Prepared by Abhishek Kumar

https://bit.ly/AKSLinkedIn (https://bit.ly/AKSLinkedIn)

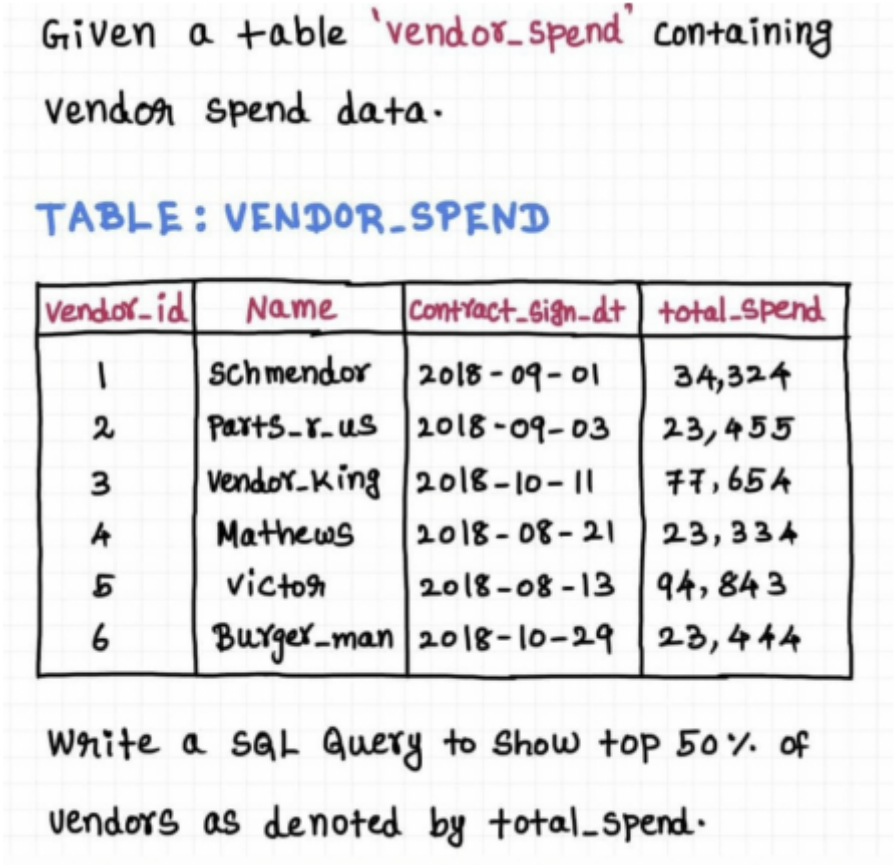
In [1]:
```
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [2]:
```
# Import libraries

import pandas as pd
import numpy as np
import sqlite3 as sql
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
# Creating a database for Vauld

conn = sql.connect('vauld.db')
```

Part 1

Question 1



Data Preparation

In [4]:
```
data = {'vendor_id':[1,2,3,4,5,6],
        'name':['Schmendor','Parts_r_us','Vendor_king','Mathews','Victor','Burger_man'],
        'contract_sign_dt':['2018-9-1','2018-9-3','2018-10-11','2018-8-21','2018-8-13','2018-10-29'],
        'total_spend':[34324,23455,77654,23334,94843,23444]}
vendor_spend = pd.DataFrame(data)
vendor_spend['contract_sign_dt'] = pd.to_datetime(vendor_spend.contract_sign_dt)
```

In [5]:
```
vendor_spend.shape
vendor_spend
```

Out[5]: (6, 4)

Out[5]:

	vendor_id	name	contract_sign_dt	total_spend
0	1	Schmendor	2018-09-01	34324
1	2	Parts_r_us	2018-09-03	23455
2	3	Vendor_king	2018-10-11	77654
3	4	Mathews	2018-08-21	23334
4	5	Victor	2018-08-13	94843
5	6	Burger_man	2018-10-29	23444

Python Implementation

In [6]:
```
vendor_spend[vendor_spend.total_spend>np.median(vendor_spend.total_spend)]
```

Out[6]:

	vendor_id	name	contract_sign_dt	total_spend
0	1	Schmendor	2018-09-01	34324
2	3	Vendor_king	2018-10-11	77654
4	5	Victor	2018-08-13	94843

SQL Implementation

In [7]:
```
# Creating the weather table in vauld database
vendor_spend.to_sql('vendor_spend',conn,if_exists='replace',index=False)
```

In [8]:
```
# Query prepartion
query = '''

select vendor_id,name,contract_sign_dt,total_spend
from (select *, percent_rank() over (order by total_spend) as percntile from vendor_spend) as per
where percntile > 0.5

'''
```

In [9]:
```
# Passing the query to vauld db for execution

result = pd.read_sql(query,conn)
result
```

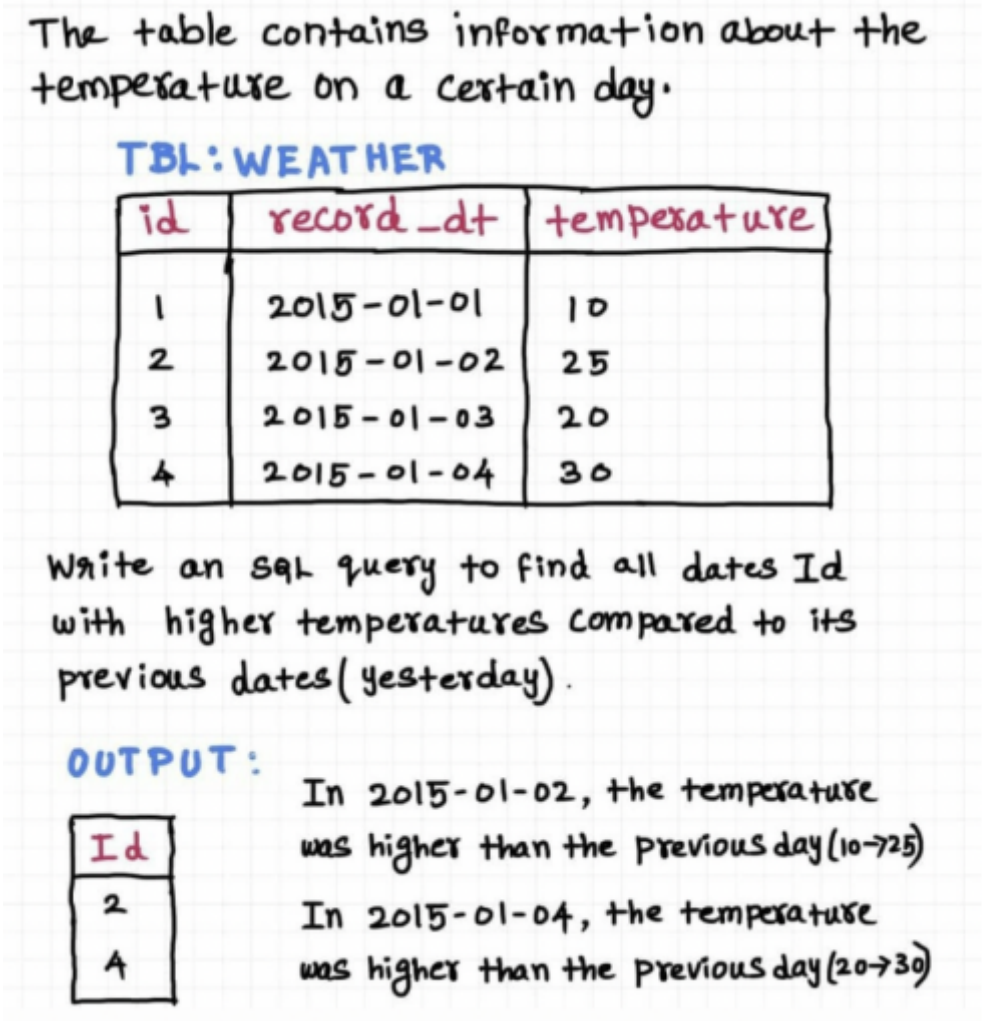
Out[9]:

	vendor_id	name	contract_sign_dt	total_spend
0	1	Schmendor	2018-09-01 00:00:00	34324
1	3	Vendor_king	2018-10-11 00:00:00	77654
2	5	Victor	2018-08-13 00:00:00	94843

Result - Vendor 1, 3 and 5 are the top 50% spender

In [ ]:
```

```

Question 2



Data Preparation

In [10]:
```
data = {'id':[1,2,3,4,5,6],
        'record_dt':['2015-1-1','2015-1-2','2015-1-3','2015-1-4','2015-1-5','2018-1-6'],
        'temperature':[10,25,20,30,28,27]}
weather = pd.DataFrame(data)
weather['record_dt'] = pd.to_datetime(weather.record_dt)
```

In [11]:
```
weather.shape
weather
```

Out[11]: (6, 3)

Out[11]:

	id	record_dt	temperature
0	1	2015-01-01	10
1	2	2015-01-02	25
2	3	2015-01-03	20
3	4	2015-01-04	30
4	5	2015-01-05	28
5	6	2018-01-06	27

Python Implementation

In [12]:
```
# Python Implementation
weather_py = weather.copy()
weather_py['prev_temp'] = weather_py.sort_values('record_dt').temperature.shift(1)
weather_py[weather_py.temperature > weather_py.prev_temp]['id'].to_frame()
```

Out[12]:

	id
1	2
3	4

SQL Implementation

In [13]:
```
# Creating the weather table in vauld database
weather.to_sql('weather',conn,if_exists='replace',index=False)
```

In [14]:
```
# Query prepartion
query = '''

select id
from (select *, lag(temperature) over (order by record_dt) as prev_temp from weather) as lag
where temperature > prev_temp


'''
```

In [15]:
```
# Passing the query to vauld db for execution

result = pd.read_sql(query,conn)
result
```

Out[15]:

	id
0	2
1	4

Result - The days with ID value 2 and 4 have higher temperatures than their previous days respectively.

Part 2

Question

Imagine there are 10 marketing campaigns by influencers and 10 paid ads campaigns running live. You have 2 main source tables -> **sessions** and **conversions** , you need to attribute each marketing campaign (by using UTM parameters) to 1) how many users it helped convert and 2) how many touch points it takes a user to get converted

Feel free to use any attribution modelling logic as possible and explain the logic, the code and the final output table you will arrive using these base raw tables to explain the same for the marketing team.

Bonus if you can also add ad spend data & show ROI at the end, but not required. Just the logic/explanation will also do.

Sessions

SESSION_ID	CUSTOMER_ID	STARTED_AT	ENDED_AT	UTM_SOURCE	UTM_MEDIUM	UTM_CAMPAIGN
1	2956	2020-02-01 12:55:16	2020-02-01 12:55:47	facebook_ads	paid_social	xyz
2	4	2020-02-02 13:06:47	2020-02-02 13:18:56	facebook_ads	paid_social	freeshipping
3	1170	2020-02-03 12:15:00	2020-02-03 12:15:19	youtube1	social_video	inf1

Conversions

CUSTOMER_ID	CONVERTED_AT
1170	2020-02-03 14:20:08
2014	2020-02-04 4:30:21
2265	2020-02-04 9:43:35

Data Preparation

In [16]:
```
1 sessions = pd.read_excel('./one.xlsx', sheet_name='sessions')
2 sessions['Started_At'] = pd.to_datetime(sessions.Started_At)
3 sessions['Ended_At'] = pd.to_datetime(sessions.Ended_At)
4 sessions
```

Out[16]:

	Session_Id	Customer_Id	Started_At	Ended_At	UTM_Source	UTM_Medium	UTM_Campaign
0	1	2956	2020-02-01 12:55:16	2020-02-01 12:55:47	facebook_ads	paid_social	xyz
1	2	4	2020-02-02 13:06:47	2020-02-02 13:18:56	facebook_ads	paid_social	freeshipping
2	3	1170	2020-02-03 12:15:00	2020-02-03 12:15:19	insta	social_video	freeshipping
3	4	1170	2020-02-03 12:21:00	2020-02-03 12:25:19	youtube	social_video	inf1
4	5	2014	2020-02-04 04:20:21	2020-02-04 04:36:21	facebook_ads	paid_social	freeshipping
5	6	1170	2020-02-05 12:13:00	2020-02-05 12:15:54	youtube	social_video	inf1

In [17]:
```
conversions = pd.read_excel('./one.xlsx', sheet_name='conversions')
conversions['converted_at'] = pd.to_datetime(conversions.converted_at)
conversions['customer_id'] = conversions.customer_id.astype('int')
conversions
```

Out[17]:

	customer_id	converted_at
0	2014	2020-02-04 04:30:21
1	2265	2020-02-04 09:43:35
2	1170	2020-02-05 12:15:19

In [18]:
```
# Creating the sessions and conversions table in vauld database
sessions.to_sql('sessions',conn,if_exists='replace',index=False)
conversions.to_sql('conversions',conn,if_exists='replace',index=False)
```

In [19]:
```
# user defined function to execute query
def run(query, conn=conn):
    # Passing the query to vauld db for execution
    result = pd.read_sql(query,conn)
    return result
```

SQL Implementation

In [20]:
```
# Query prepartion : Data overview
query = '''

select s.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
from sessions s left join conversions c
on s.customer_id = c.customer_id

'''
```

In [21]:
```
1 run(query)
```

Out[21]:

	Session_Id	Customer_Id	Started_At	Ended_At	UTM_Source	UTM_Medium	UTM_Campaign	flag
0	1	2956	2020-02-01 12:55:16	2020-02-01 12:55:47	facebook_ads	paid_social	xyz	0
1	2	4	2020-02-02 13:06:47	2020-02-02 13:18:56	facebook_ads	paid_social	freeshipping	0
2	3	1170	2020-02-03 12:15:00	2020-02-03 12:15:19	insta	social_video	freeshipping	0
3	4	1170	2020-02-03 12:21:00	2020-02-03 12:25:19	youtube	social_video	inf1	0
4	5	2014	2020-02-04 04:20:21	2020-02-04 04:36:21	facebook_ads	paid_social	freeshipping	1
5	6	1170	2020-02-05 12:13:00	2020-02-05 12:15:54	youtube	social_video	inf1	1

Question 1 - how many users it helped convert

In [22]:
```
1  # Query prepartion : Conversion Rate by Campaign
2
3  query = '''
4
5  select UTM_Campaign, count(distinct Customer_Id) as targeted_customer_count, sum(flag) as conversion_count,
6  sum(flag)*100 / count(distinct Customer_Id) as conversion_pct
7  from (
8  select s.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
9  from sessions s left join conversions c
10 on s.customer_id = c.customer_id )
11 group by UTM_Campaign
12
13 '''
```

In [23]:
```
1 run(query)
```

Out[23]:

	UTM_Campaign	targeted_customer_count	conversion_count	conversion_pct
0	freeshipping	3	1	33
1	inf1	1	1	100
2	xyz	1	0	0

In [24]:
```
1  # Query prepartion : Conversion Rate by Medium
2
3  query = '''
4
5  select UTM_Medium, count(distinct Customer_Id) as targeted_customer_count, sum(flag) as conversion_count,
6  sum(flag)*100 / count(distinct Customer_Id) as conversion_pct
7  from (
8  select s.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
9  from sessions s left join conversions c
10 on s.customer_id = c.customer_id )
11 group by UTM_Medium
12
13 '''
```

In [25]:
```
1 run(query)
```

Out[25]:

	UTM_Medium	targeted_customer_count	conversion_count	conversion_pct
0	paid_social	3	1	33
1	social_video	1	1	100

In [26]:
```
# Query prepartion : Conversion Rate by Source

query = '''

select UTM_Source, count(distinct Customer_Id) as targeted_customer_count, sum(flag) as conversion_count,
sum(flag)*100 / count(distinct Customer_Id) as conversion_pct
from (
select s.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
from sessions s left join conversions c
on s.customer_id = c.customer_id )
group by UTM_Source

'''
```

In [27]:
```
run(query)
```

Out[27]:

	UTM_Source	targeted_customer_count	conversion_count	conversion_pct
0	facebook_ads	3	1	33
1	insta	1	0	0
2	youtube	1	1	100

In [28]:
```
# Query prepartion : Conversion Rate by Campaign, Medium, Source

query = '''

select UTM_Campaign, UTM_Medium, UTM_Source, count(distinct Customer_Id) as targeted_customer_count,
sum(flag) as conversion_count, sum(flag)*100 / count(distinct Customer_Id) as conversion_pct
from (
select s.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
from sessions s left join conversions c
on s.customer_id = c.customer_id )
group by UTM_Campaign, UTM_Medium, UTM_Source

'''
```

In [29]:
```
1 run(query)
```

Out[29]:

	UTM_Campaign	UTM_Medium	UTM_Source	targeted_customer_count	conversion_count	conversion_pct
0	freeshipping	paid_social	facebook_ads	2	1	50
1	freeshipping	social_video	insta	1	0	0
2	inf1	social_video	youtube	1	1	100
3	xyz	paid_social	facebook_ads	1	0	0

- **Result - Based on the Last Interaction Attribution Model, the last touchpoint is provided the conversion credit**
- **The above result shows the conversion percentage for different sources and mediums for each campaign**
- **Youtube through Social video is 100% performant**

In [ ]:
```
1
```

Question 2 - how many touch points it takes a user to get converted

In [30]:
```
1  # Query prepartion : Touch points for each customer
2
3  query = '''
4
5  select customer_id, count(*) as count_touch_point
6  from (
7  select c.* , case when converted_at between Started_At and Ended_At then 1 else 0 end as flag
8  from conversions c left join sessions s
9  on s.customer_id = c.customer_id )
10 group by customer_id
11
12 '''
```

In [31]:
```
1 run(query)
```

Out[31]:

	customer_id	count_touch_point
0	1170	3
1	2014	1
2	2265	1

Result - The customers listed with their touch-points till conversion

In [ ]:
```
1
```

Part 3

A crypto company is working on a new feature for adding a binance smart chain (bep-20) along with existing networks that it supports like BTC, ERC-20 for all users. We will allow all deposits and withdrawals for many new tokens using this new bep-20 network.

Given engineering constraints, we can't AB test it before release.

How would you analyze the impact and effectiveness of this new feature 1 week post the release? What changes will you make for future releases based on your learnings?

Note: This question is vague intentionally, please use your own imagination, assumptions and build your own metrics, KPIs here we only want to understand your thought process and product approach here

Expected Data Model

1. Customer-Transaction Relationship
    - customer_id, tran_id
2. Transactions
    - tran_id, token, tran_amt, tran_type, network

- Consider transactions only after the launch

Descriptive stats to find performance of the new network

At the end of Week 1

1. %ge of transactions (in terms of frequency and Amount) across different network
2. %ge of bep-20 transactions (in terms of frequency and Amount) across different transaction_type
3. %ge of customers using bep-20 out of customers who did a transaction
4. %ge of Re-activated customers for this week comparing to previous 2-3 weeks
5. %ge increase in Total transaction (frequency and volume) for this week comparing to previous 2-3 weeks

6. %ge of customers using bep-20 out of customers who did a transaction. and this can be done across GENDER, AGE, GEOGRAPHY, Device-OS, etc

Daily based

1. Plot Daily transactions (in terms of Frequency and Total Volume) in a line chart for different networks
2. Analyze any impact (decrease/increase) on daily churn rates for this week compared to previous 2-3 weeks

What changes will you make for future releases based on your learnings?

- If there is no significant increase in TOTAL VOLUMES/Transacting CUSTOMERs, then a customer survey or general market data can be analysed before coming up with a feature., given the constraints in the question.

In [ ]:
```
1
```