Essense test by Marco Wong

The structure of this jupyter notebook is divided into 4 parts:

1. Set up dummy data and utils functions
2. SQL case study
3. SQL questions
4. Python questions

In order to run the set up scripts, you might need to create a new environment in your local:

- conda create -n essense_test python=3.7
- conda activate essense_test
- pip install SQLAlchemy==1.4.23
- pip install pandas==1.3.2
- pip install psycopg2==2.7.7

Part 1 Setting up Postgres DB with dummy data

Part 1 Setting up Postgres DB with dummy data
I have created a Postgres DB with some dummy data for the case study and SQL question in part 2.
I would like to set it up so we can go through the queries with some data.

Normally I wouldn't store any database credentials in the code, but since this will only be shared within the interviewer and myself, so I think it is not neccessary to create a yaml file and overcomplicate the task.

Please ignore this part if you want to go straight into case study

In [38]:

```
import sqlalchemy as s
import pandas as pd
import numpy as np
import random
import re


def create_connection():
    host = 'tai.db.elephantsql.com'
    user = 'xbsstbyy'
    db = 'xbsstbyy'
    pwd = 'mpPWwkb3YjnyKUjYGgunh5C_3PZyjuls'
    engine = s.create_engine(f'postgresql://{user}:{pwd}@{host}/{db}')
    return engine

def query(q):
    conn = create_connection()
    if 'update' not in q and 'create' not in q and 'insert' not in q and 'select' in q:
        return pd.read_sql_query(q, con=conn)
    else:
        conn.execute(q)
        return True

def db_insert(data, target):
    conn = create_connection()
    insert_statement = ''
    cols = ','.join(data.columns)
    for i in range(0, data.shape[0]):
```

```
        values = tuple(list(data.iloc[i, :]))
        values = str(values) if len(values) > 1 else str(values).replace(',', '')
        values = values.replace("'null'", "null").replace('"', "'")
        insert_statement += f'insert into public.{target} ({cols}) values {values};'
    conn.execute(insert_statement)
    conn.dispose()
    return True

#set up tables and data
table_q = '''
drop table if exists public.video_best_practices_data;
drop table if exists public.banner_best_practices_data;
drop table if exists public.creative_testing_tracker_data;
drop table if exists public.results_data;
drop table if exists public.Table_1;

create table public.video_best_practices_data
(
campaign_name varchar(100),
media_plan_id int,
creative_name varchar(100),
video_bp_count int
);

create table public.banner_best_practices_data
(
campaign_name varchar(100),
media_plan_id int,
creative_name varchar(100),
banner_bp_count int
);

create table public.creative_testing_tracker_data
(
media_plan_id int,
media_plan_name varchar(100),
creative_name varchar(100),
pri_passed char(3)
);

create table public.results_data
(
campaign_name varchar(100),
media_plan_id int,
product varchar(100),
reach int,
abs_lift float,
spends int
);

create table public.Table_1
(
campaign_id int,
Channel varchar(50),
Exposed_Count int,
Control_Count int,
Exposed_Percent float,
Control_Percent float,
Sig_Reported char(1),
Lift_Reported float
);
'''

query(table_q)

campaign_name = ['essense_campaign_1', 'essense_campaign_2', 'essense_campaign_3']
```

```
media_plan_id = [1, 2, 3]
media_plan_name = ['m1', 'm2', 'm2']
product = ['google', 'youtube', 'bing']
creative_name = ['creative_1', 'creative_2', 'creative_3']

campaign_name = campaign_name
media_plan_id = media_plan_id
media_plan_name = media_plan_name
product = product
creative_name = creative_name

random.shuffle(campaign_name)
random.shuffle(media_plan_id)
random.shuffle(product)
random.shuffle(creative_name)

result_data = {'campaign_name': campaign_name,
               'media_plan_id': media_plan_id,
               'product': product,
               'reach': [random.randint(0,22) for i in campaign_name],
               'abs_lift': np.random.uniform(low=0.0, high=1.0, size=len(campaign_name)),
               'spends': [random.randint(0,22) for i in campaign_name]
               }
creative_testing_tracker_data = {'media_plan_id': media_plan_id,
                                 'media_plan_name': media_plan_name,
                                 'creative_name': creative_name,
                                 'pri_passed': ['YES' if random.randint(0,22) > 10 else 'N
                                }
banner_best_practices_data = {'campaign_name': campaign_name,
                              'media_plan_id': media_plan_id,
                              'creative_name': creative_name,
                              'banner_bp_count': [random.randint(0,22) for i in media_plar
video_best_practices_data = {'campaign_name': campaign_name,
                             'media_plan_id': media_plan_id,
                             'creative_name': creative_name,
                             'video_bp_count': [random.randint(0,22) for i in media_plan_i
table_1 = {'campaign_id': [1, 2, 3, 4],
           'Channel': ['YouTube', 'Facebook', 'Twitter', 'YouTube'],
           'Exposed_Count': [1000, 800, 700, 2000],
           'Control_Count': [1500, 820, 750, 2000],
           'Exposed_Percent': [0.78, 0.45, 0.51, 0.63],
           'Control_Percent': [0.76, 0.42, 0.51, 0.629],
           'Sig_Reported': ['Y', 'Y', 'N', 'N'],
           'Lift_Reported': [0.02, 0.03, 0, 0.001]
           }

results_data = pd.DataFrame(result_data)
creative_testing_tracker_data = pd.DataFrame(creative_testing_tracker_data)
banner_best_practices_data = pd.DataFrame(banner_best_practices_data)
video_best_practices_data = pd.DataFrame(video_best_practices_data)
table_1 = pd.DataFrame(table_1)

db_insert(results_data, 'results_data')
db_insert(creative_testing_tracker_data, 'creative_testing_tracker_data')
db_insert(banner_best_practices_data, 'banner_best_practices_data')
db_insert(video_best_practices_data, 'video_best_practices_data')
db_insert(table_1, 'Table_1')

# add variation
query(''' insert into creative_testing_tracker_data
values
(1, 'm1', 'creative_4', 'YES'),
(1, 'm1', 'creative_5', 'YES'),
(1, 'm1', 'creative_6', 'NO'),
(2, 'm2', 'creative_7', 'NO'),
(2, 'm2', 'creative_8', 'YES'),
```

```
(2, 'm2', 'creative_9', 'NO');

insert into banner_best_practices_data
values
('essense_campaign_1', 1, 'creative_4', 5),
('essense_campaign_1', 1, 'creative_5', 1),
('essense_campaign_1', 1, 'creative_6', 2),
('essense_campaign_2', 2, 'creative_8', 10),
('essense_campaign_2', 2, 'creative_9', 8);

insert into video_best_practices_data
values
('essense_campaign_1', 1, 'creative_4', 10),
('essense_campaign_1', 1, 'creative_6', 20),
('essense_campaign_2', 2, 'creative_9', 9)''')
```

Out[38]: True

Part 2 SQL Case Study

Relationship between tables: After reading the case study, I think the relationship of the tables should be similar to the following image.

Notes:

1. campaign_name is one-to-many to media_plan_id
2. media_plan_id is one-to-many to creative_name
3. media_plan_id is one-to-one to media_plan_name
4. Not all campaign/media_plan have data in video/banner best practice (need to accomodate with nulls)
5. product is specific to media_plan_id
6. creative_name in creative_testing_trackerdata is fk to creative_name in creative_name in video/banner best practice

Assumptions:

1. There is no duplicated campaign_name, media_plan_id and creative_name combination on video_best_practices_data and banner_best_practices_data tables (unique constraint or pk)
2. There is no duplicated campaign_name, media_plan_id, product combination on results_data
3. There is no duplicated media_plan_id, media_name, creative_name combination in creative_testing_tracker_data media_plan_id column is int instead of str in results_data
4. Assume all combinations of media_plan_id and creative_name in video/banner best practices data exists in creative_testing_tracker_data

Approach: I created 4 common table expressions to make the view to be easier to follow.

1. Perform full outer join video/banner_best_practices_data on campaign_name, media_plan_id and creative_name with case when null to accomodate note 4. Then we will get a table with count of banner/video_best_practices for each capaign_name, media_plan_id and creative_name combination (video_banner_merge)

2. video_banner_merge join to creative_testing_tracker_data on media_plan_id and creative_name, then we would know which media_plan_id and creative_name combination has achieved primary goal

(video_banner_practices_cte)

3. Aggregate video_banner_practices_cte by campaign_name and media_plan_id to get number of creatives, number of best creatives (banner_bp_count >= 8 and video_bp_count >= 9), number of creatives achieved primary goal. (agg_vb_practices)

4. Calculate cost per lifted customer using results_data (cpil)

5. agg_vb_practices left join to cpil on campaign_name and media_plan_id to get the final result

In [39]:

```
vb_q = '''
drop view if exists result_view;
create view result_view as

with video_banner_merge as (
select
case when vd.campaign_name is null then bd.campaign_name else vd.campaign_name end as camp
case when vd.media_plan_id is null then bd.media_plan_id else vd.media_plan_id end as medi
case when vd.creative_name is null then bd.creative_name else vd.creative_name end as crea
case when bd.banner_bp_count is null then 0 else bd.banner_bp_count end as banner_bp_count
case when vd.video_bp_count is null then 0 else vd.video_bp_count end as video_bp_count
from video_best_practices_data vd
full outer join banner_best_practices_data bd
on vd.campaign_name = bd.campaign_name
and vd.media_plan_id = bd.media_plan_id
and vd.creative_name = bd.creative_name
),

video_banner_practices_cte as (
select
vb.campaign_name,
c.media_plan_id,
c.creative_name,
c.pri_passed,
case when vb.banner_bp_count is null then 0 else vb.banner_bp_count end banner_bp_count,
case when vb.video_bp_count is null then 0 else vb.video_bp_count end video_bp_count
from
creative_testing_tracker_data c
left join video_banner_merge vb
on c.media_plan_id = vb.media_plan_id and c.creative_name = vb.creative_name),

agg_vb_practices as
(
select
campaign_name,
media_plan_id,
count(creative_name) creatives,
sum(banner_bp_count) banner_bp_count,
sum(video_bp_count) video_bp_count,
sum(case when banner_bp_count >= 8 and video_bp_count >= 9 then 1 else 0 end) best_practic
sum(case when pri_passed = 'YES' then 1 else 0 end) successful_creatives,
sum(case when pri_passed = 'NO' then 1 else 0 end) fail_creatives
from video_banner_practices_cte
group by campaign_name, media_plan_id),

cpil as (
select
campaign_name,
media_plan_id,
product,
case when reach*abs_lift = 0 then spends else spends/(reach*abs_lift) end cost_per_lifted
from results_data)
```

```
select
c.campaign_name,
c.media_plan_id,
c.product,
c.cost_per_lifted as cpil,
cast(m.best_practice_creatives as float)/cast(m.creatives as float) as creatives_passing_k
cast(m.successful_creatives as float)/cast(creatives as float) as creatives_achieving_goal
from cpil c
left join agg_vb_practices m
on c.media_plan_id = m.media_plan_id and c.campaign_name = m.campaign_name
'''

query(vb_q)

vb = query(''' select campaign_name,
media_plan_id,
product,
cpil,
creatives_passing_best_practice_percentage,
creatives_achieving_goal_percentage
from result_view ''')
print(vb)
```

```
        campaign_name  media_plan_id  product      cpil  \
0  essense_campaign_2              3     bing  8.366990
1  essense_campaign_3              2  youtube  3.706291
2  essense_campaign_1              1   google  6.162566

   creatives_passing_best_practice_percentage  \
0                                        0.00
1                                        0.00
2                                        0.25

   creatives_achieving_goal_percentage
0                                 1.00
1                                 1.00
2                                 0.75
```

Part 3 SQL Question

For this exercise, we are converting the results based on one tailed tests to two tailed tests to allow the youtube data to be consistent with other channels.

We are going to create a new column called "Significance" to replace the original "Sig_Reported" column, where it has the following logic: Significance = 'Y' if "Lifted_Reported" > MDE, 'N' otherwise MDE = 1.645*((Control_Percent*(1-Control_Percent)/Control_Count) + (Control_Percent*(1-Control_Percent)/Exposed_Count))^0.5

The query below will insert the existing data from Table_1 into new_table_1, by removing Sig_Reported column and adding Significance column with 2-tailed test for Channel = 'YouTube'.

Assumptions:

- Channel = "YouTube" is case sensitive
- Assume all other channels have the same significance level as "YouTube"
- Assume all other channels have calculated Sig_Reported correctly and with the same significance level

In [40]:

```
table_1_q = '''
drop table if exists new_table_1;
```

```
select
campaign_id,
Channel,
Exposed_Count,
Control_Count,
Exposed_Percent,
Control_Percent,
Sig_Reported Significance,
Lift_Reported
into new_table_1
from Table_1
where Channel != 'YouTube';

with calculate_mde as (
select
campaign_id,
Channel,
Exposed_Count,
Control_Count,
Exposed_Percent,
Control_Percent,
Sig_Reported,
Lift_Reported,
case when Control_Count > 0 then (Control_Percent*(1-Control_Percent)/Control_Count) else
case when Exposed_Count > 0 then (Control_Percent*(1-Control_Percent)/Exposed_Count) else
from Table_1
where Channel = 'YouTube'
)

insert into new_table_1
(campaign_id, Channel, Exposed_Count, Control_Count,
Exposed_Percent, Control_Percent, Significance, Lift_Reported)
select
campaign_id,
Channel,
Exposed_Count,
Control_Count,
Exposed_Percent,
Control_Percent,
case when Lift_Reported > 1.645*(Control_MDE+Expose_MDE)^0.5 then 'Y' else 'N' end Signifi
Lift_Reported
from calculate_mde
'''
query(table_1_q)

table_1 = query(''' select
campaign_id,
Channel,
Exposed_Count,
Control_Count,
Exposed_Percent,
Control_Percent,
Significance,
Lift_Reported
from new_table_1 ''')

print(table_1)
```

```
   campaign_id   channel  exposed_count  control_count  exposed_percent  \
0            2  Facebook            800            820             0.45
1            3   Twitter            700            750             0.51
2            1   YouTube           1000           1500             0.78
3            4   YouTube           2000           2000             0.63
```

	control_percent	significance	lift_reported
0	0.420	Y	0.030
1	0.510	N	0.000
2	0.760	N	0.020
3	0.629	N	0.001

Additional Comment:

Since this exercise says only YouTube has one-tailed test, so my approach is assuming all other channels have calculated the tests correctly. However, when I tried to calculate using the same formula given, Facebook gave me Significance = "N". Such that MDE is approximately 0.4 and Lifted_Reported = 0.03, so it will be "N". Therefore I would double check the Sig_Reported is consistent between different channel again.

In [41]:

```
# checking facebook mde
fb_con_count = 820
fb_exp_count = 800
fb_con_per = 0.42*(1-0.42)
mde = ((fb_con_per/fb_con_count) + (fb_con_per/fb_exp_count))**0.5
mde = 1.645*mde

fb_lifted_reported = 0.03
print(mde)
```

```
0.04034689627876311
```

Part 4 Python Code

In [42]:

```
import math
import numpy as np
import pandas as pd
import re
import datetime

#Fibonacci Sequence
def fibonacci_sequence(x):
    seq = [0, 1]
    if x < 2:
        return seq[:x]
    else:
        for i in range(0, x-len(seq)):
            seq.append(seq[-1]+seq[-2])
        return seq
x = 10
print(f'1) Answer for fibonacci sequence with x = {x}: ' + str(fibonacci_sequence(x=x)))
print()

# divisible list
def divisible_list(my_list, y):
    div_list = []
    for i in my_list:
        if i % y == 0:
            div_list.append(i)
    return div_list
my_list = [3, 4, 5, 6, 7, 8, 9]
y = 3
answer = str(divisible_list(my_list=my_list, y=y))
print(f'2) Answer for divisible list with input my_list = {my_list} and y = {y}: {answer}')
print()

#find lowest common multiple
def find_lowest_common_multiple(num1, num2):
    larger = num1 if num1 > num2 else num2
```

```
    while True:
        if larger % num1 == 0 and larger % num2 == 0:
            return larger
        larger += 1
#gcd means greatest common divider
def find_lcm(num1, num2):
    return abs(num1 * num2) // math.gcd(num1, num2)

num1 = 9
num2 = 63
loop_answer = find_lowest_common_multiple(num1, num2)
math_answer = find_lcm(num1, num2)

print(f'3a) Answer for LCM using loop with inputs num1 = {num1} and num2 = {num2}: {loop_a
print(f'3b) Answer for LCM using math package with inputs num1 = {num1} and num2 = {num2}:
print()


#compute mean, std and variance
def mean_std_variance(my_list):
    mean = np.mean(my_list)
    std = np.std(my_list)
    var = np.var(my_list)
    return {'mean': mean, 'std': std, 'variance': var}
one_d_array = [1, 5, 3, 1, 2, 4, 6, 7, 1]
mean_std_var_answer = mean_std_variance(one_d_array)
print(f'4) Answer for mean, std and variance using numpy with inputs my_list = {one_d_arra
print()


# bucketing
# since I do not see Table_1 has a column call spends, so I will create a new df to perfo
table_1 = pd.DataFrame({'spends': [100000, 500000, 1000000, 2000000]})
table_1['Group'] = '3'
table_1['Group'] = np.where(table_1['spends'] > 500000, '2', table_1['Group'])
table_1['Group'] = np.where(table_1['spends'] > 1000000, '1', table_1['Group'])
print(f'5) Answer for bucketing data')
print(table_1)
print()


# date formatting function
def format_date(date):
    if type(date) == str:
        date = datetime.datetime.strptime(date, '%m/%d/%Y')
    return date.strftime('%Y%d%m')
date = '08/20/2021'
date_answer = format_date(date)
print(f'6) Answer for formating date from {date} to {date_answer}')
print()


# cleaning dataframe untidy column names
# assume columns have no special characters
untidy_df = pd.DataFrame({'Ema@il': [], 'E$$ssense ': [], 'Pyt[]hon': [], 'cost_per_head
untidy_cols = list(untidy_df.columns)
untidy_df.columns = [''.join(e for e in col if e.isalnum()) for col in untidy_df.columns]
tidy_cols = list(untidy_df.columns)
print(f'7a) Answer for tidy up columns names from {untidy_cols} to {tidy_cols}')
print()

# if we want to remain underscores in pandas to match db schema
untidy_df = pd.DataFrame({'Ema@il': [], 'E$$ssense ': [], 'Pyt[]hon': [], 'cost_per_head
untidy_cols = list(untidy_df.columns)
untidy_df.columns = [re.sub('\W+', '', col) for col in untidy_df.columns]
```

```
tidy_cols = list(untidy_df.columns)
print(f'7b) Answer for tidy up columns names with underscores from {untidy_cols} to {tidy_
```

```
1) Answer for fibonacci sequence with x = 10: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

2) Answer for divisible list with input my_list = [3, 4, 5, 6, 7, 8, 9] and y = 3: [3, 6,
9]

3a) Answer for LCM using loop with inputs num1 = 9 and num2 = 63: 63
3b) Answer for LCM using math package with inputs num1 = 9 and num2 = 63: 63

4) Answer for mean, std and variance using numpy with inputs my_list = [1, 5, 3, 1, 2, 4,
6, 7, 1]: {'mean': 3.3333333333333335, 'std': 2.160246899469287, 'variance': 4.66666666666
6667}

5) Answer for bucketing data
    spends Group
0   100000     3
1   500000     3
2  1000000     2
3  2000000     1

6) Answer for formating date from 08/20/2021 to 20212008

7a) Answer for tidy up columns names from ['Ema@il', 'E$$ssense ', 'Pyt[]hon', 'cost_per_h
ead  '] to ['Email', 'Essense', 'Python', 'costperhead']

7b) Answer for tidy up columns names with underscores from ['Ema@il', 'E$$ssense ', 'Pyt[]
hon', 'cost_per_head  '] to ['Email', 'Essense', 'Python', 'cost_per_head']
```

In [ ]: