Reading the data

First things first... Let's read the parquet file and take a look at what's inside

In [ ]:
```
# reading the CSVs
import pandas as pd
import numpy as np

df = pd.read_parquet('../data/dataset.gz.parquet')
policy_data = df.copy() # to keep raw data untouched
```

In [ ]:
```
policy_data.head()
```

Out[ ]:

	policy_holder_zipcode	policy_id	policy_start_date	policy_exposure_days	policy_claims_num_reported	policy_claims_num_paid	po
0	1000.0	22036576975396	20171229	58	0.0	0.0	
1	1001.0	01472343000	20180530	214	0.0	0.0	
2	1001.0	08024270000	20180307	298	0.0	0.0	
3	1003.0	166871902448270	20180428	246	0.0	0.0	
4	1004.0	13975805000	20170918	259	0.0	0.0	

In [ ]:
```
policy_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3024172 entries, 0 to 3024171
Data columns (total 20 columns):
 #   Column                             Dtype  
---  ------                             -----  
 0   policy_holder_zipcode              float64
 1   policy_id                          object 
 2   policy_start_date                  object 
 3   policy_exposure_days               int32  
 4   policy_claims_num_reported         float64
 5   policy_claims_num_paid             float64
 6   policy_claims_total_amount_paid_brl  float64
 7   policy_premium_received_brl        float64
 8   policy_holder_birth_date           int32  
 9   policy_holder_gender               object 
 10  policy_holder_bonus_clas           float64
 11  policy_holder_residence_latitude   float64
 12  policy_holder_residence_longitude  float64
 13  vehicle_brand                      object 
 14  vehicle_model                      object 
 15  vehicle_make_year                  float64
 16  vehicle_tarif_class                object 
 17  vehicle_value_brl                  float64
 18  policy_holder_residence_region     object 
 19  policy_holder_residence_city       object 
dtypes: float64(10), int32(2), object(8)
memory usage: 438.4+ MB
```

Not sure why, but the df.info() command didn't show me the count of NULL values. So here it is:

In [ ]:
```
policy_data.isnull().sum()
```

Out[ ]:
```
policy_holder_zipcode                     0
policy_id                                 0
policy_start_date                         0
policy_exposure_days                      0
policy_claims_num_reported                0
policy_claims_num_paid                    0
policy_claims_total_amount_paid_brl       0
policy_premium_received_brl               0
policy_holder_birth_date                  0
policy_holder_gender                      0
policy_holder_bonus_clas             277313
policy_holder_residence_latitude          0
policy_holder_residence_longitude         0
vehicle_brand                             0
vehicle_model                             0
vehicle_make_year                         0
vehicle_tarif_class                       0
vehicle_value_brl                         0
policy_holder_residence_region            0
policy_holder_residence_city              0
dtype: int64
```

Preparing the data

There are a few fields that I want to convert to better data types.

In [ ]:

```
# weird way to check if I can safely convert to int the fields I want
# extracting the decimal places and validating if they're all 0s
print(policy_data["policy_holder_zipcode"].apply(lambda x: abs(x % 1)).sum())
print(policy_data["policy_claims_num_reported"].apply(lambda x: abs(x % 1)).sum())
print(policy_data["policy_claims_num_paid"].apply(lambda x: abs(x % 1)).sum())
print(policy_data["policy_holder_bonus_clas"].apply(lambda x: abs(x % 1)).sum())
print(policy_data["vehicle_make_year"].apply(lambda x: abs(x % 1)).sum())
```

```
0.0
0.0
0.0
0.0
0.0
```

In [ ]:

```
# converting to timestamps
policy_data["policy_start_date"] = pd.to_datetime(policy_data["policy_start_date"])
policy_data["policy_holder_birth_date"] = pd.to_datetime(policy_data["policy_holder_birth_date"].astype(str))
policy_data["vehicle_make_year"] = pd.to_datetime(policy_data["vehicle_make_year"].astype(int), format='%Y')

# converting to string
policy_data["policy_holder_zipcode"] = policy_data["policy_holder_zipcode"].astype(int).astype(str)
policy_data["policy_holder_bonus_clas"] = policy_data["policy_holder_bonus_clas"].astype('Int64').astype(str) # using

# converting to int
policy_data["policy_claims_num_reported"] = policy_data["policy_claims_num_reported"].astype(int)
policy_data["policy_claims_num_paid"] = policy_data["policy_claims_num_paid"].astype(int)

# reordering columns
policy_data = policy_data[["policy_id",
                           "policy_start_date",
                           "policy_exposure_days",
                           "policy_premium_received_brl",
                           "policy_claims_num_reported",
                           "policy_claims_num_paid",
                           "policy_claims_total_amount_paid_brl",
                           "policy_holder_birth_date",
                           "policy_holder_gender",
                           "policy_holder_residence_city",
                           "policy_holder_residence_region",
                           "policy_holder_zipcode",
                           "policy_holder_residence_latitude",
                           "policy_holder_residence_longitude",
                           "policy_holder_bonus_clas",
                           "vehicle_brand",
                           "vehicle_model",
                           "vehicle_make_year",
                           "vehicle_tarif_class",
                           "vehicle_value_brl"
                          ]]

policy_data.head()
```

Out[ ]:

	policy_id	policy_start_date	policy_exposure_days	policy_premium_received_brl	policy_claims_num_reported	policy_claims_num_pa
0	22036576975396	2017-12-29	58	777.432433	0	
1	01472343000	2018-05-30	214	619.134247	0	
2	08024270000	2018-03-07	298	1295.687671	0	
3	166871902448270	2018-04-28	246	2971.898688	0	
4	13975805000	2017-09-18	259	1847.060274	0	

Final adjustments

Much better... Not even sure if all those transformation were actually necessary, but it is always better to tell python the correct data types. That might be helpful later when creating the graphs. Also, reordering the columns helps me to think when looking at a data sample.

Now I feel ready for the analysis itself.

After reading the loss ratio concept and the proposed excercise, it is really important to notice that the average of individual loss ratios is different than the total loss ratio for a specific group.

In [ ]:
```
# individual loss_ratio calculation
policy_data["policy_loss_ratio"] = policy_data["policy_claims_total_amount_paid_brl"]/policy_data["policy_premium_rece
```

Descriptive statistics

Running some basic descriptive statistics on all columns to better understand the data:

In [ ]:
```
# Dates
policy_data.describe(include=[np.datetime64], datetime_is_numeric=True)
```

Out[ ]:

	policy_start_date	policy_holder_birth_date	vehicle_make_year
count	3024172	3024172	3024172
mean	2018-01-13 20:11:56.892557568	1969-12-21 10:53:28.684162140	2012-04-02 04:38:36.534905600
min	2017-01-02 00:00:00	1900-01-01 00:00:00	1985-01-01 00:00:00
25%	2017-08-12 00:00:00	1960-01-13 00:00:00	2010-01-01 00:00:00
50%	2018-01-19 00:00:00	1972-01-12 00:00:00	2013-01-01 00:00:00
75%	2018-06-15 00:00:00	1981-10-31 00:00:00	2015-01-01 00:00:00
max	2018-12-30 00:00:00	2009-03-09 00:00:00	2018-01-01 00:00:00

In [ ]:
```
# Possible dimensions for the market analysis
policy_data.describe(include=[object])
```

Out[ ]:

	policy_id	policy_holder_gender	policy_holder_residence_city	policy_holder_residence_region	policy_holder_zipcode	policy_hol
count	3024172	3024172	3024172	3024172	3024172	
unique	3024172	2	831	52	7544	
top	22036576975396	M	SÃO PAULO	METROPOLITANA DE SÃO PAULO	15104	
freq	1	1514773	830938	1171145	12012	

In [ ]:
```
# Numerical values
policy_data.describe(include=[np.number])
```

Out[ ]:

	policy_exposure_days	policy_premium_received_brl	policy_claims_num_reported	policy_claims_num_paid	policy_claims_total_amount_p
count	3.024172e+06	3.024172e+06	3.024172e+06	3.024172e+06	3.0241
mean	1.393440e+02	4.956020e+02	2.545788e-02	1.379816e-02	1.7036
std	9.240343e+01	5.129012e+02	1.667383e-01	1.195834e-01	2.3283
min	1.000000e+00	1.598174e-03	0.000000e+00	0.000000e+00	0.0000
25%	6.100000e+01	1.670872e+02	0.000000e+00	0.000000e+00	0.0000
50%	1.350000e+02	3.730192e+02	0.000000e+00	0.000000e+00	0.0000
75%	1.830000e+02	6.603041e+02	0.000000e+00	0.000000e+00	0.0000
max	3.640000e+02	4.726853e+04	5.000000e+00	5.000000e+00	4.7308

Data accuracy check

I noticed that there are some birth dates dated 1900 (a bit odd). Let me double check that:

In [ ]:
```
policy_data["policy_holder_birth_year"] = pd.DatetimeIndex(policy_data["policy_holder_birth_date"]).year
elderly_sample = policy_data[policy_data["policy_holder_birth_year"] < 1930] # older than ~90
elderly_sample["policy_holder_birth_year"].value_counts()
```

Out[ ]:
```
1900    36748
1929      860
1928      618
1927      440
1926      314
1925      215
1924      168
1923       59
1922       42
1920       31
1921       17
1919        8
1918        3
1909        1
1917        1
1916        1
Name: policy_holder_birth_year, dtype: int64
```

Indeed a significant number of policies is being assigned (very likely incorrectly) birth dates on 1900. To be considered in case we do anything using birth dates.

Exploratory data analysis

First I'll create a separated dataframe with only the relevant columns. I could run the analysis with more than just the columns defined here, but for the sake of time I won't. We could for example create age groups and use that for the investigation.

In [ ]:
```
loss_ratio_analysis = policy_data[["policy_premium_received_brl",
                                   "policy_claims_total_amount_paid_brl",
                                  #  "policy_loss_ratio",
                                  #  "policy_holder_birth_year",
                                   "policy_holder_gender",
                                   "policy_holder_residence_city",
                                   "policy_holder_residence_region",
                                   # "policy_holder_zipcode",
                                   # "policy_holder_residence_latitude",
                                   # "policy_holder_residence_longitude",
                                   "policy_holder_bonus_clas",
                                   "vehicle_brand",
                                   # "vehicle_model",
                                   # "vehicle_make_year",
                                  #  "vehicle_tarif_class",
                                 ]]
```

In [ ]:
```
# overall loss_ratio
print(loss_ratio_analysis["policy_claims_total_amount_paid_brl"].sum()/loss_ratio_analysis["policy_premium_received_br

# loss_ratio by gender
grouped = loss_ratio_analysis.groupby(by=["policy_holder_gender"])
claims_paid = grouped["policy_claims_total_amount_paid_brl"].agg(["sum","count"])
premium_received = grouped["policy_premium_received_brl"].agg(["sum","count"])
gender_loss_ratio = pd.DataFrame({"loss_ratio": claims_paid["sum"]/premium_received["sum"], "policy_count": premium_re

gender_loss_ratio.sort_values(by="loss_ratio")
```

```
0.343762287767103
```

Out[ ]:

policy_holder_gender	loss_ratio	policy_count
F	0.335485	1509399
M	0.351023	1514773

In [ ]:
```
# loss_ratio by region
grouped = loss_ratio_analysis.groupby(by=["policy_holder_residence_region"])
claims_paid = grouped["policy_claims_total_amount_paid_brl"].agg(["sum","count"])
premium_received = grouped["policy_premium_received_brl"].agg(["sum","count"])
region_loss_ratio = pd.DataFrame({"loss_ratio": claims_paid["sum"]/premium_received["sum"], "policy_count": premium_re

region_loss_ratio[region_loss_ratio["policy_count"]>200].sort_values(by="loss_ratio")
```

Out[ ]:

policy_holder_residence_region	loss_ratio	policy_count
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	0.271126	175391
VALE DO RIBEIRA	0.294500	22223
METROPOLITANA DE SÃO PAULO	0.298284	1171145
BAIXADA SANTISTA	0.301878	32972
VALE DO PARAÍBA	0.339484	146224
LITORAL NORTE DE SÃO PAULO	0.355432	19618
CAMPINAS - CIDADE	0.364066	111630
SOROCABA - REGIÃO 2	0.388032	286221
SÃO JOSÉ DOS CAMPOS, JACAREÍ E CAÇAPAVA	0.390653	78354
SOROCABA - CIDADE E REGIÃO	0.395878	206973
DEMAIS CAMPINAS 1	0.399128	122634
DEMAIS INTERIOR DE SÃO PAULO	0.440934	466646
DEMAIS CAMPINAS 2	0.443516	49010
RIBEIRÃO PRETO E BAURU	0.513604	79474
TRIÂNGULO MINEIRO	0.537145	55145

In [ ]:
```
# loss_ratio by city
grouped = loss_ratio_analysis.groupby(by=["policy_holder_residence_region", "policy_holder_residence_city"])
claims_paid = grouped["policy_claims_total_amount_paid_brl"].agg(["sum","count"])
premium_received = grouped["policy_premium_received_brl"].agg(["sum","count"])
city_loss_ratio = pd.DataFrame({"loss_ratio": claims_paid["sum"]/premium_received["sum"], "policy_count": premium_rece

city_loss_ratio[city_loss_ratio["policy_count"]>20000].sort_values(by="loss_ratio")
```

Out[ ]:

policy_holder_residence_region	policy_holder_residence_city	loss_ratio	policy_count
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	SANTO ANDRÉ	0.248020	58222
	SÃO CAETANO DO SUL	0.254973	32920
METROPOLITANA DE SÃO PAULO	FRANCISCO MORATO	0.264812	62674
	GUARULHOS	0.285109	69050
	SÃO PAULO	0.291184	830938
	OSASCO	0.296750	37616
SOROCABA - REGIÃO 2	RIO CLARO	0.317469	20372
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	SÃO BERNARDO DO CAMPO	0.324643	47434
METROPOLITANA DE SÃO PAULO	SANTOS	0.327721	35384
VALE DO PARAÍBA	MOGI DAS CRUZES	0.346613	34474
METROPOLITANA DE SÃO PAULO	COTIA	0.351926	20833
VALE DO PARAÍBA	TAUBATÉ	0.358231	21143
SOROCABA - REGIÃO 2	LIMEIRA	0.358909	22973
CAMPINAS - CIDADE	CAMPINAS	0.364066	111630
SOROCABA - CIDADE E REGIÃO	SOROCABA	0.379884	48183
	PIRACICABA	0.388847	33779
	SANTA BARBARA D´OESTE	0.391652	27106
SÃO JOSÉ DOS CAMPOS, JACAREÍ E CAÇAPAVA	SÃO JOSÉ DOS CAMPOS	0.396622	54399
DEMAIS INTERIOR DE SÃO PAULO	SÃO CARLOS	0.416696	27947
DEMAIS CAMPINAS 1	JUNDIAÍ	0.423737	64086
METROPOLITANA DE SÃO PAULO	BARUERI	0.433111	20493
SOROCABA - CIDADE E REGIÃO	AMERICANA	0.434413	34531
DEMAIS INTERIOR DE SÃO PAULO	ARARAQUARA	0.445116	38005
RIBEIRÃO PRETO E BAURU	RIBEIRÃO PRETO	0.505484	43890
	BAURU	0.527669	35584
TRIÂNGULO MINEIRO	SÃO JOSÉ DO RIO PRETO	0.539188	54967
DEMAIS INTERIOR DE SÃO PAULO	FRANCA	0.603913	20892

In [ ]:

```
# loss_ratio by bonus_clas
grouped = loss_ratio_analysis.groupby(by=["policy_holder_bonus_clas"])
claims_paid = grouped["policy_claims_total_amount_paid_brl"].agg(["sum","count"])
premium_received = grouped["policy_premium_received_brl"].agg(["sum","count"])
bonus_clas_loss_ratio = pd.DataFrame({"loss_ratio": claims_paid["sum"]/premium_received["sum"], "policy_count": premiu

bonus_clas_loss_ratio.sort_values(by="loss_ratio")
```

Out[ ]:

policy_holder_bonus_clas	loss_ratio	policy_count
4	0.316034	309238
8	0.317195	246696
5	0.320635	301888
6	0.324383	274402
3	0.324830	317333
1	0.348932	343213
2	0.351345	318956
<NA>	0.356339	277313
0	0.384800	569156
9	0.422604	65977

In [ ]:

```
# loss_ratio by vehicle_brand
grouped = loss_ratio_analysis.groupby(by=["vehicle_brand"])
claims_paid = grouped["policy_claims_total_amount_paid_brl"].agg(["sum","count"])
premium_received = grouped["policy_premium_received_brl"].agg(["sum","count"])
vehicle_brand_loss_ratio = pd.DataFrame({"loss_ratio": claims_paid["sum"]/premium_received["sum"], "policy_count": pre

vehicle_brand_loss_ratio[vehicle_brand_loss_ratio["policy_count"]>500].sort_values(by="loss_ratio")
```

Out[ ]:

vehicle_brand	loss_ratio	policy_count
LIFAN	0.156527	1790
Porsche	0.200447	967
Volvo	0.230154	2428
Fiat	0.279694	374685
VW - VolksWagen	0.298954	459760
MINI	0.305272	5470
Subaru	0.310842	1824
Ford	0.315429	345901
Hyundai	0.320025	186434
Renault	0.331389	173196
GM - Chevrolet	0.338579	702968
Toyota	0.359907	152122
Kia Motors	0.360137	15555
Nissan	0.363973	78293
CHERY	0.367081	6322
Peugeot	0.386987	60134
Citroën	0.392443	90122
Honda	0.392901	274628
Jeep	0.398356	15645
smart	0.415184	1380
JAC	0.445067	5188
Suzuki	0.468758	2426
Mitsubishi	0.487905	10740
Chrysler	0.505601	930
BMW	0.509360	19367
Audi	0.554111	14084
Mercedes-Benz	0.579760	19598
Jaguar	0.595251	813

Potential segment opportunities

These give us some indication of market groups that we could investigate further. I'd need to refresh my memory and python skills, but I'm sure there are robust statistical techniques (ANOVA, K-Means clustering, etc) that could be used to better analyse the data.

We'd need to check the data's variance and distribution to actually identify what a "small loss_ratio" would be, but for now I could list the groups we identified (using arbitrary min sample sizes) with loss_ratio less than 30%:

In [ ]:

```
#region
small_region_loss_ratio = region_loss_ratio[region_loss_ratio["policy_count"]>200].sort_values(by="loss_ratio")
small_region_loss_ratio = small_region_loss_ratio[region_loss_ratio["loss_ratio"]<0.3]
small_region_loss_ratio
```

```
/var/folders/wf/z578m42550bgsqj4zdbtlxjw0000gn/T/ipykernel_17576/593868242.py:3: UserWarning: Boolean Series key will
be reindexed to match DataFrame index.
  small_region_loss_ratio = small_region_loss_ratio[region_loss_ratio["loss_ratio"]<0.3]
```

Out[ ]:

policy_holder_residence_region	loss_ratio	policy_count
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	0.271126	175391
VALE DO RIBEIRA	0.294500	22223
METROPOLITANA DE SÃO PAULO	0.298284	1171145

In [ ]:

```
#city
small_city_loss_ratio = city_loss_ratio[city_loss_ratio["policy_count"]>10000].sort_values(by="loss_ratio")
small_city_loss_ratio = small_city_loss_ratio[city_loss_ratio["loss_ratio"]<0.3]
small_city_loss_ratio
```

```
/var/folders/wf/z578m42550bgsqj4zdbtlxjw0000gn/T/ipykernel_17576/356288018.py:3: UserWarning: Boolean Series key will
be reindexed to match DataFrame index.
  small_city_loss_ratio = small_city_loss_ratio[city_loss_ratio["loss_ratio"]<0.3]
```

Out[ ]:

policy_holder_residence_region	policy_holder_residence_city	loss_ratio	policy_count
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	DIADEMA	0.218144	15726
BAIXADA SANTISTA	PRAIA GRANDE	0.241926	12435
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	SANTO ANDRÉ	0.248020	58222
	SÃO CAETANO DO SUL	0.254973	32920
METROPOLITANA DE SÃO PAULO	FRANCISCO MORATO	0.264812	62674
VALE DO PARAÍBA	SUZANO	0.265059	16741
ABCD, SANTO ANDRÉ, SÃO BERNARDO, SÃO CAETANO, DIADEMA	MAUÁ	0.265470	13939
SOROCABA - REGIÃO 2	SALTO	0.268131	10699
METROPOLITANA DE SÃO PAULO	CARAPICUÍBA	0.275971	14757
	GUARULHOS	0.285109	69050
DEMAIS CAMPINAS 1	SUMARÉ	0.290910	13652
METROPOLITANA DE SÃO PAULO	SÃO PAULO	0.291184	830938
	OSASCO	0.296750	37616

In [ ]:

```
#vehicle_brand
small_vehicle_brand_loss_ratio = vehicle_brand_loss_ratio[vehicle_brand_loss_ratio["policy_count"]>500].sort_values(by
small_vehicle_brand_loss_ratio = small_vehicle_brand_loss_ratio[vehicle_brand_loss_ratio["loss_ratio"]<0.3]
small_vehicle_brand_loss_ratio
```

```
/var/folders/wf/z578m42550bgsqj4zdbtlxjw0000gn/T/ipykernel_17576/3055018983.py:3: UserWarning: Boolean Series key will
be reindexed to match DataFrame index.
  small_vehicle_brand_loss_ratio = small_vehicle_brand_loss_ratio[vehicle_brand_loss_ratio["loss_ratio"]<0.3]
```

Out[ ]:

vehicle_brand	loss_ratio	policy_count
LIFAN	0.156527	1790
Porsche	0.200447	967
Volvo	0.230154	2428
Fiat	0.279694	374685
VW - VolksWagen	0.298954	459760