This project aims to demonstrate usage of IHC attribution model API from Haensel AMS GmbH and analyse the data returned by the API and visualize the results.

More information about the IHC attribution model and API can be found here:

https://ihc-attribution.com/
https://ihc-attribution.com/ihc-data-driven-attribution-model/

https://ihc-attribution.com/marketing-attribution-api/

1. Convert received dictionaries from API to dataframe as df_attribution_results: I popped out the dictionaries out of list sent by API which also included statusCode and PartialFailureErrors and converted it to a dataframe.

	conversion_id	session_id	initializer	holder	closer	ihc
0	1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-06-06_0001__aaf3b324-758d-d819-a2ee-4e51e...	0.0000	0.0000	0.0000	0.0000
1	1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-07-13_0001__aaf3b324-758d-d819-a2ee-4e51e...	0.8221	0.7667	0.0000	0.5829
2	1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-07-19_0001__2594bffe-76af-4b40-b0f7-12a8f...	0.1779	0.2333	1.0000	0.4171
3	52c8c88f-cf76-4b3c-bb02-d06b6215db29	2021-05-21_0001__7084ac0f-3021-e583-2432-a8a53...	0.0000	0.0000	0.0000	0.0000
4	52c8c88f-cf76-4b3c-bb02-d06b6215db29	2021-05-27_0001__7084ac0f-3021-e583-2432-a8a53...	0.0000	0.0000	0.0000	0.0000
5	52c8c88f-cf76-4b3c-bb02-d06b6215db29	2021-05-28_0001__7084ac0f-3021-e583-2432-a8a53...	0.0000	0.0000	0.0000	0.0000
6	52c8c88f-cf76-4b3c-bb02-d06b6215db29	2021-06-20_0001__7084ac0f-3021-e583-2432-a8a53...	1.0000	1.0000	1.0000	1.0000
7	a3f2b3d9-c8fa-4824-822d-5635a8e55141	2021-06-20_0001__138e910a-ecc5-3b5f-6b0e-df114...	1.0000	0.0000	0.1322	0.3699
8	a3f2b3d9-c8fa-4824-822d-5635a8e55141	2021-06-21_0001__138e910a-ecc5-3b5f-6b0e-df114...	0.0000	1.0000	0.8678	0.6301
9	b8ffb515-51db-4b56-8cff-e67d36d7f702	2021-07-16_0003__947aa504-f543-9224-3453-b55fe...	1.0000	0.0947	0.0000	0.3730
10	b8ffb515-51db-4b56-8cff-e67d36d7f702	2021-07-16_0004__947aa504-f543-9224-3453-b55fe...	0.0000	0.3787	0.0000	0.1519
11	b8ffb515-51db-4b56-8cff-e67d36d7f702	2021-07-17_0001__e3f3a69e-9bd0-144a-52aa-6d7c4...	0.0000	0.0000	0.0000	0.0000
12	b8ffb515-51db-4b56-8cff-e67d36d7f702	2021-07-17_0002__e3f3a69e-9bd0-144a-52aa-6d7c4...	0.0000	0.4213	0.0000	0.1689

2. Assuming that each unique value in customer_id column represents a customer, I found unique customers with customer IDs to eliminate the repetition of customers. Later I counted their total number.

```
Unique Customers with their Customer IDs:
 ['1be2b93a-6b36-4481-bed1-a7b0766847f0'
  '52c8c88f-cf76-4b3c-bb02-d06b6215db29'
  'a3f2b3d9-c8fa-4824-822d-5635a8e55141'
  'b8ffb515-51db-4b56-8cff-e67d36d7f702'
  'ef7b9c72-42ea-4374-81d0-160c743ed721'
  'cf9427e1-1e91-48ab-b75d-731e16efbf37'
  'b83b6a45-27ab-4903-a1f0-f68f517ed64a'
  '405371ec-d73d-4419-aa34-3a9274b8e628'
  '2905fd71-ea4d-4d20-aca6-140a114fef5d'
  'ebb7223d-5d84-409b-84ee-4af5a99dd8cc'
  'df515183-50d8-4c59-91f7-46ffe68afa34'
  'e9985e5c-24b2-4a2c-8567-7b29cbe4db8b'
  '1d98f466-a32b-4fbd-914d-cd2dd40c821b'
  '99182215-acbd-4aed-b428-e6efc783c32b'
  '58f939b5-09a1-44f7-a321-9637c29285dd'
  'f3d2413f-de96-4a4f-9504-fc4247468e53'
  'ca59ad35-7cd2-4e6a-9c2c-e65c536328b8'
  '3164ad9a-90bf-4aa4-b1b7-d6da720b3474'
  'd1bc5ed8-647e-421c-95ce-cadfd46c5efd'
  'c10f176e-e7f7-4eaf-baf2-ad95bd4bd906'
  'a7e21a4c-2d77-4ce5-857c-07d6d9299b4d'
  'ecd63c4c-3988-4ebb-9527-902c27a0fe27'
  '86773ab6-b6db-4c7a-aafb-9f9f6eeef1fb'
  'e1b6d544-1f3b-402b-804b-0b6eb7b18f70'
  '5b53041e-b6de-4c3f-965c-2b0677887b38'
  '7260a8e2-8437-48e0-b004-cd7a9b60ba23'
  '7ad00a02-712c-4c08-bfe6-2c37bdcb695e'
  '6d54814b-4b68-4a3f-857f-d1248f1258a7'
  '964dfa79-b4e3-4769-bbaa-b61aeeaff1da'
  'eb55040e-eda4-4bab-aa43-193f213a59bf'
  'b7d6c4f3-8388-45d7-9528-2196db8d3501'
  '0d4c064c-9845-46fa-9cbd-26c99462e074'
  '1be64951-8621-4cdd-9e8d-bcb78c9927a0'
  '4942aa75-e79d-43da-9fe6-334f109a88d8']
Total number of these customers: 34
```

3. The customer data can also be found on a monthly or quarterly basis which shows the results of IHC per month. I found the data for May, June, July by locating the date string in session IDs.

```
In [361]:
#Monthly sessions for may, june, july

mai_sessions=df_attribution_results[df_attribution_results['session_id'].str.contains('2021-05')]
juni_sessions=df_attribution_results[df_attribution_results['session_id'].str.contains('2021-06')]
juli_sessions=df_attribution_results[df_attribution_results['session_id'].str.contains('2021-07')]
juni_sessions.head(10)
```

	conversion_id	session_id	initializer	holder	closer	ihc
0	1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-06-06_0001__aaf3b324-758d-d819-a2ee-4e51e...	0.0000	0.0	0.0000	0.0000
6	52c8c88f-cf76-4b3c-bb02-d06b6215db29	2021-06-20_0001__7084ac0f-3021-e583-2432-a8a53...	1.0000	1.0	1.0000	1.0000
7	a3f2b3d9-c8fa-4824-822d-5635a8e55141	2021-06-20_0001__138e910a-ecc5-3b5f-6b0e-df114...	1.0000	0.0	0.1322	0.3699
8	a3f2b3d9-c8fa-4824-822d-5635a8e55141	2021-06-21_0001__138e910a-ecc5-3b5f-6b0e-df114...	0.0000	1.0	0.8678	0.6301
15	ef7b9c72-42ea-4374-81d0-160c743ed721	2021-06-26_0001__92471ba5-7aa7-f59c-9f84-c85f7...	0.0000	0.0	0.0000	0.0000
21	cf9427e1-1e91-48ab-b75d-731e16efbf37	2021-06-25_0001__3d7d41d5-64f1-00ab-4474-a89a2...	1.0000	0.0	1.0000	1.0000
22	b83b6a45-27ab-4903-a1f0-f68f517ed64a	2021-06-14_0001__44dd3b40-5c90-a190-d607-be75f...	0.0000	0.0	0.0000	0.0000
23	b83b6a45-27ab-4903-a1f0-f68f517ed64a	2021-06-15_0001__9b0945a1-ea0c-2ba6-2055-100ea...	0.0000	0.0	0.0000	0.0000
25	405371ec-d73d-4419-aa34-3a9274b8e628	2021-06-28_0001__fbafa2bf-4d51-4050-5aa0-28adc...	0.6004	0.0	0.0000	0.2011
39	ebb7223d-5d84-409b-84ee-4af5a99dd8cc	2021-06-18_0001__2b0db8bc-a210-be02-cf83-217b3...	0.0000	0.0	0.0000	0.0000

4. I wanted to find all the sessions related to each customer to remove the redundancy, so I used pivot method which shows each customers' sessions and its IHC and initializer, holder and closer phases.

In [189]:

```
grouping_by_customer=pd.pivot_table(df_attribution_results, index=['conversion_id','session_id','ihc']
grouping_by_customer
```

			closer	holder	initializer
conversion_id	session_id	ihc			
0d4c064c-9845-46fa-9cbd-26c99462e074	2021-06-30_0001__fe5abf1b-cff8-2e4d-2fb0-c4c167dd193e	1.0000	1.0	0.0000	1.0000
1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-06-06_0001__aaf3b324-758d-d819-a2ee-4e51eca042ff	0.0000	0.0	0.0000	0.0000
	2021-07-13_0001__aaf3b324-758d-d819-a2ee-4e51eca042ff	0.5829	0.0	0.7667	0.8221
	2021-07-19_0001__2594bffe-76af-4b40-b0f7-12a8f974c6c9	0.4171	1.0	0.2333	0.1779
1be64951-8621-4cdd-9e8d-bcb78c9927a0	2021-06-19_0001__27bc5f0f-09cd-3fc7-1835-31faf9dc290d	0.3636	0.0	0.0714	1.0000
...	...	...	...	...	...
ef7b9c72-42ea-4374-81d0-160c743ed721	2021-07-14_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.0000	0.0	0.0000	0.0000
	2021-07-14_0002__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.3358	0.0	0.8375	0.0000
	2021-07-15_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.0000	0.0	0.0000	0.0000
	2021-07-16_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.3292	1.0	0.1625	0.0000
f3d2413f-de96-4a4f-9504-fc4247468e53	2021-07-13_0001__aa03968a-72db-3e34-dc14-fb8c418142ce	1.0000	1.0	0.0000	1.0000

120 rows × 3 columns

5. If we want to find the data as per any condition, that can be also done using .query() method. I found the customers with all sessions having extreme IHC values below 0.5 or above 0.9. Further, *Average* for all session values for one customer can also be found to compare impact of our campaign over them.

In [196]:

```
ihc_specific_range=grouping_by_customer.query('ihc>= 0.9 | ihc<=0.5')
ihc_specific_range.nunique() #total count of findings per phase
ihc_specific_range #finds particular entries with a criteria
```

			closer	holder	initializer
conversion_id	session_id	ihc			
0d4c064c-9845-46fa-9cbd-26c99462e074	2021-06-30_0001__fe5abf1b-cff8-2e4d-2fb0-c4c167dd193e	1.0000	1.0	0.0000	1.0000
1be2b93a-6b36-4481-bed1-a7b0766847f0	2021-06-06_0001__aaf3b324-758d-d819-a2ee-4e51eca042ff	0.0000	0.0	0.0000	0.0000
	2021-07-19_0001__2594bffe-76af-4b40-b0f7-12a8f974c6c9	0.4171	1.0	0.2333	0.1779
1be64951-8621-4cdd-9e8d-bcb78c9927a0	2021-06-19_0001__27bc5f0f-09cd-3fc7-1835-31faf9dc290d	0.3636	0.0	0.0714	1.0000
	2021-06-20_0001__fd2e99ac-a123-fc7e-7bd7-b500894496f6	0.0000	0.0	0.0000	0.0000
...	...	...	...	...	...
ef7b9c72-42ea-4374-81d0-160c743ed721	2021-07-14_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.0000	0.0	0.0000	0.0000
	2021-07-14_0002__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.3358	0.0	0.8375	0.0000
	2021-07-15_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.0000	0.0	0.0000	0.0000
	2021-07-16_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7	0.3292	1.0	0.1625	0.0000
f3d2413f-de96-4a4f-9504-fc4247468e53	2021-07-13_0001__aa03968a-72db-3e34-dc14-fb8c418142ce	1.0000	1.0	0.0000	1.0000

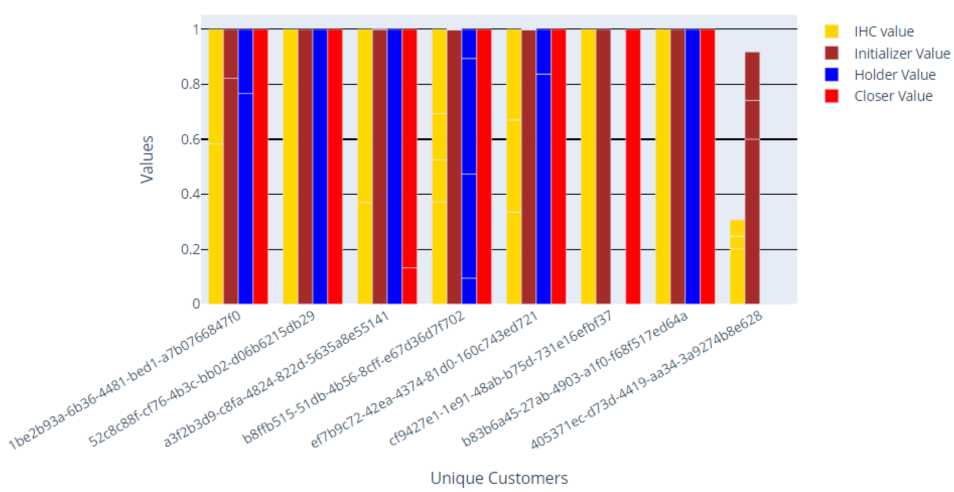
112 rows × 3 columns

6. *Batch processing*: Plotting all the conversion_ids and sessions is hard on a single plot so I sliced the 'df_attreibution_results' into first 30 entries as a batch. Then I counted their unique ids and total count.

```
In [26]:
#batch processing
df_attribution_results.head(5)
sliced_attr_result = df_attribution_results.iloc[:30,:]
sliced_attr_result.head(5)
print("Unique customers out of 30 session ids are:\n",sliced_attr_result['conversion_id'].unique(),
      "\nwith the total count of",sliced_attr_result['conversion_id'].nunique())
```

```
Unique customers out of 30 session ids are:
 ['1be2b93a-6b36-4481-bed1-a7b0766847f0'
 '52c8c88f-cf76-4b3c-bb02-d06b6215db29'
 'a3f2b3d9-c8fa-4824-822d-5635a8e55141'
 'b8ffb515-51db-4b56-8cff-e67d36d7f702'
 'ef7b9c72-42ea-4374-81d0-160c743ed721'
 'cf9427e1-1e91-48ab-b75d-731e16efbf37'
 'b83b6a45-27ab-4903-a1f0-f68f517ed64a'
 '405371ec-d73d-4419-aa34-3a9274b8e628']
with the total count of 8
```

7. I plotted those unique customers on a bar plot with all the related I/H/C values. Hovering over the bar makes it easy to get good details. *Missing bars* represent zero value.

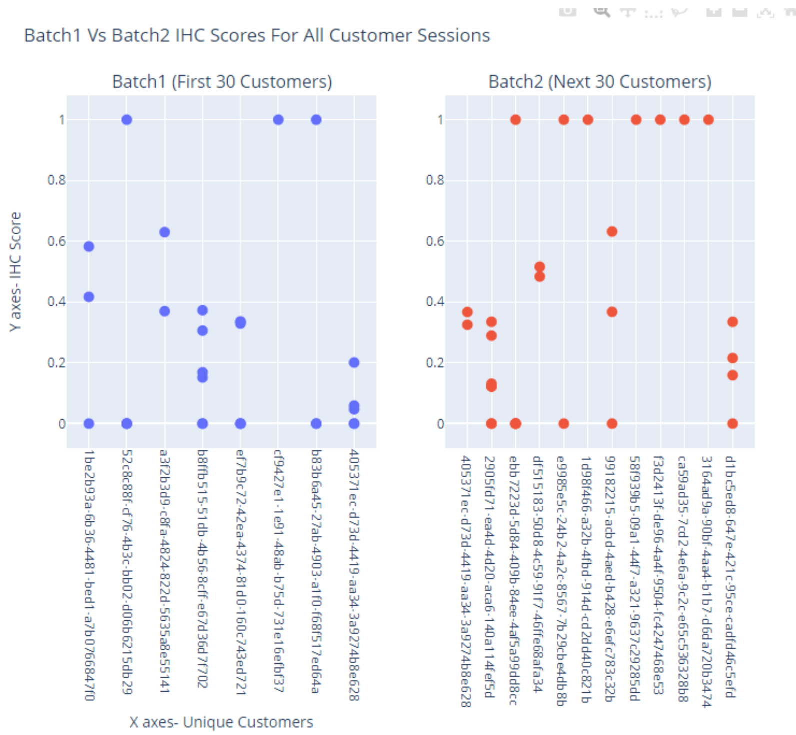


8. In I created a second batch of the next 30 customers and stored it in var 'sliced_attr_result2'.

```
In [259]:
sliced_attr_result2 = df_attribution_results.iloc[30:60,:] #creating second batch of next 30 conversio
sliced_attr_result2.head(5)
```

	conversion_id	session_id	initializer	holder	closer	ihc
30	405371ec-d73d-4419-aa34-3a9274b8e628	2021-07-16_0001__275455a2-fa51-cf30-955a-6f618...	0.0206	0.7939	0.0	0.3253
31	405371ec-d73d-4419-aa34-3a9274b8e628	2021-07-20_0001__3e560db4-5035-2e5f-13a2-805c2...	0.0616	0.2061	1.0	0.3672
32	2905fd71-ea4d-4d20-aca6-140a114fef5d	2021-07-05_0001__5e352d46-fdd5-ab6a-e650-a1470...	1.0000	0.0000	0.0	0.3350
33	2905fd71-ea4d-4d20-aca6-140a114fef5d	2021-07-06_0001__06f7e293-7382-3767-fdb9-8fc73...	0.0000	0.0000	0.0	0.0000
34	2905fd71-ea4d-4d20-aca6-140a114fef5d	2021-07-07_0001__4c0448bc-a31a-510c-4fe2-55df9...	0.0000	0.0000	0.0	0.0000

9. The purpose of slicing the next 30 conversion IDs was to compare this batch with the first batch. ‘ihc_vs_ihc2’ shows the difference of IHC values of both sets of customers. It depicts the pattern of varying IHC values and what batch has better performance with higher scores and what batch has more area for improvement. It is important to note that there can be a difference in the number of conversion IDs on x axis for both plots simply because some of them have a zero IHC score which is not shown on graph.



10. I used the not repeated mean values of I/H/C and IHC columns so that we can have a reference to compare it to the actual value of it as a baseline.

```
In [9]:
mean_IHC_vals= df_attribution_results.groupby('conversion_id', as_index=False).mean()
mean_IHC_vals.head(5)
```

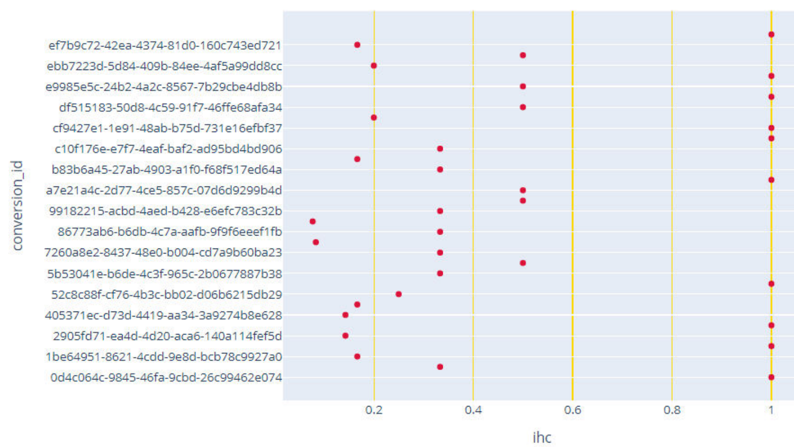
	conversion_id	initializer	holder	closer	ihc
0	0d4c064c-9845-46fa-9cbd-26c99462e074	1.000000	0.000000	1.000000	1.000000
1	1be2b93a-6b36-4481-bed1-a7b0766847f0	0.333333	0.333333	0.333333	0.333333
2	1be64951-8621-4cdd-9e8d-bcb78c9927a0	0.166667	0.166667	0.166667	0.166667
3	1d98f466-a32b-4fbd-914d-cd2dd40c821b	1.000000	0.000000	1.000000	1.000000
4	2905fd71-ea4d-4d20-aca6-140a114fef5d	0.142857	0.142857	0.142857	0.142857

11. I calculated the mean values of whole column for all the values for comparison purposed and to see the performance of one session’s I/H/C and IHC values as compared to the category average of other sessions. ‘maxwtphase’ gives us the maximum of all these values to see the strongest phase of all. This function can also be

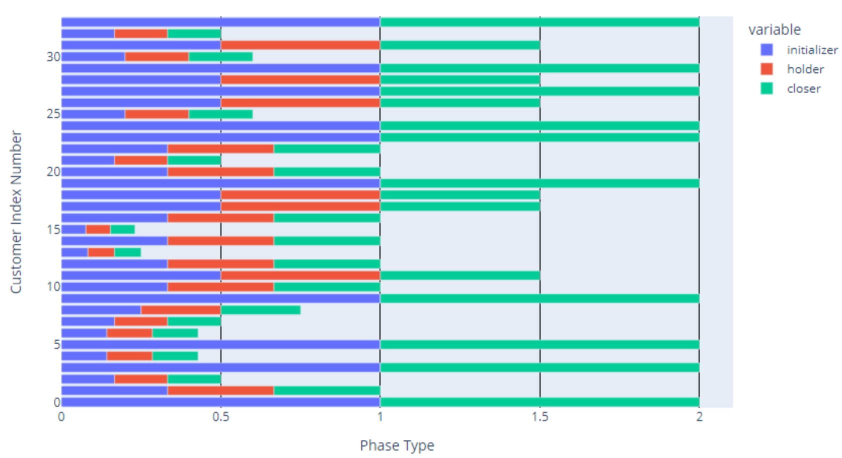
modified to give minimum, average, sum or other properties. This function can be called anytime to see the reference mean weights.

```
Average value of initializer phase is 0.2833333333333315
Average value of holder phase is 0.19999999999999968
Average value of closer phase is 0.2833333333333334

Note: More the value, more of the customers stay in that phase.

The maximun phase value is the 0.2833333333333334 for this set of customers.
```

12. To better visualize the IHC column of those 34 unique conversion IDs, I plotted a scatter plot of its mean values against the respective conversion IDs.



13. Next aim was to show all the data of all 34 unique customers into a single plot to save time and space.



14. I computed the averages of all three I/H/C phases for one customer session at a time to find the larger scale comparison of all such customer sessions. This average could also be used to see which session is the most successful session if a customer has multiple sessions. Then we will look at what phase is most impactful in that specific successful session.

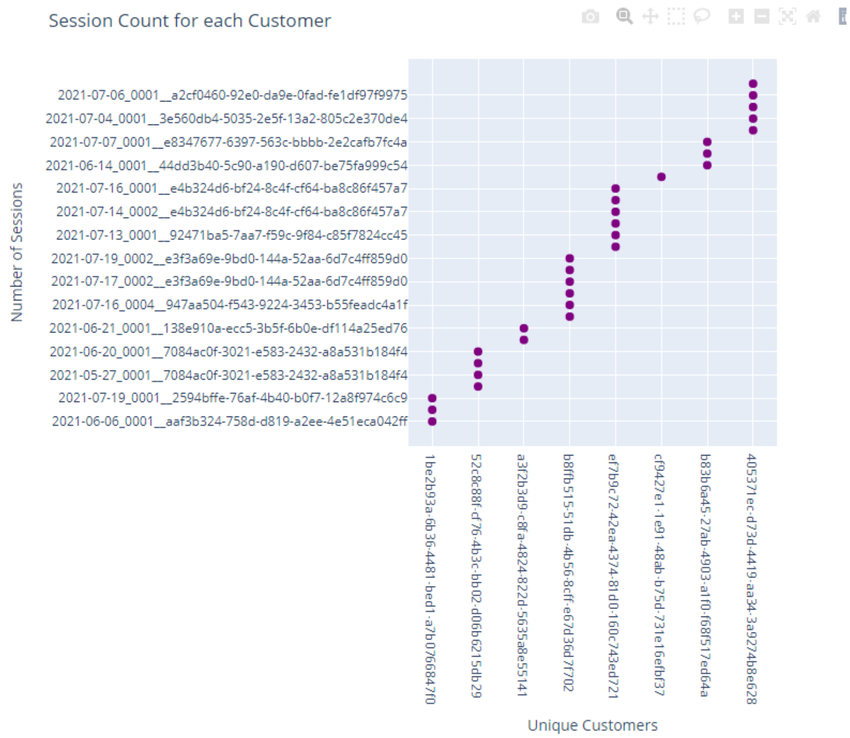
```
conversion_id                         session_id                                         ihc
0d4c064c-9845-46fa-9cbd-26c99462e074  2021-06-30_0001__fe5abf1b-cff8-2e4d-2fb0-c4c167dd193e  1.0000    0.666667
1be2b93a-6b36-4481-bed1-a7b0766847f0  2021-06-06_0001__aaf3b324-758d-d819-a2ee-4e51eca042ff  0.0000    0.000000
                                      2021-07-13_0001__aaf3b324-758d-d819-a2ee-4e51eca042ff  0.5829    0.529600
                                      2021-07-19_0001__2594bffe-76af-4b40-b0f7-12a8f974c6c9  0.4171    0.470400
1be64951-8621-4cdd-9e8d-bcb78c9927a0  2021-06-19_0001__27bc5f0f-09cd-3fc7-1835-31faf9dc290d  0.3636    0.357133
                                                                                                        ...
ef7b9c72-42ea-4374-81d0-160c743ed721  2021-07-14_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7  0.0000    0.000000
                                      2021-07-14_0002__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7  0.3358    0.279167
                                      2021-07-15_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7  0.0000    0.000000
                                      2021-07-16_0001__e4b324d6-bf24-8c4f-cf64-ba8c86f457a7  0.3292    0.387500
f3d2413f-de96-4a4f-9504-fc4247468e53  2021-07-13_0001__aa03968a-72db-3e34-dc14-fb8c418142ce  1.0000    0.666667
Length: 120, dtype: float64
```

15. Then I found the averages of one customer's all session values. For example, if xyz customer has four sessions, then I computed I/H/C values for all four sessions. It was really interesting for me to find that the sum is always 1 for all values of a specific phase scattered in multiple sessions of one customer. Therefore, if one phase has no impact in any of the sessions for that customer, naturally, average impact becomes zero showing that other two phases are dominating the average performance.

```
In [364]:
#Lets calculate averages for one customer's all the sessions.
#sum is always 1 for all values of all sessions if only one phase is considered
all_sessions_avg= grouping_by_customer.groupby(['conversion_id'], as_index=False).mean()
all_sessions_avg.tail(4)
```

	closer	holder	initializer
30	0.200000	0.200000	0.200000
31	0.500000	0.500000	0.500000
32	0.166667	0.166667	0.166667
33	1.000000	0.000000	1.000000

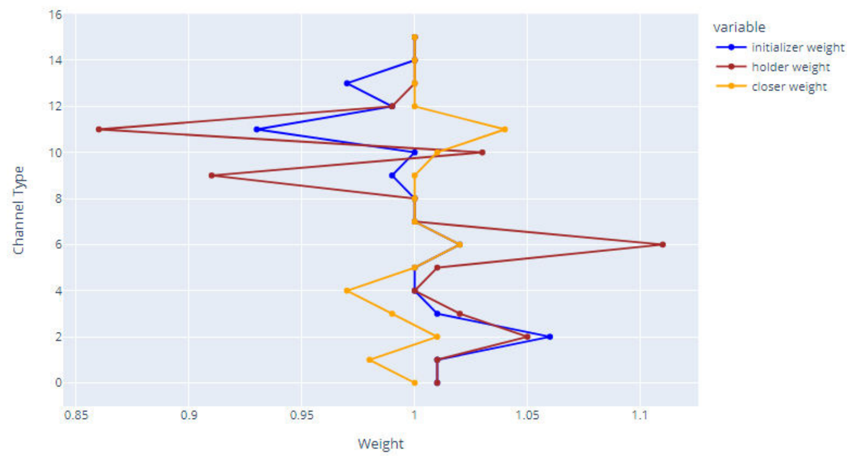
16. Then I plotted a graph showing how many sessions one customer has in that dataset. This will tell us which customer is most engaged to us so that we can focus on closing those customers and putting more efforts into single session customers to increase their future engagement.



17. I read the downloaded csv file of IHC channel weights. Unfortunately, I could not figure out the way to append the respective IHC weights into this table but I tried to analyze the data at hand efficiently.

	channel	initializer weight	holder weight	closer weight
0	affiliate	1.01	1.01	1.00
1	referral	1.01	1.01	0.98
2	search-brand	1.06	1.05	1.01
3	search-other	1.01	1.02	0.99
4	paid-other	1.00	1.00	0.97
5	talent	1.00	1.01	1.00
6	organic search (non-brand)	1.02	1.11	1.02
7	organic-social	1.00	1.00	1.00
8	organic social media	1.00	1.00	1.00
9	paid-social	0.99	0.91	1.00
10	organic search (brand)	1.00	1.03	1.01
11	direct	0.93	0.86	1.04
12	search	0.99	0.99	1.00
13	crm	0.97	1.00	1.00
14	display	1.00	1.00	1.00
15	video	1.00	1.00	1.00

18. I plotted the graph showing all the phases attached to their respective channels represented by the index number.



19. I calculated the holder weights that are more than 1. It was interesting to see that all such phases had Initializer and Closer phases with weights more than 1 too. Such high weight phase can be transitioned to same phase but with different channel to continue successful customer acquisition. This loc can be tweaked to get other phases and their channels also.

```
holder_emphasis=IHC_wts.loc[IHC_wts['holder weight'] > 1]  #holder wt is more means need to focus more
holder_emphasis#bcoz they are in deciding phase, same can be done for initializer wts
```

	channel	initializer weight	holder weight	closer weight
0	affiliate	1.01	1.01	1.00
1	referral	1.01	1.01	0.98
2	search-brand	1.06	1.05	1.01
3	search-other	1.01	1.02	0.99
5	talent	1.00	1.01	1.00
6	organic search (non-brand)	1.02	1.11	1.02
10	organic search (brand)	1.00	1.03	1.01

20. I computed the worst channels in terms of initializing, holding and closing weight assuming that the bigger value represents the stronger phase performance of that channel and vice versa. I considered 1 as the benchmark weight for good performance of a phase and found phase weights less than or equal to 1 for at least one of the three phases. *AND* operator can be used to find weights >=1. Methods with at least one phase with weight less than 1 which demands more work from marketeers on that phase.

```
In [208]:
less_efficient_methods=IHC_wts.loc[(IHC_wts['initializer weight'] <1 ) | (IHC_wts['holder weight'] <1)
less_efficient_methods #methods with atleast one phase with wt less than 1 which demads more work from
less_efficient_methods
```

	channel	initializer weight	holder weight	closer weight
1	referral	1.01	1.01	0.98
3	search-other	1.01	1.02	0.99
4	paid-other	1.00	1.00	0.97
9	paid-social	0.99	0.91	1.00
11	direct	0.93	0.86	1.04
12	search	0.99	0.99	1.00
13	crm	0.97	1.00	1.00

21. Related to closer weight, I computed the highest value which derives the most useful channel in terms of finalizing customer purchases. This channel can be weaker in other phases but still stronger in closing a customer journey. Then I summoned the other details of that respective channel for more clarification. Of course, this code can be twisted to get the data of initializer and holder phases too. We can also calculate the min() value of such phases to see the least efficient phases.

```
In [477]:
#Looking for the most efficient channel weight in terms of closing a customer even though I/H phases a

best_channel_phase_closer = IHC_wts['closer weight'].max()
better_wt= IHC_wts.loc[IHC_wts["closer weight"]==best_channel_phase_closer]
display("The strongest journey finalization(closer) is",better_wt) #more efficient the channel,lesser
```

```
'The strongest journey finalization(closer) is'
```

	channel	initializer weight	holder weight	closer weight
11	direct	0.93	0.86	1.04

22. Next task was to find the average weights of each phase. I distributed those weights over the total channel count which was 16 by dividing those three averages with channel count. Although ‘max_avg’ gives us the biggest average of all three columns, a distributed average over all channels will give us the granular detailed view of what a phase’s contribution is in that channel’s performance. In this dataset, all the values are pretty much nearby so the average does not fluctuate much but for a different dataset with large differences in values, the results could have larger differences among them.

```
Following 3 distributed averages tell us how much a phase average has contributed in that channel's overall performence:
1. Initialize phase average per channel is 0.0624609375
2. Holder phase average per channel is 0.0625
3. Closer phase average per channel is 0.062578125

The strongest phase performance(average of a phase) in general is 1.00125

The maximum average of a phase distributed all over all channels is 0.062578125 ,
tells us that phase's contribution in each channel's performance.
```