Abir Ahmed, Ioannis Paranikas

SI206 Final Project
Team Paranikas and Ahmed

Original Goal

The original goal of this project was to create a database of historic sales data from StockX, and to then monitor for any price changes. As soon as a price change was detected that was greater than a % selected by users, the goal was to send an SMS to users so they could buy the product. The goal was to send the SMS using the Twilio API, and to send product links encoded with the VigLink API to monetize any purchases made by users.

Adapted Goals

After reading the initial project spec, our team realized that the original goal was not under the scope of the project spec. Instead, we designed a script that scrapes a user defined number of products across Goat.com and StockX.com (two popular sneaker resale websites). The script then scrapes product data for said popular SKU's, calculates the average sale price by size, the total sales by size, and total bids by size across the collected data. These 3 calculations are then visualized, using MatplotLib. Visualizations are additionally generated for all StockX and Goat products scraped.

Problems Faced

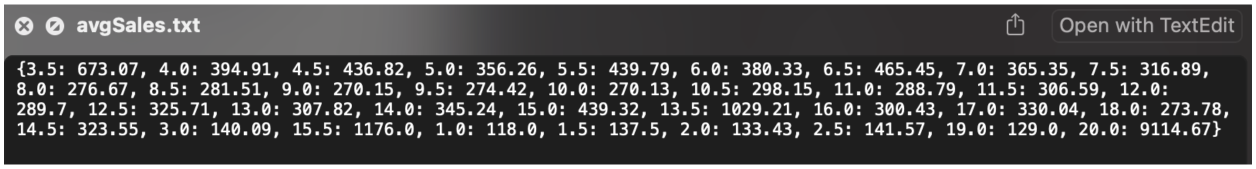
When completing the project, it became immediately clear that StockX.com and Goat.com did not offer public API's, and their website data was not easily scrape-able using BeautifulSoup. In order to get around this issue, our team used the mitmproxy program to intercept the traffic each site used with their apps. Mitmproxy only shows traffic through curl requests, so our team then converted the requests to python, to get access to the site's API's.

Another issue our team faced, once we gained access to the API's, was data protection (in the case of both apps, offered by PerimeterX). For StockX we were able to bypass the limitation by editing the headers we used when making requests. For Goat we were unable to find a bypass, and instead had to generate multiple headers in the development of our program.

A final issue we faced was dealing with the difference between the StockX and Goat API. StockX reveals a lot more information about products than Goat, greatly impacting the amount of analysis we were able to complete on the data we collected.

Calculation Files

The file “avgSales.txt” contains a dictionary with the average price of every scraped product by size across both sneaker platforms.

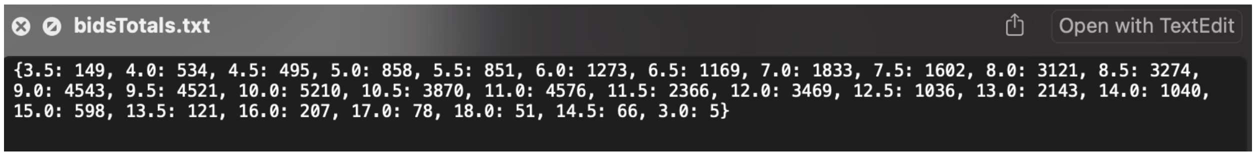


avgSales.txt

Open with TextEdit

```
{3.5: 673.07, 4.0: 394.91, 4.5: 436.82, 5.0: 356.26, 5.5: 439.79, 6.0: 380.33, 6.5: 465.45, 7.0: 365.35, 7.5: 316.89,
8.0: 276.67, 8.5: 281.51, 9.0: 270.15, 9.5: 274.42, 10.0: 270.13, 10.5: 298.15, 11.0: 288.79, 11.5: 306.59, 12.0:
289.7, 12.5: 325.71, 13.0: 307.82, 14.0: 345.24, 15.0: 439.32, 13.5: 1029.21, 16.0: 300.43, 17.0: 330.04, 18.0: 273.78,
14.5: 323.55, 3.0: 140.09, 15.5: 1176.0, 1.0: 118.0, 1.5: 137.5, 2.0: 133.43, 2.5: 141.57, 19.0: 129.0, 20.0: 9114.67}
```

The file bidsTotals.txt contains a dictionary with the total number of bids by size on StockX.

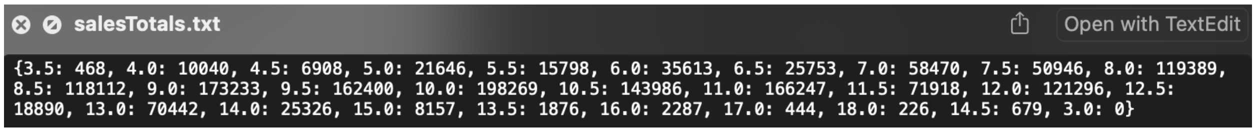


bidsTotals.txt

Open with TextEdit

```
{3.5: 149, 4.0: 534, 4.5: 495, 5.0: 858, 5.5: 851, 6.0: 1273, 6.5: 1169, 7.0: 1833, 7.5: 1602, 8.0: 3121, 8.5: 3274,
9.0: 4543, 9.5: 4521, 10.0: 5210, 10.5: 3870, 11.0: 4576, 11.5: 2366, 12.0: 3469, 12.5: 1036, 13.0: 2143, 14.0: 1040,
15.0: 598, 13.5: 121, 16.0: 207, 17.0: 78, 18.0: 51, 14.5: 66, 3.0: 5}
```

The file salesTotals.txt contains a dictionary with the total number of sales by size on StockX.



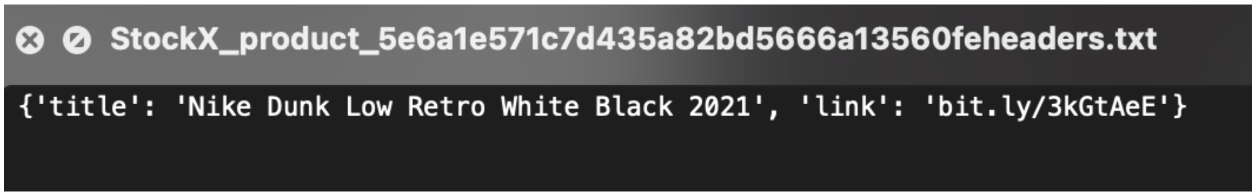
salesTotals.txt

Open with TextEdit

```
{3.5: 468, 4.0: 10040, 4.5: 6908, 5.0: 21646, 5.5: 15798, 6.0: 35613, 6.5: 25753, 7.0: 58470, 7.5: 50946, 8.0: 119389,
8.5: 118112, 9.0: 173233, 9.5: 162400, 10.0: 198269, 10.5: 143986, 11.0: 166247, 11.5: 71918, 12.0: 121296, 12.5:
18890, 13.0: 70442, 14.0: 25326, 15.0: 8157, 13.5: 1876, 16.0: 2287, 17.0: 444, 18.0: 226, 14.5: 679, 3.0: 0}
```

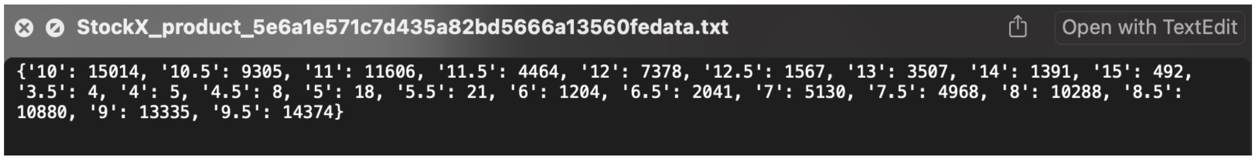
The files with the format “StockX_product_…headers.txt” contain product names, and product links that have been ran through the VigLink and Bit.ly API’s. They correspond to the “StockX_product_…data.txt” files. The same goes for the “Goat_product_…headers.txt” and “Goat_product_…data.txt” files.

Please note that since there are ~400 of these files, we are only including 4 calculations (1 of each file type).



StockX_product_5e6a1e571c7d435a82bd5666a13560feheaders.txt

```
{'title': 'Nike Dunk Low Retro White Black 2021', 'link': 'bit.ly/3kGtAeE'}
```

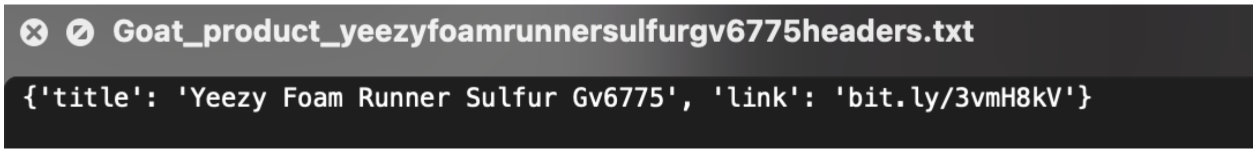


StockX_product_5e6a1e571c7d435a82bd5666a13560fedata.txt

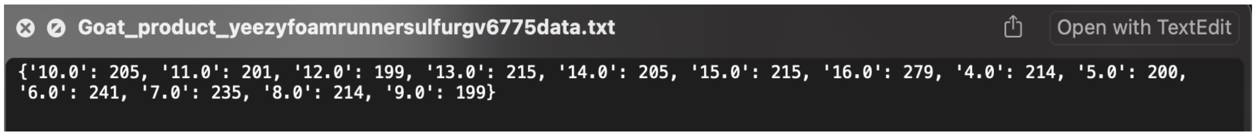
Open with TextEdit

```
{'10': 15014, '10.5': 9305, '11': 11606, '11.5': 4464, '12': 7378, '12.5': 1567, '13': 3507, '14': 1391, '15': 492,
'3.5': 4, '4': 5, '4.5': 8, '5': 18, '5.5': 21, '6': 1204, '6.5': 2041, '7': 5130, '7.5': 4968, '8': 10288, '8.5':
10880, '9': 13335, '9.5': 14374}
```

Abir Ahmed, Ioannis Paranikas

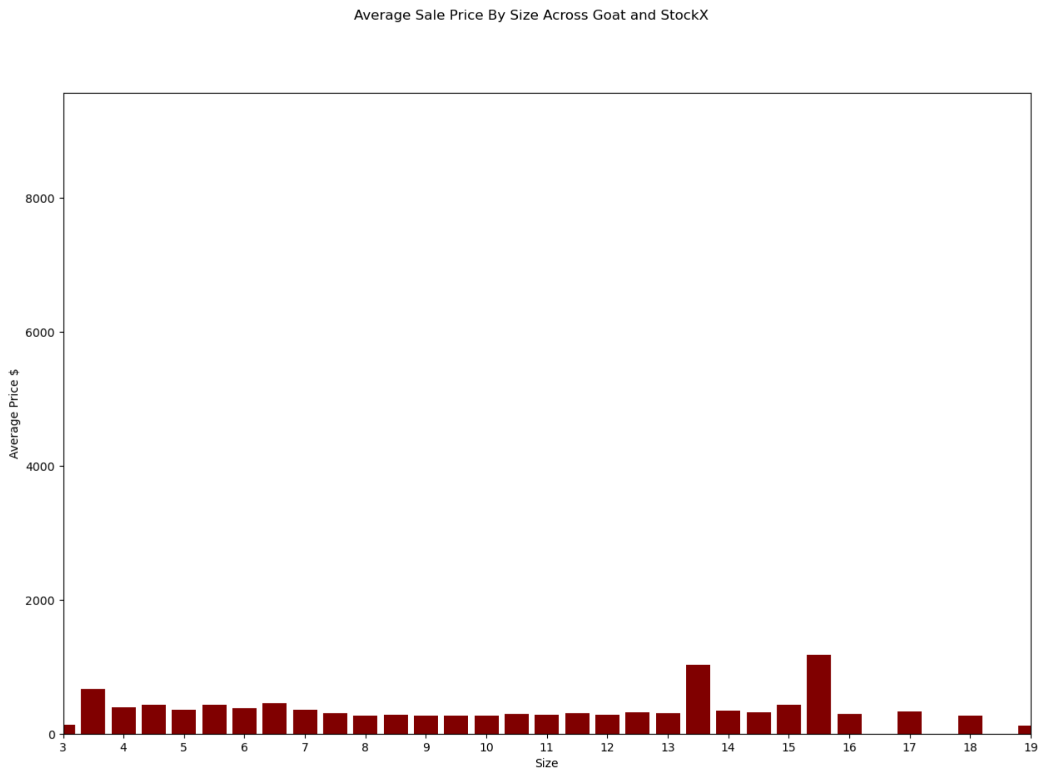

Goat_product_yeezyfoamrunnersulfurgv6775headers.txt

```
{'title': 'Yeezy Foam Runner Sulfur Gv6775', 'link': 'bit.ly/3vmH8kV'}
```

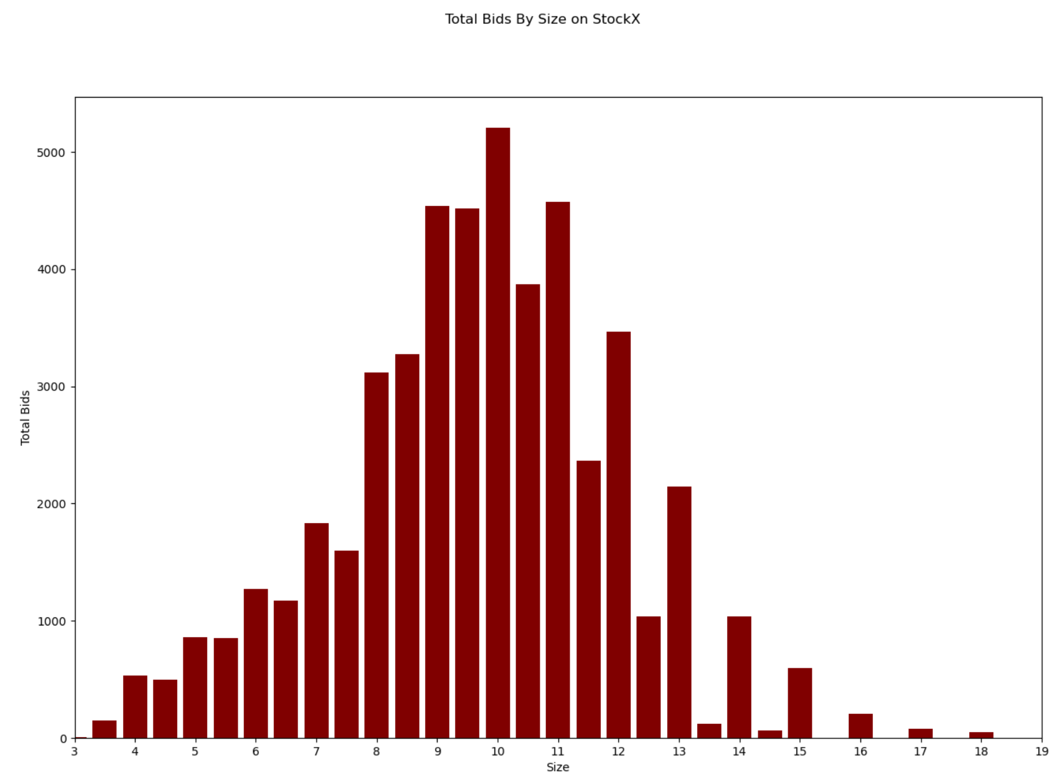

Goat_product_yeezyfoamrunnersulfurgv6775data.txt

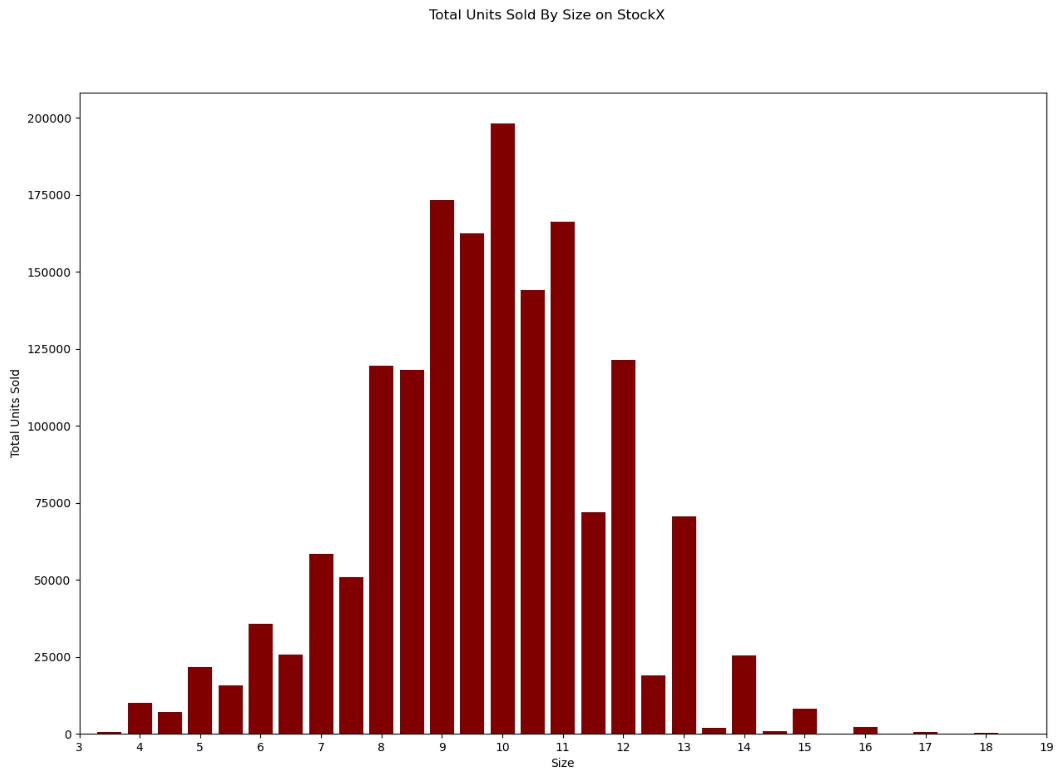
```
{'10.0': 205, '11.0': 201, '12.0': 199, '13.0': 215, '14.0': 205, '15.0': 215, '16.0': 279, '4.0': 214, '5.0': 200,
'6.0': 241, '7.0': 235, '8.0': 214, '9.0': 199}
```

Visualizations



Abir Ahmed, Ioannis Paranikas

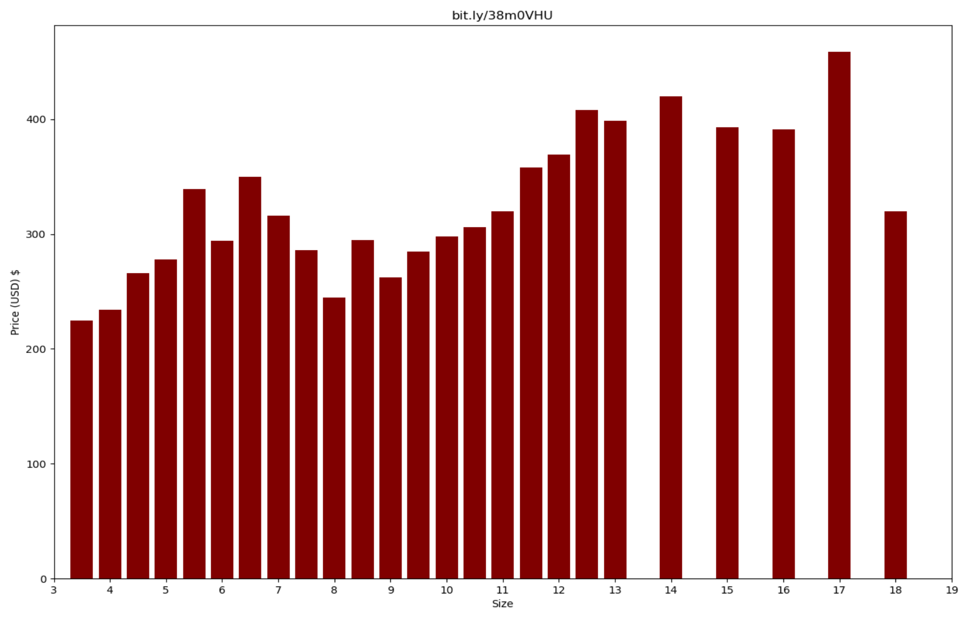




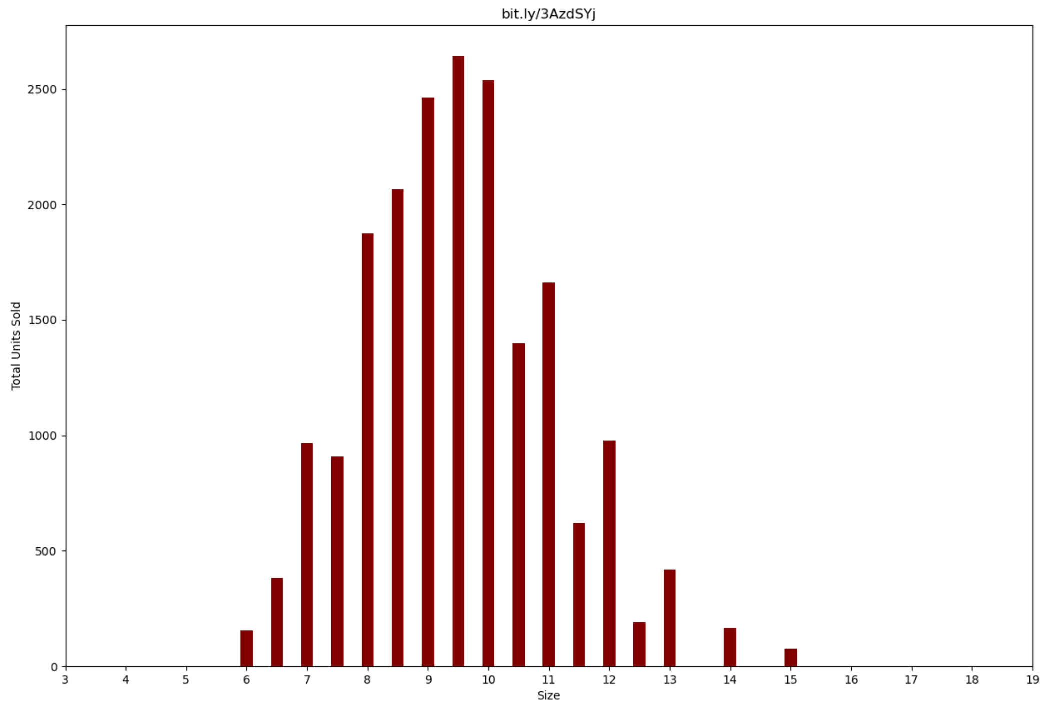
Abir Ahmed, Ioannis Paranikas

Please note that since there are ~200 of visualization files, we are only including 2 calculations (1 of each site)

Air Jordan 1 Retro High Og Patent Bred 555088 063



Nike Dunk Low Grey Fog



Abir Ahmed, Ioannis Paranikas

Instructions for Running Code:

- Navigate to “SI206FinalProject.py”
- Run the code
- Follow the instructions prompted by the program. In order to avoid any PerimeterX flagging when scraping Goat.com data, we recommend loading no more than 15 products. Use function “5” if you only have the .py file downloaded. Otherwise, feel free to use any of the functions the script describes.

Documentation for Each Function:

get_stockX_popular

Inputs: Quantity
Outputs: A list of products
API’s Used: StockX API
Description: Function takes in a quantity, and then returns the top quantity # worth of popular products on StockX. StockX defines a product as being popular if it has the most sales in the past 72 hours.

build_stockX_product

Inputs: StockX Product UUID
Outputs: Product Dictionary
API’s Used: StockX API
Description: Function takes in a UUID and builds a dictionary including product lowest ask, total asks, highest bid, total bids, total sold, average sale price, and market cap using the StockX API.

get_stockX_product_link

Inputs: StockX Product UUID
Outputs: StockX Product Link
API’s Used: StockX API
Description: Function takes in a StockX product UUID and returns its respective product link.

get_Goat_popular

Inputs: Quantity
Outputs: A list of products
API’s Used: Goat API
Description: Function takes in a quantity, and then returns the top quantity # worth of popular products on Goat.

build_goat_product

Inputs: productTemplateId
Outputs: Product Dictionary
API’s Used: Goat API

Description: Function takes in a product slug and returns the current price per size.

get_Goat_product_link

Inputs: Product Slug
Outputs: Product Link
API's Used: None
Description: Function takes in a product slug and returns its respective product link.

makeMoney

Inputs: Link
Outputs: Bit.ly link
API's Used: VigLink API, Bit.ly
Description: Function takes in a product link, and then runs it through the VigLink API to generate a product affiliate link. The function then shortens the link using the Bit.ly API.

setUpDatabase

Inputs: Database Name
Outputs: Database Cursor and Connection
API's Used: None
Description: Function takes in a name, and returns a database cursor and connection object.

build_StockX_popular_DB

Inputs: Popular StockX Styles, Database Cursor, Database Connection
Outputs: Database containing UUID, Description, and Product Links
API's Used: None
Description: Function uses data returned from get_stockX_popular and makeMoney to build StockXPopular table in database.

build_StockX_product_DB

Inputs: Product UUID, Product Data, Database Cursor, Database Connection
Outputs: Individual Product Database
API's Used: VigLink, Bit.ly
Description: Function generates the Bit.ly product link, and then populates a table using inputted product data

build_Goat_popular_DB

Inputs: Popular Goat Styles, Database Cursor, Database Connection
Outputs: Database containing ID, Slug ,and Product Links
API's Used: None
Description: Function uses data returned from get_Goat_popular and makeMoney to build StockXPopular table in database.

Abir Ahmed, Ioannis Paranikas

build_Goat_product_DB

Inputs: Product Slug, Product Data, Database Cursor, Database Connection
Outputs: Individual Product Database
API's Used: VigLink, Bit.ly
Description: Function generates the Bit.ly product link, and then populates a table using inputted product data.

read_StockX_popular_table

Inputs: Database Cursor, Database Connection
Outputs: Individual Product Databases for each product in StockX Popular Products.
API's Used: StockX API
Description: Function reads the StockXPopular Database and creates a database for every popular product (1 at a time to comply with class 25 limit).

read_Goat_popular_table

Inputs: Database Cursor, Database Connection
Outputs: Individual Product Databases for each product in Goat Popular Products.
API's Used: Goat API
Description: Function reads the GoatPopular Database and creates a database for every popular product (1 at a time to comply with class 25 limit).

get_avg_price_all

Inputs: Database Cursor, Database Connection
Outputs: avgSales.txt
API's Used: None
Description: Function parses through every product database and creates a dictionary with a calculated average sale price per size. The calculation is done across products from both Goat and StockX.

total_bids_by_size_StockX

Inputs: Database Cursor, Database Connection
Outputs: bidsTotals.txt
API's Used: None
Description: Function parses through all StockX products, and calculates the total bids per size.

total_sold_by_size_StockX

Inputs: Database Cursor, Database Connection
Outputs: salesTotals.txt
API's Used: None
Description: Function parses through all StockX products, and calculates the total units sold per size.

create_StockX_product_TXT

Inputs: Database Cursor, Database Connection
Outputs: Many TXT Files
API's Used: None
Description: Function uses a JOIN function to create two TXT files per StockX product. One file is used as the header for visualization, while the other is used as the main data for visualization. The JOIN function allows the chart title to be the correct shoe name for each visualization.

create_Goat_product_TXT

Inputs: Database Cursor, Database Connection
Outputs: Many TXT Files
API's Used: None
Description: Function uses a JOIN function to create two TXT files per Goat product. One file is used as the header for visualization, while the other is used as the main data for visualization. The JOIN function allows the chart title to be the correct shoe name for each visualization.

visualize_Goat_TXTs

Inputs: None
Outputs: Visualizations
API's Used: None
Description: Function uses the TXT files created by create_Goat_product_TXT to create a data visualization using MatplotLib.

visualize_StockX_TXTs

Inputs: None
Outputs: Visualizations
API's Used: None
Description: Function uses the TXT files created by create_StockX_product_TXT to create a data visualization using MatplotLib.

visualize_avg_price_all

Inputs: None
Outputs: Visualizations
API's Used: None
Description: Function uses avgSales.txt to create a visualization using MatplotLib.

visualize_total_bids_by_size_StockX

Inputs: None
Outputs: Visualizations
API's Used: None
Description: Function uses bidsTotals.txt to create a visualization using MatplotLib.

visualize_sales_by_size_StockX

Inputs: None
Outputs: Visualizations
API's Used: None
Description: Function uses salesTotals.txt to create a visualization using MatplotLib.

initialize_popular_data
Inputs: None
Outputs: None
API's Used: None
Description: Function initializes all popular data, and asks for user input.

refresh_SKU_data
Inputs: None
Outputs: None
API's Used: None
Description: Function refreshes all popular product data, pulling from the site popular databases.

create_all_visuals
Inputs: None
Outputs: None
API's Used: None
Description: Function calls all visualization functions and generates data for them to function properly.

Main
Inputs: None
Outputs: None
API's Used: None
Description: Function uses initialize_popular_data, refresh_SKU_data, and create_all_visuals, to synthesize the use of all created functions.

Documentation of Resources:

Date	Issue Description	Location of Resource	Result
4/19	Goat and StockX do not have a public API.	http://docs.mitmproxy.org	Resolved
4/22	Mitmproxy only returns requests as Curl requests.	https://curlconverter.com	Resolved
4/23	MatplotLib does not	https://matplotlib.org/	Resolved. Converted

	chart dictionary data correctly and lists sizes from 10 to 18, then 3.5 (aka by starting number) instead of numerically.	3.5.0/api/_as_gen/matplotlib.pyplot.xticks.html	all dictionary keys to floats, and used xticks parameter.
4/25	PerimeterX limiting amount of data we are able to retrieve from Goat API.	No suitable resource found.	Unresolved. Current headers work, but are subject to flagging at any time, for any reason.

Extra Credit

Extra API1: VigLink API

ExtraAPI2: Bit.ly API

ExtraViz1-200: StockX_product_a9b5d65f88d64044af660751c421c9a7.txt.png->
Goat_product_airjordan1retrohighogpatentbred555088063.txt.png