1. *Ocorrências aeronáuticas na aviação civil brasileira*

Introdução a Banco de Dados - 2019/2 - Turma TM

In [1]:

```
# Criacao da base de dados
!python database_creator.py
```

In [2]:

```
# Basic imports
import seaborn
import sqlite3
import numpy as np
import pandas as pd
from matplotlib import pyplot
```

In [3]:

```
pd.set_option('display.max_columns', None)
seaborn.set_style("darkgrid")
seaborn.set_context("paper", rc={"font.size":10,"axes.titlesize":10,"axes.labelsiz
e":8})
```

2. Membros

- Gabriel Luz - 2017015126
- Elves M. Rodrigues - 2017014987
- Luiz Henrique Melo - 2017014464
- Otavio Augusto Silva - 2016006808

3. Descrição dos dados

3.1 Preâmbulo

Neste tópico será feita uma amostragem dos dados por meio da biblioteca Pandas do Python e observações do tipo dos dados que serão utilizados, tais como dados faltantes, além da promoção da junção das tabelas em uma única, a fim de melhor organização dos dados para futura instanciação.

Os dados têm como fonte o Portal Brasileiro de Dados Abertos, podendo ser acessado por aqui (http://dados.gov.br/dataset/ocorrencias-aeronauticas-da-aviacao-civil-brasileira).

A imagem abaixa foi extraída da URL original da fonte dos dados e provém uma melhor descrição dos dados, tal como a sua fonte e seu schema.

3.2 Amostragem dos Dados

Aeronaves Envolvidas em Ocorrências

In [4]:

```
anv = pd.read_csv('../data/anv.csv', sep='~')
anv
```

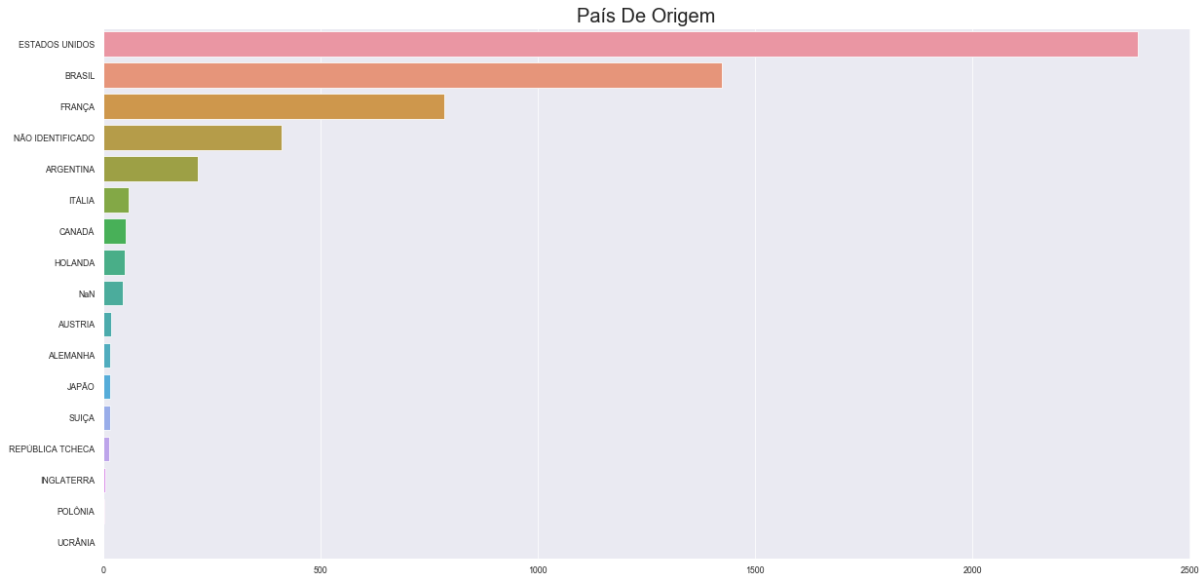
Out[4]:

	codigo_ocorrencia	aeronave_matricula	aeronave_operador_categoria	aeronave_tipo_veiculo
0	201106142171203	PPGXE	AEROCLUBE	AVIÃO
1	201205209591320	PTRBN	OPERADOR DE AERONAVE	AVIÃO
2	201012015549851	PTKUK	OPERADOR DE AERONAVE	AVIÃO
3	201708190325167	PTKUK	OPERADOR PARTICULAR	AVIÃO
4	201803182255192	PPGSZ	AEROCLUBE	AVIÃO
...	...	...	...	...
5518	201903271943019	PRFKU	OPERADOR PARTICULAR	AVIÃO
5519	201904241650016	PPSEK	OPERADOR PARTICULAR	HELICÓPTERO
5520	201904261718061	N100QR	OPERADOR PARTICULAR	AVIÃO
5521	201905061442135	PRCGF	OPERADOR DE AERONAVE	HELICÓPTERO
5522	201905171937061	LVGVB	OPERADOR DE AERONAVE	AVIÃO

5523 rows × 25 columns

In [5]:

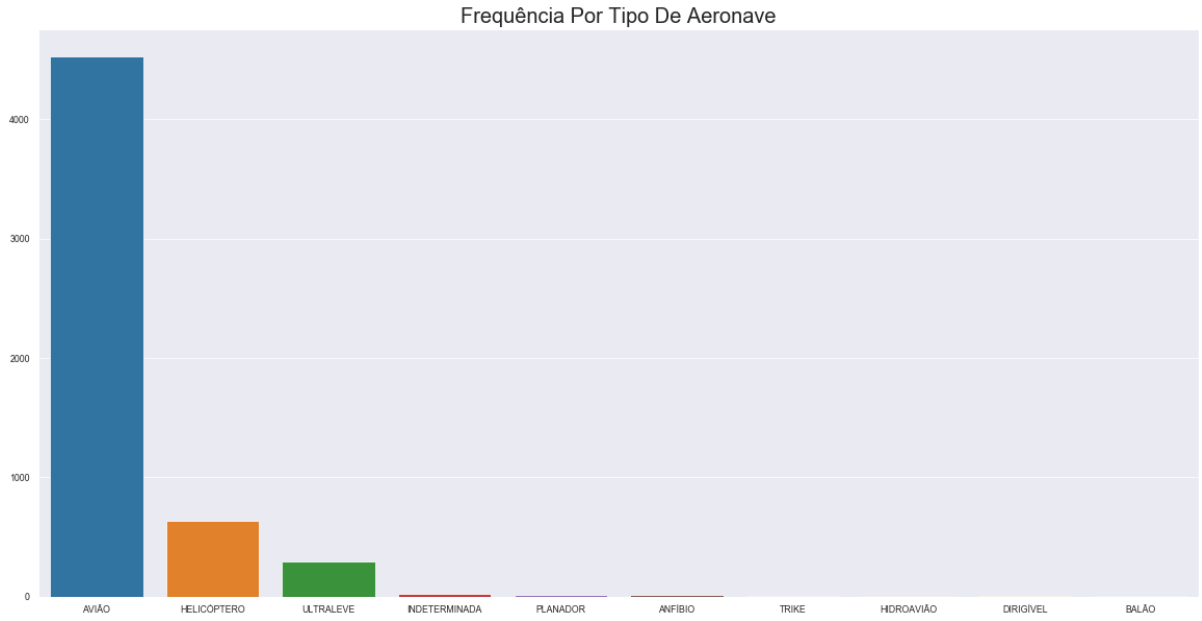
```
col = anv["aeronave_pais_fabricante"].dropna()
shape = anv["aeronave_pais_fabricante"].shape[0] - col.shape[0]
s, n = np.unique(col.to_numpy(), return_counts=True)
s = np.append(s, "NaN")
n = np.append(n, shape)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(x=n[indices], y=s[indices], orient='h')
axis.set_title("País De Origem", fontsize=20)
pyplot.show()
```



Como é possível observar pelo gráfico acima, a grande maioria das aeronaves envolvidas em ocorrências são provenientes dos Estados Unidos, do Brasil e da Fraça. Também é possível perceber uma quantidade considerável de aeronaves com país de origem registrado como "Não Identificado".

In [6]:

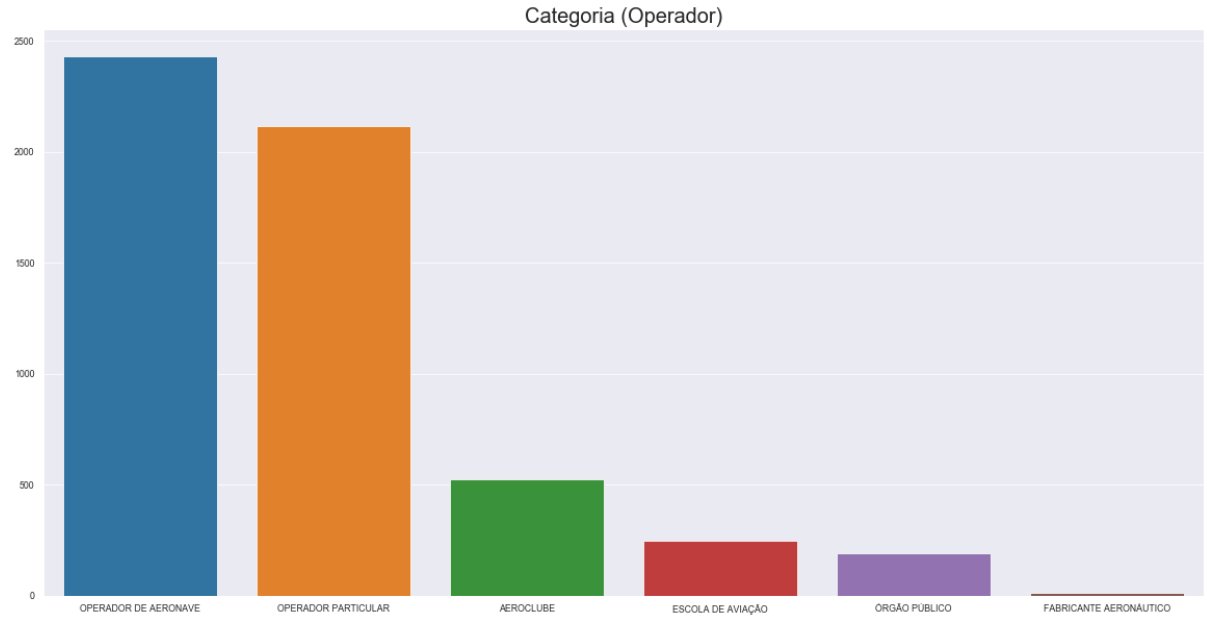
```
s, n = np.unique(anv["aeronave_tipo_veiculo"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Frequência Por Tipo De Aeronave", fontsize=20)
pyplot.show()
```



De acordo com o gráfico acima, a grande maioria das aeronaves envolvidas em ocorrências são aviões.

In [7]:

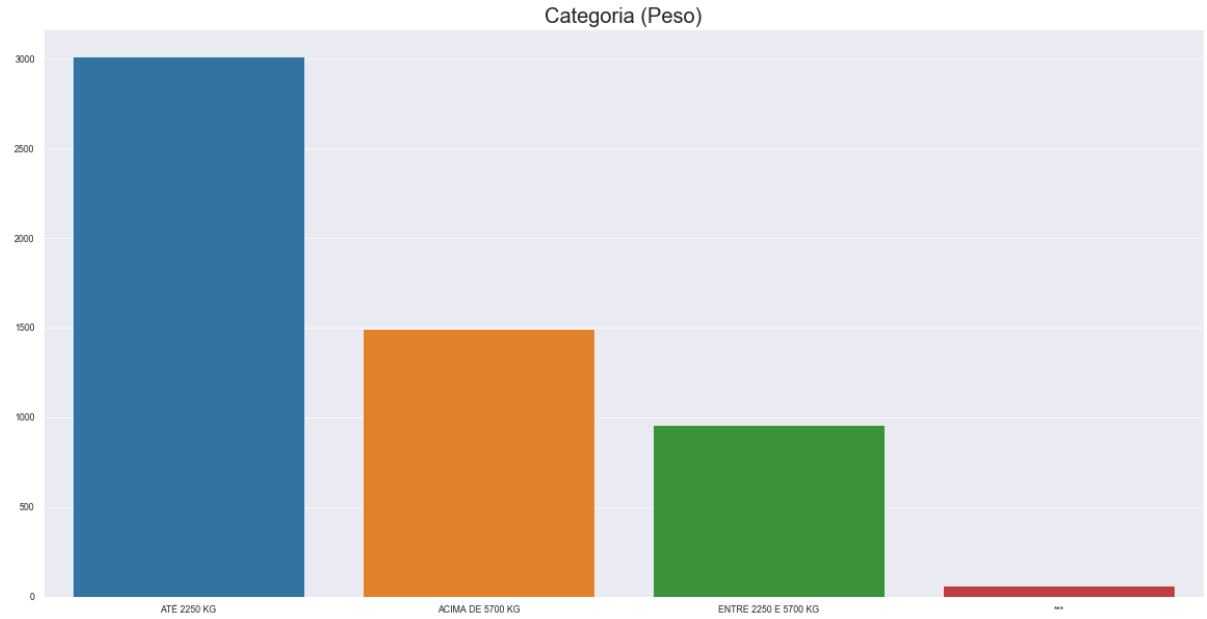
```
s, n = np.unique(anv["aeronave_operador_categoria"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Categoria (Operador)", fontsize=20)
pyplot.show()
```



Como é possível observar com o gráfico acima, boa parte dos operadores eram operadores de aeronaves ou operadores particulares.

In [8]:

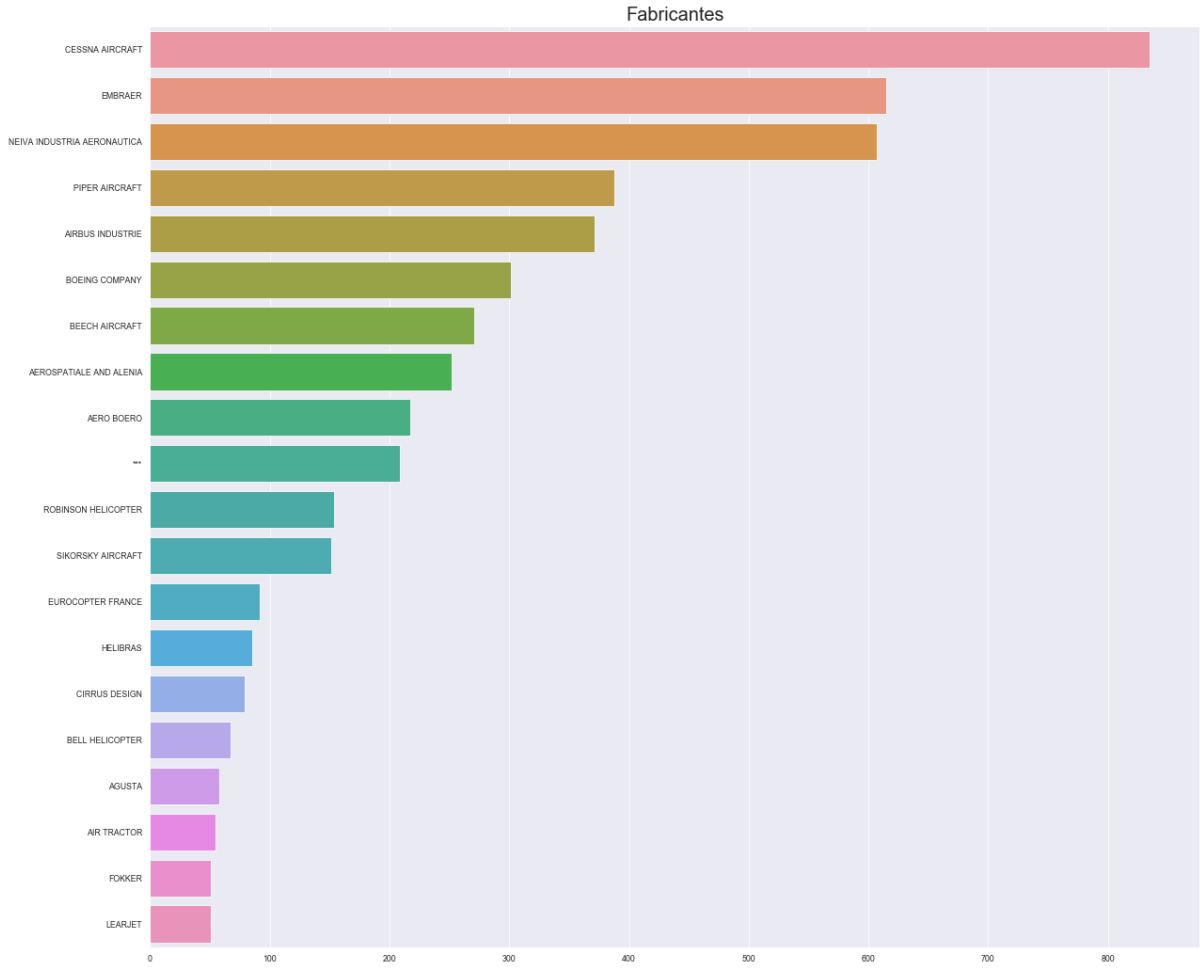
```
s, n = np.unique(anv["aeronave_pmd_categoria"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Categoria (Peso)", fontsize=20)
pyplot.show()
```



De acordo com o gráfico acima, a maioria das aeronaves envolvidas em ocorrências eram de peso até 2250 quilos, seguido por aeronaves acima de 5700 quilos e em terceiro lugar aeronaves entre 2250 e 5700 quilos.

In [9]:

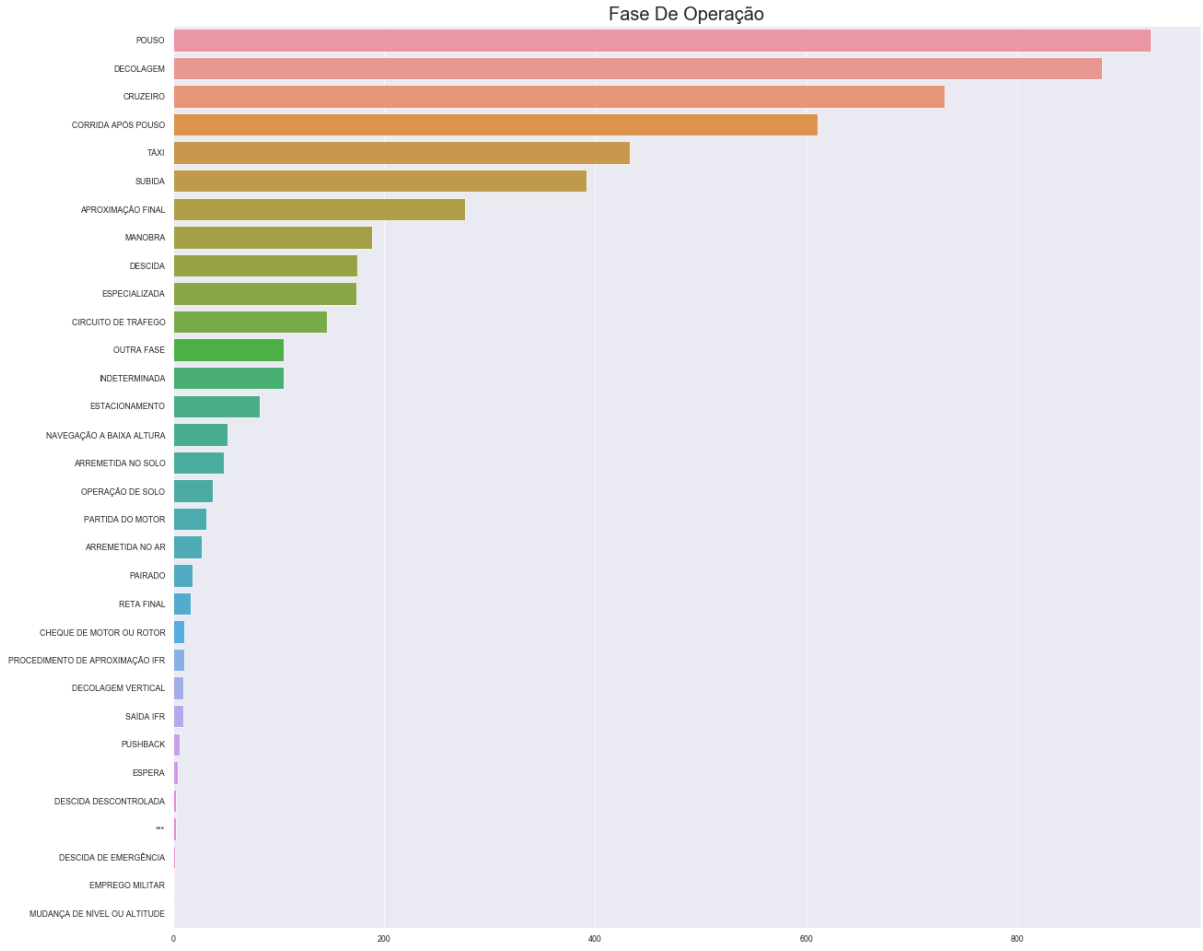
```
s, n = np.unique(anv["aeronave_fabricante"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 18))
seaborn.barplot(x=n[indices][:20], y=s[indices][:20], orient='h')
axis.set_title("Fabricantes", fontsize=20)
pyplot.show()
```



Como é possível observar, as empresas mais presentes na fabricação de aeronaves com ocorrências são a *Cessna Aircraft*, *Embraer* e *Neiva Indústria Aeronáutica*.

In [10]:

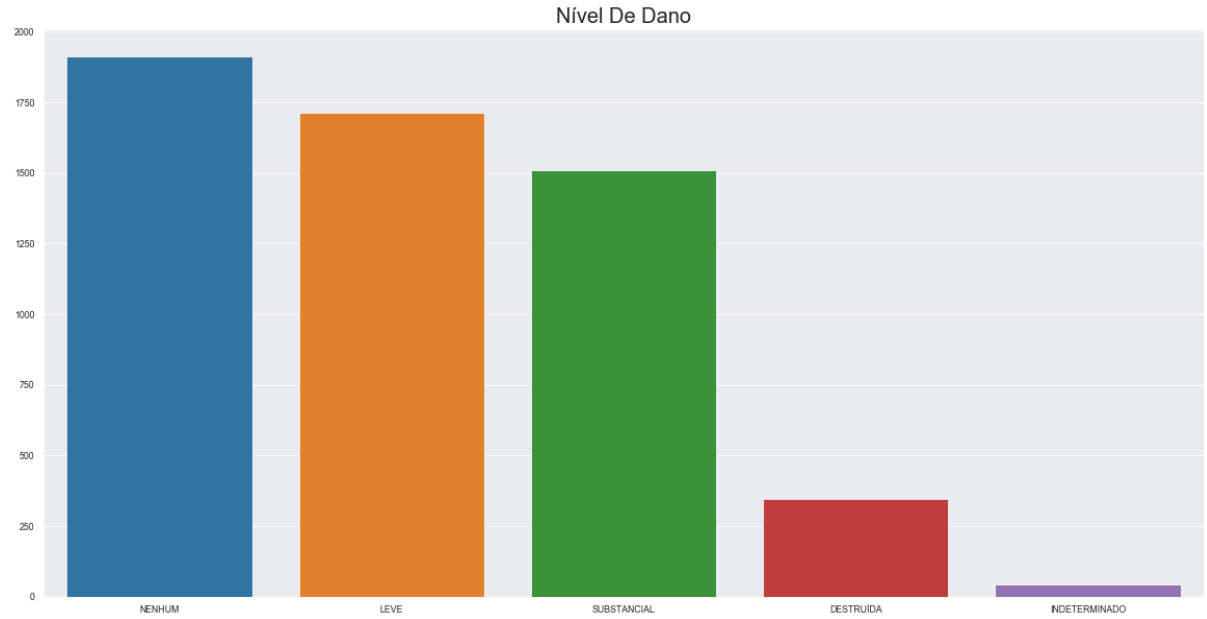
```
s, n = np.unique(anv["aeronave_fase_operacao"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 18))
seaborn.barplot(x=n[indices], y=s[indices], orient='h')
axis.set_title("Fase De Operação", fontsize=20)
pyplot.show()
```



É possível observar com o gráfico acima que a maior parte das ocorrências ocorre durante as operações de pouso, decolagem e cruzeiro.

In [11]:

```
s, n = np.unique(anv["aeronave_nivel_dano"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Nível De Dano", fontsize=20)
pyplot.show()
```



Como é possível observar, a grande maioria das ocorrências envolve nenhum dano, dano leve ou substancial.

Fatores contribuintes

In [12]:

```
ftc = pd.read_csv('../data/ftc.csv', sep='~')
ftc
```

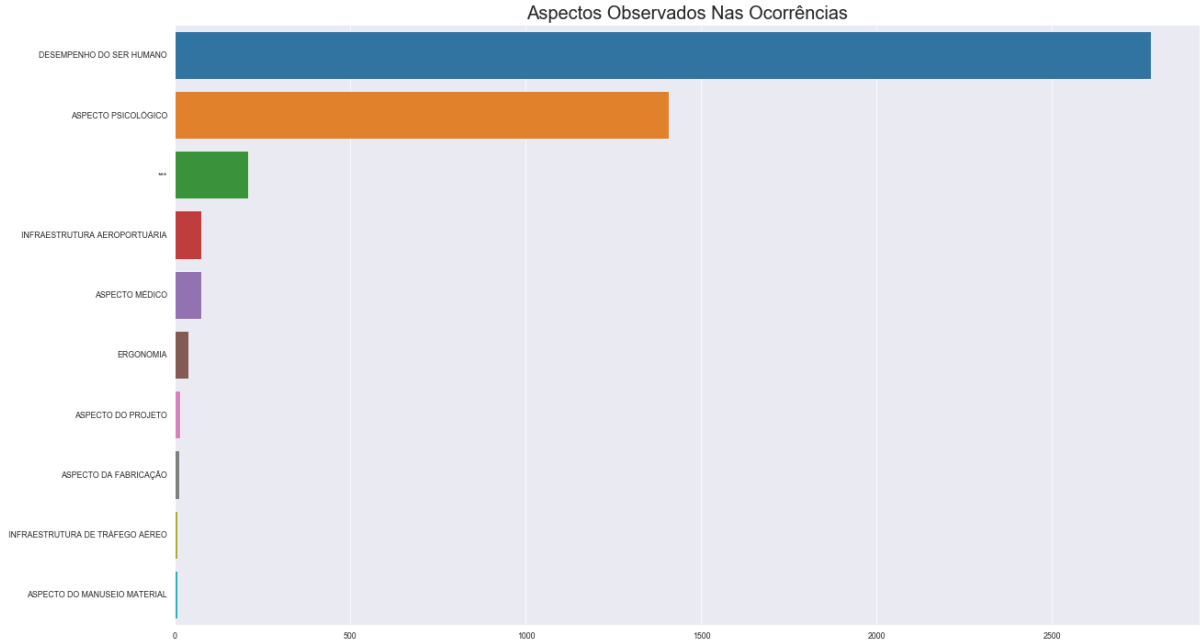
Out[12]:

	codigo_ocorrencia	fator_nome	fator_aspecto	fator_condicionante	fator_area
0	200901015424167	JULGAMENTO DE PILOTAGEM	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL
1	200901015424167	MANUTENÇÃO DE AERONAVE	DESEMPENHO DO SER HUMANO	MANUTENÇÃO DA AERONAVE	FATOR OPERACIONAL
2	200901015424167	SUPERVISÃO GERENCIAL	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL
3	200901055963381	ATITUDE	ASPECTO PSICOLÓGICO	INDIVIDUAL	FATOR HUMANO
4	200901055963381	PROCESSO DECISÓRIO	ASPECTO PSICOLÓGICO	INDIVIDUAL	FATOR HUMANO
...	...	...	...	...	...
4625	201901141204402	DESORIENTAÇÃO	ASPECTO MÉDICO	***	FATOR HUMANO
4626	201901141204402	APLICAÇÃO DE COMANDOS	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL
4627	201901141204402	CONDIÇÕES METEOROLÓGICAS ADVERSAS	***	***	***
4628	201901141204402	JULGAMENTO DE PILOTAGEM	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL
4629	201901141204402	POUCA EXPERIÊNCIA DO PILOTO	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL

4630 rows × 7 columns

In [13]:

```
s, n = np.unique(ftc["fator_aspecto"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 12))
seaborn.barplot(x=n[indices], y=s[indices], orient='h')
axis.set_title("Aspectos Observados Nas Ocorrências", fontsize=20)
pyplot.show()
```



Como observado acima, os aspectos observados mais relevantes nas ocorrências são desempenho do ser humano e o aspecto psicológico.

Ocorrências dos acidentes

In [14]:

```
oco = pd.read_csv('../data/oco.csv', sep='~')
oco
```

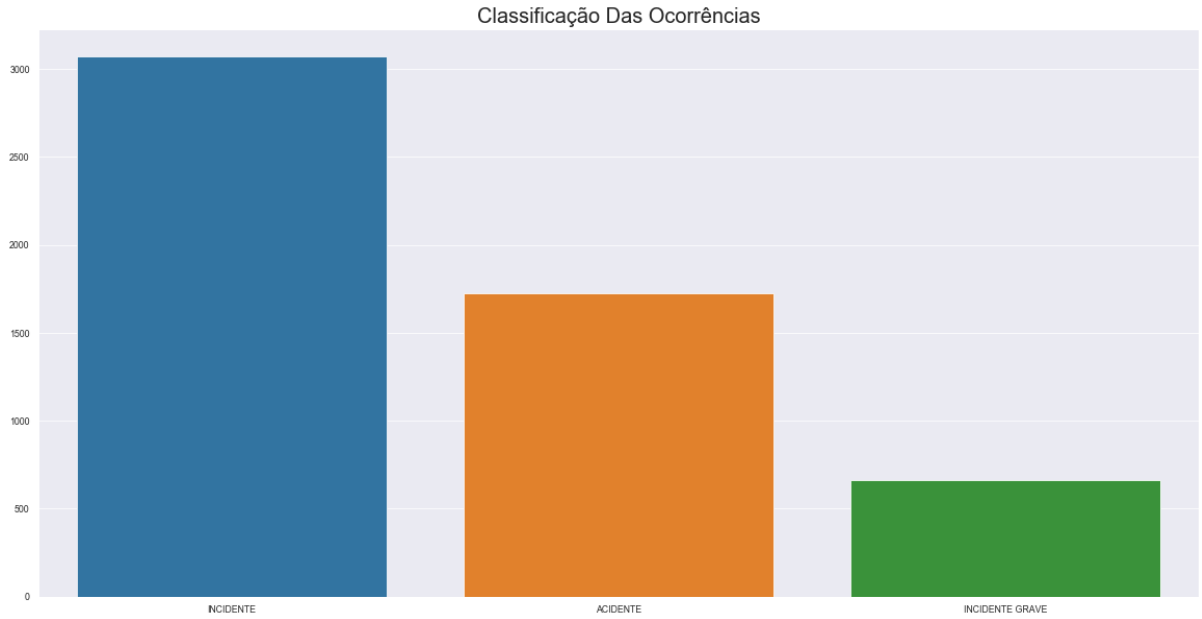
Out[14]:

	codigo_ocorrencia	ocorrencia_classificacao	ocorrencia_tipo	ocorrencia_tipo_categoria	oc
0	201305055424986	ACIDENTE	FALHA DO MOTOR EM VOO	FALHA OU MAU FUNCIONAMENTO DO MOTOR	
1	201311259977425	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
2	201605160250139	INCIDENTE GRAVE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	
3	201805021421302	INCIDENTE	AERÓDROMO	AERÓDROMO	
4	201103187273112	INCIDENTE	OUTROS	OUTROS	
...	...	...	...	...	
5455	201107163514591	INCIDENTE	TRÁFEGO AÉREO	PERDA DE SEPARAÇÃO / COLISÃO EM VOO	
5456	201101107541931	INCIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	OUTROS	
5457	201502247981603	INCIDENTE GRAVE	COM PARA-BRISAS / JANELA / PORTA	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	
5458	200910311058203	INCIDENTE	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	
5459	201309012098180	ACIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	OUTROS	

5460 rows × 22 columns

In [15]:

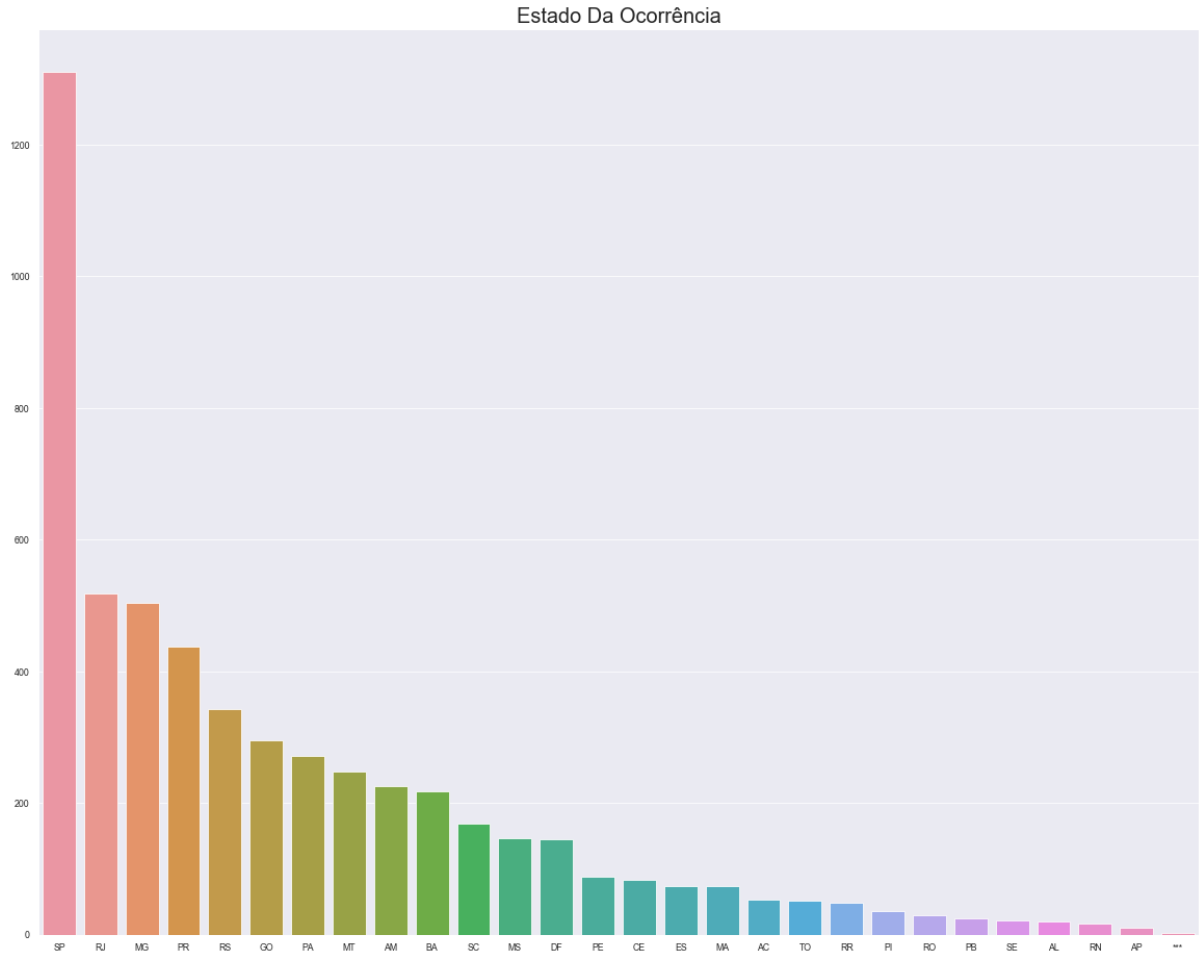
```
s, n = np.unique(oco["ocorrencia_classificacao"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Classificação Das Ocorrências", fontsize=20)
pyplot.show()
```



De acordo com o gráfico acima, a maior parte das ocorrências está classificada como incidente.

In [16]:

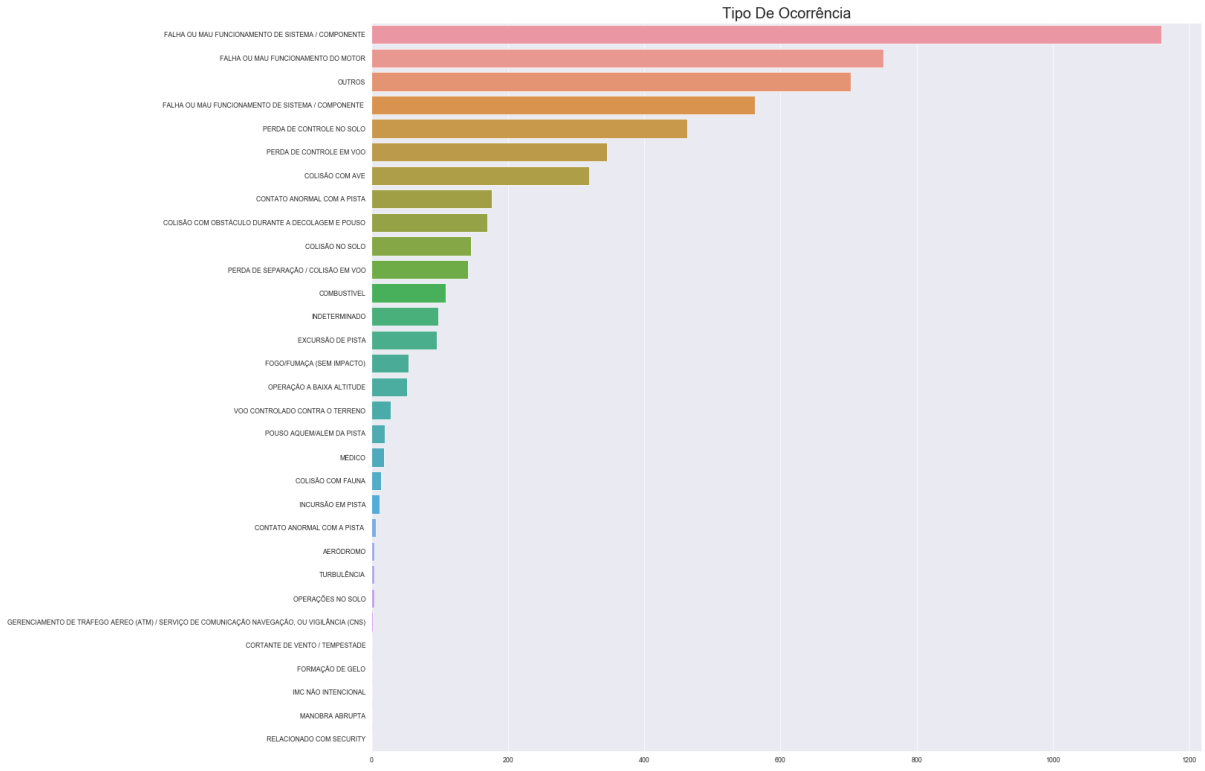
```
s, n = np.unique(oco["ocorrencia_uf"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 16))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Estado Da Ocorrência", fontsize=20)
pyplot.show()
```



Como se pode observar com o gráfico acima, o estado com maior ocorrências é São Paulo, com uma margem considerável.

In [17]:

```
s, n = np.unique(oco["ocorrencia_tipo_categoria"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 18))
seaborn.barplot(x=n[indices], y=s[indices], orient='h')
axis.set_title("Tipo De Ocorrência", fontsize=20)
pyplot.show()
```



Como se pode observar acima, a maior parte das ocorrências envolve falha ou mau funcionamento.

Recomendações de segurança

In [18]:

```
rec = pd.read_csv('../data/rec.csv', sep='~')
rec
```

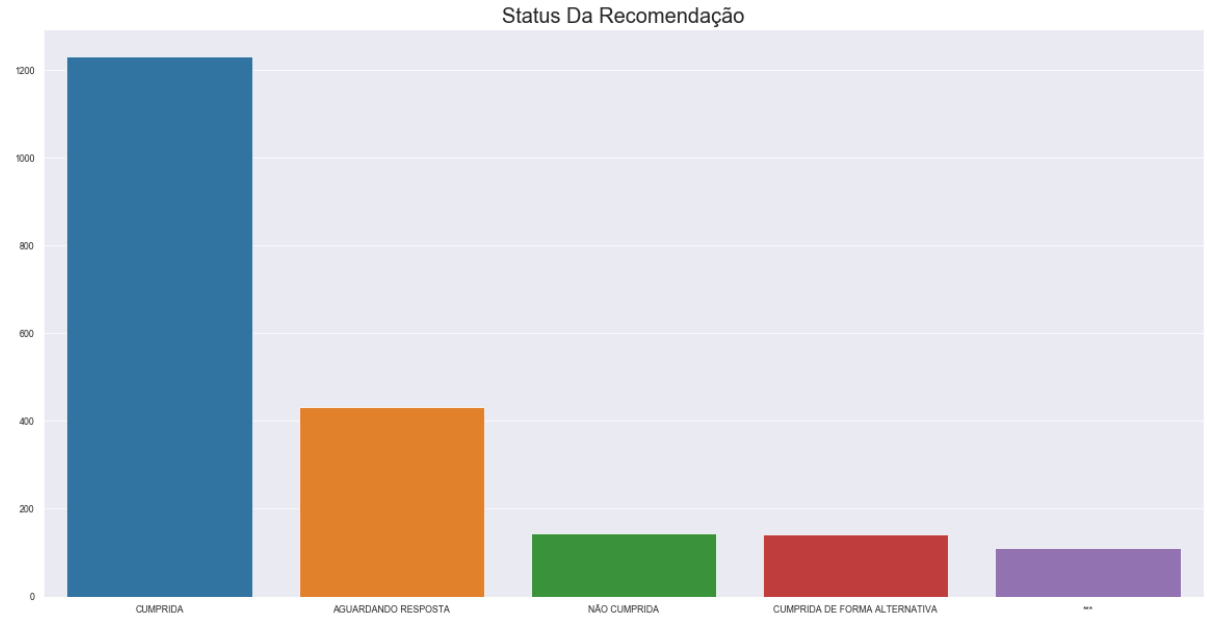
Out[18]:

	codigo_ocorrencia	recomendacao_numero	recomendacao_dia_assinatura	recomendacao_dia
0	200901015424167	123/2012	2012-03-22	
1	200901015424167	122/2012	2012-03-22	
2	200901015424167	121/2012	2012-03-22	
3	200901015424167	120/2012	2012-03-22	
4	200901015424167	119/2012	2012-03-22	
...	...	...	...	
2054	201812301440298	A-192/CENIPA/2018-01	2019-05-16	
2055	201901141204402	A-009/CENIPA/2019 - 01	2019-05-16	
2056	201902020208112	IG-025/CENIPA/2019 - 01	2019-05-16	
2057	201903061708253	IG-036/CENIPA/2019 - 01	2019-05-16	
2058	201903061708253	IG-036/CENIPA/2019 - 02	2019-05-16	

2059 rows × 10 columns

In [19]:

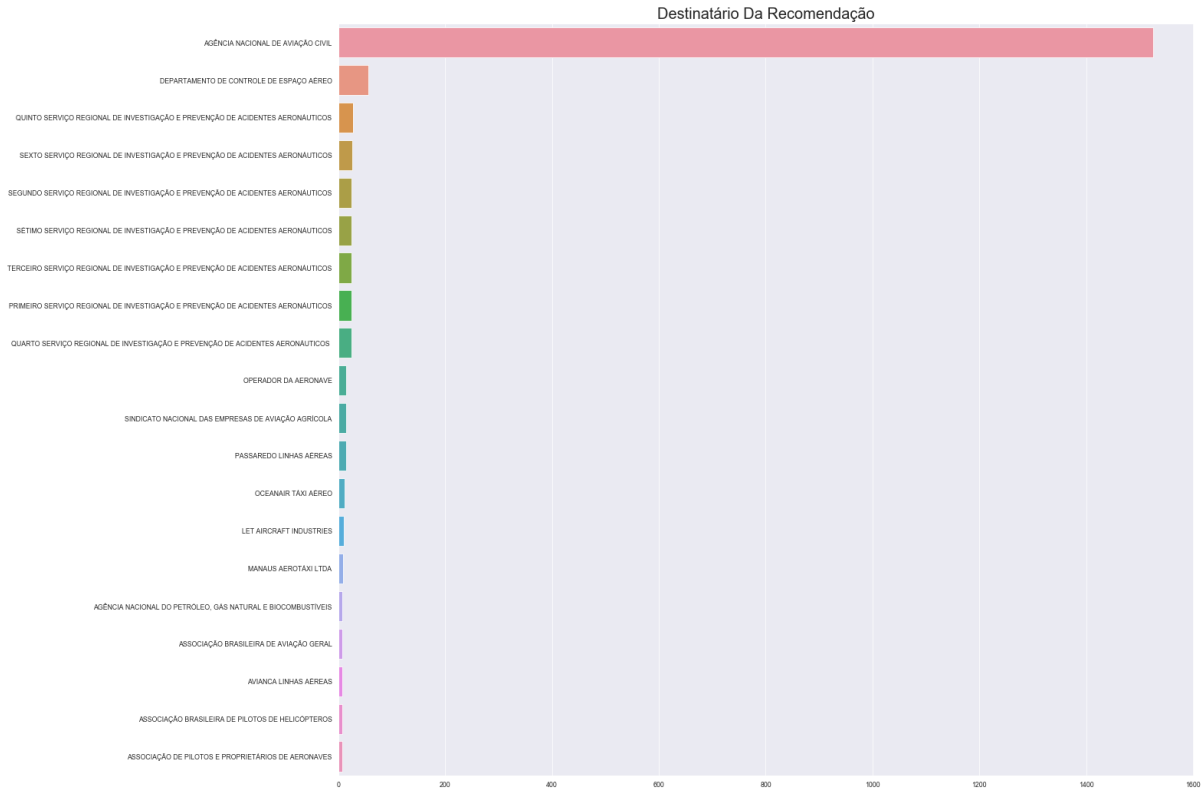
```
s, n = np.unique(rec["recomendacao_status"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Status Da Recomendação", fontsize=20)
pyplot.show()
```



Acima se pode observar que a grande maioria das ocorrências recebeu recomendação cumprida.

In [20]:

```
s, n = np.unique(rec["recomendacao_destinatario_nome"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 18))
seaborn.barplot(x=n[indices][:20], y=s[indices][:20], orient='h')
axis.set_title("Destinatário Da Recomendação", fontsize=20)
pyplot.show()
```



Como é possível observar acima, o maior destinatário das recomendações é a Agencia Nacional De Aviação Civil.

Risco

In [21]:

```
risco_col = ['tipo_ocorrencia', 'matricula', 'aerodromo', 'data', 'hora', 'fase_vo
o', 'efeito_voo']
risco = pd.read_csv('../data/risco.csv', sep=';', names=risco_col)
risco = risco.drop(risco.index[0]).reset_index().drop(['index'], axis=1)
risco
```

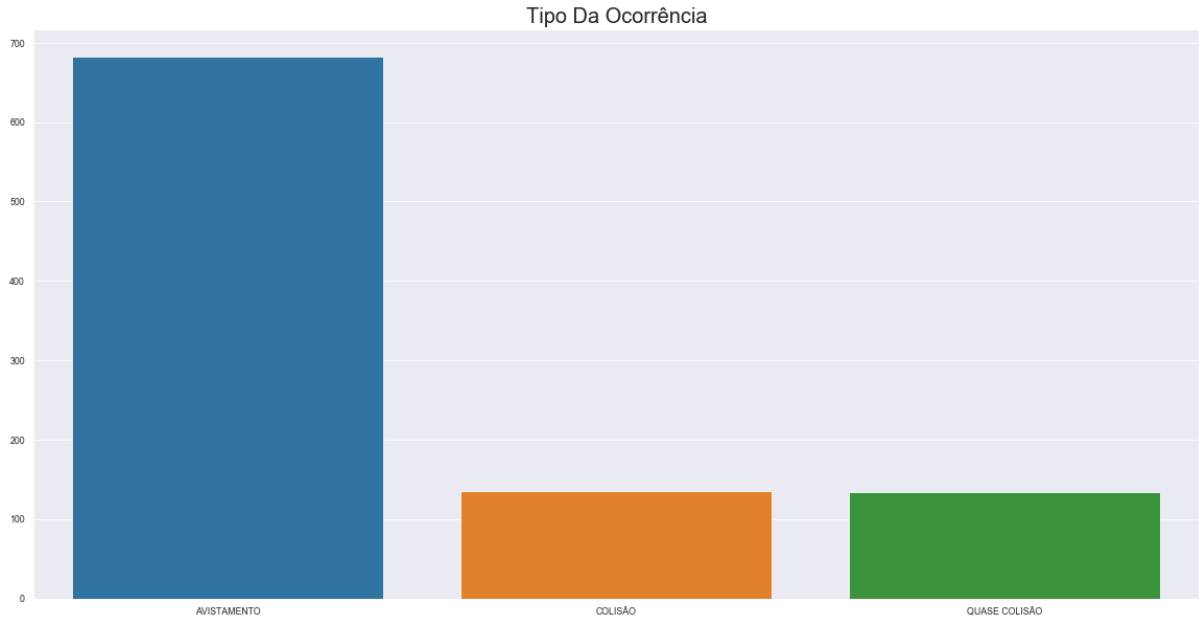
Out[21]:

	tipo_ocorrencia	matricula	aerodromo	data	hora	fase_voo	efeito_voo
0	AVISTAMENTO	FAB1925	SBYS	03/10/2019	08:10	Aproximação	Nenhum
1	AVISTAMENTO	PRUDS	SBUR	03/10/2019	17:00	Aproximação	Nenhum
2	AVISTAMENTO	NaN	SBSP	03/10/2019	09:35	Revisão de pista	Nenhum
3	AVISTAMENTO	NaN	SBUL	03/10/2019	15:30	Inspeção de trânsito/intervoo	Nenhum
4	AVISTAMENTO	NaN	SBBH	02/10/2019	13:33	Aproximação	Não reportada
...	...	...	...	...	...	...	...
947	QUASE COLISÃO	1944	SBYS	01/07/2019	16:00	Decolagem	Nenhum
948	AVISTAMENTO	FAB2318	SBNT	01/07/2019	12:45	Aproximação	Nenhum
949	COLISÃO	PRGEC	SWPI	01/07/2019	03:40	Pouso	Nenhum
950	QUASE COLISÃO	PRSED	9PCI	01/07/2019	16:20	Pouso	Nenhum
951	AVISTAMENTO	NaN	SBES	01/07/2019	08:00	Decolagem	Nenhum

952 rows × 7 columns

In [22]:

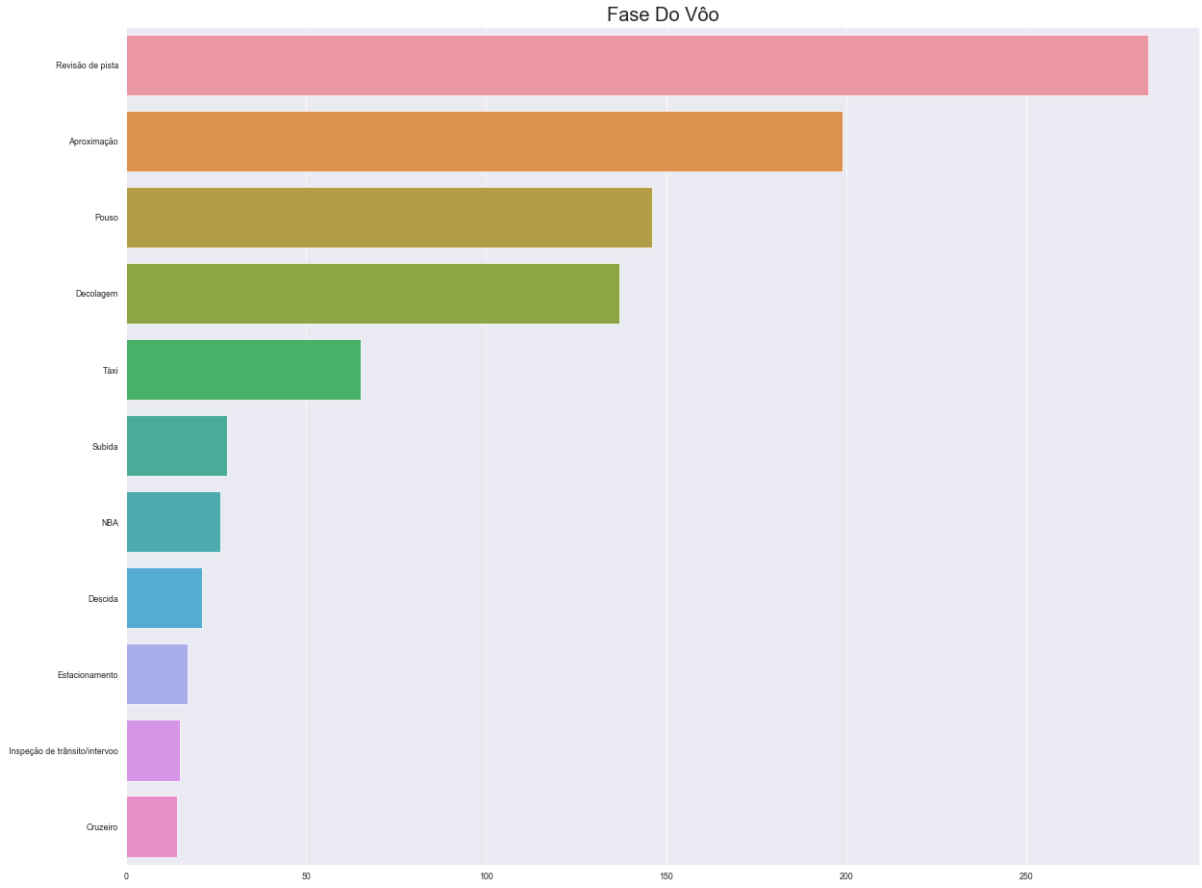
```
s, n = np.unique(risco["tipo_ocorrencia"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 10))
seaborn.barplot(y=n[indices], x=s[indices], orient='v')
axis.set_title("Tipo Da Ocorrência", fontsize=20)
pyplot.show()
```



Como visto acima, o tipo de ocorrência com maior probabilidade é no caso de avistamento.

In [23]:

```
s, n = np.unique(risco["fase_voo"].to_numpy(), return_counts=True)
indices = np.flip(np.argsort(n))
_, axis = pyplot.subplots(figsize=(20, 16))
seaborn.barplot(x=n[indices], y=s[indices], orient='h')
axis.set_title("Fase Do Vôo", fontsize=20)
pyplot.show()
```



Como visto acima, a fase de vôo com maior risco

Estabelecendo Conexão com o Banco De Dados

Nesta parte será estabelecida a conexão com o banco de dados pela interface do *SQLite*.

In [24]:

```
conn = sqlite3.connect('aeronautical_occurrences_database.db', cached_statements=0)
conn.execute("PRAGMA cache_size = 0")
```

Out[24]:

```
<sqlite3.Cursor at 0x1a1f70a650>
```

Checagem do DataBase criado

In [25]:

```
# Aeronaves envolvidas
pd.read_sql('select * from AeronavesEnvolvidas limit 5', conn)
```

Out[25]:

	codigo_ocorrencia	aeronave_matricula	aeronave_operador_categoria	aeronave_tipo_veiculo	ae
0	201106142171203	PPGXE	AEROCLUBE	AVIÃO	
1	201205209591320	PTRBN	OPERADOR DE AERONAVE	AVIÃO	
2	201012015549851	PTKUK	OPERADOR DE AERONAVE	AVIÃO	TV
3	201708190325167	PTKUK	OPERADOR PARTICULAR	AVIÃO	TV
4	201803182255192	PPGSZ	AEROCLUBE	AVIÃO	

In [26]:

```
# Fatores contribuintes
pd.read_sql('select * from FatoresContribuintes limit 5', conn)
```

Out[26]:

	codigo_ocorrencia	fator_nome	fator_aspecto	fator_condicionante	fator_area	fator_de
0	200901015424167	JULGAMENTO DE PILOTAGEM	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL	A INTI PRE AERO
1	200901015424167	MANUTENÇÃO DE AERONAVE	DESEMPENHO DO SER HUMANO	MANUTENÇÃO DA AERONAVE	FATOR OPERACIONAL	O F TÉCNI REVELO
2	200901015424167	SUPERVISÃO GERENCIAL	DESEMPENHO DO SER HUMANO	OPERAÇÃO DA AERONAVE	FATOR OPERACIONAL	O OPER/ PROC REALI
3	200901055963381	ATITUDE	ASPECTO PSICOLÓGICO	INDIVIDUAL	FATOR HUMANO	O P EXP RI
4	200901055963381	PROCESSO DECISÓRIO	ASPECTO PSICOLÓGICO	INDIVIDUAL	FATOR HUMANO	EXPER PILOT A

In [27]:

```
# Ocorrencias de acidentes
pd.read_sql('select * from Ocorrencias limit 5', conn)
```

Out[27]:

	codigo_ocorrencia	ocorrencia_classificacao	ocorrencia_tipo	ocorrencia_tipo_categoria	ocorren
0	201305055424986	ACIDENTE	FALHA DO MOTOR EM VOO	FALHA OU MAU FUNCIONAMENTO DO MOTOR	
1	201311259977425	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
2	201605160250139	INCIDENTE GRAVE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	
3	201805021421302	INCIDENTE	AERÓDROMO	AERÓDROMO	
4	201103187273112	INCIDENTE	OUTROS	OUTROS	

In [28]:

```
# Recomendacoes de seguranca
pd.read_sql('select * from RecomendacoesSeguranca limit 5', conn)
```

Out[28]:

	codigo_ocorrencia	recomendacao_numero	recomendacao_dia_assinatura	recomendacao_dia_en
0	200901015424167	123/2012	2012-03-22	
1	200901015424167	122/2012	2012-03-22	
2	200901015424167	121/2012	2012-03-22	
3	200901015424167	120/2012	2012-03-22	
4	200901015424167	119/2012	2012-03-22	

4. Diagrama ER


title

5. Diagrama relacional

6. Consultas

A seguir serão feitas várias consultas com diversos níveis de complexidade, a fim de possibilitat uma melhor visualização do comportamento dos dados e possíveis fatos curiosos e relevantes sobre a base de dados utilizada para o Trabalho Prático.

6.1 Duas consultas envolvendo seleção e projeção

6.1.1 Consulta 1 (duas versões)

Ocorrências classificadas como 'INCIDENTE GRAVE' que ocorreram em Rondônia.

Versão 1

In [29]:

```
query = """
        SELECT *
        FROM Ocorrencias
        WHERE ocorrencia_classificacao='INCIDENTE GRAVE' AND ocorrencia_uf='RO'
        """
pd.read_sql_query(query, conn)
```

Out[29]:

	codigo_ocorrencia	ocorrencia_classificacao	ocorrencia_tipo	ocorrencia_tipo_categoria	ocorren
0	201311259977425	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
1	201605160250139	INCIDENTE GRAVE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	
2	200901313794551	INCIDENTE GRAVE	PERDA DE CONTROLE NO SOLO	PERDA DE CONTROLE NO SOLO	
3	201303118764325	INCIDENTE GRAVE	FALHA DO MOTOR EM VOO	FALHA OU MAU FUNCIONAMENTO DO MOTOR	
4	201303057625078	INCIDENTE GRAVE	POUSO LONGO	EXCURSÃO DE PISTA	
5	201510231819564	INCIDENTE GRAVE	ESTOURO DE PNEU	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	
6	201705260552211	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
7	201808240124036	INCIDENTE GRAVE	COM TREM DE POUSO	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	

Tempo para a versão 1

In [30]:

```
timeit pd.read_sql_query(query, conn)
```

```
3.25 ms ± 257 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versão 2

In [31]:

```
query = """
        SELECT *
        FROM
        (SELECT *
            FROM Ocorrencias
            WHERE ocorrencia_classificacao="INCIDENTE GRAVE")
        WHERE ocorrencia_uf="RO"
        """
pd.read_sql_query(query, conn)
```

Out[31]:

	codigo_ocorrencia	ocorrencia_classificacao	ocorrencia_tipo	ocorrencia_tipo_categoria	ocorren
0	201311259977425	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
1	201605160250139	INCIDENTE GRAVE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	
2	200901313794551	INCIDENTE GRAVE	PERDA DE CONTROLE NO SOLO	PERDA DE CONTROLE NO SOLO	
3	201303118764325	INCIDENTE GRAVE	FALHA DO MOTOR EM VOO	FALHA OU MAU FUNCIONAMENTO DO MOTOR	
4	201303057625078	INCIDENTE GRAVE	POUSO LONGO	EXCURSÃO DE PISTA	
5	201510231819564	INCIDENTE GRAVE	ESTOURO DE PNEU	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	
6	201705260552211	INCIDENTE GRAVE	POUSO SEM TREM	CONTATO ANORMAL COM A PISTA	
7	201808240124036	INCIDENTE GRAVE	COM TREM DE POUSO	FALHA OU MAU FUNCIONAMENTO DE SISTEMA / COMPON...	

Tempo para a versão 2

In [32]:

```
timeit pd.read_sql_query(query, conn)
```

```
3.28 ms ± 457 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.1.2 Consulta 2 (duas versões)

Data de acidente, modelo de aeronave e ano de fabricação de ocorrências com vítimas fatais em Belo Horizonte.

Versão 1

In [33]:

```
query = """
        SELECT total_fatalidades,
               ocorrencia_dia,
               ocorrencia_horario,
               aeronave_fabricante,
               aeronave_modelo,
               aeronave_ano_fabricacao
        FROM Ocorrencias NATURAL JOIN AeronavesEnvolvidas
        WHERE ocorrencia_cidade = "BELO HORIZONTE" AND total_fatalidades > 0
        ORDER BY ocorrencia_dia desc
        """

pd.read_sql(query, conn)
```

Out[33]:

	total_fatalidades	ocorrencia_dia	ocorrencia_horario	aeronave_fabricante	aeronave_modelo	ae
0	1	2019-04-13	18:00:00	SOCATA	ST-10	
1	3	2015-06-07	18:25:00	HAWKER BEECHCRAFT	C90GTI	
2	2	2010-02-26	11:19:00	CESSNA AIRCRAFT	310R	

Tempo para a versão 1

In [34]:

```
timeit pd.read_sql_query(query, conn)
```

```
2.86 ms ± 242 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versão 2

In [35]:

```
query = """SELECT total_fatalidades,
                  ocorrencia_dia,
                  ocorrencia_horario,
                  aeronave_fabricante,
                  aeronave_modelo,
                  aeronave_ano_fabricacao
        FROM Ocorrencias as oco,
        (SELECT codigo_ocorrencia, aeronave_fabricante, aeronave_modelo, aeronave_a
no_fabricacao, total_fatalidades
            FROM AeronavesEnvolvidas
            WHERE total_fatalidades > 0) AS aeros
        WHERE oco.codigo_ocorrencia = aeros.codigo_ocorrencia AND oco.ocorrencia_ci
dade = "BELO HORIZONTE"
        ORDER BY ocorrencia_dia desc
        """

pd.read_sql(query, conn)
```

Out[35]:

	total_fatalidades	ocorrencia_dia	ocorrencia_horario	aeronave_fabricante	aeronave_modelo	ae
0	1	2019-04-13	18:00:00	SOCATA	ST-10	
1	3	2015-06-07	18:25:00	HAWKER BEECHCRAFT	C90GTI	
2	2	2010-02-26	11:19:00	CESSNA AIRCRAFT	310R	

Tempo para a versão 2

In [36]:

```
timeit pd.read_sql_query(query, conn)
```

```
2.96 ms ± 162 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.2 Três consultas envolvendo junção de duas relações

6.2.1 Consulta 3 (duas versões)

Nome do fator contribuinte que mais esteve presente nas ocorrências 'FALHA DO MOTOR EM VOO'.

Versão 1

In [37]:

```
query = """
        SELECT fator_nome, count(*) as num_ocorrencias
        FROM Ocorrencias NATURAL JOIN FatoresContribuintes
        WHERE ocorrencia_tipo = "FALHA DO MOTOR EM VOO"
        GROUP BY fator_nome
        ORDER BY count(*) DESC
        LIMIT 1
        """

pd.read_sql(query, conn)
```

Out[37]:

	fator_nome	num_ocorrencias
0	MANUTENÇÃO DE AERONAVE	123

Tempo para a versão 1

In [38]:

```
timeit pd.read_sql_query(query, conn)
```

```
8.21 ms ± 98.7 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versao 2

In [39]:

```
query = """
        SELECT fator_nome, count(*) AS num_ocorrencias
        FROM FatoresContribuintes
        WHERE codigo_ocorrencia IN
        (SELECT codigo_ocorrencia
            FROM Ocorrencias
            WHERE ocorrencia_tipo = "FALHA DO MOTOR EM VOO")
        GROUP BY fator_nome
        ORDER BY count(*) desc
        LIMIT 1
        """


pd.read_sql(query, conn)
```

Out[39]:

	fator_nome	num_ocorrencias
0	MANUTENÇÃO DE AERONAVE	123

Tempo para a versão 2

In [40]:

```
timeit pd.read_sql_query(query, conn)
```

```
2.55 ms ± 57.3 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.2.2 Consulta 4 (duas versões)

Matrícula, local, data, modelo, data de fabricação e número de fatalidades para aeronaves que se acidentaram mais de uma vez.

Versão 1

In [41]:

```
query = """
        SELECT num_ocorrencias,
               total_fatalidades,
               aeronave_matricula,
               aeronave_fabricante,
               aeronave_modelo,
               aeronave_ano_fabricacao,
               ocorrencia_cidade,
               ocorrencia_uf,
               ocorrencia_dia
        FROM (AeronavesEnvolvidas NATURAL JOIN
        (SELECT COUNT(*) AS num_ocorrencias, aeronave_matricula
            FROM AeronavesEnvolvidas
            GROUP BY aeronave_matricula
            HAVING COUNT(*) > 1)
        NATURAL JOIN Ocorrencias)
        ORDER BY num_ocorrencias desc
        """

pd.read_sql(query, conn)
```

Out[41]:

	num_ocorrencias	total_fatalidades	aeronave_matricula	aeronave_fabricante	aeronave_mode
0	10	0	PPGMA	AERO BOERO	AB-1
1	10	0	PPGMA	AERO BOERO	AB-1
2	10	0	PPGMA	AERO BOERO	AB-1
3	10	0	PPGMA	AERO BOERO	AB-1
4	10	0	PPGMA	AERO BOERO	AB-1
...	...	...	...	...	
2445	2	0	PUXXX11	***	
2446	2	0	PRZHW	DENNIS E. MOELLMAN	FOXTROT
2447	2	0	PRZHW	DENNIS E. MOELLMAN	FOXTROT
2448	2	0	PPNCG	PIPER AIRCRAFT	PA-46-35(
2449	2	0	PPNCG	PIPER AIRCRAFT	PA-46-35(

2450 rows × 9 columns

Tempo para a versão 1

In [42]:

```
timeit pd.read_sql_query(query, conn)
```

```
23.8 ms ± 363 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Versão 2

In [43]:

```
query = """
        SELECT num_ocorrencias,
               total_fatalidades,
               A.aeronave_matricula,
               aeronave_fabricante,
               aeronave_modelo,
               aeronave_ano_fabricacao,
               ocorrencia_cidade,
               ocorrencia_uf,
               ocorrencia_dia
        FROM AeronavesEnvolvidas AS A,
        (SELECT COUNT(*) AS num_ocorrencias, aeronave_matricula
            FROM AeronavesEnvolvidas
            GROUP BY aeronave_matricula)
        AS B, Ocorrencias AS oco
        WHERE A.aeronave_matricula = B.aeronave_matricula
              AND A.codigo_ocorrencia = oco.codigo_ocorrencia
              AND num_ocorrencias > 1
        ORDER BY num_ocorrencias DESC
        """

pd.read_sql(query, conn)
```

Out[43]:

	num_ocorrencias	total_fatalidades	aeronave_matricula	aeronave_fabricante	aeronave_mode
0	10	0	PPGMA	AERO BOERO	AB-1
1	10	0	PPGMA	AERO BOERO	AB-1
2	10	0	PPGMA	AERO BOERO	AB-1
3	10	0	PPGMA	AERO BOERO	AB-1
4	10	0	PPGMA	AERO BOERO	AB-1
...	...	...	...	...	
2445	2	0	PUXXX11	***	
2446	2	0	PRZHW	DENNIS E. MOELLMAN	FOXTROT
2447	2	0	PRZHW	DENNIS E. MOELLMAN	FOXTROT
2448	2	0	PPNCG	PIPER AIRCRAFT	PA-46-35(
2449	2	0	PPNCG	PIPER AIRCRAFT	PA-46-35(

2450 rows × 9 columns

Tempo para a versão 2

In [44]:

```
timeit pd.read_sql_query(query, conn)
```

```
24 ms ± 518 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

6.2.3 Consulta 5 (duas versões)

Código de ocorrências, local e data para aquelas categoria 'INCIDENTE GRAVE' que ocorreram no estado de São Paulo que tinham como status da recomendação de segurança 'AGUARDANDO RESPOSTA'.

Versão 1

In [45]:

```
query = """
        SELECT distinct codigo_ocorrencia,
               ocorrencia_cidade,
               ocorrencia_dia,
               ocorrencia_horario
        FROM Ocorrencias natural join RecomendacoesSeguranca
        WHERE ocorrencia_classificacao = "INCIDENTE GRAVE" AND ocorrencia_uf = "SP"
AND recomendacao_status = "AGUARDANDO RESPOSTA"
        ORDER BY ocorrencia_dia desc
        """
pd.read_sql(query, conn)
```

Out[45]:

	codigo_ocorrencia	ocorrencia_cidade	ocorrencia_dia	ocorrencia_horario
0	201712311507034	UBATUBA	2017-12-29	17:30:00
1	201703221347148	GUARULHOS	2017-03-22	01:30:00
2	201606301659061	PIRACICABA	2016-06-25	13:59:00
3	201410231951830	BRAGANÇA PAULISTA	2014-10-23	16:00:00
4	201006081517488	SÃO PAULO	2010-06-08	19:30:00

Tempo para a versão 1

In [46]:

```
timeit pd.read_sql_query(query, conn)
```

```
1.89 ms ± 134 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Versão 2

In [47]:

```
query = """SELECT codigo_ocorrencia,
                  ocorrencia_cidade,
                  ocorrencia_dia,
                  ocorrencia_horario
        FROM Ocorrencias
        WHERE ocorrencia_classificacao = "INCIDENTE GRAVE"
              AND ocorrencia_uf = "SP"
              AND codigo_ocorrencia IN (
                   SELECT codigo_ocorrencia
                   FROM RecomendacoesSeguranca
                   WHERE recomendacao_status = "AGUARDANDO RESPOSTA"
                 )
        ORDER BY ocorrencia_dia desc
        """
pd.read_sql(query, conn)
```

Out[47]:

	codigo_ocorrencia	ocorrencia_cidade	ocorrencia_dia	ocorrencia_horario
0	201712311507034	UBATUBA	2017-12-29	17:30:00
1	201703221347148	GUARULHOS	2017-03-22	01:30:00
2	201606301659061	PIRACICABA	2016-06-25	13:59:00
3	201410231951830	BRAGANÇA PAULISTA	2014-10-23	16:00:00
4	201006081517488	SÃO PAULO	2010-06-08	19:30:00

Tempo para a versão 2

In [48]:

```
timeit pd.read_sql_query(query, conn)
```

```
1.93 ms ± 284 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.3 Três consultas envolvendo junção de três ou mais relações

6.3.1 Consulta 6 (duas versões)

Recomendações não cumpridas, seu destinatário e data de recomendação para incidentes graves sem vítimas fatais.

Versão 1

In [49]:

```
query = """
        SELECT recomendacao_destinatario_sigla, recomendacao_conteudo, recomendacao
_dia_assinatura
        FROM RecomendacoesSeguranca NATURAL JOIN Ocorrencias NATURAL JOIN Aeronaves
Envolvidas
        WHERE ocorrencia_classificacao = "INCIDENTE GRAVE"
              AND total_fatalidades = 0
              AND recomendacao_status = "NÃO CUMPRIDA"
        ORDER BY recomendacao_dia_assinatura
        """

pd.read_sql(query, conn)
```

Out[49]:

	recomendacao_destinatario_sigla	recomendacao_conteudo	recomendacao_dia_assinatura
0	ANAC	Rever a letra *e* do item G.5, no corpo do apê...	2011-02-25
1	ANAC	Realizar Vistoria de Segurança de Voo na ofici...	2011-09-23
2	DECEA	Visto que ainda NULL um método de avalição def...	2011-12-12
3	ANAC	Definir, em caráter de regulação, requisitos q...	2011-12-12
4	DECEA	Visto que ainda NULL um método de avalição def...	2013-07-16
5	DECEA	Realizar gestões junto à ANAC no intuito de es...	2013-07-16
6	ANAC	Realizar gestões junto aos aeródromos públicos...	2013-07-16
7	ANAC	Verificar as condições de aeronavegabilidade d...	2013-09-16
8	ANAC	Atuar junto aos órgãos competentes, para ações...	2013-09-19
9	ANAC	Implantar medidas que assegurem que o operador...	2013-10-21
10	ANAC	Realizar vistoria na pista do Aeródromo Aéreo ...	2013-10-21
11	ANAC	Realizar vistoria na pista do aeródromo Aéreo ...	2013-10-21
12	ANAC	Determinar que os Operadores Regulados pelo RB...	2013-12-16
13	ANAC	Determinar que os Operadores Regulados pelo RB...	2013-12-16
14	ANAC	Atuar junto ao operador da aeronave de modo a ...	2014-02-02
15	ANAC	Assegurar-se de que os processos para execução...	2014-11-24
16	ANAC	Avaliar a mudança de classificação das aeronav...	2015-05-15
17	ANAC	Adequar os regulamentos que normatizam a ativi...	2016-06-03
18	ANAC	Certificar-se, por ocasião de vistoria técnica...	2016-06-23
19	ANAC	Adotar medidas para que o operador da aeronave...	2016-12-01
20	ANAC	Certificar-se de que a CFA – CURSOS ESCOLA DE ...	2016-12-01
21	ANAC	Atuar junto ao operador do Aeródromo Carlos Pr...	2018-03-09

	recomendacao_destinatario_sigla	recomendacao_conteudo	recomendacao_dia_assinatura
22	ANAC	Atuar junto ao operador do Aeródromo Carlos Pr...	2018-03-09
23	ANAC	Exigir dos Operadores de Aeródromo que, no cas...	2018-09-04
24	ANAC	Exigir dos Operadores de Aeródromos que, duran...	2018-09-04
25	ANAC	Exigir dos Operadores de Aeródromos (e monitor...	2018-09-04

Tempo para a versão 1

In [50]:

```
timeit pd.read_sql_query(query, conn)
```

```
5.33 ms ± 507 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versao 2

In [51]:

```
query = """
        SELECT recomendacao_destinatario_sigla, recomendacao_conteudo, recomendacao
_dia_assinatura
        FROM RecomendacoesSeguranca
        WHERE codigo_ocorrencia IN (
                SELECT codigo_ocorrencia
                FROM Ocorrencias
                WHERE ocorrencia_classificacao = "INCIDENTE GRAVE"
              )
              AND
              codigo_ocorrencia IN (
                  SELECT codigo_ocorrencia
                  FROM AeronavesEnvolvidas
                  WHERE total_fatalidades = 0
              )
              AND
              recomendacao_status = "NÃO CUMPRIDA"
              ORDER BY recomendacao_dia_assinatura
        """

pd.read_sql(query, conn)
```

Out[51]:

	recomendacao_destinatario_sigla	recomendacao_conteudo	recomendacao_dia_assinatura
0	ANAC	Rever a letra *e* do item G.5, no corpo do apê...	2011-02-25
1	ANAC	Realizar Vistoria de Segurança de Voo na ofici...	2011-09-23
2	DECEA	Visto que ainda NULL um método de avalição def...	2011-12-12
3	ANAC	Definir, em caráter de regulação, requisitos q...	2011-12-12
4	DECEA	Visto que ainda NULL um método de avalição def...	2013-07-16
5	DECEA	Realizar gestões junto à ANAC no intuito de es...	2013-07-16
6	ANAC	Realizar gestões junto aos aeródromos públicos...	2013-07-16
7	ANAC	Verificar as condições de aeronavegabilidade d...	2013-09-16
8	ANAC	Atuar junto aos órgãos competentes, para ações...	2013-09-19
9	ANAC	Implantar medidas que assegurem que o operador...	2013-10-21
10	ANAC	Realizar vistoria na pista do Aeródromo Aéreo ...	2013-10-21
11	ANAC	Realizar vistoria na pista do aeródromo Aéreo ...	2013-10-21
12	ANAC	Determinar que os Operadores Regulados pelo RB...	2013-12-16
13	ANAC	Determinar que os Operadores Regulados pelo RB...	2013-12-16
14	ANAC	Atuar junto ao operador da aeronave de modo a ...	2014-02-02
15	ANAC	Assegurar-se de que os processos para execução...	2014-11-24
16	ANAC	Avaliar a mudança de classificação das aeronav...	2015-05-15
17	ANAC	Adequar os regulamentos que normatizam a ativi...	2016-06-03
18	ANAC	Certificar-se, por ocasião de vistoria técnica...	2016-06-23
19	ANAC	Adotar medidas para que o operador da aeronave...	2016-12-01
20	ANAC	Certificar-se de que a CFA – CURSOS ESCOLA DE ...	2016-12-01
21	ANAC	Atuar junto ao operador do Aeródromo Carlos Pr...	2018-03-09

	recomendacao_destinatario_sigla	recomendacao_conteudo	recomendacao_dia_assinatura
22	ANAC	Atuar junto ao operador do Aeródromo Carlos Pr...	2018-03-09
23	ANAC	Exigir dos Operadores de Aeródromo que, no cas...	2018-09-04
24	ANAC	Exigir dos Operadores de Aeródromos que, duran...	2018-09-04
25	ANAC	Exigir dos Operadores de Aeródromos (e monitor...	2018-09-04

Tempo para a versão 2

In [52]:

```
timeit pd.read_sql_query(query, conn)
```

```
5.29 ms ± 418 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.3.2 Consulta 7 (duas versões)

Tipo, classificação, data, número de fatalidades e local de acidentes provocados por pilotos de pouca experiência com vítimas fatais.

Versão 1

In [53]:

```
query = """
        SELECT ocorrencia_classificacao,
               ocorrencia_tipo,
               total_fatalidades,
               ocorrencia_dia,
               ocorrencia_cidade,
               ocorrencia_uf
        FROM FatoresContribuintes NATURAL JOIN Ocorrencias NATURAL JOIN AeronavesEn
volvidas
        WHERE fator_nome = "POUCA EXPERIÊNCIA DO PILOTO"
              AND total_fatalidades > 0
        ORDER BY total_fatalidades DESC
        """

pd.read_sql(query, conn)
```

Out[53]:

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
0	ACIDENTE	PERDA DE CONTROLE EM VOO	10	2013-03-12	ALMEIRI
1	ACIDENTE	FALHA DO MOTOR EM VOO	8	2016-07-31	LONDRIN
2	ACIDENTE	FALHA DO MOTOR EM VOO	6	2016-01-20	LONDRIN
3	ACIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	5	2013-02-03	CÂNDIDO MOT
4	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	5	2013-10-19	CORUMB
5	ACIDENTE	OUTROS	5	2013-12-04	NOV(PROGRESS(
6	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	4	2010-09-15	SÃO FÉLIX D(XING(
7	ACIDENTE	FALHA DO MOTOR EM VOO	4	2010-12-09	BOM JESUS D(GALH(
8	ACIDENTE	FALHA DO MOTOR EM VOO	4	2013-12-16	TERESIN
9	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2015-02-19	BUENO BRANDÃ(
10	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2015-09-23	MACEI(
11	ACIDENTE	FALHA ESTRUTURAL	4	2016-07-02	BELÉ
12	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2016-09-22	CAMPINÁPOLI
13	ACIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	3	2011-05-14	BREU BRANC(
14	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2011-09-13	ÂNGUL(
15	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2012-03-06	ELISEU MARTIN
16	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2014-11-10	MACARA

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
17	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	3	2015-12-06	TRINDAD
18	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2009-05-22	VITÓRIA D/ CONQUIST/
19	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2009-09-24	LEOPOLDO D BULHÕE
20	ACIDENTE	INDETERMINADO	2	2009-12-23	CAROLIN/
21	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	2	2010-07-18	SANTO ANTÔNI DO LEVERGEI
22	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2010-11-15	BRAGANÇ/ PAULIST/
23	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	2	2011-01-30	PACAJ/
24	ACIDENTE	INDETERMINADO	2	2012-04-20	MACAP/
25	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2012-04-26	GUARUJ/
26	ACIDENTE	OUTROS	2	2012-08-21	RIO DE JANEIR
27	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2014-05-10	CACHOEIRA D SU
28	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2014-09-07	TANGARÁ D/ SERR/
29	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2015-10-22	RONDONÓPOLI
30	ACIDENTE	IMC NÃO INTENCIONAL	2	2019-01-12	CASCAVE
31	ACIDENTE	DESORIENTAÇÃO ESPACIAL	1	2009-03-08	QUERÊNCI/
32	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	1	2009-11-19	URUGUAIAN/

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
33	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2010-02-06	ITUMBIAR/
34	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2010-12-21	ARACAJ(
35	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	1	2011-12-09	SÃO BENTO D(SU
36	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2012-04-20	JUNDIA
37	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2012-08-21	PALMEIR/
38	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2014-11-17	CIDADE GAÚCH/
39	ACIDENTE	FALHA DO MOTOR EM VOO	1	2015-01-04	TOLED(
40	ACIDENTE	OPERAÇÃO A BAIXA ALTITUDE	1	2015-02-07	SENTINELA D(SU
41	ACIDENTE	OPERAÇÃO A BAIXA ALTITUDE	1	2015-03-23	SANTA LUZI/
42	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2015-11-09	BALSA
43	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2017-08-20	PALMA
44	ACIDENTE	INDETERMINADO	1	2018-07-27	ITAPEM/
45	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2018-12-29	ARACAJ(

Tempo para a versão 1

In [54]:

```
timeit pd.read_sql_query(query, conn)
```

```
5.42 ms ± 354 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versão 2

In [55]:

```
query = """
        SELECT ocorrencia_classificacao,
               ocorrencia_tipo,
               total_fatalidades,
               ocorrencia_dia,
               ocorrencia_cidade,
               ocorrencia_uf
        FROM Ocorrencias AS oco, AeronavesEnvolvidas AS aev
        WHERE  oco.codigo_ocorrencia = aev.codigo_ocorrencia
               AND oco.codigo_ocorrencia IN (
                   SELECT codigo_ocorrencia
                   FROM FatoresContribuintes
                   WHERE fator_nome = "POUCA EXPERIÊNCIA DO PILOTO"
               )
               AND
               total_fatalidades > 0
        ORDER BY total_fatalidades DESC
        """

pd.read_sql(query, conn)
```

Out[55]:

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
0	ACIDENTE	PERDA DE CONTROLE EM VOO	10	2013-03-12	ALMEIRI
1	ACIDENTE	FALHA DO MOTOR EM VOO	8	2016-07-31	LONDRIN
2	ACIDENTE	FALHA DO MOTOR EM VOO	6	2016-01-20	LONDRIN
3	ACIDENTE	OUTROS	5	2013-12-04	NOV(PROGRESS(
4	ACIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	5	2013-02-03	CÂNDIDO MOT
5	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	5	2013-10-19	CORUMB
6	ACIDENTE	FALHA DO MOTOR EM VOO	4	2013-12-16	TERESIN
7	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2015-02-19	BUENO BRANDÃ(
8	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	4	2010-09-15	SÃO FÉLIX D(XINGU
9	ACIDENTE	FALHA DO MOTOR EM VOO	4	2010-12-09	BOM JESUS D(GALH(
10	ACIDENTE	FALHA ESTRUTURAL	4	2016-07-02	BELÉM
11	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2015-09-23	MACEI(
12	ACIDENTE	PERDA DE CONTROLE EM VOO	4	2016-09-22	CAMPINÁPOLI
13	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2012-03-06	ELISEU MARTIN
14	ACIDENTE	CAUSADO POR FENÔMENO METEOROLÓGICO EM VOO	3	2011-05-14	BREU BRANC(
15	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2011-09-13	ÂNGUL(

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
16	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	3	2015-12-06	TRINDAD
17	ACIDENTE	PERDA DE CONTROLE EM VOO	3	2014-11-10	MACARAN
18	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2010-11-15	BRAGANÇ PAULIST
19	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2009-05-22	VITÓRIA D CONQUIST
20	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2009-09-24	LEOPOLDO D BULHÕE
21	ACIDENTE	INDETERMINADO	2	2009-12-23	CAROLIN
22	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	2	2010-07-18	SANTO ANTÔNI DO LEVERGE
23	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	2	2011-01-30	PACAJ
24	ACIDENTE	INDETERMINADO	2	2012-04-20	MACAP
25	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2012-04-26	GUARUJ
26	ACIDENTE	OUTROS	2	2012-08-21	RIO DE JANEIR
27	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2015-10-22	RONDONÓPOLI
28	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	2	2014-09-07	TANGARÁ D SERR
29	ACIDENTE	PERDA DE CONTROLE EM VOO	2	2014-05-10	CACHOEIRA D SU
30	ACIDENTE	IMC NÃO INTENCIONAL	2	2019-01-12	CASCAVE
31	ACIDENTE	COLISÃO COM OBSTÁCULO DURANTE A DECOLAGEM E POUSO	1	2009-11-19	URUGUAIAN

	ocorrencia_classificacao	ocorrencia_tipo	total_fatalidades	ocorrencia_dia	ocorrencia_cidad
32	ACIDENTE	OPERAÇÃO A BAIXA ALTITUDE	1	2015-03-23	SANTA LUZI
33	ACIDENTE	DESORIENTAÇÃO ESPACIAL	1	2009-03-08	QUERÊNCI
34	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2010-02-06	ITUMBIAR
35	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2012-08-21	PALMEIR
36	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2010-12-21	ARACAJ
37	ACIDENTE	VOO CONTROLADO CONTRA O TERRENO	1	2011-12-09	SÃO BENTO D SU
38	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2012-04-20	JUNDI
39	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2015-11-09	BALSA
40	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2014-11-17	CIDADE GAÚCH
41	ACIDENTE	OPERAÇÃO A BAIXA ALTITUDE	1	2015-02-07	SENTINELA D SU
42	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2017-08-20	PALMA
43	ACIDENTE	INDETERMINADO	1	2018-07-27	ITAPEM
44	ACIDENTE	FALHA DO MOTOR EM VOO	1	2015-01-04	TOLED
45	ACIDENTE	PERDA DE CONTROLE EM VOO	1	2018-12-29	ARACAJ

Tempo para a versão 2

In [56]:

```
timeit pd.read_sql_query(query, conn)
```

```
3.5 ms ± 317 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.3.3 Consulta 8 (duas versões)

Local e data de acidentes que tiveram como fator contribuinte infraestrutura do aeroporto envolvendo mais de uma aeronave sem vítimas fatais.

Versão 1

In [57]:

```
query = """
        SELECT DISTINCT total_aeronaves_envolvidas, ocorrencia_cidade, ocorrencia_u
f, ocorrencia_dia
        FROM Ocorrencias NATURAL JOIN FatoresContribuintes NATURAL JOIN AeronavesEn
volvidas
        WHERE fator_nome = "INFRAESTRUTURA AEROPORTUÁRIA"
            AND total_aeronaves_envolvidas > 1
            AND total_fatalidades = 0
        """

pd.read_sql(query, conn)
```

Out[57]:

	total_aeronaves_envolvidas	ocorrencia_cidade	ocorrencia_uf	ocorrencia_dia
0	2	CARUARU	PE	2011-08-21
1	2	RIO DE JANEIRO	RJ	2013-09-04

Tempo para a versão 1

In [58]:

```
timeit pd.read_sql_query(query, conn)
```

```
5.34 ms ± 453 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versão 2

In [59]:

```
query = """
        SELECT DISTINCT total_aeronaves_envolvidas, ocorrencia_cidade, ocorrencia_u
f, ocorrencia_dia
        FROM Ocorrencias
        WHERE codigo_ocorrencia IN (
                SELECT codigo_ocorrencia
                FROM FatoresContribuintes
                WHERE fator_nome = "INFRAESTRUTURA AEROPORTUÁRIA"
            )
            AND
            codigo_ocorrencia IN (
                SELECT codigo_ocorrencia
                FROM AeronavesEnvolvidas
                WHERE total_fatalidades = 0
            )
            AND
            total_aeronaves_envolvidas > 1
        """

pd.read_sql(query, conn)
```

Out[59]:

	total_aeronaves_envolvidas	ocorrencia_cidade	ocorrencia_uf	ocorrencia_dia
0	2	CARUARU	PE	2011-08-21
1	2	RIO DE JANEIRO	RJ	2013-09-04

Tempo para a versão 2

In [60]:

```
timeit pd.read_sql_query(query, conn)
```

```
4.41 ms ± 108 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.4 Duas consultas envolvendo agregação sobre junção de duas ou mais relações

6.4.1 Consulta 9 (duas versões)

Número de acidentes, total e média de fatalidades por acidentes aeronáuticas por estado.

Versão 1

In [61]:

```
query = """
        SELECT ocorrencia_uf,
               COUNT(*) AS num_ocorrencias,
               SUM(total_fatalidades) AS fatalidades,
               AVG(total_fatalidades) AS media_fatalidades
        FROM Ocorrencias
             NATURAL JOIN (SELECT * FROM AeronavesEnvolvidas GROUP BY codigo_ocorre
ncia)
        GROUP BY ocorrencia_uf
        ORDER BY num_ocorrencias DESC
        """

pd.read_sql(query, conn)
```

Out[61]:

	ocorrencia_uf	num_ocorrencias	fatalidades	media_fatalidades
0	SP	1310	152	0.116031
1	RJ	518	49	0.094595
2	MG	503	83	0.165010
3	PR	437	57	0.130435
4	RS	343	35	0.102041
5	GO	295	64	0.216949
6	PA	272	89	0.327206
7	MT	247	67	0.271255
8	AM	226	79	0.349558
9	BA	217	38	0.175115
10	SC	169	18	0.106509
11	MS	147	25	0.170068
12	DF	145	1	0.006897
13	PE	88	21	0.238636
14	CE	83	4	0.048193
15	MA	73	20	0.273973
16	ES	73	2	0.027397
17	AC	53	2	0.037736
18	TO	52	7	0.134615
19	RR	49	13	0.265306
20	PI	35	12	0.342857
21	RO	30	4	0.133333
22	PB	25	0	0.000000
23	SE	22	6	0.272727
24	AL	19	5	0.263158
25	RN	16	2	0.125000
26	AP	11	3	0.272727
27	***	2	0	0.000000

Tempo para a consulta 1

In [62]:

```
timeit pd.read_sql_query(query, conn)
```

```
35.4 ms ± 2.04 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Versão 2

In [63]:

```
query = """
        SELECT ocorrencia_uf,
               COUNT(*) AS num_ocorrencias,
               SUM(total_fatalidades) AS fatalidades,
               AVG(total_fatalidades) AS media_fatalidades
        FROM (SELECT codigo_ocorrencia, ocorrencia_uf FROM Ocorrencias) AS oco,
             (SELECT codigo_ocorrencia, total_fatalidades
              FROM AeronavesEnvolvidas
              GROUP BY codigo_ocorrencia
              ) AS aev
        WHERE oco.codigo_ocorrencia = aev.codigo_ocorrencia
        GROUP BY ocorrencia_uf
        ORDER BY num_ocorrencias DESC
        """
pd.read_sql(query, conn)
```

Out[63]:

	ocorrencia_uf	num_ocorrencias	fatalidades	media_fatalidades
0	SP	1310	152	0.116031
1	RJ	518	49	0.094595
2	MG	503	83	0.165010
3	PR	437	57	0.130435
4	RS	343	35	0.102041
5	GO	295	64	0.216949
6	PA	272	89	0.327206
7	MT	247	67	0.271255
8	AM	226	79	0.349558
9	BA	217	38	0.175115
10	SC	169	18	0.106509
11	MS	147	25	0.170068
12	DF	145	1	0.006897
13	PE	88	21	0.238636
14	CE	83	4	0.048193
15	MA	73	20	0.273973
16	ES	73	2	0.027397
17	AC	53	2	0.037736
18	TO	52	7	0.134615
19	RR	49	13	0.265306
20	PI	35	12	0.342857
21	RO	30	4	0.133333
22	PB	25	0	0.000000
23	SE	22	6	0.272727
24	AL	19	5	0.263158
25	RN	16	2	0.125000
26	AP	11	3	0.272727
27	***	2	0	0.000000

Tempo para a consulta 2

In [64]:

```
timeit pd.read_sql_query(query, conn)
```

```
19.4 ms ± 287 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

6.4.2 Consulta 10 (duas versões)

Estados ordenados de acordo com a razão ocorrências com fator metereológico pelo número total de ocorrências.

Versão 1

In [65]:

```
query = """
        SELECT COUNT(*) / CAST(total_ocor as float) AS meteor_by_total,
               ocorrencia_uf,
               total_ocor,
               COUNT(*) AS meteor
        FROM Ocorrencias NATURAL JOIN FatoresContribuintes NATURAL JOIN
        (
            SELECT COUNT(*) AS total_ocor, ocorrencia_uf
            FROM Ocorrencias
            GROUP BY ocorrencia_uf
        )
        WHERE fator_nome = 'CONDIÇÕES METEOROLÓGICAS ADVERSAS'
        GROUP BY ocorrencia_uf
        ORDER BY meteor_by_total DESC
        """

pd.read_sql(query, conn)
```

Out[65]:

	meteor_by_total	ocorrencia_uf	total_ocor	meteor
0	0.500000	***	2	1
1	0.090909	AP	11	1
2	0.068826	MT	247	17
3	0.045455	SE	22	1
4	0.040816	RR	49	2
5	0.040816	MS	147	6
6	0.038462	TO	52	2
7	0.035398	AM	226	8
8	0.033333	RO	30	1
9	0.033088	PA	272	9
10	0.032258	BA	217	7
11	0.030508	GO	295	9
12	0.028571	PI	35	1
13	0.027397	MA	73	2
14	0.023669	SC	169	4
15	0.023324	RS	343	8
16	0.022883	PR	437	10
17	0.022727	PE	88	2
18	0.018868	AC	53	1
19	0.017375	RJ	518	9
20	0.016794	SP	1310	22
21	0.013917	MG	503	7
22	0.012048	CE	83	1
23	0.006897	DF	145	1

Tempo para a versão 1

In [66]:

```
timeit pd.read_sql_query(query, conn)
```

```
3.32 ms ± 47.9 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Versao 2

In [67]:

```
query = """
        SELECT COUNT(CASE WHEN fator_nome = 'CONDIÇÕES METEOROLÓGICAS ADVERSAS' THE
N 1 END)/CAST(COUNT(DISTINCT(O.codigo_ocorrencia)) AS float) AS meteor_by_total,
               ocorrencia_uf,
               COUNT(DISTINCT(O.codigo_ocorrencia)) AS total_ocor,
               COUNT(CASE WHEN fator_nome = 'CONDIÇÕES METEOROLÓGICAS ADVERSAS' THE
N 1 END) as meteor
        FROM Ocorrencias as O LEFT JOIN FatoresContribuintes as F ON O.codigo_ocorr
encia = F.codigo_ocorrencia
        GROUP BY ocorrencia_uf
        HAVING meteor > 0
        ORDER BY 1 DESC
        """

pd.read_sql(query, conn)
```

Out[67]:

	meteor_by_total	ocorrencia_uf	total_ocor	meteor
0	0.500000	***	2	1
1	0.090909	AP	11	1
2	0.068826	MT	247	17
3	0.045455	SE	22	1
4	0.040816	RR	49	2
5	0.040816	MS	147	6
6	0.038462	TO	52	2
7	0.035398	AM	226	8
8	0.033333	RO	30	1
9	0.033088	PA	272	9
10	0.032258	BA	217	7
11	0.030508	GO	295	9
12	0.028571	PI	35	1
13	0.027397	MA	73	2
14	0.023669	SC	169	4
15	0.023324	RS	343	8
16	0.022883	PR	437	10
17	0.022727	PE	88	2
18	0.018868	AC	53	1
19	0.017375	RJ	518	9
20	0.016794	SP	1310	22
21	0.013917	MG	503	7
22	0.012048	CE	83	1
23	0.006897	DF	145	1

Tempo para a consulta 2

In [68]:

```
timeit pd.read_sql_query(query, conn)
```

```
9.95 ms ± 155 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

7. Autoavaliação dos membros

Breve descrição sobre cada uma das atividade desenvolvidas por cada um dos membros deste grupo.

7.1 Gabriel Luz

Fiz a consulta de número 10. Ajudei na escolha do banco de dados, na confecção da proposta, no modelo ER, no modelo relacional e em partes gerais do trabalho.

7.2 Elves M. Rodrigues

Definição e implementação de parte das consultas, além de outras contribuições gerais da estrutura do projeto.

7.3 Luiz Henrique Melo

Elaboração e início do arquivo .ipynb final do Projeto participação na modelagem do Diagrama ER e na elaboração de consultas. Ajuda na criação do DataBase utilizado.

7.4 Otavio Augusto Silva

Visualização e análise dos dados, construção do banco de dados, modelagem ER, reescrita e padronização de código.

In [ ]: