Greetings!

In the next steps I'll show how I would usually approach analyzing new data, ingesting and transforming/aggregating the data into useful info for the final user.

In [ ]:
```
pip install --upgrade pip
```

In [ ]:
```
pip install boto3 pandas s3fs
```

In [7]:
```
#import boto3
import pandas as pd

#Probably would use boto3 or the aws sdk to access the files properly with the cred

#s3 = boto3.client('s3',
#         aws_access_key_id='ACCESS_ID',
#         aws_secret_access_key= 'ACCESS_KEY')
#people = s3.get_object(Bucket='contrary-engineering-interview.s3.amazonaws.com', K
#companies = s3.get_object(Bucket='contrary-engineering-interview.s3.amazonaws.com'

people_df = pd.read_csv('https://contrary-engineering-interview.s3.amazonaws.com/da
companies_df = pd.read_csv('https://contrary-engineering-interview.s3.amazonaws.com
```

In [8]:
```
#Getting a glimpse into the people.csv file
people_df
```

Out[8]:

	PERSON_ID	COMPANY_NAME	COMPANY_LI_NAME	LAST_TITLE	GROUP_START_D
0	9fb750ce-4acd-40d6-a58b-f6718342364f	GoCardless	gocardless	Software Engineer	2019-01
1	9fb750ce-4acd-40d6-a58b-f6718342364f	Stealth startup	online-shoe-store	Founder / CTO	2018-01
2	9fb750ce-4acd-40d6-a58b-f6718342364f	Arkera	arkera	Software Engineer	2017-01
3	9fb750ce-4acd-40d6-a58b-f6718342364f	Imperial College London	imperial-college-london	UTA (Undergraduate Teaching Assistant)	2016-01
4	15f5d8ed-36ad-4cf7-8748-c50dc9589f59	Splunk	splunk	Software Engineer	2019-10
...	...	...	...	...	
5386	341cd181-1e2f-4198-b5b4-928475c17120	L Brands	lbrands	Infosys Consultant	2014-05
5387	341cd181-1e2f-4198-b5b4-928475c17120	Gap Inc.	gap-inc-	Infosys Consultant	2012-03
5388	341cd181-1e2f-4198-b5b4-928475c17120	Infosys	infosys	Systems Engineer	2011-06
5389	341cd181-1e2f-4198-b5b4-928475c17120	The Wind Energy Group ITESM	NaN	Research Assistant	2010-01
5390	341cd181-1e2f-4198-b5b4-928475c17120	ITESM Campus Monterrey	itesm-campus-monterrey	Teaching Assistant	2007-08

5391 rows × 6 columns

In [17]:
```
#Getting a glimpse into the companies.csv file
companies_df
```

Out[17]:

	NAME	COMPANY_LINKEDIN_NAMES	DESCRIPTION	HEADCOUNT	FOUNDING_DATE
0	ORSYP	[\n "orsyp"\n]	IT Operations Management Specialists At ORSYP ...	63.0	1986-01-01
1	Runwal	[\n "runwal"\n]	The Runwal Group was established in 1978 by it...	406.0	1978-01-01
2	Toast	[\n "toast-inc"\n]	Toast empowers restaurants of all sizes to bui...	3580.0	2011-12-22
3	DNA Medicine Institute	[\n "dna-medicine-institute"\n]	The DNA Medicine Institute, Inc. (DMI) is a me...	2.0	2004-01-01
4	Ally	[\n "ally"\n]	Ally Financial Inc. (NYSE: ALLY) is a leading ...	12120.0	1919-01-01
...	...	...	...	...	...
10706	Fidem LLC	[]	We are focused on creating telemedicine apps t...	NaN	2014-10-01
10707	Service Revolution	[\n "%EF%BC%88%E6%A0%AA%EF%BC%89%E3%82%B5%E3%...	Service Revolution is a software development c...	NaN	2009-02-02
10708	Phelen	[]	Phelen is the personnal holding of Rémi Voluer...	NaN	2020-07-23
10709	Cura Investment	[]	Cura Investment offers real estate investment ...	NaN	2002-09-28
10710	Beeldstijl Ontwerpstudio	[]	Beeldstijl Ontwerpstudio is an advertising age...	NaN	2008-04-01

10711 rows × 9 columns

In [16]:
```
#Comparing the total quantity with the distinct values of the column "NAME"
#The dataset probably have duplicates, at least with the same name
companies_df.NAME.nunique()
```

Out[16]:
```
10635
```

In [19]:
```
#No duplicates, so I'll just create a column "index" in the MySQL to create an easi
companies_df.duplicated().sum()
```

Out[19]:
```
0
```

In [ ]:
```
#Installing the necessary libs to insert data into my MySQL database
pip install PyMySQL sqlalchemy
```

In [30]:
```
import pymysql
from sqlalchemy import create_engine

#Creating the connection engine

user = 'root'
passw = 'mysql'
host =  'localhost'
port = 3306
schema = 'seven_apps'
#database = 'dataBaseName'

mydb = create_engine('mysql+pymysql://' + user + ':' + passw + '@' + host + ':' + s
```

SQL Script for the 'PEOPLE' table:

```
CREATE TABLE IF NOT EXISTS 'seven_apps'.'PEOPLE' (
  'PERSON_ID' VARCHAR(100) NOT NULL,
  'COMPANY_NAME' VARCHAR(255) NULL,
  'COMPANY_LI_NAME' VARCHAR(255) NULL,
  'LAST_TITLE' VARCHAR(255) NULL,
  'GROUP_START_DATE' DATE NULL,
  'GROUP_END_DATE' DATE NULL);
```

In [36]:
```
#Inserting people data without the index column:
people_df.to_sql(name='people', con=mydb, schema=schema, if_exists='append', index=
```

Out[36]:
```
5391
```

SQL Script for the 'COMPANIES' table:

```
CREATE TABLE IF NOT EXISTS 'seven_apps'.'COMPANIES' (
  'INDEX' INT NOT NULL,
  'NAME' VARCHAR(255) NULL,
  'COMPANY_LINKEDIN_NAMES' TEXT NULL,
  'DESCRIPTION' TEXT NULL,
  'HEADCOUNT' DECIMAL NULL,
  'FOUNDING_DATE' DATE NULL,
  'MOST_RECENT_RAISE' DECIMAL NULL,
  'MOST_RECENT_VALUATION' DECIMAL NULL,
  'INVESTORS' TEXT NULL,
  'KNOWN_TOTAL_FUNDING' DECIMAL NULL);
```

In [38]:
```
#Inserting companies data with the index column:
companies_df.to_sql(name='companies', con=mydb, schema=schema, if_exists='append',
```

Out[38]:
```
10711
```

Question 1:

What is the average total funding of all of the companies that the person with ID = '92a52877-8d5d-41a6-950f-1b9c6574be7a' has worked at?

```
SELECT p.PERSON_ID, AVG(c.KNOWN_TOTAL_FUNDING) as avg_total_funding
FROM people p
INNER JOIN companies c
ON UPPER(p.COMPANY_NAME) = UPPER(c.NAME)
WHERE p.PERSON_ID = '92a52877-8d5d-41a6-950f-1b9c6574be7a';
```

In [42]:
```
question1_sql = '''SELECT p.PERSON_ID, AVG(c.KNOWN_TOTAL_FUNDING) as avg_total_fund
FROM seven_apps.people p
INNER JOIN seven_apps.companies c
ON UPPER(p.COMPANY_NAME) = UPPER(c.NAME)
WHERE p.PERSON_ID = '92a52877-8d5d-41a6-950f-1b9c6574be7a';'''

print(pd.read_sql(sql=question1_sql, con=mydb))
```

```
                              PERSON_ID  avg_total_funding
0  92a52877-8d5d-41a6-950f-1b9c6574be7a        108000000.0
```

Question 2:

How many companies are in the companies table that no people in the people table have worked for?

```
SELECT count(*) AS quantity
FROM seven_apps.companies c
LEFT JOIN seven_apps.people p
ON UPPER(c.NAME) = UPPER(p.COMPANY_NAME)
WHERE p.COMPANY_NAME IS NULL
```

In [43]:
```
question2_sql = '''SELECT count(*) AS quantity
FROM seven_apps.companies c
LEFT JOIN seven_apps.people p
ON UPPER(c.NAME) = UPPER(p.COMPANY_NAME)
WHERE p.COMPANY_NAME IS NULL'''

print(pd.read_sql(sql=question2_sql, con=mydb))
```

```
   quantity
0      9389
```

Question 3:

What are the ten most popular companies that these 1,000 people have worked for?

```
SELECT COMPANY_NAME, count(*) as quantity
FROM people
GROUP BY COMPANY_NAME
ORDER BY count(*) DESC
LIMIT 10;
```

In [45]:
```
question3_sql = '''SELECT COMPANY_NAME, count(*) as quantity
FROM seven_apps.people
GROUP BY COMPANY_NAME
ORDER BY count(*) DESC
LIMIT 10;'''

print(pd.read_sql(sql=question3_sql, con=mydb))
```

```
                 COMPANY_NAME  quantity
0                   Microsoft        90
1                      Amazon        81
2           Intel Corporation        55
3                      Google        45
4                       Apple        24
5  Hewlett Packard Enterprise        23
6                    Facebook        21
7           Texas Instruments        19
8             Hewlett-Packard        17
9                        Meta        15
```

Question 4:

Identify company founders in the people table.

```
SELECT DISTINCT LAST_TITLE
FROM seven_apps.people
WHERE UPPER(last_title) like '%FOUNDER%';
```

Then identify the companies that these people have founded and list the top three largest companies by headcount, along with the name of that company and the person ID of the founder(s)

```
SELECT
    c.NAME as COMPANY,
    p.PERSON_ID as FOUNDER,
    c.HEADCOUNT
FROM seven_apps.people p
INNER JOIN seven_apps.companies c
on p.COMPANY_NAME = c.NAME
WHERE UPPER(p.last_title) like '%FOUNDER%'
GROUP BY c.NAME, p.PERSON_ID, c.HEADCOUNT
ORDER BY c.HEADCOUNT DESC
LIMIT 3;
```

In [50]:
```
question4_sql = '''SELECT
        c.NAME as COMPANY,
    p.PERSON_ID as FOUNDER,
    c.HEADCOUNT
FROM seven_apps.people p
INNER JOIN seven_apps.companies c
on p.COMPANY_NAME = c.NAME
WHERE UPPER(p.last_title) like '%%FOUNDER%%'
GROUP BY c.NAME, p.PERSON_ID, c.HEADCOUNT
ORDER BY c.HEADCOUNT DESC
LIMIT 3;'''

#Obs.: Added the double % just to escape the character on python

print(pd.read_sql(sql=question4_sql, con=mydb))
```

```
             COMPANY                                FOUNDER  HEADCOUNT
0             Dafiti  bb0d8489-4360-4a94-bd3d-c079f75afc96     2907.0
1  eBay for Business  a292842c-475e-4b4f-9671-fb09536c472e     1336.0
2             UWorld  c6f69f63-c7d5-419f-af34-d0cccf544e18      439.0
```

Question 5:

For each person in the people table, identify their 2nd most recent job (if they only have 1 job, please exclude them).

```
with seq as (
    SELECT
        *,
        RANK()
        over (partition by person_id, last_title order by
group_start_date, group_end_date desc) as rnk
    FROM seven_apps.people
    WHERE GROUP_END_DATE IS NOT NULL
)
SELECT b.person_id, b.last_title, b.group_start_date, b.group_end_date
FROM seven_apps.people a
INNER JOIN seq b
ON a.PERSON_ID = b.PERSON_ID AND rnk = 2
GROUP BY b.person_id, b.last_title, b.group_start_date, b.group_end_date;
```

What is the average duration in years of employment across everyone's 2nd most recent job? Additionally, how many people have had more than 1 job?

```
with seq as (
    SELECT
        *,
        RANK()
        over (partition by person_id, last_title order by
group_start_date, group_end_date desc) as rnk
    FROM seven_apps.people
    WHERE GROUP_END_DATE IS NOT NULL
)
SELECT COUNT(distinct b.person_id), ROUND(AVG(datediff(b.group_end_date,
b.group_start_date)/365),2) as average_years
FROM seven_apps.people a
INNER JOIN seq b
ON a.PERSON_ID = b.PERSON_ID AND rnk = 2
```

In [51]:
```
question5_sql = '''with seq as (
        SELECT
                *,
                RANK()
                over (partition by person_id, last_title order by group_start_date,
        FROM seven_apps.people
    WHERE GROUP_END_DATE IS NOT NULL
)
SELECT COUNT(distinct b.person_id), ROUND(AVG(datediff(b.group_end_date, b.group_st
FROM seven_apps.people a
INNER JOIN seq b
ON a.PERSON_ID = b.PERSON_ID AND rnk = 2'''

print(pd.read_sql(sql=question5_sql, con=mydb))
```

	COUNT(distinct b.person_id)	average_years
0	203	1.47