Link to view only copy of this notebook

https://colab.research.google.com/drive/1ZqCTqfp9L3QsVMALfRoDJ3DLAt82cX3s?usp=sharing

Q: Physics Manipulations

> **LIVE | EASY**

Given a table with 2 rows as shown below, containing velocities of 10 different aircrafts that took off from INS Vikramaditya, plot the accelerations of each aircraft as a funtion of time using PANDAS (Note that readings for an aricraft come at 1 sec intervals)

Hint: Note that acceleration is defined as "rate-of-change" of velocity. When the time intervals of readings are constants, it can be computed as the difference of current velocity and previous velocity

Data would eventually have 1000s of rows and would be avaialble as a CSV file

Aircraft Code	Velocity
Alpha	12
Alpha	18
Charlie	15
Alpha	21
Alpha	23
Charlie	24

Saved successfully!

```
import pandas as pd
codes = ['Alpha', 'Alpha', 'Charlie', 'Alpha', 'Alpha', 'Charlie', 'Delta', 'Delta'
velocity = [12, 18, 15, 21, 23, 24, 10, 13, 22, 27, 18, 30]

df = pd.DataFrame()
df['codes'] = codes
df['velocity'] = velocity
df.head()
```

	codes	velocity
0	Alpha	12
1	Alpha	18
2	Charlie	15
3	Alpha	21
4	Alpha	23

```
df['prev_vel'] = df.groupby('codes')['velocity'].shift(1)
df.head()
```

	codes	velocity	prev_vel
0	Alpha	12	NaN
1	Alpha	18	12.0
2	Charlie	15	NaN
3	Alpha	21	18.0
4	Alpha	23	21.0

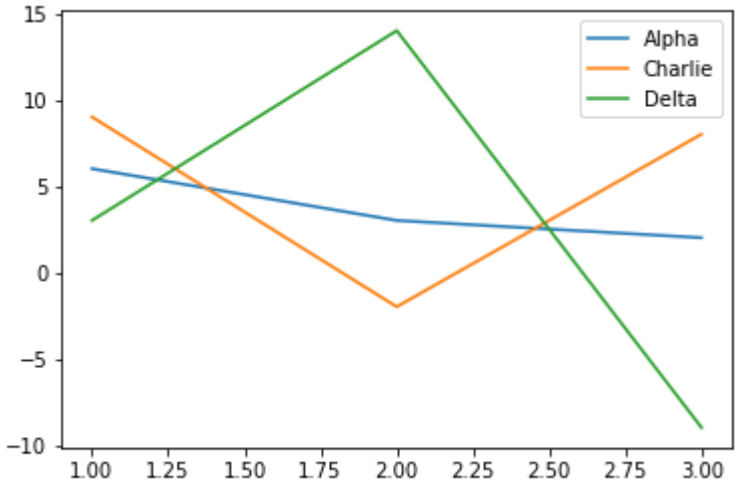
```
df['acc'] = df['velocity'] - df['prev_vel']
df.head()
```

	codes	velocity	prev_vel	acc
0	Alpha	12	NaN	NaN
1	Alpha	18	12.0	6.0
2	Charlie	15	NaN	NaN
3	Alpha	21	18.0	3.0
4	Alpha	23	21.0	2.0

```
import matplotlib.pyplot as plt
                                ):
                                ue).plot(label=i);
plt.legend()
```

Saved successfully!

```
<matplotlib.legend.Legend at 0x7f6621619290>
```

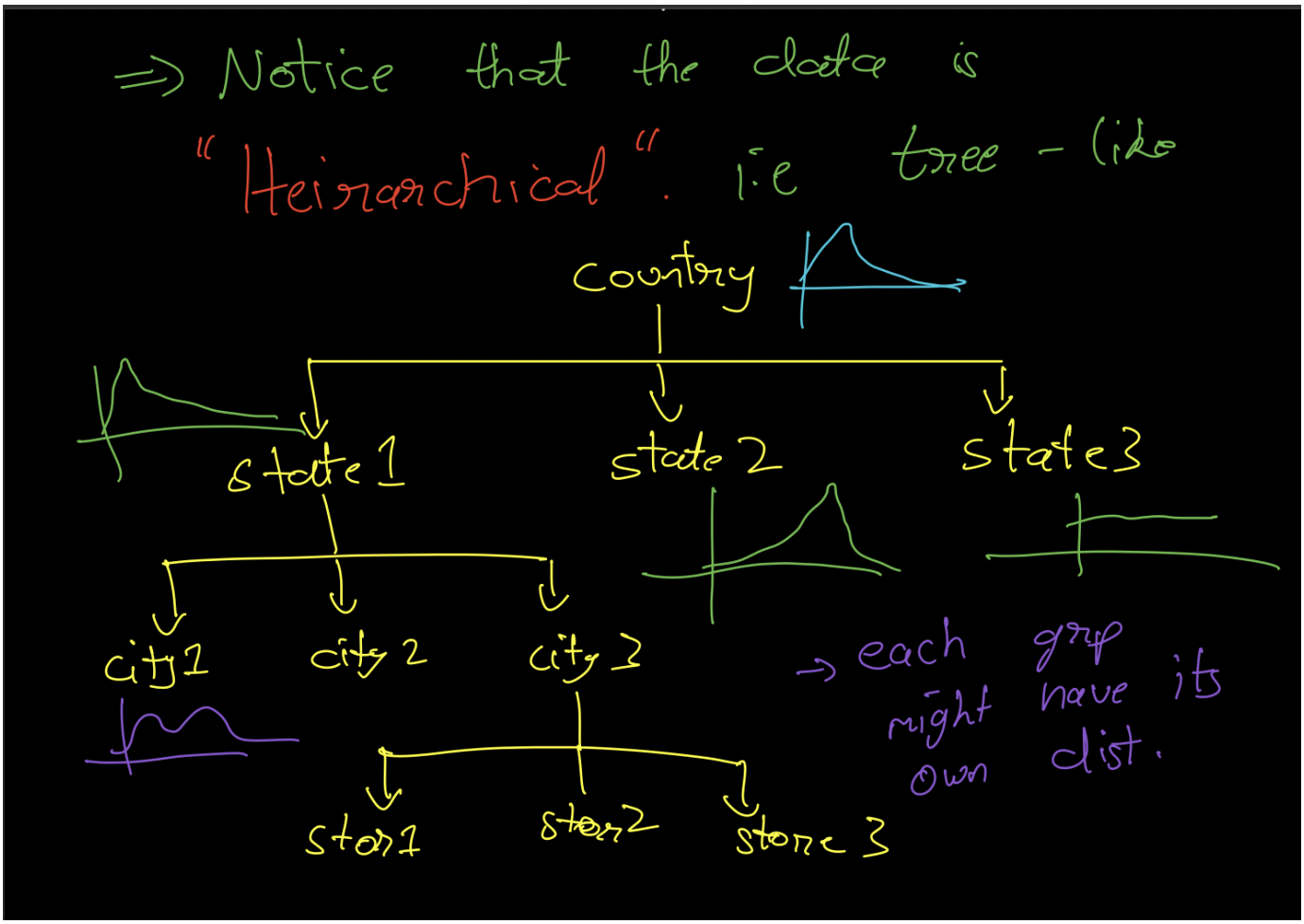


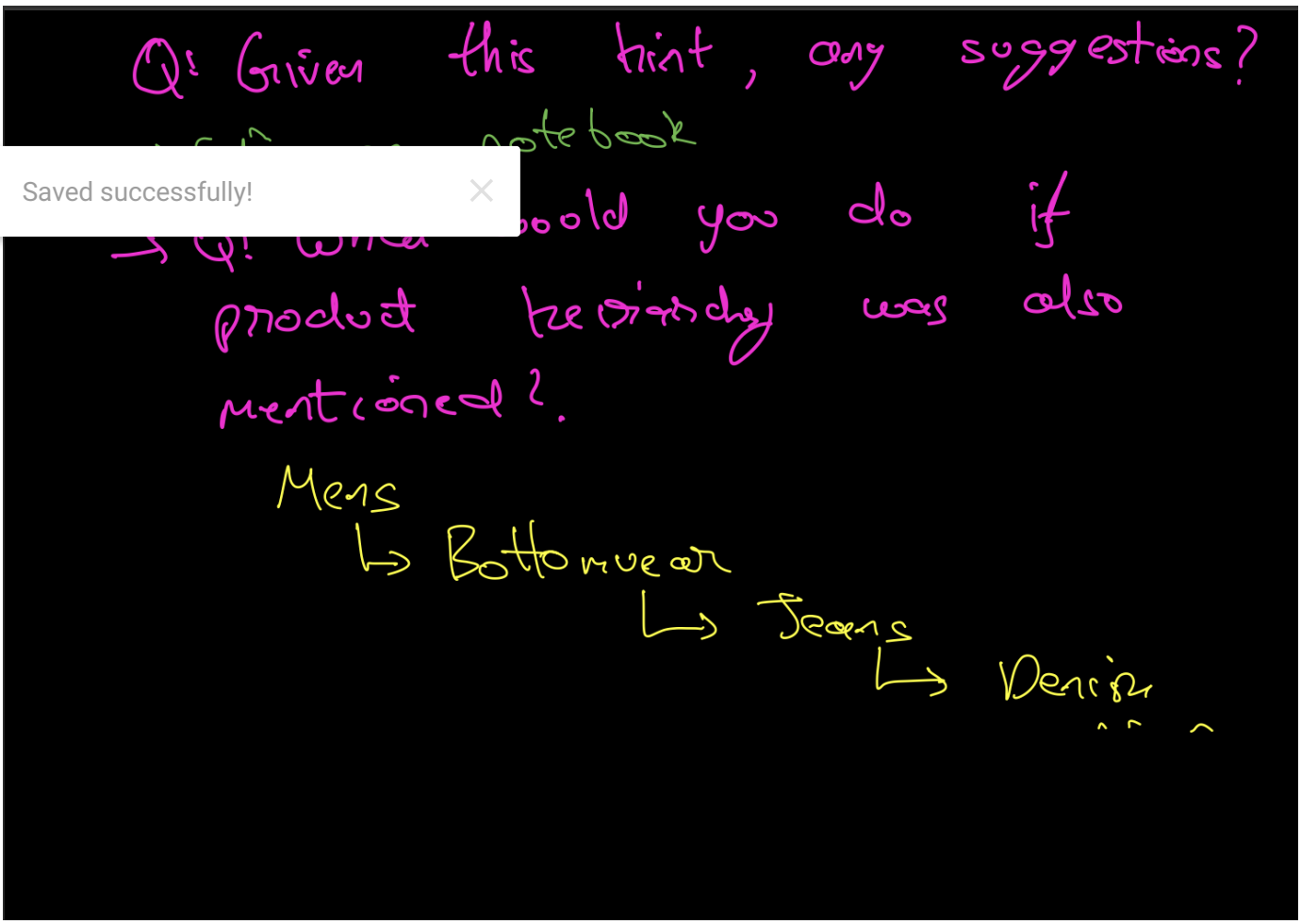
Q: Missing data on Heirarchical retail data

You are given the sales data for an international clothing brand. The stratergy team will use this data to make expansion plans. However the data at store level contains some missing data. Analyse the data and suggest the best way to impute all missing values.

| Country | State | City | Store | Sales |

Saved successfully!


⇒ Notice that the data is
"Heirarchical". i.e tree - like
Country
state 1
state 2
state 3
city 1
city 2
city 3
→ each grp might have its own dist.
store 1
store 2
store 3


Q: Given this hint, any suggestions?
notebook
Saved successfully!
should you do if
product hierarchy was also
mentioned?
Mens
Bottomwear
Jeans
Denim

```
!gdown 1hEqPWpGyfrwlPhF6FlT515IFcTWa3iO2
```

```
Downloading...
From: https://drive.google.com/uc?id=1hEqPWpGyfrwlPhF6FlT515IFcTWa3iO2
To: /content/imputaion_pandas.csv
100% 4.91M/4.91M [00:00<00:00, 242MB/s]
```

```
sales = pd.read_csv('/content/imputaion_pandas.csv')
sales.head()
```

	country	state	city	store_id	sales
0	Afghanistan	Badakhshan	Wākhān	STR_1	NaN
1	Afghanistan	Badakhshan	Wākhān	STR_2	13.0
2	Afghanistan	Baghlan	Baghlān	STR_1	NaN
3	Afghanistan	Balkh	Dowlatābād	STR_1	110.0
4	Afghanistan	Bamyan	Panjāb	STR_1	100.0

```
sales.sales.isna().sum() / len(sales)
```

```
0.27767985899695563
```

Q: Is mean median a good choice?

```
sales.sales.mean()
```

```
74.26497338065661
```

Saved successfully!

```
_ = sales.country.unique()[:10]
sales.loc[sales.country.isin(_)].groupby('country')['sales'].mean().plot(k
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ca7a6b050>
```

```
sales['city_avg'] = sales.groupby('city')['sales'].transform('mean')
sales['city_avg'].isna().sum() / len(sales)
```

```
0.12213142480727804
```

```
sales['state_avg'] = sales.groupby('state')['sales'].transform('mean')
sales['state_avg'].isna().sum() / len(sales)
```

```
0.05195036407983051
```

```
sales['country_avg'] = sales.groupby('country')['sales'].transform('mean')
sales['country_avg'].isna().sum() / len(sales)
```

```
7.121365877975396e-05
```

```
sales['global_avg'] = sales['sales'].mean()
```

```
sales['sales'] = sales['sales'].fillna(sales['city_avg']).fillna(sales['state_avg']
sales['sales'].isna().sum()
```

```
0
```

- **What would you do if product heirarchy was also mentioned**
- **What would you do is certain group had too few values, any concerns ?** [Advanced]

Saved successfully!

ate at Disney.

LIVE | MEDIUM

Imagine you work as a business analyst at an OTT company known as Disney + Hotstar. Your manager has asked you to report the month-on-month decay rate of subscribers for 6 months after joining.

The data is provided in the following 2 tables. (Head shown below)

The desired output is requested as follows

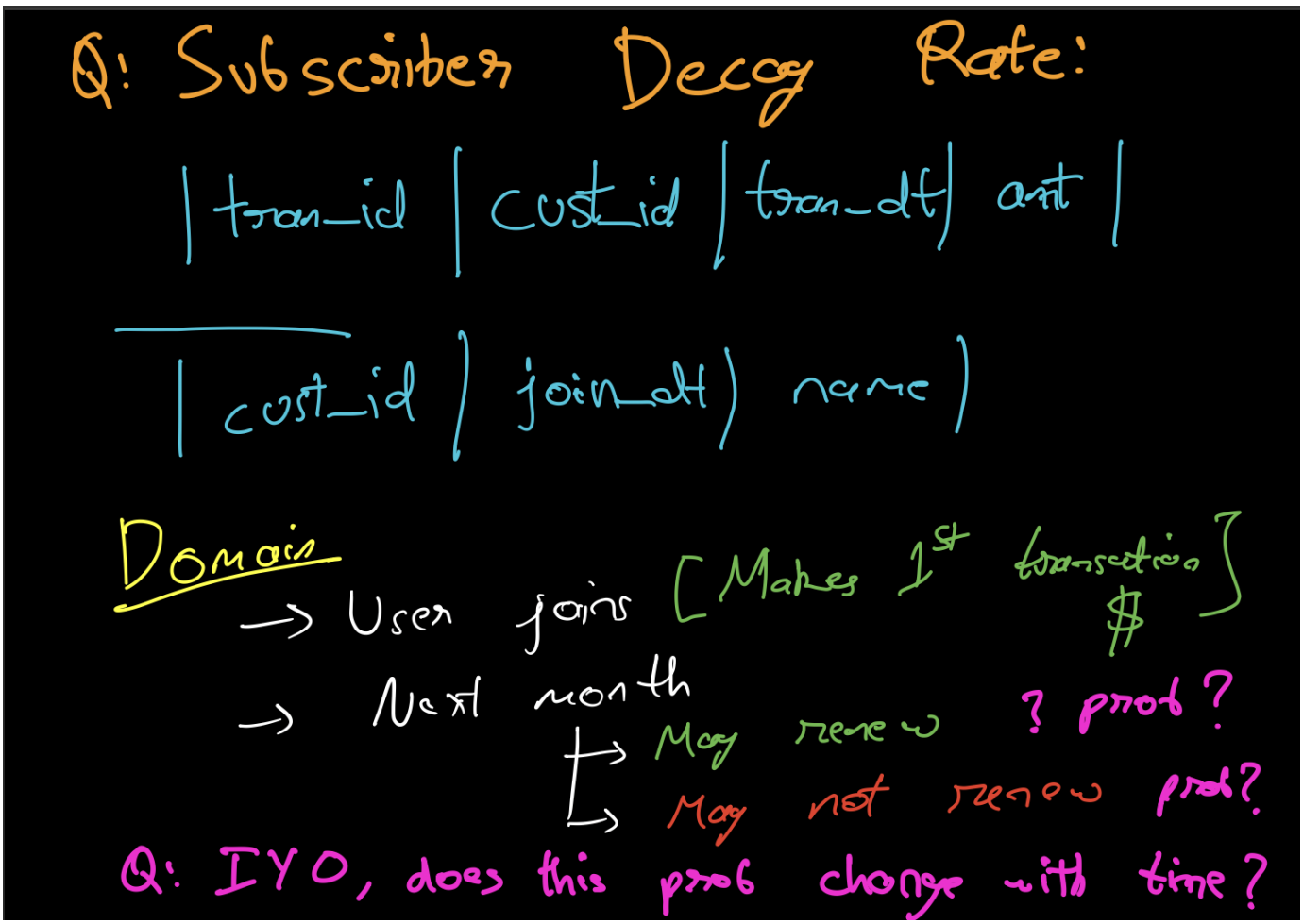
duration (in months)	% users that stayed
0	-
1	-
...	...

Instructions:

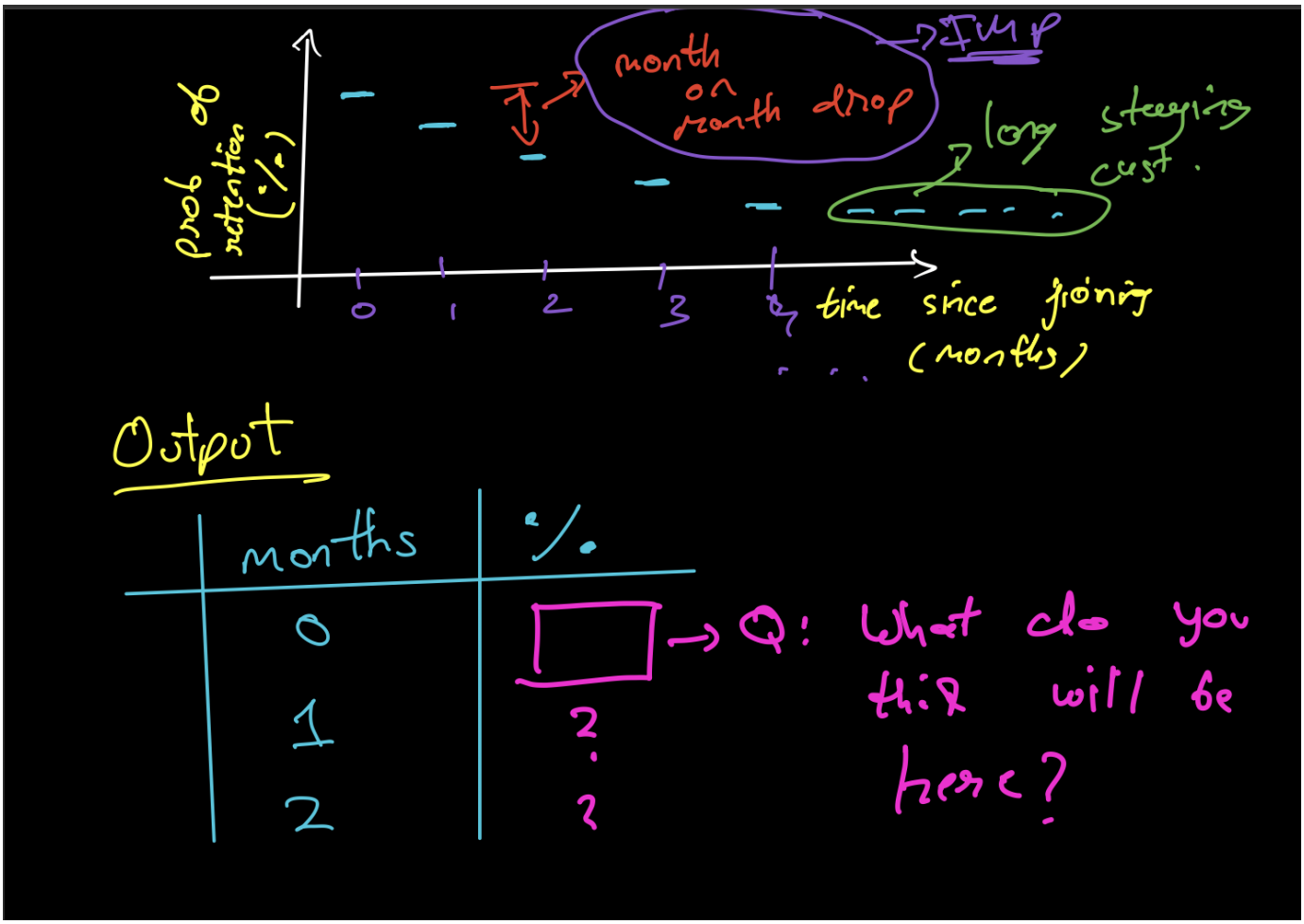
- Please feel free to ask any further questions about the data / problem.
- Please feel free to make any assumptions but inform me whatever you assume.

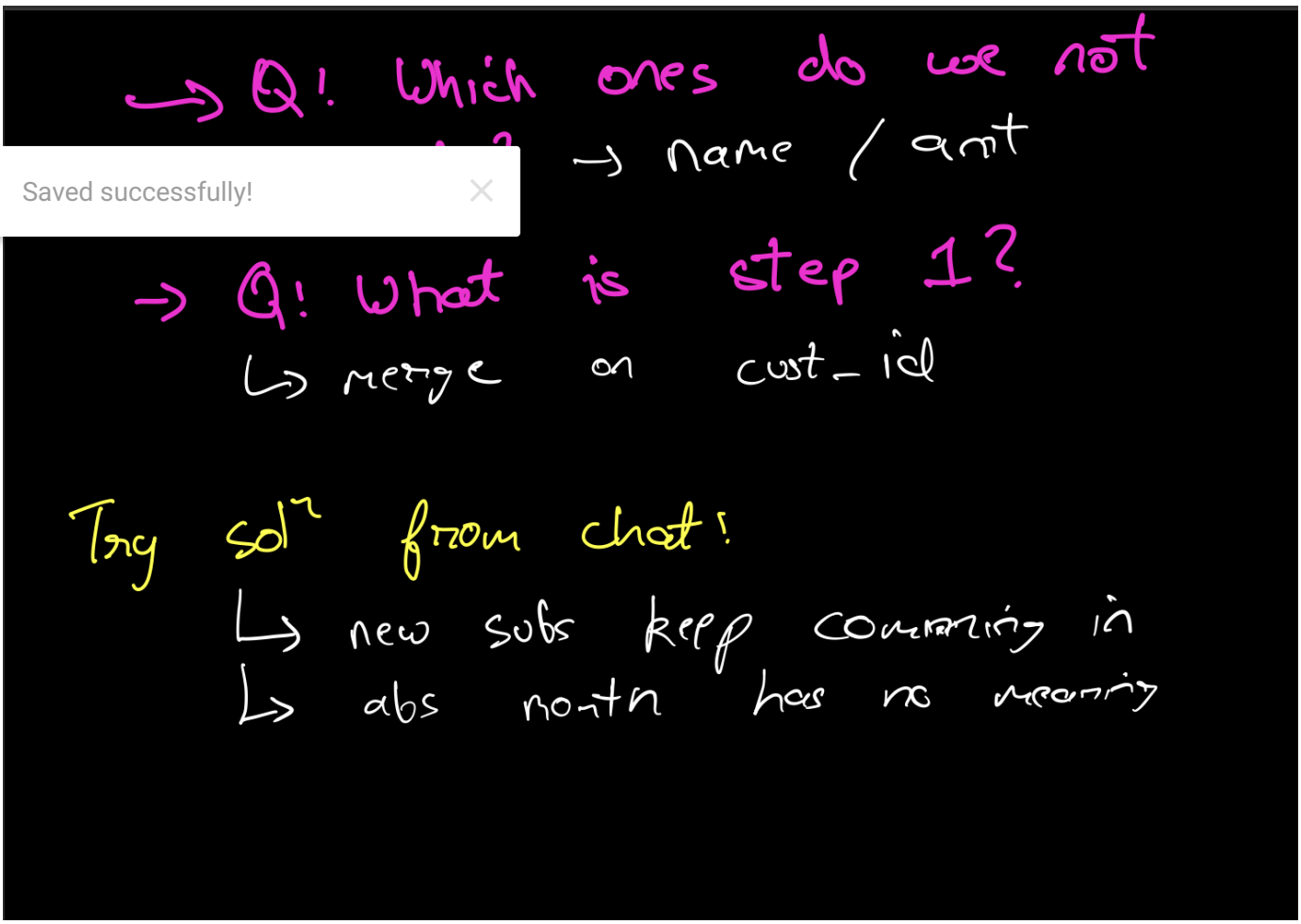
For this round, you first need to explain your thaught process and then process with the code.

Note that this question could just be telephonic and be asked without hands-on-coding.



Saved successfully!


prob of retention (%)
month on month drop
IMP
long steering cust.
0
1
2
3
time since joining (months)
Output
months
%
0
1
2
Q: What do you thik will be here?


Q: Which ones do we not
→ name / amt
Saved successfully!
Q: What is step 1?
↳ merge on cust_id
Try sol^n from chat!
↳ new subs keep coming in
↳ abs month has no meaning

```
!gdown 13JVFS9ex-UNJmegwCNuRzFmSpxxPc5yC
!gdown 1C6RUxSAnoBS9II1o_Xwvlt56n8aMKpHd
```

```
Downloading...
From: https://drive.google.com/uc?id=13JVFS9ex-UNJmegwCNuRzFmSpxxPc5yC
To: /content/transaction_subscription.csv
100% 2.00M/2.00M [00:00<00:00, 168MB/s]
Downloading...
From: https://drive.google.com/uc?id=1C6RUxSAnoBS9II1o_Xwvlt56n8aMKpHd
To: /content/customer_subscription.csv
100% 214k/214k [00:00<00:00, 87.9MB/s]
```

```
import warnings
import numpy as np
import pandas as pd

warnings.filterwarnings('ignore')

trans = pd.read_csv('/content/transaction_subscription.csv')
cust = pd.read_csv('/content/customer_subscription.csv')
```

```
trans.head()
```

	trans_id	customer_id	trans_date	trans_amount
0	96905067-0b45-4339-b962-2f53417114e8	CS4096	2011-05-16	68
1	b0742c63-991a-40d0-aa8f-185eb76f9960	CS4410	2011-05-16	105
2	969bda98-d42f-49eb-8037-9bd793e902dc	CS4053	2011-05-16	58
3	e7acb9f9-6c3a-40c4-a898-769c59eba03f	CS3586	2011-05-16	84
4	1f4a41d6f	CS4165	2011-05-16	97

Saved successfully!

```
cust.head()
```

	customer_id	join_date	name
0	CS4096	2011-05-16	1DFO3W8O50NO
1	CS4410	2011-05-16	L89IRL1DCDL5
2	CS4053	2011-05-16	PMWVNFMF381X
3	CS3586	2011-05-16	HHB8S7N4YNSV
4	CS4165	2011-05-16	D5ZS1S1J1C6J

```
df = trans.merge(cust, on='customer_id', how='inner')
df.head()
```

	trans_id	customer_id	trans_date	trans_amount	join_date	nam
0	96905067-0b45-4339-b962-2f53417114e8	CS4096	2011-05-16	68	2011-05-16	1DFO3W8O50N
1	b0742c63-991a-40d0-aa8f-185eb76f9960	CS4410	2011-05-16	105	2011-05-16	L89IRL1DCDL
2	8f21cac1-5f39-42f3-9fe0-d2669ee6c70d	CS4410	2011-06-16	105	2011-05-16	L89IRL1DCDL
3	969bda98-d42f-49eb-8037-9bd793e902dc	CS4053	2011-05-16	58	2011-05-16	PMWVNFMF381
4	855f8d07-054c-42a4-9ecd-	CS4053	2011-06-16	58	2011-05-16	PMWVNFMF381

```
# DATE FIX - Q: What should we do to the date variable
df['trans_date'] = pd.to_datetime(df['trans_date'])
df['join_date'] = pd.to_datetime(df['join_date'])


# Q: How to get month and year from date columns?
df['trans_year'] = df['trans_date'].dt.year
df['join_year'] = df['join_date'].dt.year
                                  e'].dt.month
                                  ].dt.month


# Q: How can I find the difference of join_month and trans_month?

df['trans_year'] = df['trans_year'] - 2011
df['trans_month'] = df['trans_month'] + df['trans_year'] * 12
df['join_year'] = df['join_year'] - 2011
df['join_month'] = df['join_month'] + df['join_year'] * 12



df['base'] = df.groupby(['join_month'])['customer_id'].transform('nunique')
df['retention'] = df.groupby(['join_month', 'trans_month'])['trans_id'].transform('
df['perc_retention'] = df['retention']/df['base']
df['age'] = df['trans_month'] - df['join_month']
_ = df.loc[df.age<6].groupby(['age'])['perc_retention'].mean()
_
```

Saved successfully!

```
    age
    0    1.000000
```

```
1    0.801618
2    0.660542
3    0.534393
4    0.448958
5    0.375400
Name: perc_retention, dtype: float64
```

```
def decay(m):
  m['trans_date'] = pd.to_datetime(m['trans_date'])
  m['join_date'] = pd.to_datetime(m['join_date'])
  m['trans_year'] = m['trans_date'].dt.year
  m['join_year'] = m['join_date'].dt.year
  m['trans_month'] = m['trans_date'].dt.month
  m['join_month'] = m['join_date'].dt.month
  m['trans_year'] = m['trans_year'] - 2011
  m['trans_month'] = m['trans_month'] + m['trans_year'] * 12
  m['join_year'] = m['join_year'] - 2011
  m['join_month'] = m['join_month'] + m['join_year'] * 12
  m['base'] = m.groupby(['join_month'])['customer_id'].transform('nunique')
  m['retention'] = m.groupby(['join_month', 'trans_month'])['trans_id'].transform('
  m['perc_retention'] = m['retention']/m['base']
  m['age'] = m['trans_month'] - m['join_month']
  _ = m.loc[m.age<6].groupby(['age'])['perc_retention'].mean()
  return _

decay(df)
```

```
age
0    1.000000
1    0.801618
2    0.660542
3    0.534393
```

Saved successfully!

```
Name: perc_retention, dtype: float64
```

Colab paid products - Cancel contracts here

Saved successfully!