OBJECTIVE

Obtain the **largest ETFs (AUM > $2bn)** in the world by web parsing using python, retrieving their holdings, and building a recommendation so that investors can find a better alternative.

OVERVIEW

#1

Setting up a Database

#2

Determine the universe of ETF

#3

Parse through websites for Stocks

- iShares
- Invesco
- Stock Analysis

#4

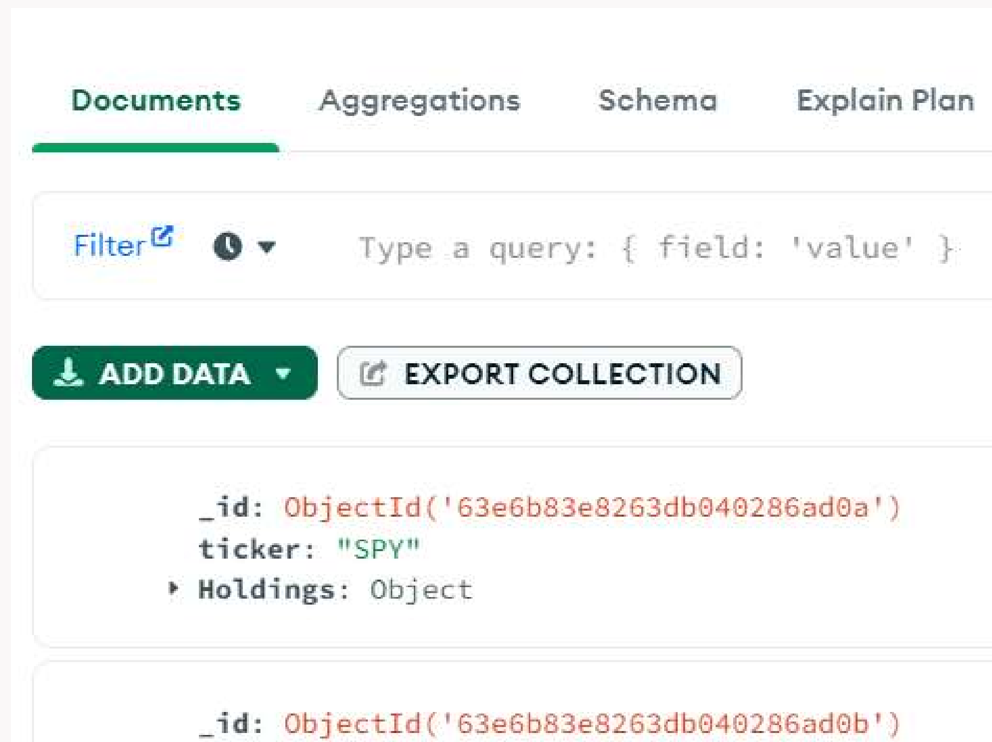
Executing similarity function

#5

ETF Recommendations

DATABASE

- **Scalable**
- **Flexible**
- **Easily Accessible**



ETF MASTER LIST

- Imported dataset of 1394 ETF tickers from Bloomberg
  - https://www.dropbox.com/s/1a4u95oj30x68k8/ETF1.xlsx?raw=1
- Data cleaning
- Filtered the spreadsheet for ETFs of > $2Bn asset values
- After filtering, we get 188 ETFs

2	Vanguard Total Stock Market ETF
3	Vanguard S&P 500 ETF
4	Invesco QQQ Trust Series 1
...	...
1389	Subversive Metaverse ETF
1390	Strategas Macro Thematic Opportunities ETF
1391	Subversive Mental Health ETF
1392	Strive 1000 Value ETF
1393	Clockwise Capital Innovation ETF

1394 rows × 7 columns

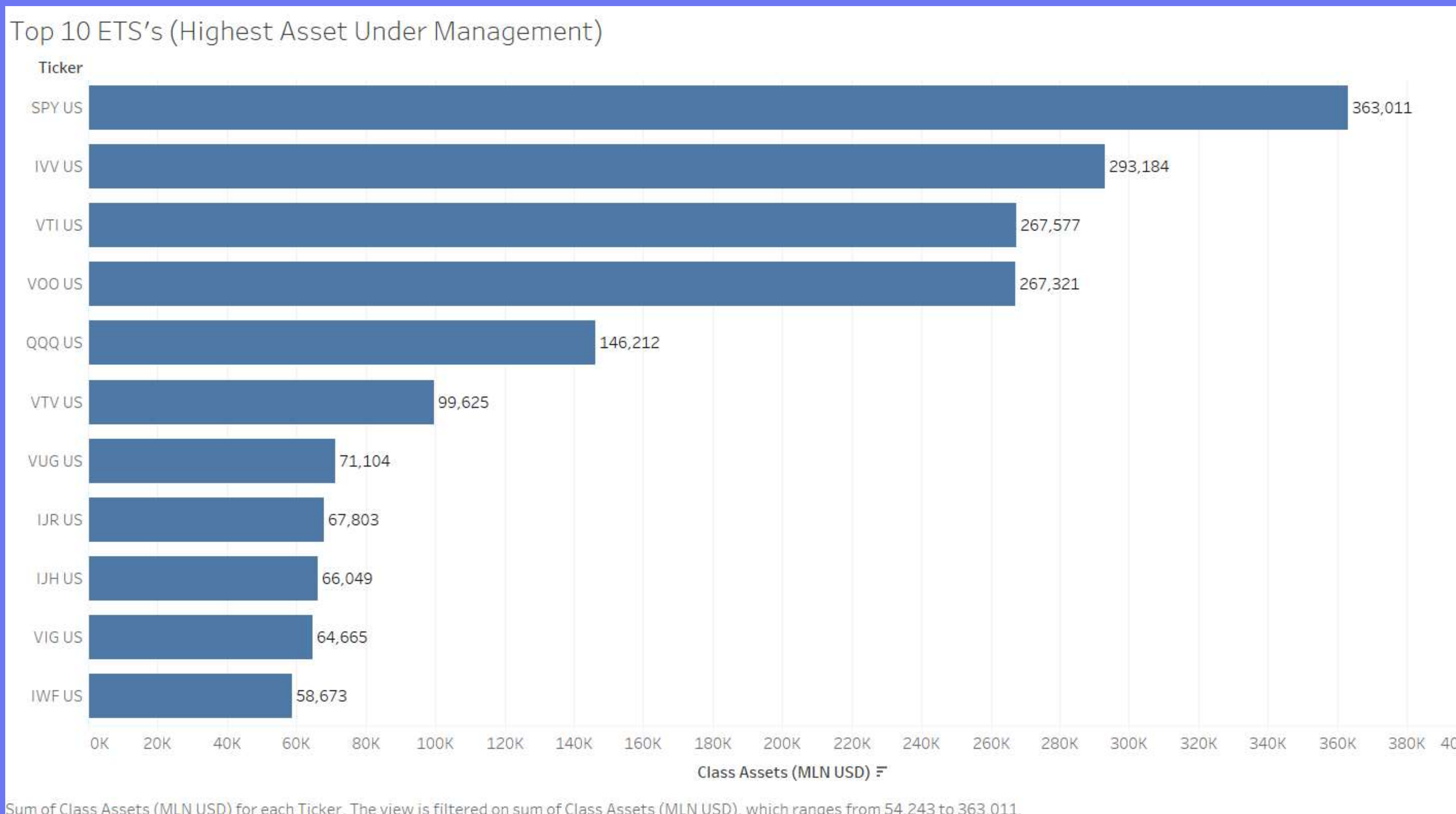
Data cleaning

Filtering

0	SPDR S&P 500 ETF Trust	SPY	363.01072	36301
1	iShares Core S&P 500 ETF	IVV	293.18409	29318
2	Vanguard Total Stock Market ETF	VTI	267.57691	26757
3	Vanguard S&P 500 ETF	VOO	267.32103	26732
4	Invesco QQQ Trust Series 1	QQQ	146.21217	14621
...	...	...	...	
183	Invesco S&P 500 GARP ETF	SPGP	2.12333	212
184	iShares Global Energy ETF	IXC	2.10788	210
185	Pacer Trendpilot US Large Cap ETF	PTLC	2.10146	210
186	VictoryShares US EQ Income Enhanced Volatility...	CDC	2.04487	204
187	iShares US Financials ETF	IYF	2.02632	202

188 rows × 7 columns

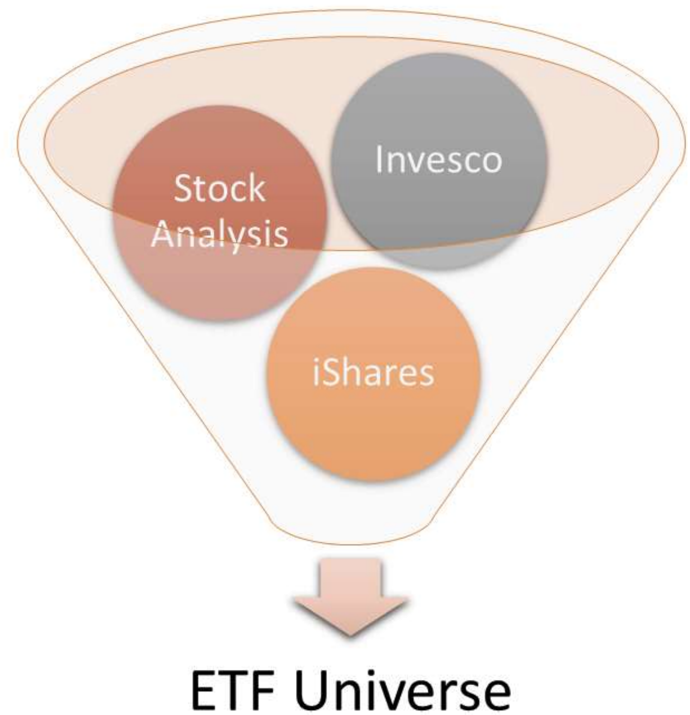
TOP TEN ETF'S


Top 10 ETS's (Highest Asset Under Management)
Ticker
SPY US
IVV US
VTI US
VOO US
QQQ US
VTV US
VUG US
IJR US
IJH US
VIG US
IWF US
363,011
293,184
267,577
267,321
146,212
99,625
71,104
67,803
66,049
64,665
58,673
0K
20K
40K
60K
80K
100K
120K
140K
160K
180K
200K
220K
240K
260K
280K
300K
320K
340K
360K
380K
400
Class Assets (MLN USD)
Sum of Class Assets (MLN USD) for each Ticker. The view is filtered on sum of Class Assets (MLN USD), which ranges from 54,243 to 363,011.

WEB-MINING

Using the **BeautifulSoup** python library we can scrap and extract the data [Stock compostion] from the following websites

- iShares
- Invesco
- Stock Analysis



WEB MINING - iShares

- Extract ETF holdings data from ishares.com
- Assume all tickers are made of <= 4 letters; filter for equity only
- Construct a **getiSharesHoldings** function that takes in an ETF ticker as input, returns a list of holdings and their market value weights
- Push data to MongoDb

```python
# Get all the href text
# <a> + href makes the link clickable, so we are trying to find all <a> plus the href text
# store everything we want to capture in a dictionary ticToURL
ticToURL={}

for row in html.find_all('tr'):
  try:
    for data in row.find_all('a'):
      if len(data.text)>0 and len(data.text)<5:
        # print(f'Ticker: {data.text} -> Link {data["href"]}')
        ticToURL[data.text]='https://www.ishares.com/'+data['href']+'/1467271812596.ajax?fileType=csv&fileName=IWM_holdings&dataType=fund'
  except:
    0
#Then get the CSV from ticToURL mapped URL
def getiShareHoldings(etfname):
  df=pd.read_csv(ticToURL[etfname],skiprows=range(0,9), thousands=',') #Read CSV from URL we did in step 1
  df['Ticker']=df['Ticker'].str.strip()
  df=df[df['Asset Class']=='Equity'].set_index('Ticker')          #Select only Equity rows
  return (df["Market Value"]/df["Market Value"].sum()).to_dict()       # Convert to a dict
```

```python
# apply this function to all ETFs that we got from iShares
# Retrieved 84 ETF holdings data from iShares
etfs = pd.DataFrame(list(ClassDb["Etf_List"].find()))
empty_etf = []
for i in ETFs_universe['Ticker']:
  if checkPresence(etfs , i.strip()):
    continue
  ticker_name = i.strip()
  try:
    temp_dict = {}
    value = getiShareHoldings(ticker_name)
    temp_dict = {"ticker":ticker_name,
                 "Holdings":value}
    # Pass result to MongoDB
    try:
      result = ClassDb["Etf_List"].insert_one(temp_dict)
    except:
      pass
  except:
    empty_etf.append(ticker_name)
    pass

len(empty_etf)
```

WEB MINING - Invesco

- Extract ETF holdings data from investco.com
- Assume all tickers are made of <= 4 letters; filter for equity only
- Easier implementation because of direct download from URL (BeautifulSoup is not needed)
- Construct a **GetInvestcoHoldings** function that takes in an ETF ticker as input, returns a list of holdings and their market value weights
- Push updated dataset to MongoDb

```
url=f'https://www.invesco.com/us/financial-products/etfs/holdings/main/holdings/0?audienceType=Investor&action=download&ticker={etfna
investcodf=pd.read_csv(url, thousands=',')
investcodf.fillna(0.0)
investcodf['Holding Ticker']=investcodf['Holding Ticker'].str.strip()

try:
  investcodf=investcodf.set_index('Holding Ticker')
  return (investcodf['MarketValue']/investcodf['MarketValue'].sum()).to_dict()
except:
  investcodf=investcodf.set_index(' Holding Ticker')
  return (investcodf[' MarketValue']/investcodf[' MarketValue'].sum()).to_dict()
```

```
getInvestcoHoldings("QQQ")
```

```
{'MSFT': 0.12231513013708659,
 'AAPL': 0.12046810315349255,
 'AMZN': 0.06264032671986491,
 'NVDA': 0.04455483370586754,
 'TSLA': 0.040918040452112774,
 'GOOG': 0.03631187857751945,
 'GOOGL': 0.03615174452930426,
 'META': 0.032440226566323734,
 'AVGO': 0.01968405356774379,
 'PEP': 0.01930138966947347,
 'COST': 0.017966588468190645,
 'CSCO': 0.015565797311244826,
 'TMUS': 0.014410902963215604,
 'ADBE': 0.014166588624610833,
```

```
etfs = pd.DataFrame(list(ClassDb["Etf_List"].find()))
empty_etf_inv = []
for i in ETFs_universe['Ticker']:
  if checkPresence(etfs , i.strip()): continue
  ticker_name = i.strip()
  try:
    temp_dict = {}
    value = getInvestcoHoldings(ticker_name)
    temp_dict = {"ticker":ticker_name,
                 "Holdings":value}
    # Pass result to MongoDB
    try:
      result = ClassDb["Etf_List"].insert_one(temp_dict)
    except:
      pass
  except:
    empty_etf_inv.append(ticker_name)
    pass

len(empty_etf_inv)
```

WEB MINING - Stock Analysis

- Assume all tickers are made of <= 4 letters; filter for equities only.
- Use Beautiful Soup to parse a complicated HTML document to a tree of Python objects
- Construct a function called **getSAHoldings** that takes in an ETF ticker as input, and returns a list of holdings and their market value weights
- Push the updated dataset to MongoDB
- Keep track of tickers that have not extracted at the same time

```python
def getSAHoldings(etf):
  #Get holdings
  url = f'https://stockanalysis.com/etf/{etf}/holdings/'
  try:
    headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36'}
    req = Request(url=url,headers=headers)
    resp = urlopen(req)
  except:
    raise Exception(f'Error for {etf}')
  html = BeautifulSoup(resp, features="lxml")
  holdings={}
  for row in html.find('table').find_all('tr'):
    # cell is stock name
    # remove numbers
    # if cell is entirely made of numbers, remove
    cells=[d.text for d in row.find_all('td')]
    if len(cells)<4:
      continue
    wgt=float(cells[3][:-1]) #Remove % sign
    holdings[(cells[1].strip()).replace('.','')]=wgt*0.01
  return holdings
```

```python
def checkPresence(etfs , ticker):

  result=etfs['ticker'].isin([ticker])
  if result.any():
      return True
  else:
      return False
```

```python
# loop over all ETFs from StockAnalysis and push data to MongoDb
etfs = pd.DataFrame(list(ClassDb["Etf_List"].find()))
empty_etf_sa = [] # list of etfs that didn't get holding data from StockAnalysis
for i in ETFs_universe['Ticker']:
  if checkPresence(etfs , i.strip()): continue
  ticker_name = i.strip()
  try:
    temp_dict = {}
    value = getSAHoldings(ticker_name)
    temp_dict = {"ticker":ticker_name,
                 "Holdings":value}
  # Pass result to MongoDB
    try:
      result = ClassDb["Etf_List"].insert_one(temp_dict)
    except:
      pass
  except:
    empty_etf_sa.append(ticker_name)
    pass
```

FINAL DATASET

Data retrieved from mongodb

	_id	ticker	
0	63e6b83e8263db040286ad0a	SPY	{'AAPL': 0.066, 'MSFT': 0.0575
1	63e6b83e8263db040286ad0b	IVV	{'AAPL': 0.06620000000000001,
2	63e6b83f8263db040286ad0c	VTI	{'AAPL': 0.053700000000000005,
3	63e6b83f8263db040286ad0d	VOO	{'AAPL': 0.0631, 'MSFT': 0.05400
4	63e6b83f8263db040286ad0e	QQQ	{'MSFT': 0.1222, 'AAPL': 0.12050
...	...	...	
183	63e6c60da740be09cad73c64	XLV	{'UNH': 0.09210000000000002, 'J
184	63e6c60da740be09cad73c65	XLY	{'AMZN': 0.2346, 'TSLA': 0.15
185	63e6c60da740be09cad73c66	XME	{'CLF': 0.0525, 'UEC': 0.047400
186	63e6c664bb75c700cdb2c790	XOP	{'PBF': 0.025400000000000002,
187	63e6c664bb75c700cdb2c791	XYLD	{'AAPL': 0.0688, 'MSFT': 0.06, '

188 rows × 3 columns

Data converted to Dataframe in order to pass through cosine similarity

	SPY	IVV	VTI	VOO	QQQ
stock tickers					
AAPL	0.0660	0.0662	0.0537	0.0631	0.1205
MSFT	0.0575	0.0571	0.0455	0.0540	0.1222
AMZN	0.0255	0.0259	0.0220	0.0268	0.0626
GOOGL	0.0166	0.0184	0.0145	0.0173	0.0361
NVDA	0.0163	0.0158	0.0112	0.0142	0.0445

5 rows × 188 columns

SIMILARITY FUNCTION

- We chose Cosine similarity instead of Jaccardi because
  - Jaccardi similarity is used in binary cases where it is symmetric (equal importance) and asymmetric (different level of importance)
  - whereas cosine similarity measures the cosine of the angle between two vectors, which is invariant to the magnitude of the vectors
- Using the Scipy library in python we import the cosine function.
- We execute the cosine similarity function where the data frame of a certain ETF is passed which has the ETF ticker and its weight and returns the cosine similarity value.

	ticker	Cosine_DIS
0	QQQM	0.999984
1	ONEQ	0.939253
2	QYLD	0.935634
3	SCHG	0.930735
4	IWF	0.925692
5	VUG	0.925330
6	MGK	0.924978
7	IWY	0.924556
8	VONG	0.920500
9	ESGV	0.897405

RECOMMENDATIONS

- By executing the Cosine similarity we get a list of similar ETFs.
- For final recommendations, we have considered the following factors
  - Less expense ratio
  - Less P/E ratio
  - Low Beta Value
- Our recommendation for QQQ (0.20%, 21.58, 1.09)
  - ESGV - 89.7%, 0.09%, 19.7, 1.03
  - QQQM - 99%, 0.15%, 21.58, 1.13
  - IWF - 92.5%, 0.18%, 26.14, 1.07
  - SCHG - 93%, 0.04%, 27.24, 1.07

	ticker	Cosine_DIS
0	QQQM	0.999984
1	ONEQ	0.939253
2	QYLD	0.935634
3	SCHG	0.930735
4	IWF	0.925692
5	VUG	0.925330
6	MGK	0.924978
7	IWY	0.924556
8	VONG	0.920500
9	ESGV	0.897405