Wagle - Commercial Data Developer - Interview Exercise (Python)

* Home
* Workspace
* Recents
* Data

[Clusters](https://community.cloud.databricks.com/?o=238865118750664" \l "setting/clusters)

[Jobs](https://community.cloud.databricks.com/?o=238865118750664" \l "joblist)

* Search

spark2.4.5





[Cmd 1](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020077)

**Commercial Data Developer - Interview Exercise**

The Problem We have a stream of event level data from our audience data platform (see sample below). Within this are events associating anonymous customer ids with behavioural segment ids. From this stream we would like to build a customer-segment data set from which we can find:

* How many customers are in each segment
* For an individual customer, which segments are they in The Exercise

[Cmd 2](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020078)

I used apache spark with databrick platform to analyse and to answer above problem statements

[Cmd 3](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020079)

Down loaded data from S3 location ([https://reach-solutions-interview-data.s3-eu-west-1.amazonaws.com](https://reach-solutions-interview-data.s3-eu-west-1.amazonaws.com/)) placed at "/FileStore/tables/wagal/".

Loaded data using spark and created dataframe for future anlysis

[Cmd 4](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020080)



# File location and type

file\_location = "/FileStore/tables/wagal/\*"

file\_type = "json"

​

# CSV options

infer\_schema = "false"

first\_row\_is\_header = "false"

delimiter = ","

​

# The applied options are for CSV files. For other file types, these will be ignored.

df = spark.read.format(file\_type) \

.option("inferSchema", infer\_schema) \

.option("header", first\_row\_is\_header) \

.option("sep", delimiter) \

.load(file\_location)

​

display(df)

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

df:pyspark.sql.dataframe.DataFrame = [country: string, events: array ... 2 more fields]

**country**

**events**

**id**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**11**

FR

[{"add": [83419080, 56540340, 83418825, 83412074, 83418819, 83413260, 83418817, 83418914, 83405560, 83411653, 83406532, 83413235, 83418790, 83418784, 83418781, 83416590, 41668667, 83414407, 83411230, 52020042, 83417574, 51307418, 51307417, 83414790, 83417562, 83417162, 83412301, 83406426, 83410394, 41668618, 41668617, 83414356, 83414352, 83414349, 41668607, 83414342, 83407974, 83411942, 83411542, 83414716, 83413125, 83418281, 83419072, 83407954, 83418275, 83405570, 83418272, 83413507, 83405567, 83419062, 83405563, 83413502, 83417868, 83411118, 83418675, 83419055, 83413496, 83417859, 49289752, 83417856, 83407929, 49289745, 49289744, 83410701, 83407920, 83411880, 83419025, 83419023, 83407901, 83419015, 83411334, 83407894, 83411861, 83418211, 83418202, 83413437, 83418993, 83418991, 83411447, 83418191, 83407868, 83418185, 83406671, 83418182, 49289751, 83418575, 83411821, 83407848, 83407842, 83410611, 83405840, 83407788, 83411396, 83405836, 83418934, 83405572, 52272296, 83405894, 83406465, 49534909, 83417336, 83407683, 83407879, 83413359, 83411371, 83407796, 83414544, 83411762, 83411364, 83418508, 83411296, 83414516, 83411628, 53157177, 53157176, 53157175, 53157174, 83413335, 83413420, 83413908, 83413330, 83414520, 83414847, 50662413, 50662412, 50662407, 50662406, 83417387, 83411725, 83418278, 83418072, 83419085, 50662389, 50662388, 83416504, 51232895, 83407874, 50662369, 50662368, 50662367, 83418892, 50662364, 50662363, 50662362, 50662361], "c": null, "remove": [71509977, 76108975, 81342672, 71510006, 76108969, 76108968, 76108962, 71509993, 76108957, 76108956, 76108952, 71509986, 71509985, 76109146, 76109145, 81342646, 71509981, 71509980, 71509979, 71509978, 76109139, 71509976, 71509975, 71509974, 71509971, 71509969, 76109130, 71509967, 71509966, 76109126, 76109121, 76109081, 76108912, 76109106, 76108908, 76109103, 76109101, 76109100, 81342601, 81342797, 81342599, 81342598, 76109094, 76109093, 81342593, 76109086, 76109149, 81342585, 81342583, 81342578, 81342576, 76109072, 76109071, 81342572, 81342571, 81342569, 76109055, 76109045, 81342737, 81342739, 81342738, 76109038, 81342735, 81342841, 76109034, 76109031, 76109039, 71509959, 76108958, 76109016, 81342785, 76109012, 76109011, 76109010, 76109008, 76109006, 76109005, 76109004, 76109003, 76109166, 76109168, 81342684, 71509991, 76108995, 76109129, 76108992, 81342690, 81342808, 76108985, 81342683, 76108983, 76108982, 81342680, 76108990], "src": "ldx", "subSrc": "04eae1bb", "tap": "S2S", "ts": 1605209845}]

{"type": "cookie", "val": "e1e1a758ae36ddbddebe5fbbed52a09f"}

GB

[{"add": [51307476, 65621308, 71510007, 71509986, 71509985, 50663117, 71509982, 71509981, 71509980, 71509979, 71509976, 41612230, 71509964, 50663187, 71509960, 66741586, 41668651, 41668646, 41668643, 41668641, 50663118, 41668637, 54786703, 76109168, 54786692, 41668618, 41668617, 41612242, 41668614, 49289822, 49534770, 49534769, 76109154, 49534765, 49289815, 49289814, 49289813, 49289812, 54083982, 54083981, 54083980, 54083979, 54083978, 54083976, 41668594, 41668592, 81342785, 41668589, 76109130, 49289820, 54786654, 76109126, 41668733, 76109121, 51307490, 54786636, 76109106, 54786628, 41668867, 81342735, 41668758, 76109071, 76109060, 83405763, 41668606, 41668881, 41668880, 76109026, 49289821, 76109003, 65609937, 76108908, 76108992, 76108985, 65609921, 65609920, 65609919, 65609918, 65609917, 65609916, 65609915, 65609914, 50663259, 50663256, 76108970, 76108969, 50663253, 50663252, 50663251, 76108958, 76108956, 50663236, 41668787, 51232912, 50663214, 76108928, 51232906, 52272251, 51232904, 50663207, 50663206, 50663199, 52020144, 50663195, 50663194, 50663193, 41668760, 50663191, 50663190, 50663189, 50663188, 76108902, 50663186, 76108900, 49289807, 49289808, 51307500, 51307499, 51307498, 51307497, 51307496, 51307495, 50663160, 49289819, 51307489, 51307488], "c": null, "remove": [84044777, 84044749], "src": "ldx", "subSrc": "04eae1bb", "tap": "S2S", "ts": 1605209891}]

{"type": "cookie", "val": "c536806aa12ff369fa5f0534657af8f9"}

GB

[{"add": [51307497, 52711361, 51307490, 50663259, 49534772, 52020144, 52272251, 50663252, 52272249, 50663207, 50663199, 50663242, 50663194, 49534770, 49534769, 50663236, 49534765, 50663187, 50663214, 51307475], "c": null, "remove": [71509986, 76109106, 41668787, 41668832], "src": "ldx", "subSrc": "04eae1bb", "tap": "S2S", "ts": 1605209903}]

{"type": "cookie", "val": "76dd138ff3d7eded92f6a3036d56e0f0"}

GB

[{"add": [50663122, 51307490, 52020144, 52272251, 49534773, 49534772, 50663194, 52711359, 49534766, 49534765, 50663187, 50663180], "c": null, "remove": null, "src": "ldx", "subSrc": "04eae1bb", "tap": "S2S", "ts": 1605209927}]

{"type": "cookie", "val": "1c4a091f5bf5026db2b08aa965aa629d"}

GR

[{"add": null, "c": null, "remove": [23305842, 23305835, 55237637, 71509980, 71509963, 23305807, 64701968, 23305778, 64701160, 64701159, 23305773, 57700619, 23305764, 64701142, 23305757, 23305756, 23305754, 23305753, 23305752, 64701134, 48202436, 64701131, 64701129, 64701127, 48202428, 64701124, 57700274, 64701117, 64701116, 48202418, 48202417, 48202416, 23305670, 64701109, 64701103, 64701100, 64701888, 64701887, 64701885, 64701086, 64701882, 48202387, 64701877, 64701876, 48202381, 23305158, 48202379, 64701161, 64702324, 76109067, 64701861, 64701850, 64701848, 64701835, 64701826, 64701824, 64701122, 64701822, 64701821, 64701819, 64701817, 64701816, 64701812, 64701811, 64701135, 64701139, 57700158, 64701147, 23305593, 57700126, 64702566, 23305586, 57700915, 23305272, 57700910, 64702548, 23306367, 23305283, 64702544, 64702334, 64701731, 48203811, 64702474, 64702470, 64702469, 64702467, 23306286, 23305478, 64702453, 64702452, 64702448, 23306267, 23306264, 76109106, 57700613, 64702420, 23305438, 23305157, 64702411, 23305433, 23305432, 23305431, 64702407, 23305291, 64702402, 23306025, 64702399, 23305418, 23306213, 64702385, 23306174, 57700699, 57700698, 76109003, 64702338, 57700692, 64702335, 23306154, 64702333, 57700679, 57700668, 57700664, 57700655, 57700654, 57700651, 64702294, 64702293, 23305315, 64702291, 23305313, 64702289, 23305311, 57700638, 64702272, 57700626, 57700624, 57700623, 64702268, 64702266, 57700620, 64702264, 57700616, 57700615, 57700614, 64702258, 57700611, 57700608, 64702252, 57700606, 57700604, 57700602, 64702290, 23305260, 23305258, 23305257, 23305256, 23305255, 23305254, 23306048, 23306043, 64702217, 23305233, 48202711, 23305224, 23306020, 23305218, 23305215, 23306007, 23305201, 23305200, 23305198, 23305196, 48202677, 48202672, 48202671, 23305984, 48202669, 23305982, 48202667, 23305183, 48202665, 23305181, 48202661, 64702153, 48202654, 23305681, 23305166, 64702142, 23305161, 48202643, 64702135, 64702134, 64702130, 23305948, 64702408, 64702126, 23305945, 64702123, 64702122, 64702120, 64702114, 48202613, 48202609, 48202606, 48202605, 48202601, 48202597, 23305910, 48202594, 48202593, 48202590, 48202589, 48202586, 23305591, 57700648], "src": "ldx", "subSrc": null, "tap": "S2S", "ts": 1605209113}]

{"type": "cookie", "val": "d9f67e873d81d8e91b9b9284413fabb6"}

IE

[{"add": [64702453, 64702452, 64702448, 64702438, 77832971, 77832654, 77832966, 77832965, 77832960, 64702420, 77832950, 77832949, 41668881, 41668880, 41668878, 64702408, 64702407, 64702402, 64702399, 64702390, 64702389, 64702388, 64702386, 64702385, 64702381, 64702380, 41668848, 64702378, 64702373, 64702372, 41668836, 41668835, 41668849, 64702379, 41668816, 41668815, 41668814, 41668813, 41668812, 41668811, 41668810, 64702338, 77832751, 41668805, 64702335, 41668803, 64702333, 64702329, 41668797, 41668796, 24256678, 64702294, 64702293, 23305984, 64702291, 64702290, 41668758, 77832819, 77832699, 23306438, 57700569, 64702266, 41668734, 41668733, 64702253, 64702252, 64702251, 77832773, 77832762, 64702217, 77832749, 41668682, 41668675, 48600603, 41668664, 41668663, 23306354, 41668647, 41668646, 41668645, 41668644, 41668639, 41668637, 41668634, 57700915, 41668631, 41668627, 41668626, 41668625, 41668623, 64702153, 41668618, 41668617, 41668616, 41668614, 64702142, 57700549, 64702140, 23305982, 64702135, 64702134, 41668602, 41668601, 64702130, 41668598, 41668597, 57700563, 41668595, 77610675, 23306286, 64702122, 77832655, 41668589, 57700573, 57700574, 64702114, 23306276, 77832972, 81342850, 23306271, 41668593, 57700602, 23306247, 57700611, 23306240, 57700616, 23306226, 23306217, 23306213, 64702048, 64702047, 64702046, 64702045, 64702043, 64702042, 64702041, 64702040, 64702039, 64702038, 64702037, 64702036, 64702034, 64702032, 64702031, 64702030, 64702028, 64702027, 64702026, 64702025, 64702024, 64702023, 64702022, 64702021, 64702019, 64702018, 64702017, 64702016, 64702015, 64702014, 64702012, 51232913, 64702007, 64702006, 57700752, 57700698, 64701983, 57700731, 23305333, 23306146, 23306138, 71509985, 71509984, 71509982, 71509981, 71509980, 71509979, 71509978, 71509977, 71509976, 71509975, 71509974, 64701960, 71509969, 71509966, 71509964, 57700704, 25937969, 57700728, 71509958, 57700726, 64701945, 57700692, 57700679, 33064945, 57700668, 57700664, 81342646, 57700655, 57700654, 57700651, 57700648, 64701888, 64701887, 64701885, 23306043, 57700631, 64701877, 57700626, 77832943, 57700624, 57700623, 64701870, 57700620, 57700619, 64701864, 57700615, 57700614, 57700613, 64701859, 64701857, 57700608, 57700606, 57700604, 64701850, 57700601, 64701848, 57700599, 23306002, 64702108, 64701835, 23305997, 23305996, 64702116, 64701826, 64702120, 64701824, 64701822, 64701821, 64701819, 64701817, 64701816, 57700565, 64701812, 64701811, 57700555, 64701797, 76109166, 76109149, 57700910, 76109146, 23305937, 76109139, 57700922, 57700514, 76109130, 76109129, 76109121, 23305910, 64701746, 76109106, 76109104, 64701731, 23305885, 23305884, 23305883, 23305882, 76109086, 76109074, 76109073, 76109071, 83405763, 76109067, 81342785, 76109060, 76109055, 41612256, 23305842, 76109045, 23305835, 76109034, 41612230, 76109026, 76109017, 76109016, 76109013, 76109011, 48600685, 76109008, 76109003, 76108992, 76108990, 23305778, 76108983, 64702334, 23305764, 76108967, 64702350, 23305757, 23305754, 23305753, 76108958, 76108956, 43204359, 43204349, 76108928, 57700307, 33064974, 76108908, 64702123, 23305681, 23305678, 23305670, 23305628, 33064880, 23596265, 23305592, 23305567, 23305555, 33064821, 33064816, 33064815, 33064811, 33064803, 33064795, 23305527, 75172877, 75172876, 33064755, 48202711, 23305478, 23305477, 41668760, 23305472, 48202678, 48202677, 48202671, 48202669, 23305438, 48202667, 48202665, 23305433, 23305432, 23305431, 48202654, 23305418, 48202645, 48202643, 48202642, 33064678, 48202639, 33064675, 76109168, 48202613, 48202609, 33064645, 33064644, 23305375, 48202601, 33064636, 48202597, 33064630, 48202590, 48202589, 23305357, 23305356, 33064622, 33064620, 64701968, 33064600, 23305326, 64701161, 23305319, 23305315, 23305302, 33064560, 23305292, 23305291, 64701124, 64701122, 23305283, 64701116, 23305272, 23305268, 48202661, 23305260, 23305258, 23305257, 23305256, 23305255, 23305254, 23305253, 64701069, 23305224, 64701060, 23305222, 23305218, 23305215, 77833172, 23305201, 23305200, 48202428, 23305196, 48202418, 48202417, 48202416, 48202414, 23305183, 23305177, 23305174, 48202586, 23305166, 48202593, 23305161, 23305158, 23305157, 23305154, 48202381, 48202376, 48202375, 48202374, 48202373, 48202372, 48202371, 64702566, 77833096, 77833095, 64702548, 64702544, 81342680, 33064624, 64702474, 64702470, 64702469, 64702467, 64702126], "c": null, "remove": null, "src": "ldx", "subSrc": "ef381196", "tap": "S2S", "ts": 1605209181}]

{"type": "cookie", "val": "79a544f3bb96ccf91b9688be6a3b3d6e"}

GB

[{"add": [49534766], "c": null, "remove": null, "src": "ldx", "subSrc": "04eae1bb", "tap": "S2S", "ts": 1605209198}]

{"type": "cookie", "val": "c6441acd106ecc160f2e7a06af066f2e"}

GB

[{"add": [35780778, 35780589, 35780498, 77832645, 77832646, 67339086, 37393738, 77832971, 37393736, 54083982, 51947195, 54083980, 37393732, 54083978, 77832964, 77832963, 35780755, 51947185, 77832957, 50663195, 51947182, 51947181, 51947180, 50663191, 50663190, 77832950, 77832949, 77832947, 51947173, 41668880, 51947171, 77832679, 77832942, 50663180, 51947167, 77832937, 77832936, 77832934, 50663172, 51947157, 35780723, 51947153, 77832699, 50663160, 51947147, 51947145, 51947144, 41668849, 51947140, 37392684, 41668846, 41668844, 41668843, 50663144, 50663143, 50663137, 51947122, 35780690, 37392942, 51947119, 51947117, 51947114, 35780682, 50663124, 50663122, 51947109, 50663120, 51947107, 50663118, 50663117, 50663116, 50663115, 41668810, 50663113, 50663112, 41668804, 41668803, 35780662, 37392270, 77832864, 84044609, 41668797, 41668796, 41668795, 35780655, 41668793, 41668792, 41668791, 41668790, 35780650, 35780649, 77832852, 41668784, 41668783, 41668780, 35780637, 35780479, 35780730, 77832786, 35780636, 35780629, 35780628, 41668760, 77832798, 77832799, 77832821, 35780616, 77832819, 35780611, 35780610, 35780608, 77832655, 77832972, 35780603, 41668741, 35780601, 77832803, 41668737, 77832800, 41668734, 41668733, 77832797, 77832796, 77832795, 77832794, 77832793, 77832792, 77832791, 77832790, 77832789, 37393554, 37391956, 37391955, 37391954, 37391952, 35780694, 37392470, 77832773, 37391940, 35780566, 37392829, 37391935, 77832853, 77832764, 77832763, 37391931, 37392835, 41668695, 41668694, 37393524, 37391926, 37391925, 41668688, 41668687, 41668683, 41668682, 41668681, 35780541, 41668675, 35779966, 77893346, 41668667, 41668666, 41668664, 41668663, 41668656, 41668655, 41668654, 41668653, 41668652, 41668651, 35780510, 35780508, 41668646, 37391877, 35780503, 41668639, 77832862, 41668637, 41668635, 35780495, 41668633, 41668631, 77832695, 77832694, 77832693, 77832692, 77832691, 77832690, 77832689, 41668623, 77832687, 35780482, 41668618, 41668617, 41668616, 41668614, 41668613, 41668612, 41668611, 77832943, 41668609, 41668608, 77832946, 41668606, 35780464, 35780463, 35780806, 37392933, 77832858, 37392935, 41668595, 77610676, 41668593, 41668592, 41668591, 41668590, 41668589, 77832652, 77610668, 77832650, 77832648, 77832647, 77610663, 77893331, 77832644, 35780673, 77610658, 35780436, 35780674, 35780431, 35780430, 35780428, 51232902, 37392971, 35780420, 33064777, 41668602, 35780406, 35780399, 35780392, 35780389, 35780388, 37393093, 50663159, 37391754, 77893269, 77893265, 35780359, 51232928, 51232913, 35780337, 35780333, 52026403, 51307476, 51232893, 37392329, 35780316, 35780314, 77832958, 71509986, 71509982, 71509981, 71509980, 37392340, 71509976, 35780293, 35780291, 35780289, 71509966, 71509963, 71509960, 35780017, 35780274, 35780272, 35780271, 35780268, 35780266, 35780265, 41668758, 35780252, 52020144, 52020142, 48600964, 51947197, 41668787, 41668789, 35780220, 35780218, 35780216, 35780215, 35780213, 37391928, 37393737, 51947217, 35780205, 35780204, 41668816, 35780196, 51947223, 35780193, 35780191, 35780187, 35780185, 84044713, 41668839, 41668840, 35780171, 41668848, 35780166, 35780163, 75644394, 35780156, 37393124, 37393123, 37393122, 35780149, 51947113, 35780145, 37393114, 37393113, 35780139, 41668878, 37393107, 41668881, 37393096, 69301645, 35780123, 37393092, 37393084, 37393081, 35780109, 52026389, 37393075, 35780104, 35780103, 37393070, 37393066, 35780092, 35780091, 35780087, 37393056, 37393055, 35780114, 35780082, 35780081, 76109121, 35780076, 35780075, 35780073, 37393035, 35780058, 35780057, 35780056, 37393025, 35780054, 37393023, 35780047, 51947038, 35780043, 54083979, 37393011, 37393010, 83405773, 48600754, 35780036, 37393004, 52272251, 76109071, 35780024, 35780023, 35780022, 37392988, 37392987, 41612256, 37392979, 37392978, 37392977, 35780006, 48600721, 37392974, 37392972, 50663121, 41612242, 83415302, 84044558, 35779978, 37392947, 37392946, 37392944, 48600688, 37392941, 37392940, 37392938, 37392937, 37392936, 52711359, 83415280, 51947121, 37392931, 37392930, 35779958, 35779955, 35779954, 50663173, 35779948, 35779945, 50663181, 35779939, 35779935, 37392903, 37392902, 37392900, 37392897, 35779924, 37392891, 37392890, 35779919, 35779917, 37392885, 51947170, 35779912, 35779911, 35779908, 51947177, 35779897, 51947190, 43204346, 35779889, 48600603, 35779884, 35779882, 37391916, 37392846, 33064974, 35779872, 51307500, 51307499, 51307498, 51307497, 51307496, 51307495, 37392831, 37392830, 51307490, 51307489, 37392826, 37392825, 37392824, 37392823, 37392822, 37392821, 37392820, 33064945, 37392814, 37392813, 37392811, 37392810, 37392808, 37392570, 37392800, 37392777, 37392794, 37392792, 37392788, 37392784, 37392582, 49534771, 52272249, 35779869, 49534766, 35779874, 35779879, 35779887, 35779893, 37392749, 33064877, 33064871, 33064869, 33064867, 33064864, 33064863, 33064862, 33064860, 37392726, 37392725, 37392724, 37392723, 37392722, 37392721, 37392720, 37392718, 37392701, 77610672, 33064821, 37392628, 37392686, 37392685, 33064814, 35779963, 33064810, 33064809, 33064803, 35780305, 33064800, 33064799, 33064798, 33064795, 33064794, 75172879, 75172876, 75172875, 37392645, 35780001, 33064771, 37462899, 37462898, 37462897, 33064758, 37462895, 37393733, 37393734, 37462890, 37393739, 37392618, 35780030, 35780032, 37462880, 37392611, 41668647, 35780038, 37392605, 37392604, 35780042, 37392600, 37392591, 37392589, 37392588, 37392587, 37392584, 37392583, 37392682, 37392581, 33064710, 37392579, 37392577, 37392576, 33064705, 37392574, 37392573, 37392572, 33064700, 33064699, 37392568, 37392566, 37392564, 37392562, 37392559, 37392558, 37392557, 41849781, 37392553, 33064682, 33064681, 37392550, 35780096, 33064678, 37392547, 37392546, 33064675, 35780105, 37392539, 37392538, 35780108, 37392536, 33064665, 37392533, 33064662, 33064661, 33064660, 33064659, 33064658, 33064657, 33064655, 37392522, 37392520, 35780126, 37392516, 33064645, 37392514, 50663189, 33064638, 33064637, 33064634, 37392266, 51947110, 37392554, 35780153, 37392290, 37392737, 33064600, 33064598, 33064597, 33064596, 33064595, 33064594, 33064592, 37394058, 33064590, 33064589, 33064588, 33064583, 37392450, 37392447, 37462886, 37392445, 33064574, 37392443, 37392439, 35779961, 35780209, 37392569, 37392434, 37392433, 37392432, 33064560, 37392428, 37462896, 33064555, 33064553, 33064552, 33064551, 33064550, 41612230, 52026404, 37392776, 35779976, 35780447, 37392384, 37392382, 37392381, 37392379, 37392368, 37392367, 35780279, 37392355, 37392354, 35780295, 35780299, 77833171, 37393935, 37392334, 37392436, 77833158, 37392323, 37392444, 37392446, 37392314, 37392462, 37392301, 37392467, 37392297, 54083976, 77832654, 54083981, 77833113, 37392280, 37392834, 37392836, 37392273, 37392837, 37392838, 77832668, 77833096, 77833095, 37392261, 77610677, 37392256, 77832678, 77832758, 37392528, 37392530, 37392532, 77832686, 35780527, 77832688, 37392540, 35780421, 33064780, 37392549, 37392551, 37392552, 35780434, 77832696, 77833035, 77833031, 35780826, 77833029, 77833027, 77833026, 35780451, 77833024, 77833023, 77833022, 37392575, 35780458, 50663256, 37392580, 77832760, 50663253, 49289829, 49289828, 49289827, 49289826, 84044535, 37392889, 51947225, 51947224, 49289815, 49289814, 49289813, 77832992, 35780786], "c": null, "remove": null, "src": "ldx", "subSrc": "5244a27b", "tap": "S2S", "ts": 1605209256}]

{"type": "cookie", "val": "4dcd06c1152d22d85aac3eec906a1577"}

GB

[{"add": [50621866, 80095535, 80095534, 80095533, 83418911, 71510007, 71510006, 83408094, 50621847, 83407297, 50621833, 71509986, 71509985, 71509982, 71509981, 71509980, 71509978, 71509977, 71509976, 71509975, 71509974, 71509971, 71509969, 83407661, 71509966, 71509964, 71509960, 83415588, 50621794, 83418750, 83408030, 83418747, 80095453, 83408018, 76109168, 50621368, 83412765, 83407999, 76109146, 81342785, 76109130, 81342780, 83406765, 81342737, 76109101, 83419061, 83406753, 83419059, 83419056, 50621295, 83419052, 81342739, 81342738, 76109086, 81342736, 81342735, 76109071, 83418228, 76109060, 76109055, 76109045, 50622038, 83419398, 83419392, 76109029, 76109026, 83419382, 50621623, 50622019, 50622017, 50621821, 76109017, 76109016, 83419373, 76109013, 83417385, 76109011, 76109010, 76109008, 76109004, 76109003, 83419360, 50621996, 76109024, 76108992, 76108990, 76108987, 76108985, 50621581, 50621578, 76109012, 83419332, 83415757, 50620775, 76109020, 76108967, 83408132, 76108958, 76108956, 81342601, 50621941, 83416124, 83419297, 50621520, 83415950, 83417388, 83418474, 76108908, 83415294, 83418985, 83418466, 83415289, 50621897, 83419248, 50621884, 51232866, 50621882, 83418051, 83415668, 50621869], "c": null, "remove": null, "src": "ldx", "subSrc": "ef381196", "tap": "S2S", "ts": 1605209284}]

{"type": "cookie", "val": "99f371b0ad54f7054bba81b2b2824d5"}

GB

[{"add": [76108908, 51307500, 71509986, 76109060, 76109067, 50663191, 50663118, 76109168], "c": null, "remove": null, "src": "ldx", "subSrc": "06cf82e8", "tap": "S2S", "ts": 1605209360}]

{"type": "cookie", "val": "5e7d12e618e9013b62c5a793b933d109"}

GB

[{"add": [48690285, 8851984, 81151927, 8722685, 641639, 24729698, 33231715], "c": 7101, "remove": null, "src": null, "subSrc": null, "tap": "DEVICE", "ts": 1605208848}, {"add": [48690285, 8851984, 81151927, 8722685, 641639, 24729698, 33231715], "c": 7101, "remove": null, "src": null, "subSrc": null, "tap": "DEVICE", "ts": 1605208500}, {"add": null, "c": 7101, "remove": [641639, 24729698, 33231715, 81151927, 948588, 7381710, 48690285, 648106, 8737601, 8851923, 8851984, 14198094, 1867314, 35111024, 8722685], "src": null, "subSrc": null, "tap": "S2S", "ts": 1605209177}]

{"type": "cookie", "val": "fb703fc1c9e3992f75c2505da3fef92"}

Showing the first 528 rows.

[Cmd 5](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020081)

**Filter out the events for GB users only, which associate customer ids with 1st party behavioral segment ids**

[Cmd 6](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020082)



#Diplay the schema to see available fields

df.printSchema()

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

root |-- country: string (nullable = true) |-- events: array (nullable = true) | |-- element: struct (containsNull = true) | | |-- add: array (nullable = true) | | | |-- element: long (containsNull = true) | | |-- c: long (nullable = true) | | |-- remove: array (nullable = true) | | | |-- element: long (containsNull = true) | | |-- src: string (nullable = true) | | |-- subSrc: string (nullable = true) | | |-- tap: string (nullable = true) | | |-- ts: long (nullable = true) |-- id: struct (nullable = true) | |-- type: string (nullable = true) | |-- val: string (nullable = true) |-- region: string (nullable = true)

[Cmd 7](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020083)



**from** pyspark.sql.functions **import** \*

​

#Flatten the events array and fill 1st party behavioral segment ids with -1 which are not associated with customer ids

flatten\_df = df.withColumn("event", explode("events")) \

.na.fill({'event.c': -1}) \

.na.fill({'event.add':  -1})

​

​

#Filter recrods which are associate customer ids with 1st party behavioral segment ids

res\_df = flatten\_df.filter((flatten\_df.country == "GB") & (flatten\_df.event.c != -1) ).drop("events")

​

#Pritnt schema of dataframe.

res\_df.printSchema()

​

#Show some sample records.

res\_df.show()

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

flatten\_df:pyspark.sql.dataframe.DataFrame = [country: string, events: array ... 3 more fields]

res\_df:pyspark.sql.dataframe.DataFrame = [country: string, id: struct ... 2 more fields]

root |-- country: string (nullable = true) |-- id: struct (nullable = true) | |-- type: string (nullable = true) | |-- val: string (nullable = true) |-- region: string (nullable = true) |-- event: struct (nullable = true) | |-- add: array (nullable = true) | | |-- element: long (containsNull = true) | |-- c: long (nullable = true) | |-- remove: array (nullable = true) | | |-- element: long (containsNull = true) | |-- src: string (nullable = true) | |-- subSrc: string (nullable = true) | |-- tap: string (nullable = true) | |-- ts: long (nullable = true) +-------+--------------------+------+--------------------+ |country| id|region| event| +-------+--------------------+------+--------------------+ | GB|[cookie, fb703fc1...| eur|[[48690285, 88519...| | GB|[cookie, fb703fc1...| eur|[[48690285, 88519...| | GB|[cookie, fb703fc1...| eur|[, 7101, [641639,...| | GB|[cookie, 4b8e2b9a...| eur|[, 7101, [1867303...| | GB|[cookie, 4b8e2b9a...| eur|[[34922509, 88519...| | GB|[cookie, f173a8c6...| eur|[, 7101, [1419809...| | GB|[cookie, 6351ea57...| eur|[, 7101, [2472970...| | GB|[cookie, 6351ea57...| eur|[[8851984, 810538...| | GB|[cookie, 80953c11...| eur|[[8851984, 810538...| | GB|[cookie, 80953c11...| eur|[, 7101, [8851923...| | GB|[cookie, 80953c11...| eur|[[8851984, 810538...| | GB|[cookie, 209c12d5...| eur|[[24729668, 65260...| | GB|[cookie, 209c12d5...| eur|[, 7101, [4868913...| | GB|[cookie, 202f2a2f...| eur|[, 7101, [948588,...| | GB|[cookie, 202f2a2f...| eur|[[8851891, 840083...| | GB|[cookie, f7d82cd4...| eur|[[8851891, 775339...| | GB|[cookie, f7d82cd4...| eur|[, 7101, [3323171...| | GB|[cookie, 8dd57b77...| eur|[, 7101, [649148,...| | GB|[cookie, 7fddfe92...| eur|[[24730841, 64636...| | GB|[cookie, 46576505...| eur|[[24729599, 88519...| +-------+--------------------+------+--------------------+ only showing top 20 rows

[Cmd 8](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020084)

To generate the report using sql way we need to create a temp table in spark

[Cmd 9](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020085)

temp\_table\_name = "view\_customer\_segemnt"

res\_df.createOrReplaceTempView(temp\_table\_name)

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

[Cmd 10](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020086)

**Question 1 : How many customers are in each segment**

[Cmd 11](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020087)

%sql

**SELECT** indi\_segment\_id **as** segment\_id, count(id.val)**as** customer\_count

**FROM** view\_customer\_segemnt

**LATERAL** **VIEW** explode(event.add) t\_segment **AS** indi\_segment\_id

**GROUP** **BY** indi\_segment\_id

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

**segment\_id**

**customer\_count**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**15**

**16**

**17**

**18**

648366

79

24729426

56

24729589

40

667129

6

651287

21

2296547

31

24729504

153

676082

2

24759451

11

24729494

36

34929332

69

651285

37

24731015

6

24730449

24

24729942

18

667065

7

24729944

27

24729401

104

Showing all 779 rows.

[Cmd 12](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020088)

Doing same as above, but using spark sql, however the query is same.

The advanatge is we can do any other operation on dataframe after we anlysys as like saving data in to a file etc

[Cmd 13](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020089)

query = "SELECT indi\_segment\_id as segment\_id, count(id.val) as customer\_count FROM view\_customer\_segemnt LATERAL VIEW explode(event.add) t\_segment AS indi\_segment\_id GROUP BY indi\_segment\_id"

res1\_df = spark.sql(query)

res1\_df.write.mode("overwrite").format("json").json("/FileStore/tables/count\_customers\_segment")

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

res1\_df:pyspark.sql.dataframe.DataFrame = [segment\_id: long, customer\_count: long]

[Cmd 14](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020090)

**Questions 2: For an individual customer, which segments are they in**

[Cmd 15](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020091)



%sql

​

**SELECT** id.val **as** customer\_id, collect\_set(indi\_segment\_id) **as** related\_customer\_segments\_ids, count(indi\_segment\_id) count\_related\_customer\_segments\_ids

**FROM** view\_customer\_segemnt

**LATERAL** **VIEW** explode(event.add) t\_segment **AS** indi\_segment\_id

**GROUP** **BY** id;

​

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

**customer\_id**

**related\_customer\_segments\_ids**

**count\_related\_customer\_segments\_ids**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**15**

**16**

**17**

**18**

1007bd3329ed45b202d5d0084a4a3bdb

[77803299, 24729646, 8722685, 81151927, 649132, 33231715, 8851891]

7

100934fa47c81a0df29da15ae431fd6d

[77803299, 8722685, 81151927, 33231715, 649346, 8851891, 24729525]

7

1021406292b3c7080ed834e0ea980e3f

[8722685, 14198230, 33231715, 28356555, 649147, 24729423, 8851895]

7

1027d8053008ea3541352f58950c338f

[77803299, 8722685, 84008342, 84615117, 8851891]

5

1052ae341da6042ab956470d1ab0a4dc

[8722685, 8851984, 81151927, 54181438, 656137, 33231715, 24730756, 62338977]

14

1073617381268b4038ed360ed3a2a1a5

[24729290, 8722685, 8851984, 81151927, 648373, 33231715, 62338977]

14

107539f0ee4e09f1991022b1d6d46b26

[48690285, 8722685, 648390, 8851984, 81151927, 24729427, 84266535]

7

107dd93d16b034e1b72400cc15ba7044

[48689131, 8722685, 8851984, 24729358, 33231715, 62338977, 649156]

7

109e24801cd3b0ce171f2a5602019336

[8722685, 8851984, 653124, 81151927, 33231715, 24729787, 62338977]

7

109fd8733cff78659406d5af2ff113b1

[77803299, 651769, 8722685, 81151927, 33231715, 8851891, 24729464]

7

10ad7cac86a8b1d01eb11593949783f2

[77803299, 24729385, 79628485, 33231715, 8851891, 651096, 2296544]

7

10b3fa41ed85ac328ff5cb301d610c36

[77803299, 48689131, 8722685, 677516, 33231715, 24742362, 8851891]

7

10b7c900ec9dc9c7af0dfe18ca9d1b0e

[77803299, 24729385, 8722685, 81151927, 84266535, 8851891, 651096]

21

10b84c0d9e330cae948172dcd66bcdde

[77803299, 649331, 8722685, 81151927, 84266535, 8851891, 24729569]

21

10ba5a01b88f640f9da716a12018e3fa

[81053882, 34922299, 8722685, 24729414, 8852078, 84266535, 649329]

14

10cb78cfd40bc4e56acb2dfa220ba849

[77803299, 8722685, 81151927, 33231715, 8851891, 649137, 24729438]

14

10e6ac1898c4082d6c597662068c162

[77803299, 48689131, 85012211, 648895, 8722685, 24729349, 84008342, 8851891]

14

10f62f9c556d5552aeb2b72695978e1f

[77394070, 649331, 8722685, 8851984, 33231715, 62338414, 24729569]

7

Showing the first 1000 rows.

[Cmd 16](https://community.cloud.databricks.com/?o=238865118750664#notebook/1951586186020076/command/1951586186020092)



# Doing same as above, but using spark sql, however the query is same.

# The advanatge is we can do any other operation on dataframe after we anlysys as like saving data in to a file etc

query = """SELECT id.val as customer\_id, collect\_set(indi\_segment\_id) as related\_customer\_segments\_ids, count(indi\_segment\_id) count\_related\_customer\_segments\_ids

FROM view\_customer\_segemnt \

LATERAL VIEW explode(event.add) t\_segment AS indi\_segment\_id \

GROUP BY id"""

​

res2\_df = spark.sql(query)

​

res2\_df.show()

​

res2\_df.write.mode("overwrite").format("json").json("/FileStore/tables/customers\_segment")

https://community.cloud.databricks.com/media/ellipse-2s-30px.d5e7584e.gifWaiting for cluster to start

res2\_df:pyspark.sql.dataframe.DataFrame = [customer\_id: string, related\_customer\_segments\_ids: array ... 1 more fields]

+--------------------+-----------------------------+-----------------------------------+ | customer\_id|related\_customer\_segments\_ids|count\_related\_customer\_segments\_ids| +--------------------+-----------------------------+-----------------------------------+ |1007bd3329ed45b20...| [77803299, 247296...| 7| |100934fa47c81a0df...| [77803299, 872268...| 7| |1021406292b3c7080...| [8722685, 1419823...| 7| |1027d8053008ea354...| [77803299, 872268...| 5| |1052ae341da6042ab...| [8722685, 8851984...| 14| |1073617381268b403...| [24729290, 872268...| 14| |107539f0ee4e09f19...| [48690285, 872268...| 7| |107dd93d16b034e1b...| [48689131, 872268...| 7| |109e24801cd3b0ce1...| [8722685, 8851984...| 7| |109fd8733cff78659...| [77803299, 651769...| 7| |10ad7cac86a8b1d01...| [77803299, 247293...| 7| |10b3fa41ed85ac328...| [77803299, 486891...| 7| |10b7c900ec9dc9c7a...| [77803299, 247293...| 21| |10b84c0d9e330cae9...| [77803299, 649331...| 21| |10ba5a01b88f640f9...| [81053882, 349222...| 14| |10cb78cfd40bc4e56...| [77803299, 872268...| 14| |10e6ac1898c4082d6...| [77803299, 486891...| 14| |10f62f9c556d5552a...| [77394070, 649331...| 7| |10f7039b963fc5a0c...| [79628485, 649131...| 7| |10f89eda2d6bdf531...| [24729385, 349230...| 14| +--------------------+-----------------------------+-----------------------------------+ only showing top 20 rows

Shift+Enter to run    shortcuts