Johns Hopkins Data Analytics Botcamp

Cyber Data Breaches and Common Passwords Technical Report

Oluwatobi Akinsanya

Temidayo Akinsanya

October 13th, 2020

TABLE OF CONTENTS

INTRODUCTION

This project Extracted, Transformed, and Loaded data from two different sources. The first data source was an Comma Separated Values (CSV) document that contained a list of all the companies that were either breached or hacked in the last 11 years; including both foriegn and domestic companies (Figure 1). The second data source was a wikipedia webpage that contained the 10,000 often used passwords (Figure 2). The data were extracted from the sources, transformed into an analysis ready format, and then loaded into a Postgres database. The combination of the two data can perhaps be valuable in informing users on which passwords to avoid, and which company may have exposed their personal information so they may change their passwords if they have an account with such a company.

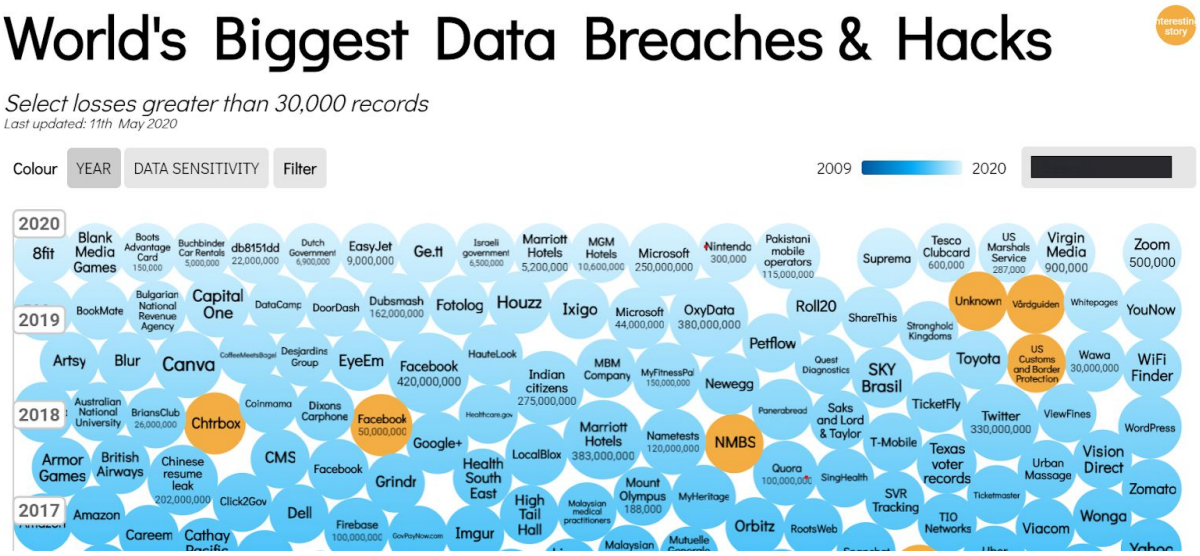


Figure 1: Cyber Data Breaches and Hacks Data Source

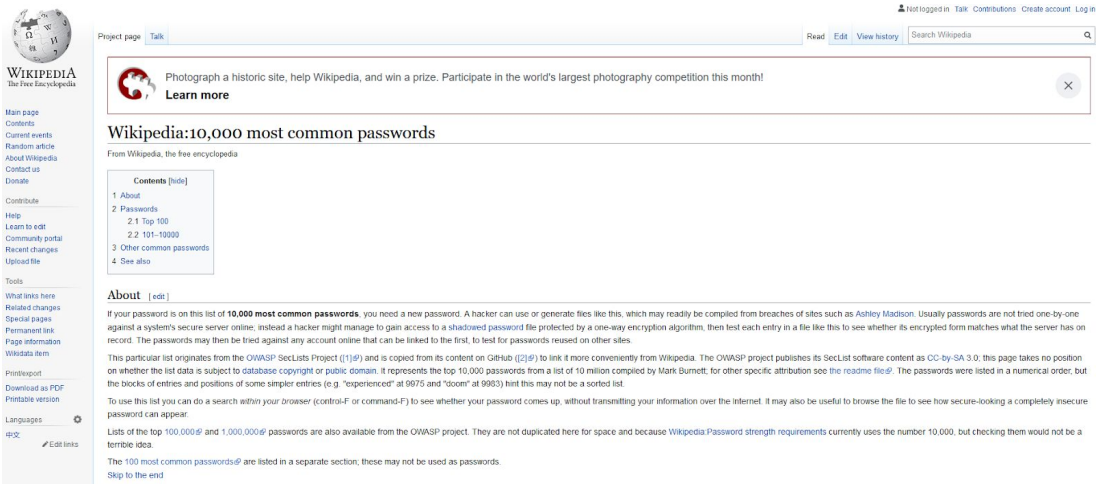


Figure 2: 10,000 Most Common Passwords Data Source

PROCESS OF DATA EXTRACTION

10,000 Most Common Passwords

The source for gathering the 10,000 often used passwords was a wikipedia webpage. The method used for the extraction was web scraping via BeautifulSoup. The passwords were in a list format so that meant examining the HTML script to identify the location of the tags where the passwords were located.

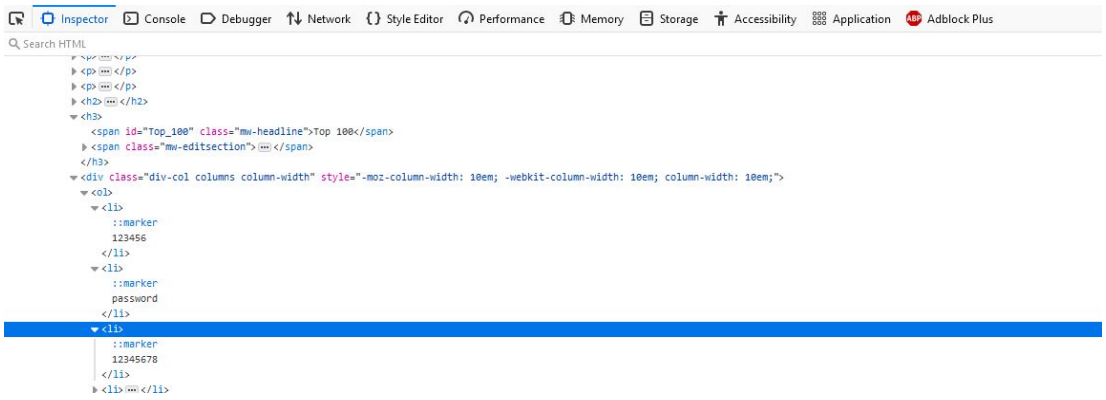


Figure 3: HTML Script

Using the Beautiful Soup library in Python, all 10,000 passwords in the list were extracted into a Python list and stored into a DataFrame. The data were separated into two categories; the top 100 passwords and the remaining 9900 to allow for a cleaner extraction.

```
#Extract data via BS4
soup = BeautifulSoup(response.text, 'html.parser')
```

```
#Extract the lists of common passwords
all_lists = soup.find_all('div', class_="div-col columns column-width")
```

Figure 4: Extraction of All Contents Within the Password Contained Division

```
#Exract the top 100 list
top_100_list = []
top_100 = all_lists[0].find_all('li')
x=np.arange(len(top_100))
for x in np.arange(len(top_100)):
    top_100_list.append(top_100[x].text)
```

```
#Extract the remaining 9900 passwords
leftover_passwords_list = []
leftover_passwords = all_lists[1].find_all('li')
x=np.arange(len(leftover_passwords))
for x in np.arange(len(leftover_passwords)):
    leftover_passwords_list.append(leftover_passwords[x].text)
```

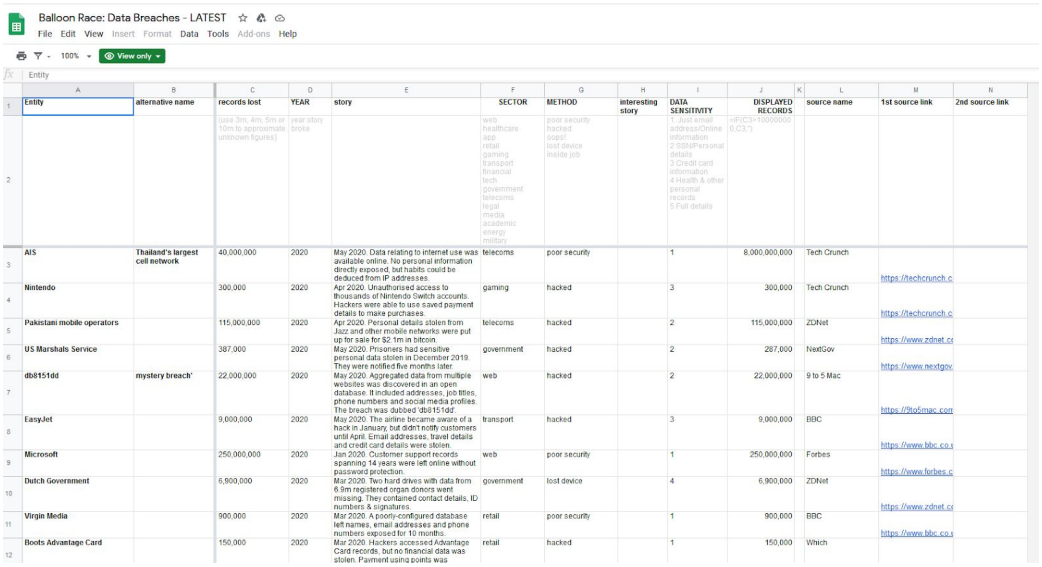
Figure 5: Using a For Loop to Extract the Text of Each Passwords Only

```
#Store lists into dataframe
top100_df = pd.DataFrame(top_100_list)
remainder_df = pd.DataFrame(leftover_passwords_list)
```

Figure 6: The Passwords Stored into a DataFrame

Cyber Breaches and Hacks

The source for gathering the Cyber Breaches and Hacks is information is beautiful. This source compiles the world's biggest data breaches and hacks that have occurred since 2009. The data from the source is displayed on their website as a visualization by year but the data is stored in a google document.



Balloon Race: Data Breaches - LATEST

File Edit View Insert Format Data Tools Add-ons Help

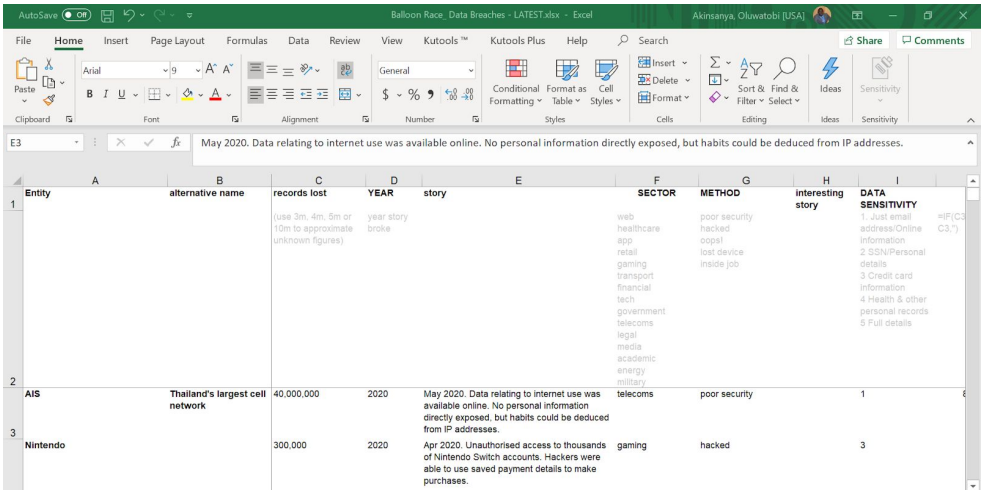
100% View only

Entity

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Entity	alternative name	records lost	YEAR	story	SECTOR	METHOD	interesting story	DATA SENSITIVITY	DISPLAYED RECORDS		source name	1st source link	2nd source link
2			(use 3m, 4m, 5m or 10m to approximate unknown figures)	year story broke		web healthcare app retail gaming transport financial tech government telecoms legal media academic energy military	poor security hacked oops! lost device inside job		1. Just email address/Online information 2 SSN/Personal details 3 Credit card information 4 Health & other personal records 5 Full details	=IF(C3=10000000 0,C3,")			
3	AIS	Thailand's largest cell network	40,000,000	2020	May 2020. Data relating to internet use was available online. No personal information directly exposed, but habits could be deduced from IP addresses.	telecoms	poor security		1	8,000,000,000		Tech Crunch		
4	Nintendo		300,000	2020	Apr 2020. Unauthorised access to thousands of Nintendo Switch accounts. Hackers were able to use saved payment details to make purchases.	gaming	hacked		3	300,000		Tech Crunch	https://techcrunch.c	
5	Pakistani mobile operators		115,000,000	2020	Apr 2020. Personal details stolen from Jazz and other mobile networks were put up for sale for $2.1m in bitcoin.	telecoms	hacked		2	115,000,000		ZDNet	https://techcrunch.c	
6	US Marshals Service		387,000	2020	May 2020. Prisoners had sensitive personal data stolen in December 2019. They were notified five months later.	government	hacked		2	287,000		NextGov	https://www.zdnet.c	
7	db8151dd	mystery breach'	22,000,000	2020	May 2020. Aggregated data from multiple websites was discovered in an open database. It included addresses, job titles, phone numbers and social media profiles. The breach was dubbed 'db8151dd'.	web	hacked		2	22,000,000		9 to 5 Mac	https://www.nextgov	
8	EasyJet		9,000,000	2020	May 2020. The airline became aware of a hack in January, but didn't notify customers until April. Email addresses, travel details and credit card details were stolen.	transport	hacked		3	9,000,000		BBC	https://9to5mac.com	
9	Microsoft		250,000,000	2020	Jan 2020. Customer support records spanning 14 years were left online without password protection.	web	poor security		1	250,000,000		Forbes	https://www.bbc.co.u	
10	Dutch Government		6,900,000	2020	Mar 2020. Two hard drives with data from 6.9m registered organ donors went missing. They contained contact details, ID numbers & signatures.	government	lost device		4	6,900,000		ZDNet	https://www.forbes.c	
11	Virgin Media		900,000	2020	Mar 2020. A poorly-configured database left names, email addresses and phone numbers exposed for 10 months.	retail	poor security		1	900,000		BBC	https://www.zdnet.co	
12	Boots Advantage Card		150,000	2020	Mar 2020. Hackers accessed Advantage Card records, but no financial data was stolen. Payment using points was	retail	hacked		1	150,000		Which	https://www.bbc.co.u	

Figure 7: Cyber Data Breaches and Hacks Data

To extract this file, we simply had to click file in the google document and download the data file as an excel sheet.



AutoSave Off — Balloon Race_Data Breaches - LATEST.xlsx - Excel — Akinsanya, Oluwatobi [USA]

File Home Insert Page Layout Formulas Data Review View Kutools ™ Kutools Plus Help Search Share Comments

Paste Arial 9 General Conditional Formatting Format as Table Cell Styles Insert Delete Format Sort & Filter Find & Select Ideas Sensitivity

Clipboard Font Alignment Number Styles Cells Editing Ideas Sensitivity

E3 May 2020. Data relating to internet use was available online. No personal information directly exposed, but habits could be deduced from IP addresses.

	A	B	C	D	E	F	G	H	I
1	Entity	alternative name	records lost	YEAR	story	SECTOR	METHOD	interesting story	DATA SENSITIVITY
2			(use 3m, 4m, 5m or 10m to approximate unknown figures)	year story broke		web healthcare app retail gaming transport financial tech government telecoms legal media academic energy military	poor security hacked oops! lost device inside job		1. Just email address/Online information 2 SSN/Personal details 3 Credit card information 4 Health & other personal records 5 Full details
3	AIS	Thailand's largest cell network	40,000,000	2020	May 2020. Data relating to internet use was available online. No personal information directly exposed, but habits could be deduced from IP addresses.	telecoms	poor security		1
	Nintendo		300,000	2020	Apr 2020. Unauthorised access to thousands of Nintendo Switch accounts. Hackers were able to use saved payment details to make purchases.	gaming	hacked		3

Figure 8: Cyber Data Breaches and Hacks Data

PROCESS OF DATA TRANSFORMATION

10,000 Most Common Passwords

The extracted data was transformed into an analysis ready format by changing the column names, and removing empty strings. The pre-existing column header was the number 0 and that was changed into a readable column name using the ".rename" python function. The DataFrame also contained empty columns because the HTML script had an empty string. Since it was an empty string instead of "NaN" value, the python drop function was used on the row with the empty string.

```
top100_df= top100_df.rename(columns={0:"common_passwords"})
remainder_df= remainder_df.rename(columns={0:"common_passwords"})
remainder_df = remainder_df.drop([0])
```

Figure 9: Data Cleaned and Transformed

Upon complete transformation of the data, both DataFrames were merged into one DataFrame.

```
#Merge both dataframes into one
common_passwords_df = pd.merge(left=top100_df, right=remainder_df,
                               on="common_passwords", how='outer')
```

Figure 10: Merging of the DataFrame

	common_passwords
0	123456
1	password
2	12345678
3	qwerty
4	123456789
...	...
9995	caca
9996	c2h5oh
9997	bubbles1
9998	brook

Figure 11: Merged DataFrame

Cyber Breaches and Hacks

After extracting the data by downloading the excel file, the data needed to be cleaned and formatted for analysis. To clean the data, we first deleted the first row of data which only contained title comments that are not a part of the raw data. Next, we searched for duplicated information in the data and columns with no information provided or null values. Then, we deleted the column with no information and renamed the null value appropriately in each column. Finally, we reformatted the case of the column titles and renamed the column titles to be user and database friendly for our data upload.

```
data_breaches.drop([0], inplace=True)
```

Figure 12: Dropping First Row

```
data_breaches.duplicated()
```

Figure 13: Searching for Duplicate Rows

```
data_breaches.isnull().sum()
```

```
Entity                  0
alternative name      225
records lost            1
YEAR                    0
story                   0
SECTOR                  0
METHOD                  1
interesting story     300
DATA SENSITIVITY        0
DISPLAYED RECORDS     310
Unnamed: 10           369
source name             0
1st source link         0
2nd source link       337
dtype: int64
```

Figure 14: Searching for Nulls and Empty Columns (Null = 369)

```
del data_breaches["Unnamed: 10"]
```

Figure 15: Deleting Empty Column

```
col_values = {'alternative name': "none",
              'records lost': "unknown",
              "METHOD": "hacked",
              "interesting story":"n",
              "DISPLAYED RECORDS": "unknown",
              '2nd source link': "none"}
data_breaches.fillna(value=col_values, inplace=True)
```

Figure 16: Renaming the Null Value Appropriately

```
data_breaches["alternative name"]=data_breaches["alternative name"].str.title()
data_breaches["SECTOR"]=data_breaches["SECTOR"].str.capitalize()
data_breaches["METHOD"]=data_breaches["METHOD"].str.title()
data_breaches["interesting story"]=data_breaches["interesting story"].str.capitalize()
```

Figure 17: Reformatting the Case of the Column Titles

```
data_breaches.rename(columns={"Entity": "company_agency",
                              "alternative name": "company_agency_description",
                              "records lost": "total_records_lost",
                              "YEAR" : "year",
                              "story" : "background_story",
                              "SECTOR" : "sector",
                              "METHOD" : "method",
                              "interesting story" : "interesting_story",
                              "DATA SENSITIVITY" : "data_sensitivity_level",
                              "DISPLAYED RECORDS" : "displayed_records",
                              "source name" : "source_name",
                              "1st source link" : "primary_source",
                              "2nd source link" : "secondary_source"},inplace=True)
```

Figure 18: Renaming the Column Titles

PROCESS OF DATA LOADING

Once the data was transformed, it was loaded into a relational database. The reason a relational database was chosen was because of the structure of the data set, and the use of structured query language. Prior to loading the data, a database was created in postgres along with a corresponding schema.

```
CREATE TABLE public.common_passwords
(
    id integer NOT NULL DEFAULT nextval('common_passwords_id_seq'::regclass),
    common_passwords character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT common_passwords_pkey PRIMARY KEY (id)
)
```

Figure 19: Creating Table within Database

After creating the database and table, connection was established with the database in the jupyter notebook; then the DataFrame was loaded into the postgres database

```
#create connection to SQL DB
rds_connection_string = "postgres:MolovesTemi2020!@localhost:5432/securi
engine = create_engine(f'postgresql://{rds_connection_string}')
```

```
common_passwords_df.to_sql(name='common_passwords', con=engine, if_exist
```

Figure 20: Loading of Data into Database for Common Passwords

```
rds_connection_string = "postgres:Lekan011singer!@localhost:5432/security_breaches_db"
engine = create_engine(f'postgresql://{rds_connection_string}')
```

```
data_breaches.to_sql(name='data_breaches', con=engine, if_exists='append', index=False)
```

Figure 21: Loading of Data into Database for Cyber Data Breach

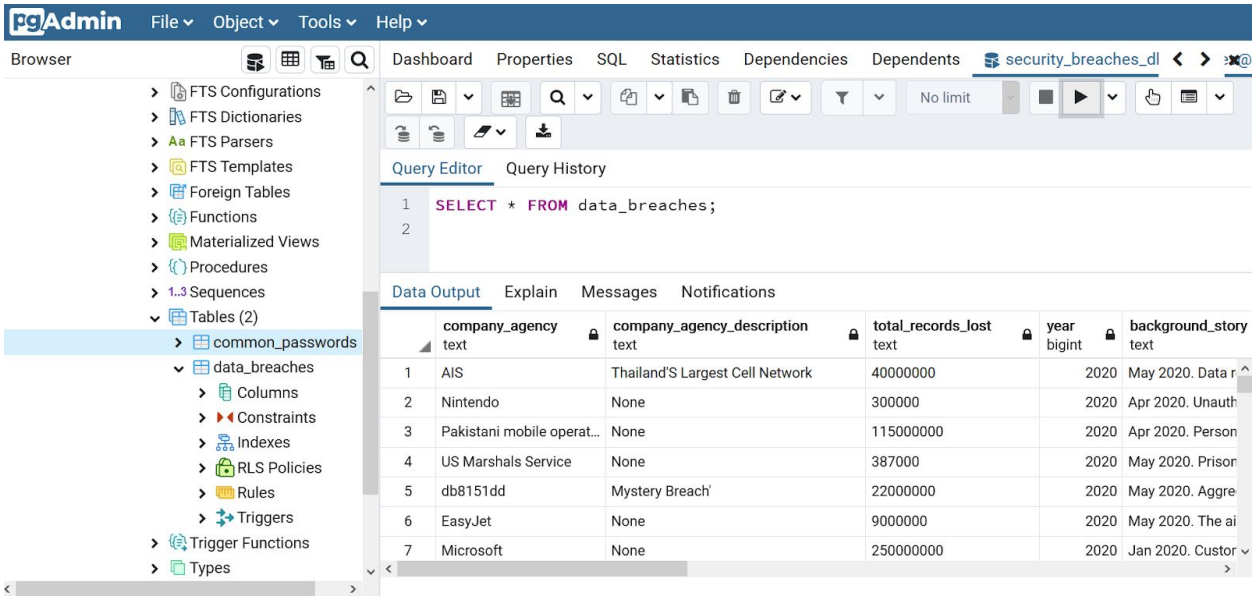


Figure 21: Database for Cyber Data Breach

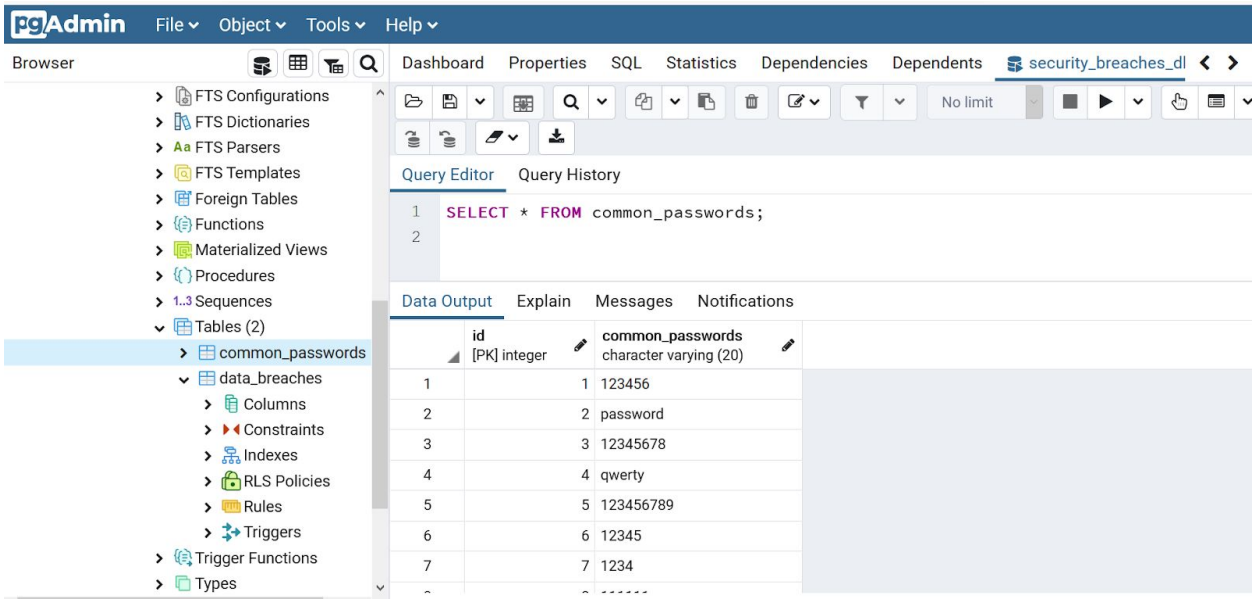


Figure 22: Database for Common Passwords

CONCLUSION

Performing ETL on two distinct data sources was a valuable learning experience as it involved utilizing different methodologies to extract the data. It was a prime opportunity to put recently learned skills into actionable practice. Additionally, there is an acute awareness of the needed transformation that must take place in order for a data set to be analysis and database ready.