Muhammad_Dastgir

November 4, 2024

1 Mid-Term Project

Overview

- Use Medicare CCLF Claims from Syntegra dataset to answer key business questions
- Extra credit for building up on the questions below (additional questions + answers)
- One Jupyter notebook solution with clear Python code and all cell outputs available
- At least two data quality checks

1.1 Step 0. Prepare raw input datasets

Here we will 1) load original datasets, 2) remove unused columns, 3) de-duplicate rows, and 4) join datasets, not necessarily in this order

Assumptions: - Claim ID (cur_clm_uniq_id) represents one claim, which may or may not have more than one code (code could be HCPCS/CPT, diagnosis, procedure…) - There is a one-to-many relationship between patient IDs (bene_mbi_id) and claim IDs (cur_clm_uniq_id), i.e. each claim is unique to one patient, but one patient can have more than one claim

1.1.1 0.1 Import required packages

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
```

[2]:
```
# Turn off the automatic setting that redacts the columns/rows from the␣
 ↪dataframe output
pd.set_option('display.max_columns', None)
#pd.set_option('display.max_rows', 200)
```

1.1.2 0.2 Load & select columns to be used from raw (original) datasets

1.1.3 0.2.1 Load & select columns from Claims Header dataset

[3]:
```
# Load Claims Header dataset
parta_claims_header_raw_df = pd.read_csv("/Users/hamiddastgir/Library/
 ↪CloudStorage/Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/
 ↪Syntegra Datasets Files/parta_claims_header.csv")
parta_claims_header_raw_df.sort_values(by=['cur_clm_uniq_id'])
```

[3]:

	cur_clm_uniq_id	prvdr_oscar_num	bene_mbi_id	bene_hic_num	\
510	100190	111821	1228	NaN	
521	100402	100226	1261	NaN	
525	100464	360051	12978	NaN	
536	100698	140276	11789	NaN	
540	100750	230216	12138	NaN	
…	…	…	…	…	
230	1698691	390145	10007	NaN	
4365	1698722	200021	10985	NaN	
4366	1698935	210022	1297	NaN	
4367	1699005	100057	12194	NaN	
4368	1699102	330191	11842	NaN	

	clm_type_cd	clm_from_dt	clm_thru_dt	clm_bill_fac_type_cd	\
510	40	2018-06-10	2018-06-10	7	
521	60	2017-05-27	2017-06-02	1	
525	40	2017-06-26	2017-06-26	1	
536	40	2017-07-28	2017-07-28	1	
540	40	2018-01-13	2018-01-13	1	
…	…	…	…	…	
230	40	2016-12-11	2016-12-11	1	
4365	40	2018-06-16	2018-06-16	1	
4366	40	2018-04-06	2018-04-06	1	
4367	40	2016-04-27	2016-04-27	1	
4368	60	2017-01-30	2017-02-03	1	

	clm_bill_clsfctn_cd	prncpl_dgns_cd	admtg_dgns_cd	clm_mdcr_npmt_rsn_cd	\
510	7	M1611	NaN	NaN	
521	1	K5733	K5733	NaN	
525	3	R079	NaN	NaN	
536	3	M545	NaN	NaN	
540	3	Z0289	NaN	N	
…	…	…	…	…	
230	3	Z01818	NaN	NaN	
4365	3	E782	NaN	NaN	
4366	3	I110	NaN	NaN	
4367	3	I348	NaN	NaN	
4368	1	I441	R42	NaN	

	clm_pmt_amt	clm_nch_prmry_pyr_cd	prvdr_fac_fips_st_cd	\
510	127.79	NaN	11	

```
521      10602.46                  NaN                    10
525        199.45                  NaN                    36
536         85.25                  NaN                    14
540          0.00                  NaN                    23
…             …                    …                      …
230         43.01                  NaN                    39
4365       179.09                  NaN                    20
4366       400.81                  NaN                    21
4367       265.19                  NaN                    10
4368      6476.96                  NaN                    33

      bene_ptnt_stus_cd  dgns_drg_cd clm_op_srvc_type_cd  fac_prvdr_npi_num  \
510                   1          NaN                   F         1780608992
521                   6        330.0                 NaN         1689611501
525                   1          NaN                   C         1073688354
536                   1          NaN                   C         1376521575
540                   9          NaN                   C         1982685384
…                   …            …                     …                  …
230                   1          NaN                   C         1689691214
4365                  1          NaN                   C         1932164795
4366                  9          NaN                   C         1205896446
4367                  1          NaN                   C         1821019571
4368                  3        287.0                 NaN         1871606764

      oprtg_prvdr_npi_num  atndg_prvdr_npi_num  othr_prvdr_npi_num  \
510                   NaN         1.972732e+09                 NaN
521                   NaN         1.285688e+09                 NaN
525                   NaN         1.982693e+09                 NaN
536                   NaN         1.912991e+09                 NaN
540                   NaN         1.063442e+09                 NaN
…                     …                    …                   …
230                   NaN         1.679505e+09                 NaN
4365                  NaN         1.548289e+09                 NaN
4366                  NaN         1.922016e+09                 NaN
4367                  NaN         1.437130e+09                 NaN
4368                  NaN         1.679594e+09                 NaN

      clm_adjsmt_type_cd  clm_efctv_dt  clm_idr_ld_dt  \
510                  NaN           NaN            NaN
521                  NaN           NaN            NaN
525                  NaN           NaN            NaN
536                  NaN           NaN            NaN
540                  NaN           NaN            NaN
…                    …             …              …
230                  NaN           NaN            NaN
4365                 NaN           NaN            NaN
4366                 NaN           NaN            NaN
```

```
4367                 NaN           NaN            NaN
4368                 NaN           NaN            NaN

      bene_eqtbl_bic_hicn_num  clm_admsn_type_cd clm_admsn_src_cd  \
510                       NaN                NaN              NaN
521                       NaN                1.0                1
525                       NaN                NaN              NaN
536                       NaN                NaN              NaN
540                       NaN                NaN              NaN
…                         …                  …                …
230                       NaN                NaN              NaN
4365                      NaN                NaN              NaN
4366                      NaN                NaN              NaN
4367                      NaN                NaN              NaN
4368                      NaN                1.0                1

     clm_bill_freq_cd  clm_query_cd  dgns_prcdr_icd_ind  \
510                 1             3                   0
521                 1             3                   0
525                 1             3                   0
536                 1             3                   0
540                 0             3                   0
…               …           …                 …
230                 1             3                   0
4365                1             3                   0
4366                1             3                   0
4367                1             3                   0
4368                1             3                   0

      clm_mdcr_instnl_tot_chrg_amt  clm_mdcr_ip_pps_cptl_ime_amt  \
510                         415.80                           NaN
521                       70795.63                        609.13
525                        2709.80                           NaN
536                         115.00                           NaN
540                         226.00                           NaN
…                             …                            …
230                         235.00                           NaN
4365                       1939.35                           NaN
4366                        554.00                           NaN
4367                       8423.00                           NaN
4368                      17897.91                        467.56

      clm_oprtnl_ime_amt  clm_mdcr_ip_pps_dsprprtnt_amt  \
510                  NaN                            NaN
521                 0.00                          13.92
525                  NaN                            NaN
536                  NaN                            NaN
```

```
540                   NaN                                NaN
…                     …                                  …
230                   NaN                                NaN
4365                  NaN                                NaN
4366                  NaN                                NaN
4367                  NaN                                NaN
4368                20.53                              39.19

      clm_hipps_uncompd_care_amt  clm_oprtnl_dsprtnt_amt
510                          NaN                     NaN
521                       231.15                     NaN
525                          NaN                     NaN
536                          NaN                     NaN
540                          NaN                     NaN
…                            …                       …
230                          NaN                     NaN
4365                         NaN                     NaN
4366                         NaN                     NaN
4367                         NaN                     NaN
4368                      669.18                     NaN

[8626 rows x 37 columns]
```

Data Quality Check #1: If true, the original dataset was unique on claim ID

[4]:
```
parta_claims_header_raw_df_count = parta_claims_header_raw_df.shape[0]
parta_claims_header_raw_uniq_clm_id_df = parta_claims_header_raw_df['cur_clm_uniq_id'].drop_duplicates()

parta_claims_header_raw_df_count == parta_claims_header_raw_uniq_clm_id_df.shape[0]
```

[4]:
```
True
```

[5]:
```
# Select only the desired columns (renaming if needed) and remove duplicates if any
parta_claims_header_df = parta_claims_header_raw_df[[
    'cur_clm_uniq_id', 'bene_mbi_id', 'atndg_prvdr_npi_num',
    'clm_from_dt', 'prncpl_dgns_cd', 'clm_pmt_amt'
]].drop_duplicates().rename(
    columns={
        'cur_clm_uniq_id': 'claim_id',
        'bene_mbi_id': 'patient_id',
        'clm_from_dt': 'claim_date',
        'atndg_prvdr_npi_num': 'npi_id'
    }
)
parta_claims_header_df
```

[5]:
```
      claim_id  patient_id        npi_id  claim_date prncpl_dgns_cd  \
0      1001595       10226  1.366492e+09  2018-02-28         M25551
1      1004555       10133  1.942275e+09  2018-11-02          Z9861
2      1011605       10163  1.578546e+09  2018-01-02           C439
3      1011758        1003  1.952368e+09  2018-06-12           R310
4       101424       10052  1.336125e+09  2016-04-13           L821
…         …          …            …          …              …
8621    999774       10367           NaN  2017-11-06           R072
8622    999808       10496  1.740225e+09  2017-07-19           R079
8623    999878       12160  1.497784e+09  2018-01-18          R5383
8624    999961       12090  1.083691e+09  2018-03-10            C73
8625    999976       10768  1.770564e+09  2016-01-18           E785

      clm_pmt_amt
0          259.01
1           29.56
2           45.88
3            9.40
4           34.18
…           …
8621       374.08
8622       360.89
8623        90.73
8624       329.44
8625        25.60

[8626 rows x 6 columns]
```

Data Quality Check #2: If true, the filtered dataset did not have any duplicates

[6]:
```
parta_claims_header_df_count = parta_claims_header_df.shape[0]

parta_claims_header_raw_df_count == parta_claims_header_df_count
```

[6]:
```
True
```

Data Quality Check #3: If the resulting dataframe is empty, it means all the records have diagnosis code (if it's not empty it should be removed now since we want only the ones with valid codes for analysis)

[7]:
```
parta_claims_header_df.loc[~parta_claims_header_df.prncpl_dgns_cd.notnull()]
```

[7]:
```
Empty DataFrame
Columns: [claim_id, patient_id, npi_id, claim_date, prncpl_dgns_cd, clm_pmt_amt]
Index: []
```

1.1.4 0.2.2 Load & select columns from Claims Revenue Center dataset

[8]:
```
# Load Claims Revenue Center dataset
# Note this dataset has more than one record for each claim ID (cur_clm_uniq_id)
# Also note there are two sets of date columns,
#  one for claim ID (clm_from/thru_dt) and one for claim line (clm_line_from/
 ↪thru_dt)
parta_claims_revenue_center_detail_raw_df = pd.read_csv(
    "/Users/hamiddastgir/Library/CloudStorage/Dropbox/Semester 3/BIA 810 -␣
 ↪Healthcare Analytics/Mid Term/Syntegra Datasets Files/
 ↪parta_claims_revenue_center_detail.csv"
)
parta_claims_revenue_center_detail_raw_df.sort_values(by=['cur_clm_uniq_id'])
```

```
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/3269972768.py:5
: DtypeWarning: Columns (19) have mixed types. Specify dtype option on import or
set low_memory=False.
  parta_claims_revenue_center_detail_raw_df = pd.read_csv(
```

[8]:

	cur_clm_uniq_id	clm_line_num	bene_mbi_id	bene_hic_num	clm_type_cd	\
318	100073	1	12620	NaN	40	
383	100184	1	10080	NaN	40	
384	100190	1	1228	NaN	40	
385	100190	2	1228	NaN	40	
386	100190	3	1228	NaN	40	
…	…	…	…	…	…	
29896	1699197	2	1177	NaN	40	
29898	1699212	1	1262	NaN	60	
29901	1699236	3	10580	NaN	40	
29899	1699236	1	10580	NaN	40	
29900	1699236	2	10580	NaN	40	

	clm_line_from_dt	clm_line_thru_dt	clm_line_prod_rev_ctr_cd	\
318	2018-12-02 00:00:00	2018-12-02 00:00:00	403	
383	2018-09-06 00:00:00	2018-09-06 00:00:00	1	
384	2018-06-10 00:00:00	2018-06-10 00:00:00	521	
385	2018-06-10 00:00:00	2018-06-10 00:00:00	521	
386	2018-06-10 00:00:00	2018-06-10 00:00:00	521	
…	…	…	…	
29896	2016-05-22 00:00:00	2016-05-22 00:00:00	302	
29898	2018-12-24 00:00:00	2018-12-25 00:00:00	730	
29901	2017-09-20 00:00:00	2017-09-20 00:00:00	370	
29899	2017-09-20 00:00:00	2017-09-20 00:00:00	258	
29900	2017-09-20 00:00:00	2017-09-20 00:00:00	360	

	clm_line_instnl_rev_ctr_dt	clm_line_hcpcs_cd	bene_eqtbl_bic_hicn_num	\
318	2018-12-02 00:00:00	77063	NaN	
383	NaN	NaN	NaN	

```
384          2018-06-10 00:00:00             G0467                      NaN
385          2018-06-10 00:00:00             98960                      NaN
386          2018-06-10 00:00:00             J1100                      NaN
…                            …                …                          …
29896        2016-05-22 00:00:00             86592                      NaN
29898                        NaN               NaN                      NaN
29901        2017-09-20 00:00:00               NaN                      NaN
29899        2017-09-20 00:00:00               NaN                      NaN
29900        2017-09-20 00:00:00             45385                      NaN

       prvdr_oscar_num          clm_from_dt          clm_thru_dt  \
318                NaN  2018-12-02 00:00:00  2018-12-02 00:00:00
383                NaN  2018-09-06 00:00:00  2018-09-06 00:00:00
384                NaN  2018-06-10 00:00:00  2018-06-10 00:00:00
385                NaN  2018-06-10 00:00:00  2018-06-10 00:00:00
386                NaN  2018-06-10 00:00:00  2018-06-10 00:00:00
…                  …                    …                    …
29896              NaN  2016-05-22 00:00:00  2016-05-22 00:00:00
29898              NaN  2018-12-24 00:00:00  2018-12-25 00:00:00
29901              NaN  2017-09-20 00:00:00  2017-09-20 00:00:00
29899              NaN  2017-09-20 00:00:00  2017-09-20 00:00:00
29900              NaN  2017-09-20 00:00:00  2017-09-20 00:00:00

       clm_line_srvc_unit_qty  clm_line_cvrd_pd_amt hcpcs_1_mdfr_cd  \
318                         1                 24.11             NaN
383                         0                  0.00             NaN
384                         1                133.74             NaN
385                         1                  0.00             NaN
386                         4                  0.00             NaN
…                         …                     …               …
29896                       1                  5.43             NaN
29898                       1                   NaN             NaN
29901                       2                  0.00             NaN
29899                       1                  0.00             NaN
29900                       1                543.04             NaN

      hcpcs_2_mdfr_cd hcpcs_3_mdfr_cd hcpcs_4_mdfr_cd  hcpcs_5_mdfr_cd  \
318               NaN             NaN             NaN              NaN
383               NaN             NaN             NaN              NaN
384               NaN             NaN             NaN              NaN
385               NaN             NaN             NaN              NaN
386               NaN             NaN             NaN              NaN
…                 …               …               …                …
29896             NaN             NaN             NaN              NaN
29898             NaN             NaN             NaN              NaN
29901             NaN             NaN             NaN              NaN
29899             NaN             NaN             NaN              NaN
```

```
29900             NaN             NaN             NaN             NaN

      clm_rev_apc_hipps_cd
318                  00000
383                  00000
384                  00000
385                  00000
386                  00000
…                    …
29896                00000
29898                00000
29901                00000
29899                00000
29900                05312

[59419 rows x 22 columns]
```

Data Quality Check #4: If the resulting dataframe is empty, it means there is no difference between columns 'clm_line_from_dt' and 'clm_from_dt' for all the rows

[9]:
```
parta_claims_revenue_center_detail_raw_df.loc[
    ~(parta_claims_revenue_center_detail_raw_df['clm_line_from_dt']
      == parta_claims_revenue_center_detail_raw_df['clm_from_dt'])
]
```

[9]:
```
Empty DataFrame
Columns: [cur_clm_uniq_id, clm_line_num, bene_mbi_id, bene_hic_num, clm_type_cd,
clm_line_from_dt, clm_line_thru_dt, clm_line_prod_rev_ctr_cd,
clm_line_instnl_rev_ctr_dt, clm_line_hcpcs_cd, bene_eqtbl_bic_hicn_num,
prvdr_oscar_num, clm_from_dt, clm_thru_dt, clm_line_srvc_unit_qty,
clm_line_cvrd_pd_amt, hcpcs_1_mdfr_cd, hcpcs_2_mdfr_cd, hcpcs_3_mdfr_cd,
hcpcs_4_mdfr_cd, hcpcs_5_mdfr_cd, clm_rev_apc_hipps_cd]
Index: []
```

[10]:
```
# Select only the desired columns (renaming if needed) and remove duplicates if␣
 ↪any
# Select 'clm_from_dt' as the column for claim dates since we want uniqueness␣
 ↪on claim ID, not claim line
parta_claims_revenue_center_detail_df =␣
 ↪parta_claims_revenue_center_detail_raw_df[[
    'cur_clm_uniq_id', 'bene_mbi_id', 'clm_from_dt',
    'clm_line_hcpcs_cd', 'clm_line_cvrd_pd_amt'
]].drop_duplicates().rename(
    columns={
        'cur_clm_uniq_id': 'claim_id',
        'bene_mbi_id': 'patient_id',
        'clm_from_dt': 'claim_date',
        'clm_line_hcpcs_cd': 'hcpcs_code'
```

```
    }
)
parta_claims_revenue_center_detail_df
```

[10]:

	claim_id	patient_id	claim_date	hcpcs_code	clm_line_cvrd_pd_amt
0	1001122	10081	2018-05-30 00:00:00	NaN	0.00
1	1001595	10226	2018-02-28 00:00:00	G0283	9.67
7	1001595	10226	2018-02-28 00:00:00	G8978	0.00
8	1001595	10226	2018-02-28 00:00:00	G8979	0.00
10	1001595	10226	2018-02-28 00:00:00	97110	24.97
…	…	…	…	…	…
59414	999961	12090	2018-03-10 00:00:00	A9516	0.00
59415	999961	12090	2018-03-10 00:00:00	G8996	0.00
59416	999976	10768	2016-01-18 00:00:00	80053	11.37
59417	999976	10768	2016-01-18 00:00:00	80061	12.83
59418	999976	10768	2016-01-18 00:00:00	NaN	0.00

[46823 rows x 5 columns]

Data Quality Check #5: If the resulting dataframe is empty, it means all the records have HCPCS code (if it's not empty it should be removed now since we want only the ones with valid codes for analysis)

[11]:
```
parta_claims_revenue_center_detail_df.loc[
    ~parta_claims_revenue_center_detail_df.hcpcs_code.notnull()
]
```

[11]:

	claim_id	patient_id	claim_date	hcpcs_code
0	1001122	10081	2018-05-30 00:00:00	NaN
25	1001595	10226	2018-02-28 00:00:00	NaN
29	1004555	10133	2018-11-02 00:00:00	NaN
30	1004904	10106	2018-02-26 00:00:00	NaN
32	100974	10042	2017-02-20 00:00:00	NaN
…	…	…	…	…
59369	999008	12473	2018-08-04 00:00:00	NaN

```
59402    999774       10367  2017-11-06 00:00:00        NaN
59407    999808       10496  2017-07-19 00:00:00        NaN
59410    999943       11021  2016-11-20 00:00:00        NaN
59418    999976       10768  2016-01-18 00:00:00        NaN

       clm_line_cvrd_pd_amt
0                       0.0
25                      0.0
29                      0.0
30                      0.0
32                      0.0
…                       …
59369                   0.0
59402                   0.0
59407                   0.0
59410                   0.0
59418                   0.0

[10799 rows x 5 columns]
```

[12]:
```
# Data Quality Check #5 failed, so remove rows with no HCPCS codes
parta_claims_revenue_center_detail_df = parta_claims_revenue_center_detail_df.
  ↪loc[
    parta_claims_revenue_center_detail_df.hcpcs_code.notnull()
]
```

[14]:
```
# Update date format for claim dates to match that of Claims Header dataset for␣
  ↪easy join
parta_claims_revenue_center_detail_df['claim_date'] = pd.to_datetime(
    parta_claims_revenue_center_detail_df['claim_date']
).dt.strftime('%Y-%m-%d')
parta_claims_revenue_center_detail_df
```

```
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/2671745514.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  parta_claims_revenue_center_detail_df['claim_date'] = pd.to_datetime(
```

[14]:
```
        claim_id  patient_id  claim_date hcpcs_code  clm_line_cvrd_pd_amt
1       1001595       10226  2018-02-28      G0283                  9.67
7       1001595       10226  2018-02-28      G8978                  0.00
8       1001595       10226  2018-02-28      G8979                  0.00
10      1001595       10226  2018-02-28      97110                 24.97
```

```
18      1001595       10226  2018-02-28      97140                 20.33
…         …           …        …         …                   …
59413    999961       12090  2018-03-10      78014                400.05
59414    999961       12090  2018-03-10      A9516                  0.00
59415    999961       12090  2018-03-10      G8996                  0.00
59416    999976       10768  2016-01-18      80053                 11.37
59417    999976       10768  2016-01-18      80061                 12.83

[36024 rows x 5 columns]
```

Mini-Analysis #1: Find whether there are matching claims between Claim Header and Claims Revenue Center datasets

***

[15]:
```
claims_header_unique_claims_df = parta_claims_header_df[[
    'claim_id'
]].drop_duplicates()

claims_header_unique_claims_df['header'] = 1

revenue_center_unique_claims_df = parta_claims_revenue_center_detail_df[[
    'claim_id'
]].drop_duplicates()

revenue_center_unique_claims_df['revenue'] = 1

joined_df1 = pd.merge(
    claims_header_unique_claims_df,
    revenue_center_unique_claims_df,
    on='claim_id', how = 'outer'
)
joined_df1
```

[15]:
```
       claim_id  header  revenue
0       1001595     1.0      1.0
1       1004555     1.0      1.0
2       1011605     1.0      1.0
3       1011758     1.0      1.0
4        101424     1.0      NaN
…          …      …        …
15721    999074     NaN      1.0
15722    999324     NaN      1.0
15723    999350     NaN      1.0
15724    999514     NaN      1.0
15725    999943     NaN      1.0

[15726 rows x 3 columns]
```

[16]:
```
print('# of unique claims in Claims Header dataset: '
      + str(claims_header_unique_claims_df.shape[0])
     )
print('# of unique claims in Claims Revenue Center dataset: '
      + str(revenue_center_unique_claims_df.shape[0])
     )
```

```
# of unique claims in Claims Header dataset: 8626
# of unique claims in Claims Revenue Center dataset: 13406
```

[17]:
```
print('# of unique claims in Claims Header and Claims Revenue Center datasets␣
 ↪combined: '
      + str(joined_df1.shape[0])
     )
print('From combined list of unique claims - ')
print('# of unique claims in only Claims Header dataset: '
      + str(joined_df1.loc[(joined_df1.header == 1) & ~(joined_df1.revenue ==␣
 ↪1)].shape[0])
     )
print('# of unique claims in only Claims Revenue Center dataset: '
      + str(joined_df1.loc[~(joined_df1.header == 1) & (joined_df1.revenue ==␣
 ↪1)].shape[0])
     )
print('# of unique claims in both Claims Header AND Claims Revenue Center␣
 ↪datasets: '
      + str(joined_df1.loc[(joined_df1.header == 1) & (joined_df1.revenue ==␣
 ↪1)].shape[0])
     )
```

```
# of unique claims in Claims Header and Claims Revenue Center datasets combined:
15726
From combined list of unique claims -
# of unique claims in only Claims Header dataset: 2320
# of unique claims in only Claims Revenue Center dataset: 7100
# of unique claims in both Claims Header AND Claims Revenue Center datasets:
6306
```

Conclusion: There are quite a number of claims available in both datasets, so join them on claim ID as an outer join to get all possible claims without duplicates

***

1.1.5 0.2.3 Load & select columns from Diagnosis dataset

[18]:
```
# Load the Diagnosis dataset
# Note that 'clm_from_dt' has some records with null values, but we need claim␣
 ↪dates for all claims
```

```
parta_diagnosis_code_raw_df = pd.read_csv("/Users/hamiddastgir/Library/
 ↪CloudStorage/Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/
 ↪Syntegra Datasets Files/parta_diagnosis_code.csv")
parta_diagnosis_code_raw_df.sort_values(by=['cur_clm_uniq_id',␣
 ↪'clm_val_sqnc_num'])
```

[18]:

```
       cur_clm_uniq_id  bene_mbi_id  bene_hic_num  clm_type_cd  \
244             100190         1228           NaN           40
243             100190         1228           NaN           40
246             100190         1228           NaN           40
245             100190         1228           NaN           40
366             100402         1261           NaN           60
…                  …            …             …            …
16232          1699102        11842           NaN           60
16236          1699137        10873           NaN           40
16235          1699137        10873           NaN           40
16237          1699155        11689           NaN           40
16238          1699155        11689           NaN           40

       clm_prod_type_cd  clm_val_sqnc_num clm_dgns_cd  \
244                 NaN                 1       M1611
243                 NaN                 2      M25572
246                 NaN                 3      M25551
245                 NaN                 4       M5136
366                 NaN                11        E119
…                   …                  …          …
16232               NaN                13       Z8673
16236               NaN                 1        N390
16235               NaN                 2        N390
16237               NaN                 3        K219
16238               NaN                 4        E039

       bene_eqtbl_bic_hicn_num  prvdr_oscar_num          clm_from_dt  \
244                        NaN              NaN                  NaN
243                        NaN              NaN                  NaN
246                        NaN              NaN                  NaN
245                        NaN              NaN                  NaN
366                        NaN         100256.0  2017-05-28 00:00:00
…                          …                …                    …
16232                      NaN         330191.0  2017-01-31 00:00:00
16236                      NaN              NaN                  NaN
16235                      NaN              NaN                  NaN
16237                      NaN              NaN                  NaN
16238                      NaN              NaN                  NaN

              clm_thru_dt clm_poa_ind  dgns_prcdr_icd_ind
244    2018-06-10 00:00:00         NaN                   0
```

```
243    2018-06-10 00:00:00         NaN                  0
246    2018-06-10 00:00:00         NaN                  0
245    2018-06-10 00:00:00         NaN                  0
366    2017-06-02 00:00:00           Y                  0
…                     …         …                    …
16232  2017-02-03 00:00:00           0                  0
16236  2018-07-12 00:00:00         NaN                  0
16235  2018-07-12 00:00:00         NaN                  0
16237  2018-12-06 00:00:00         NaN                  0
16238  2018-12-06 00:00:00         NaN                  0

[32052 rows x 13 columns]
```

[19]:
```
# Select only the desired columns (renaming if needed) and remove duplicates if any
# Use 'clm_thru_dt' as claim date columns since 'clm_from_dt' has some nulls
parta_diagnosis_code_df = parta_diagnosis_code_raw_df[[
    'cur_clm_uniq_id', 'bene_mbi_id', 'clm_thru_dt', 'clm_dgns_cd'
]].drop_duplicates().rename(
    columns={
        'cur_clm_uniq_id': 'claim_id',
        'bene_mbi_id': 'patient_id',
        'clm_thru_dt': 'claim_date'
    }
)
parta_diagnosis_code_df
```

[19]:
```
       claim_id  patient_id           claim_date clm_dgns_cd
0       1001122       10081  2018-05-30 00:00:00       K5289
1       1001595       10226  2018-02-28 00:00:00      M25551
2       1001595       10226  2018-02-28 00:00:00      M79604
3       1001865       10133  2018-09-14 00:00:00        G459
4       1004555       10133  2018-11-02 00:00:00       Z9861
…           …           …                    …          …
32047    999878       12160  2018-01-18 00:00:00        N390
32048    999943       11021  2016-11-20 00:00:00        M545
32049    999961       12090  2018-03-10 00:00:00         C73
32050    999962       11030  2018-07-17 00:00:00       G8194
32051    999976       10768  2016-01-18 00:00:00        E785

[30487 rows x 4 columns]
```

Data Quality Check #6: If the resulting dataframe is empty, it means all the records have values for 'clm_thru_dt' (if it's not empty it should be removed now since without claim dates it'd be difficult to use)

[20]:
```
parta_diagnosis_code_df.loc[~parta_diagnosis_code_df.claim_date.notnull()]
```

[20]:
```
Empty DataFrame
Columns: [claim_id, patient_id, claim_date, clm_dgns_cd]
Index: []
```

Data Quality Check #7: If the resulting dataframe is empty, it means all the records have diagnosis code (if it's not empty it should be removed now since we want only the ones with valid codes for analysis)

[21]:
```
parta_diagnosis_code_df.loc[~parta_diagnosis_code_df.clm_dgns_cd.notnull()]
```

[21]:
```
Empty DataFrame
Columns: [claim_id, patient_id, claim_date, clm_dgns_cd]
Index: []
```

[22]:
```
# Update date format for claim dates to match that of Claims Header dataset for easy join
parta_diagnosis_code_df['claim_date'] = pd.to_datetime(
    parta_diagnosis_code_df['claim_date']
).dt.strftime('%Y-%m-%d')
parta_diagnosis_code_df
```

[22]:

	claim_id	patient_id	claim_date	clm_dgns_cd
0	1001122	10081	2018-05-30	K5289
1	1001595	10226	2018-02-28	M25551
2	1001595	10226	2018-02-28	M79604
3	1001865	10133	2018-09-14	G459
4	1004555	10133	2018-11-02	Z9861
…	…	…	…	…
32047	999878	12160	2018-01-18	N390
32048	999943	11021	2016-11-20	M545
32049	999961	12090	2018-03-10	C73
32050	999962	11030	2018-07-17	G8194
32051	999976	10768	2016-01-18	E785

[30487 rows x 4 columns]

Mini-Analysis #2: Find whether there are matching claims between above two datasets and the Diagnosis dataset

***

[23]:
```
diagnosis_unique_claims_df = parta_diagnosis_code_df[[
    'claim_id'
]].drop_duplicates()

diagnosis_unique_claims_df['diagnosis'] = 1

joined_df2 = pd.merge(
```

```
    joined_df1,
    diagnosis_unique_claims_df,
    on='claim_id', how = 'outer'
)
joined_df2
```

[23]:

	claim_id	header	revenue	diagnosis
0	1001595	1.0	1.0	1.0
1	1004555	1.0	1.0	1.0
2	1011605	1.0	1.0	1.0
3	1011758	1.0	1.0	1.0
4	101424	1.0	NaN	NaN
…	…	…	…	…
19452	998726	NaN	NaN	1.0
19453	999064	NaN	NaN	1.0
19454	999766	NaN	NaN	1.0
19455	999799	NaN	NaN	1.0
19456	999962	NaN	NaN	1.0

[19457 rows x 4 columns]

[24]:
```
print('# of unique claims in Claims Header+Claims Revenue Center datasets: '
      + str(joined_df1.shape[0])
     )
print('# of unique claims in Diagnosis dataset: '
      + str(diagnosis_unique_claims_df.shape[0])
     )
```

```
# of unique claims in Claims Header+Claims Revenue Center datasets: 15726
# of unique claims in Diagnosis dataset: 13432
```

[25]:
```
print('# of unique claims in Claims Header+Claims Revenue Center and Diagnosis␣
 ↪datasets combined: '
      + str(joined_df2.shape[0])
     )
print('From combined list of unique claims - ')
print('# of unique claims only in either Claims Header or Claims Revenue Center␣
 ↪datasets: '
      + str(joined_df2.loc[
          ((joined_df2.header == 1) | (joined_df2.revenue == 1))
          & ~(joined_df2.diagnosis == 1)
      ].shape[0])
     )
print('# of unique claims in only Diagnosis dataset: '
      + str(joined_df2.loc[
          (~(joined_df2.header == 1) & ~(joined_df2.revenue == 1))
          & (joined_df2.diagnosis == 1)
```

```
      ].shape[0])
     )
print('# of unique claims in all three datasets: '
      + str(joined_df2.loc[
          (joined_df2.header == 1) & (joined_df2.revenue == 1) & (joined_df2.
 ↪diagnosis == 1)
      ].shape[0])
     )
```

```
# of unique claims in Claims Header+Claims Revenue Center and Diagnosis datasets
combined: 19457
From combined list of unique claims -
# of unique claims only in either Claims Header or Claims Revenue Center
datasets: 6025
# of unique claims in only Diagnosis dataset: 3731
# of unique claims in all three datasets: 5266
```

Conclusion: There are quite a number of claims available in all three datasets, so join diagnosis to the first two datasets on claim ID as an outer join to get all possible claims without duplicates

***

1.1.6 0.2.4 Load & select columns from Procedure dataset

[26]:
```
# Load the Procedure dataset
parta_procedure_code_df = pd.read_csv("/Users/hamiddastgir/Library/CloudStorage/
 ↪Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/Syntegra Datasets␣
 ↪Files/parta_procedure_code.csv")
parta_procedure_code_df
```

[26]:

	cur_clm_uniq_id	bene_mbi_id	bene_hic_num	clm_type_cd	\
0	100402	1261	NaN	60	
1	100402	1261	NaN	60	
2	100402	1261	NaN	60	
3	100402	1261	NaN	60	
4	1008371	1074	NaN	60	
..	…	…	…	…	
457	357821	10200	NaN	60	
458	357821	10200	NaN	60	
459	412998	10106	NaN	60	
460	460114	10133	NaN	60	
461	766818	10010	NaN	60	

	clm_val_sqnc_num	clm_prcdr_cd	clm_prcdr_prfrm_dt	\
0	1	0DJD8ZZ	2017-05-31 00:00:00	
1	2	0D9670Z	2017-05-29 00:00:00	
2	3	0DJD8ZZ	2017-06-01 00:00:00	
3	4	0DB78ZX	2017-05-30 00:00:00	

```
4                    1      0T9B7ZZ  2016-12-03 00:00:00
..                 …          …                  …
457                  2      4A023N7  2018-06-18 00:00:00
458                  1      4A023N7  2018-06-18 00:00:00
459                  1      0SRC0J9  2016-12-09 00:00:00
460                  1      0QSH04Z  2018-05-17 00:00:00
461                  1      B246ZZZ  2016-01-12 00:00:00

     bene_eqtbl_bic_hicn_num  prvdr_oscar_num          clm_from_dt  \
0                        NaN           100256  2017-05-28 00:00:00
1                        NaN           100256  2017-05-28 00:00:00
2                        NaN           100256  2017-05-28 00:00:00
3                        NaN           100256  2017-05-28 00:00:00
4                        NaN           140007  2016-12-02 00:00:00
..                       …              …                   …
457                      NaN           100258  2018-06-16 00:00:00
458                      NaN           100258  2018-06-16 00:00:00
459                      NaN           250104  2016-12-09 00:00:00
460                      NaN           150112  2018-05-07 00:00:00
461                      NaN           190263  2016-01-09 00:00:00

             clm_thru_dt  dgns_prcdr_icd_ind
0    2017-06-02 00:00:00                   0
1    2017-06-02 00:00:00                   0
2    2017-06-02 00:00:00                   0
3    2017-06-02 00:00:00                   0
4    2016-12-08 00:00:00                   0
..                   …                   …
457  2018-06-19 00:00:00                   0
458  2018-06-19 00:00:00                   0
459  2016-12-10 00:00:00                   0
460  2018-05-23 00:00:00                   0
461  2016-01-15 00:00:00                   0

[462 rows x 12 columns]
```

Conclusion: Don't join procedure dataset since the only useful info for sake of this analysis is the procedure codes and we won't be using them in our analysis

1.1.7 0.2.5 Load & select columns from DME dataset

[27]:
```
# Load the DME dataset
partb_dme_raw_df = pd.read_csv("/Users/hamiddastgir/Library/CloudStorage/
  ↪Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/Syntegra Datasets␣
  ↪Files/partb_dme.csv")
partb_dme_raw_df.sort_values(by='cur_clm_uniq_id')
```

[27]:

	cur_clm_uniq_id	clm_line_num	bene_mbi_id	bene_hic_num	clm_type_cd
267	100441	1	12064	NaN	82
268	100441	2	12064	NaN	82
269	100441	3	12064	NaN	82
270	100441	4	12064	NaN	82
271	100441	5	12064	NaN	82
…	…	…	…	…	…
1541	1696080	2	11689	NaN	82
129	1696545	1	10046	NaN	82
1548	1696792	1	12086	NaN	82
1549	1697987	1	11074	NaN	82
1550	1698182	1	12549	NaN	82

	clm_from_dt	clm_thru_dt	clm_fed_type_srvc_cd	clm_pos_cd
267	2016-10-10	2016-10-10	P	12
268	2016-10-10	2016-10-10	P	12
269	2016-10-10	2016-10-10	P	12
270	2016-10-10	2016-10-10	P	12
271	2016-10-10	2016-10-10	P	12
…	…	…	…	…
1541	2016-11-18	2016-11-18	P	12
129	2017-07-25	2017-07-25	R	12
1548	2018-12-23	2018-12-23	P	12
1549	2018-04-06	2018-04-06	P	12
1550	2016-02-19	2016-02-19	R	12

	clm_line_from_dt	clm_line_thru_dt	clm_line_hcpcs_cd
267	2016-10-10	2016-10-10	A4256
268	2016-10-10	2016-10-10	E0607
269	2016-10-10	2016-10-10	A4253
270	2016-10-10	2016-10-10	A4259
271	2016-10-10	2016-10-10	A4258
…	…	…	…
1541	2016-11-18	2016-11-18	A7038
129	2017-07-25	2017-07-25	E0570
1548	2018-12-23	2018-12-23	A4604
1549	2018-04-06	2018-04-06	A4253
1550	2016-02-19	2016-02-19	E0570

	clm_line_cvrd_pd_amt	clm_prmry_pyr_cd	payto_prvdr_npi_num
267	3.24	NaN	1972744431
268	59.80	NaN	1972744431
269	38.57	NaN	1972744431
270	4.20	NaN	1972744431
271	2.27	NaN	1972744431
…	…	…	…
1541	0.00	NaN	1376599084

```
129                   6.30               NaN           1346347374
1548                 35.58               NaN           1790823722
1549                 27.92               NaN           1902842065
1550                  0.00               NaN           1356586747

      ordrg_prvdr_npi_num  clm_carr_pmt_dnl_cd  clm_prcsg_ind_cd  \
267          1.407814e+09                    1               NaN
268          1.407814e+09                    1               NaN
269          1.407814e+09                    1               NaN
270          1.407814e+09                    1               NaN
271          1.407814e+09                    1               NaN
…                     …                    …                 …
1541         1.659342e+09                    0               NaN
129          1.952397e+09                    1               NaN
1548         1.518936e+09                    1               NaN
1549         1.750382e+09                    1               NaN
1550         1.336253e+09                    1               NaN

      clm_adjsmt_type_cd  clm_efctv_dt  clm_idr_ld_dt  clm_cntl_num  \
267                  NaN           NaN            NaN           NaN
268                  NaN           NaN            NaN           NaN
269                  NaN           NaN            NaN           NaN
270                  NaN           NaN            NaN           NaN
271                  NaN           NaN            NaN           NaN
…                    …             …              …             …
1541                 NaN           NaN            NaN           NaN
129                  NaN           NaN            NaN           NaN
1548                 NaN           NaN            NaN           NaN
1549                 NaN           NaN            NaN           NaN
1550                 NaN           NaN            NaN           NaN

      bene_eqtbl_bic_hicn_num  clm_line_alowd_chrg_amt  clm_disp_cd
267                       NaN                     3.98            1
268                       NaN                    76.54            1
269                       NaN                    49.92            1
270                       NaN                     4.80            1
271                       NaN                     2.84            1
…                         …                       …               …
1541                      NaN                     0.00            1
129                       NaN                     8.12            1
1548                      NaN                    48.17            1
1549                      NaN                    33.28            1
1550                      NaN                    14.49            1

[2775 rows x 25 columns]
```

[28]:
```
# Select only the desired columns (rename columns if needed) and remove␣
 ↪duplicates if any
partb_dme_df = partb_dme_raw_df[[
    'cur_clm_uniq_id', 'bene_mbi_id', 'ordrg_prvdr_npi_num',
    'clm_from_dt', 'clm_line_hcpcs_cd', 'clm_line_alowd_chrg_amt'
]].drop_duplicates().rename(
    columns={
        'cur_clm_uniq_id': 'claim_id',
        'bene_mbi_id': 'patient_id',
        'ordrg_prvdr_npi_num': 'npi_id',
        'clm_from_dt': 'claim_date',
        'clm_line_hcpcs_cd': 'hcpcs_code',
        'clm_line_alowd_chrg_amt': 'claim_cost'
    }
)
partb_dme_df
```

[28]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	claim_cost
0	1004024	10202	1.841430e+09	2016-07-18	E0601	41.91
1	1034063	10137	1.669460e+09	2016-04-22	E0601	62.46
2	1046877	10202	1.093713e+09	2016-02-03	E0601	29.31
3	1072934	10202	1.285602e+09	2016-08-15	E0601	27.82
4	1082554	10174	1.003895e+09	2016-08-30	E0431	18.75
…	…	…	…	…	…	…
2770	998097	10396	1.891706e+09	2016-12-06	A4253	69.69
2771	999226	1095	1.518066e+09	2017-12-28	A4256	3.68
2772	999226	1095	1.518066e+09	2017-12-28	A4253	49.92
2773	999226	1095	1.518066e+09	2017-12-28	A4259	4.26
2774	999929	10261	1.497738e+09	2018-06-12	E0570	3.00

[2731 rows x 6 columns]

Data Quality Check #8: If the resulting dataframe is empty, it means all the records have HCPCS code (if it's not empty it should be removed now since we want only the ones with valid codes for analysis)

[29]:
```
partb_dme_df.loc[~partb_dme_df.hcpcs_code.notnull()]
```

[29]:
```
Empty DataFrame
Columns: [claim_id, patient_id, npi_id, claim_date, hcpcs_code, claim_cost]
Index: []
```

Mini-Analysis #3: Find whether there are matching claims between the first three datasets above and the DME dataset

***

[30]:
```
dme_unique_claims_df = partb_dme_df[[
    'claim_id'
]].drop_duplicates()

dme_unique_claims_df['dme'] = 1

joined_df3 = pd.merge(
    joined_df2,
    dme_unique_claims_df,
    on='claim_id', how = 'outer'
)
joined_df3
```

[30]:

	claim_id	header	revenue	diagnosis	dme
0	1001595	1.0	1.0	1.0	NaN
1	1004555	1.0	1.0	1.0	NaN
2	1011605	1.0	1.0	1.0	NaN
3	1011758	1.0	1.0	1.0	NaN
4	101424	1.0	NaN	NaN	NaN
…	…	…	…	…	…
20960	994844	NaN	NaN	NaN	1.0
20961	994885	NaN	NaN	NaN	1.0
20962	998097	NaN	NaN	NaN	1.0
20963	999226	NaN	NaN	NaN	1.0
20964	999929	NaN	NaN	NaN	1.0

[20965 rows x 5 columns]

[31]:
```
print('# of unique claims in first three datasets: '
      + str(joined_df2.shape[0])
     )
print('# of unique claims in DME dataset: '
      + str(dme_unique_claims_df.shape[0])
     )
```

```
# of unique claims in first three datasets: 19457
# of unique claims in DME dataset: 1508
```

[32]:
```
print('# of unique claims in the four datasets combined: '
      + str(joined_df3.shape[0])
     )
print('From combined list of unique claims - ')
print('# of unique claims in only the first three datasets: '
      + str(joined_df3.loc[
          ((joined_df3.header == 1)
          | (joined_df3.revenue == 1)
          | (joined_df3.diagnosis == 1))
          & ~(joined_df3.dme == 1)
```

```
      ].shape[0])
     )
print('# of unique claims in only DME dataset: '
      + str(joined_df3.loc[
          ~(joined_df3.header == 1)
          & ~(joined_df3.revenue == 1)
          & ~(joined_df3.diagnosis == 1)
          & (joined_df3.dme == 1)
      ].shape[0])
     )
print('# of unique claims in all four datasets: '
      + str(joined_df3.loc[
          (joined_df3.header == 1)
          & (joined_df3.revenue == 1)
          & (joined_df3.diagnosis == 1)
          & (joined_df3.dme == 1)
      ].shape[0])
     )
print('# of unique claims in DME and any of the first three datasets: '
      + str(joined_df3.loc[
          ((joined_df3.header == 1)
          | (joined_df3.revenue == 1)
          | (joined_df3.diagnosis == 1))
          & (joined_df3.dme == 1)
      ].shape[0])
     )
```

```
# of unique claims in the four datasets combined: 20965
From combined list of unique claims -
# of unique claims in only the first three datasets: 19457
# of unique claims in only DME dataset: 1508
# of unique claims in all four datasets: 0
# of unique claims in DME and any of the first three datasets: 0
```

Conclusion: None of the claims from DME is in any of the first three datasets, so append them to the output after joining the first three datasets

***

1.1.8 0.2.6 Load & select columns from Physicians dataset

[33]:
```
# Load the Physicians dataset
partb_physicians_raw_df = pd.read_csv("/Users/hamiddastgir/Library/CloudStorage/
  ↪Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/Syntegra Datasets␣
  ↪Files/partb_physicians.csv")
partb_physicians_raw_df.sort_values(by='cur_clm_uniq_id')
```

[33]:

	cur_clm_uniq_id	clm_line_num	bene_mbi_id	bene_hic_num	clm_type_cd
520	100020	1	1070	NaN	71
525	100024	1	11654	NaN	71
529	100030	1	12052	NaN	71
555	100038	1	12345	NaN	71
592	100061	1	10252	NaN	71
…	…	…	…	…	…
5485	1699176	1	1008	NaN	71
66051	1699182	1	13175	NaN	71
66052	1699186	1	10710	NaN	71
66053	1699204	1	11540	NaN	71
66054	1699222	1	11556	NaN	71

	clm_from_dt	clm_thru_dt	rndrg_prvdr_type_cd	rndrg_prvdr_fips_st_cd
520	2016-10-04	2016-10-04	5	36
525	2016-12-10	2016-12-10	1	39
529	2017-04-15	2017-04-15	1	5
555	2018-07-02	2018-07-02	1	34
592	2016-07-04	2016-07-04	1	33
…	…	…	…	…
5485	2018-10-18	2018-10-18	1	18
66051	2016-11-21	2016-11-21	5	31
66052	2016-01-18	2016-01-18	1	14
66053	2018-05-08	2018-05-08	1	28
66054	2016-03-16	2016-03-16	1	33

	clm_prvdr_spclty_cd	clm_fed_type_srvc_cd	clm_pos_cd	clm_line_from_dt
520	69	5	81	2016-10-04
525	26	T	11	2016-12-10
529	06	5	21	2017-04-15
555	30	4	19	2018-07-02
592	48	1	11	2016-07-04
…	…	…	…	…
5485	29	1	21	2018-10-18
66051	69	5	81	2016-11-21
66052	30	4	23	2016-01-18
66053	13	1	11	2018-05-08
66054	94	1	11	2016-03-16

	clm_line_thru_dt	clm_line_hcpcs_cd	clm_line_cvrd_pd_amt
520	2016-10-04	85610	5.10
525	2016-12-10	90834	61.17
529	2017-04-15	93010	6.92
555	2018-07-02	72158	89.30
592	2016-07-04	99213	65.83
…	…	…	…
5485	2018-10-18	99232	56.62

```
66051       2016-11-21             80053                  7.43
66052       2016-01-18             73110                  7.41
66053       2018-05-08             99214                 80.42
66054       2016-03-16             J7060                  0.00

      clm_line_prmry_pyr_cd clm_line_dgns_cd  clm_rndrg_prvdr_tax_num  \
520                     NaN             I482                      NaN
525                     NaN             F319                      NaN
529                     NaN             R001                      NaN
555                     NaN           M47816                      NaN
592                     NaN           L03032                      NaN
…                       …               …                         …
5485                    NaN            J9601                      NaN
66051                   NaN             E782                      NaN
66052                   NaN          S52502A                      NaN
66053                   NaN            M5116                      NaN
66054                     G             I872                      NaN

       rndrg_prvdr_npi_num clm_carr_pmt_dnl_cd clm_prcsg_ind_cd  \
520          1.619972e+09                    1                A
525          1.811965e+09                    1                A
529          1.336344e+09                    1                A
555          1.295730e+09                    1                A
592          1.861493e+09                    1                A
…                   …                      …                …
5485         1.730182e+09                    1                A
66051        1.063497e+09                    1                A
66052        1.427027e+09                    1                A
66053        1.275519e+09                    1                A
66054        1.932188e+09                    1                S

       clm_adjsmt_type_cd  clm_efctv_dt  clm_idr_ld_dt  clm_cntl_num  \
520                   NaN           NaN            NaN           NaN
525                   NaN           NaN            NaN           NaN
529                   NaN           NaN            NaN           NaN
555                   NaN           NaN            NaN           NaN
592                   NaN           NaN            NaN           NaN
…                     …             …              …             …
5485                  NaN           NaN            NaN           NaN
66051                 NaN           NaN            NaN           NaN
66052                 NaN           NaN            NaN           NaN
66053                 NaN           NaN            NaN           NaN
66054                 NaN           NaN            NaN           NaN

       bene_eqtbl_bic_hicn_num  clm_line_alowd_chrg_amt  \
520                        NaN                     5.49
525                        NaN                    79.36
```

```
529                        NaN                     8.53
555                        NaN                   112.57
592                        NaN                    82.36
…                          …                       …
5485                       NaN                    73.06
66051                      NaN                     7.87
66052                      NaN                     8.97
66053                      NaN                   101.91
66054                      NaN                    10.66

       clm_line_srvc_unit_qty hcpcs_1_mdfr_cd hcpcs_2_mdfr_cd  \
520                       1.0             NaN             NaN
525                       1.0             NaN             NaN
529                       1.0             NaN             NaN
555                       1.0              26             NaN
592                       1.0             NaN             NaN
…                         …               …               …
5485                      1.0             NaN             NaN
66051                     1.0             NaN             NaN
66052                     1.0              26              LT
66053                     1.0             NaN             NaN
66054                     1.0              RT             NaN

       hcpcs_3_mdfr_cd  hcpcs_4_mdfr_cd  hcpcs_5_mdfr_cd  clm_disp_cd  \
520                NaN              NaN              NaN            1
525                NaN              NaN              NaN            1
529                NaN              NaN              NaN            1
555                NaN              NaN              NaN            1
592                NaN              NaN              NaN            1
…                  …                …                …              …
5485               NaN              NaN              NaN            1
66051              NaN              NaN              NaN            1
66052              NaN              NaN              NaN            1
66053              NaN              NaN              NaN            1
66054              NaN              NaN              NaN            1

      clm_dgns_1_cd clm_dgns_2_cd clm_dgns_3_cd clm_dgns_4_cd clm_dgns_5_cd  \
520            I482           NaN           NaN           NaN           NaN
525            F319           NaN           NaN           NaN           NaN
529            R001           NaN           NaN           NaN           NaN
555          M47816           NaN           NaN           NaN           NaN
592          L03032          B351          L853           NaN           NaN
…              …               …             …             …             …
5485          J9601          J810           NaN           NaN           NaN
66051          E782           NaN           NaN           NaN           NaN
66052       S52502A       S52602A           NaN           NaN           NaN
66053         M5116        M47816         M48061           NaN           NaN
```

```
66054          I872           NaN           NaN           NaN           NaN

      clm_dgns_6_cd clm_dgns_7_cd clm_dgns_8_cd  dgns_prcdr_icd_ind  \
520             NaN           NaN           NaN                   0
525             NaN           NaN           NaN                   0
529             NaN           NaN           NaN                   0
555             NaN           NaN           NaN                   0
592             NaN           NaN           NaN                   0
…               …             …             …                     …
5485            NaN           NaN           NaN                   0
66051           NaN           NaN           NaN                   0
66052           NaN           NaN           NaN                   0
66053           NaN           NaN           NaN                   0
66054           NaN           NaN           NaN                   0

      clm_dgns_9_cd clm_dgns_10_cd clm_dgns_11_cd clm_dgns_12_cd  \
520             NaN            NaN            NaN            NaN
525             NaN            NaN            NaN            NaN
529             NaN            NaN            NaN            NaN
555             NaN            NaN            NaN            NaN
592             NaN            NaN            NaN            NaN
…               …              …              …              …
5485            NaN            NaN            NaN            NaN
66051           NaN            NaN            NaN            NaN
66052           NaN            NaN            NaN            NaN
66053           NaN            NaN            NaN            NaN
66054           NaN            NaN            NaN            NaN

      hcpcs_betos_cd
520              T1H
525              M5B
529              T2A
555              I2D
592              M1B
…                …
5485             M2B
66051            T1B
66052            I1B
66053            M1B
66054            O1E

[130699 rows x 49 columns]
```

[34]:
```
#Possible expansio of analysis - keep as side note

partb_physicians_raw_df.groupby('clm_pos_cd').agg(
    uniq_clm_cnt=('cur_clm_uniq_id', 'nunique')
```

```
).sort_values(by='uniq_clm_cnt', ascending=False)
```

[34]:

clm_pos_cd	uniq_clm_cnt
11	34209
81	17512
22	7218
21	6171
23	4505
24	1912
41	1619
19	1435
31	891
32	575
60	530
99	200
20	163
12	99
49	89
13	76
61	38
33	33
53	9
50	7
15	7
51	6
42	4
54	4
71	4
72	3
14	3
65	2
2	2
56	1
17	1

[35]:
```
# Select only the desired columns and remove duplicates if any
partb_physicians_df = partb_physicians_raw_df[[
    'cur_clm_uniq_id', 'bene_mbi_id', 'rndrg_prvdr_npi_num', 'clm_from_dt',
    'clm_line_hcpcs_cd', 'clm_line_dgns_cd', 'clm_line_alowd_chrg_amt'
]].drop_duplicates().rename(
    columns={
        'cur_clm_uniq_id': 'claim_id',
        'bene_mbi_id': 'patient_id',
        'rndrg_prvdr_npi_num': 'npi_id',
        'clm_from_dt': 'claim_date',
        'clm_line_hcpcs_cd': 'hcpcs_code',
```

```
        'clm_line_dgns_cd': 'diagnosis_code',
        'clm_line_alowd_chrg_amt': 'claim_cost'
    }
)
partb_physicians_df
```

[35]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	diagnosis_code	claim_cost
0	100117	10046	1.073515e+09	2016-11-19	83861	H04123	0.00
1	1001777	10133	1.053398e+09	2016-12-15	99213	I480	69.50
2	1001907	10113	1.245238e+09	2017-02-09	11721	B351	43.37
3	1002867	10049	1.255316e+09	2017-09-23	88312	D0359	143.39
4	1002871	10026	1.265419e+09	2016-03-11	87086	N390	10.66
…	…	…	…	…	…	…	…
130694	999919	12345	1.962494e+09	2018-05-03	99214	E782	105.49
130695	999919	12345	1.962494e+09	2018-05-03	90732	Z23	108.14
130696	999919	12345	1.962494e+09	2018-05-03	G0009	Z23	19.91
130697	999959	11445	1.548250e+09	2016-09-24	66984	H2512	838.19
130698	999959	11445	1.548250e+09	2016-09-24	G8918	H2512	0.00

[128904 rows x 7 columns]

Data Quality Check #9: If the resulting dataframe is empty, it means all the records have HCPCS or diagnosis code (if it's not empty it should be removed now since we want only the ones with valid codes for analysis)

[36]:
```
partb_physicians_df.loc[
    (~partb_physicians_df.hcpcs_code.notnull()) | (~partb_physicians_df.
  ↪diagnosis_code.notnull())
]
```

[36]:
```
Empty DataFrame
Columns: [claim_id, patient_id, npi_id, claim_date, hcpcs_code, diagnosis_code,
claim_cost]
Index: []
```

Mini-Analysis #4: Find whether there are matching claims between above four datasets and the Physicians dataset

***

[37]:
```
physicians_unique_claims_df = partb_physicians_df[[
    'claim_id'
]].drop_duplicates()

physicians_unique_claims_df['physicians'] = 1

joined_df4 = pd.merge(
    joined_df3,
    physicians_unique_claims_df,
    on='claim_id', how = 'outer'
)
joined_df4
```

[37]:

	claim_id	header	revenue	diagnosis	dme	physicians
0	1001595	1.0	1.0	1.0	NaN	NaN
1	1004555	1.0	1.0	1.0	NaN	NaN
2	1011605	1.0	1.0	1.0	NaN	NaN
3	1011758	1.0	1.0	1.0	NaN	NaN
4	101424	1.0	NaN	NaN	NaN	NaN
…	…	…	…	…	…	…
97953	999905	NaN	NaN	NaN	NaN	1.0
97954	999908	NaN	NaN	NaN	NaN	1.0
97955	999916	NaN	NaN	NaN	NaN	1.0
97956	999919	NaN	NaN	NaN	NaN	1.0
97957	999959	NaN	NaN	NaN	NaN	1.0

[97958 rows x 6 columns]

[38]:
```
print('# of unique claims in first four datasets: '
      + str(joined_df3.shape[0])
     )
print('# of unique claims in Physicians dataset: '
      + str(physicians_unique_claims_df.shape[0])
     )
```

```
# of unique claims in first four datasets: 20965
# of unique claims in Physicians dataset: 76993
```

[39]:
```
print('# of unique claims in the five datasets combined: '
      + str(joined_df4.shape[0])
     )
print('From combined list of unique claims - ')
print('# of unique claims in only the first four datasets: '
      + str(joined_df4.loc[
```

```
          ((joined_df4.header == 1)
          | (joined_df4.revenue == 1)
          | (joined_df4.diagnosis == 1)
          | (joined_df4.dme == 1))
          & ~(joined_df4.physicians == 1)
      ].shape[0])
     )
print('# of unique claims in only Physicians dataset: '
      + str(joined_df4.loc[
          ~(joined_df4.header == 1)
          & ~(joined_df4.revenue == 1)
          & ~(joined_df4.diagnosis == 1)
          & ~(joined_df4.dme == 1)
          & (joined_df4.physicians == 1)
      ].shape[0])
     )
print('# of unique claims in all five datasets: '
      + str(joined_df4.loc[
          (joined_df4.header == 1)
          & (joined_df4.revenue == 1)
          & (joined_df4.diagnosis == 1)
          & (joined_df4.dme == 1)
          & (joined_df4.physicians == 1)
      ].shape[0])
     )
print('# of unique claims in Physicians and any of the first four datasets: '
      + str(joined_df4.loc[
          ((joined_df4.header == 1)
          | (joined_df4.revenue == 1)
          | (joined_df4.diagnosis == 1)
          | (joined_df4.dme == 1))
          & (joined_df4.physicians == 1)
      ].shape[0])
     )
```

```
# of unique claims in the five datasets combined: 97958
From combined list of unique claims -
# of unique claims in only the first four datasets: 20965
# of unique claims in only Physicians dataset: 76993
# of unique claims in all five datasets: 0
# of unique claims in Physicians and any of the first four datasets: 0
```

Conclusion: None of the claims from Physicians is in any of the first four datasets, so append them to the output after combining the first four datasets

***

1.1.9 0.2.7 Load & select columns from Patients dataset

[40]:
```
# Load the Patients dataset
# For sake of simplicity in concept, beneficiary = patient
beneficiary_demographics_df = pd.read_csv("/Users/hamiddastgir/Library/
 ↪CloudStorage/Dropbox/Semester 3/BIA 810 - Healthcare Analytics/Mid Term/
 ↪Syntegra Datasets Files/beneficiary_demographics.csv")
beneficiary_demographics_df
```

[40]:

	bene_mbi_id	bene_member_month	bene_hic_num	bene_fips_state_cd	\
0	10	1/1/16 0:00	NaN	55	
1	10	2/1/16 0:00	NaN	55	
2	10	3/1/16 0:00	NaN	55	
3	10	4/1/16 0:00	NaN	55	
4	10	5/1/16 0:00	NaN	55	
…	…	…	…	…	
31179	13380	2/1/18 0:00	NaN	44	
31180	13380	3/1/18 0:00	NaN	44	
31181	13380	4/1/18 0:00	NaN	44	
31182	13380	5/1/18 0:00	NaN	44	
31183	13380	6/1/18 0:00	NaN	44	

	bene_fips_cnty_cd	bene_zip_cd	bene_dob	bene_sex_cd	\
0	79	NaN	5/16/45 0:00	1	
1	79	NaN	5/16/45 0:00	1	
2	79	NaN	5/16/45 0:00	1	
3	79	NaN	5/16/45 0:00	1	
4	79	NaN	5/16/45 0:00	1	
…	…	…	…	…	
31179	7	NaN	3/31/47 0:00	2	
31180	7	NaN	3/31/47 0:00	2	
31181	7	NaN	3/31/47 0:00	2	
31182	7	NaN	3/31/47 0:00	2	
31183	7	NaN	3/31/47 0:00	2	

	bene_race_cd	bene_age	bene_mdcr_stus_cd	bene_dual_stus_cd	\
0	1	71	10.0	NaN	
1	1	71	10.0	NaN	
2	1	71	10.0	NaN	
3	1	71	10.0	NaN	
4	1	71	10.0	NaN	
…	…	…	…	…	
31179	1	71	10.0	NaN	
31180	1	71	10.0	NaN	
31181	1	71	10.0	NaN	
31182	1	71	10.0	NaN	
31183	1	71	10.0	NaN	

```
      bene_death_dt  bene_rng_bgn_dt  bene_rng_end_dt  bene_1st_name  \
0               NaN              NaN              NaN            NaN
1               NaN              NaN              NaN            NaN
2               NaN              NaN              NaN            NaN
3               NaN              NaN              NaN            NaN
4               NaN              NaN              NaN            NaN
…               …                …                …              …
31179           NaN              NaN              NaN            NaN
31180           NaN              NaN              NaN            NaN
31181           NaN              NaN              NaN            NaN
31182           NaN              NaN              NaN            NaN
31183           NaN              NaN              NaN            NaN

       bene_midl_name  bene_last_name  bene_orgnl_entlmt_rsn_cd  \
0                 NaN             NaN                         0
1                 NaN             NaN                         0
2                 NaN             NaN                         0
3                 NaN             NaN                         0
4                 NaN             NaN                         0
…                 …               …                           …
31179             NaN             NaN                         0
31180             NaN             NaN                         0
31181             NaN             NaN                         0
31182             NaN             NaN                         0
31183             NaN             NaN                         0

      bene_entlmt_buyin_ind  bene_part_a_enrlmt_bgn_dt  \
0                         3                        NaN
1                         3                        NaN
2                         3                        NaN
3                         3                        NaN
4                         3                        NaN
…                         …                        …
31179                     3                        NaN
31180                     3                        NaN
31181                     3                        NaN
31182                     3                        NaN
31183                     3                        NaN

       bene_part_b_enrlmt_bgn_dt  bene_line_1_adr  bene_line_2_adr  \
0                            NaN              NaN              NaN
1                            NaN              NaN              NaN
2                            NaN              NaN              NaN
3                            NaN              NaN              NaN
4                            NaN              NaN              NaN
…                            …                …                …
```

```
31179                      NaN              NaN              NaN
31180                      NaN              NaN              NaN
31181                      NaN              NaN              NaN
31182                      NaN              NaN              NaN
31183                      NaN              NaN              NaN

       bene_line_3_adr  bene_line_4_adr  bene_line_5_adr  bene_line_6_adr  \
0                  NaN              NaN              NaN              NaN
1                  NaN              NaN              NaN              NaN
2                  NaN              NaN              NaN              NaN
3                  NaN              NaN              NaN              NaN
4                  NaN              NaN              NaN              NaN
…                  …                …                …                …
31179              NaN              NaN              NaN              NaN
31180              NaN              NaN              NaN              NaN
31181              NaN              NaN              NaN              NaN
31182              NaN              NaN              NaN              NaN
31183              NaN              NaN              NaN              NaN

       geo_zip_plc_name  geo_usps_state_cd  geo_zip5_cd  geo_zip4_cd
0                   NaN                 52          NaN          NaN
1                   NaN                 52          NaN          NaN
2                   NaN                 52          NaN          NaN
3                   NaN                 52          NaN          NaN
4                   NaN                 52          NaN          NaN
…                   …                   …           …            …
31179               NaN                 41          NaN          NaN
31180               NaN                 41          NaN          NaN
31181               NaN                 41          NaN          NaN
31182               NaN                 41          NaN          NaN
31183               NaN                 41          NaN          NaN

[31184 rows x 32 columns]
```

[41]:
```
# Select only the desired columns (rename columns if needed) and remove␣
 ↪duplicates if any
beneficiary_demographics_df = beneficiary_demographics_df[[
    'bene_mbi_id', 'bene_dob', 'bene_sex_cd'
]].drop_duplicates().rename(
    columns={
        'bene_mbi_id': 'patient_id',
        'bene_dob': 'patient_birth_date'
    }
)
beneficiary_demographics_df
```

[41]:
```
       patient_id patient_birth_date  bene_sex_cd
0              10       5/16/45 0:00            1
34          10007        1/4/56 0:00            2
63          10010       12/3/32 0:00            2
95          10013       8/23/52 0:00            2
131         10017      11/23/84 0:00            1
…             …                 …          …
31037       13374       7/11/48 0:00            2
31072       13376      11/28/52 0:00            2
31103       13377       1/16/56 0:00            1
31122       13379      12/10/26 0:00            2
31155       13380       3/31/47 0:00            2

[1000 rows x 3 columns]
```

[42]:
```
# Gender code as identified by the CMS CCLF resource (1 = male, 2 = female, 0 = Unknown = N/A)
# Convert gender code into readable acronym and drop original column
beneficiary_demographics_df['patient_gender'] = ''
beneficiary_demographics_df.loc[beneficiary_demographics_df.bene_sex_cd == 1,'patient_gender'] = 'M'
beneficiary_demographics_df.loc[beneficiary_demographics_df.bene_sex_cd == 2,'patient_gender'] = 'F'
beneficiary_demographics_df = beneficiary_demographics_df.drop('bene_sex_cd',axis=1)
beneficiary_demographics_df
```

[42]:
```
       patient_id patient_birth_date patient_gender
0              10       5/16/45 0:00              M
34          10007        1/4/56 0:00              F
63          10010       12/3/32 0:00              F
95          10013       8/23/52 0:00              F
131         10017      11/23/84 0:00              M
…             …                 …            …
31037       13374       7/11/48 0:00              F
31072       13376      11/28/52 0:00              F
31103       13377       1/16/56 0:00              M
31122       13379      12/10/26 0:00              F
31155       13380       3/31/47 0:00              F

[1000 rows x 3 columns]
```

[45]:
```
beneficiary_demographics_df['patient_birth_date'].head()
```

[45]:
```
0       5/16/45 0:00
34       1/4/56 0:00
63      12/3/32 0:00
```

```
95      8/23/52 0:00
131    11/23/84 0:00
Name: patient_birth_date, dtype: object
```

[46]:
```
print(beneficiary_demographics_df['patient_birth_date'].head(10))
```

```
0       5/16/45 0:00
34       1/4/56 0:00
63      12/3/32 0:00
95      8/23/52 0:00
131    11/23/84 0:00
160     5/18/38 0:00
195      3/1/48 0:00
228     9/26/45 0:00
262      2/9/46 0:00
289      5/1/50 0:00
Name: patient_birth_date, dtype: object
```

2 Changing Birth Date Formatting Since it Causes Issues Later Onwards

[47]:
```
def parse_patient_birth_date(date_str):
    if pd.isna(date_str) or date_str.strip() == '':
        return pd.NaT
    date_str = date_str.strip()
    for fmt in ('%m/%d/%y %H:%M', '%m/%d/%Y %H:%M', '%Y-%m-%d'):
        try:
            dt = datetime.strptime(date_str, fmt)
            if dt.year > datetime.now().year:
                dt = dt.replace(year=dt.year - 100)
            return dt
        except ValueError:
            continue
    print(f"Could not parse date: {date_str}")
    return pd.NaT
```

[48]:
```
beneficiary_demographics_df['patient_birth_date'] =␣
↪beneficiary_demographics_df['patient_birth_date'].
↪apply(parse_patient_birth_date)
```

[49]:
```
beneficiary_demographics_df['patient_birth_date']
```

[49]:
```
0       1945-05-16
34      1956-01-04
63      1932-12-03
95      1952-08-23
131     1984-11-23
```

```
          …
31037   1948-07-11
31072   1952-11-28
31103   1956-01-16
31122   1926-12-10
31155   1947-03-31
Name: patient_birth_date, Length: 1000, dtype: datetime64[ns]
```

[50]:
```
beneficiary_demographics_df
```

[50]:
```
       patient_id patient_birth_date patient_gender
0              10         1945-05-16              M
34          10007         1956-01-04              F
63          10010         1932-12-03              F
95          10013         1952-08-23              F
131         10017         1984-11-23              M
…             …                …             …
31037       13374         1948-07-11              F
31072       13376         1952-11-28              F
31103       13377         1956-01-16              M
31122       13379         1926-12-10              F
31155       13380         1947-03-31              F

[1000 rows x 3 columns]
```

Mini-Analysis #5: Find whether there are matching patients between the claims datasets and the Patients dataset

***

[51]:
```
claims_header_unique_patients_df = parta_claims_header_df[[
    'patient_id'
]].drop_duplicates()

claims_header_unique_patients_df['header'] = 1

revenue_center_unique_patients_df = parta_claims_revenue_center_detail_df[[
    'patient_id'
]].drop_duplicates()

revenue_center_unique_patients_df['revenue'] = 1

diagnosis_unique_patients_df = parta_diagnosis_code_df[[
    'patient_id'
]].drop_duplicates()

diagnosis_unique_patients_df['diagnosis'] = 1
```

```
dme_unique_patients_df = partb_dme_df[[
    'patient_id'
]].drop_duplicates()

dme_unique_patients_df['dme'] = 1

physicians_unique_patients_df = partb_physicians_df[[
    'patient_id'
]].drop_duplicates()

physicians_unique_patients_df['physicians'] = 1

beneficiary_unique_patients_df = beneficiary_demographics_df[[
    'patient_id'
]].drop_duplicates()

beneficiary_unique_patients_df['beneficiary'] = 1

joined_patients_df = pd.merge(
    pd.merge(
        pd.merge(
            pd.merge(
                pd.merge(
                    claims_header_unique_patients_df,
                    revenue_center_unique_patients_df,
                    on='patient_id', how = 'outer'
                ),
                diagnosis_unique_patients_df,
                on='patient_id', how = 'outer'
            ),
            dme_unique_patients_df,
            on='patient_id', how = 'outer'
        ),
        physicians_unique_patients_df,
        on='patient_id', how = 'outer'
    ),
    beneficiary_unique_patients_df,
    on='patient_id', how = 'outer'
)

joined_patients_df
```

[51]:

	patient_id	header	revenue	diagnosis	dme	physicians	beneficiary
0	10226	1.0	1.0	1.0	NaN	1.0	1
1	10133	1.0	1.0	1.0	1.0	1.0	1
2	10163	1.0	1.0	1.0	NaN	1.0	1
3	1003	1.0	1.0	1.0	NaN	1.0	1

```
4          10052     1.0      1.0         1.0  NaN          1.0             1
..           …     …       …          …   …           …           …
995        12868     NaN      NaN         NaN  NaN          NaN             1
996        13001     NaN      NaN         NaN  NaN          NaN             1
997        13157     NaN      NaN         NaN  NaN          NaN             1
998        13298     NaN      NaN         NaN  NaN          NaN             1
999        13351     NaN      NaN         NaN  NaN          NaN             1

[1000 rows x 7 columns]
```

[52]:
```
print('# of unique patients in the five datasets combined: '
      + str(joined_patients_df.shape[0])
     )
print('From combined list of unique patients - ')
print('# of unique patients in only the claims datasets: '
      + str(joined_patients_df.loc[
          ((joined_patients_df.header == 1)
          | (joined_patients_df.revenue == 1)
          | (joined_patients_df.diagnosis == 1)
          | (joined_patients_df.dme == 1)
          | (joined_patients_df.physicians == 1))
          & ~(joined_patients_df.beneficiary == 1)
      ].shape[0])
     )
print('# of unique patients in only Beneficiary dataset: '
      + str(joined_patients_df.loc[
          ~(joined_patients_df.header == 1)
          & ~(joined_patients_df.revenue == 1)
          & ~(joined_patients_df.diagnosis == 1)
          & ~(joined_patients_df.dme == 1)
          & ~(joined_patients_df.physicians == 1)
          & (joined_patients_df.beneficiary == 1)
      ].shape[0])
     )
print('# of unique patients in all five datasets: '
      + str(joined_patients_df.loc[
          (joined_patients_df.header == 1)
          & (joined_patients_df.revenue == 1)
          & (joined_patients_df.diagnosis == 1)
          & (joined_patients_df.dme == 1)
          & (joined_patients_df.physicians == 1)
          & (joined_patients_df.beneficiary == 1)
      ].shape[0])
     )
print('# of unique patients in Beneficiary and any of the claims datasets: '
      + str(joined_patients_df.loc[
          ((joined_patients_df.header == 1)
```

```
          | (joined_patients_df.revenue == 1)
          | (joined_patients_df.diagnosis == 1)
          | (joined_patients_df.dme == 1)
          | (joined_patients_df.physicians == 1))
         & (joined_patients_df.beneficiary == 1)
      ].shape[0])
     )
```

```
# of unique patients in the five datasets combined: 1000
From combined list of unique patients -
# of unique patients in only the claims datasets: 0
# of unique patients in only Beneficiary dataset: 38
# of unique patients in all five datasets: 276
# of unique patients in Beneficiary and any of the claims datasets: 962
```

Conclusion: Most of the patients have some claims, so we can join the beneficiary dataset to the claims to get some of the patient demographics, i.e. age and gender

***

2.0.1 0.3 Combine all datasets

2.0.2 0.3.1. Join datasets with common records

[53]:
```
# Join Claims Header and Claims Revenue Center datasets on claim ID, patient␣
 ↪ID, and claim date
# Perform outer join to capture all possible claims
medicare_df = pd.merge(
    parta_claims_header_df,
    parta_claims_revenue_center_detail_df,
    on=['claim_id','patient_id','claim_date'], how='outer'
)
medicare_df.sort_values(by='claim_id')#.head(100)
```

[53]:

	claim_id	patient_id	npi_id	claim_date	prncpl_dgns_cd	\
22501	100073	12620	NaN	2018-12-02	NaN	
1281	100190	1228	1.972732e+09	2018-06-10	M1611	
1285	100190	1228	1.972732e+09	2018-06-10	M1611	
1284	100190	1228	1.972732e+09	2018-06-10	M1611	
1283	100190	1228	1.972732e+09	2018-06-10	M1611	
…	…	…	…	…	…	
30555	1699195	10958	NaN	2017-04-19	NaN	
30556	1699195	10958	NaN	2017-04-19	NaN	
30557	1699197	1177	NaN	2016-05-22	NaN	
30558	1699197	1177	NaN	2016-05-22	NaN	
30559	1699236	10580	NaN	2017-09-20	NaN	

	clm_pmt_amt	hcpcs_code	clm_line_cvrd_pd_amt
22501	NaN	77063	24.11

```
1281        127.79      98960                  0.00
1285        127.79      99213                  0.00
1284        127.79      J2270                  0.00
1283        127.79      J1885                  0.00
…              …        …                      …
30555          NaN      00810                  0.00
30556          NaN      J2250                  0.00
30557          NaN      36415                  0.00
30558          NaN      86592                  5.43
30559          NaN      45385                543.04

[38394 rows x 8 columns]
```

[54]:
```
# Join Medicare and Diagnosis datasets on claim ID, patient ID, and claim date
# Perform outer join to capture all possible claims
medicare_df = pd.merge(
    medicare_df,
    parta_diagnosis_code_df,
    on=['claim_id','patient_id','claim_date'], how='outer'
)
medicare_df.sort_values(by='claim_id')#.head(100)
```

[54]:
```
       claim_id  patient_id        npi_id  claim_date prncpl_dgns_cd  \
62201    100073       12620           NaN  2018-12-02            NaN
3396     100190        1228  1.972732e+09  2018-06-10          M1611
3395     100190        1228  1.972732e+09  2018-06-10          M1611
3394     100190        1228  1.972732e+09  2018-06-10          M1611
3393     100190        1228  1.972732e+09  2018-06-10          M1611
…           …           …           …           …              …
79460   1699195       10958           NaN  2017-04-19            NaN
79461   1699195       10958           NaN  2017-04-19            NaN
79464   1699197        1177           NaN  2016-05-22            NaN
79465   1699197        1177           NaN  2016-05-22            NaN
79466   1699236       10580           NaN  2017-09-20            NaN

       clm_pmt_amt hcpcs_code  clm_line_cvrd_pd_amt clm_dgns_cd
62201          NaN      77063                 24.11         NaN
3396        127.79      99213                  0.00       M5136
3395        127.79      99213                  0.00       M1611
3394        127.79      99213                  0.00      M25572
3393        127.79      J2270                  0.00      M25551
…              …        …                      …          …
79460          NaN      88342                  0.00         NaN
79461          NaN      43239                1275.58         NaN
79464          NaN      36415                  0.00         NaN
79465          NaN      86592                  5.43         NaN
79466          NaN      45385                543.04         NaN
```

```
[106785 rows x 9 columns]
```

[55]:
```
# Since Claims Header dataset has some principal diagnosis codes and the␣
 ↪Diagnosis dataset
#  supplements them with additional codes wherever possible,
#  coalesce them with preference to the principal code from Claim Header dataset
# Once the diagnosis codes are combined into one column, remove the older␣
 ↪columns and any duplicates
medicare_df['diagnosis_code'] = medicare_df[['prncpl_dgns_cd', 'clm_dgns_cd']].
 ↪bfill(axis=1).iloc[:, 0]
medicare_df = medicare_df.drop(['prncpl_dgns_cd', 'clm_dgns_cd'], axis=1).
 ↪drop_duplicates()
medicare_df.sort_values(by='claim_id').head(20)
```

[55]:

	claim_id	patient_id	npi_id	claim_date	clm_pmt_amt	hcpcs_code	clm_line_cvrd_pd_amt	diagnosis_code
62201	100073	12620	NaN	2018-12-02	NaN	77063	24.11	NaN
3386	100190	1228	1.972732e+09	2018-06-10	127.79	J1885	0.00	M1611
3378	100190	1228	1.972732e+09	2018-06-10	127.79	98960	0.00	M1611
3374	100190	1228	1.972732e+09	2018-06-10	127.79	G0467	133.74	M1611
3390	100190	1228	1.972732e+09	2018-06-10	127.79	J2270	0.00	M1611
3394	100190	1228	1.972732e+09	2018-06-10	127.79	99213	0.00	M1611
3382	100190	1228	1.972732e+09	2018-06-10	127.79	J1100	0.00	M1611
62426	100227	12140	NaN	2018-10-24	NaN	J2785	0.00	NaN
96298	100402	1261	NaN	2017-06-02	NaN	NaN	NaN	E119
96297	100402	1261	NaN	2017-06-02	NaN	NaN	NaN	E785
3670	100402	1261	1.285688e+09	2017-05-27	10602.46	NaN	0.00	E785
3682	100464	12978	1.982693e+09	2017-06-26	199.45	74230	0.00	E785
96310	100564	11929	NaN	2018-04-12	NaN	NaN	NaN	E785
3700	100698	11789	1.912991e+09	2017-07-28	85.25	G0463	0.00	E785
3705	100750	12138	1.063442e+09	2018-01-13	0.00	NaN	0.00	E785
62051	100974	10042	NaN	2017-02-20	NaN	G0202	0.00	E785
3763	100982	12086	1.902839e+09	2018-06-23	608.45	95811	0.00	E785
3776	101001	11663	1.932133e+09	2016-11-08	50.20	71020	0.00	E785
96381	101117	11835	NaN	2018-04-10	NaN	NaN	NaN	E785
62674	101147	13359	NaN	2017-09-02	NaN	80053	0.00	NaN

```
96297                  NaN          R197
3670                   NaN         K5733
3682                 83.17          R079
96310                  NaN        F17210
3700                 85.05          M545
3705                   NaN         Z0289
62051                99.75           NaN
3763                832.94         G4733
3776                 48.65          R918
96381                  NaN         N6002
62674                11.21          E785
```

[56]:
```
# Since Claims Header dataset records the amount Medicare paid for the claims
#  and Claims Revenue Center dataset records the amount Medicare reimbursed the␣
 ↪provider,
#  assume they were separate charges and add them to get the total cost for␣
 ↪claim (for particular code)
# Once the costs are combined into one column, remove the older columns and any␣
 ↪duplicates
medicare_df['claim_cost'] =␣
 ↪medicare_df['clm_pmt_amt']+medicare_df['clm_line_cvrd_pd_amt']
medicare_df = medicare_df.drop(['clm_pmt_amt', 'clm_line_cvrd_pd_amt'], axis=1).
 ↪drop_duplicates()
medicare_df.sort_values(by='claim_id').head(20)
```

[56]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code
62201	100073	12620	NaN	2018-12-02	77063
3374	100190	1228	1.972732e+09	2018-06-10	G0467
3394	100190	1228	1.972732e+09	2018-06-10	99213
3390	100190	1228	1.972732e+09	2018-06-10	J2270
3378	100190	1228	1.972732e+09	2018-06-10	98960
3386	100190	1228	1.972732e+09	2018-06-10	J1885
3382	100190	1228	1.972732e+09	2018-06-10	J1100
62426	100227	12140	NaN	2018-10-24	J2785
3670	100402	1261	1.285688e+09	2017-05-27	NaN
96297	100402	1261	NaN	2017-06-02	NaN
96298	100402	1261	NaN	2017-06-02	NaN
3682	100464	12978	1.982693e+09	2017-06-26	74230
96310	100564	11929	NaN	2018-04-12	NaN
3700	100698	11789	1.912991e+09	2017-07-28	G0463
3705	100750	12138	1.063442e+09	2018-01-13	NaN
62051	100974	10042	NaN	2017-02-20	G0202
3763	100982	12086	1.902839e+09	2018-06-23	95811
3776	101001	11663	1.932133e+09	2016-11-08	71020
96381	101117	11835	NaN	2018-04-10	NaN
62674	101147	13359	NaN	2017-09-02	80053

```
      diagnosis_code  claim_cost
62201            NaN         NaN
3374           M1611      261.53
3394           M1611      127.79
3390           M1611      127.79
3378           M1611      127.79
3386           M1611      127.79
3382           M1611      127.79
62426            NaN         NaN
3670           K5733         NaN
96297           R197         NaN
96298           E119         NaN
3682            R079      282.62
96310         F17210         NaN
3700            M545      170.30
3705           Z0289         NaN
62051            NaN         NaN
3763           G4733     1441.39
3776            R918       98.85
96381          N6002         NaN
62674           E785         NaN
```

[63]:
```
claims_header_revenue_diagnosis_df_count = medicare_df.shape[0]
```

2.0.3 0.3.2 Append datasets with no common records

[57]:
```
# Create a dummy column for diagnosis code in DME dataset before appending so␣
 ↪the list of columns match
partb_dme_df['diagnosis_code'] = np.nan
partb_dme_df = partb_dme_df[[
    'claim_id', 'patient_id', 'npi_id', 'claim_date', 'hcpcs_code',␣
 ↪'diagnosis_code', 'claim_cost'
]]
partb_dme_df
```

[57]:
```
      claim_id  patient_id        npi_id  claim_date hcpcs_code  \
0      1004024       10202  1.841430e+09  2016-07-18      E0601
1      1034063       10137  1.669460e+09  2016-04-22      E0601
2      1046877       10202  1.093713e+09  2016-02-03      E0601
3      1072934       10202  1.285602e+09  2016-08-15      E0601
4      1082554       10174  1.003895e+09  2016-08-30      E0431
…        …          …            …          …         …
2770    998097       10396  1.891706e+09  2016-12-06      A4253
2771    999226        1095  1.518066e+09  2017-12-28      A4256
2772    999226        1095  1.518066e+09  2017-12-28      A4253
2773    999226        1095  1.518066e+09  2017-12-28      A4259
2774    999929       10261  1.497738e+09  2018-06-12      E0570
```

```
      diagnosis_code  claim_cost
0                NaN       41.91
1                NaN       62.46
2                NaN       29.31
3                NaN       27.82
4                NaN       18.75
…                …          …
2770             NaN       69.69
2771             NaN        3.68
2772             NaN       49.92
2773             NaN        4.26
2774             NaN        3.00

[2731 rows x 7 columns]
```

[58]:
```
# Append DME dataset to the first three claims datasets
medicare_df = pd.concat([medicare_df, partb_dme_df])
medicare_df
```

[58]:
```
      claim_id  patient_id        npi_id  claim_date hcpcs_code  \
0      1001595       10226  1.366492e+09  2018-02-28      G0283
2      1001595       10226  1.366492e+09  2018-02-28      G8978
4      1001595       10226  1.366492e+09  2018-02-28      G8979
6      1001595       10226  1.366492e+09  2018-02-28      97110
8      1001595       10226  1.366492e+09  2018-02-28      97140
…        …          …           …          …        …
2770    998097       10396  1.891706e+09  2016-12-06      A4253
2771    999226        1095  1.518066e+09  2017-12-28      A4256
2772    999226        1095  1.518066e+09  2017-12-28      A4253
2773    999226        1095  1.518066e+09  2017-12-28      A4259
2774    999929       10261  1.497738e+09  2018-06-12      E0570

     diagnosis_code  claim_cost
0            M25551      268.68
2            M25551      259.01
4            M25551      259.01
6            M25551      283.98
8            M25551      279.34
…             …          …
2770            NaN       69.69
2771            NaN        3.68
2772            NaN       49.92
2773            NaN        4.26
2774            NaN        3.00

[69648 rows x 7 columns]
```

[61]:
```
claims_header_revenue_diagnosis_dme_df_count = medicare_df.shape[0]
```

Data Quality Check #10: If True, we appended DME dataset to the first three claims datasets without any unexpected rows accruing

[64]:
```
partb_dme_df_count = partb_dme_df.shape[0]
claims_w_dme_df_count = medicare_df.shape[0]

print('Claim count for Claims Header + Revenue Center + Diagnosis: '
      + str(claims_header_revenue_diagnosis_df_count))
print('Claim count for DME: ' + str(partb_dme_df_count))
print('Expected claim count after appending DME dataset: '
     + str(claims_header_revenue_diagnosis_df_count+partb_dme_df_count))
print('Actual claim count after appending DME dataset: '
     + str(claims_w_dme_df_count))
print('Expected and actual claim count matches: '
     + str(claims_header_revenue_diagnosis_df_count+partb_dme_df_count ==␣
 ↪claims_w_dme_df_count))
```

```
Claim count for Claims Header + Revenue Center + Diagnosis: 69648
Claim count for DME: 2731
Expected claim count after appending DME dataset: 72379
Actual claim count after appending DME dataset: 69648
Expected and actual claim count matches: False
```

[65]:
```
# Append Physicians dataset to the first four claims datasets
medicare_df = pd.concat([medicare_df, partb_physicians_df])
medicare_df
```

[65]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	diagnosis_code	claim_cost
0	1001595	10226	1.366492e+09	2018-02-28	G0283	M25551	268.68
2	1001595	10226	1.366492e+09	2018-02-28	G8978	M25551	259.01
4	1001595	10226	1.366492e+09	2018-02-28	G8979	M25551	259.01
6	1001595	10226	1.366492e+09	2018-02-28	97110	M25551	283.98
8	1001595	10226	1.366492e+09	2018-02-28	97140	M25551	279.34
…	…	…	…	…	…		
130694	999919	12345	1.962494e+09	2018-05-03	99214		
130695	999919	12345	1.962494e+09	2018-05-03	90732		
130696	999919	12345	1.962494e+09	2018-05-03	G0009		
130697	999959	11445	1.548250e+09	2016-09-24	66984		
130698	999959	11445	1.548250e+09	2016-09-24	G8918		

```
…                …          …
130694           E782      105.49
130695            Z23      108.14
130696            Z23       19.91
130697          H2512      838.19
130698          H2512        0.00

[198552 rows x 7 columns]
```

Data Quality Check #11: If True, we appended Physicians dataset to the first four claims datasets without any unexpected rows accruing

[66]:
```
partb_physicians_df_count = partb_physicians_df.shape[0]
claims_w_physicians_df_count = medicare_df.shape[0]

print('Claim count for Claims Header + Revenue Center + Diagnosis + DME: '
      + str(claims_header_revenue_diagnosis_dme_df_count))
print('Claim count for Physicians: ' + str(partb_physicians_df_count))
print('Expected claim count after appending Physicians dataset: '
     +␣
 ↪str(claims_header_revenue_diagnosis_dme_df_count+partb_physicians_df_count))
print('Actual claim count after appending Physicians dataset: '
     + str(claims_w_physicians_df_count))
print('Expected and actual claim count matches: '
     +␣
 ↪str(claims_header_revenue_diagnosis_dme_df_count+partb_physicians_df_count
           == claims_w_physicians_df_count))
```

```
Claim count for Claims Header + Revenue Center + Diagnosis + DME: 69648
Claim count for Physicians: 128904
Expected claim count after appending Physicians dataset: 198552
Actual claim count after appending Physicians dataset: 198552
Expected and actual claim count matches: True
```

[67]:
```
# Capture # records now to compare after joining patient details
medicare_claims_df_count = medicare_df.shape[0]
```

2.0.4 0.3.3 Join patient information

[68]:
```
# Join claims data with patient details on patient ID
# Perform left join to only provide patient details for existing claims
medicare_df = pd.merge(
    medicare_df,
    beneficiary_demographics_df,
    on=['patient_id'], how='left'
)
medicare_df.sort_values(by='claim_id')#.head(100)
```

[68]:
```
         claim_id  patient_id        npi_id  claim_date hcpcs_code  \
70160      100020        1070  1.619972e+09  2016-10-04      85610
70165      100024       11654  1.811965e+09  2016-12-10      90834
70169      100030       12052  1.336344e+09  2017-04-15      93010
70195      100038       12345  1.295730e+09  2018-07-02      72158
70230      100061       10252  1.861493e+09  2016-07-04      99213
…             …           …           …           …          …
39703     1699197        1177           NaN  2016-05-22      36415
39704     1699197        1177           NaN  2016-05-22      86592
134759    1699204       11540  1.275519e+09  2018-05-08      99214
134760    1699222       11556  1.932188e+09  2016-03-16      J7060
39705     1699236       10580           NaN  2017-09-20      45385

       diagnosis_code  claim_cost patient_birth_date patient_gender
70160            I482        5.49         1947-06-06              M
70165            F319       79.36         1977-11-08              M
70169            R001        8.53         1949-06-25              F
70195          M47816      112.57         1947-01-28              F
70230          L03032       82.36         1933-02-09              F
…                 …           …                  …            …
39703             NaN         NaN         1948-09-07              M
39704             NaN         NaN         1948-09-07              M
134759          M5116      101.91         1952-03-08              F
134760           I872       10.66         1955-02-04              F
39705             NaN         NaN         1947-01-13              F

[198552 rows x 9 columns]
```

Data Quality Check #12: If True, we joined Patient dataset to the claims datasets without any unexpected rows accruing

[69]:
```
# Check # records to ensure no extra records happened accidentally
medicare_claims_n_patient_info_df_count = medicare_df.shape[0]

print('Claims count before adding patient details: ' +␣
 ↪str(medicare_claims_df_count))
print('Claims count after adding patient details: ' +␣
 ↪str(medicare_claims_n_patient_info_df_count))
print('If True, no extra records were added accidentally from joining patient␣
 ↪details into the claims: '
      + str(medicare_claims_df_count ==␣
 ↪medicare_claims_n_patient_info_df_count))
```

```
Claims count before adding patient details: 198552
Claims count after adding patient details: 198552
If True, no extra records were added accidentally from joining patient details
into the claims: True
```

[70]:
```
# Get claim year
medicare_df['claim_year'] = pd.to_datetime(
    medicare_df['claim_date']
).dt.strftime('%Y')
medicare_df
```

[70]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	\
0	1001595	10226	1.366492e+09	2018-02-28	G0283	
1	1001595	10226	1.366492e+09	2018-02-28	G8978	
2	1001595	10226	1.366492e+09	2018-02-28	G8979	
3	1001595	10226	1.366492e+09	2018-02-28	97110	
4	1001595	10226	1.366492e+09	2018-02-28	97140	
…	…	…	…	…	…	
198547	999919	12345	1.962494e+09	2018-05-03	99214	
198548	999919	12345	1.962494e+09	2018-05-03	90732	
198549	999919	12345	1.962494e+09	2018-05-03	G0009	
198550	999959	11445	1.548250e+09	2016-09-24	66984	
198551	999959	11445	1.548250e+09	2016-09-24	G8918	

	diagnosis_code	claim_cost	patient_birth_date	patient_gender	claim_year
0	M25551	268.68	1951-02-27	M	2018
1	M25551	259.01	1951-02-27	M	2018
2	M25551	259.01	1951-02-27	M	2018
3	M25551	283.98	1951-02-27	M	2018
4	M25551	279.34	1951-02-27	M	2018
…	…	…	…	…	…
198547	E782	105.49	1947-01-28	F	2018
198548	Z23	108.14	1947-01-28	F	2018
198549	Z23	19.91	1947-01-28	F	2018
198550	H2512	838.19	1945-04-03	F	2016
198551	H2512	0.00	1945-04-03	F	2016

[198552 rows x 10 columns]

[71]:
```
# Get patient age - subtract birthdate from claim date year to get the patient␣
 ↪age at the time of claims
medicare_df['patient_birth_year'] = pd.to_datetime(
    medicare_df['patient_birth_date']
).dt.strftime('%Y')

medicare_df['patient_age'] = (
    medicare_df['claim_year'].astype('int') - medicare_df['patient_birth_year'].
 ↪astype('int')
)

medicare_df = medicare_df.drop('patient_birth_year', axis=1)
```

```
medicare_df
```

[71]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	diagnosis_code	claim_cost	patient_birth_date	patient_gender	claim_year	patient_age
0	1001595	10226	1.366492e+09	2018-02-28	G0283	M25551	268.68	1951-02-27	M	2018	67
1	1001595	10226	1.366492e+09	2018-02-28	G8978	M25551	259.01	1951-02-27	M	2018	67
2	1001595	10226	1.366492e+09	2018-02-28	G8979	M25551	259.01	1951-02-27	M	2018	67
3	1001595	10226	1.366492e+09	2018-02-28	97110	M25551	283.98	1951-02-27	M	2018	67
4	1001595	10226	1.366492e+09	2018-02-28	97140	M25551	279.34	1951-02-27	M	2018	67
…	…	…	…	…	…	…	…	…	…	…	…
198547	999919	12345	1.962494e+09	2018-05-03	99214	E782	105.49	1947-01-28	F	2018	71
198548	999919	12345	1.962494e+09	2018-05-03	90732	Z23	108.14	1947-01-28	F	2018	71
198549	999919	12345	1.962494e+09	2018-05-03	G0009	Z23	19.91	1947-01-28	F	2018	71
198550	999959	11445	1.548250e+09	2016-09-24	66984	H2512	838.19	1945-04-03	F	2016	71
198551	999959	11445	1.548250e+09	2016-09-24	G8918	H2512	0.00	1945-04-03	F	2016	71

[198552 rows x 11 columns]

2.0.5 0.4 Analyze the top 100 HCPCS/CPT codes

2.0.6 0.4.1 Group by HCPCS/CPT codes and count the number of unique claims IDs in descending order, and take first 100 codes with the most number of claims

[72]:
```
claim_count_per_hcpcs_df = medicare_df.groupby('hcpcs_code').agg(
    uniq_clm_cnt=('claim_id', 'nunique')
).sort_values('uniq_clm_cnt', ascending=False)

claim_count_per_hcpcs_top100_df = claim_count_per_hcpcs_df.head(100)
claim_count_per_hcpcs_top100_df
```

[72]:
```
            uniq_clm_cnt
hcpcs_code
36415               8188
99214               7796
99213               6600
80053               5087
85025               4911
…                   …
74176                336
99222                332
G0283                324
11721                315
84481                314

[100 rows x 1 columns]
```

2.0.7 0.4.2 Look up descriptions of the codes online and categorize them into broader medical fields/activities

Please note the categories might not be medically/officially accurate and are only for educational purposes.

[81]:
```
hcpcs_code_100_df = pd.DataFrame({
    'hcpcs_code': [
        '36415', '99214', '99213', '80053', '85025',
        '80061', '84443', '83036', '80048', '93010',
        '81001', 'G8427', '93000', '88305', '82306',
        'G0463', '87086', '99232', 'G0008', '85610',
        '93306', '1036F', '84439', 'A0425', '99285',
        '81003', '92014', '93005', '71020', '85027',
        '70450', '82607', '71045', '99204', '97110',
        '99212', 'G0202', '77063', '83735', 'G9637',
        '99203', '71010', '90662', '99284', '71046',
        'G0439', '82570', '84484', '84153', '17000',
        '97140', '78452', '99215', '87186', '82043',
        '77067', '87088', '99233', '4040F', '96372',
```

```
        '82550', '83540', 'G8907', 'A0427', '99223',
        'Q9967', '92012', '84550', '81002', 'G8918',
        '92134', '82565', '82728', '17003', '83550',
        '77080', '87077', 'A9270', '7025F', 'G0471',
        '74177', '92015', '85652', '98941', '80076',
        'G8420', '82746', '93880', '77052', '11100',
        'G9551', '86140', 'G0009', '83880', '66984',
        '74176', '99222', 'G0283', '11721', '84481'
    ],
    'description': [
        'Venous Procedures','Established Patient Office or Other Outpatient␣
↪Services','Established Patient Office or Other Outpatient Services','Organ␣
↪or Disease Oriented Panels','Blood count',
        'Organ or Disease Oriented Panels','Chemistry␣
↪Procedures','Hemoglobin','Organ or Disease Oriented␣
↪Panels','Electrocardiogram, routine ECG with at least 12 leads',
        'Urinalysis, by dip stick or tablet reagent for bilirubin, glucose,␣
↪hemoglobin, ketones, leukocytes, nitrite, pH, protein, specific gravity,␣
↪urobilinogen, any number of these constituents','Eligible clinician attests␣
↪to documenting in the medical record they obtained, updated, or reviewed the␣
↪patient\'s current medications','Electrocardiogram, routine ECG with at␣
↪least 12 leads','Surgical pathology, gross and microscopic␣
↪examination','Vitamin D',
        'Hospital outpatient clinic visit for assessment and management of a␣
↪patient','Culture, bacterial','Subsequent Hospital Inpatient or Observation␣
↪Care','Administration of influenza virus vaccine','Prothrombin time',
        'Echocardiography, transthoracic, real-time with image documentation␣
↪(2D), includes M-mode recording, when performed','Patient␣
↪History','Thyroxine','Ground mileage, per statute mile','Emergency␣
↪department visit for the evaluation and management of a patient, which␣
↪requires a medically appropriate history',
        'Urinalysis, by dip stick or tablet reagent for bilirubin, glucose,␣
↪hemoglobin, ketones, leukocytes, nitrite, pH, protein, specific gravity,␣
↪urobilinogen, any number of these constituents','Ophthalmological services:␣
↪medical examination and evaluation, with initiation or continuation of␣
↪diagnostic and treatment program','Electrocardiogram, routine ECG with at␣
↪least 12 leads','DELETED','Blood count',
        'Computed tomography, head or brain','Cyanocobalamin (Vitamin␣
↪B-12)','Radiologic examination, chest','New Patient Office or Other␣
↪Outpatient Services','Therapeutic procedure, 1 or more areas, each 15␣
↪minutes',
```

```
      'Established Patient Office or Other Outpatient Services','Screening␣
↪mammography, bilateral (2-view study of each breast), including␣
↪computer-aided detection (cad) when performed','Breast,␣
↪Mammography','Chemistry Procedures','Final reports with documentation of one␣
↪or more dose reduction techniques (e.g., automated exposure control,␣
↪adjustment of the ma and/or kv according to patient size, use of iterative␣
↪reconstruction technique)',
      'New Patient Office or Other Outpatient Services','DELETED','Influenza␣
↪virus vaccine','Emergency department visit for the evaluation and management␣
↪of a patient, which requires a medically appropriate history','Radiologic␣
↪examination, chest',
      'Annual wellness visit, includes a personalized prevention plan of␣
↪service (pps), subsequent visit','Creatinine','Chemistry␣
↪Procedures','Prostate specific antigen (PSA)','Destruction (eg, laser␣
↪surgery, electrosurgery, cryosurgery, chemosurgery, surgical curettement),␣
↪premalignant lesions (eg, actinic keratoses)',
      'Therapeutic Procedures','Myocardial perfusion imaging, tomographic␣
↪(SPECT) (including attenuation correction, qualitative or quantitative wall␣
↪motion, ejection fraction by first pass or gated technique, additional␣
↪quantification, when performed)','Established Patient Office or Other␣
↪Outpatient Services','Susceptibility studies, antimicrobial agent','Albumin',
      'Breast, Mammography','Culture, bacterial','Subsequent Hospital␣
↪Inpatient or Observation Care','Therapeutic, Preventive or Other␣
↪Interventions','Therapeutic, prophylactic, or diagnostic injection (specify␣
↪substance or drug)',
      'Creatine kinase (CK), (CPK)','Chemistry Procedures','Patient␣
↪documented not to have experienced any of the following events: a burn prior␣
↪to discharge; a fall within the facility; wrong site/side/patient/procedure/
↪implant event; or a hospital transfer or hospital admission upon discharge␣
↪from the facility','Ambulance service, advanced life support, emergency␣
↪transport, level 1 (als 1 - emergency)','New or Established Patient',
      'Low osmolar contrast material, 300-399 mg/ml iodine concentration, per␣
↪ml','Ophthalmological services: medical examination and evaluation, with␣
↪initiation or continuation of diagnostic and treatment program','Uric␣
↪acid','Urinalysis, by dip stick or tablet reagent for bilirubin, glucose,␣
↪hemoglobin, ketones, leukocytes, nitrite, pH, protein, specific gravity,␣
↪urobilinogen, any number of these constituents','Patient without␣
↪preoperative order for iv antibiotic surgical site infection (ssi)␣
↪prophylaxis',
      'Scanning computerized ophthalmic diagnostic imaging, anterior segment,␣
↪with interpretation and report','Creatinine','Chemistry␣
↪Procedures','Destruction (eg, laser surgery, electrosurgery, cryosurgery,␣
↪chemosurgery, surgical curettement), premalignant lesions (eg, actinic␣
↪keratoses)','Chemistry Procedures',
```

```
      'Dual-energy X-ray absorptiometry (DXA), bone density study, 1 or more␣
↪sites','Culture, bacterial','Non-covered item or service','Structural␣
↪Measures','Collection of venous blood by venipuncture or urine sample by␣
↪catheterization from an individual in a skilled nursing facility (snf) or by␣
↪a laboratory on behalf of a home health agency (hha)',
      'Computed tomography, abdomen and pelvis','Special Ophthalmological␣
↪Services and Procedures','Sedimentation rate, erythrocyte','Chiropractic␣
↪manipulative treatment (CMT)','Organ or Disease Oriented Panels',
      'Bmi is documented within normal parameters and no follow-up plan is␣
↪required','Folic acid','Duplex scan of extracranial␣
↪arteries','DELETED','Biopsy of skin, subcutaneous tissue and/or mucous␣
↪membrane (including simple closure), unless otherwise listed',
      'Final reports for imaging studies without an incidentally found lesion␣
↪noted','C-reactive protein','Administration of pneumococcal␣
↪vaccine','Chemistry Procedures','Intraocular Lens Procedures',
      'Computed tomography, abdomen and pelvis','New or Established␣
↪Patient','Electrical stimulation (unattended), to one or more areas for␣
↪indication(s) other than wound care, as part of a therapy plan of␣
↪care','Debridement of nail(s) by any method(s)','Triiodothyronine T3'
  ],
  'category': [
      'Cardiac','Administrative','Administrative','Panels','Blood test',
      'Panels','Chemistry','Blood test','Panels','Cardiac',
      'Urinalysis','Administrative','Cardiac','Pathology','Blood test',
      'Administrative','Pathology','Administrative','Vaccine','Liver',
      'Cardiac','Administrative','Blood test','Others','Administrative',
      'Urinalysis','Ophthalmology','Cardiac','Others','Blood test',
      'Tomography','Blood test','Radiology','Administrative','Therapy',
    ␣
↪'Administrative','Mammography','Mammography','Chemistry','Administrative',
      'Administrative','Others','Vaccine','Administrative','Radiology',
      'Administrative','Blood test','Chemistry','Blood test','Destructive␣
↪surgical procedures',
      'Therapy','Cardiac','Administrative','Pathology','Blood test',
      'Mammography','Pathology','Administrative','Therapy','Therapy',
      'Blood␣
↪test','Chemistry','Administrative','Administrative','Administrative',
      'Radiology','Ophthalmology','Urinalysis','Urinalysis','Administrative',
      'Ophthalmology','Blood test','Chemistry','Destructive surgical␣
↪procedures','Chemistry',
      'Radiology','Pathology','Others','Administrative','Pathology',
      'Tomography','Ophthalmology','Blood test','Chiropractic','Panels',
      'Administrative','Blood test','Radiology','Others','Pathology',
      'Administrative','Blood test','Vaccine','Chemistry','Ophthalmology',
      'Tomography','Administrative','Therapy','Destructive surgical␣
↪procedures','Blood test'
```

```
    ]
})
hcpcs_code_100_df.head(100)
```

[81]:
```
   hcpcs_code                                        description  \
0       36415                                   Venous Procedures
1       99214  Established Patient Office or Other Outpatient…
2       99213  Established Patient Office or Other Outpatient…
3       80053                    Organ or Disease Oriented Panels
4       85025                                         Blood count
..        …                                                   …
95      74176            Computed tomography, abdomen and pelvis
96      99222                          New or Established Patient
97      G0283  Electrical stimulation (unattended), to one or…
98      11721            Debridement of nail(s) by any method(s)
99      84481                                 Triiodothyronine T3

                          category
0                          Cardiac
1                   Administrative
2                   Administrative
3                           Panels
4                       Blood test
..                             …
95                      Tomography
96                  Administrative
97                         Therapy
98  Destructive surgical procedures
99                      Blood test

[100 rows x 3 columns]
```

3 Start of Mid Term Exam

[82]:
```
## Keeping the original medicare_df and hcpcs_code_100_df for posterity
```

[83]:
```
original_medicare_df = medicare_df
```

[76]:
```
unfiltered_hcpcs_code_100_df = hcpcs_code_100_df
```

4 Removing All NULL Values

Instead of having to drop null categories one by one, I dropped all null values

[77]:
```
medicare_df.isnull().sum()
```

[77]:
```
claim_id                0
patient_id              0
npi_id                  44806
claim_date              0
hcpcs_code              12993
diagnosis_code          9783
claim_cost              46886
patient_birth_date      0
patient_gender          0
claim_year              0
patient_age             0
dtype: int64
```

[78]:
```
len(medicare_df)
```

[78]:
```
198552
```

[79]:
```
medicare_df = medicare_df.dropna()
```

[80]:
```
medicare_df.isnull().sum()
```

[80]:
```
claim_id              0
patient_id            0
npi_id                0
claim_date            0
hcpcs_code            0
diagnosis_code        0
claim_cost            0
patient_birth_date    0
patient_gender        0
claim_year            0
patient_age           0
dtype: int64
```

[84]:
```
len(medicare_df)
```

[84]:
```
148688
```

5 Removing ClaimID duplicates

5.0.1 Ensuring no two FULL rows have duplicates

[85]:
```
medicare_df.duplicated().sum()
```

[85]:
```
0
```

5.1 checking and dropping duplicates of ClaimID

[86]:
```
medicare_df['claim_id'].duplicated().sum()
```

[86]:
```
65507
```

[87]:
```
medicare_df = medicare_df.drop_duplicates(subset=['claim_id'])
```

[88]:
```
medicare_df['claim_id'].duplicated().sum()
```

[88]:
```
0
```

[89]:
```
len(medicare_df)
```

[89]:
```
83181
```

5.2 Excluding non-procedural categories (Administrative & Others)

[90]:
```
hcpcs_code_100_df = hcpcs_code_100_df[(hcpcs_code_100_df['category'] !=␣
↪'Administrative') & (hcpcs_code_100_df['category'] != 'Others')]
```

[91]:
```
hcpcs_code_100_df
```

[91]:

	hcpcs_code	description	category
0	36415	Venous Procedures	Cardiac
3	80053	Organ or Disease Oriented Panels	Panels
4	85025	Blood count	Blood test
5	80061	Organ or Disease Oriented Panels	Panels
6	84443	Chemistry Procedures	Chemistry
..	…	…	…
94	66984	Intraocular Lens Procedures	Ophthalmology
95	74176	Computed tomography, abdomen and pelvis	Tomography
97	G0283	Electrical stimulation (unattended), to one or…	Therapy
98	11721	Debridement of nail(s) by any method(s)	Destructive surgical procedures
99	84481	Triiodothyronine T3	Blood test

```
[72 rows x 3 columns]
```

[92]:
```
len(medicare_df[medicare_df['hcpcs_code'] == '78452'])
```

[92]:
```
413
```

5.2.1 Making NPI_ID into String for easier viewing of NPI_ID

[93]:
```
medicare_df['npi_id'] = medicare_df['npi_id'].astype(str)
```

```
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/3764644550.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  medicare_df['npi_id'] = medicare_df['npi_id'].astype(str)
```

5.3 Q1. Based on the trends for the share of CVM claims as a percentage of total claims over the years 2016 through 2018, what are some business insights you can gather? What are some additional analyses you could do based on these trends?

[94]:
```
len(medicare_df['claim_id'])
```

[94]:
```
83181
```

5.3.1 Removing years other than 2016, 2017, 2018

[95]:
```
medicare_df = medicare_df[medicare_df['claim_year'].isin(['2016','2017',␣
↪'2018'])]
```

Checking

[96]:
```
medicare_df[medicare_df['claim_year'] == '2014']
```

[96]:
```
Empty DataFrame
Columns: [claim_id, patient_id, npi_id, claim_date, hcpcs_code, diagnosis_code,
claim_cost, patient_birth_date, patient_gender, claim_year, patient_age]
Index: []
```

[97]:
```
cardiac_hcpcs_codes = hcpcs_code_100_df[hcpcs_code_100_df['category'] ==␣
↪'Cardiac']['hcpcs_code']
```

[98]:
```
cardiac_hcpcs_codes
```

[98]:
```
0     36415
9     93010
12    93000
20    93306
27    93005
51    78452
Name: hcpcs_code, dtype: object
```

[99]:
```
cardiac_medicare = medicare_df[medicare_df['hcpcs_code'].
↪isin(cardiac_hcpcs_codes)]
```

[100]:
```
cardiac_medicare
```

[100]:

	claim_id	patient_id	npi_id	claim_date	hcpcs_code	diagnosis_code	claim_cost	patient_birth_date	patient_gender	claim_year	patient_age
6	1011605	10163	1578545943.0	2018-01-02	36415	C439	49.51	1944-12-25	M	2018	74
11	1016102	10017	1306804737.0	2018-01-25	36415	Z79899	78.46	1984-11-23	M	2018	34
34	104681	10174	1114949963.0	2016-11-19	36415	J441	71.49	1954-04-13	M	2016	62
36	1049025	10200	1538126966.0	2017-02-21	36415	Z7901	12.12	1931-04-20	M	2017	86
40	1054205	10200	1649262882.0	2016-02-03	36415	I4891	11.19	1931-04-20	M	2016	85
…	…	…	…	…	…	…	…	…	…	…	…
198436	998390	12634	1104010131.0	2017-12-15	93306	R011	222.72	1925-01-09	F	2017	92
198451	998607	10000	1881646099.0	2016-08-17	36415	Z13220	3.00	1952-06-20	F		
198471	998886	11923	1326018839.0	2016-02-25	93306	R079	170.16	1947-01-05	F		
198478	998994	1262	1932166386.0	2018-06-14	36415	I10	3.00	1948-07-25	M		
198526	999618	11357	1962518530.0	2017-11-12	93010	I10	8.81	1951-10-10	F		

```
198451       2016           64
198471       2016           69
198478       2018           70
198526       2017           66

[8049 rows x 11 columns]
```

5.3.2 Percentage of CVM claims

[101]:
```
len(cardiac_medicare)
```

[101]:
```
8049
```

[102]:
```
len(medicare_df)
```

[102]:
```
83181
```

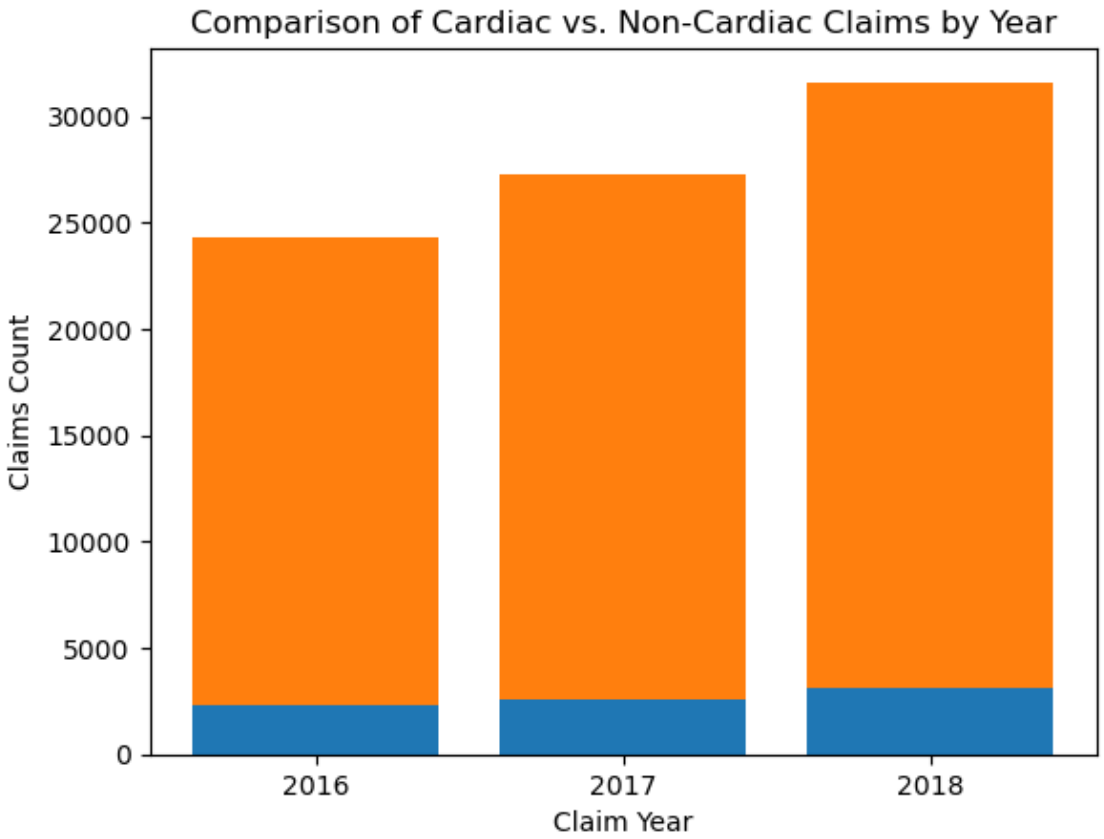
[103]:
```
print((len(cardiac_medicare)/len(medicare_df)) * 100, '%')
```

```
9.676488621199553 %
```

[104]:
```
total_counts = medicare_df['claim_year'].value_counts().sort_index()
cardiac_counts = cardiac_medicare['claim_year'].value_counts().sort_index()
non_cardiac_counts = total_counts - cardiac_counts.reindex(total_counts.index).
  ↪fillna(0)
percentage_increase = cardiac_counts.pct_change() * 100
percentage_increase = percentage_increase.round(2)

print("Percentage Increase of Cardiac Claims per Year:")
print(percentage_increase)
```

```
Percentage Increase of Cardiac Claims per Year:
claim_year
2016      NaN
2017    12.25
2018    20.25
Name: count, dtype: float64
```

[105]:
```
plt.bar(total_counts.index, cardiac_counts, label = 'Cardiac')
plt.bar(total_counts.index, non_cardiac_counts, bottom= cardiac_counts, label =␣
  ↪'non-Cardiac')

plt.xlabel('Claim Year')
plt.ylabel('Claims Count')
plt.title("Comparison of Cardiac vs. Non-Cardiac Claims by Year")

plt.show()
```



[ ]:

5.4 Q2. Evaluate the HCP behavior in context of claim volume from 2016-2018. How many HCPs are submitting 1 CVM claim; how many HCPs are associated with more than 10 claims, etc.? Once you perform this analysis, explain how this trend can influence the sales force deployment. That is, how would you segment the HCPs and how would you allocate In-Person (sales force) vs Non-Personal Promotions (NPP, i.e. Emails, Social Media, Digital etc.) efforts?

[106]:
```
medicare_df['npi_id'] = medicare_df['npi_id'].astype(str)
```

[107]:
```
medicare_df['npi_id'].nunique()
```

[107]: 46162

[108]:
```
HCP_claim_count = medicare_df.sort_index()
```

[109]:
```
HCP_claim_count['npi_id'].value_counts()
```

[109]:
```
npi_id
1538144910.0    988
1619913449.0    661
1063497451.0    598
1659352276.0    594
1518903350.0    513
                …
1508855529.0      1
1003987967.0      1
1124061361.0      1
1164645016.0      1
1881665362.0      1
Name: count, Length: 46162, dtype: int64
```

[110]:
```
cardiac_medicare['npi_id'].value_counts()
```

[110]:
```
npi_id
1619913449.0    94
1346233277.0    64
1326104613.0    57
1245307818.0    54
1932145778.0    51
                ..
1356352025.0     1
1992719421.0     1
1023086147.0     1
1518963362.0     1
1962518530.0     1
Name: count, Length: 5599, dtype: int64
```

[111]:
```
disease_aware = (cardiac_medicare['npi_id'].value_counts() == 1).sum()
trialist = ((cardiac_medicare['npi_id'].value_counts() >= 2 ) &␣
 ↪(cardiac_medicare['npi_id'].value_counts() <= 4)).sum()
rising_stars = ((cardiac_medicare['npi_id'].value_counts() >= 5) &␣
 ↪(cardiac_medicare['npi_id'].value_counts() <= 9)).sum()
high_volume_stars = (cardiac_medicare['npi_id'].value_counts() >= 10).sum()
print('disease_aware:', disease_aware)
print('trialist:', trialist)
print('rising_stars:', rising_stars)
print('high_volume_stars', high_volume_stars)
```

```
disease_aware: 4911
trialist: 579
rising_stars: 50
high_volume_stars 59
```

[112]:
```
cardiac_medicare['claim_date'] = pd.to_datetime(cardiac_medicare['claim_date'])

cardiac_medicare['claim_year'] = cardiac_medicare['claim_date'].dt.year
cardiac_medicare['claim_quarter'] = cardiac_medicare['claim_date'].dt.
 ↪to_period('Q')
```

```
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/2964846158.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cardiac_medicare['claim_date'] =
pd.to_datetime(cardiac_medicare['claim_date'])
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/2964846158.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cardiac_medicare['claim_year'] = cardiac_medicare['claim_date'].dt.year
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/2964846158.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cardiac_medicare['claim_quarter'] =
cardiac_medicare['claim_date'].dt.to_period('Q')
```

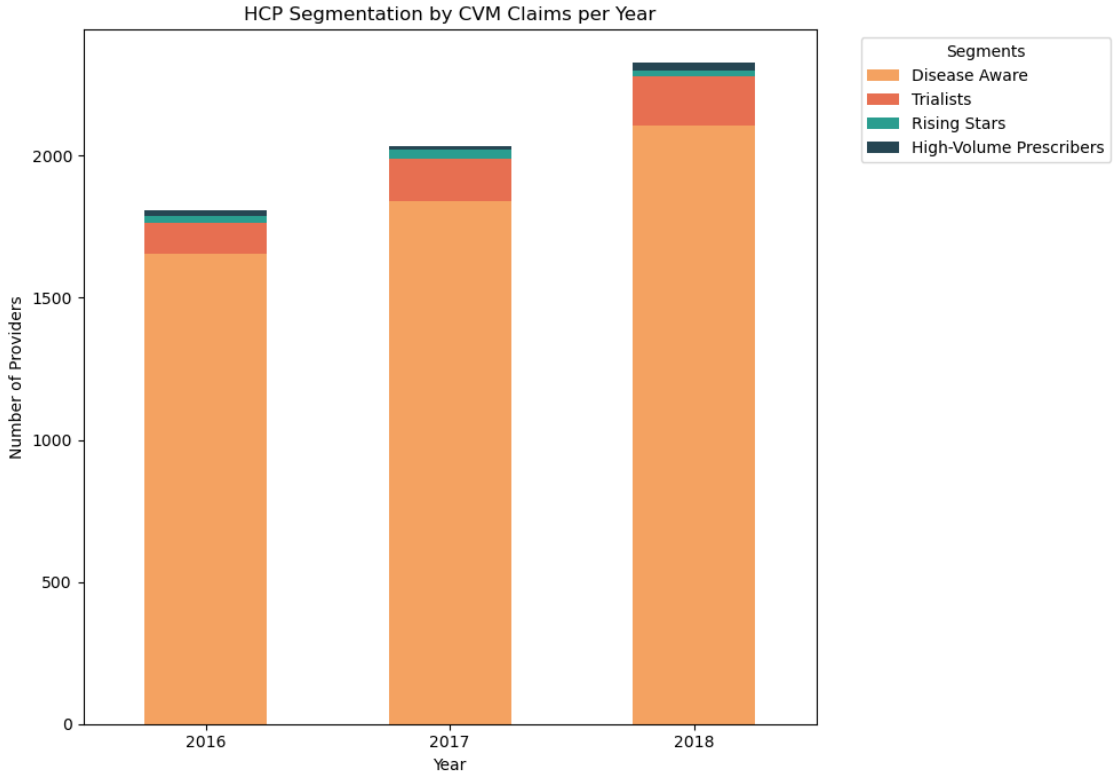
[113]:
```
segments = ['Disease Aware', 'Trialists', 'Rising Stars', 'High-Volume␣
 ↪Prescribers']
segment_counts_per_year = pd.DataFrame(columns=segments, index=[2016, 2017,␣
 ↪2018])

for year in [2016, 2017, 2018]:
    cardiac_medicare_year = cardiac_medicare[cardiac_medicare['claim_year'] ==␣
 ↪year]

    hcp_claim_counts = cardiac_medicare_year.groupby('npi_id')['claim_id'].
 ↪nunique()

    disease_aware = (hcp_claim_counts == 1).sum()
    trialists = ((hcp_claim_counts >= 2) & (hcp_claim_counts <= 4)).sum()
```

```
    rising_stars = ((hcp_claim_counts >= 5) & (hcp_claim_counts <= 9)).sum()
    high_volume_prescribers = (hcp_claim_counts >= 10).sum()

    segment_counts_per_year.loc[year, 'Disease Aware'] = disease_aware
    segment_counts_per_year.loc[year, 'Trialists'] = trialists
    segment_counts_per_year.loc[year, 'Rising Stars'] = rising_stars
    segment_counts_per_year.loc[year, 'High-Volume Prescribers'] =␣
 ↪high_volume_prescribers
```

[114]:
```
print("Counts per Segment per Year:")
print(segment_counts_per_year)
```

```
Counts per Segment per Year:
     Disease Aware Trialists Rising Stars High-Volume Prescribers
2016          1654       109           25                       19
2017          1841       149           31                       14
2018          2106       174           20                       27
```

[115]:
```
segment_counts_per_year = segment_counts_per_year.astype(int)

segment_counts_per_year.plot(kind='bar', stacked=True, figsize=(10,7),␣
 ↪color=['#f4a261', '#e76f51', '#2a9d8f', '#264653'])

plt.xlabel('Year')
plt.ylabel('Number of Providers')
plt.title('HCP Segmentation by CVM Claims per Year')
plt.legend(title='Segments', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



6 Extra Experimentation: Trying to figure out how much does each category contributes in claim cost

[116]:
```
cardiac_medicare['claim_cost'] = pd.to_numeric(cardiac_medicare['claim_cost'],␣
↪errors='coerce')

hcp_claim_costs = cardiac_medicare.groupby(['claim_year',␣
↪'npi_id'])['claim_cost'].sum().reset_index()
```

```
/var/folders/zk/ffvbnsts2dg8tf_rqkmdm36m0000gn/T/ipykernel_63105/3064055160.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cardiac_medicare['claim_cost'] = pd.to_numeric(cardiac_medicare['claim_cost'],
errors='coerce')
```

[117]:
```
segment_claim_costs_per_year = pd.DataFrame(columns=segments, index=[2016,␣
↪2017, 2018])
```

```
for year in [2016, 2017, 2018]:
    cardiac_medicare_year = cardiac_medicare[cardiac_medicare['claim_year'] ==␣
 ↪year]

    hcp_claim_counts = cardiac_medicare_year.groupby('npi_id')['claim_id'].
 ↪nunique()

    hcp_data = hcp_claim_counts.to_frame('claim_count').reset_index()
    hcp_costs = hcp_claim_costs[hcp_claim_costs['claim_year'] == year]

    hcp_data = pd.merge(hcp_data, hcp_costs[['npi_id', 'claim_cost']],␣
 ↪on='npi_id', how='left')

    def assign_segment(count):
        if count == 1:
            return 'Disease Aware'
        elif 2 <= count <= 4:
            return 'Trialists'
        elif 5 <= count <= 9:
            return 'Rising Stars'
        else:
            return 'High-Volume Prescribers'

    hcp_data['Segment'] = hcp_data['claim_count'].apply(assign_segment)

    segment_costs = hcp_data.groupby('Segment')['claim_cost'].sum()

    for segment in segments:
        segment_claim_costs_per_year.loc[year, segment] = segment_costs.
 ↪get(segment, 0)
```

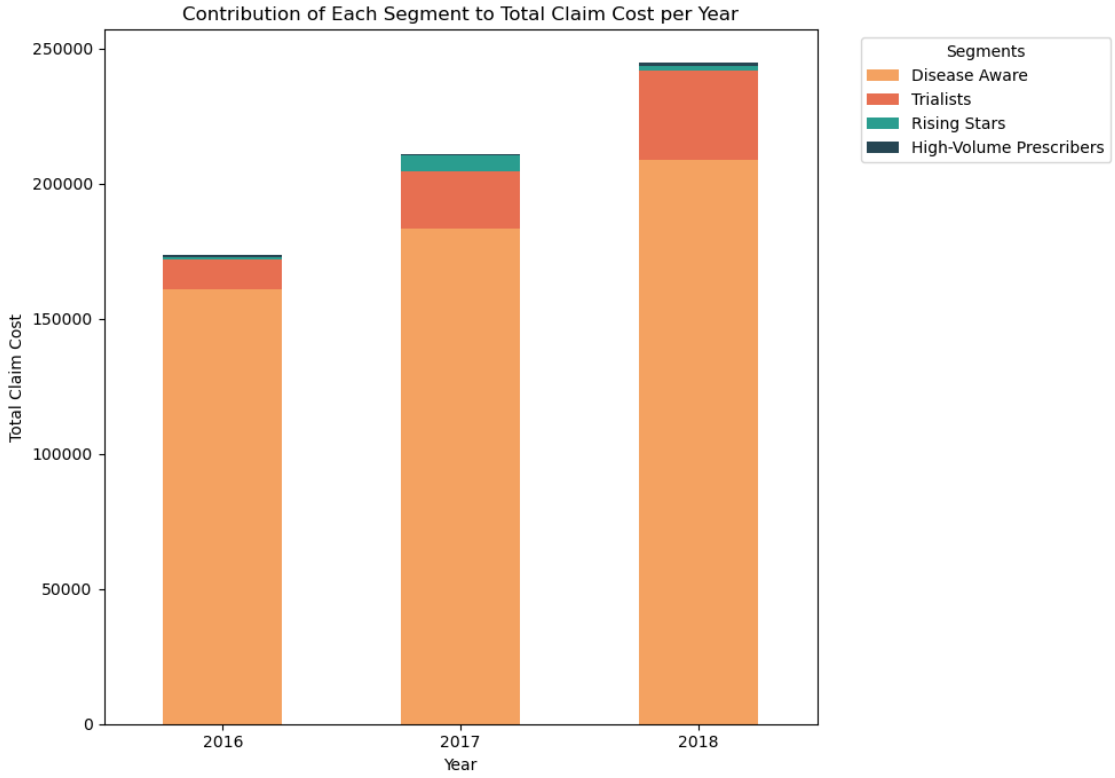
[118]:
```
segment_claim_costs_per_year = segment_claim_costs_per_year.astype(float)
```

[119]:
```
print("Claim Costs per Segment per Year:")
print(segment_claim_costs_per_year)
```

```
Claim Costs per Segment per Year:
      Disease Aware  Trialists  Rising Stars  High-Volume Prescribers
2016      161116.41   10785.10        953.43                    708.0
2017      183489.71   20883.69       5941.42                    651.0
2018      208810.39   33208.92       1359.16                   1362.0
```

[120]:
```
segment_claim_costs_per_year.plot(
    kind='bar',
    stacked=True,
    figsize=(10, 7),
```

```
    color=['#f4a261', '#e76f51', '#2a9d8f', '#264653']
)

plt.xlabel('Year')
plt.ylabel('Total Claim Cost')
plt.title('Contribution of Each Segment to Total Claim Cost per Year')
plt.legend(title='Segments', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



6.1 Q3. Evaluate the Patient Age demographics in the context of claim volume from 2016-2018. Bucket the patients into groups based on their age and explain the trends. How would you position the Marketing Budgets and the Promotions with respect to the changing landscape of the CVM claims and the respective patient segments?

[121]:
```
medicare_df['patient_age'] = pd.to_numeric(medicare_df['patient_age'])
```

[132]:
```
segment_one = ((medicare_df['patient_age'] >= 18) & (medicare_df['patient_age']␣
↪<= 59)).sum()
```

```
segment_two = ((medicare_df['patient_age'] >= 60) & (medicare_df['patient_age']␣
 ↪<= 69)).sum()
segment_three = ((medicare_df['patient_age'] >= 70) &␣
 ↪(medicare_df['patient_age'] <= 79)).sum()
segment_four = (medicare_df['patient_age'] >= 80).sum()
segment_five = (medicare_df['patient_age'] <= 18).sum()
```

[133]:
```
print('segment_one: ',segment_one)
print('segment_two: ',segment_two)
print('segment_three: ',segment_three)
print('segment_four: ',segment_four)
print('segment_five: ',segment_five)
```

```
segment_one:  8835
segment_two:  26379
segment_three:  27774
segment_four:  18654
segment_five:  1539
```

[124]:
```
def assign_age_segment(age):
    if 18 <= age <= 59:
        return '18-59'
    elif 60 <= age <= 69:
        return '60-69'
    elif 70 <= age <= 79:
        return '70-79'
    elif age >= 80:
        return '80+'
    else:
        return 'Under 18'

medicare_df['age_segment'] = medicare_df['patient_age'].
 ↪apply(assign_age_segment)
```

[130]:
```
claims_by_year_segment = medicare_df.groupby(['claim_year', 'age_segment']).
 ↪size().reset_index(name='claim_count')
claims_pivot = claims_by_year_segment.pivot(index='claim_year',␣
 ↪columns='age_segment', values='claim_count').fillna(0)
print(claims_pivot)
```

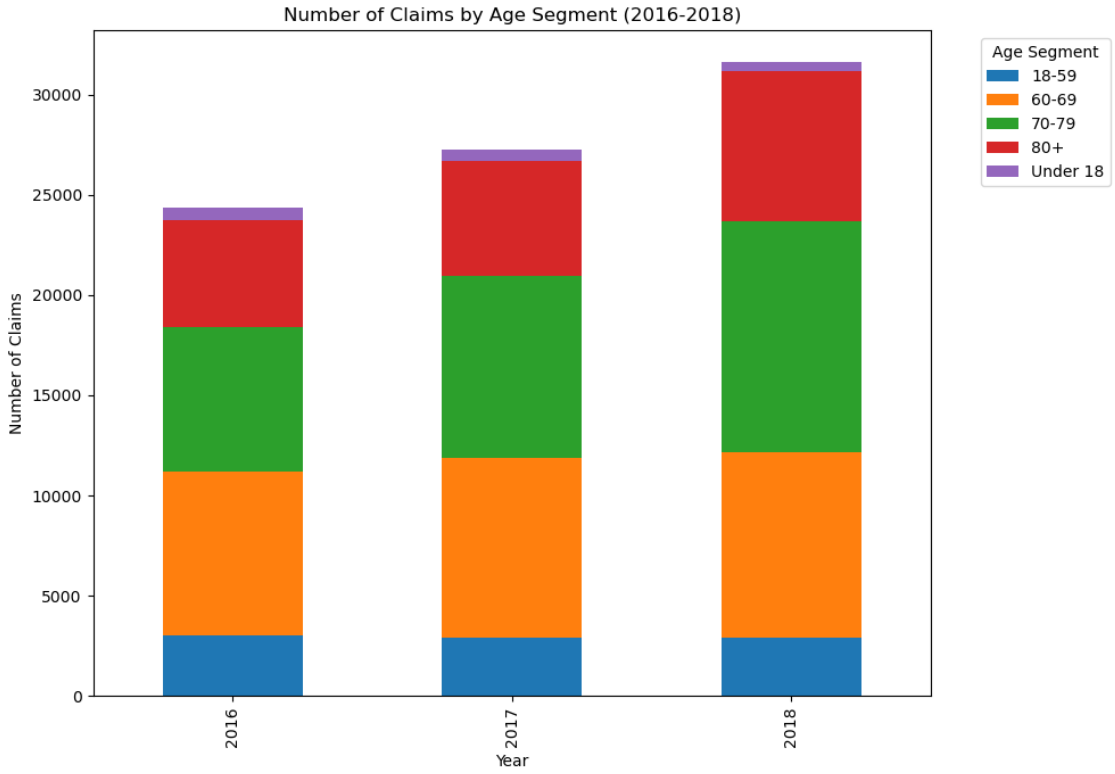
```
age_segment  18-59  60-69  70-79   80+  Under 18
claim_year
2016          2999   8191   7195  5365       583
2017          2908   8958   9087  5756       526
2018          2928   9230  11492  7533       430
```

[127]:
```
claims_pct_change = claims_pivot.pct_change().fillna(0) * 100
```

```
claims_pct_change = claims_pct_change.round(2)

print(claims_pct_change)
```

```
age_segment  18-59  60-69  70-79    80+  Under 18
claim_year
2016          0.00   0.00   0.00   0.00      0.00
2017         -3.03   9.36  26.30   7.29     -9.78
2018          0.69   3.04  26.47  30.87    -18.25
```

[128]:
```
claims_pivot.plot(kind='bar', stacked=True, figsize=(10,7))

plt.xlabel('Year')
plt.ylabel('Number of Claims')
plt.title('Number of Claims by Age Segment (2016-2018)')
plt.legend(title='Age Segment', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```



[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: