	Web-scraping and analysing Beatport.com Introduction In the following Jupyter Notebook I will perform web scraping of the Beatport.com website to generate a dataframe with all the top 100 songs by genre and then perform an analysis of the generated table. I will start by showing the results of this study and then the code.
	What is Beatport? Beatport is a digital electronic music store headquartered in the USA. Since the company's founding in 2004, its goal has been the same: to meet the distinctive needs of its community by providing top-notch goods and services that uplift and unite the community of artists, DJs, and fans. Through this dedication, Beatport hopes to continuously inspire and push innovation, leading and defining the development of dance music culture. With over 36 million unique users, 465 thousand DJ customers, and 11 million curated tracks provided by 75 thousand label relationships, Beatport is still the acknowledged industry leader for the DJ community today. Disclaimer: The data was scraped on 27 October 2022. The code will work as long as Beatport does not
	Findings The first 10 columns of the dataframe obtained after scraping the web site look like the following: Song names Of the 3200 songs there are 2911 songs with different names. This is good and is because within electronic music many times a single song is interpreted by different artists in different genres and different types of Remix. We can see that the song that appears more times is "Do it to it" (8 times).
	Remix There are 647 different types of remixes, the ones that appear most often are the "Original Remix" and the "Extended Mix". Artist 2153 unique artists are performing in the top 100 by genre within Beatport. The artist who plays the most songs is Block & Crown with 28 songs, followed by Ondamike with 19 songs, and David Guetta with Dusky with 13 songs each.
	Labels are companies, large or small, that manufacture, distribute, and promote the recordings of affiliated musicians. Essentially, record labels work to sell the brand of the artist and the products they create. 1396 unique labels play the top 100 by genre within Beatport. The labels with the most songs are Defected, Spinning' Records, Ravesta Records, Deadbeats and Musical Freedom with 33, 31, 29, 25 and 19 songs respectively. Genre Within genres, there should be 32 genres with 100 songs each. But Beatport within its top 100, for some genres, mixes their names, for example, if I enter the website I can see that in the top 100 of DJ Tools there are Acapellas and DJ Tools. Date
	Although the data are of the songs that on October 27, 2022 occupy the top 100 by genre of Baetport, we can see that not all songs are from the year 2022, there are songs that are much older: About me Besides working as a data analyst, one of my hobbies is DJing. My favorite genres to play are Deep House, Progressive House, Melodic Techno, and Tech House. I want to share with you my soundcloud channel where you can find my mixes. If you like them, I invite you to follow me and like the mixes. Thank you very much! Visit my Soundcloud CODE
	<pre># Import the libraries to be used from bs4 import BeautifulSoup import requests import pandas as pd import matplotlib.pyplot as plt import plotly.express import os # I store the url as a variable, create the variable response and get the information from the website with the # requests library. Response should be 200. url = 'https://www.beatport.com/'</pre>
	response = requests.get(url) print(response) # Now, I store the entire contents of the response in the variable scr src = response.content # Now that I've saved the page content I'm going to use Beautiful Soup to parse and process the content. # To do this I will create # a Beautiful Soup object based on the src variable. soup = BeautifulSoup(src, 'html.parser') # I create the results variable to find all the "a" tags which is # where all the "href" I need are located. results = soup.find('div', {'class': 'genres-drop head-drop header-tooltip-menu'}).find_all('a') # I create the list links
	<pre>link = [] # with a for loop I will append to the links list all the "href" for result in results: link.append(result['href']) # To get all the links from which I am going to scrap the information I need to remove the last "/" from the o # and add all the items from the new link list.I'm going to do this # with a for loop creating a new list url_ url_list = [] for i in link: url_list.append(url.strip('/') + i + '/top-100') # I create de pandas DataFrame and exported as csv df = pd.DataFrame({'link':url_list}) df.to_csv('links_beatport.csv', index=False)</pre>
Out[3]:	<pre>df.head() link thtps://www.beatport.com/genre/140-deep-dubstep-grime/95/top-100 thttps://www.beatport.com/genre/afro-house/89/top-100 thttps://www.beatport.com/genre/amapiano/98/top-100 thttps://www.beatport.com/genre/bass-club/85/top-100 thttps://www.beatport.com/genre/bass-club/85/top-100 thttps://www.beatport.com/genre/bass-house/91/top-100 # create lists of all the data I want for my dataframe. url = df['link'] # link column of my df name = [] remix = []</pre>
	<pre>artist = [] label = [] genre = [] release_date = [] # get the information from the website with the requests library. for url in url: response = requests.get(url) # I print this so I can know if every url works print(response) # Now, I store the entire contents of the response in the variable scr src = response.content soup = BeautifulSoup(src, 'html.parser') # Beautiful Soup object based src variable results = soup.find('div', {'class':'bucket tracks top-hundred-tracks'}).find('ul').find_all('li')</pre>
	<pre># with a for loop I will append to the info scraped to every list created before. for result in results: name append(result.find('span', 'class':'buk-track-primary-title')).text.strip(""")) remix.append(result.find('span', 'class':'buk-track-remixed')).text) artist.append(result.find('p', 'class':'buk-track-artists')).find('a').text.strip()) label.append(result.find('p', 'class':'buk-track-artists')).find('a').text.strip()) genre.append(result.find('p', 'class':'buk-track-artists')).text.strip()) release_date.append(result.find('p', 'class':'buk-track-encome 'buk-track-released')).text.strip()) <pre>Response [200]> Response [200]> R</pre></pre>
	<pre><response [200]=""> # Create DataFrame beatport_top100 = pd.DataFrame(</response></pre>
In [8]:	beatport_top100.head(10)
In [10]: Out[10]:	<pre>beatport_top100['Release Date'] = pd.to_datetime(beatport_top100['Release Date'], infer_datetime_format=True) # These are all of the columns columns = beatport_top100.columns columns Index(['Name', 'Remix', 'Artist', 'Label', 'Genre', 'Release Date'], dtype='object')</pre>
In [11]:	<pre>for i in columns: print(i,'({})'.format(beatport_top100[i].nunique())) print(beatport_top100[i].value_counts()) print() print() Name (2911) Do It To It</pre>
	Kings of the Rollers Remix 1 Bladerunner Remix 1 Nick The Lot Remix 1 Burr Oak Remix 1 El-B Mix 1 Name: Remix, Length: 647, dtype: int64 Artist (2153) Block & Crown 28 Ondamike 19 David Guetta 13 Dusky 13 Ghostbusterz 11 Univac 1 No Moon 1 Kris Baha 1 Joal 1 Beat Merchants 1 Name: Artist, Length: 2153, dtype: int64 Label (1396) Defected 33
	SPINNIN' RECORDS 31 Ravesta Records 29 Deadbeats 25 Musical Freedom 19 Kinetika Music 1 Tiger Records 1 Ghetto Ghetto 1 Casa Rossa 1 Beats Galore 1 Name: Label, Length: 1396, dtype: int64 Genre (57) Minimal / Deep Tech 100 Funky House 100 Electro (Classic / Detroit / Modern) 100 Psy—Trance 100 Melodic House & Techno 100 Drum & Bass 100 Tech House 100 Techno (Peak Time / Driving) 100
	Progressive House 100 Deep House 100 Dance / Electro Pop 100 Breaks / Breakbeat / UK Bass 100 Bass House 100 Techno (Raw / Deep / Hypnotic) 100 Trap / Wave 100 Nu Disco / Disco 100 Hard Techno 100 House 100 Amapiano 100 Indie Dance 100 Jackin House 100 Electronica 100 Trance 99 Organic House / Downtempo 96 140 / Deep Dubstep / Grime 95 Dubstep Afro House 93 Bass / Club 79
	UK Garage 59 DJ Tools 58 Hard Dance / Hardcore 57 Future Rave 37 Acapellas 36 UK Garage / Bassline 34 Hardstyle 31 Mainstage 26 Big Room 15 Electro House 11 UK Funky 8 Future House 8 Bassline 7 Afro / Latin 7 Hard House 6 Grime 5 Melodic Dubstep 5 Battle Tools 5 Jersey Club 5
	Juke / Footwork 5 UK / Happy Hardcore 5 Organic House 4 Midtempo 2 Global Club 2 Speed House 1 Hard Trance 1 Loops 1 Bass/Club 1 Uplifting 1 Name: Genre, dtype: int64 Release Date (476) 2022-10-21 379 2022-10-21 379 2022-10-3 98 2010-11-09 1 2010-11-27 1
	<pre>2019-03-22 1 2018-11-26</pre>
In [14]:	df_pivot.append(pivot_table) print(df_pivot[0]) Name
In [16]: Out[16]: In [17]:	<pre>import plotly.express as px import plotly.io as pio df_pivot[0].columns[0] 'Name' # Create a folder called images to save my graphs if not os.path.exists("images"): os.mkdir("images") # Filter the dataframe so that it does not crash the browser df_pivot_1_filtered = df_pivot[0].head(25) # Construct the graph and style it. fig = px.bar(df_pivot_1_filtered, x=df_pivot[0].columns[0], y='0', template='plotly_dark', text_auto=True) fig.update_layout(title='Top 25 songs that appear more than once', xaxis = dict(</pre>
	<pre>showgrid=True,) , yaxis = dict(showgrid=True), legend = dict(orientation='v'), # barmode='group', # paper_bgcolor='#000000') fig.show(renderer="iframe") fig.write_image("images/names.png")</pre>
In [19]:	<pre># Filter the dataframe so that it does not crash the browser df_pivot_1_filtered = df_pivot[1].head(10) # Construct the graph and style it. fig = px.bar(df_pivot_1_filtered, x=df_pivot[1].columns[0], y='Q', template='plotly_dark', text_auto=True) fig.update_layout(title='Top 10 types of Remix', xaxis = dict(showgrid=True,) , yaxis = dict(showgrid=True),</pre>
	<pre>legend = dict(orientation='v'), # barmode='group', # paper_bgcolor='#000000') fig.show(renderer="iframe") fig.write_image("images/remix.png")</pre>
In [20]:	<pre># Filter the dataframe so that it does not crash the browser df_pivot_1_filtered = df_pivot[2].head(25) # Construct the graph and style it. fig = px.bar(df_pivot_1_filtered, x=df_pivot[2].columns[0], y='Q', template='plotly_dark', text_auto=True) fig.update_layout(title='Top 25 Artists and its number of songs', xaxis = dict(showgrid=True,) , yaxis = dict(showgrid=True), legend = dict(</pre>
	orientation='v'), # barmode='group', # paper_bgcolor='#000000') fig.show(renderer="iframe") fig.write_image("images/artists.png")
In [21]:	<pre># Filter the dataframe so that it does not crash the browser df_pivot_1_filtered = df_pivot[3].head(25) # Construct the graph and style it. fig = px.bar(df_pivot_1_filtered, x=df_pivot[3].columns[0], y='Q', template='plotly_dark', text_auto=True) fig.update_layout(title='Top 25 Labels and its number of songs', xaxis = dict(showgrid=True,) , yaxis = dict(showgrid=True), legend = dict(orientation='v'</pre>
	<pre> /, # barmode='group', # paper_bgcolor='#000000') fig.show(renderer="iframe") fig.write_image("images/labels.png") </pre>
<pre>In [22]: Out[22]: In [23]:</pre>	Name Remix Artist Label Genre Release Date
In [23]: In [24]: Out[24]:	beatport_top100.sort_values(by='Release Date', inplace=True) Name Remix Artist Label Genre Release Date
In [31]:	

	Requirement already satisfied: jupyter-core>-4, 1n /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (on shoowert-nontebook-as-pdf) (1.4.9.2) Requirement already satisfied: tinycss2 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (from nbco etnontebook-as-pdf) (1.2.1) Requirement already satisfied: junja2>-3.0 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (from nbcovertnontebook-as-pdf) (1.3.1) Requirement already satisfied: stratlets>-5.0 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (from nbcovertnontebook-as-pdf) (1.5.1) Requirement already satisfied: stratlets>-5.0 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (from nbcovertnontebook-as-pdf) (4.1.0) Requirement already satisfied: bleach in /Users/juanpe/opt/anaconda3/lib/python3.9/site-packages (from nbcovertnontebook-as-pdf) (4.1.0) Requirement already satisfied: pandocfilters>-1.4.1 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-package (from nbcovertnontebook-as-pdf) (4.1.0) Requirement already satisfied: pandocfilters>-1.4.1 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-package (from nbcovertnontebook-as-pdf) (4.1.1) Requirement already satisfied: importlib-metadata=-3.6 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-package (from nbcovertnontebook-as-pdf) (4.1.1) Requirement already satisfied: typing-cettosions>-3.10.0.0 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-package (from pypeteernotebook-as-pdf) (4.1.1) Requirement already satisfied: appdirs>-2.0.0, >=1.1.0 in /Users/juanpe/opt/anaconda3/lib/python3.9/site-package (from pypeteernotebook-as-pdf) (4.2.2) Requirement already satisfied: without all package (from pypeteernotebook-as-pdf) (4.2.2) Requirement already satisfied: without all package (from pypeteernotebook-as-pdf) (4.2.2) Requirement already satisfied: without all package (from pypeteernotebook-as-pdf) (4.6.0) Requirement already satisfied: sati
in [32]:	<pre>!pyppeteer-install [INFO] Starting Chromium download. 100% </pre>