app['Analysis_date']
    if app['Permissions_warning_count'] != 'na': estr['Permissions_warning_count'] = app['Permissions_warning_count']
    if app['Developer'] != 'na': estr['Developer'] = app['Developer']
    if app['Country'] != 'na': estr['Country'] = app['Country']
    #Navegar por los distintos rastreadores de la app y marcar 1 en el caso de coincidir con las estudiadas
    for rastreador in app['Trackers']:
        for key, value in rastreador.items():
            if key.upper() in lista_trackers: estr[key.upper()] = 1
    #Navegar por los distintos rastreadores de la app y marcar 1 en el caso de coincidir con las estudiadas
    for permiso in app['Permissions']:
        if permiso.upper() in lista_permissions: estr[permiso.upper()] = 1

    lista_carga.append(list(estr.values()))

#Crear el DataFrame a partir de la lista
df_exodus = pd.DataFrame(lista_carga, columns = columnas)
print(df_exodus.info())
df_exodus.head(3).append(df_exodus.tail(3))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153012 entries, 0 to 153011
Data columns (total 80 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Id                              153012 non-null  int64
 1   Name                            153012 non-null  object
 2   Tracker_count                   153012 non-null  int64
 3   Permissions_count               153012 non-null  int64
 4   Version                         149880 non-null  object
 5   Downloads                       80730 non-null   object
 6   Analysis_date                   152871 non-null  object
 7   GOOGLE FIREBASE ANALYTICS       153012 non-null  int64
 8   GOOGLE ADMOB                    153012 non-null  int64
 9   GOOGLE CRASHLYTICS              153012 non-null  int64
 10  GOOGLE ANALYTICS                153012 non-null  int64
 11  FACEBOOK LOGIN                  153012 non-null  int64
 12  FACEBOOK SHARE                  153012 non-null  int64
 13  FACEBOOK ANALYTICS              153012 non-null  int64
 14  GOOGLE TAG MANAGER              153012 non-null  int64
 15  FACEBOOK ADS                    153012 non-null  int64
 16  FACEBOOK PLACES                 153012 non-null  int64
 17  FLURRY                          153012 non-null  int64
 18  INMOBI                          153012 non-null  int64
 19  MOAT                            153012 non-null  int64
 20  APPSFLYER                       153012 non-null  int64
 21  TWITTER MOPUB                   153012 non-null  int64
 22  UNITY3D ADS                     153012 non-null  int64
 23  APPLOVIN (MAX AND SPARKLABS)    153012 non-null  int64
 24  ADJUST                          153012 non-null  int64
 25  INTEGRAL AD SCIENCE             153012 non-null  int64
 26  ADCOLONY                        153012 non-null  int64
 27  AMAZON ADVERTISEMENT            153012 non-null  int64
 28  IRONSOURCE                      153012 non-null  int64
 29  CHARTBOOST                      153012 non-null  int64
 30  BRANCH                          153012 non-null  int64
 31  ONESIGNAL                       153012 non-null  int64
 32  MILLENNIAL MEDIA                153012 non-null  int64
 33  HOCKEYAPP                       153012 non-null  int64
 34  TAPJOY                          153012 non-null  int64
 35  MIXPANEL                        153012 non-null  int64
 36  MYTARGET                        153012 non-null  int64
 37  INTERNET                        153012 non-null  int64
 38  ACCESS_NETWORK_STATE            153012 non-null  int64
```

```
 39  WAKE_LOCK                           153012 non-null  int64
 40  WRITE_EXTERNAL_STORAGE              153012 non-null  int64
 41  RECEIVE                             153012 non-null  int64
 42  READ_EXTERNAL_STORAGE               153012 non-null  int64
 43  ACCESS_WIFI_STATE                   153012 non-null  int64
 44  VIBRATE                             153012 non-null  int64
 45  RECEIVE_BOOT_COMPLETED              153012 non-null  int64
 46  BIND_GET_INSTALL_REFERRER_SERVICE   153012 non-null  int64
 47  ACCESS_FINE_LOCATION                153012 non-null  int64
 48  BILLING                             153012 non-null  int64
 49  C2D_MESSAGE                         153012 non-null  int64
 50  ACCESS_COARSE_LOCATION              153012 non-null  int64
 51  CAMERA                              153012 non-null  int64
 52  READ_PHONE_STATE                    153012 non-null  int64
 53  GET_ACCOUNTS                        153012 non-null  int64
 54  FOREGROUND_SERVICE                  153012 non-null  int64
 55  WRITE_SETTINGS                      153012 non-null  int64
 56  BLUETOOTH                           153012 non-null  int64
 57  RECORD_AUDIO                        153012 non-null  int64
 58  READ_CONTACTS                       153012 non-null  int64
 59  READ_GSERVICES                      153012 non-null  int64
 60  SYSTEM_ALERT_WINDOW                 153012 non-null  int64
 61  CHANGE_WIFI_STATE                   153012 non-null  int64
 62  READ_SETTINGS                       153012 non-null  int64
 63  READ                                153012 non-null  int64
 64  MODIFY_AUDIO_SETTINGS               153012 non-null  int64
 65  WRITE                               153012 non-null  int64
 66  BROADCAST_BADGE                     153012 non-null  int64
 67  UPDATE_SHORTCUT                     153012 non-null  int64
 68  USE_CREDENTIALS                     153012 non-null  int64
 69  BLUETOOTH_ADMIN                     153012 non-null  int64
 70  UPDATE_COUNT                        153012 non-null  int64
 71  UPDATE_BADGE                        153012 non-null  int64
 72  USE_FINGERPRINT                     153012 non-null  int64
 73  CHANGE_BADGE                        153012 non-null  int64
 74  GET_TASKS                           153012 non-null  int64
 75  PROVIDER_INSERT_BADGE               153012 non-null  int64
 76  READ_APP_BADGE                      153012 non-null  int64
 77  Permissions_warning_count           153012 non-null  int64
 78  Developer                           124870 non-null  object
 79  Country                             119230 non-null  object
dtypes: int64(74), object(6)
memory usage: 93.4+ MB
None
```

Out[4]:

	Id	Name	Tracker_count	Permissions_count	Version	Downloads	Analysis_date	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	...	UPDATE_COUNT	UPDATE_BADGE	USE_FINGERPRINT	CHANGE_BADGE	GET_TASKS	PROVIDER_INSERT_BADGE	READ_APP_BADGE	Pern
0	1	Tan	3	9	5.7.0	100,000+	12-11-2017	0	1	0	...	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	5,000,000+	12-11-2017	0	0	0	...	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	100,000+	12-11-2017	0	0	0	...	0	0	0	0	0	0	0	
153009	153371	Jelbi	2	13	3.21.0	NaN	8-11-2020	1	0	0	...	0	0	0	0	0	0	0	
153010	153372	kvgOF Hopper	2	8	2.8.11	NaN	8-11-2020	1	0	1	...	0	0	0	0	0	0	0	
153011	153373	Microsoft SwiftKey Keyboard	3	11	7.6.7.5	NaN	8-11-2020	0	0	0	...	0	0	0	0	0	0	0	

6 rows × 80 columns

2.3. Integración de datos

2.3.1. Obtención, generación del Data Frame y exploración del dataset de integración.

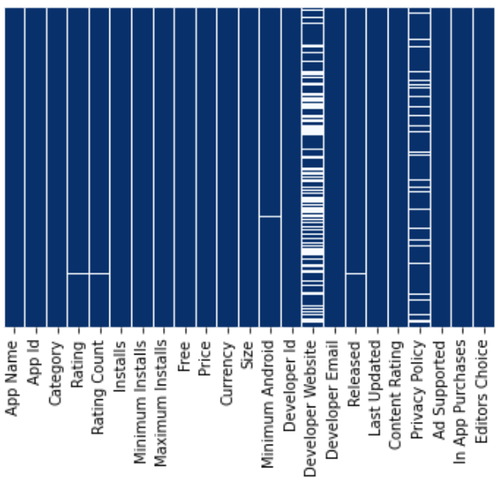
Existen, en repositorios como Kaggle, muchos conjuntos de datos acerca de aplicaciones de Android. Disponer de esta información nos puede servir para **integrar verticalmente** información extra en nuestro dataset que podría ser útil.

Utilizamos el dataset **Google Play Store Apps** *(Google Play Store App data of 1.1 Million applications.)* de Gautham P. disponible en la plataforma **Kaggle** en la url https://www.kaggle.com/gauthamp10/google-playstore-apps donde se podrá encontrar la descripción detallada del dataset y su enlace de descarga.

Una vez descargado el dataset *Google-Playstore.csv* y descomprimido en la carpeta *data* del directorio de trabajo de Python, procedemos a su cargarlo como variable Python y realizar una sencilla exploración.

In [5]:
```
###Carga del fichero del dataset de integración Google-Playstore.csv y exploración del contenido.###

#Importar librerías necesarias
import pandas as pd
pd.options.display.max_columns = None
import seaborn as sns

#Carga del fichero
df_integracion = pd.read_csv('data/Google-Playstore.csv', sep = ',', quotechar = '"', header = 0)

print('Descripción de las variables. Nombre, nulos y tipo de datos\n')
df_integracion.info()
print('\nInformación de unicidad por el atributo App Name\n')
print(df_integracion['App Name'].value_counts().sort_values(ascending=False).head(5))
print('\nDistribución de los valores nulos en los atributos')
sns.heatmap(df_integracion.isnull(), yticklabels = False, cbar = False, cmap = 'Blues_r')
plt.show()
```

```
Descripción de las variables. Nombre, nulos y tipo de datos

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1118136 entries, 0 to 1118135
Data columns (total 23 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   App Name          1118135 non-null  object 
 1   App Id            1118136 non-null  object 
 2   Category          1118133 non-null  object 
 3   Rating            1111286 non-null  float64
 4   Rating Count      1111286 non-null  float64
 5   Installs          1117975 non-null  object 
 6   Minimum Installs  1117975 non-null  float64
 7   Maximum Installs  1118136 non-null  int64  
 8   Free              1118136 non-null  bool   
 9   Price             1118136 non-null  float64
 10  Currency          1117975 non-null  object 
 11  Size              1118136 non-null  object 
 12  Minimum Android   1116123 non-null  object 
 13  Developer Id      1118134 non-null  object 
 14  Developer Website  703770 non-null   object 
 15  Developer Email   1118114 non-null  object 
 16  Released          1110406 non-null  object 
 17  Last Updated      1118136 non-null  object 
 18  Content Rating    1118136 non-null  object 
 19  Privacy Policy    964612 non-null   object 
 20  Ad Supported      1118136 non-null  bool   
 21  In App Purchases  1118136 non-null  bool   
 22  Editors Choice    1118136 non-null  bool   
dtypes: bool(4), float64(4), int64(1), object(14)
memory usage: 166.3+ MB

Información de unicidad por el atributo App Name

Tic Tac Toe       143
Flashlight        140
Age Calculator    122
Solitaire         119
BMI Calculator    109
Name: App Name, dtype: int64

Distribución de los valores nulos en los atributos
```



Observamos:

- El dataset tiene 1.118.136 registros con 23 atributos.
- Existen varias ocurrencias de aplicaciones distintas pero que comparten el mismo nombre.
- El dataset contiene elementos vacíos.
  - Podemos ver en el gráfico la distribución de estos nulos en las distintas variables. Las más afectadas *Developer Website* y *Privacy Policy*

In [6]:
```
#Visualización de algunos datos#
df_integracion.head(5)
```

Out[6]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Currency	Size	Minimum Android	Developer Id	Developer Website	Developer Email	Released	Last Updated	Content Rating	
0	HTTrack Website Copier	com.httrack.android	Communication	3.6000	2848.0000	100,000+	100000.0000	351560	True	0.0000	USD	2.7M	2.3 and up	Xavier Roche	http://www.httrack.com/	roche+android@httrack.com	Aug 12, 2013	May 20, 2017	Everyone	http://andr
1	World War 2: Offline Strategy	com.skizze.wwii	Strategy	4.3000	17297.0000	1,000,000+	1000000.0000	2161778	True	0.0000	USD	86M	5.1 and up	Skizze Games	http://stereo7.com/	Skizze.Games@gmail.com	Jul 19, 2018	Nov 26, 2020	Everyone 10+	
2	WPSApp	com.themausoft.wpsapp	Tools	4.2000	488639.0000	50,000,000+	50000000.0000	79304739	True	0.0000	USD	5.8M	4.1 and up	TheMauSoft	http://www.themausoft.com	wpsapp.app@gmail.com	Mar 7, 2016	Oct 21, 2020	Everyone	https://sites.go
3	OfficeSuite - Office, PDF, Word, Excel, PowerP...	com.mobisystems.office	Business	4.2000	1224420.0000	100,000,000+	100000000.0000	163660067	True	0.0000	USD	59M	4.4 and up	MobiSystems	http://www.mobisystems.com	support-officesuite-android@mobisystems.com	Dec 22, 2011	Nov 23, 2020	Everyone	http://www
4	Loud Player Free	com.arthelion.loudplayer	Music & Audio	4.2000	665.0000	50,000+	50000.0000	73463	True	0.0000	USD	29M	5.0 and up	Arthelion92	http://www.arthelion.com	arthelion92@gmail.com	Sep 24, 2016	Nov 22, 2020	Everyone	http://www.art

In [7]:
```
#Visualización de la entrada correspondiente al ejemplo del dataset exodusNoIcon con Id = 1 de nombre "Tan"

df_integracion[df_integracion['App Name'] == 'Tan']
```

Out[7]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Currency	Size	Minimum Android	Developer Id	Developer Website	Developer Email	Released	Last Updated	Content Rating	Privacy Policy	Ad Supported	In App Purchases	Editors Choice
275074	Tan	com.semitan.tan	Maps & Navigation	2.4000	3385.0000	100,000+	100000.0000	484633	True	0.0000	USD	35M	5.0 and up	Réseau Tan	http://www.tan.fr	webmestre@tan.fr	Mar 12, 2012	Jun 17, 2020	Everyone	https://www.tan.fr/charte-des-donnees-personne...	False	False	False

Podemos obtener conclusiones de cara a la integración de ambos datasets:

- El dataset *Google Play Store Apps* para integrar tiene muchos más registros que el dataset de privacidad *exodusNoIcon*.
- Hay varios registros de aplicaciones con el mismo nombre.
- El atributo nexo entre ambos datasets, es *App Name* en el dataset de integración y *Name* en el dataset integrado.
  - **Deberemos prestar cuidado en el caso de registros de aplicaciones que comparten nombre.**
  - También tenemos en común el atributo *Installs* del dataset de integración con *Downloads* del dataset integrado.

In [8]:
```
#Visualización de registros de aplicaciones con el mismo nombre

df_integracion[df_integracion['App Name'] == 'Tic Tac Toe'].head(5)
```

Out[8]:

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	Currency	Size	Minimum Android	Developer Id	Developer Website	Developer Email	Released	Last Updated	Content Rating	
10944	Tic Tac Toe	aexyn.tictactoe.zero.kaata	Puzzle	4.9000	47.0000	1,000+	1000.0000	2823	True	0.0000	USD	3.8M	4.0.3 and up	Stupefying Labs	NaN	stupefyinglabs@gmail.com	Nov 30, 2016	Sep 07, 2017	Everyone	https://stupefyinglabs.h
14026	Tic Tac Toe	com.bigbrainkraken.tictactoe	Board	4.0000	22737.0000	1,000,000+	1000000.0000	1892227	True	0.0000	USD	8.8M	4.1 and up	The Angry Kraken	http://www.theangrykraken.com	theangrykraken@gmail.com	Jan 26, 2016	Oct 16, 2020	Everyone	http://www.th
21170	Tic Tac Toe	com.inglesdivino.tictactoe	Puzzle	3.6000	25.0000	1,000+	1000.0000	2776	True	0.0000	USD	1.7M	4.0 and up	inglesdivino	http://www.inglesdivino.com/	fabril.and@hotmail.com	Dec 11, 2013	Dec 15, 2018	Everyone	http://www.inglesdivi
40148	Tic Tac Toe	com.ezeontech.tic_tac_toe	Puzzle	4.9000	38.0000	50+	50.0000	83	True	0.0000	USD	3.6M	4.0.3 and up	EasyApp Developer	http://www.ezeontech.com	dev.genius21@gmail.com	Sep 23, 2019	Oct 01, 2019	Everyone	
43696	Tic Tac Toe	com.pinkpointer.tictactoe	Board	4.1000	20892.0000	1,000,000+	1000000.0000	2144843	True	0.0000	USD	13M	4.4 and up	Pink Pointer	http://www.pinkpointer.com	contact@pinkpointer.com	Aug 11, 2014	Jul 08, 2020	Everyone	http://www.pinl

Se puede ver que existe un atributo único que distingue a cualquier aplicación *App Id*, pero no disponemos de esta información en el dataset de *Exodus*

Realizar una **integración vertical** a partir de la tupla (*App Name*), atributos de los que sí disponemos de información en el dataset *Exodus* para especificar la unión de los datasets, da la sensación que va a provocar una **inconsistencia** en la información final, al poder atribuir atributos extra a aplicaciones que no corresponden.

Es vital buscar una solución para integrar un atributo identificador único en el dataset *Exodus* para llevar a cabo la integración.

2.3.1.1. Integración horizontal intermedia del identificador único de aplicaciones en el dataset *Exodus* (Nuevo Web Scraping)

Desgraciadamente en la primera práctica, por desconocimiento de cómo se identifican las aplicaciones Android, no incluí información de la uri que distingue unívocamente cada aplicación de Android. Esta información sí está presente en las páginas web que describen las aplicaciones del sitio *Exodus* y es utilizada como clave para aglutinar las distintas versiones de la misma aplicación y para redirigir al usuario a su página de información y descarga en la propia web de *Google Play*

Tomando como base el rastreador creado en la primera práctica, se decide generar un segundo rastreador más sencillo que extraiga, para cada identificador de aplicación de *Exodus* el identificador propio de cada aplicación analizada.

El rastreador *exodusAppId*:

- Genera un fichero en formato CSV *exodusAppId.csv* con la siguiente estructura:
  - Una fila por página de análisis de aplicación con los siguientes atributos: *idExodus*, *idAplicacion* donde:
    - *idExodus* es el identificador de análisis de la aplicación y coincide con el atributo *Id* del dataset *Exodus* objetivo de limpieza y análisis de la práctica.
    - *idAplicacion* es el identificador de aplicaciones de Android, en formato URL *(Ejemplo: com.semitan.tan)*
- Se establece como inicio el análisis 1 y como fin el anális 153.373, que son el primero y el último que hemos identificado al cargar el dataset.
- Por si hay que realizar distintas sesiones de rastreo, se evalúa si existe el fichero CSV previamente y se extraen los identificadores de página pendientes de tratar.
- Se conforma la dirección web y para cada una:
  - A través de BeautifulSoup se extrae la información que se encuentra en el tag de hiperenlace a la información y descarga de la app en la Google Play Store

```
def rastrearHtml(html):
    appId = 'na'
    soup = BeautifulSoup(html, features='lxml')

    #Id de la aplicación
    try:
        tag = soup.find('a', {'class': 'link main-link'})
        appId = tag['href'][tag['href'].find('=') + 1:]
    except Exception as e:
        appId = 'na'

    return appId
```

  - Se crea una lista por cada uno de los atributos del nuevo dataset y se añade al final del fichero.

Cargamos el nuevo dataset y realizamos una sencilla exploración

In [9]:
```
###Carga del fichero del dataset de integración Google-Playstore.csv y exploración del contenido.###

#Carga del fichero
df_integracion_appId = pd.read_csv('data/exodusAppId.csv', sep = ',', header = None, names = ['Id','App Id'])

print('Descripción de las variables. Nombre, nulos y tipo de datos\n')
df_integracion_appId.info()
print('\nVisualizar los cinco primeros registros\n')
df_integracion_appId.head(5)
```

```
Descripción de las variables. Nombre, nulos y tipo de datos

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 152900 entries, 0 to 152899
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   Id      152900 non-null  int64 
 1   App Id  152900 non-null  object
dtypes: int64(1), object(1)
memory usage: 2.3+ MB

Visualizar los cinco primeros registros
```

Out[9]:

	Id	App Id
0	1	com.semitan.tan
1	2	com.allocine.androidapp
2	3	com.frostnerd.dnschanger
3	4	com.fullsix.android.labanquepostale.accountaccess
4	5	fr.axa.monaxa

2.3.1.2. Integración horizontal del dataset de identificadores intermedio con el dataset *Exodus*

Integramos ahora el DataFrame *df_exodus* con el Data Frame obtenido *df_integracion_appId*, a partir de la clave conjunta *Id*.

Utilizamos un left outer join, dejando como Data Frame base *df_exodus* por si no se hubiera obtenido correctamente el dato en el rastreo del último dataset *df_integracion_appId* se consigne la información del atributo a nulo *na*.

Disponemos al finalizar una instrucción para contar este número de **elementos vacíos** y visualizamos el principio y el final del Data Frame con la nueva información incorporada.

In [10]:
```
df_exodus = pd.merge(df_exodus, df_integracion_appId, how = 'left', on = ['Id'])

print('Número de registros donde no hemos podido encontrar el identificador de aplicación con el nuevo dataset rastreado: ' + str(df_exodus['App Id'].isna().sum()) + '\n')
df_exodus.head(3).append(df_exodus.tail(3))
```

```
Número de registros donde no hemos podido encontrar el identificador de aplicación con el nuevo dataset rastreado: 112
```

Out[10]:

	Id	Name	Tracker_count	Permissions_count	Version	Downloads	Analysis_date	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK SHARE	FACEBOOK ANALYTICS	GOOGLE TAG MANAGER	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	U
0	1	Tan	3	9	5.7.0	100,000+	12-11-2017	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	5,000,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	100,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153009	153371	Jelbi	2	13	3.21.0	NaN	8-11-2020	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153010	153372	kvgOF Hopper	2	8	2.8.11	NaN	8-11-2020	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
153011	153373	Microsoft SwiftKey Keyboard	3	11	7.6.7.5	NaN	8-11-2020	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	

2.3.2. Integración vertical final del dataset de información de aplicaciones *Google Play Store Apps* con el dataset *Exodus*

Finalmente, teniendo ya como nexo de unión el atributo *App Id*, podemos unir de manera más eficiente el dataset original de estudio con el incorporado desde el repositorio Kaggle.

De nuevo la estrategia consistirá en un left outer join tomando como base el dataset de estudio *Exodus* y completando con la información que consiga "casar" del dataset de aplicaciones de la *Google Play Store Apps*.

En este caso, tenemos que utilizar un identificador distinto para contar el número de registros con **elementos vacíos** que no hemos podido unir. Esto es debido a que en la integración anterior, el atributo nexo *App Id* ya tiene valores *na* en los registros donde no disponíamos de información de la URL de identificación, así que utilizando de nuevo este último atributo, obtendríamos la misma cuenta.
Es preciso utilizar un atributo del dataset de integración que sepamos que no tiene valores nulos como *App Name*. Esto es así puesto que, si no se produce la unión, la operación de left outer join lo consignará a *na*. Los valores así obtenidos, incluirán los 112 registros anteriores que ya contienen el valor indeterminado *na* puesto que la operación de merge no puede asumir que *na* en *Exodus* = *alguna cosa* en *Google Play Store Apps*.

In [11]:
```
df_exodus = pd.merge(df_exodus, df_integracion, how = 'left', on = ['App Id'])

print('Número de registros donde no hemos podido consolidar el identificador de aplicación con el dataset de integración: ' + str(df_exodus['App Name'].isna().sum()) + '\n')
df_exodus.head(3).append(df_exodus.tail(3))
```

```
Número de registros donde no hemos podido consolidar el identificador de aplicación con el dataset de integración: 47499
```

Out[11]:

	Id	Name	Tracker_count	Permissions_count	Version	Downloads	Analysis_date	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK SHARE	FACEBOOK ANALYTICS	GOOGLE TAG MANAGER	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	U
0	1	Tan	3	9	5.7.0	100,000+	12-11-2017	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	5,000,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	100,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153009	153371	Jelbi	2	13	3.21.0	NaN	8-11-2020	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153010	153372	kvgOF Hopper	2	8	2.8.11	NaN	8-11-2020	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
153011	153373	Microsoft SwiftKey Keyboard	3	11	7.6.7.5	NaN	8-11-2020	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	

2.3.2.1. Intento para minimizar la cantidad de elementos perdidos en la integración

Puesto que no hemos conseguido establecer un nexo de unión concreto para casi 50.000 registros (un tercio del total) mediante el dato que recoge la uri de identificación de la aplicación, vamos a tratar de juntar estos elementos mediante el nexo *nombre*. Aunque no es un atributo idóneo como nexo de unión entre fuentes de datos, como habíamos estudiado en el primer apartado de esta sección 2.3, si consigue rescatarnos algunos registros de este bloque de **registros con elementos perdidos**, podría ser interesante.

En el estudio de integración anterior, también vimos la problemática adicional de que es posible que dos o más aplicaciones compartan nombre. Este hándicap obliga a realizar un tratamiento adicional para eliminar los duplicados. Los pasos para realizar esta integración serán los siguientes:

- Extraer del Data Frame del paso anterior el subconjunto de registros de *Exodus* que no se había podido integrar con *Google Play Store Apps* a partir de la uri de identificación, y quitar las columnas de este último para volverlas a incluir tras la realización del proceso.
- Normalizar los nombres de aplicación en las columnas de ambos Data Frames para aumentar las posibilidades de encontrar nombres coincidentes
- Volver a realizar el left outer join, esta vez sobre al atributo del nombre de la aplicación normalizado.
  - Tenemos en consideración que de esta manera, se van a crear tantos registros como aplicaciones del dataset de integración coincidan en nombre.
- Puede llevar a pérdida de calidad en la integracion, pero podríamos asumir que, de las n posibles aplicaciones, siempre escojamos únicamente la primera de ellas.
  - De esta manera, asumimos un error en las aplicaciones que se llamen también igual en el Data Frame *Exodus* puesto que van a tener los mismos valores en los atributos heredados del Data Frame integrado *Google Play Store Apps*
- Calcular los registros que hemos podido integrar en esta tanda
- Obtener del Data Frame únicamente aquellos registros que se han integrado.

In [12]:
```
#Creamos un Data Frame con los registros que no hemos podido unir en la integración vertical
df_exodus_na = df_exodus[df_exodus['App Name'].isna()]

#Quitamos las columnas de datos del dataset de integración.
col_eliminar = list(df_integracion.keys())
df_exodus_na = df_exodus_na.drop(col_eliminar, axis = 1)
```

```
#Técnica específica para realizar la unión
import unicodedata
def normalizar_unicode(s):
    '''
    Esta función devuelve, para un carácter dado, la normalización NFD de unicode para evitar tildes o carácteres de otros idiomas.
    '''
    return ''.join(c for c in unicodedata.normalize('NFD', s) if unicodedata.category(c) != 'Mn')

#Normalizar las cadenas de texto quitando carácteres unicode y transformando a mayúculas el atributo nombre que es por el que vamos a unir ambos datasets
df_exodus_na['App Name'] = df_exodus_na['Name'].apply(lambda x: normalizar_unicode(x).upper().replace(' ',''))
df_integracion['App Name'] = df_integracion['App Name'].apply(lambda x: normalizar_unicode(str(x)).upper().replace(' ', ''))

#Se hace la left join por nombre, pero de esta manera una aplicación puede hacer match con varias del dataset de integración porque los nombres de aplicación puden repetirse en
#diferentes aplicaciones.
df_exodus_na_aux = pd.merge(df_exodus_na, df_integracion, how = 'left', on = 'App Name')
#La siguiente línea de código, toma el primer match producido.
df_exodus_na = df_exodus_na_aux.groupby(['Id']).first().reset_index()

print('Número de registros donde no hemos podido consolidar el nombre de aplicación con el dataset de integración: ' + str(df_exodus_na['App Id'].isna().sum()) + '\n')

#Eliminamos del dataframe los registros para los que no hemos encontrado igual en el dataset de integración
df_exodus_na.dropna(subset=['App Id'], inplace = True)

print('Número de registros que hemos podido integrar con el segundo intento: ' + str(df_exodus_na.shape[0]))
```

```
Número de registros donde no hemos podido consolidar el nombre de aplicación con el dataset de integración: 42646

Número de registros que hemos podido integrar con el segundo intento: 4853
```

Con esta técnica, hemos conseguido integrar información del dataset *Google Play Store Apps* a 4.853 registros más en el dataset original *Exodus*.

Obtenemos así en el Data Frame *df_exodus* la integración final.

In [13]:
```
df_exodus.drop(df_exodus[df_exodus['Id'].isin(df_exodus_na['Id'])].index, inplace = True)
df_exodus = df_exodus.append(df_exodus_na, ignore_index = True, sort = False)
df_exodus.head(3).append(df_exodus.tail(3))
```

Out[13]:

	Id	Name	Tracker_count	Permissions_count	Version	Downloads	Analysis_date	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK SHARE	FACEBOOK ANALYTICS	GOOGLE TAG MANAGER	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UI
0	1	Tan	3	9	5.7.0	100,000+	12-11-2017	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	5,000,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	100,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153009	153115	Gallery	0	6	1.3.1	NaN	6-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153010	153270	Nebula	0	4	0.0.38	NaN	7-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153011	153354	Suntimes	0	8	0.13.2	NaN	8-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

2.4. Selección

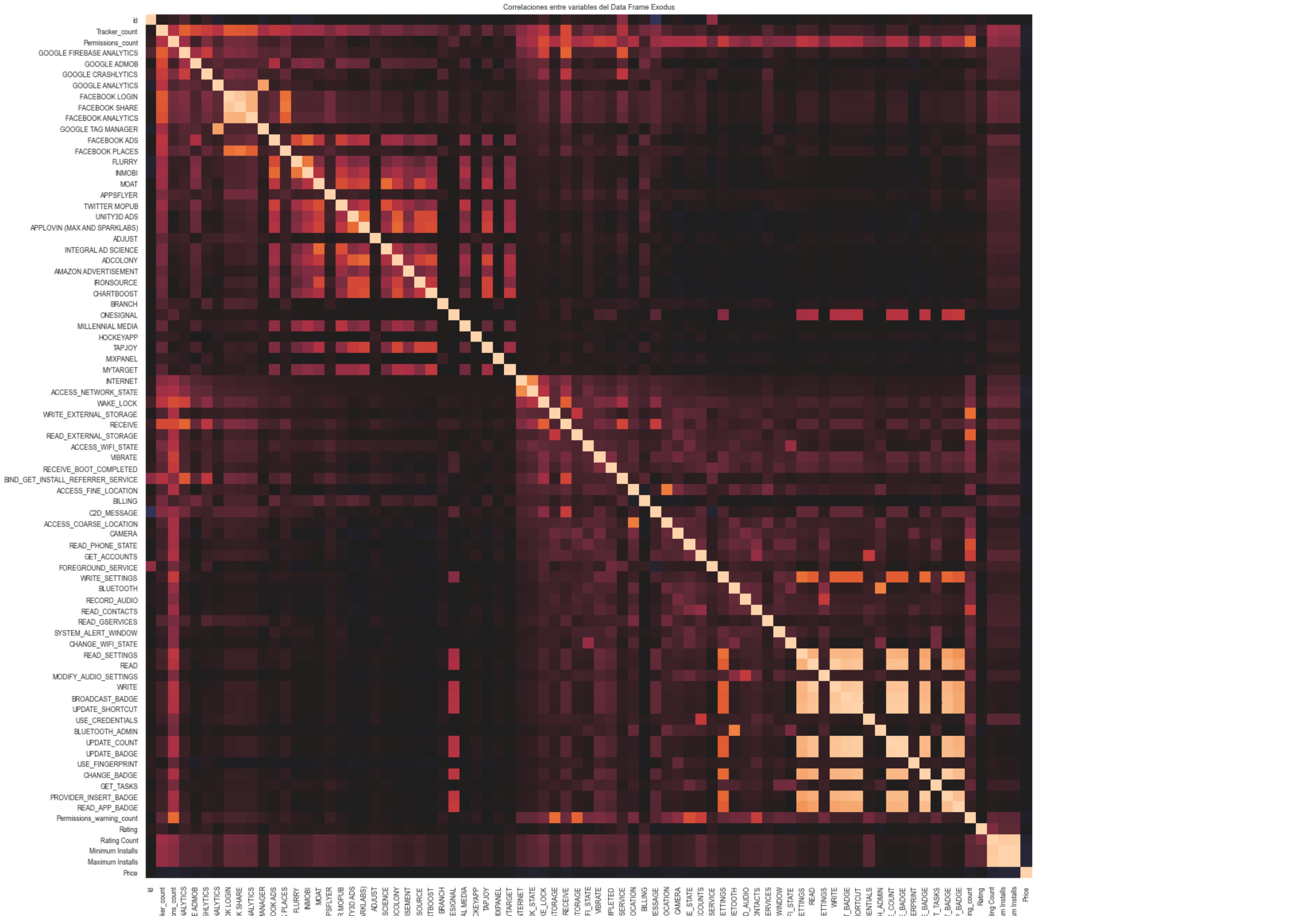
Haciendo una repaso al a estructura del nuevo Data Frame integrado, observamos que disponemos de 153.012 registros con 103 atributos, de los cuales 79 son numéricos y 24 categóricos. Teniendo en cuenta la existencia de nulos puesto que no han sido tratados.

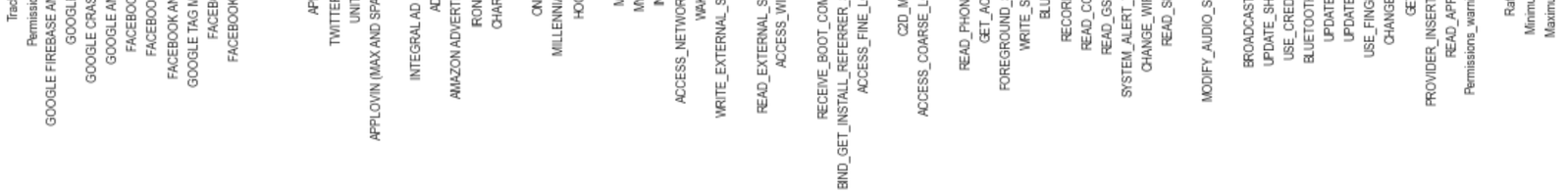
2.4.1 Exploración

Una manera de filtrar atributos es realizar una exploración de correlación entre atributos numéricos para poder descartar aquellos que estén fuertemente correlados con otra variable que no van a aportar variabilidad para los análisis posteriores de los datos.

Utilizo la función de correlación de pandas, la cual ignora el cálculo cuando la componente es indeterminada, con el metodo de Spearman, porque no hemos analizado todavía la distribución de los datos y es muy probable que no sigan una distribución normal. P ara analizar los resultados obtenidos, al ser una matriz simétrica, recorro únicamente los elementos encima de la diagonal y obtengo un listado que discrimina aquellos pares de variables donde su correlación, en valor absoluto, sea superior a *0,8*.

In [14]:
```
plt.figure(figsize=(20,20))
sns.set(font_scale=0.75)
sns.heatmap(df_exodus.corr(method = 'spearman'), center = 0, cbar = False)
plt.title('Correlaciones entre variables del Data Frame Exodus')
plt.show()
```





In [15]:
```
mtx_corr = df_exodus.corr(method = 'spearman')

for i in range(0, 79):
    for j in range ((i + 1), 79): #Solamente considerar la diagonal superior.
        if abs(mtx_corr.iloc[i,j]) >= 0.8:
            print('Detectada una alta correlación de ' + str(round(mtx_corr.iloc[i,j], 2)) + ' entre ' + mtx_corr.columns[i] + ' y ' + mtx_corr.columns[j])
```

```
Detectada una alta correlación de 0.84 entre GOOGLE ANALYTICS y GOOGLE TAG MANAGER
Detectada una alta correlación de 0.96 entre FACEBOOK LOGIN y FACEBOOK SHARE
Detectada una alta correlación de 0.89 entre FACEBOOK LOGIN y FACEBOOK ANALYTICS
Detectada una alta correlación de 0.89 entre FACEBOOK SHARE y FACEBOOK ANALYTICS
Detectada una alta correlación de 0.89 entre READ_SETTINGS y READ
Detectada una alta correlación de 0.91 entre READ_SETTINGS y WRITE
Detectada una alta correlación de 0.9 entre READ_SETTINGS y BROADCAST_BADGE
Detectada una alta correlación de 0.92 entre READ_SETTINGS y UPDATE_SHORTCUT
Detectada una alta correlación de 0.89 entre READ_SETTINGS y UPDATE_COUNT
Detectada una alta correlación de 0.89 entre READ_SETTINGS y UPDATE_BADGE
Detectada una alta correlación de 0.86 entre READ_SETTINGS y CHANGE_BADGE
Detectada una alta correlación de 0.85 entre READ_SETTINGS y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.81 entre READ_SETTINGS y READ_APP_BADGE
Detectada una alta correlación de 0.97 entre READ y WRITE
Detectada una alta correlación de 0.94 entre READ y BROADCAST_BADGE
Detectada una alta correlación de 0.95 entre READ y UPDATE_SHORTCUT
Detectada una alta correlación de 0.93 entre READ y UPDATE_COUNT
Detectada una alta correlación de 0.93 entre READ y UPDATE_BADGE
Detectada una alta correlación de 0.89 entre READ y CHANGE_BADGE
Detectada una alta correlación de 0.89 entre READ y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.84 entre READ y READ_APP_BADGE
Detectada una alta correlación de 0.95 entre WRITE y BROADCAST_BADGE
Detectada una alta correlación de 0.96 entre WRITE y UPDATE_SHORTCUT
Detectada una alta correlación de 0.95 entre WRITE y UPDATE_COUNT
Detectada una alta correlación de 0.95 entre WRITE y UPDATE_BADGE
Detectada una alta correlación de 0.9 entre WRITE y CHANGE_BADGE
Detectada una alta correlación de 0.9 entre WRITE y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.86 entre WRITE y READ_APP_BADGE
Detectada una alta correlación de 0.98 entre BROADCAST_BADGE y UPDATE_SHORTCUT
Detectada una alta correlación de 0.96 entre BROADCAST_BADGE y UPDATE_COUNT
Detectada una alta correlación de 0.96 entre BROADCAST_BADGE y UPDATE_BADGE
Detectada una alta correlación de 0.91 entre BROADCAST_BADGE y CHANGE_BADGE
Detectada una alta correlación de 0.92 entre BROADCAST_BADGE y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.87 entre BROADCAST_BADGE y READ_APP_BADGE
Detectada una alta correlación de 0.97 entre UPDATE_SHORTCUT y UPDATE_COUNT
Detectada una alta correlación de 0.97 entre UPDATE_SHORTCUT y UPDATE_BADGE
Detectada una alta correlación de 0.92 entre UPDATE_SHORTCUT y CHANGE_BADGE
Detectada una alta correlación de 0.92 entre UPDATE_SHORTCUT y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.87 entre UPDATE_SHORTCUT y READ_APP_BADGE
Detectada una alta correlación de 0.99 entre UPDATE_COUNT y UPDATE_BADGE
Detectada una alta correlación de 0.91 entre UPDATE_COUNT y CHANGE_BADGE
Detectada una alta correlación de 0.92 entre UPDATE_COUNT y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.89 entre UPDATE_COUNT y READ_APP_BADGE
Detectada una alta correlación de 0.91 entre UPDATE_BADGE y CHANGE_BADGE
Detectada una alta correlación de 0.92 entre UPDATE_BADGE y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.9 entre UPDATE_BADGE y READ_APP_BADGE
Detectada una alta correlación de 0.97 entre CHANGE_BADGE y PROVIDER_INSERT_BADGE
Detectada una alta correlación de 0.93 entre CHANGE_BADGE y READ_APP_BADGE
Detectada una alta correlación de 0.94 entre PROVIDER_INSERT_BADGE y READ_APP_BADGE
Detectada una alta correlación de 0.96 entre Rating Count y Minimum Installs
Detectada una alta correlación de 0.97 entre Rating Count y Maximum Installs
Detectada una alta correlación de 0.99 entre Minimum Installs y Maximum Installs
```

Aparecen 52 atributos correlacionados que habrá que examinar previamente para poder descartar puesto que en algunos casos, "*correlación no implica causalidad*".

- Sobre los rastreadores GOOGLE ANALYTICS y GOOGLE TAG MANAGER (0,84), sí podemos asumir que si se ha instalado *analytics* es muy probable que se haya instalado *tag manager* por lo que nos quedaremos con el primero.
- Sobre los rastreadores de FACEBOOK (LOGIN, SHARE y ANALYTICS) existe también correlación.
  - Los dos primeros están más fuertemente correlados (0,96) porque cuando se utiliza la utilidad de Facebook para hacer *login* se comparten *share* ciertos datos de la propia cuenta a la aplicación. De éstos nos quedamos con el primero que es más genérico.
  - No ocurre lo mismo, la con el rastreador de analítica (0,89) porque en principio son funcionalidades distintas. Por esto, no considero filtrar este atributo.
- Hay una fuerte correlación entre los permisos READ y WRITE, pero son conceptos distintos que convendría no fusionar.
- Se distinguen una serie de permisos muy relacionados con los de lectura y escritura, y por extensión, entre ellos mismos. Son BROADCAST_BADGE, UPDATE_SHORTCUT, UPDATE_COUNT, UPDATE_BADGE, CHANGE_BADGE, PROVIDER_INSERT_BADGE y READ_APP_BADGE.

- Los permisos de actualización de distintos elementos del teléfono, están fuertemente correlados con el permiso de escritura WRITE (0,95 difusión, 0,96 actualizar_icono, 0,93 a 0,84 permisos relacionados con actualización de notificaciones entrantes) que en cierta manera podría asumir.
- También existe correlacción entre READ y READ_SETTINGS (0,89) que aunque son permisos para lectura, al estar orientados en aspectos distintos del teléfono, creo que es mejor mantener este último para lectura exclusiva de la configuración del teléfono.

- Acerca de la correlación entre Minimum Installs y Maximum Installs (0,99), si observamos la anterior exposición del Data Frame, vemos que Minimum Installs es un intento de discretización numérica por la que, en Maximum Installs se encuentra el número exacto de terminales donde la aplicación se encuentra instalada, y Minimum Installs, es una categoría suelo en múltiplos de 5 de este número de instalaciones. Por ejemplo, una aplicación con 484.633 instalaciones, entra en la categoría de 100.000 como mínimo. De esta manera, considero que utilizaré esta discretización por Minimum Installs que ofrece variabilidad y también un atributo más robusto y menos granular por el que agrupar aplicaciones.
  - Con ambos atributos, se relaciona Rating Count (0,96 con Minimum Installs), se intuye que puede haber un porcentaje más o menos fiel de personas que suelen calificar aplicaciones y de ahí esta correlación, pero el dato que representa es distinto por lo que no la descartaré en favor del número de instalaciones.

In [16]:
```
df_exodus.drop(columns = ['GOOGLE TAG MANAGER','FACEBOOK SHARE','BROADCAST_BADGE','UPDATE_SHORTCUT','UPDATE_COUNT','UPDATE_BADGE','CHANGE_BADGE','PROVIDER_INSERT_BADGE','READ_APP_BADGE',
                          'Maximum Installs'], inplace = True)
df_exodus.head(3).append(df_exodus.tail(3))
```

Out[16]:

	Id	Name	Tracker_count	Permissions_count	Version	Downloads	Analysis_date	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	A
0	1	Tan	3	9	5.7.0	100,000+	12-11-2017	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	5,000,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	100,000+	12-11-2017	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153009	153115	Gallery	0	6	1.3.1	NaN	6-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153010	153270	Nebula	0	4	0.0.38	NaN	7-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
153011	153354	Suntimes	0	8	0.13.2	NaN	8-11-2020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

2.4.2 Creación de nuevas variables

2.4.2.1 Financiación (Free, Add Supported, App Purchases)

Otra manera de seleccionar atributos, es crear nuevas variables a partir de atributos existentes que expliquen éstas en conjunto.

Casi desde la aparición de los smartphones, se generó una corriente en las sociedad donde consideraba que no tenía por qué pagar por software adicional en su terminal. Esto llevó a los desarrolladores a buscar otras corrientes de financiación que encontraron en la inclusión de anuncios o compra de mejoras o funcionalidades dentro de una versión gratuita de la aplicación.

Es un dato relevante porque el término gratuito puede ser equívoco para el usuario y con este reclamo, se puede aprovechar para la inclusión de más rastreadores o permisos por los que recolectar datos del usuario como otra manera de financiación.

En el conjunto de datos, tenemos las variables binarias *Free*, *Ad Supported* y *In App Purchases* que podríamos utilizar para la creación de una nueva variable cuyos datos sirvieran de esquema de clasificación de la financiación de la aplicación bajo las siguientes condiciones:

```
    * Si Free = True AND Ad Supported = False AND In App Purchaes = False -> Gratis
    * Si Free = True AND Ad Supported = True AND In App Purchases = False -> Anuncios
    * Si Free = True AND Ad Supported = False AND In App Purchases = True -> PagoInApp
    * Si Free = True AND Ad Supported = True AND In App Purchases = True -> AnunciosPagoInApp
    * Si Free = False AND Ad Supported = False AND In App Purchases = False -> Pago
    * Si Free = False AND (Ad Supported = True OR In App Purchases = True) -> Copago
```

In [17]:
```
def financiacion(gratis, anuncios, compras):
    if np.isnan(gratis) or np.isnan(anuncios) or np.isnan(compras): return np.nan #Si el dato es vacío, obviamos la categorización.
    if gratis == True:
        if anuncios == False and compras == False: return 'Gratis'
        if anuncios == True and compras == False: return 'Anuncios'
        if anuncios == False and compras == True: return 'PagoInApp'
        if anuncios == True and compras == True: return 'AnunciosPagoInApp'
    else:
        if anuncios == False and compras == False:
            return 'Pago'
        else:
```

```
            return 'Copago'

df_exodus['Financiacion'] = np.vectorize(financiacion)(df_exodus['Free'], df_exodus['Ad Supported'], df_exodus['In App Purchases'])
df_exodus['Financiacion'] = df_exodus['Financiacion'].replace('nan', np.nan) #Como la vectorización devuelve 'nan' como string, reemplazampos por el NaN de pandas

print('Data Frame Exodus: ' + str(df_exodus.shape[0]) + ' registros y ' + str(df_exodus.shape[1]) + ' atributos')
df_exodus.head(3).append(df_exodus.tail(3))[['Name', 'Free', 'Ad Supported', 'In App Purchases', 'Financiacion']]
```

```
Data Frame Exodus: 153012 registros y 94 atributos
```

Out[17]:

	Name	Free	Ad Supported	In App Purchases	Financiacion
0	Tan	True	False	False	Gratis
1	AlloCine	True	True	True	AnunciosPagoInApp
2	DNS Changer (No Root - IPv6 - 3G/wifi)	True	False	False	Gratis
153009	Gallery	True	True	True	AnunciosPagoInApp
153010	Nebula	True	False	False	Gratis
153011	Suntimes	True	True	False	Anuncios

In [18]:
```
#Eliminar los atributos dependientes
df_exodus.drop(columns = ['Free', 'Ad Supported', 'In App Purchases'], inplace = True)
```

2.4.2.2 Ratio valoración (Rating Count, Minimum Installs)

Atendiendo a la correlación de ambas variables, donde una persona no puede valorar una aplicación si, como mínimo, no se la ha instalado, se podría establecer un ratio entre ellas.

In [19]:
```
df_exodus['Ratio_valoracion'] = df_exodus['Rating Count']/df_exodus['Minimum Installs']
df_exodus.head(3).append(df_exodus.tail(3))[['Name', 'Rating Count', 'Minimum Installs', 'Ratio_valoracion']]
```

Out[19]:

	Name	Rating Count	Minimum Installs	Ratio_valoracion
0	Tan	3385.0000	100000.0000	0.0338
1	AlloCine	102977.0000	5000000.0000	0.0206
2	DNS Changer (No Root - IPv6 - 3G/wifi)	24645.0000	1000000.0000	0.0246
153009	Gallery	21466.0000	1000000.0000	0.0215
153010	Nebula	478.0000	50000.0000	0.0096
153011	Suntimes	38.0000	10000.0000	0.0038

3. Limpieza de los datos

3.1. Elementos vacíos

Para acometer esta sección de limpieza de los datos, correspondiente al tratamiento de elementos vacíos, realizamos primero el estudio de valores indeterminados actuales del Data Frame *Exodus*.

En primer lugar el número de registros donde alguno de sus elementos contiene un valor nulo.

In [20]:
```
print('Número de registros con algún atributo indeterminado: ' + str(df_exodus[df_exodus.isna().sum(axis = 1) > 0].shape[0]) + ' que representan un ' +
      str(round((df_exodus[df_exodus.isna().sum(axis = 1) > 0].shape[0] * 100)/ df_exodus.shape[0], 2)) + '% del total')
```

```
Número de registros con algún atributo indeterminado: 120006 que representan un 78.43% del total
```

El cálculo nos reseña un porcentaje relativamente bajo de registros completamente informados, alrededor de un 22% del conjunto de datos. De esta manera:

- No podríamos descartar directamente los registros con algún valor indeterminado porque supondría reducir demasiado el tamaño del conjunto de datos.
- Hay que analizar la distribución de nulos por atributos.

Observamos el número de nulos existentes por registro y su porcentaje respecto al total de registros

In [21]:
```
pd.DataFrame((df_exodus.isna().sum(axis = 1).value_counts(sort = False),
              round(df_exodus.isna().sum(axis = 1).value_counts(sort = False)*100/df_exodus.shape[0], 2)),
              index = ['Cuenta', 'Porcentaje']).transpose()
```

Out[21]:

	Cuenta	Porcentaje
0	33006.0000	21.5700

	Cuenta	Porcentaje
1	45512.0000	29.7400
2	20048.0000	13.1000
3	9187.0000	6.0000
4	1965.0000	1.2800
5	489.0000	0.3200
6	128.0000	0.0800
7	22.0000	0.0100
8	9.0000	0.0100
20	17433.0000	11.3900
21	17187.0000	11.2300
22	5980.0000	3.9100
23	1832.0000	1.2000
24	148.0000	0.1000
25	1.0000	0.0000
26	65.0000	0.0400

Podemos observar que la mayoría de registros (29,75%) de registros solamente tienen alguno de sus atributos nulo. El siguiente mayor porcentaje (13,10%) corresponde a los registros donde dos de sus atributos son desconocidos, y a partir de ahí el promedio de registros de tres a más atributos indeterminados va descendiendo.

No obstante, observamos que a partir de 19 atributos indeterminados, vuelve a haber un incremento importante de registros (22,62% en total) con este número o superior de atributos nulos. Estos 19 atributos indeterminados, corresponden a los del Data Frame de integración *Google Play Store Apps* en aquellas instancias donde no pudimos encontrar información coincidente con el Data Frame *Exodus*.

En este sentido, ya podemos tomar la primera decisión de tratamiento de valores nulos: `Considero que disponer de registros sin la información de integración no beneficia a los análisis que se pudieran realizar sobre el conjunto de datos. De esta forma, lo óptimo es descartar los registros que no hayan podido integrarse completamente.`

In [22]:
```
df_exodus = df_exodus[df_exodus.isna().sum(axis = 1) < 19].reset_index(drop = True)

print('Data Frame Exodus: ' + str(df_exodus.shape[0]) + ' registros y ' + str(df_exodus.shape[1]) + ' atributos')

pd.DataFrame((df_exodus.isna().sum(axis = 1).value_counts(sort = False),
              round(df_exodus.isna().sum(axis = 1).value_counts(sort = False)*100/df_exodus.shape[0], 2)),
              index = ['Cuenta', 'Porcentaje']).transpose()
```

```
Data Frame Exodus: 110366 registros y 92 atributos
```

Out[22]:

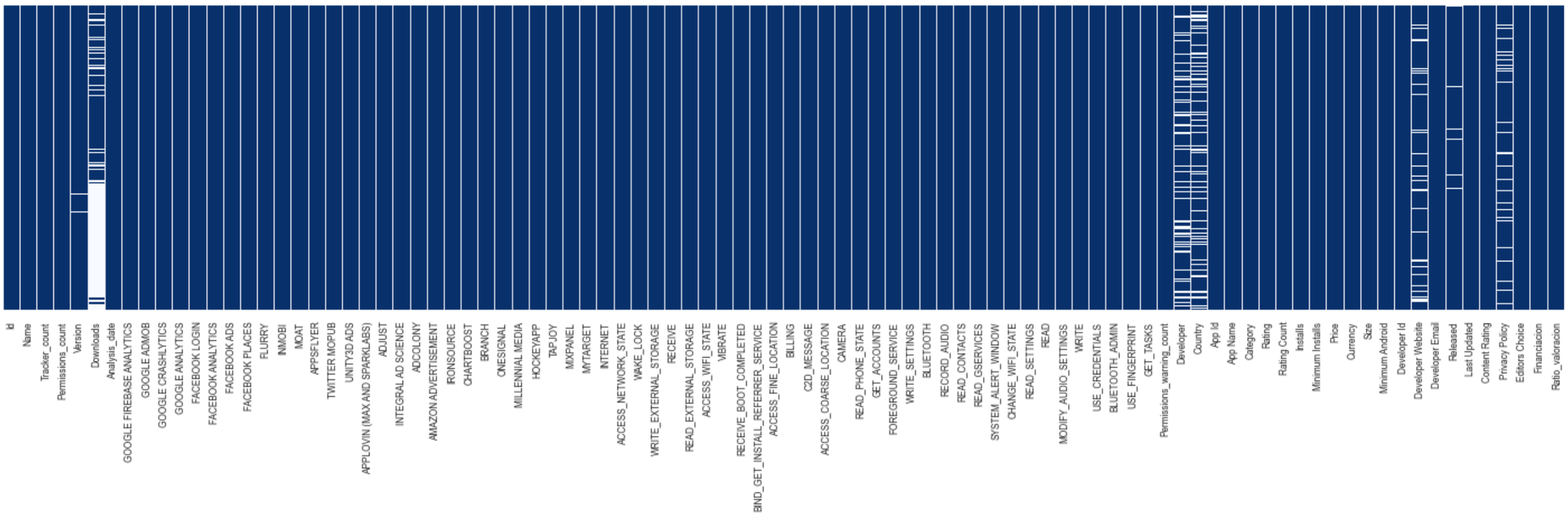
	Cuenta	Porcentaje
0	33006.0000	29.9100
1	45512.0000	41.2400
2	20048.0000	18.1700
3	9187.0000	8.3200
4	1965.0000	1.7800
5	489.0000	0.4400
6	128.0000	0.1200
7	22.0000	0.0200
8	9.0000	0.0100

Estudiamos y graficamos el número de nulos que existe esta vez agrupada por atributos.

In [23]:
```
print('\nDistribución de los valores nulos en los atributos')
plt.figure(figsize = (25,5))
sns.heatmap(df_exodus.isna(), yticklabels = False, cbar = False, cmap = 'Blues_r')
plt.show()

#Listado de atributos con presencia de valores nulos
```

```
df_nulos = pd.DataFrame((df_exodus.isna().sum(), round(df_exodus.isna().sum()*100/df_exodus.shape[0], 2)), index = ['Cuenta', 'Porcentaje']).transpose()
df_nulos[df_nulos['Cuenta'] > 0].sort_values('Cuenta', ascending = False)
```

Distribución de los valores nulos en los atributos



Out[23]:

	Cuenta	Porcentaje
Downloads	54082.0000	49.0000
Country	24899.0000	22.5600
Developer	20736.0000	18.7900
Developer Website	10492.0000	9.5100
Privacy Policy	8193.0000	7.4200
Released	2550.0000	2.3100
Version	2132.0000	1.9300
Ratio_valoracion	431.0000	0.3900
Rating	418.0000	0.3800
Rating Count	418.0000	0.3800
Analysis_date	65.0000	0.0600
Minimum Android	48.0000	0.0400
Installs	1.0000	0.0000
Minimum Installs	1.0000	0.0000
Currency	1.0000	0.0000
Developer Email	1.0000	0.0000

Vemos que por lo general es un conjunto de datos bastante completo, donde destaca la ausencia de valores en 3 únicos atributos de 91. Iremos analizando cada uno de estos atributos y determinando una técnica para imputar un valor y completar la información.

Downloads

Es el atributo con mayor porcentaje de elementos perdidos (49%) del total de registros. Afortunadamente, tras la integración con el dataset *Google Play Store Apps* disponemos del atributo *Installs* que representa la misma información y únicamente tiene un registro no informado. La actuación sobre este atributo, más que imputar los valores, va a ser prescindir del atributo *Downloads* de *Exodus* en favor de *Installs*

In [24]:
```
df_exodus.drop('Downloads', axis = 1, inplace = True)
```

Country

El siguiente atributo con mayor número de ausencias con un 22,56%, se trata del país del desarrollador. Echamos un vistazo a los diez primeros y últimos valores que contiene esta variable y su número de ocurrencias

In [25]:
```
pd.DataFrame(df_exodus['Country'].value_counts()).head(10).append(pd.DataFrame(df_exodus['Country'].value_counts()).tail(10))
```

Out[25]:

	Country
US	29243
DE	5000
FR	2581
Unknown	2564
RU	2476
CN	2417
UK	1837
CA	1733
IN	1535
91	1351
233	1
562	1
08389	1
28710	1
CJ	1
007	1
Switzerland	1
CM	1
542342423452	1
156-ISO 3166-2:CN	1

Vemos que se ha tratado de un campo de libre introducción en el formulario de datos de entrada de publicación de aplicaciones y ha dado lugar a valores variopintos como *Lund*, *dragonpass*, *91* etc. Pero también observamos que la gran mayoría de datos con sentido, utilizan la codificación ISO 3166-1 de países a 2 caracteres. Así pues, la primera acción va a ser normalizar estos valores y considerar correctos aquellos que pertenezcan a la lista elaborada a partir de los 261 códigos ISO oficiales y extendidos aprobados para la denominación de paises. Aquellos que no cumplan, o sean valores perdidos, los estableceremos con el valor 'COM' cuya razón veremos más adelante.

También tendremos en cuenta, que para el código correspondiente al país Gran Bretaña **GB**, también se puede utilizar el código extendido **UK**, con lo que consideramos unificar ambos valores en **GB** que es el oficial.

In [26]:
```
iso_paises = ['AD','AE','AF','AG','AI','AL','AM','AO','AQ','AR','AS','AT','AU','AW','AX','AZ','BA','BB','BD','BE','BF','BG','BH','BI','BJ','BL','BM','BN','BO','BQ','BR','BS','BT','BV','BW','BY','BZ',
              'CA','CC','CD','CF','CG','CH','CI','CK','CL','CM','CN','CO','CR','CU','CV','CW','CX','CY','CZ','DE','DJ','DK','DM','DO','DZ','EC','EE','EG','EH','ER','ES','ET','FI','FJ','FK','FM','FO',
              'FR','GA','GB','GD','GE','GF','GG','GH','GI','GL','GM','GN','GP','GQ','GR','GS','GT','GU','GW','GY','HK','HM','HN','HR','HT','HU','ID','IE','IL','IM','IN','IO','IQ','IR','IS','IT','JE',
              'JM','JO','JP','KE','KG','KH','KI','KM','KN','KP','KR','KW','KY','KZ','LA','LB','LC','LI','LK','LR','LS','LT','LU','LV','LY','MA','MC','MD','ME','MF','MG','MH','MK','ML','MM','MN','MO',
              'MP','MQ','MR','MS','MT','MU','MV','MW','MX','MY','MZ','NA','NC','NE','NF','NG','NI','NL','NO','NP','NR','NU','NZ','OM','PA','PE','PF','PG','PH','PK','PL','PM','PN','PR','PS','PT','PW',
              'PY','QA','RE','RO','RS','RU','RW','SA','SB','SC','SD','SE','SG','SH','SI','SJ','SK','SL','SM','SN','SO','SR','SS','ST','SV','SX','SY','SZ','TC','TD','TF','TG','TH','TJ','TK','TL','TM',
              'TN','TO','TR','TT','TV','TW','TZ','UA','UG','UM','US','UY','UZ','VA','VC','VE','VG','VI','VN','VU','WF','WS','YE','YT','ZA','ZM','ZW','AC','CP','DG','EA','EU','EZ','FX','IC','SU','TA',
              'UK','UN']

def normaliza_pais(country):
    '''
    La función normaliza_pais, estandarizará el dato consignado en el atributo Country en base a los códigos ISO de países de 2 caracteres. Generalizará aquellos que corresponda
    a COM y les dará formato de letras mayúsculas.
    '''
    if str(country).upper().strip().replace(' ', '') in iso_paises:
        if country.upper() == 'UK': return 'GB'
        return country.upper()
    else:
        return 'COM'

#Ejecución de la estandarización
df_exodus['Country'] = np.vectorize(normaliza_pais)(df_exodus['Country'])
#Visualización de resultados
pd.DataFrame(df_exodus['Country'].value_counts()).head(5).append(pd.DataFrame(df_exodus['Country'].value_counts()).tail(5))
```

Out[26]:

	Country
COM	38854
US	29516

	Country
DE	5513
GB	3057
FR	2930
SB	1
GP	1
GM	1
BS	1
GD	1

Para poder resolver los registros donde se ha consignado COM, revisamos el resto del Data Frame para encontrar algún atributo que nos pueda servir, y vemos que el Data Frame de integración nos ha incorporado el atributo *Developer Website* y *Developer Email*, los cuales al final de su dominio, incorporan información que podríamos interpretar como país del desarrollador. En el peor de los casos, en el dominio genérico .COM, coincidiría con el valor genérico al que hemos establecido el atributo en el primer paso.

In [27]:
```
def tratamiento_country(country, developerWebsite, developerEmail):
    '''
    La función tratamiento_country, recibe información sobre los atributos pais, web y e-mail del desarrollador para imputar, a aquellos valores generales 'COM'
    el valor del dominio que hayan consignado en su página web o correo electrónico.
    Se tiene en cuenta que los datos de referencia puden venir nulos e inicializamos para que el tratamiento les adjudique COM
    También tendremos en cuenta que los valores de los datos de referencia correspondan a la codificación ISO por la que estamos estandariazando el atributo.
    '''
    #Tratamiento previo para los valores indeterminados actuales en los atributos de referencia
    if pd.isna(developerWebsite): developerWebsite = 'http://www.algo.com'
    if pd.isna(developerEmail): developerEmail = 'algo@algo.com'

    try:
        if country == 'COM':  #Solamente tratamos de concretar los valores establecidos al genérico COM
            #preferencia al Website
            if developerWebsite.find('/', developerWebsite.rfind('.')) > 0:
                pais = developerWebsite[developerWebsite.rfind('.') + 1:developerWebsite.find('/', developerWebsite.rfind('.'))].upper()
            else:
                pais = developerWebsite[developerWebsite.rfind('.') + 1:].upper()
            if (pais != 'COM') and (pais in iso_paises): #Si el código de país extraído de la web del desarrollador es específica y además se trata de un país
                return(pais) #Imputamos el valor encontrado
            else: #Damos oportunidad al dominio del e-mail
                pais = developerEmail[developerEmail.rfind('.') + 1:].upper()
                if (pais != 'COM') and (pais in iso_paises): #Si el código de país extraído del e-mail es específica y además se trata de un país
                    return(pais) #Imputamos el valor encontrado
                else: #En cualquier otro caso, mantenemos el original COM
                    return(country)
    except:
        print(country, developerWebsite, developerEmail)
    finally:
        return country #Si no se consigue imputar, se devuele el original.

#Ejecución del tratamiento
df_exodus['Country'] = np.vectorize(tratamiento_country)(df_exodus['Country'], df_exodus['Developer Website'], df_exodus['Developer Email'])
#Visualizar resultado
pd.DataFrame(df_exodus['Country'].value_counts()).head(5).append(pd.DataFrame(df_exodus['Country'].value_counts()).tail(5))
```

Out[27]:

	Country
COM	38854
US	29516
DE	5513
GB	3057
FR	2930
SB	1
GP	1
GM	1
BS	1
GD	1

Se comprueba que no hemos conseguido imputar valores distintos a COM por este método, por lo que utilizamos los valores disponibles ya completos del atributo.

Developer

Developer cuenta con un porcentaje de valores perdidos del 18,79%. Afortunadamente nos encontramos como en el caso de *Downloads* con información mucho más completa y confiable proveniente del atributo *Developer Id* del data set *Google Play Store Apps*.

Por ejemplo, para el desarrollador Google, vemos en el dataset de integración unas denominaciones estandarizadas como el uso de la partícula LLC que concuerda con el otro gigante Amazon. Contrasta con el dataset original donde no se utiliza la misma denominación para la empresa, en este caso Inc., hay muchas más aplicaciones registradas para el desarrollador Google, lo que parece indicar que es un campo de libre entrada en el formulario utilizado por Exodus y también observamos otra aparición de Google que nos hace sospechar falta de integridad.
También vemos en el dataset original la marca desconocido *Unknown* que no se especifica en el de integración, por lo que la calidad del dato de este último dataset es mucho mayor.

Así pues, resolveremos este caso descartando el atributo del data set original *Exodus* para utilizar el del dataset integrador.

In [28]:
```
#Exposición de los valores actuales en el atributo de integración y original
print('Valores encontrados en el dataset de integración')
print(df_exodus['Developer Id'].value_counts())
print('\nValores encontraods el dataset original')
print(df_exodus['Developer'].value_counts())

#Ejecución de la eliminación del atributo en el dataset original.
df_exodus.drop('Developer', axis = 1, inplace = True)
```

```
Valores encontrados en el dataset de integración
Google LLC                                          1565
Microsoft Corporation                                544
Facebook                                             246
Amazon Mobile LLC                                    216
Mozilla                                              193
                                                    ...
Xiaomiui                                               1
Fraunhofer-Gesellschaft  e.V.                         1
Gnaural                                                1
Zahid Hussain Chihpa / Islamic Technology Mission      1
CUTBOSS                                                1
Name: Developer Id, Length: 29711, dtype: int64

Valores encontraods el dataset original
Google Inc.              15669
Unknown                   2370
Microsoft Corporation      538
fdroid.org                 409
Google                     405
                         ...
MyOxygen Ltd                 1
File Creator LTD             1
designspacestudio            1
Alibaba Inc.                 1
Meta Priori                  1
Name: Developer, Length: 18705, dtype: int64
```

Developer Website

Este es un atributo del dataset de integración que se ha utilizado previamente para la imputación del país del desarrollador. Aun conociendo que contiene un 9,51% de valores perdidos, lo utilizamos para la imputación anterior porque estos casos eran irrelevantes por considerar la utilización de la etiqueta general COM.

La técnica para imputar estos valores, conociendo que el atributo *Developer Id* está completo, será la siguiente:

- Si encontramos un registro donde el desarrollador coincide y tiene informada la página web, asignar este valor.
- Si no se encuentra el desarrollador o tampoco tiene informada la página web, dejaremos la "etiqueta desconocido" asumiendo que el atributo puede no tener web asignada.

In [29]:
```
#Creamos un diccionario para almacenar la información desarrollador:sitio web
dict_developer_web = {}
for index, row in df_exodus[['Developer Id', 'Developer Website']].iterrows():
    if pd.isna(row['Developer Website']) == False:
        dict_developer_web[row['Developer Id']] = row['Developer Website']
    else:
        dict_developer_web[row['Developer Id']] = np.NaN

def imputa_website(developer):
    '''
    La función retorna el sitio web asociado al desarrollador que recibe como argumento dentro del diccionario confeccionado de los datos del propio dataframe
    '''
    return dict_developer_web[developer]

df_exodus['Developer Website'] = np.vectorize(imputa_website)(df_exodus['Developer Id'])
df_exodus['Developer Website'] = df_exodus['Developer Website'].replace('nan', np.nan) #Como la vectorización devuelve 'nan' como string, reemplazampos por el NaN de pandas
```

```
pd.DataFrame((df_exodus['Developer Website'].isna().sum(),
              round(df_exodus['Developer Website'].isna().sum()*100/df_exodus.shape[0], 2)),
              index = ['Cuenta', 'Porcentaje'], columns = ['Developer Website']).transpose()
```

Out[29]:

	Cuenta	Porcentaje
Developer Website	10101.0000	9.1500

Con esta imputación hemos podido recuperar 391 valores perdidos y reducir en 0,36% la incidencia de valores perdidos del atributo.

Privacy Policy

Para este atributo proveniente del data set de integración, vamos a realizar dos rondas de imputación. La primera, consignando la página web del desarrollador, donde conocemos que el 9,15% de los datos no están completos, y en una segunda ronda, la información del correo electrónico, donde se garantiza la completitud de prácticamente el 100% de los casos.
Este atributo aportará al menos información de contacto para consultar estas políticas.

In [30]:
```
#En la primera ronda, a los atributos perdidos se asigna la web del desarrollador
df_exodus.loc[df_exodus['Privacy Policy'].isna(), 'Privacy Policy'] = df_exodus['Developer Website']
#En la segunda ronda, a los atributos perdidos resultantes se asigna el mail del desarrollador
df_exodus.loc[df_exodus['Privacy Policy'].isna(), 'Privacy Policy'] = df_exodus['Developer Email']
```

Esta imputación ha conseguido restituir los valores perdidos del atributo *Privacy Policy*

Released

Atributo del data set de integración, contiene la información de lanzamiento de la aplicación. Una buena aproximación a esta fecha podría ser el atributo *Analysis_date* del data set original *Exodus* puesto que el análisis se realiza por versión de la aplicación. De todas las versiones de la aplicación que mantenemos en el conjunto de datos *Exodus*, lo correcto sería tomar la menor de ellas puesto que se trata de la más cercana a la aparición de la aplicación.

In [31]:
```
#En primer lugar, estandarizamos el formato de la fecha de las columnas de integración Relased y Last Updated y original Analysis_date
#Aplico a los valores perdidos una constante extrema '01-01-1678' para evitar fallos en la función de estandarización por máscara
df_exodus.loc[df_exodus['Released'].isna(), 'Released'] = 'Jan 01, 1678'
df_exodus.loc[df_exodus['Analysis_date'].isna(), 'Analysis_date'] = '01-01-1678'

from datetime import datetime

mascara = '%b %d, %Y'
df_exodus['Released_std'] = np.vectorize(datetime.strptime)(df_exodus['Released'], mascara)
df_exodus['Last Updated_std'] = np.vectorize(datetime.strptime)(df_exodus['Last Updated'], mascara)

mascara = '%d-%m-%Y'
df_exodus['Analysis_date_std'] = np.vectorize(datetime.strptime)(df_exodus['Analysis_date'], mascara)

#Creamos la función de imputación
def minFechaAnalisis(appId, fechaActual):
    if fechaActual == pd.Timestamp('1678-01-01'):
        return df_exodus[df_exodus['App Id'] == appId]['Analysis_date_std'].min()

    return fechaActual

df_exodus['Released_std'] = np.vectorize(minFechaAnalisis)(df_exodus['App Id'], df_exodus['Released_std'])
```

Analysis_date

Continuando con el caso anterior, los casos en los que la fecha de análisis no estaba informada, y que en el paso anterior se ha consignado el valor extremo *01-01-1678*, se puede considerar por un criterio conmutativo al anterior, imputar la propia fecha de lanzamiento de la aplicación

In [32]:
```
df_exodus.loc[df_exodus['Analysis_date_std'] == pd.Timestamp('1678-01-01'), 'Analysis_date_std'] = df_exodus['Released_std']

#Al finalizar el trabajo con los atributos de fecha, elimino los originales no estandarizados y me quedo con los estandarizados utilizando el nombre original del atributo.
df_exodus.drop(['Analysis_date', 'Released', 'Last Updated'], axis = 1, inplace = True)
df_exodus = df_exodus.rename(columns={'Analysis_date_std': 'Analysis_date', 'Released_std': 'Released', 'Last Updated_std': 'Last Updated'})
```

Version

No disponemos en el Data Frame original ni en el Data Frame de integración, columnas que permitan imputar directamente la información de versionado de las aplicaciones que han sido analizadas. Tampoco sería factible imputar un formato de versiones tal y como se suele realizar habitualmente en secciones principales y subsecciones (x.xx.xxx).

Con todo, habría una manera eficiente de establecer un atributo ordinal consistente en un contador de las ocurrencias de las aplicaciones con valor nulo en *Version* a partir de la fecha en que han sido analizadas *Analysis_date*. Siendo la primera ocurrencia la versión 1 e incrementando esta cifra en una unidad según la antigüedad de dicha revisión de la aplicación.

In [33]:
```
#Creamos un dataframe auxiliar que contemple las aplicaciones sin versión y su fecha de análisis ordenado por esta última ascendentemente.
df_sinVersion_aux = df_exodus[df_exodus['Version'].isna()][['App Id', 'Analysis_date']].sort_values(by = ['App Id', 'Analysis_date'])
```

```
#Con la función de suma acumulativa a partir de 1, creamos el contador y lo almacenamos en una nueva columna del dataframe auxiliar.
df_sinVersion_aux['VersionAux'] = df_sinVersion_aux.groupby(['App Id']).cumcount() + 1

#Imputamos en el Data Frame origen los valores obtenidos en el dataframe auxiliar.
for i, row in df_exodus.iterrows():
    if pd.isna(row['Version']):
        df_exodus.at[i, 'Version'] = [df_sinVersion_aux[(df_sinVersion_aux['App Id'] == row['App Id']) & (df_sinVersion_aux['Analysis_date'] == row['Analysis_date'])]['VersionAux'].values][0][0]
```

Rating y Rating Count

Ambos atributos forman parte del Data Frame de integración y la falta de información, afecta a ambos atributos del registro por igual. Es decir, si en un registro uno de ellos no está informado, el otro tampoco lo está.

Se trata de valores numéricos y susceptibles de tener dependencia con otros factores presentes en el conjunto de datos, por lo que vamos a adoptar una estrategia de imputación basada en k-vecinos.

In [34]:
```
#Creamos el dataset de entrenamiento con aquellos atributos que puedan ser susceptibles de condicionar la votación de los usuarios.
X = df_exodus[['Tracker_count', 'Permissions_count', 'Permissions_warning_count', 'Category', 'Rating', 'Rating Count', 'Installs',
               'Price', 'Content Rating', 'Editors Choice', 'Financiacion']]

#Creamos el dataset de entrenamiento con los registros donde las valoraciones están informadas y como dataset de test donde no lo están.
X = pd.get_dummies(X, drop_first = True) #Transformar en binarios cada uno de los posibles valores de los atributos categóricos.

X_train = X[X['Rating'].isna() == False] #El conjunto de entrenamiento son aquellos registros con valores en Rating y por extensión Rating Count
X_test = X[X['Rating'].isna()] #El conjunto de test son los registros que queremos imputar porque Rating y Rating Count son nulos
y_train_rating = X_train['Rating'] #Los valores de Rating para el conjunto de entrenamiento
y_train_ratingCount = X_train['Rating Count'] #Los valores de Rating count para el conjunto de entrenamiento.
X_train = X_train.drop(['Rating', 'Rating Count'], axis = 1) #Eliminar las variables destino del conjunto de entrenamiento
X_test = X_test.drop(['Rating', 'Rating Count'], axis = 1) #Eliminar las variables destino del conjunto de test

from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor() #Crear el modelo por defecto
knn.fit(X_train, y_train_rating) #Entrenamiento del modelo para predecir Rating
pred_rating = knn.predict(X_test) #Predicciones a imputar en Rating según los atributos independientes en los registros nulos.
knn.fit(X_train, y_train_ratingCount) #Entrenamiento del modelo para predecir Rating Count
pred_ratingCount = knn.predict(X_test) #Predicciones a imputar en Rating Count según los atributos independientes en los registros nulos.

#Imputamos las predicciones en el Data Frame
df_exodus.loc[df_exodus['Rating'].isna(), 'Rating'] = pred_rating
df_exodus.loc[df_exodus['Rating Count'].isna(), 'Rating Count'] = pred_ratingCount
```

Minimum Android

Se trata de un atributo categórico del Data Frame de integración el cual vamos a imputar también con k-vecinos a partir de la probabilidad condicionada por la fecha de lanzamiento y actualización de la aplicación que son los atributos más condicionante a la hora de utilizar una versión de Android con la que programar la aplicación.

In [35]:
```
#Creamos el dataset de entrenamiento con aquellos atributos que son suceptibles de condicionar la versión mínima de Android
X = df_exodus[['Released','Last Updated','Minimum Android']].copy()
#De las fechas, obtenemos los años puesto que las versiones de Android son anuales.
X['Released'] = pd.DatetimeIndex(X['Released']).year
X['Last Updated'] = pd.DatetimeIndex(X['Last Updated']).year

X_train = X[X['Minimum Android'].isna() == False] #El conjunto de entrenamiento son aquellos registros con valores en Minimum Android
X_test = X[X['Minimum Android'].isna()] #El conjunto de test son los registros que queremos imputar cuando Minimum Android es nulo
y_train = X_train['Minimum Android'] #Los valores de Rating para el conjunto de entrenamiento

X_train = X_train.drop(['Minimum Android'], axis = 1) #Eliminar las variables destino del conjunto de entrenamiento
X_test = X_test.drop(['Minimum Android'], axis = 1) #Eliminar las variables destino del conjunto de test

#Categorizar los valores de Minimum Android
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
y_train = le.fit_transform(y_train)

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB() #Creación del modelo
gnb.fit(X_train, y_train) #Entrenar el modelo
y_pred = gnb.predict(X_test) #Predicciones a imputar en 'Minimum Android' según la probabilidad dicatada por los años de lanzamiento y actualización

#Imputamos las predicciones en el Data Frame
df_exodus.loc[df_exodus['Minimum Android'].isna(), 'Minimum Android'] = le.inverse_transform(y_pred)
```

Installs, Minimum Installs, Currency, Developer Email

Estos son casos simples que podremos resolver imputando los valores manualmente.

En el caso de *Installs*, *Minimum Installs* y *Currency*, estos tres valores perdidos pertenecen al mismo registro por lo que buscamos la aplicación en la Play Store y completamos la información https://play.google.com/store/apps/details?id=com.elex.ufcgp&hl=es_419&gl=US

El caso del *Developer Email* pertenece a otro registro, pero lo podemos resolver de la misma manera. https://play.google.com/store/apps/details?id=com.brother.ptouch.designandprint&hl=es&gl=US

In [36]:
```
df_exodus[df_exodus['Minimum Installs'].isna()].append(df_exodus[df_exodus['Developer Email'].isna()])
```

Out[36]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLO
101923	148853	Dream Score	7	14	1.0.1789	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	
32350	55981	Brother Image Viewer	0	1	1.00.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [37]:
```
#En el caso del juego de fútbol, vemos que la aplicación está en fase beta pre-registro por lo que no hubo instalaciones y aunque el NaN es legítimo consolidaremos 0+ u USD en currency
#puesto que es el valor más repetido.
df_exodus.loc[df_exodus['App Id'] == 'com.elex.ufcgp', 'Installs'] = '0+'
df_exodus.loc[df_exodus['App Id'] == 'com.elex.ufcgp', 'Minimum Installs'] = 0
df_exodus.loc[df_exodus['App Id'] == 'com.elex.ufcgp', 'Currency'] = 'USD'
#En el caso de la aplicación de Brother, hemos indagado en otras aplicaciones de la misma compañía y aquí sí ofrecían el mail que vamos a consignar.
df_exodus.loc[df_exodus['App Id'] == 'com.brother.mfc.brimgvwr', 'Developer Email'] = 'mobile-apps-es@brother.com'
```

De esta manera, así queda el Data Frame *Exodus* finalmente tras la gestión de datos perdidos. Prácticamente completo salvo por la información de la web del desarrollador *Developer Website* donde queda como indeterminado en un 9,15% de los registros.

Ratio_valoracion

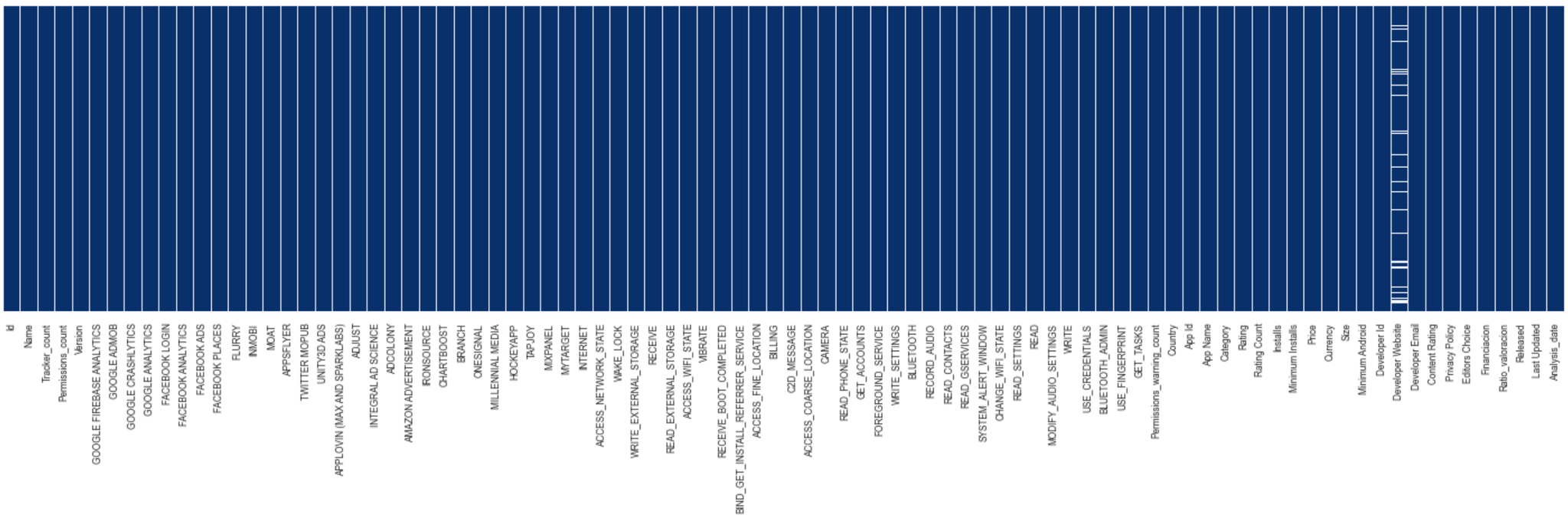
Una vez imputados valores en Rating Count y Minimum Installs, se pueden calcular de manera completa todos los valores del atributo.

Y aquellos valores que no se pueden calcular porque no han sido instaladas por nadie, se imputa también 0.

In [38]:
```
#Recálculo del ratio con todos los valores establecidos.
df_exodus['Ratio_valoracion'] = df_exodus['Rating Count']/df_exodus['Minimum Installs']
#Casos en que no se han producido instalaciones (Aplicaciones Beta)
df_exodus.loc[df_exodus['Ratio_valoracion'].isna(), 'Ratio_valoracion'] = 0
```

In [39]:
```
print('\nDistribución de los valores nulos en los atributos')
plt.figure(figsize = (25,5))
sns.heatmap(df_exodus.isna(), yticklabels = False, cbar = False, cmap = 'Blues_r')
plt.show()

#Listado de atributos con presencia de valores nulos
df_nulos = pd.DataFrame((df_exodus.isna().sum(), round(df_exodus.isna().sum()*100/df_exodus.shape[0], 2)), index = ['Cuenta', 'Porcentaje']).transpose()
df_nulos[df_nulos['Cuenta'] > 0].sort_values('Cuenta', ascending = False)
```

```
Distribución de los valores nulos en los atributos
```



Out[39]:

	Cuenta	Porcentaje
Developer Website	10101.0000	9.1500

3.2. Tratamiento de valores extremos

Para la detección de valores extremos, analizamos las estadísticas de distribución de los atributos **numéricos** y *categóricos* relevantes.

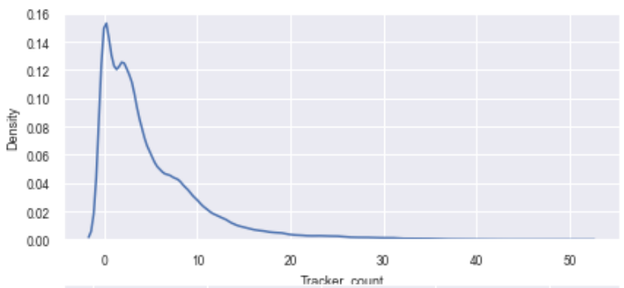
- Id: Los campos secuenciales y únicos de identificación no se consideran susceptibles de tener valores extremos.
- Name: Este campo de denominación no es susceptible de contener valores extremos.
- **Tracker_count**: Estudiaremos del atributo numérico la distribución de sus valores.
- **Permissions_count**: Estudiaremos del atributo numérico la distribución de sus valores.
- *Version*: Es un atributo categórico del cual podremos estudiar la distribución de los distintos valores de la variable.
- Atributos de rastreadores y permisos específicos: Estas variables dicotomizadas no son susceptibles de contener valores extremos.
- **Permissions_warning_count**: Estudiaremos el atributo numérico de permisos solicitados potencialmente de riesgo.
- *Country*: Es un atributo categórico del cual podremos estudiar la distribución de los distintos valores de la variable.
- App Id: Es un campo secuencial y único de identificación y no se considera susceptible de contener valores extremos.
- App Name: De nuevo este campo de denominación no es susceptible de contener valores extremos.
- *Category*: Campo categórico del que se puede estudiar la distribución de sus valores.
- **Rating**: Estudiaremos del atributo numérico la distribución de sus valores.
- **Rating Count**: Estudiaremos del atributo numérico la distribución de sus valores.
- *Installs*: Campo categórico del que se puede estudiar la distribución de sus valores.
- **Minimum Installs**: Estudiaremos del atributo numérico la distribución de sus valores.
- Developer Id, Developer Website, Developer Email: Son campos descriptivos no susceptibles de contener valores extremos.
- *Content Rating*: Campo categórico del que se puede estudiar la distribución de valores.
- Privacy Policy: Atributo descriptivo no susceptible de contener valores extremos.
- Editors Choice: Atributo booleano no susceptible de contener valores extremos.
- *Financiacion*: Atributo categórico del que se puede estudiar la distribución de sus valores.
- **Ratio_valoracion**: Nuevo atributo creado a partir de Rating Count y Minimum Installs. Se estudiará la distribución de sus valores.
- *Released, Last Updated, Analysis_date*: En Los atributos de fecha podremos estudiar algunos de sus estadísticos para detectar valores extremos.

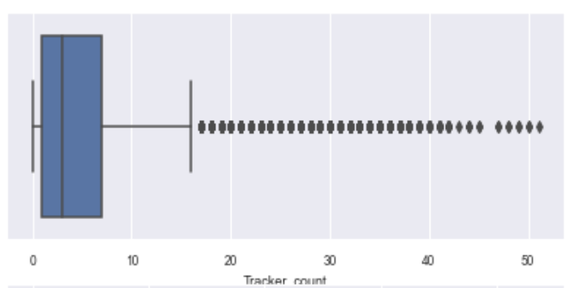
3.2.1 Atributos numéricos

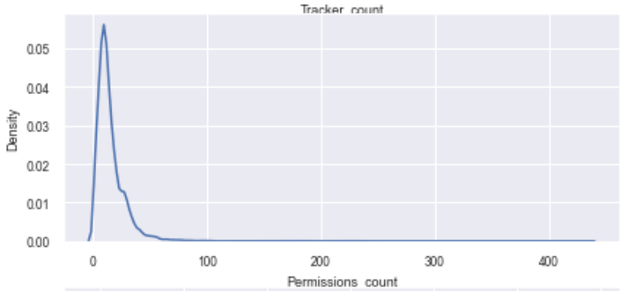
Analizaremos la presencia de valores extremos en las variables mediante el análisis de estadísticos de referencia de distribución de los valores de la variable. También graficamos la curva de la distribución para observar si sigue una *distribución normal* y un Box Plot para una mejor visualización de estos *outliers*.

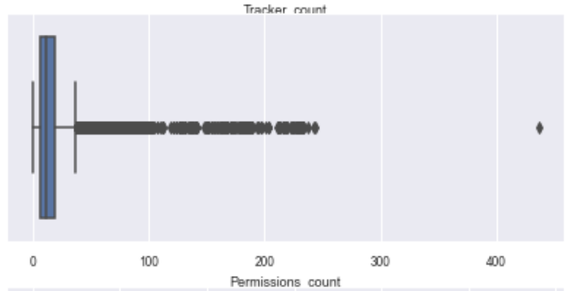
In [40]:
```
#Visualizar la distribución
```

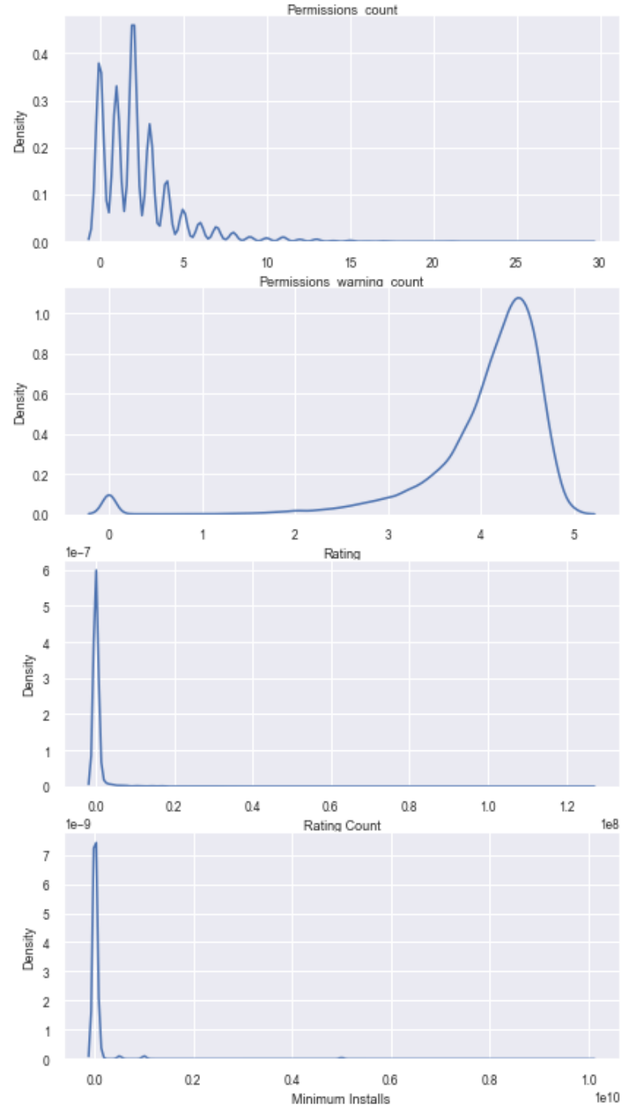
```
fig, axs = plt.subplots(nrows = 6,ncols = 2,figsize=(15,20))

sns.kdeplot(df_exodus['Tracker_count'], ax = axs[0][0])
sns.boxplot(data = df_exodus, x = 'Tracker_count', ax = axs[0][1])

sns.kdeplot(df_exodus['Permissions_count'], ax = axs[1][0])
sns.boxplot(data = df_exodus, x = 'Permissions_count', ax = axs[1][1])

sns.kdeplot(df_exodus['Permissions_warning_count'], ax = axs[2][0])
sns.boxplot(data = df_exodus, x = 'Permissions_warning_count', ax = axs[2][1])

sns.kdeplot(df_exodus['Rating'], ax = axs[3][0])
sns.boxplot(data = df_exodus, x = 'Rating', ax = axs[3][1])

sns.kdeplot(df_exodus['Rating Count'], ax = axs[4][0])
sns.boxplot(data = df_exodus, x = 'Rating Count', ax = axs[4][1])

sns.kdeplot(df_exodus['Minimum Installs'], ax = axs[5][0])
sns.boxplot(data = df_exodus, x =  'Minimum Installs', ax = axs[5][1])
plt.show()

df_exodus[['Tracker_count','Permissions_count','Permissions_warning_count','Rating','Rating Count','Minimum Installs']].describe()
```

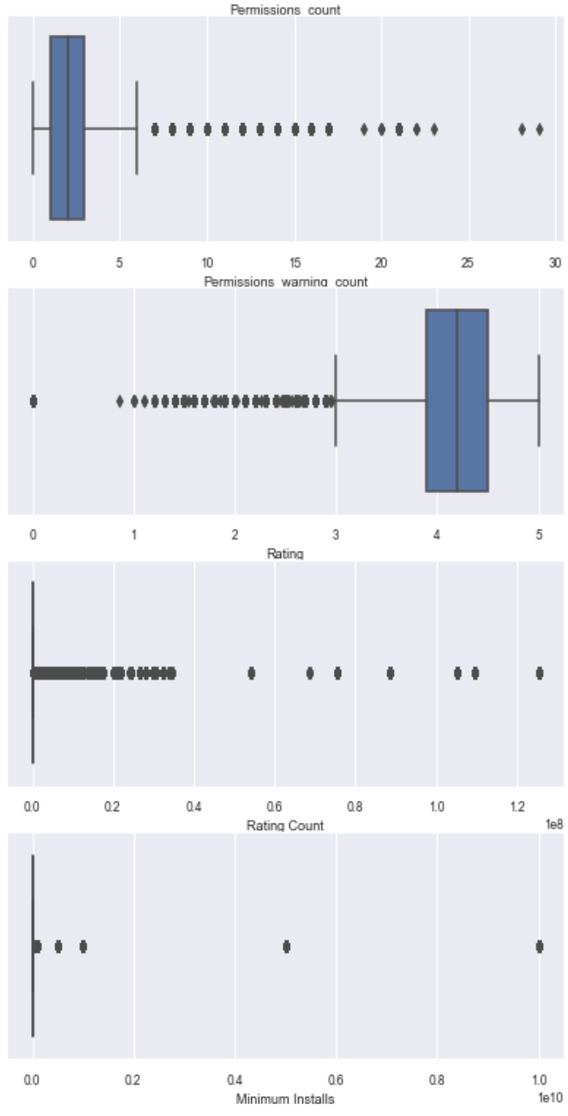












Out[40]:

	Tracker_count	Permissions_count	Permissions_warning_count	Rating	Rating Count	Minimum Installs
count	110366.0000	110366.0000	110366.0000	110366.0000	110366.0000	110366.0000
mean	4.9232	15.0675	2.2435	4.0550	771866.9348	52496820.6004

	Tracker_count	Permissions_count	Permissions_warning_count	Rating	Rating Count	Minimum Installs
std	5.5820	14.1773	2.2698	0.7490	6106066.1508	430224174.4410
min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25%	1.0000	7.0000	1.0000	3.9000	727.0000	100000.0000
50%	3.0000	12.0000	2.0000	4.2000	7678.0000	500000.0000
75%	7.0000	19.0000	3.0000	4.5000	73019.0000	5000000.0000
max	51.0000	436.0000	29.0000	5.0000	125380770.0000	10000000000.0000

Tracker_count

Sobre el dato que indica el número de rastreadores presentes en las aplicaciones, se observa una anomalía clara en el valor máximo donde existe una aplicación que alberga hasta *51* rastreadores cuando la media son *5 rastreadores* y *7* el valor de los elementos en el Q3. El resto de estadísticos se pueden considerar dentro de lo normal. La visualización del diagrama de cajas considera la existencia de *34* valores anómalos para considerar que un registro contiene un valor extremo en el atributo.

Permissions_count

El dato que registra el número de permisos que tiene que aceptar el usuario también presenta un comportamiento similar al anterior donde se observa un valor máximo de *436 permisos* sabiendo que de media las aplicaciones demandan la utilización de unos *15* y *19* es el valor representativo del Q3.
En este caso, la visualización del diagrama de cajas nos considera un buen número de valores anómalos correspondientes a aplicaciones que piden más de *40 permisos* para su instalación.

Permissions_warning_count

En el caso de la cuenta de permisos potencialmente delicados, puede existir una anomalía clara en el valor máximo donde existe una aplicación que alberga hasta *29* permisos delicados cuando la media son *2 permisos* y *3* el valor de los elementos en el Q3. El resto de estadísticos se pueden considerar dentro de lo normal. La visualización del diagrama de cajas considera la existencia de *18* valores anómalos para considerar que un registro contiene un valor extremo en el atributo.

Rating

El dato que representa la valoración de los usuarios entra dentro de la normalidad puesto que esta valoración se define en el rango entre *0* y *5* que son, justamente, los valores mínimo y máximo encontrados en el Data Frame.
El diagrama de cajas, nos marca como extremas las votaciones por debajo de *3*, pero en este caso no deberían ser consideradas extremas porque entran dentro del rango de valoraciones.

Rating Count, Minimum Installs

Son las dos variables más complejas con una distribución muy complicada.

En la variable *Rating Count*, si apreciamos, la media *(771.867)* está muy por encima del valor del Q3 *(73.019)*. Esto provocado por la existencia de unos cuantos valores, apreciables en el diagrama de cajas, que superan los *6 millones de votaciones*, siendo el valor máximo *(125.380.770)*. Cabría observar a qué aplicación pertenecen estos datos y contrastarlos. Si son legítimos, quizá cabría discretizar la variable para poder establecer un techo.
Minimum Installs se encuentra relacionado con la variable anterior puesto que para valorar la aplicación es necesario tenerla instalada. Como en el anterior caso, la media se encuentra desvirtuada *(52.000.000)*, muy superior al valor de la Q3 establecido en *5 millones de instalaciones*. Esta situación también estaría explicada por unos pocos valores, como por ejemplo el máximo establecido en *10.000 millones*.

El ratio de la cuenta de votantes con el número de aplicaciones de la instalación en su extremo es de *0,0125* y es un buen dato porque se asemeja al de la media *0,0147*. Aunque mínimo, cuanta más gente tiene instalada la aplicación ésta se valora algo menos.

Tracker_count

Obtenemos los umbrales inferior y superior para discriminar los valores alejados más de tres desviaciones típicas de la media.

In [41]:
```
z = df_exodus['Tracker_count'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Tracker_count'].mean() - (3 * z)
sup_3z = df_exodus['Tracker_count'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Tracker_count'] < inf_3z) | (df_exodus['Tracker_count'] > sup_3z)].shape[0]))

#Observar los valores outliers
print('\nRastreadores | Cuenta apps\n' +
      str(df_exodus[(df_exodus['Tracker_count'] < inf_3z) | (df_exodus['Tracker_count'] > sup_3z)]['Tracker_count'].value_counts().sort_values(ascending = False)) + '\n')

print('\n25 aplicaciones afectadas: ' +
      str(df_exodus[(df_exodus['Tracker_count'] < inf_3z) | (df_exodus['Tracker_count'] > sup_3z)]['Name'].value_counts().index.to_list()[:25]) + '\n')

df_exodus[(df_exodus['Tracker_count'] < inf_3z) | (df_exodus['Tracker_count'] > sup_3z)].sort_values(by = ['Tracker_count'], ascending = False).head(5)
```

```
Umbrales valores extremos: (-11.822795306647443, 21.669197278269138)
Registros con valores extremos: 2510

Rastreadores | Cuenta apps
23    306
22    278
25    276
24    268
26    196
28    174
27    160
```

```
29    143
31    139
30    130
34     79
33     74
32     73
35     68
36     33
37     29
38     23
39     15
40     13
41     10
42      8
44      3
47      2
50      2
48      2
45      2
49      2
43      1
51      1
Name: Tracker_count, dtype: int64


25 aplicaciones afectadas: ['Meme Generator Free', 'Photo Studio', 'Perfect365', 'Perfect365: One-Tap Makeover', 'WiFi Overview 360', 'PicsArt Photo Studio: Collage Maker & Pic Editor', 'mobile.de – Germany‘s largest car market',
'Alarmy', 'CodeCheck', 'TextNow', 'Wiseplay', 'My Tide Times - Tide Tables, Forecasts & Tides!', 'Amino: Communities and Chats', 'Speedcheck', 'CamStream - Live Camera Streaming', 'GO Launcher - 3D parallax Themes & HD Wallpapers',
'Magic Tiles 3', 'mobile.de', 'Shopping List - Buy Me a Pie!', 'Covet Fashion - Dress Up Game', 'SPEEDCHECK - Speed Test', 'Tandem', 'Lister', 'Mr. Pillster pill box & pill reminder tracker rx', 'My Talking Tom 2']
```

Out[41]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLO
32596	56347	Perfect365: One-Tap Makeover	51	56	7.65.6	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	
30556	53348	Perfect365: One-Tap Makeover	50	59	7.61.12	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	
35028	60012	Perfect365: One-Tap Makeover	50	56	7.67.12	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	
20803	37305	Perfect365: One-Tap Makeover	49	56	7.53.9	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	
26504	47092	Perfect365: One-Tap Makeover	49	59	7.57.13	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1	

De los *2.510* registros con valores que superan el umbral *más de 21 rastreadores*, vemos que la cuenta agrupada de aplicaciones por número de rastreadores es coherente y decreciente a medida que aumenta el número de rastreadores incluidos. Esto podría ser una señal de que los datos sean legítimos.

Los nombres de las 25 primeras aplicaciones, también nos hacen presumir que tiene pinta de ser apps reclamo orientadas a una captura de datos masiva.

Finalmente, una investigación en la propia web de Exodus https://reports.exodus-privacy.eu.org/es/reports/56347/ y la Google Play Store, confirma que ese valor máximo de *51 rastreadores* en una sola aplicación *Perfect365: One-Tap Makeover* es legítimo.

Se opta por no corregir los valores de estos registro por la **legitimidad** y su valor para identificar las aplicaciones más problemáticas.

Permissions_count

Obtenemos los umbrales inferior y superior para discriminar los valores alejados más de tres desviaciones típicas de la media.

In [42]:
```
z = df_exodus['Permissions_count'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Permissions_count'].mean() - (3 * z)
sup_3z = df_exodus['Permissions_count'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Permissions_count'] < inf_3z) | (df_exodus['Permissions_count'] > sup_3z)].shape[0]))

#Observar los valores outliers
print('\nPermisos | Cuenta apps\n' +
      str(df_exodus[(df_exodus['Permissions_count'] < inf_3z) | (df_exodus['Permissions_count'] > sup_3z)]['Permissions_count'].value_counts().sort_values(ascending = False).head(5)) +
      '\n...\n' +
```

```
      str(df_exodus[(df_exodus['Permissions_count'] < inf_3z) | (df_exodus['Permissions_count'] > sup_3z)]['Permissions_count'].value_counts().sort_values(ascending = False).tail(25)))

print('\n25 aplicaciones afectadas: ' +
      str(df_exodus[(df_exodus['Permissions_count'] < inf_3z) | (df_exodus['Permissions_count'] > sup_3z)]['Name'].value_counts().index.to_list()[:25]) + '\n')

df_exodus[(df_exodus['Permissions_count'] < inf_3z) | (df_exodus['Permissions_count'] > sup_3z)].sort_values(by = ['Permissions_count'], ascending = False).head(5)
```

```
Umbrales valores extremos: (-27.46447592191764, 57.5995356323357)
Registros con valores extremos: 1363

Permisos | Cuenta apps
65    74
58    67
61    55
64    48
62    46
Name: Permissions_count, dtype: int64
...
156    1
139    1
189    1
130    1
121    1
211    1
244    1
217    1
228    1
237    1
201    1
142    1
243    1
213    1
222    1
220    1
193    1
149    1
148    1
224    1
194    1
226    1
195    1
160    1
436    1
Name: Permissions_count, dtype: int64

25 aplicaciones afectadas: ['Google', 'TikTok', 'Signal', 'Signal Private Messenger', 'Google Play services', 'Phone', 'Viber Messenger', 'SHAREit - Transfer & Share', 'ManageEngine MDM', 'Viber', 'Microsoft Launcher', 'Truecaller: Caller ID, spam blocking & call record', 'MacroDroid - Device Automation', 'Google Settings', 'Messenger', 'Private Zone - AppLock, Video & Photo Vault', 'SHAREit', 'Avast Mobile Security', 'Xender', 'Whoscall – The best caller ID and block App', 'POCO Launcher', 'UC Browser', 'TextNow', 'GO Launcher - 3D parallax Themes & HD Wallpapers', 'Whoscall']
```

Out[42]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLO
106619	24986	MoChat	12	436	2.4.2	1	1	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	
14340	25603	Dr. Clone	4	244	1.5	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
17573	31582	Dr. Clone 64Bit Support	0	243	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
39669	66878	Dr.Clone: Parallel Accounts, Dual App, 2nd Acc...	2	237	1.5	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
39667	66876	Dr. Clone 64Bit Support	0	234	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

De los *1.363* registros con valores que superan el umbral *más de 57 permisos*, vemos que la cuenta agrupada de aplicaciones por número de rastreadores parece marcar una tendencia coherente al principio, sobre el número de permisos próximos al umbral, pero en valores extremos, es más fácil encontrar casuísticas particulares (1 o 2 ocurrencias de ese número elevado de permisos). Esto podría ser una señal de que los datos extremos se deban a algún error de adquisición.

Vemos en esta muestra de 25 aplicaciones afectadas por valores extremos, empresas bastante reputadas como Google o Microsoft y aplicaciones de tipo escritorio *Launcher* que son las que más permisos necesitan. Esto puede indicar que el umbral superior puede haberse quedado corto, pero no sería un mal candidato para imputar en las aplicaciones que superan el umbral porque ese *57 permisos* parece a todas luces legítimo.

Si observamos los casos más extremos en el origen de los datos, vemos como por ejemplo en la aplicación *MoChat* con *436 permisos demandados* https://reports.exodus-privacy.eu.org/es/reports/info.cloneapp.mochat.in.goast/latest/ hay muchos repetidos que el web scraper tomó como legítimos.

Así pues, **se opta por corregir** los valores de estos registros utilizando el valor límite de tres veces la suma de la desviación típica a la media de permisos redondeada.

In [43]:
```
df_exodus.loc[df_exodus['Permissions_count'] > sup_3z, 'Permissions_count'] = int(np.floor(sup_3z))
```

Permissions_warning_count

Obtenemos los umbrales inferior y superior para discriminar los valores alejados más de tres desviaciones típicas de la media.

In [44]:
```
z = df_exodus['Permissions_warning_count'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Permissions_warning_count'].mean() - (3 * z)
sup_3z = df_exodus['Permissions_warning_count'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Permissions_warning_count'] < inf_3z) | (df_exodus['Permissions_warning_count'] > sup_3z)].shape[0]))

#Observar los valores outliers
print('\nPermisos críticos | Cuenta apps\n' +
      str(df_exodus[(df_exodus['Permissions_warning_count'] < inf_3z) | (df_exodus['Permissions_warning_count'] > sup_3z)]['Permissions_warning_count'].value_counts().sort_values(ascending = False)) + '\n')

print('\n25 aplicaciones afectadas: ' +
      str(df_exodus[(df_exodus['Permissions_warning_count'] < inf_3z) | (df_exodus['Permissions_warning_count'] > sup_3z)]['Name'].value_counts().index.to_list()[:25]) + '\n')

df_exodus[(df_exodus['Permissions_warning_count'] < inf_3z) | (df_exodus['Permissions_warning_count'] > sup_3z)].sort_values(by = ['Permissions_warning_count'], ascending = False).head(5)
```

```
Umbrales valores extremos: (-4.565838872333073, 9.052891044197596)
Registros con valores extremos: 1973

Permisos críticos | Cuenta apps
11    585
10    456
13    291
12    269
15    147
14     99
17     54
16     34
21     29
20      3
22      2
19      1
23      1
28      1
29      1
Name: Permissions_warning_count, dtype: int64


25 aplicaciones afectadas: ['Google', 'Signal', 'Signal Private Messenger', 'Messenger – Text and Video Chat for Free', 'Google Play services', 'Messages', 'Phone', '\ufeffSkype – free IM & video calls', 'Messenger', 'SMS Backup &
Restore', 'All Backup & Restore', 'Truecaller: Caller ID, spam blocking & call record', 'Microsoft Launcher', 'Textra SMS', 'MacroDroid - Device Automation', 'Google Settings', 'Facebook Lite', 'Backup Your Mobile', 'chomp SMS', 'C
aller ID, True Call & Call Blocker: Showcaller', 'Eyecon', 'MyPhoneExplorer Client', 'Contacts+', 'Whoscall – The best caller ID and block App', 'KakaoTalk']
```

Out[44]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOL
15672	28102	FunDo Pro	0	57	1.3.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
40128	67526	Lenovo Life	3	57	1.2.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14012	24992	Parallel Accounts	24	57	2.2.9	1	1	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	
106569	23931	Private Zone - AppLock, Video & Photo Vault	16	57	4.9.3	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOL
5963	10032	INMOTION SCV - Bluetooth	4	57	6.7.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

De los *1.973* registros con valores que superan el umbral *más de 9 permisos críticos*, vemos que la cuenta agrupada de aplicaciones por número de permisos críticos, como en el caso anterior de permisos, parece marcar una tendencia coherente al principio, sobre el número de permisos críticos próximos al umbral, pero en valores extremos, es más fácil encontrar casuísticas particulares. Esto podría ser una señal de que los datos extremos se deban a algún error de adquisición.

También se observa en esta muestra de 25 aplicaciones afectadas por valores extremos, empresas bastante reputadas como Google o Microsoft y aplicaciones de tipo escritorio *Launcher*, *Backups*, *Configuracion* que son las que más permisos críticos necesitan. Esto puede indicar que el umbral superior puede haberse quedado corto, pero no sería un mal candidato para imputar en las aplicaciones que superan el umbral porque esos *11 permisos* que reclaman la mayoría de las aplicaciones que superan el umbral, parece a todas luces legítimo.

Si observamos los casos más extremos en el origen de los datos, vemos como por ejemplo en la aplicación *FunDo Pro* con *29 permisos críticos demandados* https://reports.exodus-privacy.eu.org/es/reports/com.kct.fundo.btnotification/latest/ hay algunos repetidos que el web scraper tomó como legítimos.

Así pues, **se opta por corregir** los valores de estos registros utilizando en este caso la moda que establece en *11 los permisos críticos*.

In [45]:
```
import statistics as stat

df_exodus.loc[df_exodus['Permissions_warning_count'] > sup_3z,
              'Permissions_warning_count'] = stat.mode(df_exodus[(df_exodus['Permissions_warning_count'] < inf_3z) |
                                                  (df_exodus['Permissions_warning_count'] > sup_3z)]['Permissions_warning_count'].value_counts().index)
```

Rating Count

Obtenemos los umbrales inferior y superior para discriminar los valores alejados más de tres desviaciones típicas de la media.

In [46]:
```
z = df_exodus['Rating Count'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Rating Count'].mean() - (3 * z)
sup_3z = df_exodus['Rating Count'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Rating Count'] < inf_3z) | (df_exodus['Rating Count'] > sup_3z)].shape[0]))

#Observar los valores outliers
print('\nValoraciones | Cuenta apps\n' +
      str(df_exodus[(df_exodus['Rating Count'] < inf_3z) | (df_exodus['Rating Count'] > sup_3z)]['Rating Count'].value_counts().sort_values(ascending = False)) + '\n')

print('\n25 aplicaciones afectadas: ' +
      str(df_exodus[(df_exodus['Rating Count'] < inf_3z) | (df_exodus['Rating Count'] > sup_3z)]['Name'].value_counts().index.to_list()[:25]) + '\n')

df_exodus[(df_exodus['Rating Count'] < inf_3z) | (df_exodus['Rating Count'] > sup_3z)].sort_values(by = ['Rating Count'], ascending = False).head(5)
```

```
Umbrales valores extremos: (-17546331.517449204, 19090065.387072098)
Registros con valores extremos: 834

Valoraciones | Cuenta apps
20845533.0000     101
125380770.0000     75
88579975.0000      70
109593315.0000     67
28207013.0000      62
105279686.0000     57
32425599.0000      56
75418942.0000      54
24286765.0000      50
26444937.0000      44
30472376.0000      44
21758332.0000      38
29910196.0000      24
54218724.0000      19
34519744.0000      17
20048123.0000      16
33822865.0000      14
68507088.0000      10
29944311.0000       9
24108473.0000       7
Name: Rating Count, dtype: int64


25 aplicaciones afectadas: ['YouTube', 'Instagram', 'Facebook', 'TikTok', 'Snapchat', 'Spotify: Free Music and Podcasts Streaming', 'WhatsApp Messenger', 'Chrome', 'WhatsApp', 'Messenger – Text and Video Chat for Free', 'Google Pla
```

y services', 'Spotify', 'Photos', 'Candy Crush Saga', 'Spotify: Free Music Streaming', 'Google Settings', 'Clash of Clans', 'Messenger', '8 Ball Pool', 'Google Photos', 'UC Browser', 'PUBG MOBILE', 'Subway Surfers', 'Clash Royale', 'UC Browser – Video Downloader &Win Cash & Vouchers']

Out[46]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOL
83480	125610	WhatsApp	1	56	2.20.123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
85432	128048	WhatsApp	1	56	2.20.140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6275	10527	WhatsApp Messenger	1	51	2.18.156	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
81339	122748	WhatsApp	1	56	2.20.108	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
46646	76955	WhatsApp Messenger	1	54	2.19.150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Minimum Installs

Análogamente realizamos el mismo estudio para el número de instalaciones mínimas

In [47]:

```
z = df_exodus['Minimum Installs'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Minimum Installs'].mean() - (3 * z)
sup_3z = df_exodus['Minimum Installs'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Minimum Installs'] < inf_3z) | (df_exodus['Minimum Installs'] > sup_3z)].shape[0]))

#Observar los valores outliers
print('\nInstalaciones | Cuenta apps\n' +
      str(df_exodus[(df_exodus['Minimum Installs'] < inf_3z) | (df_exodus['Minimum Installs'] > sup_3z)]['Minimum Installs'].value_counts().sort_values(ascending = False)) + '\n')

print('\n25 aplicaciones afectadas: ' +
      str(df_exodus[(df_exodus['Minimum Installs'] < inf_3z) | (df_exodus['Minimum Installs'] > sup_3z)]['Name'].value_counts().index.to_list()[:25]) + '\n')

df_exodus[(df_exodus['Minimum Installs'] < inf_3z) | (df_exodus['Minimum Installs'] > sup_3z)].sort_values(by = ['Minimum Installs'], ascending = False).head(5)
```

```
Umbrales valores extremos: (-1238175702.7226748, 1343169343.9234252)
Registros con valores extremos: 600

Instalaciones | Cuenta apps
5000000000.0000     544
10000000000.0000     56
Name: Minimum Installs, dtype: int64


25 aplicaciones afectadas: ['YouTube', 'Google', 'Facebook', 'Gmail', 'WhatsApp Messenger', 'Chrome', 'WhatsApp', 'Google Play services', 'Maps', 'Drive', 'Google Drive', 'Google Settings', 'Maps - Navigate & Explore', 'Google Play Music', 'Google Play Movies', 'Google Play Movies & TV', 'Google Text-to-speech Engine', 'Google Text-to-Speech', 'Google Chrome: Fast & Secure', 'Android Accessibility Suite', 'Google Text-to-speech', 'Maps - Navigation & Transit', 'UnifiedNlp', 'Voice Search', 'Google TalkBack']
```

Out[47]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLON
56857	91007	Google Play services	3	57	18.7.19	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
6166	10357	Google Settings	3	57	12.6.85	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
19627	35236	Google Settings	3	57	14.3.66	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLON
92623	136928	Google Play services	3	57	20.21.17	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
93858	138571	Google Play services	3	57	20.24.14	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

Como se puede apreciar, las dos aplicaciones, con más valoraciones y con más instalaciones corresponden a dos gigantes de la plataforma Android.

En el caso del **número de valoradores**, la aplicación que más usuarios han valorado es Whatsapp y los datos observados en la Google Play Store, https://play.google.com/store/apps/details?id=com.whatsapp&hl=es&gl=US, coinciden por lo que es un dato legítimo.
El resto de aplicaciones que aparecen en esa muestra de 25, se puede ver que son aplicaciones de mensajería, multimedia y juegos archiconocidas por lo que lógico que el número de valoradores sea elevado.

Acerca del **número de instalaciones**, Whatsapp también aparece en el primer grupo con más número de instalaciones "extremas" *5.000 millones* por lo que en este caso el dato también es legítimo.
Pero, como aparece en la muestra obtenida, la aplicación más instalada es *Google Play Services* el cual es un elemento que viene preinstalado de serie en los móviles de Android y podría ser coherente. Revisando no obstante la información de la tienda de aplicaciones de Google, https://play.google.com/store/apps/details?id=com.google.android.gms&hl=es_419&gl=US se refrenda ese dato de *10.000 millones de instalaciones*.

Se decide **no tratar estos valores extremos por su legitimidad**, pero en ciertos estudios sensibles a valores extremos, se recomendaría utilizar el atributo categorizado *Installs*.

Ratio_valoracion

In [48]:
```
z = df_exodus['Ratio_valoracion'].std() #Cálculo de la desviación típica

#Cálculo aplicado de los umbrales
inf_3z = df_exodus['Ratio_valoracion'].mean() - (3 * z)
sup_3z = df_exodus['Ratio_valoracion'].mean() + (3 * z)
print('Umbrales valores extremos: ' + str((inf_3z, sup_3z)) + '\n' +
      'Registros con valores extremos: ' + str(df_exodus[(df_exodus['Ratio_valoracion'] < inf_3z) | (df_exodus['Ratio_valoracion'] > sup_3z)].shape[0]))

#Observar los valores outliers
print(str(df_exodus[df_exodus['Ratio_valoracion'] > 1][['Rating Count','Minimum Installs','Ratio_valoracion']]) + '\n')

#El número de instalaciones es inferior al de la cuenta de votantes. Hay que arreglar estos valores imputando en la cuenta de votantes el número de instalaciones
#(No podrían votar aquellos que no han instalado) y se recalcula la columna Ratio_valoracion
df_exodus.loc[df_exodus['Ratio_valoracion'] > 1, 'Rating Count'] = df_exodus[df_exodus['Ratio_valoracion'] > 1]['Minimum Installs']
df_exodus['Ratio_valoracion'] = df_exodus['Rating Count']/df_exodus['Minimum Installs']
df_exodus.loc[df_exodus['Ratio_valoracion'].isna(), 'Ratio_valoracion'] = 0
```

```
Umbrales valores extremos: (nan, nan)
Registros con valores extremos: 0
        Rating Count  Minimum Installs  Ratio_valoracion
3463   11265953.0000      5000000.0000            2.2532
7165        487.6000          100.0000            4.8760
7365   11265953.0000      5000000.0000            2.2532
7806       6115.6000          100.0000           61.1560
8997     401595.6000       100000.0000            4.0160
...              ...              ...               ...
109494   105036.0000          100.0000         1050.3600
109534   105036.0000          100.0000         1050.3600
109796       15.0000           10.0000            1.5000
110096       15.4000            5.0000            3.0800
110170       15.0000           10.0000            1.5000

[80 rows x 3 columns]
```

Este atributo se calculó a partir de la división entre *Minimum Installs* y *Rating Count*, entendiendo que para votar te has de haber instalado la aplicación previamente. De esta manera, los valores deben estar comprendidos entre 0 y 1.
Así, en este caso un valor anómalo es aquel que supera la unidad. En estos casos lo que hacemos es **corregir el dato extremo** igualando los votantes al número de instalaciones y volver a regenerar el campo como se hizo en el apartado de valores perdidos.

3.2.2 Atributos categóricos

Obteniendo un resumen de las distintas categorías presentes en las variables discretas, podemos obtener información acerca de posibles valores incoherentes que podrían candidatos a un tratamiento específico para su corrección.

Version

Representamos las 10 primeras y últimas agrupaciones del atributo Version. En éste se puede apreciar que son muchas las aplicaciones que comienzan su numeración en *1* y menos aquellas con versiones más avanzadas.
También llama la atención que, aparte del criterio que seguimos cuando imputamos los valores nulos utilizando simples números naturales, no existe un criterio único para expresar el versionado. Unos desarrolladores optan por varios niveles en el versionado, otros utilizan fechas, otros ambos criterios...

La conclusión es que éste es un atributo más bien descriptivo que categórico que únicamente va a aportar información de la aplicación, pero no aportará una información muy relevante en el análisis, por lo que **no se va a adoptar una revisión de sus valores extremos**.

In [49]:
```
pd.DataFrame(df_exodus['Version'].value_counts().head(10).append(df_exodus['Version'].value_counts().tail(10)))
```

Out[49]:

	Version
1.0	2071
1	1204
1.1	984
1.2	862
2.0	687
1.3	644
1.0.1	604
1.4	564
1.0.0	494
1.5	491
194.0.0.27.193	1
1.11.1.15	1
11.9.75	1
3.1.2281	1
3.2.8007	1
1.891804210	1
4.2.1.0.10	1
0.22.7	1
3.6.01.5	1
7.1.20.9.388	1

Country

Cabe recordar, que en el apartado de estudio de los valores nulos, ya realizamos un tratamiento de los valores de esta variable para categorizar con el criterio que adoptaban el mayor número de registros que es el formato ISO de dos caracteres.
Se han obtenido un total de *168 países* de procedencia de aplicaciones, siendo la mayor parte de ellos indeterminado bajo el estándar *COM*.

In [50]:
```
#Total de categorías
print('Número total de categorías: ' + str(len(df_exodus['Country'].value_counts())) + '\n')

#Muestra de
pd.DataFrame(df_exodus['Country'].value_counts()).head(5).append(pd.DataFrame(df_exodus['Country'].value_counts()).tail(5))
```

```
Número total de categorías: 168
```

Out[50]:

	Country
COM	38854
US	29516
DE	5513
GB	3057
FR	2930
SB	1
GP	1
GM	1
BS	1
GD	1

Category

Existen un total de *48 categorías* de aplicaciones y todas ellas son representativas con datos de calidad.
La categoría de aplicaciones más extendida es para herramientas y en menor medida aplicaciones de belleza.

Los datos consignados son correctos y **no se realizará actuaciones sobre ellos**.

In [51]:
```
#Total de categorías
print('Número total de categorías: ' + str(len(df_exodus['Category'].value_counts())) + '\n')

print(str(df_exodus['Category'].unique()) + '\n')

pd.DataFrame(df_exodus['Category'].value_counts()).head(5).append(pd.DataFrame(df_exodus['Category'].value_counts()).tail(5))
```

```
Número total de categorías: 48

['Maps & Navigation' 'Entertainment' 'Tools' 'Finance' 'News & Magazines'
 'Auto & Vehicles' 'Communication' 'Shopping' 'Weather' 'Lifestyle'
 'Productivity' 'Social' 'Health & Fitness' 'Travel & Local' 'Dating'
 'Business' 'Sports' 'Music & Audio' 'Food & Drink' 'Casual' 'Strategy'
 'Photography' 'Books & Reference' 'Medical' 'Word'
 'Video Players & Editors' 'Education' 'Parenting' 'Arcade' 'Comics'
 'Role Playing' 'Casino' 'House & Home' 'Personalization' 'Card' 'Events'
 'Simulation' 'Action' 'Puzzle' 'Trivia' 'Music' 'Libraries & Demo'
 'Racing' 'Adventure' 'Board' 'Educational' 'Art & Design' 'Beauty']
```

Out[51]:

	Category
Tools	18414
Productivity	10075
Communication	6091
Education	5645
Finance	5443
Comics	142
Music	136
Events	124
Casino	103
Beauty	98

Installs

Existen un total de *22* categorías para clasificar el número de dispositivos donde se encuentra instalada la aplicación y todas ellas son representativas con datos de calidad.
La mayor parte de instalaciones se encuentran en *1 millón* y *100.000* dispositivos y hay un número pequeño de aplicaciones Beta presentes en *0 dispositivos*.

También observamos aquí el dato extremo de *10.000 millones* observado en el tratamiento de *Minimum Installs* por lo que se aconseja utilizar esta categorización en el caso de necesidad de tener en cuenta el número de dispositivos.

Los datos consignados son correctos y **no se realizará actuaciones sobre ellos**.

In [52]:
```
#Total de categorías
print('Número total de instalaciones: ' + str(len(df_exodus['Installs'].value_counts())) + '\n')

print(str(df_exodus['Installs'].unique()) + '\n')

pd.DataFrame(df_exodus['Installs'].value_counts()).head(5).append(pd.DataFrame(df_exodus['Installs'].value_counts()).tail(5))
```

```
Número total de instalaciones: 22

['100,000+' '5,000,000+' '1,000,000+' '10,000,000+' '500,000,000+'
 '100,000,000+' '500,000+' '50,000,000+' '1,000,000,000+' '10,000+'
 '5,000,000,000+' '50,000+' '5,000+' '1,000+' '100+' '500+' '10+'
 '10,000,000,000+' '50+' '1+' '5+' '0+']
```

Out[52]:

	Installs
1,000,000+	21902
100,000+	20108
10,000,000+	13285
10,000+	11131

	Installs
500,000+	9757
50+	268
5+	69
1+	66
10,000,000,000+	56
0+	17

Content Rating

Existen un total de *5* categorías para catalogar el contenido de las aplicaciones. Todas ellas son representativas con datos de calidad.
La mayor parte de aplicaciones son aptas para todo el mundo y hay solamente *2* catalogadas para adultos.

Los datos consignados son correctos y **no se realizará actuaciones sobre ellos**.

In [53]:
```
print('Catalogacion de contenido: ' + str(len(df_exodus['Content Rating'].value_counts())) + '\n')

print(str(df_exodus['Content Rating'].unique()) + '\n')

pd.DataFrame(df_exodus['Content Rating'].value_counts().head(5))
```

```
Catalogacion de contenido: 5

['Everyone' 'Mature 17+' 'Teen' 'Everyone 10+' 'Adults only 18+']
```

Out[53]:

	Content Rating
Everyone	95712
Teen	9495
Mature 17+	2756
Everyone 10+	2401
Adults only 18+	2

Financiacion

Esta es el atributo que creamos en el proceso de selección para clasificar el tipo de financiación de la aplicación a partir de tres atributos existentes previamente.

Se recuerdan aquí las categorías creadas y **no se realizarán actuaciones sobre sus datos**.

In [54]:
```
print('Financiación: ' + str(len(df_exodus['Financiacion'].value_counts())) + '\n')

print(str(df_exodus['Financiacion'].unique()) + '\n')

pd.DataFrame(df_exodus['Financiacion'].value_counts().head(6))
```

```
Financiación: 6

['Gratis' 'AnunciosPagoInApp' 'PagoInApp' 'Anuncios' 'Copago' 'Pago']
```

Out[54]:

	Financiacion
Gratis	41758
AnunciosPagoInApp	28844
Anuncios	22107
PagoInApp	15633
Pago	1487
Copago	537

Released, Last Updated, Analysis_date

Los estadísticos del análisis descriptivo en los datos contenidos en los tres atributos de fechas, no arrojan ningún valor que pueda destacar por ser extremo.

No obstante, sí existen ciertas incoherencias entre los atributos atendiendo a la realidad que representan:

- Por ejemplo, vemos que la aplicación más joven se lanzó en enero de 2010, pero la primera de las actualizaciones data de casi un mes antes
- También se observa que al menos el análisis de seguridad de la aplicación más próximo, fue efectuado casi un mes antes del lanzamiento de la aplicación más reciente, lo que es incongruente.
- No obstante, los valores observados en los intervalos intercuartílicos sí que son coherentes en general.

El tratamiento a realizar, consistirá en igualar las fechas para **corregir las incongruencias** detectadas en los dos primeros puntos.

In [55]:
```
pd.DataFrame(df_exodus[['Released','Last Updated', 'Analysis_date']].describe(datetime_is_numeric = True))
```

Out[55]:

	Released	Last Updated	Analysis_date
count	110366	110366	110366
mean	2015-09-03 19:06:49.365217024	2020-05-04 00:43:25.324011520	2019-07-15 10:37:59.789038080
min	2010-01-28 00:00:00	2009-12-22 00:00:00	2012-02-21 00:00:00
25%	2013-07-02 00:00:00	2020-05-14 00:00:00	2018-11-25 00:00:00
50%	2015-11-23 00:00:00	2020-10-21 00:00:00	2019-07-08 00:00:00
75%	2017-12-07 00:00:00	2020-11-18 00:00:00	2020-03-11 00:00:00
max	2020-11-27 00:00:00	2020-12-03 00:00:00	2020-11-08 00:00:00

In [56]:
```
#Volcar la fecha de lanzamiento en la fecha de actualización, en los casos en los que ésta sea anterior a la del propio lanzamiento.
df_exodus.loc[df_exodus['Released'] > df_exodus['Last Updated'], 'Last Updated'] = df_exodus[df_exodus['Released'] > df_exodus['Last Updated']]['Released']

#Volcar la fecha de lanzamiento en la fecha de análisis, en aquellos casos donde el análisis precedía al lanzamiento.
df_exodus.loc[df_exodus['Released'] > df_exodus['Analysis_date'], 'Analysis_date'] = df_exodus[df_exodus['Released'] > df_exodus['Analysis_date']]['Released']
```

Tras las tareas de integración y limpieza efectuadas en los apartados anteriores, ya tenemos listo el **Data Frame Exodus** para su explotación analítica.

In [57]:
```
df_exodus
```

Out[57]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCO
0	1	Tan	3	9	5.7.0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	2	AlloCine	2	7	3.3.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	3	DNS Changer (No Root - IPv6 - 3G/wifi)	0	5	1.15.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	4	La Banque Postale	0	13	5.7.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	6	AlloCine	29	20	7.2.2	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
110361	153088	myTU	1	15	1.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110362	153101	LibreSpeed	0	1	1.2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110363	153115	Gallery	0	6	1.3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110364	153270	Nebula	0	4	0.0.38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110365	153354	Suntimes	0	8	0.13.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

110366 rows × 90 columns

4. Análisis de los datos

4.1. Comparación de grupos de datos por contraste de hipótesis

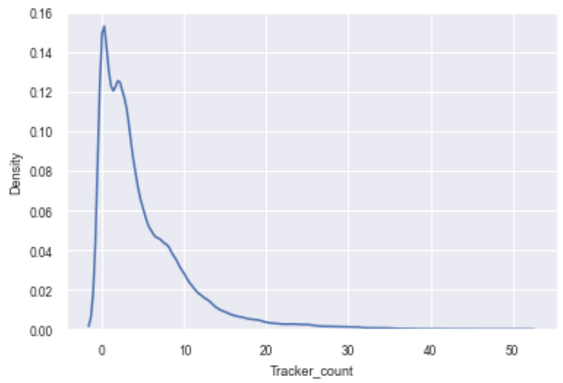
4.1.1. Comparación entre dos grupos de datos

Se va a realizar un primer análisis, estadístico inferencial, para poder **comparar dos grupos de datos**. En concreto, el análisis se centra en determinar si la inclusión de *rastreadores* como elemento susceptible de invadir de la privacidad o recopilar datos de usuario, influye en la evaluación de las aplicaciones por parte del usuario. Es decir, conocer si colectivamente existe algún tipo de "defensa" contra las aplicaciones invasivas utilizando la nota de la aplicación como disuasión para nuevos usuarios.

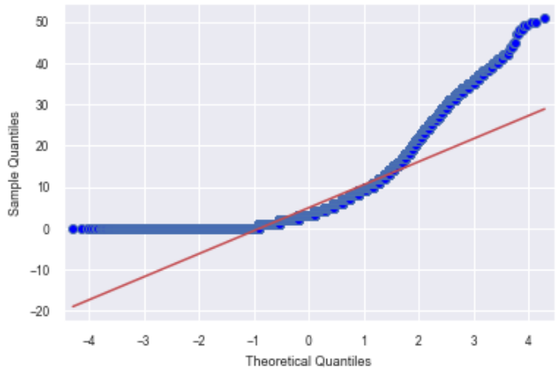
Para el estudio vamos a utilizar del Data Frame *Exodus* los atributos *Tracker_count* y *Rating*

Comprobación de la normalidad Se comprueba si la variable a comprobar cumple la condición de normalidad

In [58]:
```
#Visualizar la distribución de la gráfica
sns.kdeplot(df_exodus['Tracker_count'])
plt.show()

print('Número de datos: ' + str(len(df_exodus['Tracker_count'])))

#Comprobación de normalidad mediante Saphiro-Wilks
from scipy.stats import shapiro
estad, p = shapiro(df_exodus['Tracker_count'])
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
#Interpretación del test
alpha = 0.05
if p > alpha:
    print('\tLa muestra parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Comprobación visual
from statsmodels.graphics.gofplots import qqplot
qqplot(df_exodus['Tracker_count'], line='s')
plt.show()
```




```
Número de datos: 110366
Estadístico = 0.786,
p-value = 0.000
        La muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)

D:\Programas\Anaconda3\lib\site-packages\scipy\stats\morestats.py:1676: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```



Tanto el test de Saphiro-Wilk como el gráfico QQ nos indican que los datos no siguen una distribución normal. Pero el test de Saphiro-Wilk nos ha lanzado una advertencia acerca del tamaño de la muestra. https://www.linkedin.com/pulse/when-big-data-too-much-good-thing-jon-neff/. En el artículo de Neff, J. nos explica que todo el detalle existente en un volumen de datos muy elevado, va a provocar que los test no encuentren patrón y rechacen siempre la hipótesis nula.
Para tratar de resolver este problema, vamos a hacer una agregación de los datos de las aplicaciones por su identificador (recordando que el Data Frame contiene distintas versiones de la misma aplicación) y calcular una media de los rastreadores utilizados y las votaciones del agregado de versiones.

In [59]:
```
#Agrupación de los datos por el identificador de aplicación y cálculo de las medias de rastreadores y nota.
df_analisis = df_exodus.groupby('App Id', as_index=True)[['Tracker_count', 'Rating']].mean()

print('Número de datos: ' + str(len(df_analisis['Tracker_count'])))
```

```
Número de datos: 47121
```

Vemos que la agregación construye un dataset de *47.121* aplicaciones, con lo que optaremos por utilizar el test de Kolmogorov-Smirnov en su versión asintótica.
Lazariv, T. y Lehmann C. "*Goodness-of-Fit Tests for Large Datasets*"

In [60]:
```
#Comprobación de normalidad mediante Kolmogorov-Smirnov
from scipy.stats import kstest

estad, p = kstest(df_analisis['Tracker_count'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))

#Interpretación del test
alpha = 0.05
if p > alpha:
    print('\tLa muestra parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')
```

```
Estadístico = 0.628,
p-value = 0.000
        La muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)
```

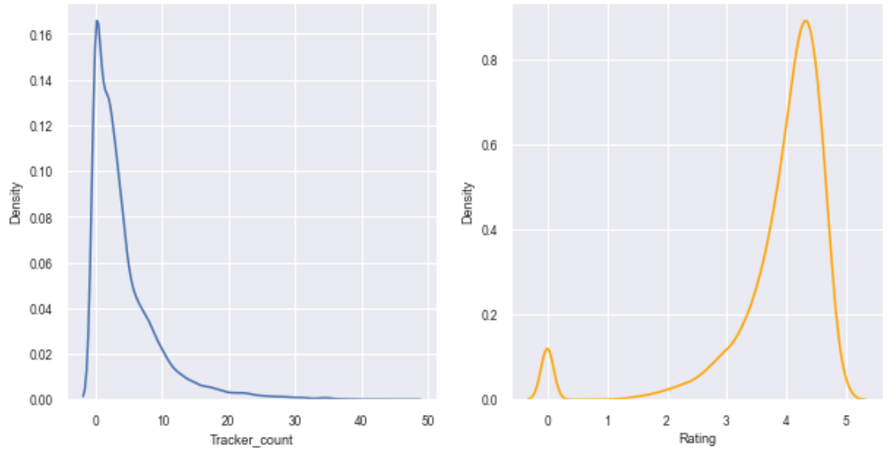
La interpretación visual de la distribución y el gráfico QQ ya nos anticipaba que no seguiría una distribución normal y el test lo confirma.

De esta manera veo innecesario calcular la homocedasticidad puesto que nos vemos obligados a recurrir al test **no paramétrico Wilcoxon** para estudiar las diferencias entre ambas variables.

In [61]:
```
from scipy.stats import wilcoxon

estad, p = wilcoxon(df_analisis['Tracker_count'], df_analisis['Rating'])
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))

#Interpretación del test
alpha = 0.05
if p > alpha:
    print('\tLa distribución de ambas variables es similar y NO presentan diferencias significativas (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tSÍ existen diferencias significativas en la distribución de los grupos analizados (rechazamos la hipótesis nula H0)\n')
```

```
Estadístico = 481204106.000,
p-value = 0.000
        SÍ existen diferencias significativas en la distribución de los grupos analizados (rechazamos la hipótesis nula H0)
```

In [62]:
```
#Visualizar las distribuciones
fig, axs = plt.subplots(nrows = 1,ncols = 2,figsize=(10,5))

sns.kdeplot(df_analisis['Tracker_count'], ax = axs[0])
sns.kdeplot(df_analisis['Rating'], ax = axs[1], color = 'orange')
plt.show()
```



Observando ambas distribuciones, es posible ver que no iban a seguir la distribución normal y además son muy diferentes.

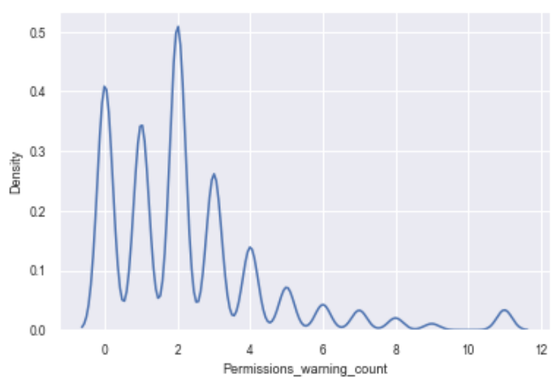
El test nos confirma que son diferentes, por lo que no por tener menor número de rastreadores se evalúa peor, pero tampoco se están evaluando mal las aplicaciones con mayor número de rastreadores. Es decir, **el número de rastreadores no es un factor que para el público influya en rebajar la nota**.

4.1.2. Comparación entre más de dos grupos

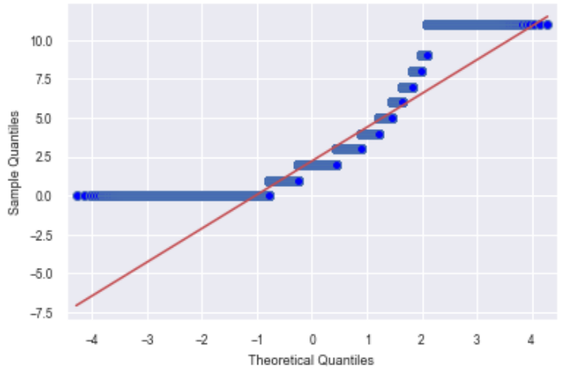
El segundo análisis quiere **comparar más de dos grupos de datos**. En concreto, el análisis se centra en determinar la inclusión de permisos potencialmente críticos dependiendo del tipo de aplicación. Es decir, conocer si la petición de recursos críticos del dispositivo se está produciendo en aplicaciones concretas (juegos, salud, herramientas, etc.) o bien se está produciendo esta supuesta invasión del dispositivo en cualquier tipo de aplicación para llegar a más usuarios.

Para el estudio vamos a utilizar del Data Frame *Exodus* los atributos *Permission_warning_count* y *Category*

In [63]:
```
#Visualizar la distribución de la gráfica
sns.kdeplot(df_exodus['Permissions_warning_count'])
plt.show()

print('Número de datos: ' + str(len(df_exodus['Permissions_warning_count'])))

#Comprobación de normalidad mediante Kolmogorov-Smirnov
estad, p = kstest(df_exodus['Permissions_warning_count'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))

#Interpretación del test
alpha = 0.05
if p > alpha:
    print('\tLa muestra parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Comprobación visual
from statsmodels.graphics.gofplots import qqplot
qqplot(df_exodus['Permissions_warning_count'], line='s')
plt.show()
```




```
Número de datos: 110366
Estadístico = 0.623,
p-value = 0.000
        La muestra parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)
```



La interpretación visual de la distribución y el gráfico QQ ya nos anticipaba que no seguiría una distribución normal y el test lo confirma.

De esta manera veo innecesario calcular la homocedasticidad puesto que nos vemos obligados a recurrir al test **no paramétrico Kruskal-Walls** para estudiar las diferencias entre los grupos de datos.

Para realizar este test, sí que tendremos que proporcionar a función los datos de permisos críticos separados por cada una de las *48 categorías*.

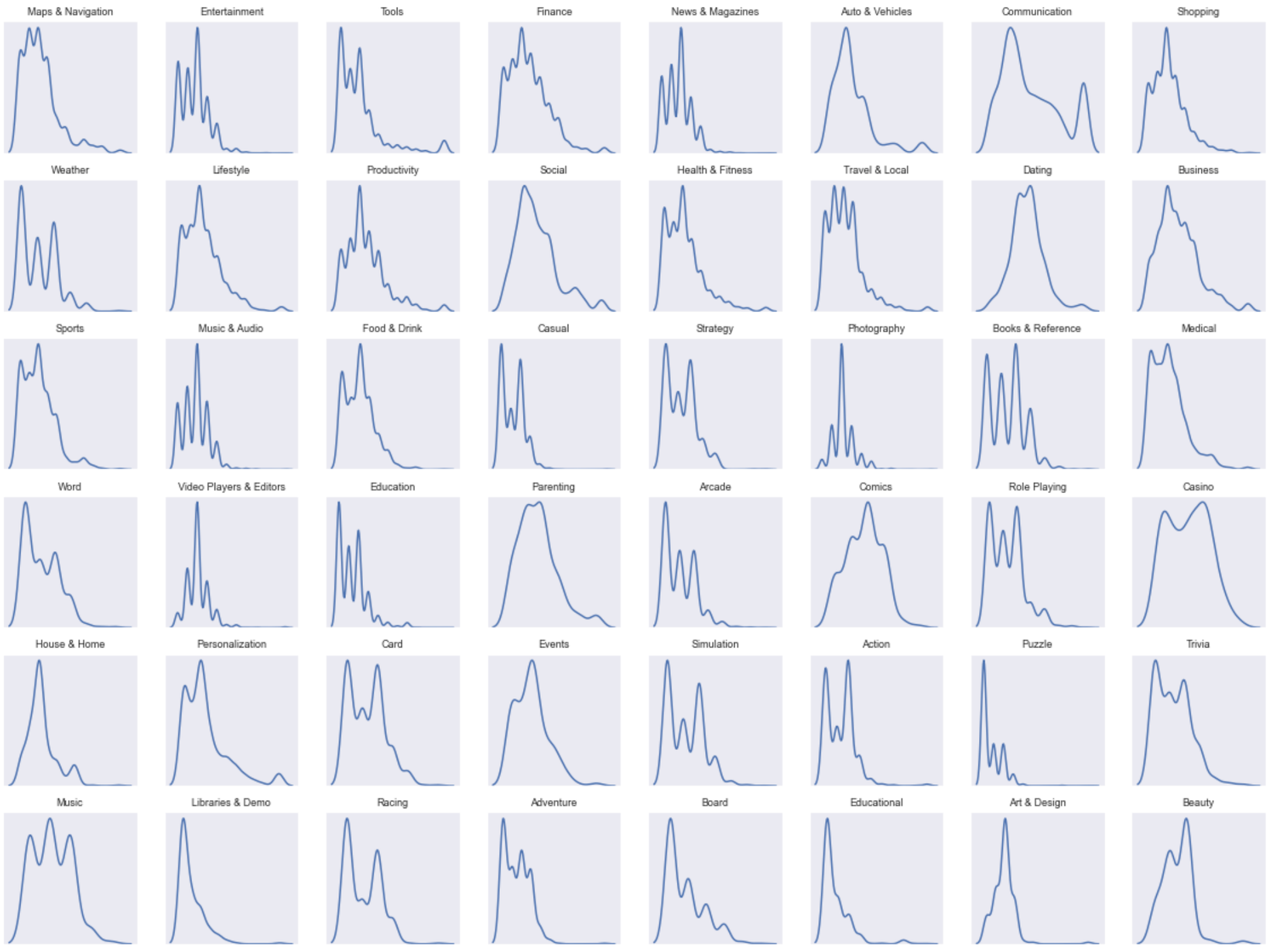
In [64]:
```
from scipy.stats import kruskal

estad,p = kruskal(*[group['Permissions_warning_count'].values for name, group in df_exodus.groupby('Category')])
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))

#Interpretación del test
alpha = 0.05
if p > alpha:
    print('\tLa distribución de ambas variables es similar y NO presentan diferencias significativas (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tSÍ existen diferencias significativas en la distribución de los grupos analizados (rechazamos la hipótesis nula H0)\n')
```

```
Estadístico = 13835.109,
p-value = 0.000
        SÍ existen diferencias significativas en la distribución de los grupos analizados (rechazamos la hipótesis nula H0)
```

El test de Kruskal, nos ha dicho que efectivamente hay diferencias en la distribución de permisos críticos en los grupos, es decir, que no todos los tipos de aplicaciones deberían requerir el mismo número de permisos críticos.

No obstante, la función no nos indica las diferencias entre los distintos tipos, por curiosidad, se pueden ver las distribuciones de permisos críticos en las distintas categorías, y hacer un examen Kruskal entre todas las categorías para ver entre cuáles no se presentarían diferencias.

In [65]:
```
#Visualización de la distribución de permisos críticos según categorías de aplicaciones
fig, axs = plt.subplots(nrows = 6, ncols = 8, figsize=(20,15))
for num, cat in enumerate(df_exodus['Category'].unique()):
    sns.kdeplot(df_exodus[df_exodus['Category'] == cat]['Permissions_warning_count'], ax = axs[(num // 8)][(num % 8)])
    axs[(num // 8)][(num % 8)].set_title(cat)
    axs[(num // 8)][(num % 8)].get_xaxis().set_visible(False)
    axs[(num // 8)][(num % 8)].get_yaxis().set_visible(False)
plt.show()

#Realizar el estudio de entre los grupos de categorías dos a dos para ver cuáles no presentan diferencias.
analizadas = []
for catA in df_exodus['Category'].unique():
    for catB in df_exodus['Category'].unique():
        if (catB not in analizadas) and (catA != catB):
            estad, p = kruskal(df_exodus[df_exodus['Category'] == catA]['Permissions_warning_count'], df_exodus[df_exodus['Category'] == catB]['Permissions_warning_count'])
            if p > 0.05:
                print('Las diferencias de permisos críticos de las categorías: ' + catA + ' y ' + catB + ', no serían significativas: p = ' + str(p))
        analizadas.append(catA)
```



Las diferencias de permisos críticos de las categorías: Maps & Navigation y Health & Fitness, no serían significativas: p = 0.8124200489933976
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Travel & Local, no serían significativas: p = 0.9754340832439377
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Sports, no serían significativas: p = 0.07510554414347377
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Medical, no serían significativas: p = 0.3656183311047084
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Comics, no serían significativas: p = 0.1538100128495566
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Casino, no serían significativas: p = 0.2066186507176992
Las diferencias de permisos críticos de las categorías: Maps & Navigation y House & Home, no serían significativas: p = 0.06515981016544989
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Personalization, no serían significativas: p = 0.11335760026440396
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Events, no serían significativas: p = 0.22126811998955018
Las diferencias de permisos críticos de las categorías: Maps & Navigation y Art & Design, no serían significativas: p = 0.21569550427928938
Las diferencias de permisos críticos de las categorías: Entertainment y News & Magazines, no serían significativas: p = 0.546264700947517
Las diferencias de permisos críticos de las categorías: Entertainment y Comics, no serían significativas: p = 0.3166148759903703
Las diferencias de permisos críticos de las categorías: Entertainment y Casino, no serían significativas: p = 0.4294629198303074
Las diferencias de permisos críticos de las categorías: Entertainment y Events, no serían significativas: p = 0.35228568917313074
Las diferencias de permisos críticos de las categorías: Tools y Casino, no serían significativas: p = 0.1943315924707358
Las diferencias de permisos críticos de las categorías: Tools y Events, no serían significativas: p = 0.11219747214466387
Las diferencias de permisos críticos de las categorías: Finance y Dating, no serían significativas: p = 0.24071576646997211
Las diferencias de permisos críticos de las categorías: Finance y Beauty, no serían significativas: p = 0.3487409724996944
Las diferencias de permisos críticos de las categorías: News & Magazines y Comics, no serían significativas: p = 0.3970876369401444
Las diferencias de permisos críticos de las categorías: News & Magazines y Casino, no serían significativas: p = 0.4885939225724778
Las diferencias de permisos críticos de las categorías: News & Magazines y Events, no serían significativas: p = 0.42365432568967276
Las diferencias de permisos críticos de las categorías: Auto & Vehicles y Shopping, no serían significativas: p = 0.30340330952662425
Las diferencias de permisos críticos de las categorías: Auto & Vehicles y Lifestyle, no serían significativas: p = 0.852376001281723
Las diferencias de permisos críticos de las categorías: Auto & Vehicles y Photography, no serían significativas: p = 0.8760603454087417
Las diferencias de permisos críticos de las categorías: Auto & Vehicles y Video Players & Editors, no serían significativas: p = 0.05631768926589857
Las diferencias de permisos críticos de las categorías: Auto & Vehicles y House & Home, no serían significativas: p = 0.43230313675202814

Las diferencias de permisos críticos de las categorías: Shopping y Lifestyle, no serían significativas: p = 0.16541974232160744
Las diferencias de permisos críticos de las categorías: Shopping y Photography, no serían significativas: p = 0.3232673209640431
Las diferencias de permisos críticos de las categorías: Shopping y Video Players & Editors, no serían significativas: p = 0.07974969414510048
Las diferencias de permisos críticos de las categorías: Shopping y House & Home, no serían significativas: p = 0.7388799396557089
Las diferencias de permisos críticos de las categorías: Shopping y Personalization, no serían significativas: p = 0.11496472209929709
Las diferencias de permisos críticos de las categorías: Weather y Word, no serían significativas: p = 0.9630543049710338
Las diferencias de permisos críticos de las categorías: Weather y Arcade, no serían significativas: p = 0.2107020661328333
Las diferencias de permisos críticos de las categorías: Weather y Simulation, no serían significativas: p = 0.06261811967206067
Las diferencias de permisos críticos de las categorías: Weather y Music, no serían significativas: p = 0.44263709924662575
Las diferencias de permisos críticos de las categorías: Weather y Racing, no serían significativas: p = 0.975015121665846
Las diferencias de permisos críticos de las categorías: Lifestyle y Photography, no serían significativas: p = 0.5312607911561926
Las diferencias de permisos críticos de las categorías: Lifestyle y House & Home, no serían significativas: p = 0.3403959105135004
Las diferencias de permisos críticos de las categorías: Productivity y Beauty, no serían significativas: p = 0.1129007932258506
Las diferencias de permisos críticos de las categorías: Social y Parenting, no serían significativas: p = 0.5263585887337972
Las diferencias de permisos críticos de las categorías: Health & Fitness y Travel & Local, no serían significativas: p = 0.8689350108455804
Las diferencias de permisos críticos de las categorías: Health & Fitness y Sports, no serían significativas: p = 0.08830897120730934
Las diferencias de permisos críticos de las categorías: Health & Fitness y Medical, no serían significativas: p = 0.45592098156281713
Las diferencias de permisos críticos de las categorías: Health & Fitness y Comics, no serían significativas: p = 0.15268324838501368
Las diferencias de permisos críticos de las categorías: Health & Fitness y Casino, no serían significativas: p = 0.22410831997902161
Las diferencias de permisos críticos de las categorías: Health & Fitness y House & Home, no serían significativas: p = 0.0675170959813108
Las diferencias de permisos críticos de las categorías: Health & Fitness y Events, no serían significativas: p = 0.23122852396820767
Las diferencias de permisos críticos de las categorías: Health & Fitness y Art & Design, no serían significativas: p = 0.18983876570446015
Las diferencias de permisos críticos de las categorías: Travel & Local y Sports, no serían significativas: p = 0.07255093380926504
Las diferencias de permisos críticos de las categorías: Travel & Local y Medical, no serían significativas: p = 0.3940423800965136
Las diferencias de permisos críticos de las categorías: Travel & Local y Comics, no serían significativas: p = 0.11862206937409958
Las diferencias de permisos críticos de las categorías: Travel & Local y Casino, no serían significativas: p = 0.19931886782434477
Las diferencias de permisos críticos de las categorías: Travel & Local y House & Home, no serían significativas: p = 0.06855202174519795
Las diferencias de permisos críticos de las categorías: Travel & Local y Personalization, no serían significativas: p = 0.0809676471135325
Las diferencias de permisos críticos de las categorías: Travel & Local y Events, no serían significativas: p = 0.20672021980759228
Las diferencias de permisos críticos de las categorías: Travel & Local y Art & Design, no serían significativas: p = 0.1329316605972661
Las diferencias de permisos críticos de las categorías: Dating y Beauty, no serían significativas: p = 0.41563732376978413
Las diferencias de permisos críticos de las categorías: Business y Parenting, no serían significativas: p = 0.27848114334702534
Las diferencias de permisos críticos de las categorías: Business y Beauty, no serían significativas: p = 0.1716230223879745
Las diferencias de permisos críticos de las categorías: Sports y Music & Audio, no serían significativas: p = 0.13042122461953282
Las diferencias de permisos críticos de las categorías: Sports y Food & Drink, no serían significativas: p = 0.4495941636011487
Las diferencias de permisos críticos de las categorías: Sports y Medical, no serían significativas: p = 0.5101183715677243
Las diferencias de permisos críticos de las categorías: Sports y Comics, no serían significativas: p = 0.3834700342956586
Las diferencias de permisos críticos de las categorías: Sports y Casino, no serían significativas: p = 0.4579156632881961
Las diferencias de permisos críticos de las categorías: Sports y Events, no serían significativas: p = 0.5236939251933113
Las diferencias de permisos críticos de las categorías: Sports y Art & Design, no serían significativas: p = 0.6284112333663989
Las diferencias de permisos críticos de las categorías: Music & Audio y Food & Drink, no serían significativas: p = 0.5437845977344449
Las diferencias de permisos críticos de las categorías: Music & Audio y Comics, no serían significativas: p = 0.6753941897658551
Las diferencias de permisos críticos de las categorías: Music & Audio y Casino, no serían significativas: p = 0.849956460952737
Las diferencias de permisos críticos de las categorías: Music & Audio y Events, no serían significativas: p = 0.8265497808683475
Las diferencias de permisos críticos de las categorías: Music & Audio y Art & Design, no serían significativas: p = 0.8400815720966229
Las diferencias de permisos críticos de las categorías: Food & Drink y Medical, no serían significativas: p = 0.16454058065258972
Las diferencias de permisos críticos de las categorías: Food & Drink y Comics, no serían significativas: p = 0.5459864097474125
Las diferencias de permisos críticos de las categorías: Food & Drink y Casino, no serían significativas: p = 0.6915469849602331
Las diferencias de permisos críticos de las categorías: Food & Drink y Events, no serían significativas: p = 0.6966809976585842
Las diferencias de permisos críticos de las categorías: Food & Drink y Art & Design, no serían significativas: p = 0.8412418851286856
Las diferencias de permisos críticos de las categorías: Casual y Strategy, no serían significativas: p = 0.6174760822402363
Las diferencias de permisos críticos de las categorías: Casual y Word, no serían significativas: p = 0.07876763028426968
Las diferencias de permisos críticos de las categorías: Casual y Education, no serían significativas: p = 0.08191202833148467
Las diferencias de permisos críticos de las categorías: Casual y Role Playing, no serían significativas: p = 0.3195725489988859
Las diferencias de permisos críticos de las categorías: Casual y Card, no serían significativas: p = 0.18510419968731467
Las diferencias de permisos críticos de las categorías: Casual y Simulation, no serían significativas: p = 0.3386808148757616
Las diferencias de permisos críticos de las categorías: Casual y Action, no serían significativas: p = 0.12044683046742571
Las diferencias de permisos críticos de las categorías: Casual y Trivia, no serían significativas: p = 0.7293863851851734
Las diferencias de permisos críticos de las categorías: Casual y Music, no serían significativas: p = 0.4362101978063948
Las diferencias de permisos críticos de las categorías: Casual y Racing, no serían significativas: p = 0.05986537980724484
Las diferencias de permisos críticos de las categorías: Strategy y Education, no serían significativas: p = 0.35827055887129344
Las diferencias de permisos críticos de las categorías: Strategy y Role Playing, no serían significativas: p = 0.6347369358835615
Las diferencias de permisos críticos de las categorías: Strategy y Card, no serían significativas: p = 0.3926740312820285
Las diferencias de permisos críticos de las categorías: Strategy y Simulation, no serían significativas: p = 0.16335010912697912
Las diferencias de permisos críticos de las categorías: Strategy y Action, no serían significativas: p = 0.3371327390381542
Las diferencias de permisos críticos de las categorías: Strategy y Trivia, no serían significativas: p = 0.9974011201269423
Las diferencias de permisos críticos de las categorías: Strategy y Music, no serían significativas: p = 0.32256217457026326
Las diferencias de permisos críticos de las categorías: Photography y House & Home, no serían significativas: p = 0.140085631025106
Las diferencias de permisos críticos de las categorías: Books & Reference y Adventure, no serían significativas: p = 0.8681760638177438
Las diferencias de permisos críticos de las categorías: Medical y Comics, no serían significativas: p = 0.2550344133528265
Las diferencias de permisos críticos de las categorías: Medical y Casino, no serían significativas: p = 0.32758833205120785
Las diferencias de permisos críticos de las categorías: Medical y Events, no serían significativas: p = 0.3609075091932633
Las diferencias de permisos críticos de las categorías: Medical y Art & Design, no serían significativas: p = 0.3629314834254217
Las diferencias de permisos críticos de las categorías: Word y Arcade, no serían significativas: p = 0.49704246353172143
Las diferencias de permisos críticos de las categorías: Word y Simulation, no serían significativas: p = 0.25632155167690407
Las diferencias de permisos críticos de las categorías: Word y Trivia, no serían significativas: p = 0.0950008822928119
Las diferencias de permisos críticos de las categorías: Word y Music, no serían significativas: p = 0.4879255009814607
Las diferencias de permisos críticos de las categorías: Word y Racing, no serían significativas: p = 0.9487034245792688
Las diferencias de permisos críticos de las categorías: Video Players & Editors y Casino, no serían significativas: p = 0.06335998144214461
Las diferencias de permisos críticos de las categorías: Video Players & Editors y House & Home, no serían significativas: p = 0.8011474442337044
Las diferencias de permisos críticos de las categorías: Video Players & Editors y Personalization, no serían significativas: p = 0.31488621431438735
Las diferencias de permisos críticos de las categorías: Education y Role Playing, no serían significativas: p = 0.7522004098779677
Las diferencias de permisos críticos de las categorías: Education y Card, no serían significativas: p = 0.7921740545926136

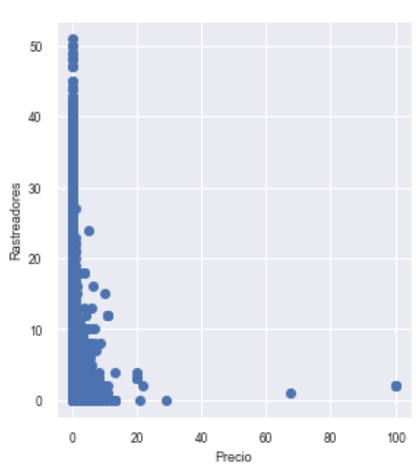
```
Las diferencias de permisos críticos de las categorías: Education y Action, no serían significativas: p = 0.8233949287935965
Las diferencias de permisos críticos de las categorías: Education y Trivia, no serían significativas: p = 0.5839190410462183
Las diferencias de permisos críticos de las categorías: Education y Music, no serían significativas: p = 0.16291208863615128
Las diferencias de permisos críticos de las categorías: Parenting y Beauty, no serían significativas: p = 0.08223771035343945
Las diferencias de permisos críticos de las categorías: Arcade y Music, no serían significativas: p = 0.16260594274033502
Las diferencias de permisos críticos de las categorías: Arcade y Racing, no serían significativas: p = 0.4667083101997903
Las diferencias de permisos críticos de las categorías: Comics y Casino, no serían significativas: p = 0.8508656602536065
Las diferencias de permisos críticos de las categorías: Comics y Personalization, no serían significativas: p = 0.06514644680270947
Las diferencias de permisos críticos de las categorías: Comics y Events, no serían significativas: p = 0.9374730775638858
Las diferencias de permisos críticos de las categorías: Comics y Art & Design, no serían significativas: p = 0.6380326161439038
Las diferencias de permisos críticos de las categorías: Role Playing y Card, no serían significativas: p = 0.6138125966777004
Las diferencias de permisos críticos de las categorías: Role Playing y Action, no serían significativas: p = 0.5634518450929534
Las diferencias de permisos críticos de las categorías: Role Playing y Trivia, no serían significativas: p = 0.7356136737883119
Las diferencias de permisos críticos de las categorías: Role Playing y Music, no serían significativas: p = 0.1995527888310964
Las diferencias de permisos críticos de las categorías: Casino y House & Home, no serían significativas: p = 0.06073225874228106
Las diferencias de permisos críticos de las categorías: Casino y Personalization, no serían significativas: p = 0.10967635405910087
Las diferencias de permisos críticos de las categorías: Casino y Events, no serían significativas: p = 0.977571987220928
Las diferencias de permisos críticos de las categorías: Casino y Art & Design, no serían significativas: p = 0.9657727553809877
Las diferencias de permisos críticos de las categorías: House & Home y Personalization, no serían significativas: p = 0.4462449118087619
Las diferencias de permisos críticos de las categorías: Personalization y Events, no serían significativas: p = 0.09285156602765901
Las diferencias de permisos críticos de las categorías: Personalization y Art & Design, no serían significativas: p = 0.05354598909357832
Las diferencias de permisos críticos de las categorías: Card y Action, no serían significativas: p = 0.8972772857221788
Las diferencias de permisos críticos de las categorías: Card y Trivia, no serían significativas: p = 0.5162367725853204
Las diferencias de permisos críticos de las categorías: Card y Music, no serían significativas: p = 0.11939821060481424
Las diferencias de permisos críticos de las categorías: Card y Adventure, no serían significativas: p = 0.08502829507480995
Las diferencias de permisos críticos de las categorías: Events y Art & Design, no serían significativas: p = 0.7086023743224164
Las diferencias de permisos críticos de las categorías: Simulation y Trivia, no serían significativas: p = 0.33816272181240803
Las diferencias de permisos críticos de las categorías: Simulation y Music, no serían significativas: p = 0.8189362053251502
Las diferencias de permisos críticos de las categorías: Simulation y Racing, no serían significativas: p = 0.2262268418575773
Las diferencias de permisos críticos de las categorías: Action y Trivia, no serían significativas: p = 0.517174707480369
Las diferencias de permisos críticos de las categorías: Action y Music, no serían significativas: p = 0.09003125518417605
Las diferencias de permisos críticos de las categorías: Puzzle y Libraries & Demo, no serían significativas: p = 0.5354336322933045
Las diferencias de permisos críticos de las categorías: Puzzle y Board, no serían significativas: p = 0.06966637444809724
Las diferencias de permisos críticos de las categorías: Puzzle y Educational, no serían significativas: p = 0.134739819113001
Las diferencias de permisos críticos de las categorías: Trivia y Music, no serían significativas: p = 0.39837152624413974
Las diferencias de permisos críticos de las categorías: Trivia y Racing, no serían significativas: p = 0.08377952236624932
Las diferencias de permisos críticos de las categorías: Music y Racing, no serían significativas: p = 0.5581467339418402
Las diferencias de permisos críticos de las categorías: Libraries & Demo y Board, no serían significativas: p = 0.526619777459221
Las diferencias de permisos críticos de las categorías: Libraries & Demo y Educational, no serían significativas: p = 0.16116323346620098
```

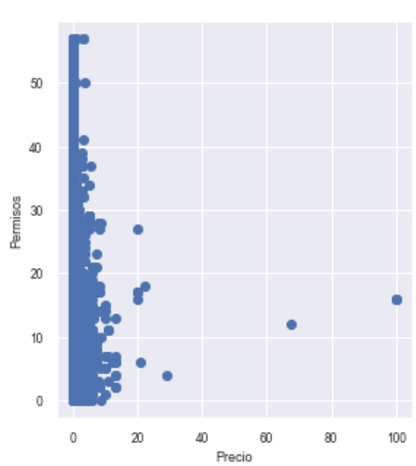
4.2. Estimar el precio que dar a una nueva aplicación para utilizar si incluimos o no técnicas invasivas (rastreadores, permisos críticos)

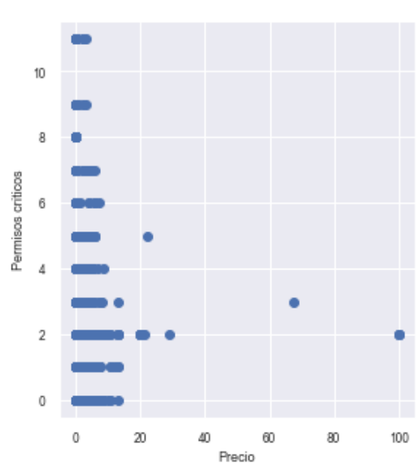
Nos gustaría conocer si la inclusión de rastreadores o permisos críticos, que permitan realizar capturas de datos del dispositivo, influye en la monetización de las aplicaciones.
Para esto realizaremos un modelo de regresión lineal múltiple que permita predecir el precio de una aplicación en relación al número de rastreadores, permisos, permisos críticos y número de posibles usuarios de la aplicación para conocer el precio que podrían llegar a pagar en base a los datos de las aplicaciones ya conocidos del Data Frame *Exodus*

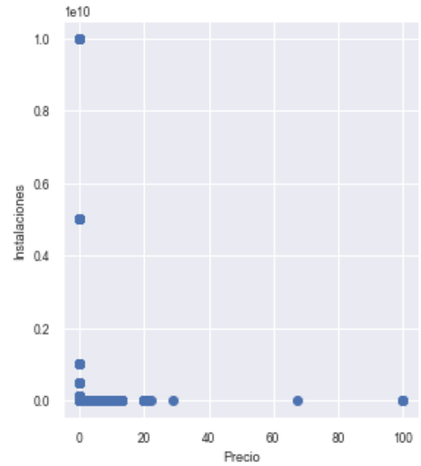
In [66]:
```
#Visualizar la relación entre el precio (variable dependiente) y el resto de variables dependientes elegidas
fig, axs = plt.subplots(nrows = 1, ncols = 4, figsize=(20,5))
axs[0].scatter(df_exodus['Price'], df_exodus['Tracker_count'])
axs[0].set_xlabel('Precio')
axs[0].set_ylabel('Rastreadores')
axs[1].scatter(df_exodus['Price'], df_exodus['Permissions_count'])
axs[1].set_xlabel('Precio')
axs[1].set_ylabel('Permisos')
axs[2].scatter(df_exodus['Price'], df_exodus['Permissions_warning_count'])
axs[2].set_xlabel('Precio')
axs[2].set_ylabel('Permisos críticos')
axs[3].scatter(df_exodus['Price'], df_exodus['Minimum Installs'])
axs[3].set_xlabel('Precio')
axs[3].set_ylabel('Instalaciones')
plt.show()

#Creación del dataset para el modelo
X = df_exodus[['Tracker_count', 'Permissions_count', 'Permissions_warning_count', 'Minimum Installs']]
y = df_exodus['Price']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=444)

#Creamos el modelo
from sklearn.linear_model import LinearRegression
lm_multi = LinearRegression()

#Entrenamos el modelo
lm_multi.fit(X_train, y_train)

#Cálculo de coeficientes
coeff_df = pd.DataFrame(np.append(lm_multi.intercept_, lm_multi.coef_),
```

```
                      np.append('B0', X.columns),
                      columns = ['Coeficientes'])

#Calidad del modelo
from sklearn.metrics import r2_score

r2 = r2_score(y_test, lm_multi.predict(X_test))
print('El valor de R-cuadrado para nuestro modelo es:\n%.4f ' % r2)

coeff_df
```










```
El valor de R-cuadrado para nuestro modelo es:
0.0009
```

Out[66]:

	Coeficientes
B0	0.0824
Tracker_count	-0.0032
Permissions_count	-0.0014
Permissions_warning_count	-0.0006
Minimum Installs	-0.0000

A tenor de los resultados, podemos ver que la relación de la variable independiente *Price* con las variables dependientes *Trackers_count*, *Permissions*, *Permissions_warning_count*, *Minimum Installs* no es lineal. Esto es debido a que la gran mayoría de aplicaciones existentes actualmente son de carácter gratuito.

- Para el número de rastreadores, sí se observa que las aplicaciones gratuitas tienden a incorporar un número mayor de rastreadores que las que son de pago. De hecho las aplicaciones que resaltan por tener un precio mayor están en la parte baja en cuanto a número de rastreadores se refiere pues se monetizan con la propia compra de la aplicación.
- En cuanto al número de permisos está más distribuido, no obstante, las aplicaciones que demandan más permisos siguen sin ser en la gran mayoría de pago.
- Sobre los permisos críticos, se observa una tendencia similar, donde se ve que las aplicaciones que más cuestan rondan los 2 y 3 permisos críticos. También se observa la tendencia de pirámide del atributo anterior, donde las aplicaciones que demandan más permisos críticos suelen ser gratuitas.
- El atributo con menor influencia sería el número de instalaciones. Es curioso ver cómo las aplicaciones de pago tienen pocos o muy pocos usuarios, mientras que las gratuitas convencen más al público.

Ya se intuye que los resultados van a ser poco prometedores y tenderán a aplicaciones gratuitas invariablemente de incluir objetos invasivos en la aplicación, no obstante se genera el modelo, se entrena y se pueden observar los coeficientes de la recta de regresión. Vemos que todas las variables tienen una importancia cercana a 0 y además negativa. Por ejemplo, para el número de rastreadores, el coeficiente más influyente, si todos los demás atributos fueran cero, por cada rastreador que añadiéramos, tendríamos que bajar el precio un promedio del 0,0032.

Como se puede observar, el propio test R2, indica que el modelo apenas explica la variabilidad de los datos en torno a la media.

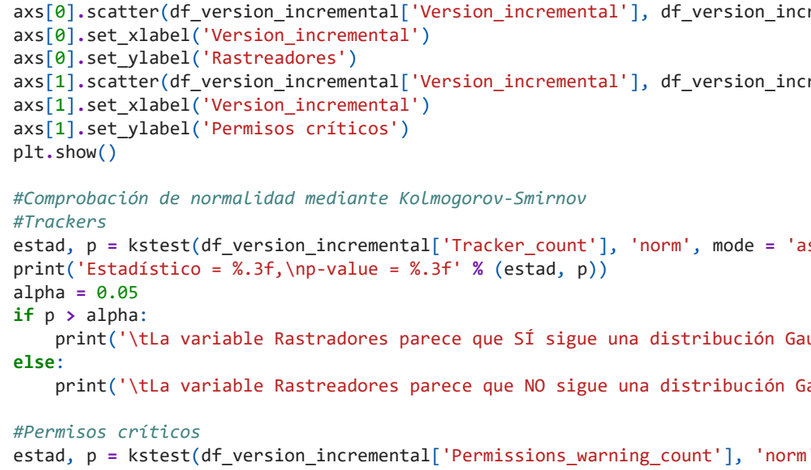
En conclusión, aunque parece que las aplicaciones peor intencionadas se ofrecen de manera gratuita añadiendo un mayor número de rastreadores y permisos innecesarios, no deberíamos plantear la monetización inicial, o el precio de nuestra aplicación basándonos en la incorporación de estos elementos. El beneficio no sería significante de cara a la aceptación del público, el cual comulga con aplicaciones gratuitas sin importar realmente salvaguardar la privacidad. Lo que se traduce, por el momento, en que llegaremos a más público si la aplicación es gratuita y no se va a notar un descenso de ventas con la incorporación de rastreadores o añadiendo segundas funcionalidades con permisos innecesarios.

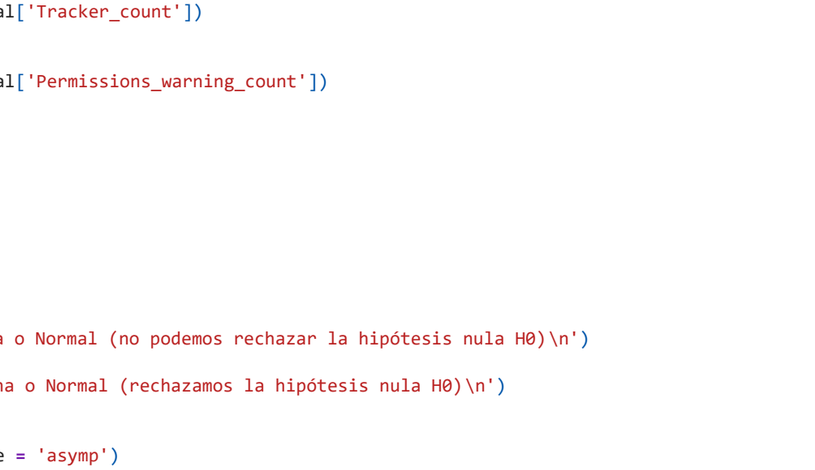
4.3. Observar la correlación entre la invasión de la privacidad y otras características.

4.3.1. ¿Integran las aplicaciones nuevos rastreadores a medida que actualizan nuevas versiones?

Se pretende observar la medida en la que se actualizan las aplicaciones, sus desarrolladores optan por incluir mayor número de rastreadores o permisos críticos observando la correlación de las variables *Tracker_count*, *Permissions_warning_count* y la transformación numérica incremental de *Version*.

In [67]:
```
#Adaptar el data set para tener una versión auxiliar incremental numérica
df_version_incremental = df_exodus[['App Id', 'Last Updated','Tracker_count', 'Permissions_warning_count']].sort_values(by = ['App Id', 'Last Updated'])
df_version_incremental['Version_incremental'] = df_version_incremental.groupby(['App Id']).cumcount() + 1
df_version_incremental

#Visualización de la correlación
fig, axs = plt.subplots(nrows = 1, ncols = 2, figsize=(20,5))
axs[0].scatter(df_version_incremental['Version_incremental'], df_version_incremental['Tracker_count'])
axs[0].set_xlabel('Version_incremental')
axs[0].set_ylabel('Rastreadores')
axs[1].scatter(df_version_incremental['Version_incremental'], df_version_incremental['Permissions_warning_count'])
axs[1].set_xlabel('Version_incremental')
axs[1].set_ylabel('Permisos críticos')
plt.show()

#Comprobación de normalidad mediante Kolmogorov-Smirnov
#Trackers
estad, p = kstest(df_version_incremental['Tracker_count'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Rastradores parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Rastreadores parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Permisos críticos
estad, p = kstest(df_version_incremental['Permissions_warning_count'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Permisos Críticos parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Permisos Críticos parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Versionados
estad, p = kstest(df_version_incremental['Version_incremental'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Version incremental parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Version incremental parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Las variables no siguen distribución normal y habrá que aplicar la correlación de Spearman no paramétrico
df_version_incremental[['Tracker_count','Permissions_warning_count','Version_incremental']].corr(method = 'spearman')
```






```
Estadístico = 0.681,
p-value = 0.000
        La variable Rastreadores parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)

Estadístico = 0.623,
p-value = 0.000
        La variable Permisos Críticos parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)

Estadístico = 0.841,
p-value = 0.000
        La variable Version incremental parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)
```

Out[67]:

	Tracker_count	Permissions_warning_count	Version_incremental
Tracker_count	1.0000	0.2337	0.1284
Permissions_warning_count	0.2337	1.0000	0.2245
Version_incremental	0.1284	0.2245	1.0000

La variable *Version*, como hemos estudiado en puntos anteriores, es una etiqueta no estandarizada con la que los desarrolladores identifican las distintas actualizaciones incrementales de la aplicación. En los casos en los que no teníamos dato, lo imputamos utilizando una cuenta incremental con números enteros. De esta manera la variable será ahora numérica y podremos utilizar tests de correlación.

Comprobamos la relación entre las variables dependientes del estudio, rastreadores y permisos críticos, frente al incremento en el número de revisiones de las aplicaciones:
Visualmente se aprecia que no existe una clara correlación.

Observamos la distribución de las variables en estudio para conocer si realizaremos el test de correlación de Pearson o Spearman:
Las variables no siguen una distribución normal y se utiliza Spearman.

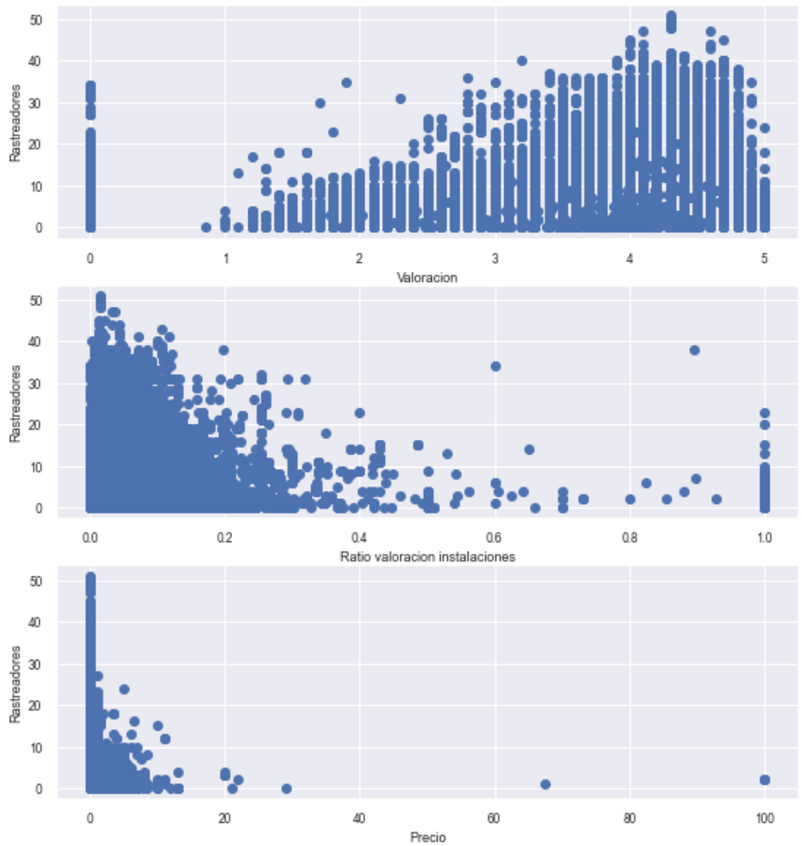
En la tabla de resultados, podemos observar que existe una pequeña correlación positiva entre las variables, es decir, a medida que una aplicación se actualiza se suelen añadir más rastreadores o nuevos permisos críticos. Por el valor, es más acentuado el incremento en rastreadores que en permisos. También destacar que al ser una correlación muy cercana a cero, implica que de manera general, las aplicaciones ya suelen nacer con el número de rastreadores y permisos críticos que se hayan decidido en los requisitos del proyecto, desde la primera versión.

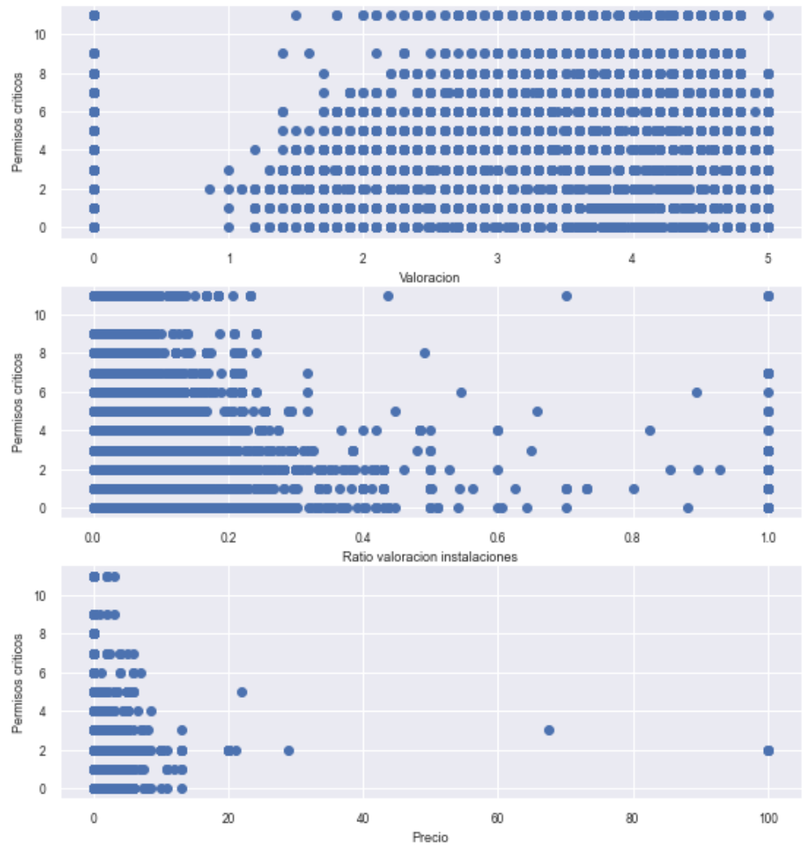
4.3.2. ¿Tiene la valoración de los usuarios, el ratio de instalaciones y valoradores o el precio relación con la inclusión de rastreadores o permisos críticos?

Se pretende observar la medida en la que, sobre el precio, la valoración de la aplicación y el ratio de valoradores que se han instalado la aplicación, afecta a la presencia de rastreadores o permisos críticos observando la correlación de las variables *Tracker_count*, *Permissions_warning_count* y las variables *Rating*, *Ratio_valoracion* y *Price*

In [68]:
```
df_rating_price = df_exodus[['Tracker_count','Permissions_warning_count','Rating','Price','Ratio_valoracion']]

#Visualización de la correlación
fig, axs = plt.subplots(nrows = 3, ncols = 2, figsize=(20,10))
axs[0][0].scatter(df_rating_price['Rating'], df_rating_price['Tracker_count'])
axs[0][0].set_xlabel('Valoracion')
axs[0][0].set_ylabel('Rastreadores')
axs[0][1].scatter(df_rating_price['Rating'], df_rating_price['Permissions_warning_count'])
axs[0][1].set_xlabel('Valoracion')
axs[0][1].set_ylabel('Permisos críticos')

axs[1][0].scatter(df_rating_price['Ratio_valoracion'], df_rating_price['Tracker_count'])
axs[1][0].set_xlabel('Ratio valoracion instalaciones')
axs[1][0].set_ylabel('Rastreadores')
axs[1][1].scatter(df_rating_price['Ratio_valoracion'], df_rating_price['Permissions_warning_count'])
axs[1][1].set_xlabel('Ratio valoracion instalaciones')
axs[1][1].set_ylabel('Permisos críticos')

axs[2][0].scatter(df_rating_price['Price'], df_rating_price['Tracker_count'])
axs[2][0].set_xlabel('Precio')
axs[2][0].set_ylabel('Rastreadores')
axs[2][1].scatter(df_rating_price['Price'], df_rating_price['Permissions_warning_count'])
axs[2][1].set_xlabel('Precio')
axs[2][1].set_ylabel('Permisos críticos')
plt.show()

#Comprobación de normalidad mediante Kolmogorov-Smirnov
#Trackers
estad, p = kstest(df_rating_price['Rating'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Rating parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Rating parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Permisos críticos
estad, p = kstest(df_rating_price['Ratio_valoracion'], 'norm', mode = 'asymp')
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Ratio_valoracion parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Ratio_valoracion parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Versionados
estad, p = kstest(df_rating_price['Price'], 'norm', mode = 'asymp')
```

```
print('Estadístico = %.3f,\np-value = %.3f' % (estad, p))
alpha = 0.05
if p > alpha:
    print('\tLa variable Price parece que SÍ sigue una distribución Gaussiana o Normal (no podemos rechazar la hipótesis nula H0)\n')
else:
    print('\tLa variable Price parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)\n')

#Las variables no siguen distribución normal y habrá que aplicar la correlación de Spearman no paramétrico
df_rating_price.corr(method = 'spearman')
```






```
Estadístico = 0.963,
p-value = 0.000
        La variable Rating parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)

Estadístico = 0.500,
p-value = 0.000
        La variable Ratio_valoracion parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)

Estadístico = 0.500,
p-value = 0.000
        La variable Price parece que NO sigue una distribución Gaussiana o Normal (rechazamos la hipótesis nula H0)
```

Out[68]:

	Tracker_count	Permissions_warning_count	Rating	Price	Ratio_valoracion
Tracker_count	1.0000	0.2337	0.1443	-0.0929	0.1769
Permissions_warning_count	0.2337	1.0000	-0.0025	-0.0717	0.0947
Rating	0.1443	-0.0025	1.0000	-0.0163	0.4233
Price	-0.0929	-0.0717	-0.0163	1.0000	0.0595
Ratio_valoracion	0.1769	0.0947	0.4233	0.0595	1.0000

Comprobamos la relación entre las variables dependientes del estudio, rastreadores y permisos críticos, frente a las variables en estudio:
Visualmente se aprecia que no existe una clara correlación.

Observamos la distribución de las variables en estudio para conocer si realizaremos el test de correlación de Pearson o Spearman:
Las variables no siguen una distribución normal y se utiliza Spearman.

En la tabla de resultados, podemos observar que al igual que con las variables anteriores, el número de rastreadores y registros presenta una correlación muy pequeña con respecto a la valoración, el precio y el ratio de valoradores. Son las siguientes:

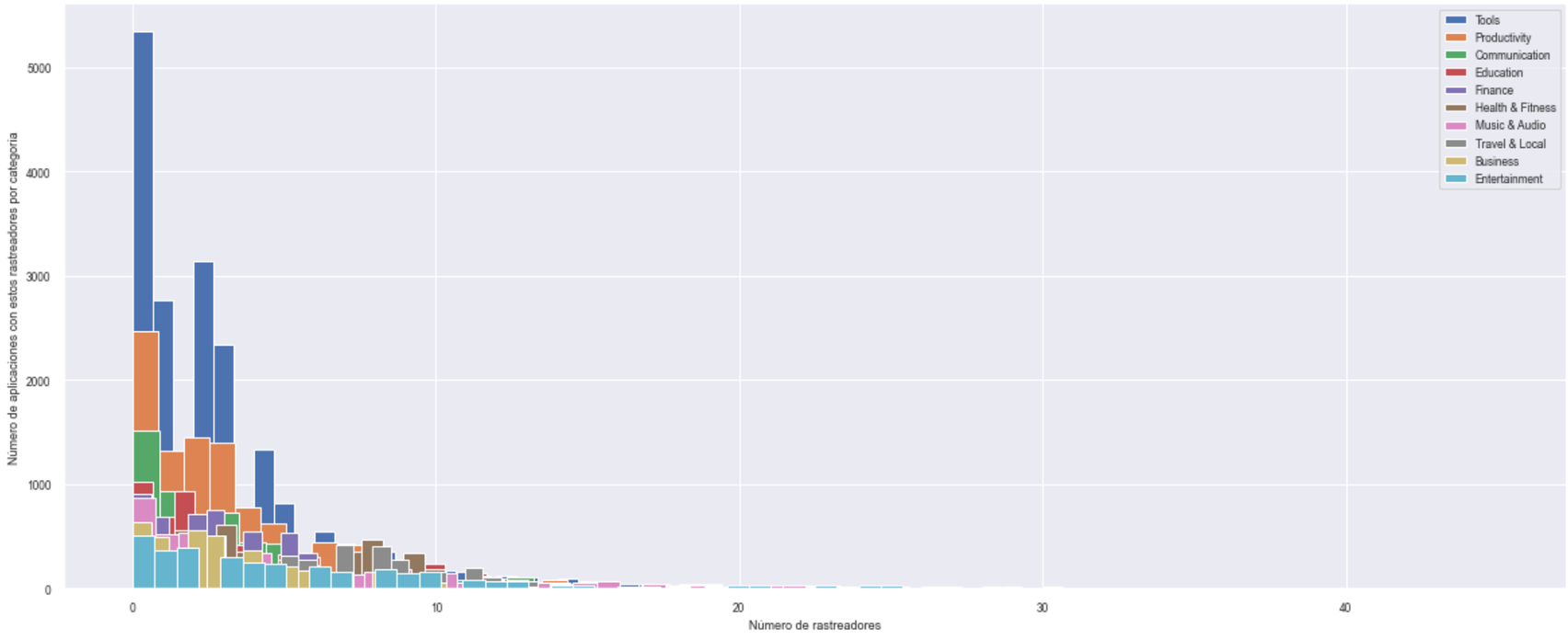
- Sobre las valoraciones, la correlación es positiva con respecto a rastreadores y casi nula, aunque negativa, en el caso de permisos importantes. Quiere decir que las aplicaciones con mayor número de rastreadores, están recibiendo valoraciones positivas. Al contrario, aunque sin apenas repercusión, se ve que el incremento de permisos críticos haga un poco de "daño" en la valoración.
- Sobre el precio de la aplicación, también es lo esperado, cuantos más rastreadores y permisos, las aplicaciones bajan su precio.
- Sobre el ratio de valoradores, el porcentaje de gente que valora las aplicaciones, crece en aquellas con rastreadores, seguramente por su popularidad, pero también cuando piden permisos que pueden resultar innecesarios. Este dato parece ser bueno para publicitar las malas intenciones de las aplicaciones redactando más comentarios (aunque por el momento no parecen ser muy críticos por lo visto en la correlación con *Rating*

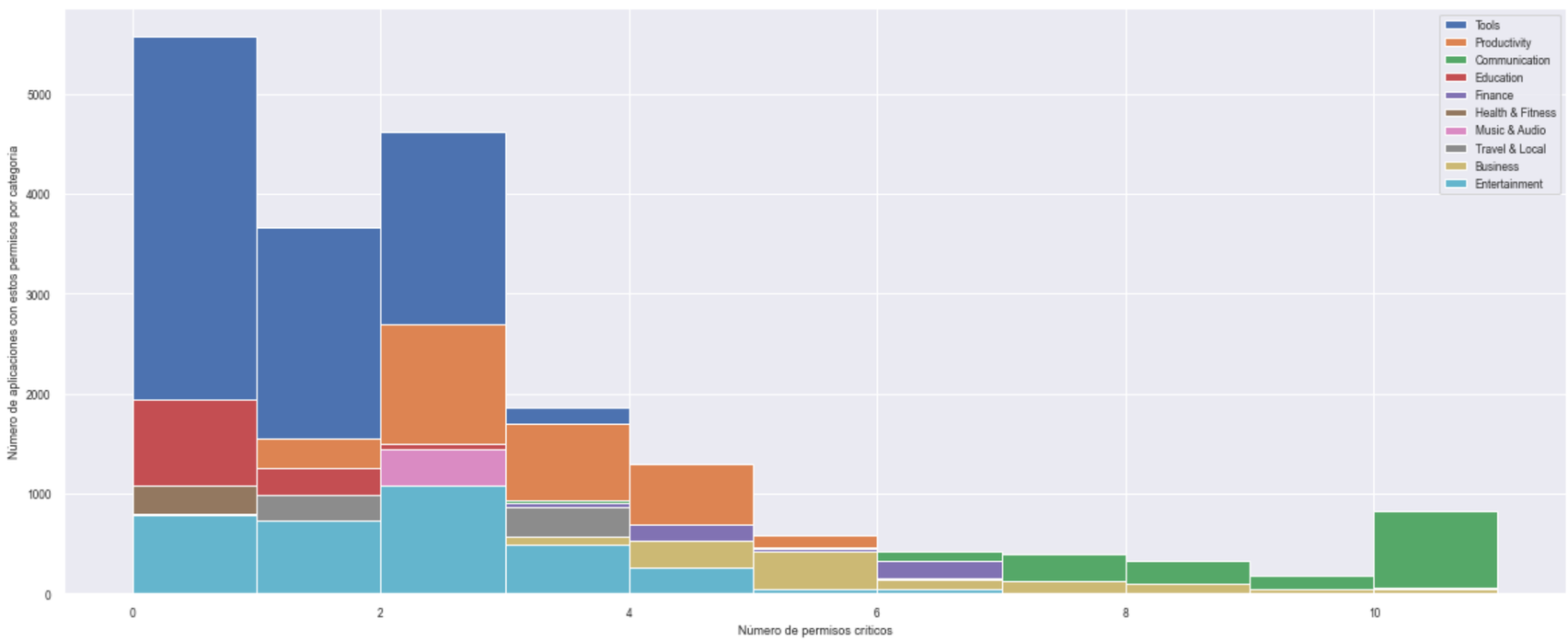
4.3.3. Correlaciones con variables categóricas

Se quiere observar la relación existente entre el incremento de rastreadores y permisos críticos por categorías de aplicaciones, selección de editores y financiación.

Categorías de aplicaciones

In [69]:
```
plt.figure(figsize = (20,8))
for cat in df_exodus['Category'].value_counts().head(10).index:
    df_exodus[df_exodus['Category'] == cat]['Tracker_count'].hist(bins = 51, label = cat)
plt.xlabel('Número de rastreadores')
plt.ylabel('Número de aplicaciones con estos rastreadores por categoría')
plt.legend()
plt.show()

plt.figure(figsize = (20,8))
for cat in df_exodus['Category'].value_counts().head(10).index:
    df_exodus[df_exodus['Category'] == cat]['Permissions_warning_count'].hist(bins = 11, label = cat)
plt.xlabel('Número de permisos críticos')
plt.ylabel('Número de aplicaciones con estos permisos por categoría')
plt.legend()
plt.show()
```



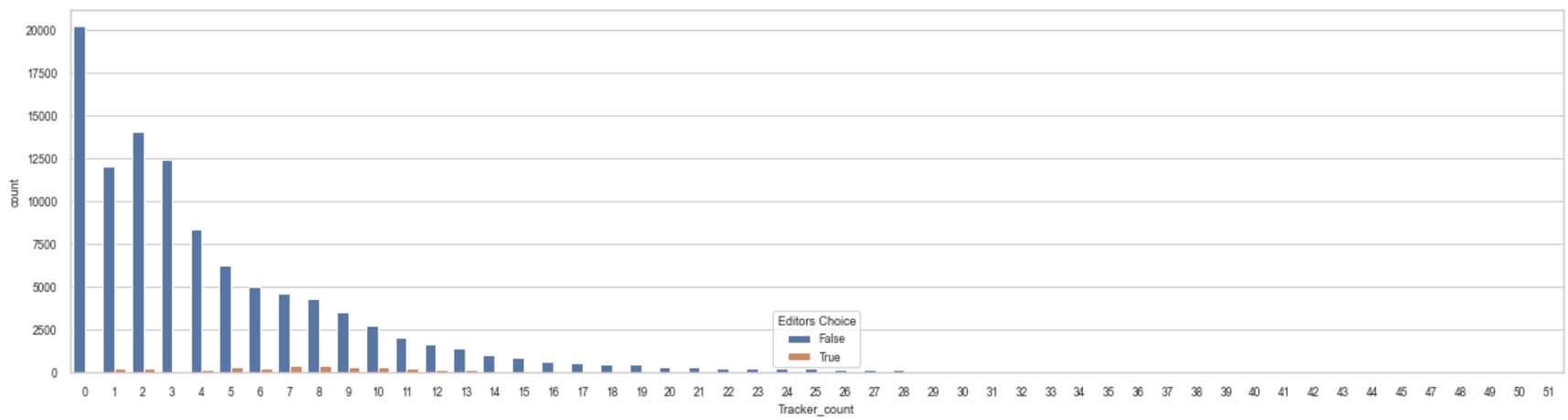


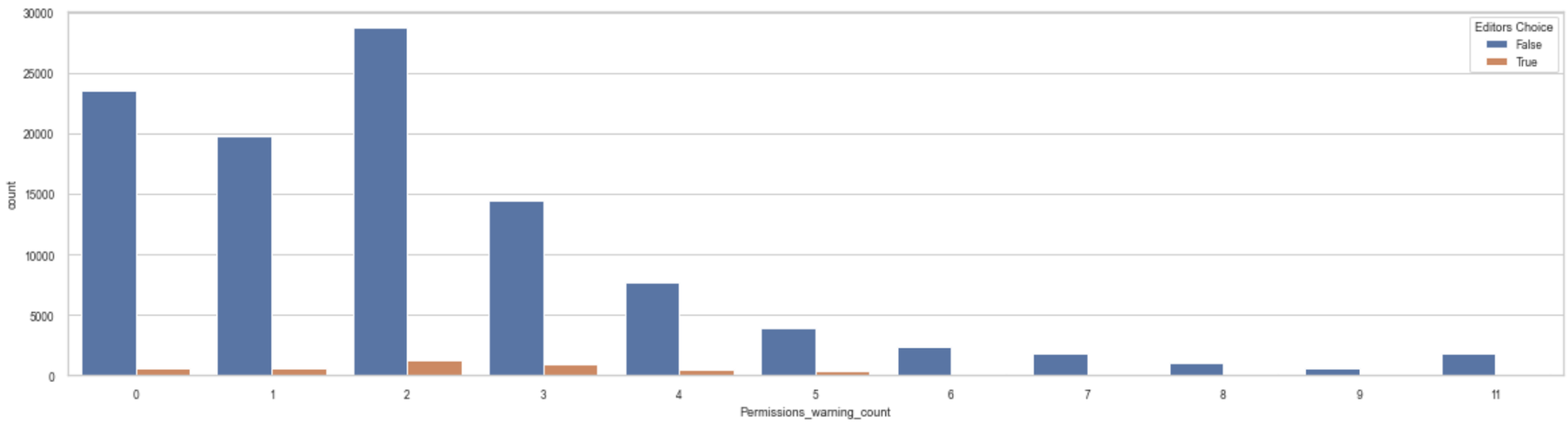
Es interesante observar cómo el número de rastreadores utilizados por categoría es tan variable. Si bien la gran mayoría utilizan un número comedido de rastreadores, aún siendo pocas las aplicaciones en estas categorías hay géneros como el Entretenimiento, Viajes, Salud y la Música que son los que más rastreadores van acumulando. Otros géneros más serios como herramientas o productividad, engloban muchas aplicaciones pero tratan de utilizar el mínimo de rastreadores posible.

En cuanto a los permisos, es muy interesante observar cómo las aplicaciones de mensajería , entretenimiento o negocios, abusan de la utilización de permisos críticos. El género Tools, que engloba aplicaciones de configuración del terminal, ni siquiera aparece en las aplicaciones que más permisos críticos dicen consumir.

Selección de editores

In [70]:
```
plt.figure(figsize = (20,5))
sns.set_style('whitegrid')
sns.countplot(x = 'Tracker_count', hue = 'Editors Choice', data = df_exodus, palette = 'deep')
plt.show()

plt.figure(figsize = (20,5))
sns.set_style('whitegrid')
sns.countplot(x = 'Permissions_warning_count', hue = 'Editors Choice', data = df_exodus, palette = 'deep')
plt.show()
```

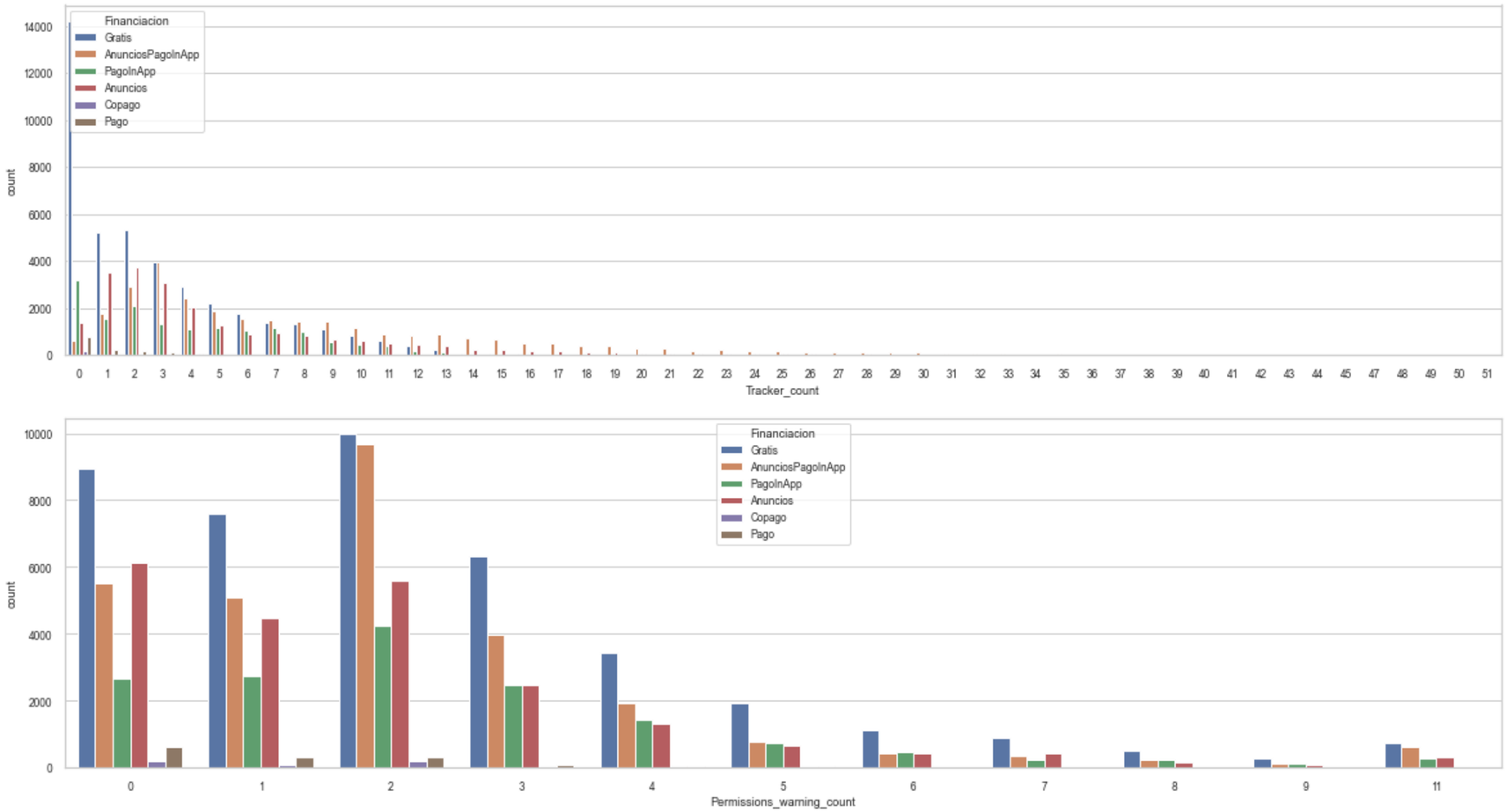




Con esta comparativa, vemos que es útil utilizar aplicaciones que estén certificadas por editores de la *Google Play Store* puesto que, con diferencia, no recomiendan aplicaciones con muchos rastreadores incluidos o, sobre todo, que soliciten demasiados permisos.

Financiación

In [71]:
```
plt.figure(figsize = (20,5))
sns.set_style('whitegrid')
sns.countplot(x = 'Tracker_count', hue = 'Financiacion', data = df_exodus, palette = 'deep')
plt.show()

plt.figure(figsize = (20,5))
sns.set_style('whitegrid')
sns.countplot(x = 'Permissions_warning_count', hue = 'Financiacion', data = df_exodus, palette = 'deep')
plt.show()
```



Sobre la financiación de las aplicaciones y el número de rastreadores, es interesante observar como las aplicaciones gratuitas destacan por no incorporar rastreadores y el número va decreciendo a medida que aumentan los rastreadores incluidos en favor de las

aplicaciones con Anuncios y micropagos o solamente anuncios.
En general, las aplicaciones que contienen Anuncios o micropagos con anuncios, son muy propensas a incorporar el mayor número de rastreadores posible.

En cuanto a los permisos críticos, sí que vemos que cuando se solicitan un número elevado de permisos críticos, las aplicaciones detectadas son en su mayoría las gratuitas. En un claro reclamo por poder tomar control del dispositivo en funciones ajenas a la de la propia aplicación.

5. Representación de los resultados a partir de tablas y gráficas

Durante el desarrollo del documento se han ido utilizando tablas de datos y gráficas como histogramas, diagramas qq, box-plot, gráficos de barras, de distribución, diagrama de puntos y de correlación en las situaciones necesarias para ilustrar los resultados o situaciones que se han ido planteando.

6. Resolución del problema

6.1. Métodos no supervisados

Puesto que no existe un atributo que defina la peligrosidad o riesgo de instalación de una aplicación, se pretende realizar un modelo de clustering que nos permita establecer este criterio de clasificación de aplicaciones en base a características relacionadas con la privacidad del usuario de aplicaciones móviles para poder utilizar el data set posteriormente para generar modelos supervisados.

Elección del conjunto de datos
Realizamos una selección vertical de atributos que incluyan aquellos que registran rastreadores y permisos.
También una selección horizontal de los registros para utilizar la información de la última actualización de cada aplicación.

De la misma manera, se prepara el data set para normalizar los atributos numéricos y evitar valores extremos problemáticos en algoritmos de cálculo de distancias.

In [72]:
```
#Selección vertical de atributos a analizar y necesarios para generar el dataset.
atributos = ['App Id','Tracker_count','Permissions_count','Permissions_warning_count','GOOGLE FIREBASE ANALYTICS','GOOGLE ADMOB','GOOGLE CRASHLYTICS','GOOGLE ANALYTICS',
             'FACEBOOK LOGIN','FACEBOOK ANALYTICS','FACEBOOK ADS','FACEBOOK PLACES','FLURRY','INMOBI','MOAT','APPSFLYER','TWITTER MOPUB','UNITY3D ADS','APPLOVIN (MAX AND SPARKLABS)',
             'ADJUST','INTEGRAL AD SCIENCE','ADCOLONY','AMAZON ADVERTISEMENT','IRONSOURCE','CHARTBOOST','BRANCH','ONESIGNAL','MILLENNIAL MEDIA','HOCKEYAPP','TAPJOY','MIXPANEL','MYTARGET',
             'INTERNET','ACCESS_NETWORK_STATE','WAKE_LOCK','WRITE_EXTERNAL_STORAGE','RECEIVE','READ_EXTERNAL_STORAGE','ACCESS_WIFI_STATE','VIBRATE','RECEIVE_BOOT_COMPLETED',
             'BIND_GET_INSTALL_REFERRER_SERVICE','ACCESS_FINE_LOCATION','BILLING','C2D_MESSAGE','ACCESS_COARSE_LOCATION','CAMERA','READ_PHONE_STATE','GET_ACCOUNTS','FOREGROUND_SERVICE',
             'WRITE_SETTINGS','BLUETOOTH','RECORD_AUDIO','READ_CONTACTS','READ_GSERVICES','SYSTEM_ALERT_WINDOW','CHANGE_WIFI_STATE','READ_SETTINGS','READ','MODIFY_AUDIO_SETTINGS','WRITE',
             'USE_CREDENTIALS','BLUETOOTH_ADMIN','USE_FINGERPRINT','GET_TASKS', 'Last Updated']

#Selección horizontal agrupando por el identificador de la aplicación y tomando la última versión de la misma
X = df_exodus[atributos].sort_values(by = ['App Id', 'Last Updated'], ascending = False).reset_index(drop = True).groupby('App Id').tail(1).reset_index(drop = True)

#Eliminar columnas auxiliares y binarizar las variables categóricas
X.drop(columns = ['Last Updated'], inplace = True)

#Normalizar valores numéricos
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_norm = pd.DataFrame(scaler.fit_transform(X.drop(columns = ['App Id'])), columns = X.columns.delete(0))

X_norm.head(3).append(X_norm.tail(3))
```

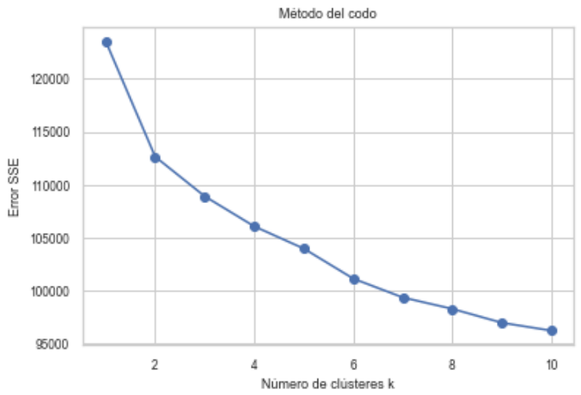
Out[72]:

	Tracker_count	Permissions_count	Permissions_warning_count	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLO
0	0.0213	0.1053	0.1818	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0(
1	0.0213	0.2456	0.2727	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0(
2	0.2128	0.1404	0.0909	1.0000	1.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000	1.0000	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0(
47118	0.0213	0.1053	0.0909	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0(
47119	0.0000	0.0702	0.0909	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0(
47120	0.2128	0.7544	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0(

Creación del modelo
Estimando el número de particiones, parece que un número interesante para agruparlas sería en 6 categorías, puesto que a partir de este límite, el error corregido al incorporar nuevas agrupaciones no es tan grande como en los saltos de partición anteriores.

In [73]:
```
from scipy.spatial.distance import cdist
from sklearn import cluster
```

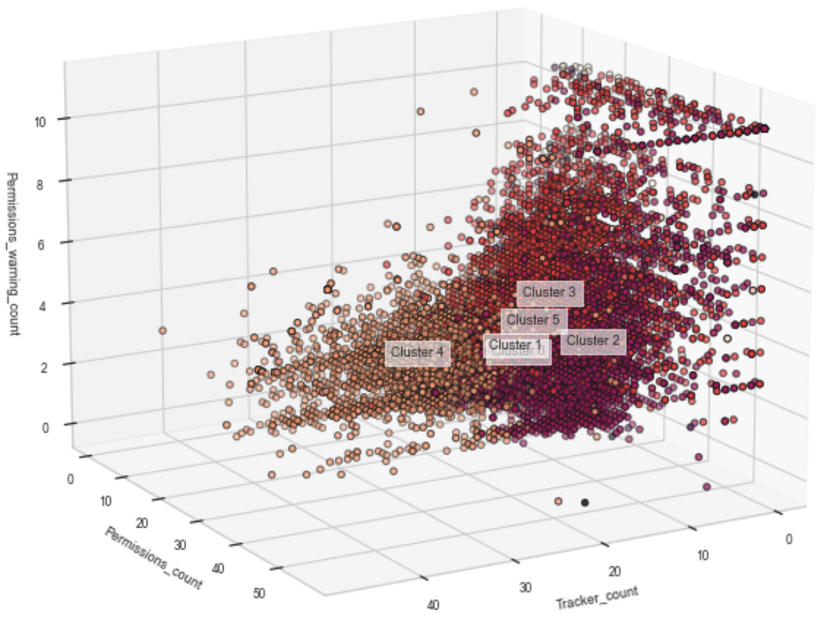
```
K = range(1, 11)
SSE = []
for k in K:
    kmeans = cluster.KMeans(n_clusters = k)
    kmeans.fit(X_norm)
    SSE.append(sum(np.min(cdist(X_norm, kmeans.cluster_centers_, 'euclidean'), axis = 1)))

plt.plot(K, SSE, 'bo-')
plt.xlabel('Número de clústeres k')
plt.ylabel('Error SSE')
plt.title('Método del codo')
plt.show()
```



Creación de un modelo de agrupamiento a partir de los datos anteriores para establecer *6* clasificaciones de aplicaciones atendiendo a la información de rastreadores y permisos que utilizan.

Tras realizar el modelo, se visualizan los datos característicos los representantes centrales de cada una de las clasificaciones establecidas y la clasificación gráfica de las aplicaciones según el tipo donde se agrupa.

In [74]:
```
#Ejecutamos un modelo k-means con el número de clústeres identificados con la técnica del codo
kmeans = cluster.KMeans(n_clusters = 6)
kmeans.fit(X_norm)

#Utilizamos esta función para obtener el número de instancias agrupadas en cada clúster
unique, counts = np.unique(kmeans.labels_, return_counts=True)

#Como aplicamos normalización para realizar el modelo, se desnormaliza previamente.
descriptores = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
                            columns = X.columns.delete(0))
clases = descriptores.join(pd.DataFrame(counts, columns = ['Instancias'])).round(0).astype(int)

#Visualización del dataFramede las aplicaciones
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=15, azim=60)
ax.scatter(X['Tracker_count'],
           X['Permissions_count'],
           X['Permissions_warning_count'],
           c = kmeans.labels_.astype(np.float), edgecolor = 'k')
ax.set_xlabel('Tracker_count')
ax.set_ylabel('Permissions_count')
ax.set_zlabel('Permissions_warning_count')
for k in range(0, 6):
    nombre = 'Cluster ' + str(k)
    ax.text3D(scaler.inverse_transform(kmeans.cluster_centers_)[k, 0],
              scaler.inverse_transform(kmeans.cluster_centers_)[k, 1],
              scaler.inverse_transform(kmeans.cluster_centers_)[k, 2] + 0.1,
              nombre, horizontalalignment='center',
              bbox=dict(alpha=.6, edgecolor='w', facecolor='w'))

plt.show()

#Visualización de datos
clases
```



Out[74]:

	Tracker_count	Permissions_count	Permissions_warning_count	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCOLONY
0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	4	9	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	6	31	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	6	20	4	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	18	13	2	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1
5	2	9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nuestro modelo ha detectado 6 clases distintas de aplicaciones que podríamos describir:

- Clase 3: Es la mejor clase, perteneciente a las aplicaciones que no incluyen rastreadores y utilizan los permisos necesarios, ninguno crítico, para el desarrollo de sus funciones. Hay *7.173 aplicaciones* en este grupo.
- Clase 4: Es la peor clase de todas, se trata de aplicaciones que incorporan un número exagerado de rastreadores y un número elevado de permisos algunos críticos. Existen *3.514 aplicaciones* en este grupo.
- Clase 2 y Clase 1: Son aplicaciones similares que incorporan un número elevado de rastreadores pero sobre todo de permisos y con un número elevado de permisos críticos. La Clase 2 exige menos recursos pero más críticos y la Clase 1 al contrario menos permisos pero más críticos.
- Clase 0 y Clase 5: También son aplicaciones con características similares pero de menor riesgo. Tienen un número similar en requerimientos de permisos y críticos pero difieren del número de rastreadores incluidos, menor en la Clase 5.

Viendo los resultados, creo que sería posible agrupar las clases similares y etiquetar según el siguiente criterio:

- Riesgo Alto: Clase 4
- Riesgo Moderado: Clases 1 y 2
- Segura: Clases 0 y 5
- Respetuosa: Clase 3

Incorporar la información

Creamos la nueva columna en el Data Frame *Exodus*

In [75]:
```
X = X.join(pd.DataFrame(kmeans.labels_, columns = ['Riesgo']))
```

```
cat_riesgo = ['Segura', 'Moderado', 'Moderado', 'Respetuosa', 'Alto', 'Segura'] #Agregación de las clases en las 4 clases finales

def asignarRiesgo(clase):
    '''
    A partir de la etiqueta definida para cada una de las agrupaciones de clases estudiadas, devuelve su nombre para imputarlo
    '''
    return cat_riesgo[clase]

#Traducción del índice de clase por su etiqueta
X['Riesgo'] = np.vectorize(asignarRiesgo)(X['Riesgo'])

#Integración en el Data Frame final
df_exodus = pd.merge(df_exodus, pd.DataFrame(X[['App Id', 'Riesgo']]), how = 'inner', on = 'App Id')
```

In [76]:
```
df_exodus.head(3).append(df_exodus.tail(3))
```

Out[76]:

	Id	Name	Tracker_count	Permissions_count	Version	GOOGLE FIREBASE ANALYTICS	GOOGLE ADMOB	GOOGLE CRASHLYTICS	GOOGLE ANALYTICS	FACEBOOK LOGIN	FACEBOOK ANALYTICS	FACEBOOK ADS	FACEBOOK PLACES	FLURRY	INMOBI	MOAT	APPSFLYER	TWITTER MOPUB	UNITY3D ADS	APPLOVIN (MAX AND SPARKLABS)	ADJUST	INTEGRAL AD SCIENCE	ADCO
0	1	Tan	3	9	5.7.0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	30486	Tan	2	9	5.7.1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	50419	Tan	2	9	5.8.2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
110363	152871	Toy Of War	0	1	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110364	152903	rain	2	18	1.0.3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110365	153101	LibreSpeed	0	1	1.2.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

6.2. Métodos supervisados

Al haber incorporado un nuevo atributo de calificación de la seguridad, con el Data Frame *Exodus* podemos establecer un modelo a partir de un método supervisado de clasificación para poder categorizar nuevas aplicaciones en relación al riesgo de privacidad del usuario.

Elección del conjunto de datos

Utilizamos la selección horizontal y vertical de atributos realizada en el apartado anterior que incluyen los atributos más relacionados con la privacidad de las aplicaciones incorporando la información del *Riesgo* como atributo de clase para poder elaborar un clasificador.

In [77]:
```
#Selección vertical de atributos a analizar y necesarios para generar el dataset.
atributos = ['App Id','Tracker_count','Permissions_count','Permissions_warning_count','GOOGLE FIREBASE ANALYTICS','GOOGLE ADMOB','GOOGLE CRASHLYTICS','GOOGLE ANALYTICS',
             'FACEBOOK LOGIN','FACEBOOK ANALYTICS','FACEBOOK ADS','FACEBOOK PLACES','FLURRY','INMOBI','MOAT','APPSFLYER','TWITTER MOPUB','UNITY3D ADS','APPLOVIN (MAX AND SPARKLABS)',
             'ADJUST','INTEGRAL AD SCIENCE','ADCOLONY','AMAZON ADVERTISEMENT','IRONSOURCE','CHARTBOOST','BRANCH','ONESIGNAL','MILLENNIAL MEDIA','HOCKEYAPP','TAPJOY','MIXPANEL','MYTARGET',
             'INTERNET','ACCESS_NETWORK_STATE','WAKE_LOCK','WRITE_EXTERNAL_STORAGE','RECEIVE','READ_EXTERNAL_STORAGE','ACCESS_WIFI_STATE','VIBRATE','RECEIVE_BOOT_COMPLETED',
             'BIND_GET_INSTALL_REFERRER_SERVICE','ACCESS_FINE_LOCATION','BILLING','C2D_MESSAGE','ACCESS_COARSE_LOCATION','CAMERA','READ_PHONE_STATE','GET_ACCOUNTS','FOREGROUND_SERVICE',
             'WRITE_SETTINGS','BLUETOOTH','RECORD_AUDIO','READ_CONTACTS','READ_GSERVICES','SYSTEM_ALERT_WINDOW','CHANGE_WIFI_STATE','READ_SETTINGS','READ','MODIFY_AUDIO_SETTINGS','WRITE',
             'USE_CREDENTIALS','BLUETOOTH_ADMIN','USE_FINGERPRINT','GET_TASKS', 'Last Updated', 'Riesgo']

#Selección horizontal agrupando por el identificador de la aplicación y tomando la última versión de la misma
X = df_exodus[atributos].sort_values(by = ['App Id', 'Last Updated'], ascending = False).reset_index(drop = True).groupby('App Id').tail(1).reset_index(drop = True)

#Eliminar columnas auxiliares y binarizar las variables categóricas
X.drop(columns = ['Last Updated'], inplace = True)

#Normalizar valores numéricos
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_norm = pd.DataFrame(scaler.fit_transform(X.drop(columns = ['App Id','Riesgo'])), columns = X.columns.delete(0).delete(64)) #Hemos de quitar los valores de texto

#Volver a incorporar la variable destino
y = X['Riesgo']

X_norm.join(pd.DataFrame(y, columns = ['Riesgo'])).head(3).append(X_norm.join(pd.DataFrame(y, columns = ['Riesgo'])).tail(3))

#Creación de los conjuntos de entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split(X_norm, y, random_state = 444, test_size=0.33)
```

Generación del modelo

Vamos a utilizar el clasificador k-vecinos utilizando validación cruzada estratificada a 10 folds y midiendo el rendimiento a partir del parámetro de vecinos elegido.

In [78]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier

folds = StratifiedKFold(n_splits = 10, shuffle = True, random_state = 444)

grid_base = GridSearchCV(estimator = KNeighborsClassifier(), param_grid = {'n_neighbors': [1,3,5,7,9]}, scoring = ['accuracy'], cv = folds, refit = 'accuracy')
knn = grid_base.fit(X_train, y_train)
```

In [79]:
```
print('Número óptimo de vecinos: ' + str(knn.best_params_))
print('Precision obtenida: ' + str(knn.best_score_))

#Resultados del entrenamiento
pd.DataFrame(knn.cv_results_)
```

```
Número óptimo de vecinos: {'n_neighbors': 9}
Precision obtenida: 0.947451695649845
```

Out[79]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_accuracy	split1_test_accuracy	split2_test_accuracy	split3_test_accuracy	split4_test_accuracy	split5_test_accuracy	split6_test_accuracy	split7_test_accuracy	split8_test_a
0	1.2556	0.0543	5.0053	0.2385	1	{'n_neighbors': 1}	0.9379	0.9417	0.9360	0.9338	0.9417	0.9389	0.9376	0.9370	
1	1.2489	0.0628	6.3695	0.2036	3	{'n_neighbors': 3}	0.9424	0.9471	0.9344	0.9471	0.9465	0.9417	0.9427	0.9433	
2	1.2143	0.0410	6.7741	0.1083	5	{'n_neighbors': 5}	0.9455	0.9481	0.9373	0.9471	0.9468	0.9455	0.9531	0.9363	
3	1.2234	0.0442	7.0869	0.0915	7	{'n_neighbors': 7}	0.9468	0.9465	0.9366	0.9493	0.9528	0.9481	0.9493	0.9414	
4	1.2158	0.0516	7.3536	0.0786	9	{'n_neighbors': 9}	0.9500	0.9446	0.9382	0.9487	0.9544	0.9471	0.9500	0.9433	

Desempeño del modelo

In [80]:
```
from sklearn import metrics
import sklearn.metrics as metricas

#Cálculo de métricas
real = y_test
pred = knn.predict(X_test)

print(metricas.classification_report(real,pred))

#Matriz de confusión
matriz_confusion = metricas.confusion_matrix(real, pred)
print('\nMatriz de confusión (Filas: Real, Columnas: Predicción)')
pd.DataFrame(matriz_confusion, columns = ['Alto','Moderado','Respetuosa','Segura'], index = ['Alto','Moderado','Respetuosa','Segura'])
```

```
              precision    recall  f1-score   support

        Alto       0.97      0.85      0.91      1171
    Moderado       0.91      0.98      0.94      5341
  Respetuosa       0.96      0.83      0.89      2575
      Segura       0.97      0.99      0.98      6463

    accuracy                           0.95     15550
   macro avg       0.95      0.91      0.93     15550
weighted avg       0.95      0.95      0.95     15550
```

```
Matriz de confusión (Filas: Real, Columnas: Predicción)
```

Out[80]:

	Alto	Moderado	Respetuosa	Segura
Alto	999	117	24	31
Moderado	17	5213	42	69
Respetuosa	9	326	2130	110
Segura	2	52	21	6388

La combinación de los dos métodos, ha conseguido explotar muy bien la información contenida en el Data Frame obtenido en la primera práctica y limpiado y analizado en esta segunda práctica.

Se ha conseguido realizar una agrupación bastante acertada de aplicaciones basadas en los elementos que pueden afectar a la privacidad del usuario y a partir de éste un clasificador con una precisión muy decente del 93% que conseguiría informar al usuario o calificar en tiendas de aplicaciones, el aspecto de riesgo a la seguridad del dispositivo y los datos generados por el usuario.

Ha sido un trabajo de limpieza muy extenso e intenso debido al origen del dato proveniente del rastreo de un sitio Web, la integración con un dataset existente y la necesidad de incorporar una nueva adquisición de datos para poder casar ambos.
La limpieza de los datos y los análisis se han extendido para asegurar la calidad de ambos datasets con bastantes atributos de muy distinta índole. Se han utilizado bastantes técnicas y soluciones para la imputación de datos tanto ausentes como extremos.

La parte de análisis ha resultado poco gratificante por el hecho de contar con muchos datos y ser un conjunto de datos tan heterogéneo que las correlaciones no eran muy significativas.
Pero como se comenta al principio, se ha conseguido conjugar un Data Frame capaz de producir una herramienta para medir el riesgo el cual era el objetivo primordial de la creación del Data Frame.

In [82]:
```
contribuciones = ['Investigación previa','Redacción de respuestas','Desarrollo código']
firmas = ['LMMC', 'LMMC', 'LMMC']
pd.DataFrame({'Contribuciones': contribuciones, 'Firma': firmas})
```

Out[82]:

	Contribuciones	Firma
0	Investigación previa	LMMC
1	Redacción de respuestas	LMMC
2	Desarrollo código	LMMC