02_Data_Exploration

April 15, 2022

Predicting Airline Delays

Team: Jimmy Nguyen, Maha Jayapal, Roberto Cancel

[1]:
```
!pip install --upgrade numpy #ensure numpy and pandas are upgraded to same␣
↪versions for easier exploration (avoiding errors)
!pip install --upgrade pandas #ensure numpy and pandas are upgraded to same␣
↪versions for easier exploration (avoiding errors)

# IMPORT LIBRARIES REQUIRED THROUGHOUT THE NOTEBOOK
import boto3 # AWS SDK for Python
import pandas as pd # for importing and manipulating data
import numpy as np
import io # for encoding issues with raw data sets
```

```
/opt/conda/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16:
CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
instead
  from cryptography.utils import int_from_bytes
/opt/conda/lib/python3.7/site-packages/secretstorage/util.py:25:
CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages
(1.21.5)
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.3.1; however, version 22.0.4 is

available.

You should consider upgrading via the '/opt/conda/bin/python -m pip install

--upgrade pip' command.
/opt/conda/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16:
CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
instead
  from cryptography.utils import int_from_bytes
```

```
/opt/conda/lib/python3.7/site-packages/secretstorage/util.py:25:
CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes
instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages
(1.3.5)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.7/site-
packages (from pandas) (1.21.5)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-
packages (from pandas) (2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.7/site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-
packages (from python-dateutil>=2.7.3->pandas) (1.14.0)
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.3.1; however, version 22.0.4 is
available.
You should consider upgrading via the '/opt/conda/bin/python -m pip install
--upgrade pip' command.
```

[2]:
```python
# IDENTIFY FILES IN S3 BUCKET
session = boto3.Session()

#Then use the session to get the resource
s3 = session.resource('s3')

my_bucket = s3.Bucket('ads-508-airline')

for my_bucket_object in my_bucket.objects.all():
    print(my_bucket_object.key)
```

```
raw/
raw/B43_AIRCRAFT_INVENTORY.csv
raw/CARRIER_DECODE.csv
raw/ONTIME_REPORTING_12.csv
raw/P10_EMPLOYEES.csv
raw/airport_weather_dec_2019.csv
raw/airports_list.csv
transformed/
transformed/B43_AIRCRAFT_INVENTORY.csv
transformed/CARRIER_DECODE.csv
transformed/ON_TIME_REPORTING_12.csv
transformed/P10_EMPLOYEES.csv
```

```
transformed/airport_weather_dec_2019.csv
transformed/airports_list.csv
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
INGEST DATA SETS xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[3]:
```python
# INGEST FLIGHT DATA

s3_client = boto3.client("s3")

BUCKET='ads-508-airline'
KEY='transformed/ON_TIME_REPORTING_12.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
dec_flight = pd.read_csv(response.get("Body"))
dec_flight.head()
```

[3]:
```
   DAY_OF_MONTH  DAY_OF_WEEK OP_UNIQUE_CARRIER TAIL_NUM ORIGIN DEST  \
0             8            7                WN   N8651A    STL  SAN
1             8            7                WN   N939WN    STL  SAT
2             8            7                WN   N7741C    STL  SAT
3             8            7                WN   N550WN    STL  SEA
4             8            7                WN   N8319F    STL  SFO

   DEP_DEL15 DEP_TIME_BLK ARR_TIME_BLK  CANCELLED  CRS_ELAPSED_TIME  DISTANCE  \
0        0.0    1100-1159    1300-1359        0.0             245.0    1557.0
1        0.0    1200-1259    1400-1459        0.0             145.0     786.0
2        0.0    2100-2159    0001-0559        0.0             140.0     786.0
3        0.0    0900-0959    1200-1259        0.0             275.0    1709.0
4        1.0    1800-1859    2000-2059        0.0             270.0    1735.0

   DISTANCE_GROUP  CARRIER_DELAY  WEATHER_DELAY  NAS_DELAY  SECURITY_DELAY  \
0               7            0.0            0.0       18.0             0.0
1               4            NaN            NaN        NaN             NaN
2               4            NaN            NaN        NaN             NaN
3               7            NaN            NaN        NaN             NaN
4               7            NaN            NaN        NaN             NaN

   LATE_AIRCRAFT_DELAY
0                  0.0
1                  NaN
2                  NaN
3                  NaN
4                  NaN
```

[4]:
```python
# INGEST AIRCRAFT DATA - raw data that requires encoding='latin1'

KEY='transformed/B43_AIRCRAFT_INVENTORY.csv'
```

```
response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
s3_data = io.BytesIO(response.get('Body').read())
aircraft = pd.read_csv(s3_data, encoding='latin1')
aircraft.head()
```

[4]:

```
   MANUFACTURE_YEAR TAIL_NUM  NUMBER_OF_SEATS
0              1944   N54514              0.0
1              1945   N1651M              0.0
2              1953   N100CE              0.0
3              1953   N141FL              0.0
4              1953   N151FL              0.0
```

[5]:

```
# INGEST CARRIER NAMES DICTIONARY

KEY='transformed/CARRIER_DECODE.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
names = pd.read_csv(response.get("Body"))
names.head()
```

[5]:

```
   AIRLINE_ID OP_UNIQUE_CARRIER  CARRIER_NAME
0       21754               2PQ    21 Air LLC
1       20342                Q5   40-Mile Air
2       20342               WRB   40-Mile Air
3       19627               CIQ    A/S Conair
4       19072               AAE  AAA Airlines
```

[6]:

```
# INGEST CARRIER EMPLOYEE / STAFFING DATA

KEY='transformed/P10_EMPLOYEES.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
employees = pd.read_csv(response.get("Body"))
employees.head()
```

[6]:

```
  OP_UNIQUE_CARRIER  PILOTS_COPILOTS  PASSENGER_HANDLING  PASS_GEN_SVC_ADMIN  \
0                AA             8586                8586               15502
1                AS             2893                1062                5737
2                B6             2840                4905                3888
3                DL             9293               15331               15809
4                F9             1473                2496                 154

   MAINTENANCE
0         9677
1          898
2          726
```

```
3         6122
4          237
```

[7]:
```
# INGEST DECEMBER 2019 DAILY WEATHER OBSERVATIONS

KEY='transformed/airport_weather_dec_2019.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
weather_report = pd.read_csv(response.get("Body"))
weather_report.head()
```

[7]:
```
        DATE                                               NAME  PRCP  SNOW  \
0  12/1/2019  ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO…  0.04  64.0
1  12/1/2019      AUSTIN BERGSTROM INTERNATIONAL AIRPORT, TX US  0.00  62.0
2  12/1/2019  BALTIMORE WASHINGTON INTERNATIONAL AIRPORT, MD US  0.62  41.0
3  12/1/2019                   CHARLOTTE DOUGLAS AIRPORT, NC US  0.60  56.0
4  12/1/2019   CINCINNATI MUNICIPAL AIRPORT LUNKEN FIELD, OH US  0.09   NaN

   SNWD   TMAX   AWND
0  67.0  270.0  16.11
1  66.0  360.0  10.29
2  45.0   70.0   8.05
3  68.0  230.0  10.29
4  60.0  250.0  11.41
```

[8]:
```
# INGEST CITY AND AIRPORT NAME DICTIONARY

KEY='transformed/airports_list.csv'

response = s3_client.get_object(Bucket=BUCKET, Key=KEY)
cities = pd.read_csv(response.get("Body"))
cities.head()
```

[8]:
```
   ORIGIN_AIRPORT_ID               DISPLAY_AIRPORT_NAME ORIGIN_CITY_NAME  \
0              12992                        Adams Field  Little Rock, AR
1              10257                Albany International       Albany, NY
2              10140  Albuquerque International Sunport  Albuquerque, NM
3              10299             Anchorage International    Anchorage, AK
4              10397                   Atlanta Municipal      Atlanta, GA

                                             NAME
0                  NORTH LITTLE ROCK AIRPORT, AR US
1               ALBANY INTERNATIONAL AIRPORT, NY US
2          ALBUQUERQUE INTERNATIONAL AIRPORT, NM US
3  ANCHORAGE TED STEVENS INTERNATIONAL AIRPORT, A…
4  ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO…
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DESCRIBE DEC_FLIGHT DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[9]:
```
# DESCRIBE SHAPE
dec_flight.shape
```

[9]:
```
(625763, 18)
```

[10]:
```
# DESCRIBE FEATURE TYPES (object = categorical string features)
dec_flight.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 625763 entries, 0 to 625762
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   DAY_OF_MONTH         625763 non-null  int64
 1   DAY_OF_WEEK          625763 non-null  int64
 2   OP_UNIQUE_CARRIER    625763 non-null  object
 3   TAIL_NUM             625306 non-null  object
 4   ORIGIN               625763 non-null  object
 5   DEST                 625763 non-null  object
 6   DEP_DEL15            620253 non-null  float64
 7   DEP_TIME_BLK         625763 non-null  object
 8   ARR_TIME_BLK         625763 non-null  object
 9   CANCELLED            625763 non-null  float64
 10  CRS_ELAPSED_TIME     625763 non-null  float64
 11  DISTANCE             625763 non-null  float64
 12  DISTANCE_GROUP       625763 non-null  int64
 13  CARRIER_DELAY        126945 non-null  float64
 14  WEATHER_DELAY        126945 non-null  float64
 15  NAS_DELAY            126945 non-null  float64
 16  SECURITY_DELAY       126945 non-null  float64
 17  LATE_AIRCRAFT_DELAY  126945 non-null  float64
dtypes: float64(9), int64(3), object(6)
memory usage: 85.9+ MB
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **DEC_FLIGHT FEATURES: NAME, DESCRIPTION, DROP RATIONALE (IF APPROPRIATE)** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DAY_OF_MONTH - Day of the Month DAY_OF_WEEK - Day of the Week - 1= MON, 2=TUES, 3=WED, 4=THUR, 5=FRI, 6=SAT, 7=SUN, 9=UNKNOWN OP_UNIQUE_CARRIER - Unique Carrier Code (ID) - NEEDED TO MERGE WITH EMPLOYEES AND NAMES TAIL_NUM - Aircraft Tail # (ID) - NEEDED TO MERGE WITH AIRCRAFT ORIGIN - Departure Airport ID (LINKS TO WEATHER STATIONS) DEST - Arrival Airport ID (LINKS TO WEATHER STATIONS) DEP_DEL15 - Departure Delay Indicator, 15 minutes or more(1=YES) - **TARGET FEATURE** DEP_TIME_BLK - Scheduled Departure Time block, Hourly interval ARR_TIME_BLK - Scheduled Arrival Time Block, HOURLY

INTERVALS CANCELLED - Cancelled Flight Indicator (1=YES) CRS_ELAPSED_TIME - Scheduled Elapsed Time DISTANCE - Distance Traveled in Miles DISTANCE_GROUP - Distance Traveled block in increments of 250 miles (1 < 250, 2 = 250-499, …, 11=2500+ miles CARRIER_DELAY - Carrier Delay in Minutes WEATHER_DELAY - Extreme Weather Delay in Minutes NAS_DELAY - National Air System Delay in Minutes SECURITY_DELAY - Security Delay in Minutes LATE_AIRCRAFT_DELAY - Late Arrival Delay in Minutes

[11]:
```
# EXPLORE MISSINGNESS OF EACH FEATURE
dec_flight.isna().sum()
```

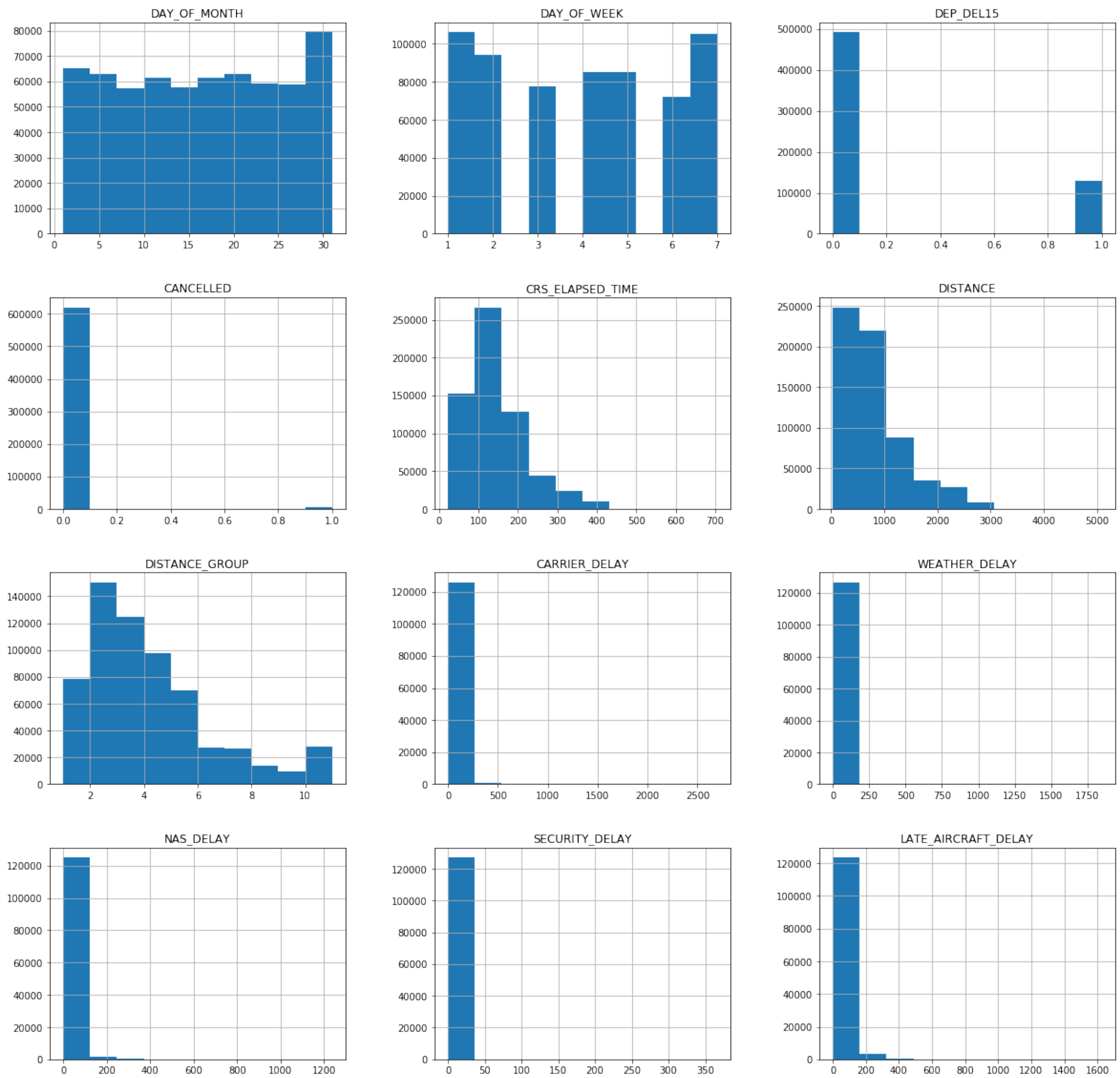
[11]:
```
DAY_OF_MONTH                0
DAY_OF_WEEK                 0
OP_UNIQUE_CARRIER           0
TAIL_NUM                  457
ORIGIN                      0
DEST                        0
DEP_DEL15                5510
DEP_TIME_BLK                0
ARR_TIME_BLK                0
CANCELLED                   0
CRS_ELAPSED_TIME            0
DISTANCE                    0
DISTANCE_GROUP              0
CARRIER_DELAY          498818
WEATHER_DELAY          498818
NAS_DELAY              498818
SECURITY_DELAY         498818
LATE_AIRCRAFT_DELAY    498818
dtype: int64
```

FEATURE MISSINGNESS NOTES:

1 - 457 observations with missing Tail Number. 2 - 5510 observations with missing target variables - drop observations in Module 4. 3 - 498,818 observations missing delay flags/minutes delayed info - likely on time departures.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **DEC_FLIGHT NUMERIC FEATURES DISTRIBUTION** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[12]:
```
# Graph Distributions of numerical features
histlist = dec_flight.hist(figsize = (20,20))
```



Interesting Numeric Feature distribution discussion:

Day of Month - Small variations throughout the month with peak on 30th. Day of Week - Sunday + Monday = Peak | Satuday = low | Tuesday - Friday Variation Cancelled - Cancelled flights need to be removed b/c delay info is not available - Module 4 CRS (Scheduled) Elapsed time - Right-skewed - most flights are shorter in duration Distance / Distance Group - VERY right-skewed with almost 50% of flights below 500 miles. DEP_DELAY15 (Target Variable) - IMBALANCED DATA SET - will need to be balanced before ML Delay codes disprortionally 0 = no delay - difficult to interpret.

0.1 What days of the month are best and worst for departure delays?

[13]:
```
# Explore DAY_OF_MONTH with DEP_DEL15
month = pd.crosstab(dec_flight['DAY_OF_MONTH'], dec_flight['DEP_DEL15'])
month['Total'] = month.sum(axis=1)
month.loc['Total'] = month.sum()
month['Percent_Delayed'] = ((month.iloc[:,1])/((month.iloc[:,0])+(month.iloc[:
 ↪,1])))
month = month.sort_values('Percent_Delayed')
month
```

[13]:

DEP_DEL15 / DAY_OF_MONTH	0.0	1.0	Total	Percent_Delayed
7	14303	1732	16035	0.108014
25	14579	1884	16463	0.114438
8	17762	2459	20221	0.121606
5	18275	2906	21181	0.137198
6	18070	3116	21186	0.147078
10	16760	2909	19669	0.147898
15	17126	3057	20183	0.151464
12	17992	3217	21209	0.151681
24	13902	2769	16671	0.166097
14	13307	2824	16131	0.175067
31	14214	3059	17273	0.177097
27	17091	3851	20942	0.183889
16	16932	3954	20886	0.189313
4	16227	3893	20120	0.193489
26	16736	4232	20968	0.201831
13	16839	4357	21196	0.205558
Total	492096	128157	620253	0.206621
20	16868	4483	21351	0.209967
21	15726	4212	19938	0.211255
11	15720	4410	20130	0.219076
19	16478	4771	21249	0.224528
18	15605	4790	20395	0.234861
9	15855	4907	20762	0.236345
3	15347	4769	20116	0.237075
30	15674	5119	20793	0.246189
29	15595	5144	20739	0.248035
17	14537	4943	19480	0.253747
22	15206	5730	20936	0.273691
23	15125	5795	20920	0.277008
28	14195	5449	19644	0.277387
2	14894	6402	21296	0.300620
1	15156	7014	22170	0.316373

Summary: Days of the month show differences, but no obvious pattern - let's explore day of the week.

0.2 What day of the week is the best and worst for departure delays?

[14]:
```python
# Explore DAY_OF_WEEK with DEP_DEL15
week = pd.crosstab(dec_flight['DAY_OF_WEEK'], dec_flight['DEP_DEL15'])
week['Total'] = week.sum(axis=1)
week.loc['Total'] = week.sum()
week['Percent_Delayed'] = ((week.iloc[:,1])/((week.iloc[:,0])+(week.iloc[:,1])))
week = week.sort_values('Percent_Delayed')
week
```

[14]:

DEP_DEL15	0.0	1.0	Total	Percent_Delayed
DAY_OF_WEEK				
4	69481	15126	84607	0.178780
5	68868	15807	84675	0.186678
3	62131	14977	77108	0.194234
2	74760	18449	93209	0.197932
6	57531	14217	71748	0.198152
Total	492096	128157	620253	0.206621
7	80845	23404	104249	0.224501
1	78480	26177	104657	0.250122

Summary: The best days of the week are 4 (Thursday) @ 17.9% and 5 (Friday) @ 18.7%. The worst days of the week are 1 (Monday) @ 25% and 7 (Sunday) @ 22.5%.

0.3 What distance groups perform best and worst for departure delays?

[15]:
```python
# Explore DISTANCE_GROUP with DEP_DEL15
Dist = pd.crosstab(dec_flight['DISTANCE_GROUP'], dec_flight['DEP_DEL15'])
Dist['Total'] = Dist.sum(axis=1)
Dist.loc['Total'] = Dist.sum()
Dist['Percent_Delayed'] = ((Dist.iloc[:,1])/((Dist.iloc[:,0])+(Dist.iloc[:,1])))
Dist = Dist.sort_values('Percent_Delayed')
Dist
```

[15]:

DEP_DEL15	0.0	1.0	Total	Percent_Delayed
DISTANCE_GROUP				
1	63452	14152	77604	0.182362
3	98656	24926	123582	0.201696
Total	492096	128157	620253	0.206621
4	77200	20107	97307	0.206635
2	117550	30683	148233	0.206992
6	21412	5642	27054	0.208546
10	12621	3554	16175	0.219722
5	54398	15357	69755	0.220156
7	20416	5856	26272	0.222899
9	7205	2086	9291	0.224518
11	8931	2640	11571	0.228157
8	10255	3154	13409	0.235215

**Summary:* The best distance groups are 1 (<250 miles) @ 18.2% and 3 (500-749 miles) @ 21.2%. The worst distance groups are 8 (1750-1999 Miles) @ 23.5% and 11 (>2500 Miles) @ 23.5%.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
DEC_FLIGHT OBJECT FEATURES DISTRIBUTION
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

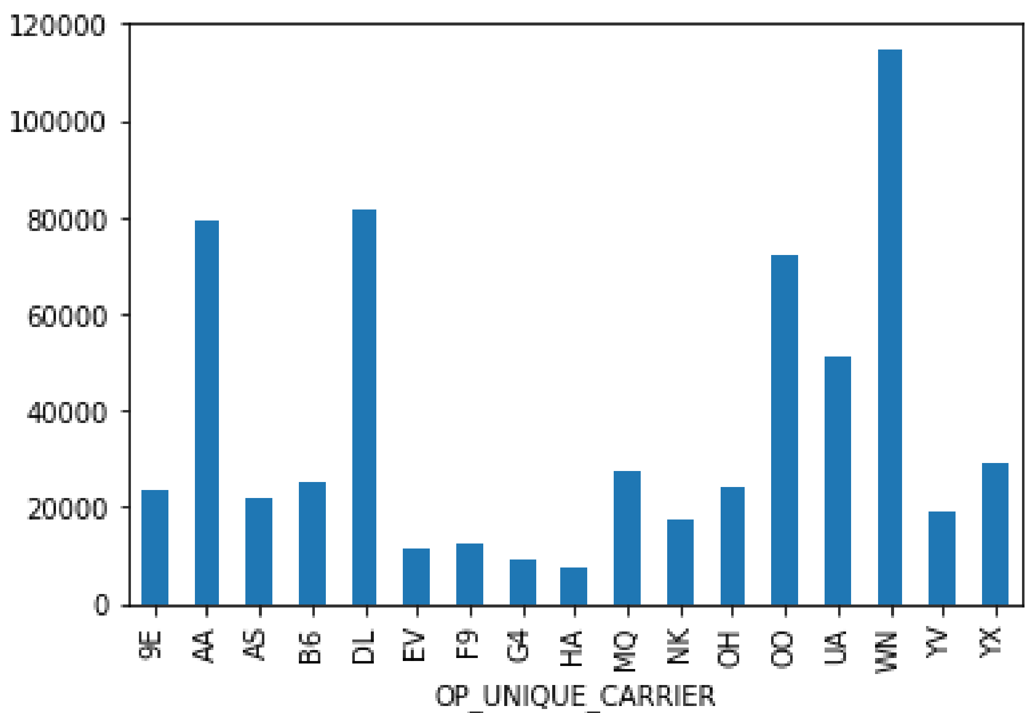
[16]:
```
# Count the unique values of categorical/object-stored features
dec_flight[['OP_UNIQUE_CARRIER', 'TAIL_NUM', 'ORIGIN', 'DEST','DEP_TIME_BLK',␣
↪'ARR_TIME_BLK']].nunique()
```

[16]:
```
OP_UNIQUE_CARRIER      17
TAIL_NUM             5478
ORIGIN                350
DEST                  350
DEP_TIME_BLK           19
ARR_TIME_BLK           19
dtype: int64
```

NOTE:
High cardinality in Tail_Num, Origin, Dest make analysis difficult.

[17]:
```
# Carrier Distribution
dec_flight.groupby('OP_UNIQUE_CARRIER').size().plot.bar()
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d7b7c9ad0>
```



Note: We see a significant variation in the number of flights per carrier.

Of the 17 carriers, AA (American Airlines), DL (Delta Airlines) , OO (Skywest - regional), WN (Southwest), and UA (United Airlines) are the leaders in terms of volume.

0.4 What are the best and worst performing airlines for departure delays?

[18]:
```
# Explore OP_UNIQUE_CARRIER with DEP_DEL15
carrier = pd.crosstab(dec_flight['OP_UNIQUE_CARRIER'], dec_flight['DEP_DEL15'])
carrier['Total'] = carrier.sum(axis=1)
carrier.loc['Total'] = carrier.sum()
carrier['Percent_Delayed'] = ((carrier.iloc[:,1])/((carrier.iloc[:,0])+(carrier.
  ↪iloc[:,1])))
carrier = carrier.sort_values('Percent_Delayed')
print(carrier)
```

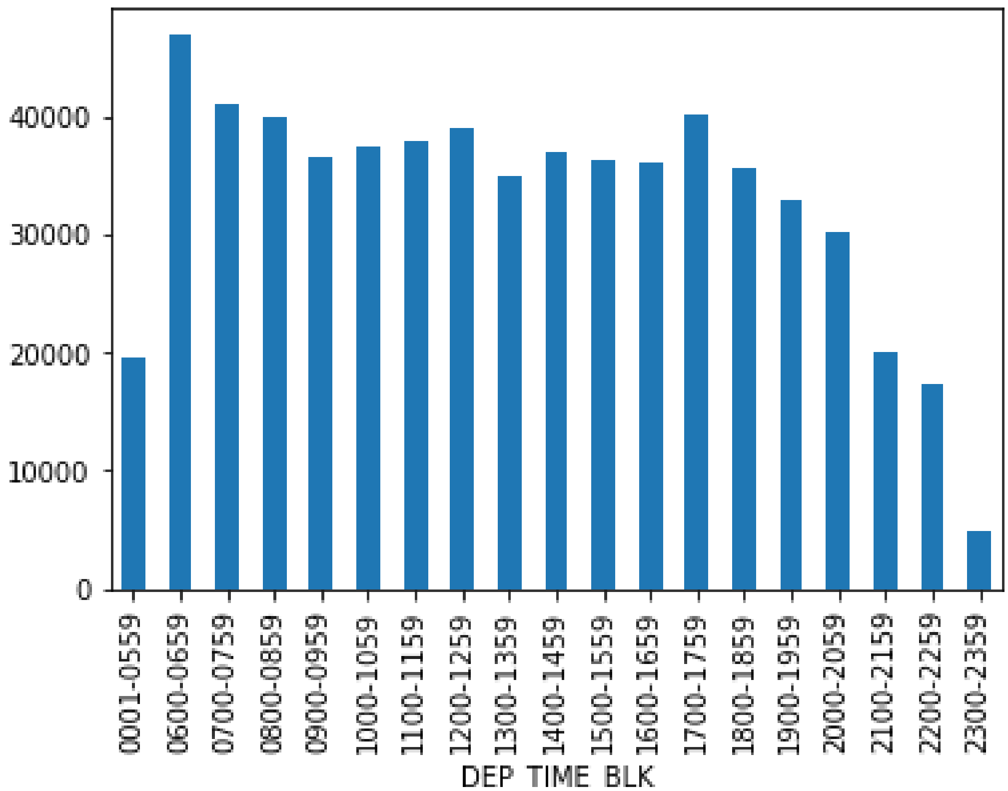
```
DEP_DEL15              0.0     1.0    Total  Percent_Delayed
OP_UNIQUE_CARRIER
HA                    6618     651     7269         0.089558
DL                   68764   12736    81500         0.156270
9E                   19174    3960    23134         0.171177
AA                   65242   14001    79243         0.176684
MQ                   21869    4877    26746         0.182345
NK                   14064    3146    17210         0.182801
YX                   23081    5263    28344         0.185683
UA                   41276    9889    51165         0.193277
OO                   56526   14016    70542         0.198690
Total               492096  128157   620253         0.206621
OH                   18804    5218    24022         0.217218
EV                    8569    2409    10978         0.219439
AS                   16700    5073    21773         0.232995
YV                   14221    4338    18559         0.233741
G4                    7124    2191     9315         0.235212
F9                    9135    3007    12142         0.247653
WN                   83724   29540   113264         0.260807
B6                   17205    7842    25047         0.313091
```

Summary: We note a wide range of percent of departures delayed by carrier. This could indicate that carrier-specific data (such as staffing) could be good indicators for predicting delays.

Mean % Delayed departures = 20.7% Worst performaing carriers = B6 (JetBlue) @ 31.3% and WN (Southwest) @ 26.1% Best performing carriers = HA (Hawaiian Airlines) @ 9% and DL (Delta Airlines) @ 15.6%

[19]:
```
# Departure Time Block distribution
dec_flight.groupby('DEP_TIME_BLK').size().plot.bar()
```

[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6d79ee4290>



Summary: Highest number of departures from 6-659am, lowest from 11-1159pm. Hourly variations exist as well

0.5 What are the best and worst performing time blocks for departure delays?

[20]:
```
# Explore DEPT_TIME_BULK with DEP_DEL15
time_block = pd.crosstab(dec_flight['DEP_TIME_BLK'], dec_flight['DEP_DEL15'])
time_block['Total'] = time_block.sum(axis=1)
time_block.loc['Total'] = time_block.sum()
time_block['Percent_Delayed'] = ((time_block.iloc[:,1])/((time_block.iloc[:
 ↪,0])+(time_block.iloc[:,1])))
time_block = time_block.sort_values('Percent_Delayed')
time_block
```

[20]:

DEP_DEL15	0.0	1.0	Total	Percent_Delayed
DEP_TIME_BLK				
0600-0659	42566	3952	46518	0.084956
0001-0559	17565	1964	19529	0.100568
0700-0759	36653	4376	41029	0.106656

```
0800-0859       34384    5254   39638            0.132550
0900-0959       30593    5706   36299            0.157194
1000-1059       30423    6876   37299            0.184348
1100-1159       30424    7374   37798            0.195090
Total          492096  128157  620253            0.206621
1200-1259       30721    8133   38854            0.209322
1300-1359       27005    7768   34773            0.223392
2300-2359        3738    1098    4836            0.227047
1400-1459       28186    8609   36795            0.233972
1500-1559       27086    8999   36085            0.249383
1600-1659       27000    8989   35989            0.249771
1700-1759       29585   10279   39864            0.257852
2200-2259       12515    4737   17252            0.274577
1800-1859       25551    9795   35346            0.277118
2000-2059       21364    8626   29990            0.287629
2100-2159       13859    5864   19723            0.297318
1900-1959       22878    9758   32636            0.298995
```

Answer: Best times = 6-659am @ 8.5% and 1201am - 6am @ 10% Worst times = 7-759pm @ 29.9% and 9-959pm @ 29.7%

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DESCRIBE AIRCRAFT DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[21]:
```
# DESCRIBE SHAPE
aircraft.shape
```

[21]: (7383, 3)

[22]:
```
# DESCRIBE FEATURE TYPES
aircraft.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7383 entries, 0 to 7382
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   MANUFACTURE_YEAR  7383 non-null   int64
 1   TAIL_NUM          7383 non-null   object
 2   NUMBER_OF_SEATS   7376 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 173.2+ KB
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx AIRCRAFT FEATURES: NAME, DESCRIPTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

(LIST EACH FEATURE AND DESCRIPTOIN)

[23]:
```python
# Count the unique values per feature
aircraft.nunique()
```

[23]:
```
MANUFACTURE_YEAR      62
TAIL_NUM            7361
NUMBER_OF_SEATS      121
dtype: int64
```

[24]:
```python
# DESCRIBE MISSINGNESS OF FEATURES
aircraft.isna().sum()
```

[24]:
```
MANUFACTURE_YEAR    0
TAIL_NUM            0
NUMBER_OF_SEATS     7
dtype: int64
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DESCRIBE NAMES DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[25]:
```python
# DESCRIBE SHAPE
names.shape
```

[25]:
```
(1744, 3)
```

[26]:
```python
# DESCRIBE FEATURE TYPES
names.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1744 entries, 0 to 1743
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   AIRLINE_ID         1744 non-null   int64
 1   OP_UNIQUE_CARRIER  1743 non-null   object
 2   CARRIER_NAME       1744 non-null   object
dtypes: int64(1), object(2)
memory usage: 41.0+ KB
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx NAMES FEATURES: NAME, DESCRIPTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

(LIST EACH FEATURE AND DESCRIPTION)

[27]:
```python
# Count the unique values per feature
names.nunique()
```

[27]:
```
AIRLINE_ID           1597
OP_UNIQUE_CARRIER    1743
CARRIER_NAME         1606
```

```
dtype: int64
```

[28]:
```
# DESCRIBE MISSINGNESS OF FEATURES
names.isna().sum()
```

[28]:
```
AIRLINE_ID           0
OP_UNIQUE_CARRIER    1
CARRIER_NAME         0
dtype: int64
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
DESCRIBE EMPLOYEES DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[29]:
```
# DESCRIBE SHAPE
employees.shape
```

[29]:
```
(16, 5)
```

[30]:
```
# DESCRIBE FEATURE TYPES
employees.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16 entries, 0 to 15
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   OP_UNIQUE_CARRIER   16 non-null     object
 1   PILOTS_COPILOTS     16 non-null     int64
 2   PASSENGER_HANDLING  16 non-null     int64
 3   PASS_GEN_SVC_ADMIN  16 non-null     int64
 4   MAINTENANCE         16 non-null     int64
dtypes: int64(4), object(1)
memory usage: 768.0+ bytes
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
EMPLOYEES FEATURES: NAME, DESCRIPTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

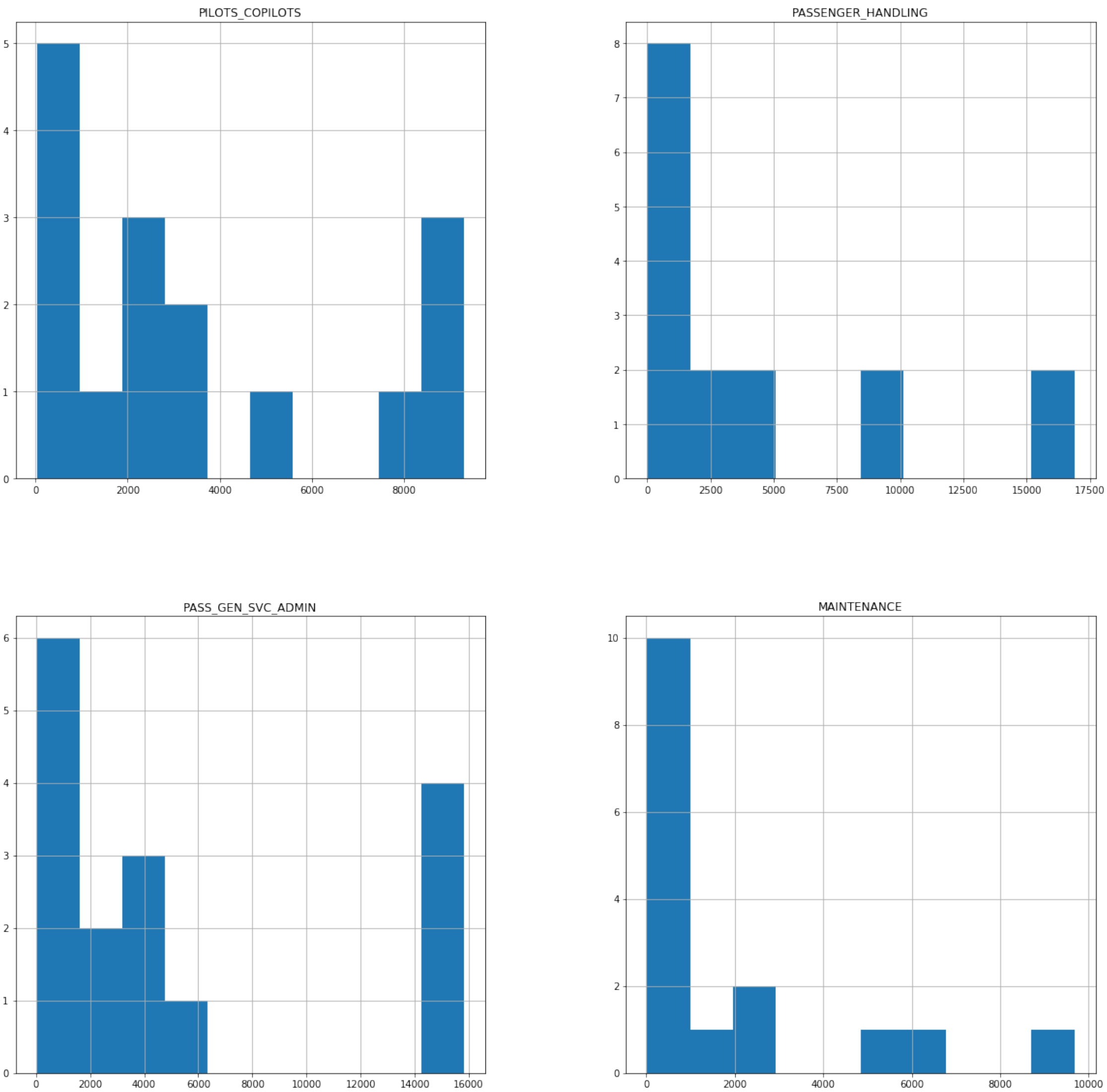
(LIST FEATURES AND DESCRIPTION)

[31]:
```
# Count the unique values per feature
employees.nunique()
```

[31]:
```
OP_UNIQUE_CARRIER     16
PILOTS_COPILOTS       16
PASSENGER_HANDLING    16
PASS_GEN_SVC_ADMIN    16
MAINTENANCE           16
dtype: int64
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
EMPLOYEES FEATURES DISTRIBUTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[32]:
```
# Graph Distributions of numerical features
histlist3 = employees.hist(figsize = (20, 20))
```



[33]:
```
# DESCRIBE MISSINGNESS OF FEATURES
employees.isna().sum()
```

[33]:
```
OP_UNIQUE_CARRIER     0
PILOTS_COPILOTS       0
PASSENGER_HANDLING    0
PASS_GEN_SVC_ADMIN    0
MAINTENANCE           0
```

```
dtype: int64
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DESCRIBE WEATHER_REPORT DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[34]:
```
# DESCRIBE SHAPE
weather_report.shape
```

[34]:
```
(3286, 7)
```

[35]:
```
# DESCRIBE FEATURE TYPES
weather_report.info()
```

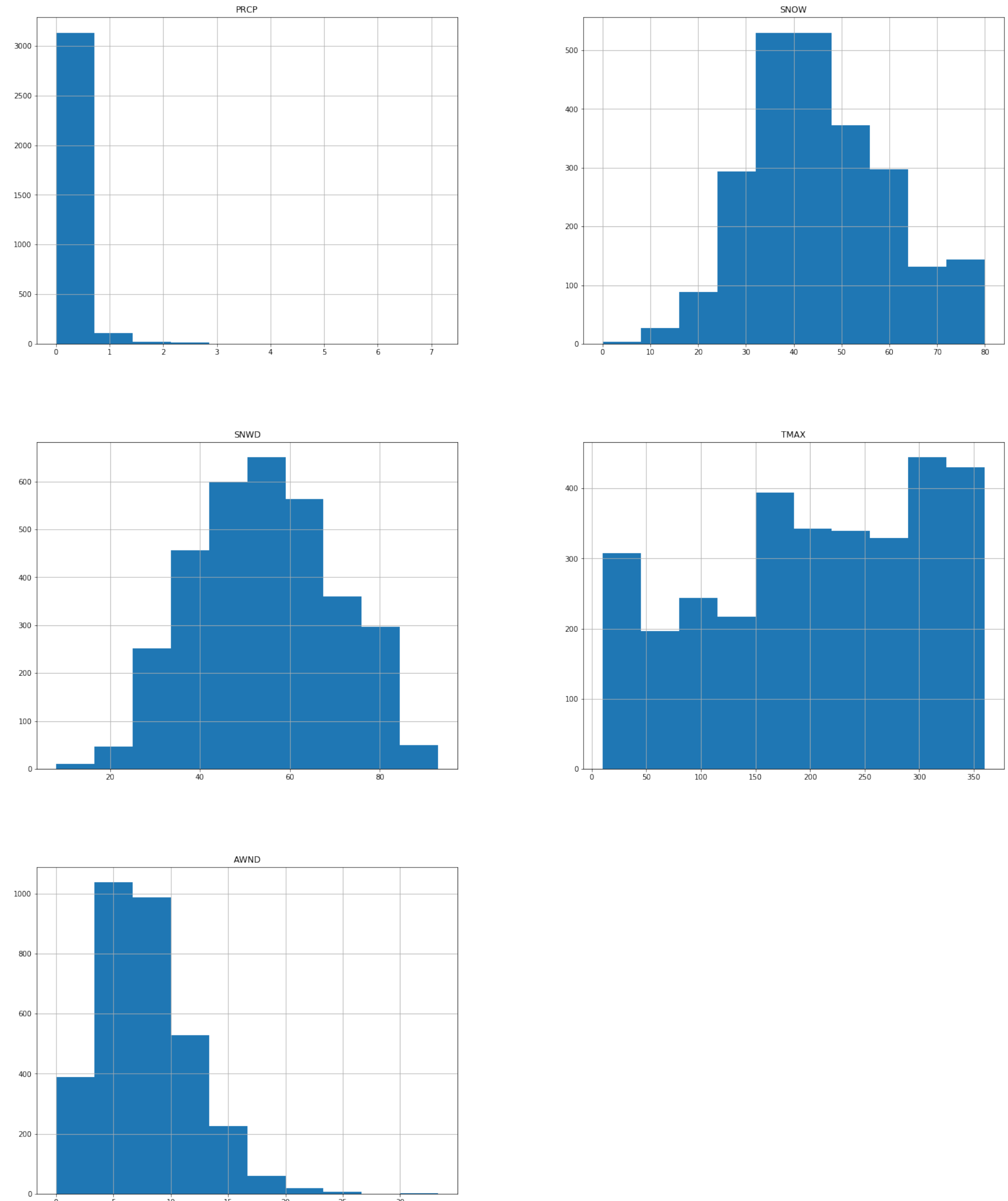
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3286 entries, 0 to 3285
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   DATE    3286 non-null   object
 1   NAME    3286 non-null   object
 2   PRCP    3286 non-null   float64
 3   SNOW    2418 non-null   float64
 4   SNWD    3284 non-null   float64
 5   TMAX    3244 non-null   float64
 6   AWND    3255 non-null   float64
dtypes: float64(5), object(2)
memory usage: 179.8+ KB
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx WEATHER_REPORT FEATURES: NAME, DESCRIPTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

(LIST FEATURES AND DESCRIPTION)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx WEATHER_REPORT FEATURES DISTRIBUTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[36]:
```
# Graph Distributions of numerical features
histlist3 = weather_report.hist(figsize = (24, 30))
```




```
[37]: # DESCRIBE MISSINGNESS OF FEATURES
      weather_report.isna().sum()
```

```
[37]: DATE     0
      NAME     0
      PRCP     0
```

```
SNOW    868
SNWD      2
TMAX     42
AWND     31
dtype: int64
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx DESCRIBE CITIES DATA xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

[38]:
```
# DESCRIBE SHAPE
cities.shape
```

[38]:
```
(97, 4)
```

[39]:
```
# DESCRIBE FEATURE TYPES
cities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97 entries, 0 to 96
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ORIGIN_AIRPORT_ID     97 non-null     int64
 1   DISPLAY_AIRPORT_NAME  97 non-null     object
 2   ORIGIN_CITY_NAME      97 non-null     object
 3   NAME                  97 non-null     object
dtypes: int64(1), object(3)
memory usage: 3.2+ KB
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx CITIES FEATURES: NAME, DESCRIPTION xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

(LIST CITY FEATURES AND DESCRIPTIONS)

[40]:
```
# Count the unique values per feature
cities.nunique()
```

[40]:
```
ORIGIN_AIRPORT_ID       97
DISPLAY_AIRPORT_NAME    97
ORIGIN_CITY_NAME        94
NAME                    86
dtype: int64
```

[41]:
```
# DESCRIBE MISSINGNESS OF FEATURES
cities.isna().sum()
```

[41]:
```
ORIGIN_AIRPORT_ID       0
DISPLAY_AIRPORT_NAME    0
ORIGIN_CITY_NAME        0
```

```
NAME                        0
dtype: int64
```

[ ]:

```

```