

United States Airlines Analysis

```
In [1]: # Let's import the necessary library.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
In [2]: # Let's remove the unnecessary warnings.
import warnings
warnings.filterwarnings("ignore")
```

Project Task: Week 1 (Applied data science with Python)

1. Import and aggregate data:

a. Collect information related to flights, airports (e.g., type of airport and elevation), and runways (e.g., length_ft, width_ft, surface, and number of runways). Gather all fields you believe might cause avoidable delays in one dataset.

Hint: In this case, you would have to determine the keys to join the tables. A data description will be useful.

```
In [3]: # Now Let's import the data for the further operation.
airline = pd.read_excel("Airlines.xlsx")
```

```
In [4]: airline.shape
```

```
Out[4]: (518556, 9)
```

```
In [5]: airline.head()
```

```
Out[5]:
```

	id	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
0	1	CO	269	SFO	IAH	3	15	205	1
1	2	US	1558	PHX	CLT	3	15	222	1
2	3	AA	2400	LAX	DFW	3	20	165	1
3	4	AA	2466	SFO	DFW	3	20	195	1
4	5	AS	108	ANC	SEA	3	30	202	0

```
In [6]: airport = pd.read_excel("airports.xlsx")
```

```
In [7]: airport.shape
```

```
Out[7]: (73805, 18)
```

In [8]:

airpot.head()

Out[8]:

	id	ident	type	name	latitude_deg	longitude_deg	elevation_ft	continent	iso_country	iso_region
0	6523	00A	heliport	Total Rf Heliport	40.070801	-74.933601	11.0	NaN	US	US-PA
1	323361	00AA	small_airport	Aero B Ranch Airport	38.704022	-101.473911	3435.0	NaN	US	US-KS
2	6524	00AK	small_airport	Lowell Field	59.947733	-151.692524	450.0	NaN	US	US-AK
3	6525	00AL	small_airport	Epps Airpark	34.864799	-86.770302	820.0	NaN	US	US-AL
4	6526	00AR	closed	Newport Hospital & Clinic Heliport	35.608700	-91.254898	237.0	NaN	US	US-AR

In [9]:

runway = pd.read_excel("runways.xlsx")

In [10]:

runway.shape

Out[10]:

(43977, 20)

In [11]:

runway.head()

Out[11]:

	id	airport_ref	airport_ident	length_ft	width_ft	surface	lighted	closed	le_ident	le_latitude_deg	le_longitude_deg
0	269408	6523	00A	80.0	80.0	ASPH-G	1	0	H1	NaN	NaN
1	255155	6524	00AK	2500.0	70.0	GRVL	0	0	N	NaN	NaN
2	254165	6525	00AL	2300.0	200.0	TURF	0	0	1	NaN	NaN
3	270932	6526	00AR	40.0	40.0	GRASS	0	0	H1	NaN	NaN
4	322128	322127	00AS	1450.0	60.0	Turf	0	0	1	NaN	NaN

In [12]: *# Before merging the data Lets drop the columns that will not play an important role in the model*
 runway.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43977 entries, 0 to 43976
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                     43977 non-null  int64
1   airport_ref                           43977 non-null  int64
2   airport_ident                           43977 non-null  object
3   length_ft                             43753 non-null  float64
4   width_ft                              41088 non-null  float64
5   surface                               43520 non-null  object
6   lighted                               43977 non-null  int64
7   closed                                43977 non-null  int64
8   le_ident                               43793 non-null  object
9   le_latitude_deg                       15016 non-null  float64
10  le_longitude_deg                      15000 non-null  float64
11  le_elevation_ft                       12781 non-null  float64
12  le_heading_degT                       14624 non-null  float64
13  le_displaced_threshold_ft             2883 non-null  float64
14  he_ident                               37332 non-null  object
15  he_latitude_deg                       14971 non-null  float64
16  he_longitude_deg                      14973 non-null  float64
17  he_elevation_ft                       12620 non-null  float64
18  he_heading_degT                       16428 non-null  float64
19  he_displaced_threshold_ft             3176 non-null  float64
dtypes: float64(12), int64(4), object(4)
memory usage: 6.7+ MB
```

In [13]: `runways = runway.drop(['le_ident', 'le_latitude_deg', 'le_longitude_deg', 'le_elevation_ft', 'le_displaced_threshold_ft', 'he_ident', 'he_latitude_deg', 'he_longitude_deg', 'he_elevation_ft', 'he_displaced_threshold_ft'], axis = 1)`

In [14]: `runways`

Out[14]:

	id	airport_ref	airport_ident	length_ft	width_ft	surface	lighted	closed
0	269408	6523	00A	80.0	80.0	ASPH-G	1	0
1	255155	6524	00AK	2500.0	70.0	GRVL	0	0
2	254165	6525	00AL	2300.0	200.0	TURF	0	0
3	270932	6526	00AR	40.0	40.0	GRASS	0	0
4	322128	322127	00AS	1450.0	60.0	Turf	0	0
...
43972	235186	27243	ZYTX	10499.0	148.0	CON	1	0
43973	235169	27244	ZYYJ	8530.0	148.0	CON	1	0
43974	354997	317861	ZYYK	8202.0	NaN	NaN	0	0
43975	346789	346788	ZZ-0003	1800.0	15.0	Turf	0	0
43976	313663	313629	ZZZZ	1713.0	82.0	concrete	0	0

43977 rows × 8 columns

```
In [15]: # Now Lets remove the feature from the airport data that is not usefull.
airport.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73805 entries, 0 to 73804
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    73805 non-null  int64
1   ident                 73805 non-null  object
2   type                 73805 non-null  object
3   name                 73805 non-null  object
4   latitude_deg         73805 non-null  float64
5   longitude_deg        73805 non-null  float64
6   elevation_ft         59683 non-null  float64
7   continent            38086 non-null  object
8   iso_country          73546 non-null  object
9   iso_region          73805 non-null  object
10  municipality         68739 non-null  object
11  scheduled_service    73805 non-null  object
12  gps_code             42996 non-null  object
13  iata_code            9160 non-null   object
14  local_code          32975 non-null  object
15  home_link            3492 non-null   object
16  wikipedia_link       10705 non-null  object
17  keywords             13951 non-null  object
dtypes: float64(3), int64(1), object(14)
memory usage: 10.1+ MB
```

```
In [16]: airpots = airport.drop(['continent', 'iso_country', 'iso_region', 'municipality', 'gps_code', 'local_code',
                                'wikipedia_link', 'keywords'], axis=1)
```

```
In [17]: airpots
```

Out[17]:

	id	ident	type	name	latitude_deg	longitude_deg	elevation_ft	scheduled_service	iata_code
0	6523	00A	heliport	Total Rf Heliport	40.070801	-74.933601	11.0	no	NaN
1	323361	00AA	small_airport	Aero B Ranch Airport	38.704022	-101.473911	3435.0	no	NaN
2	6524	00AK	small_airport	Lowell Field	59.947733	-151.692524	450.0	no	NaN
3	6525	00AL	small_airport	Epps Airpark	34.864799	-86.770302	820.0	no	NaN
4	6526	00AR	closed	Newport Hospital & Clinic Heliport	35.608700	-91.254898	237.0	no	NaN
...
73800	46378	ZZ-0001	heliport	Sealand Helipad	51.894444	1.482500	40.0	no	NaN
73801	307326	ZZ-0002	small_airport	Glorioso Islands Airstrip	-11.584278	47.296389	11.0	no	NaN
73802	346788	ZZ-0003	small_airport	Fainting Goat Airport	32.110587	-97.356312	690.0	no	NaN
73803	342102	ZZZW	closed	Scandium City Heliport	69.355287	-138.939310	4.0	no	ZYW
73804	313629	ZZZZ	small_airport	Satsuma Iijima Airport	30.784722	130.270556	338.0	no	NaN

73805 rows × 9 columns

```
In [18]: # Now Lets merge the runways and airport data.
airpot_runway = pd.merge(airpots, runways, left_on = "ident", right_on = "airport_ident")
airpot_runway.drop(['id_x', 'id_y'], axis=1, inplace=True)
```

```
In [19]: airpot_runway
```

Out[19]:

	ident	type	name	latitude_deg	longitude_deg	elevation_ft	scheduled_service	iata_code
0	00A	heliport	Total Rf Heliport	40.070801	-74.933601	11.0	no	NaN
1	00AK	small_airport	Lowell Field	59.947733	-151.692524	450.0	no	NaN
2	00AL	small_airport	Epps Airpark	34.864799	-86.770302	820.0	no	NaN
3	00AR	closed	Newport Hospital & Clinic Heliport	35.608700	-91.254898	237.0	no	NaN
4	00AS	small_airport	Fulton Airport	34.942803	-97.818019	1100.0	no	NaN
...
43972	ZYTX	large_airport	Shenyang Taoxian International Airport	41.639801	123.483002	198.0	yes	SHE
43973	ZYYJ	medium_airport	Yanji Chaoyangchuan Airport	42.882801	129.451004	624.0	yes	YNJ
43974	ZYYK	medium_airport	Yingkou Lanqi Airport	40.542524	122.358600	NaN	yes	YKH
43975	ZZ-0003	small_airport	Fainting Goat Airport	32.110587	-97.356312	690.0	no	NaN
43976	ZZZZ	small_airport	Satsuma Iscjima Airport	30.784722	130.270556	338.0	no	NaN

43977 rows × 15 columns



```
In [20]: # Now Lets merge the final column airline.
final_df = pd.merge(airline,airpot_runway,how = "inner", left_on = "AirportFrom", right_on = "i
```

```
In [21]: final_df.drop_duplicates(subset=['id'], keep='first', inplace=True)
```

```
In [22]: final_df
```

Out[22]:

	id	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay	ident	...	elevation_ft	s
0	1	CO	269	SFO	IAH	3	15	205	1	KSFO	...	13.0	
4	4	AA	2466	SFO	DFW	3	20	195	1	KSFO	...	13.0	
8	9	DL	2606	SFO	MSP	3	35	216	1	KSFO	...	13.0	
12	129	DL	1580	SFO	DTW	3	345	270	0	KSFO	...	13.0	
16	150	UA	756	SFO	DEN	3	348	158	0	KSFO	...	13.0	
...	
2160266	451344	CO	2	GUM	HNL	1	400	430	1	PGUM	...	298.0	
2160268	469866	CO	2	GUM	HNL	2	400	430	1	PGUM	...	298.0	
2160270	488365	CO	2	GUM	HNL	3	400	430	0	PGUM	...	298.0	
2160272	506855	CO	2	GUM	HNL	4	400	430	1	PGUM	...	298.0	
2160274	525138	CO	2	GUM	HNL	5	400	430	1	PGUM	...	298.0	

518525 rows × 24 columns



b. When it comes to on-time arrivals, different airlines perform differently based on the amount of experience they have. The major airlines in this field include US Airways Express (founded in 1967) Continental Airlines (founded in 1934), and Express Jet (founded in 19860. Pull such information specific to various airlines

from the Wikipedia page link given below.https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States (https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States).

Hint: Here, you should use web scraping to learn how long an airline has been operating.

```
In [23]: # Now Lets use the web scrapping to import the data frome the wikipedia.
url = "https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States"
tables = pd.read_html(url)
```

```
In [24]: print(tables)
```

	Airline	Image	IATA	ICAO	Callsign	\
0	Alaska Airlines	NaN	AS	ASA	ALASKA	
1	Allegiant Air	NaN	G4	AAY	ALLEGiant	
2	American Airlines	NaN	AA	AAL	AMERICAN	
3	Avelo Airlines	NaN	XP	VXP	AVELO	
4	Breeze Airways	NaN	MX	MXV	MOXY	
5	Delta Air Lines	NaN	DL	DAL	DELTA	
6	Eastern Airlines	NaN	2D	EAL	EASTERN	
7	Frontier Airlines	NaN	F9	FFT	FRONTIER FLIGHT	
8	Hawaiian Airlines	NaN	HA	HAL	HAWAIIAN	
9	JetBlue	NaN	B6	JBU	JETBLUE	
10	Southwest Airlines	NaN	WN	SWA	SOUTHWEST	
11	Spirit Airlines	NaN	NK	NKS	SPIRIT WINGS	
12	Sun Country Airlines	NaN	SY	SCX	SUN COUNTRY	
13	United Airlines	NaN	UA	UAL	UNITED	

	Primary hubs	Secondary hubs	Founded	\
0	Seattle/Tacoma	AnchoragePortland (OR)San Franci...	1932	
1	Las Vegas	CincinnatiFort Walton BeachIndianapol...	1997	

In [25]:

tables[0]

Out[25]:

	Airline	Image	IATA	ICAO	Callsign	Primary hubs, Secondary hubs	Founded	Notes
0	Alaska Airlines	NaN	AS	ASA	ALASKA	Seattle/TacomaAnchoragePortland (OR)San Franci...	1932	Founded as McGee Airways and commenced operati...
1	Allegiant Air	NaN	G4	AAY	ALLEGIANT	Las VegasCincinnatiFort Walton BeachIndianapol...	1997	Founded as WestJet Express and commenced opera...
2	American Airlines	NaN	AA	AAL	AMERICAN	WorthCharlotteChicago-Dallas/Fort WorthO'HareLos An...	1926	Founded as American Airways and commenced oper...
3	Avelo Airlines	NaN	XP	VXP	AVELO	BurbankNew HavenOrlando	1987	First did business as Casino Express Airlines ...
4	Breeze Airways	NaN	MX	MXV	MOXY	CharlestonHartfordNew OrleansNorfolkProvoTampa	2018	NaN
5	Delta Air Lines	NaN	DL	DAL	DELTA	AtlantaBostonDetroitLos AngelesMinneapolis/St....	1924	Founded as Huff Daland Dusters and commenced o...
6	Eastern Airlines	NaN	2D	EAL	EASTERN	MiamiNew York-JFK	2010	NaN
7	Frontier Airlines	NaN	F9	FFT	FRONTIER FLIGHT	DenverAtlantaChicago-O'HareCincinnatiCleveland...	1994	NaN
8	Hawaiian Airlines	NaN	HA	HAL	HAWAIIAN	HonoluluKahului	1929	Founded as Inter-Island Airways in early 1929 ...
9	JetBlue	NaN	B6	JBU	JETBLUE	New York-JFKBostonLos AngelesFort LauderdaleOr...	1998	Founded as New Air and commenced operations in...
10	Southwest Airlines	NaN	WN	SWA	SOUTHWEST	LoveAtlantaBaltimoreChicago-MidwayDenve...	1967	Founded as Air Southwest and commenced operati...
11	Spirit Airlines	NaN	NK	NKS	SPIRIT WINGS	Atlantic CityDetroitLas VegasFort LauderdaleCh...	1980	Founded as Charter One.
12	Sun Country Airlines	NaN	SY	SCX	SUN COUNTRY	Minneapolis/St. PaulDallas/Fort WorthLas Vegas	1982	Commenced operations in 1983.Operates some Ama...
13	United Airlines	NaN	UA	UAL	UNITED	Chicago-O'HareDenverGuamHouston-Intercontinent...	1926	Founded as Varney Air Lines and commenced oper...

In [26]:

tables[6]

Out[26]:

	Airline	Image	IATA	ICAO	Callsign	Primary Hubs, Secondary Hubs	Founded	Notes
0	Comco	NaN	NaN	NaN	NaN	NaN	2002	NaN
1	Janet	NaN	NaN	WWW	JANET	Las Vegas	1972	NaN
2	Justice Prisoner and Alien Transportation System	NaN	NaN	JUD	JUSTICE	Oklahoma City	1980	Commenced operations in 1995.

In [27]:

Lets first merge all wikipedia table.
wiki_table = [tables[0],tables[1],tables[2],tables[3],tables[4],tables[5],tables[6]]

In [28]:

wiki_tables = pd.concat(wiki_table, ignore_index=True)

In [29]:

wiki_tables

Out[29]:

	Airline	Image	IATA	ICAO	Callsign	Primary hubs, Secondary hubs	Founded	Notes	Primary Hub Secondary Hub
0	Alaska Airlines	NaN	AS	ASA	ALASKA	Seattle/TacomaAnchoragePortland (OR)San Franci...	1932.0	Founded as McGee Airways and commenced operati...	Na
1	Allegiant Air	NaN	G4	AAY	ALLEGiant	Las VegasCincinnatiFort Walton BeachIndianapol...	1997.0	Founded as WestJet Express and commenced opera...	Na
2	American Airlines	NaN	AA	AAL	AMERICAN	WorthCharlotteChicago-O'HareLos An...	1926.0	Founded as American Airways and commenced oper...	Na
3	Avelo Airlines	NaN	XP	VXP	AVELO	BurbankNew HavenOrlando	1987.0	First did business as Casino Express Airlines ...	Na
4	Breeze Airways	NaN	MX	MXV	MOXY	CharlestonHartfordNew OrleansNorfolkProvoTampa	2018.0	NaN	Na
...	
137	Lifestar	NaN	NaN	NaN	NaN		NaN	NaN	Na
138	Life Lion	NaN	NaN	NaN	NaN		NaN	NaN	Na
139	Comco	NaN	NaN	NaN	NaN		NaN	2002.0	Na
140	Janet	NaN	NaN	WWW	JANET		NaN	1972.0	NaN
141	Justice Prisoner and Alien Transportation System	NaN	NaN	JUD	JUSTICE		NaN	1980.0	Commenced operations in 1995.

142 rows × 9 columns



c. You should then get all the information gathered so far in one place.


```
In [30]: # First we got only that column from wiki pedia table that we need to merge.
wiki_df = wiki_tables[['IATA', "Founded"]]
wiki_df
```

```
Out[30]:
```

	IATA	Founded
0	AS	1932.0
1	G4	1997.0
2	AA	1926.0
3	XP	1987.0
4	MX	2018.0
...
137	NaN	NaN
138	NaN	NaN
139	NaN	2002.0
140	NaN	1972.0
141	NaN	1980.0

142 rows × 2 columns

```
In [31]: # Now we gather all the information that we got from wiki pedia link and the data that we have.
df = final_df.merge(wiki_df, left_on = 'Airline', right_on = "IATA")
```

```
In [32]: df
```

```
Out[32]:
```

	id	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay	ident	...	iata_code	airp
0	4	AA	2466	SFO	DFW	3	20	195	1	KSFO	...	SFO	
1	231	AA	526	SFO	DFW	3	360	215	0	KSFO	...	SFO	
2	234	AA	552	SFO	MIA	3	360	315	1	KSFO	...	SFO	
3	905	AA	810	SFO	ORD	3	385	255	0	KSFO	...	SFO	
4	1739	AA	24	SFO	JFK	3	425	325	1	KSFO	...	SFO	
...	
434919	497838	9E	4292	LWB	JFK	3	890	110	1	KLWB	...	LWB	
434920	516333	9E	4292	LWB	JFK	4	890	110	0	KLWB	...	LWB	
434921	534123	9E	4292	LWB	JFK	5	890	110	0	KLWB	...	LWB	
434922	69058	9E	3752	ABR	MSP	7	410	76	1	KABR	...	ABR	
434923	189396	9E	3752	ABR	MSP	7	410	76	0	KABR	...	ABR	

434924 rows × 26 columns

d. The total passenger traffic may also contribute to flight delays. The term hub refers to busy commercial airports. Large hubs are airports that account for at least 1 percent of the total passenger enplanements in the United States. Airports that account for 0.25 percent to 1 percent of total passenger enplanements are considered medium hubs. Pull passenger traffic data from the Wikipedia page given below using web scraping and collate it in a table.

https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States
(https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States)

```
In [33]: # Now Lets use the web scrapping to import the data frome the wikipedia.
url2 = "https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States"
table = pd.read_html(url2)
```

In [34]: table

```
Out[34]: [   Rank(2021)      Airports (large hubs) IATACode \
0           1  Hartsfield-Jackson Atlanta International Airport  ATL
1           2           Los Angeles International Airport      LAX
2           3      Chicago O'Hare International Airport      ORD
3           4  Dallas/Fort Worth International Airport      DFW
4           5           Denver International Airport      DEN
5           6  John F. Kennedy International Airport      JFK
6           7      San Francisco International Airport      SFO
7           8  Seattle-Tacoma International Airport      SEA
8           9           Orlando International Airport      MCO
9          10      Harry Reid International Airport      LAS
10         11  Charlotte-Douglas International Airport      CLT
11         12      Newark Liberty International Airport      EWR
12         13  Phoenix Sky Harbor International Airport      PHX
13         14      George Bush Intercontinental Airport      IAH
14         15           Miami International Airport      MIA
15         16      Boston Logan International Airport      BOS
16         17  Minneapolis-Saint Paul International Airport      MSP
17         18      Detroit Metropolitan Airport      DTW
```

In [35]: table[0] = table[0].drop(['2021', '2013[10]', '2012[11]', '2011[12]'], axis=1)

In [36]: table[0].head()

```
Out[36]:
```

	Rank(2021)	Airports (large hubs)	IATACode	Major cities served	State	2020[3]	2019[4]	2018[5]	2017[6]	2016[7]	2015[8]
0	1	Hartsfield-Jackson Atlanta International Airport	ATL	Atlanta	GA	20559866	53505795	51865797	50251964	50501858	493407
1	2	Los Angeles International Airport	LAX	Los Angeles	CA	18593421	35778573	32821799	31816933	31283579	315898
2	3	Chicago O'Hare International Airport	ORD	Chicago	IL	16243216	33592945	31362941	29809097	28267394	262800
3	4	Dallas/Fort Worth International Airport	DFW	Dallas/Fort Worth	TX	14606034	40871223	39873927	38593028	37589899	363056
4	5	Denver International Airport	DEN	Denver	CO	14055777	42939104	42624050	41232432	39636042	363512

In [37]: table[0]['traffic_Chg19_20'] = table[0]['2020[3]'] - table[0]['2019[4]']

In [38]: table[0]['traffic_Chg18_19'] = table[0]['2019[4]'] - table[0]['2018[5]']
table[0]['hubs'] = str('large_hub')

```
In [39]: table[0] = table[0][['IATACode', 'traffic_Chg19_20', 'traffic_Chg18_19', 'hubs']]
table[0]
```

```
Out[39]:
```

	IATACode	traffic_Chg19_20	traffic_Chg18_19	hubs
0	ATL	-32945929	1639998	large_hub
1	LAX	-17185152	2956774	large_hub
2	ORD	-17349729	2230004	large_hub
3	DFW	-26265189	997296	large_hub
4	DEN	-28883327	315054	large_hub
5	JFK	-11246819	1917739	large_hub
6	SFO	-14094543	1359791	large_hub
7	SEA	-14144302	933349	large_hub
8	MCO	-11902116	810972	large_hub
9	LAS	-12635024	399391	large_hub
10	CLT	-15539351	976854	large_hub
11	EWR	-13222751	747911	large_hub
12	PHX	-22766836	415886	large_hub
13	IAH	-15175289	363161	large_hub
14	MIA	-9935245	338658	large_hub
15	BOS	-12123197	830975	large_hub
16	MSP	-20034173	-11487	large_hub
17	DTW	-11320716	706203	large_hub
18	FLL	-14663925	692856	large_hub
19	PHL	-7087602	614111	large_hub
20	LGA	-10253150	713719	large_hub
21	BWI	-7833332	-87129	large_hub
22	SLC	-6011981	610242	large_hub
23	SAN	-8010836	474468	large_hub
24	IAD	-11246485	335100	large_hub
25	DCA	-5845178	-596237	large_hub
26	TPA	-4921659	918307	large_hub
27	MDW	-8021459	262494	large_hub

```
In [40]: table[1].head()
```

Out[40]:

	Rank(2020)	Airports (medium hubs)	IATACode	City served	State	2020	2019	2018	2017	2016	2015
0	29	Daniel K. Inouye International Airport	HNL	Honolulu	HI	9893930	8408457	8134848.0	7876769.0	7554596.0	7040
1	30	Portland International Airport	PDX	Portland	OR	9790489	11595454	11367176.0	11506310.0	11470854.0	11242
2	31	Nashville International Airport	BNA	Nashville	TN	8498877	9797408	9940866.0	9435473.0	9071154.0	8340
3	32	Austin–Bergstrom International Airport	AUS	Austin	TX	3141505	8683711	7921797.0	6973115.0	6095545.0	5797
4	33	Dallas Love Field	DAL	Dallas	TX	8069178	7069614	6937061.0	6741870.0	6285181.0	5937

```
In [41]: table[1]['traffic_Chg19_20'] = table[1]['2020'] - table[1]['2019']
table[1]['traffic_Chg18_19'] = table[1]['2019'] - table[1]['2018']
table[1]['hubs'] = str('Medium_hub')
```

```
In [42]: table[1] = table[1][['IATACode', 'traffic_Chg19_20', 'traffic_Chg18_19', 'hubs']]
table[1]
```

```
Out[42]:
```

	IATACode	traffic_Chg19_20	traffic_Chg18_19	hubs
0	HNL	1485473	273609.0	Medium_hub
1	PDX	-1804965	228278.0	Medium_hub
2	BNA	-1298531	-143458.0	Medium_hub
3	AUS	-5542206	761914.0	Medium_hub
4	DAL	999564	132553.0	Medium_hub
5	STL	-2238287	410173.0	Medium_hub
6	SJC	-270221	124712.0	Medium_hub
7	HOU	1912672	424899.0	Medium_hub
8	RDU	452929	422783.0	Medium_hub
9	MSY	141501	151623.0	Medium_hub
10	OAK	1950734	556705.0	Medium_hub
11	SMF	-470933	502607.0	Medium_hub
12	MCI	-2080699	688269.0	Medium_hub
13	SNA	-1409936	-238091.0	Medium_hub
14	RSW	-719803	-175712.0	Medium_hub
15	SAT	126615	57961.0	Medium_hub
16	CLE	174943	14143.0	Medium_hub
17	IND	-338022	178553.0	Medium_hub
18	PIT	-477440	-163873.0	Medium_hub
19	SJU	-172541	45914.0	Medium_hub
20	CVG	-17062	144199.0	Medium_hub
21	CMH	-16471	117495.0	Medium_hub
22	OGG	316303	197387.0	Medium_hub
23	JAX	-7422	361383.0	Medium_hub
24	PBI	79312	-174744.0	Medium_hub
25	MKE	635944	223831.0	Medium_hub
26	BDL	604458	70942.0	Medium_hub
27	BUR	-338581	-7120.0	Medium_hub
28	ONT	-1074666	219674.0	Medium_hub
29	ANC	334018	NaN	Medium_hub
30	ABQ	182971	-1813.0	Medium_hub
31	OMA	132539	NaN	Medium_hub
32	BUF	390491	NaN	Medium_hub
33	CHS	284	NaN	Medium_hub
34	MEM	-67924	NaN	Medium_hub
35	RIC	44986	NaN	Medium_hub

```
In [43]: # Lets first merge all wikipedia table.
wiki_data = [table[0], table[1]]
```

```
In [44]: wiki_data = pd.concat(wiki_data, ignore_index=True)
```

In [45]: wiki_data

Out[45]:

	IATACode	traffic_Chg19_20	traffic_Chg18_19	hubs
0	ATL	-32945929	1639998.0	large_hub
1	LAX	-17185152	2956774.0	large_hub
2	ORD	-17349729	2230004.0	large_hub
3	DFW	-26265189	997296.0	large_hub
4	DEN	-28883327	315054.0	large_hub
...
59	OMA	132539	NaN	Medium_hub
60	BUF	390491	NaN	Medium_hub
61	CHS	284	NaN	Medium_hub
62	MEM	-67924	NaN	Medium_hub
63	RIC	44986	NaN	Medium_hub

64 rows × 4 columns

In [46]: *# Now we gather all the information that we got from wiki pedia link and the data that we have.*
 final_df = df.merge(wiki_data, left_on='iata_code', right_on='IATACode')

In [47]: final_df

Out[47]:

	id	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay	ident	...	width_ft	surfac
0	4	AA	2466	SFO	DFW	3	20	195	1	KSFO	...	200.0	ASI
1	231	AA	526	SFO	DFW	3	360	215	0	KSFO	...	200.0	ASI
2	234	AA	552	SFO	MIA	3	360	315	1	KSFO	...	200.0	ASI
3	905	AA	810	SFO	ORD	3	385	255	0	KSFO	...	200.0	ASI
4	1739	AA	24	SFO	JFK	3	425	325	1	KSFO	...	200.0	ASI
...
363125	506267	9E	4052	DAL	MEM	4	370	90	0	KDAL	...	150.0	COI
363126	512858	9E	3704	DAL	MEM	4	705	92	1	KDAL	...	150.0	COI
363127	518247	9E	4060	DAL	MEM	4	990	90	0	KDAL	...	150.0	COI
363128	524678	9E	4052	DAL	MEM	5	370	90	1	KDAL	...	150.0	COI
363129	530841	9E	3704	DAL	MEM	5	705	92	0	KDAL	...	150.0	COI

363130 rows × 30 columns

2. You should then examine the missing values in each field, perform missing value treatment, and justify your actions.

In [48]: *# Now we have the final data first we remove some column that is not useable.*
 final_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 363130 entries, 0 to 363129
Data columns (total 30 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                     363130 non-null  int64
1   Airline                363130 non-null  object
2   Flight                 363130 non-null  int64
3   AirportFrom            363130 non-null  object
4   AirportTo              363130 non-null  object
5   DayOfWeek              363130 non-null  int64
6   Time                   363130 non-null  int64
7   Length                 363130 non-null  int64
8   Delay                  363130 non-null  int64
9   ident                  363130 non-null  object
10  type                    363130 non-null  object
11  name                    363130 non-null  object
12  latitude_deg            363130 non-null  float64
13  longitude_deg           363130 non-null  float64
14  elevation_ft            363130 non-null  float64
15  scheduled_service       363130 non-null  object
16  iata_code               363130 non-null  object
17  airport_ref             363130 non-null  int64
18  airport_ident           363130 non-null  object
19  length_ft               363130 non-null  float64
20  width_ft                363130 non-null  float64
21  surface                 363130 non-null  object
22  lighted                 363130 non-null  int64
23  closed                  363130 non-null  int64
24  IATA                    363130 non-null  object
25  Founded                 363130 non-null  float64
26  IATACode                363130 non-null  object
27  traffic_Chg19_20        363130 non-null  int64
28  traffic_Chg18_19        351555 non-null  float64
29  hubs                    363130 non-null  object
dtypes: float64(7), int64(10), object(13)
memory usage: 85.9+ MB
```

In [49]: `final_df = final_df.drop(['id', 'AirportFrom', 'airport_ident', 'iata_code', 'AirportTo', 'surface', 'IATA', 'IATACode', 'name'], axis=1)`

In [50]: `final_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 363130 entries, 0 to 363129
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Airline                363130 non-null object
 1   Flight                363130 non-null int64
 2   DayOfWeek              363130 non-null int64
 3   Time                  363130 non-null int64
 4   Length                363130 non-null int64
 5   Delay                 363130 non-null int64
 6   type                  363130 non-null object
 7   latitude_deg          363130 non-null float64
 8   longitude_deg         363130 non-null float64
 9   elevation_ft          363130 non-null float64
10   scheduled_service     363130 non-null object
11   airport_ref           363130 non-null int64
12   length_ft             363130 non-null float64
13   width_ft              363130 non-null float64
14   lighted               363130 non-null int64
15   closed                363130 non-null int64
16   Founded               363130 non-null float64
17   traffic_Chg19_20      363130 non-null int64
18   traffic_Chg18_19      351555 non-null float64
19   hubs                  363130 non-null object
dtypes: float64(7), int64(9), object(4)
memory usage: 58.2+ MB
```

In [51]: `# Now Lets check the null value and treat them.`
`final_df.isnull().sum()`

```
Out[51]: Airline                0
Flight                0
DayOfWeek             0
Time                  0
Length                0
Delay                 0
type                  0
latitude_deg          0
longitude_deg         0
elevation_ft          0
scheduled_service     0
airport_ref           0
length_ft             0
width_ft              0
lighted               0
closed                0
Founded               0
traffic_Chg19_20      0
traffic_Chg18_19      11575
hubs                  0
dtype: int64
```

Only one column contain the null value so simply we will drop that rows of null value because we have plenty of data.

In [52]: `final_df = final_df.dropna(axis=0)`


```
In [53]: final_df.head()
```

```
Out[53]:
```

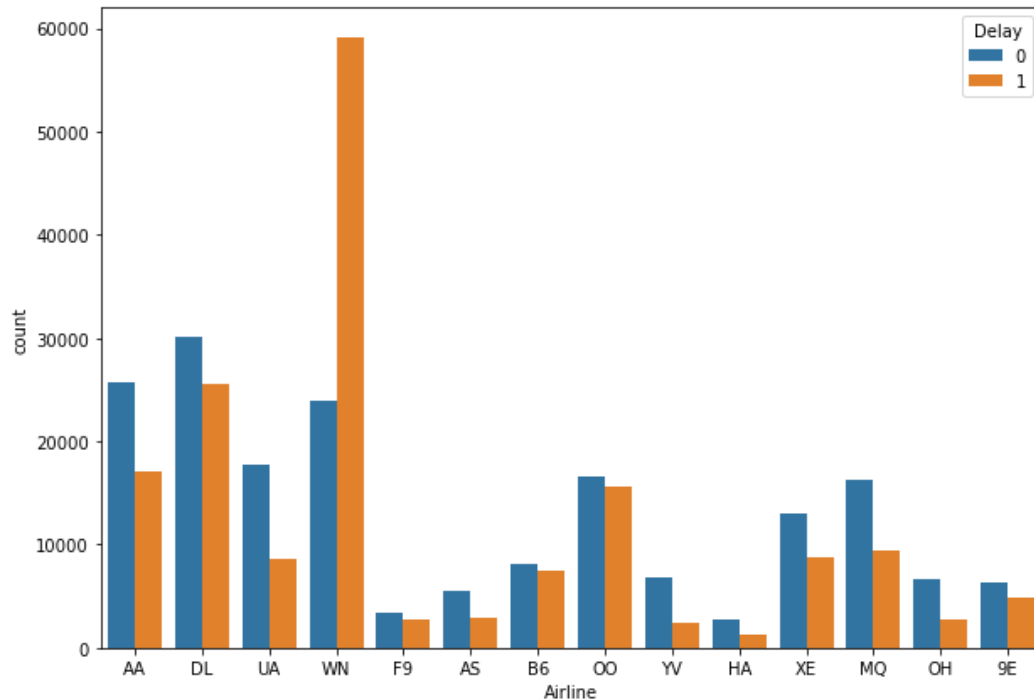
	Airline	Flight	DayOfWeek	Time	Length	Delay	type	latitude_deg	longitude_deg	elevation_ft	scheduled
0	AA	2466	3	20	195	1	large_airport	37.618999	-122.375	13.0	
1	AA	526	3	360	215	0	large_airport	37.618999	-122.375	13.0	
2	AA	552	3	360	315	1	large_airport	37.618999	-122.375	13.0	
3	AA	810	3	385	255	0	large_airport	37.618999	-122.375	13.0	
4	AA	24	3	425	325	1	large_airport	37.618999	-122.375	13.0	

3. Perform data visualization and share your insights on the following points:

a. According to the data provided, approximately 70% of Southwest Airlines flights are delayed. Visualize it to compare it with the data of other airlines.

Airline code WN represent the southwest airlines.

```
In [133]: plt.figure(figsize=(10,7))
sns.countplot(final_df['Airline'], hue= final_df['Delay'])
plt.show()
```



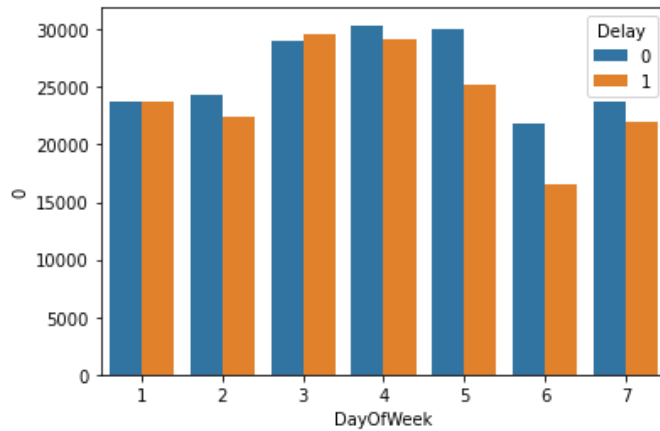
The graph clear show that 70% of flight of south west airline is delayed

b. Flights were delayed on various weekdays. Which day of the week is the safest for travel?

```
In [115]: weekday_df = final_df[['DayOfWeek', 'Delay']].value_counts().reset_index()
```

```
In [117]: sns.barplot(weekday_df['DayOfWeek'], weekday_df[0], hue= weekday_df['Delay'])
```

```
Out[117]: <AxesSubplot:xlabel='DayOfWeek', ylabel='0'>
```

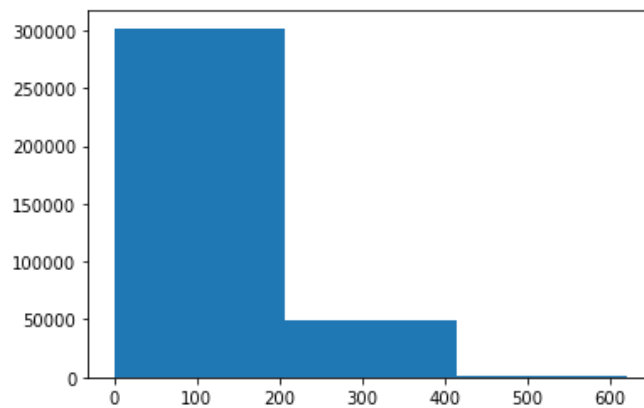


On the 5th day of week its clear that there is less no of flight delay.

c. Which airlines should be recommended for short-, medium-, and long-distance travel?

We divided the length parameter in three range and from that basis we findout airline acc to the distance

```
In [121]: plt.hist(final_df['Length'], bins = 3)  
plt.show()
```



airlines should be recommended for short distance Travel.

```
In [123]: final_df['Airline'][final_df['Length']<200].value_counts()
```

```
Out[123]: WN      73809
DL      42200
OO      31468
AA      29948
MQ      25466
XE      21341
UA      16157
B6      11628
9E      11192
YV       9280
OH       9192
AS       5731
F9       5406
HA       3034
Name: Airline, dtype: int64
```

```
In [128]: final_df['Airline'][final_df['Length']>400].value_counts()
```

```
Out[128]: UA       549
AA       304
DL       226
B6        83
AS        31
HA        14
Name: Airline, dtype: int64
```

Airlines should be recommended for long distance Travel and remaining for the medium distance.

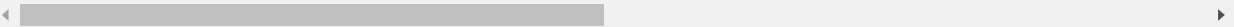
d. Do you notice any patterns in the departure times of long-duration flights?

```
In [129]: final_df['Time'][final_df['Length']>400]
```

Out[129]:

	Airline	Flight	DayOfWeek	Time	Length	Delay	type	latitude_deg	longitude_deg	elevation_ft	sc
46345	HA	5	4	1045	405	1	large_airport	36.083361	-115.151817	2181.0	
46348	HA	5	5	1045	405	0	large_airport	36.083361	-115.151817	2181.0	
46356	HA	5	1	1045	405	1	large_airport	36.083361	-115.151817	2181.0	
46364	HA	5	4	1045	405	1	large_airport	36.083361	-115.151817	2181.0	
46367	HA	5	5	1045	405	1	large_airport	36.083361	-115.151817	2181.0	
...	
315043	UA	92	1	1416	404	0	medium_airport	20.898543	-156.431212	54.0	
315049	UA	92	2	1416	404	0	medium_airport	20.898543	-156.431212	54.0	
315055	UA	92	3	1416	404	0	medium_airport	20.898543	-156.431212	54.0	
315061	UA	92	4	1416	404	0	medium_airport	20.898543	-156.431212	54.0	
315067	UA	92	5	1416	404	0	medium_airport	20.898543	-156.431212	54.0	

1207 rows × 20 columns

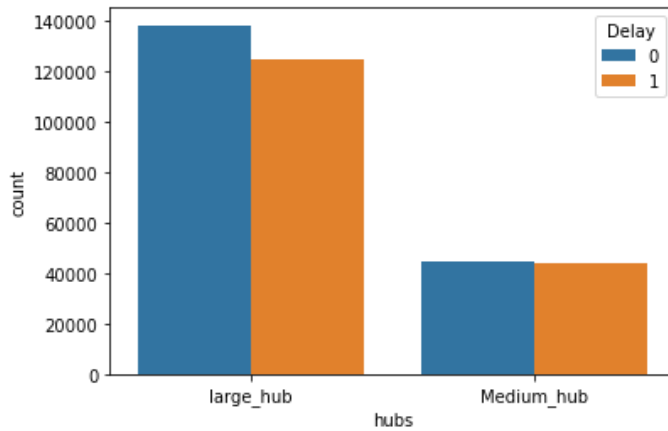


It is clear from the above table that is only of that flight which travel a long distance and comman thing in the departure time is all long distance flight leave the airport above 1045 time.

4. How many flights were delayed at large hubs compared to medium hubs? Use appropriate visualization to represent your findings.

```
In [132]: sns.countplot(final_df['hubs'], hue = final_df['Delay'])
```

```
Out[132]: <AxesSubplot:xlabel='hubs', ylabel='count'>
```



From the large hubs its clear approx 120000 flight is delayed but from the small hubs approx 40000 is delayed.

5. Use hypothesis testing strategies to discover:

a. If the airport's altitude has anything to do with flight delays for incoming and departing flights

```
In [134]: from scipy.stats import chi2_contingency
table = [final_df['latitude_deg'], final_df['Delay']]
stat, p, dof, expected = chi2_contingency(table)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
```

```
stat=186602.569, p=1.000
Probably independent
```

So its clear from the above hypothesis testing that altitude is nothing to do with the flight delay

b. If the number of runways at an airport affects flight delays

```
In [135]: from scipy.stats import chi2_contingency
table = [final_df['airport_ref'], final_df['Delay']]
stat, p, dof, expected = chi2_contingency(table)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
```

```
stat=192200.911, p=1.000
Probably independent
```

So its clear from the above hypothesis testing that no of runway is nothing to do with the flight delay

c. If the duration of a flight (length) affects flight delays

```
In [150]: from scipy.stats import spearmanr
data1 = final_df['Length']
data2 = final_df['Delay']
stat, p = spearmanr(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
```

```
stat=-0.002, p=0.179
Probably independent
```

Both the variable are independent so that length of the flight is not affecting directly the delay.

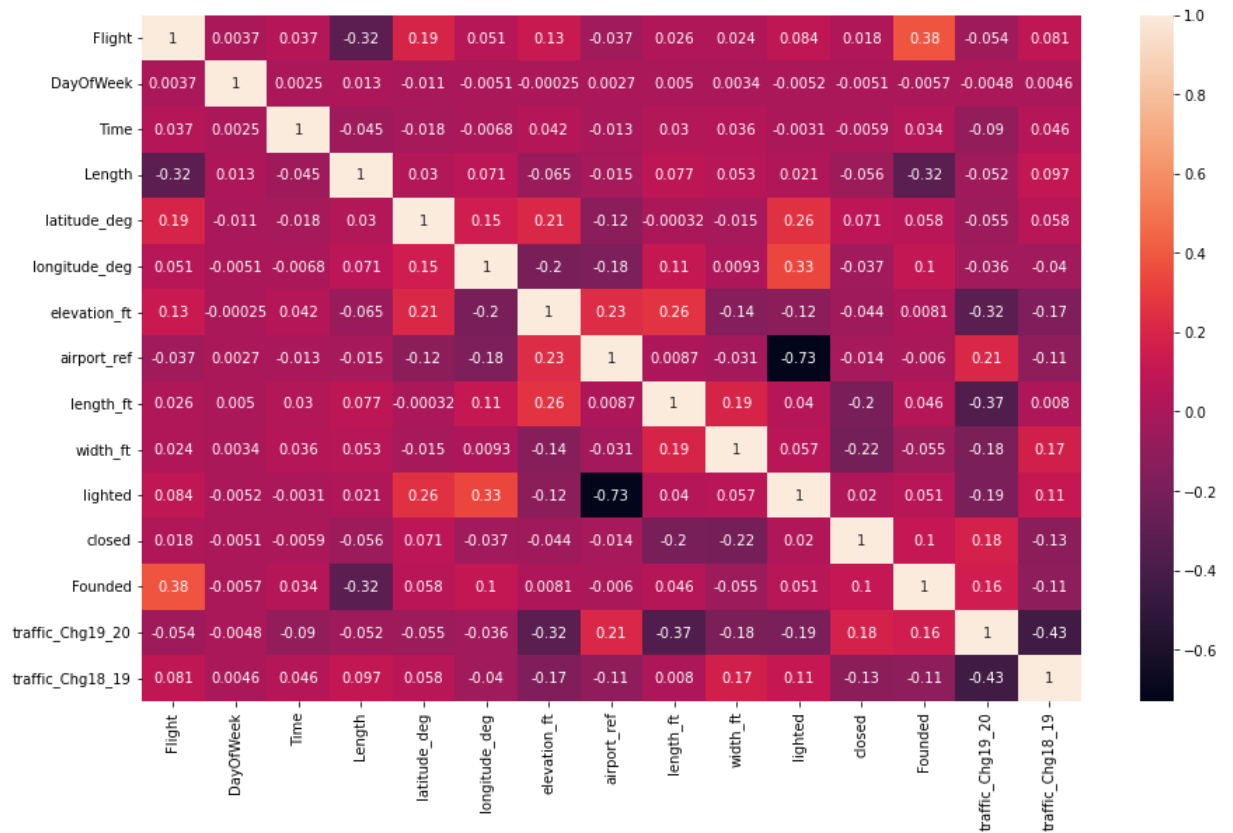
6. Find the correlation matrix between the flight delay predictors, create a heatmap to visualize this, and share your findings

```
In [148]: predictor = final_df.drop(['Delay'], axis=1)
corr = predictor.corr()
corr
```

```
Out[148]:
```

	Flight	DayOfWeek	Time	Length	latitude_deg	longitude_deg	elevation_ft	airport_ref	length_ft	width_ft	lighted	closed	Founded	traffic_Chg19_20	traffic_Chg18_19
Flight	1.000000	0.003732	0.037147	-0.315231	0.194294	0.050626	0.127833	-0.036501	0.025819	0.024280	0.084263	0.018225	0.384262	-0.054194	0.081196
DayOfWeek	0.003732	1.000000	0.002477	0.013215	-0.010733	-0.005069	-0.000254	0.002677	0.004980	0.003404	-0.005173	-0.005079	-0.005709	-0.004771	0.004565
Time	0.037147	0.002477	1.000000	-0.045410	-0.017776	-0.006839	0.041580	-0.012562	0.030107	0.036335	-0.003140	-0.005892	0.033724	-0.089522	0.046173
Length	-0.315231	0.013215	-0.045410	1.000000	0.029843	0.070918	-0.065413	-0.015262	0.077367	0.053432	0.020547	-0.055789	-0.321202	-0.052246	0.096530
latitude_deg	0.194294	-0.010733	-0.017776	0.029843	1.000000	0.149229	0.214040	-0.120146	-0.000323	-0.014539	0.255750	0.070942	0.058121	-0.054513	0.058426
longitude_deg	0.050626	-0.005069	-0.006839	0.070918	0.149229	1.000000	-0.196951	-0.181168	0.114557	0.009334	0.334031	-0.036947	0.099585	-0.036013	-0.039925
elevation_ft	0.127833	-0.000254	0.041580	-0.065413	0.214040	-0.196951	1.000000	0.232130	0.259572	-0.144024	-0.123519	-0.043553	0.008079	-0.322286	-0.172352
airport_ref	-0.036501	0.002677	-0.012562	-0.015262	-0.120146	-0.181168	0.232130	1.000000	0.008687	-0.031283	-0.730141	-0.014319	-0.006035	0.210036	-0.108319
length_ft	0.025819	0.004980	0.030107	0.077367	-0.000323	0.114557	0.259572	0.008687	1.000000	0.009334	0.334031	-0.036947	0.099585	-0.036013	-0.039925
width_ft	0.024280	0.003404	0.036335	0.053432	-0.014539	0.009334	-0.144024	-0.031283	0.009334	1.000000	0.334031	-0.036947	0.099585	-0.036013	-0.039925
lighted	0.084263	-0.005173	-0.003140	0.020547	0.255750	0.334031	-0.123519	-0.730141	0.334031	0.334031	1.000000	-0.036947	0.099585	-0.036013	-0.039925
closed	0.018225	-0.005079	-0.005892	-0.055789	0.070942	-0.036947	-0.043553	-0.014319	-0.036947	-0.036947	-0.036947	1.000000	0.099585	-0.036013	-0.039925
Founded	0.384262	-0.005709	0.033724	-0.321202	0.058121	0.099585	0.008079	-0.006035	0.099585	0.099585	0.099585	0.099585	1.000000	-0.036013	-0.039925
traffic_Chg19_20	-0.054194	-0.004771	-0.089522	-0.052246	-0.054513	-0.036013	-0.322286	0.210036	-0.036013	-0.036013	-0.036013	-0.036013	-0.036013	1.000000	-0.039925
traffic_Chg18_19	0.081196	0.004565	0.046173	0.096530	0.058426	-0.039925	-0.172352	-0.108319	-0.039925	-0.039925	-0.039925	-0.039925	-0.039925	-0.039925	1.000000

```
In [149]: plt.figure(figsize=(15,9))
sns.heatmap(corr, annot=True)
plt.show()
```



Project Task: Week 1 (Machine learning)

1. Use OneHotEncoder and OrdinalEncoder to deal with categorical variables

```
In [153]: # Before applying the one hot encoding or the label encoding first we check all feature data type
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 351555 entries, 0 to 363129
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Airline                351555 non-null object
 1   Flight                351555 non-null int64
 2   DayOfWeek              351555 non-null int64
 3   Time                  351555 non-null int64
 4   Length                351555 non-null int64
 5   Delay                 351555 non-null int64
 6   type                  351555 non-null object
 7   latitude_deg          351555 non-null float64
 8   longitude_deg         351555 non-null float64
 9   elevation_ft          351555 non-null float64
10   scheduled_service     351555 non-null object
11   airport_ref           351555 non-null int64
12   length_ft             351555 non-null float64
13   width_ft              351555 non-null float64
14   lighted               351555 non-null int64
15   closed                351555 non-null int64
16   Founded               351555 non-null float64
17   traffic_Chg19_20      351555 non-null int64
18   traffic_Chg18_19      351555 non-null float64
19   hubs                  351555 non-null object
dtypes: float64(7), int64(9), object(4)
memory usage: 56.3+ MB
```

```
In [155]: final_df['Airline'].value_counts()
```

```
Out[155]: WN      82903
DL       55724
AA       42841
OO       32315
UA       26303
MQ       25698
XE       21733
B6       15497
9E       11192
OH        9440
YV        9337
AS        8355
F9        6180
HA        4037
Name: Airline, dtype: int64
```

```
In [156]: final_df['type'].value_counts()
```

```
Out[156]: large_airport    334982
medium_airport    16573
Name: type, dtype: int64
```

```
In [157]: final_df['scheduled_service'].value_counts()
```

```
Out[157]: yes      351555
Name: scheduled_service, dtype: int64
```

```
In [158]: final_df['hubs'].value_counts()
```

```
Out[158]: large_hub      262540
Medium_hub      89015
Name: hubs, dtype: int64
```

The scheduled_service column through has same value so it will not help in prediction so lets remove it and other three object column we will change through label encoder.

```
In [160]: final_df = final_df.drop(['scheduled_service'], axis=1)
```

```
In [163]: # Now using the ordinal encoder.
from sklearn.preprocessing import LabelEncoder
```

```
In [164]: le = LabelEncoder()
```

```
In [165]: final_df['Airline'] = le.fit_transform(final_df['Airline'])
final_df['type'] = le.fit_transform(final_df['type'])
final_df['hubs'] = le.fit_transform(final_df['hubs'])
```

```
In [166]: final_df.head()
```

```
Out[166]:
```

	Airline	Flight	DayOfWeek	Time	Length	Delay	type	latitude_deg	longitude_deg	elevation_ft	airport_ref	length
0	1	2466	3	20	195	1	0	37.618999	-122.375	13.0	3878	750
1	1	526	3	360	215	0	0	37.618999	-122.375	13.0	3878	750
2	1	552	3	360	315	1	0	37.618999	-122.375	13.0	3878	750
3	1	810	3	385	255	0	0	37.618999	-122.375	13.0	3878	750
4	1	24	3	425	325	1	0	37.618999	-122.375	13.0	3878	750

2. Perform the following model building steps:

a. Apply logistic regression (use stochastic gradient descent optimizer) and decision tree models

b. Use the stratified five-fold method to build and validate the models

Note: Make sure you use standardization effectively, ensuring no data leakage and leverage pipelines to have a cleaner code

c. Use RandomizedSearchCV for hyperparameter tuning, and use k-fold for cross validation

d. Keep a few data points (10%) for prediction purposes to evaluate how you would make the final prediction, and do not use this data for testing or validation

Note: The final prediction will be based on the voting (majority class by 5 models created using the stratified 5-fold method)

g. Compare the results of logistic regression and decision tree classifier

```
In [205]: # Lets first separate the predictors and the output Variable.
x = final_df.drop(['Delay'], axis= 1)
y = final_df["Delay"]
```

```
In [206]: from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
x = scaler.fit_transform(x)
```

```
In [207]: # First Split the data into the training and testing set before performing the further operation
from sklearn.model_selection import train_test_split
```

```
In [208]: x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.9, random_state=10)
```

LogisticRegression


```
In [209]: # Lets apply the Logistic regression with the randomsearchcv hypermeter tunning.
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

```
In [210]: from sklearn.model_selection import RandomizedSearchCV
```

```
In [211]: params = {"penalty": ["l1", "l2"],
                  'solver': ['newton-cg', 'liblinear']}

# Cross Validation
folds = 5

rscv = RandomizedSearchCV(estimator = lr,
                          param_distributions = params,
                          scoring = "accuracy",
                          verbose = 1,
                          cv= folds)

rscv.fit(x_train, y_train)
```

Fitting 3 folds for each of 4 candidates, totalling 12 fits

```
Out[211]: RandomizedSearchCV(cv=3, estimator=LogisticRegression(),
                             param_distributions={'penalty': ['l1', 'l2'],
                                                  'solver': ['newton-cg', 'liblinear']},
                             scoring='accuracy', verbose=1)
```

```
In [212]: print(rscv.best_params_)
print(rscv.best_score_)

{'solver': 'newton-cg', 'penalty': 'l2'}
0.592195292796719
```

```
In [213]: lr = LogisticRegression(penalty= 'l2', solver= 'newton-cg')
lr.fit(x_train,y_train).score(x_train,y_train)
```

```
Out[213]: 0.5923280414919136
```

```
In [214]: lr.score(x_test, y_test)
```

```
Out[214]: 0.593013994766185
```

DecisionTreeClassifier

```
In [215]: from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

params = {'criterion': ["gini", "entropy"],
          'min_samples_leaf' : [2,3,4,5,6,7,8,9],
          "max_depth": [2,3,4,5,6,7,8,9]}

rscv = RandomizedSearchCV(estimator = dt,
                          param_distributions= params,
                          scoring = "accuracy",
                          cv= 5,
                          verbose=1)

rscv.fit(x_train, y_train)
```

Fitting 5 folds for each of 10 candidates, totalling 50 fits

```
Out[215]: RandomizedSearchCV(cv=5, estimator=DecisionTreeClassifier(),
                             param_distributions={'criterion': ['gini', 'entropy'],
                                                  'max_depth': [2, 3, 4, 5, 6, 7, 8, 9],
                                                  'min_samples_leaf': [2, 3, 4, 5, 6, 7, 8, 9]},
                             scoring='accuracy', verbose=1)
```

```
In [216]: print(rscv.best_params_)
          print(rscv.best_score_)
```

```
{'min_samples_leaf': 6, 'max_depth': 9, 'criterion': 'entropy'}
0.6469110137916109
```

```
In [220]: dtc = DecisionTreeClassifier(max_depth= 9, criterion= 'entropy',min_samples_leaf= 6)
```

```
In [221]: dtc.fit(x_train, y_train).score(x_train, y_train)
```

```
Out[221]: 0.6539464410443775
```

```
In [222]: dtc.score(x_test, y_test)
```

```
Out[222]: 0.649049948799636
```

After seeing the result its clear decision tree has good accuracy.

3. Use the stratified five-fold method to build and validate the models using the XGB classifier, compare all methods, and share your findings

```
In [224]: from xgboost import XGBClassifier
```

```
# Create the parameter grid: gbm_param_grid
```

```
gbm_param_grid = {
    'n_estimators': range(8, 20),
    'max_depth': range(6, 10),
    'learning_rate': [.4, .45, .5, .55, .6],
    'colsample_bytree': [.6, .7, .8, .9, 1]
}
```

```
# Instantiate the regressor: gbm
```

```
gbm = XGBClassifier()
```

```
# Perform random search: grid_mse
```

```
xgb_random = RandomizedSearchCV(param_distributions=gbm_param_grid,
                                estimator = gbm, scoring = "accuracy",
                                verbose = 1, n_iter = 50, cv = 3)
```

```
# Fit randomized_mse to the data
```

```
xgb_random.fit(x_train, y_train)
```

```
# Print the best parameters and Lowest RMSE
```

```
print("Best parameters found: ", xgb_random.best_params_)
```

```
print("Best accuracy found: ", xgb_random.best_score_)
```

Fitting 3 folds for each of 50 candidates, totalling 150 fits

```
Best parameters found: {'n_estimators': 14, 'max_depth': 9, 'learning_rate': 0.45, 'colsample_bytree': 0.9}
```

```
Best accuracy found: 0.6612157449541393
```

```
In [225]: xgb = XGBClassifier(n_estimators=14, max_depth=9, learning_rate=0.45, colsample_bytree=0.9)
          xgb.fit(x_train,y_train).score(x_train,y_train)
```

```
Out[225]: 0.6860830786443699
```

```
In [226]: # Now Lets compare the all method.
```

```
print(lr.score(x_test, y_test))
```

```
print(dtc.score(x_test, y_test))
```

```
print(xgb.score(x_test, y_test))
```

```
0.593013994766185
```

```
0.649049948799636
```

```
0.6630447149846399
```

After comparing the accuracy of the diffrent model the best result we getting from the XGBclassifier.

In []:

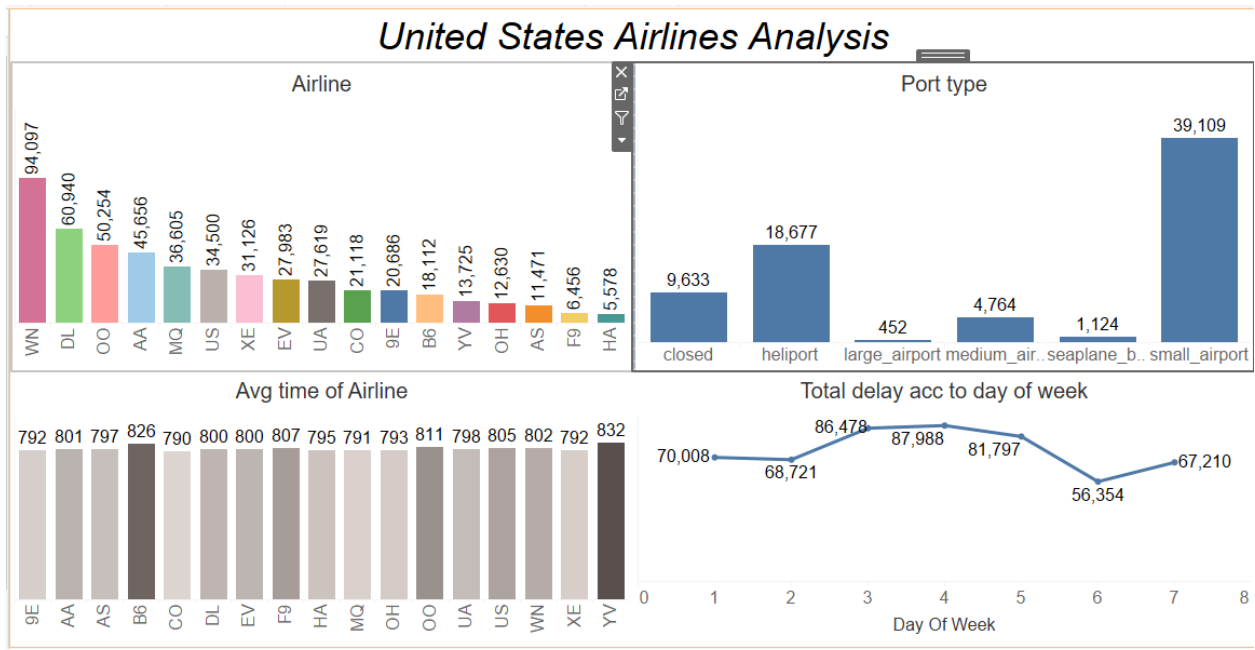
In []:

Project Task: Week 2

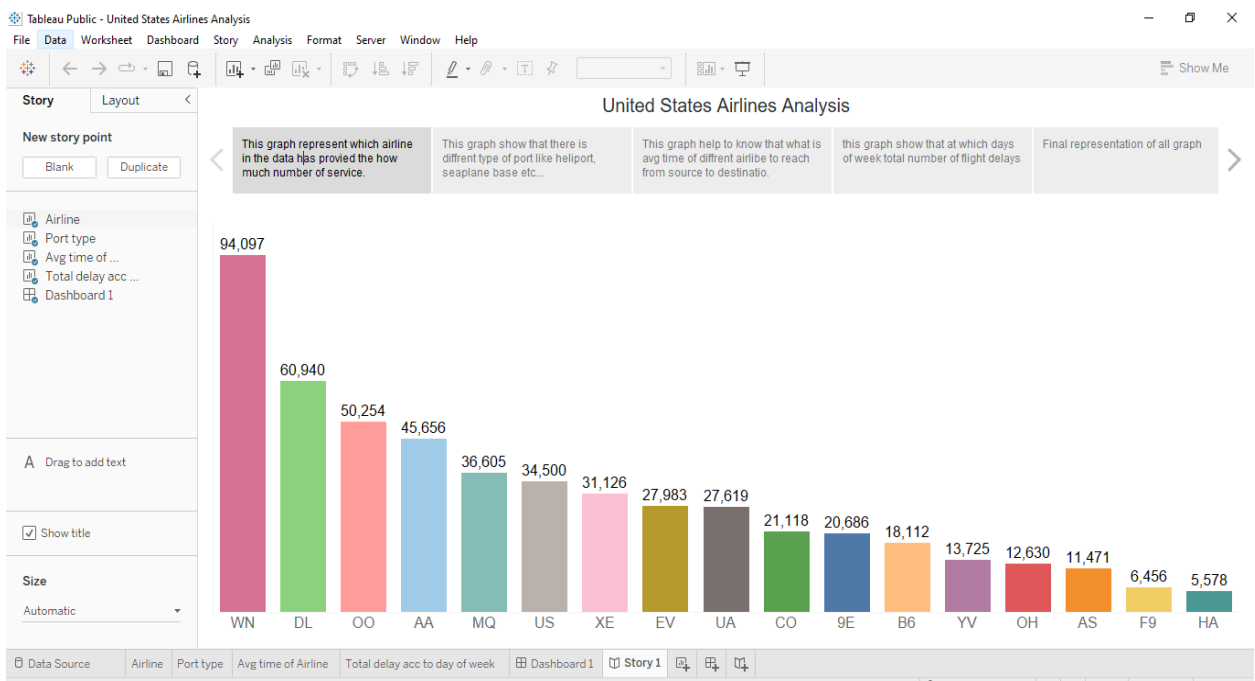
1. Create a dashboard in Tableau by selecting appropriate chart types and metrics for the business

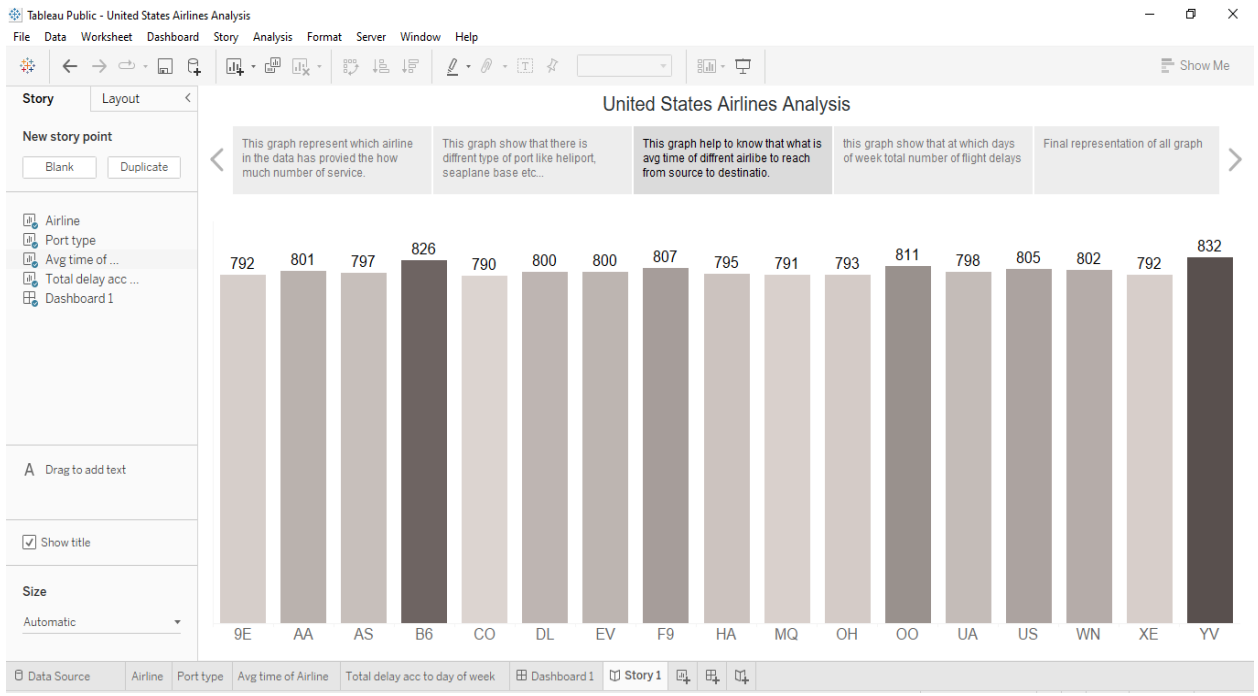
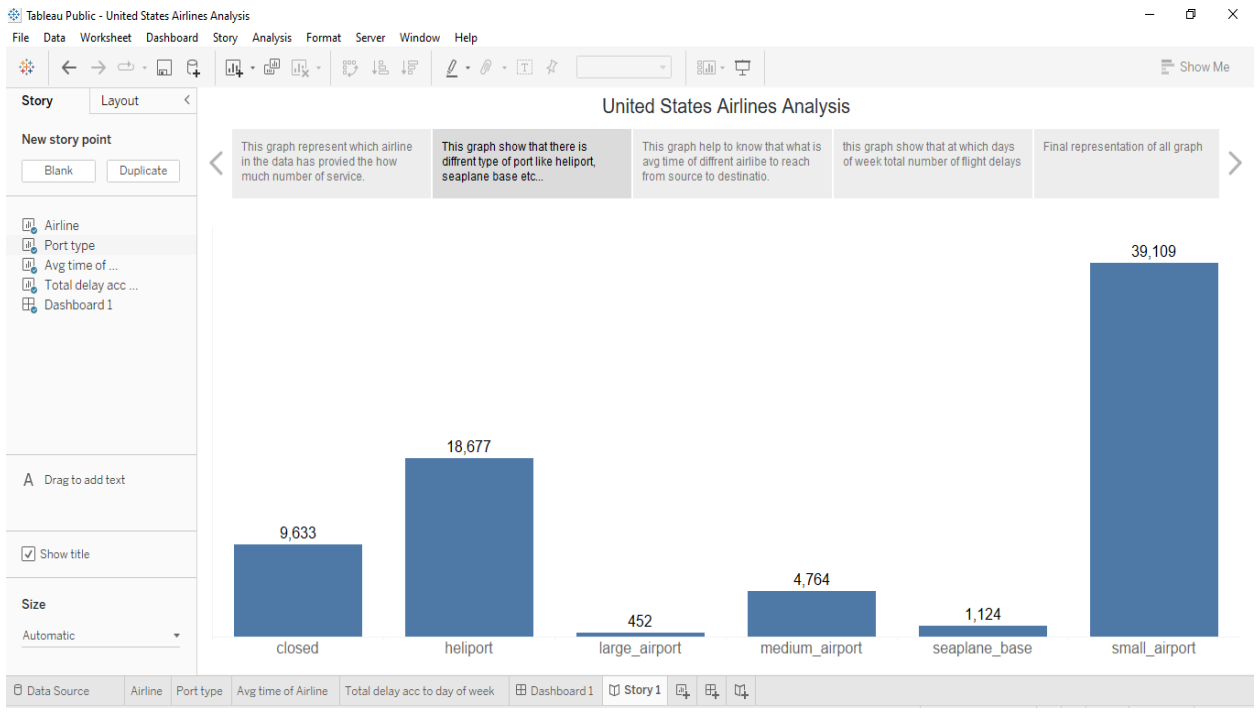
Note: Put more emphasis on data storytelling

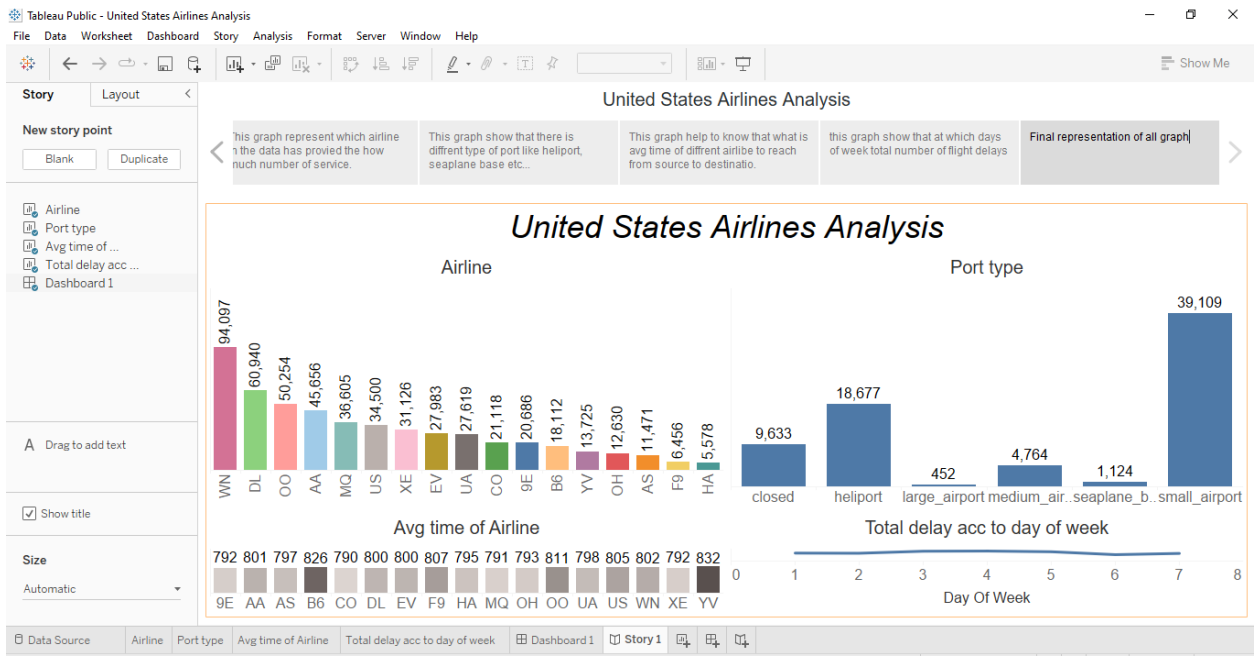
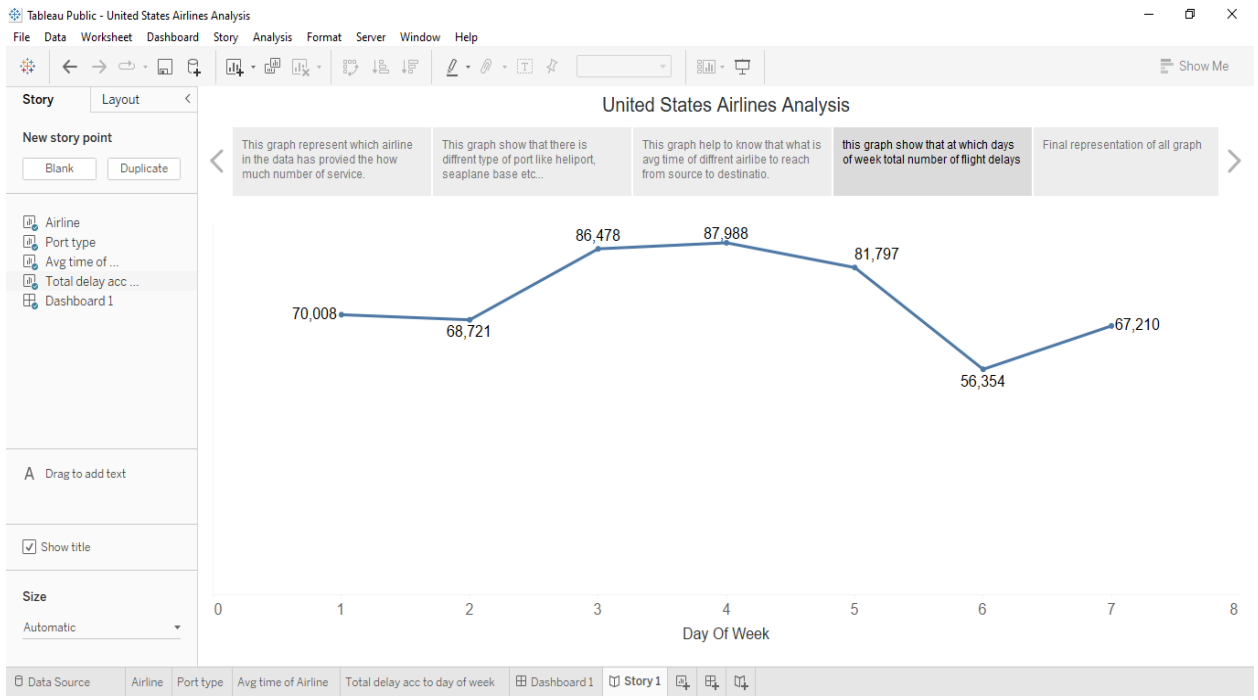
Dashboard



Storytelling

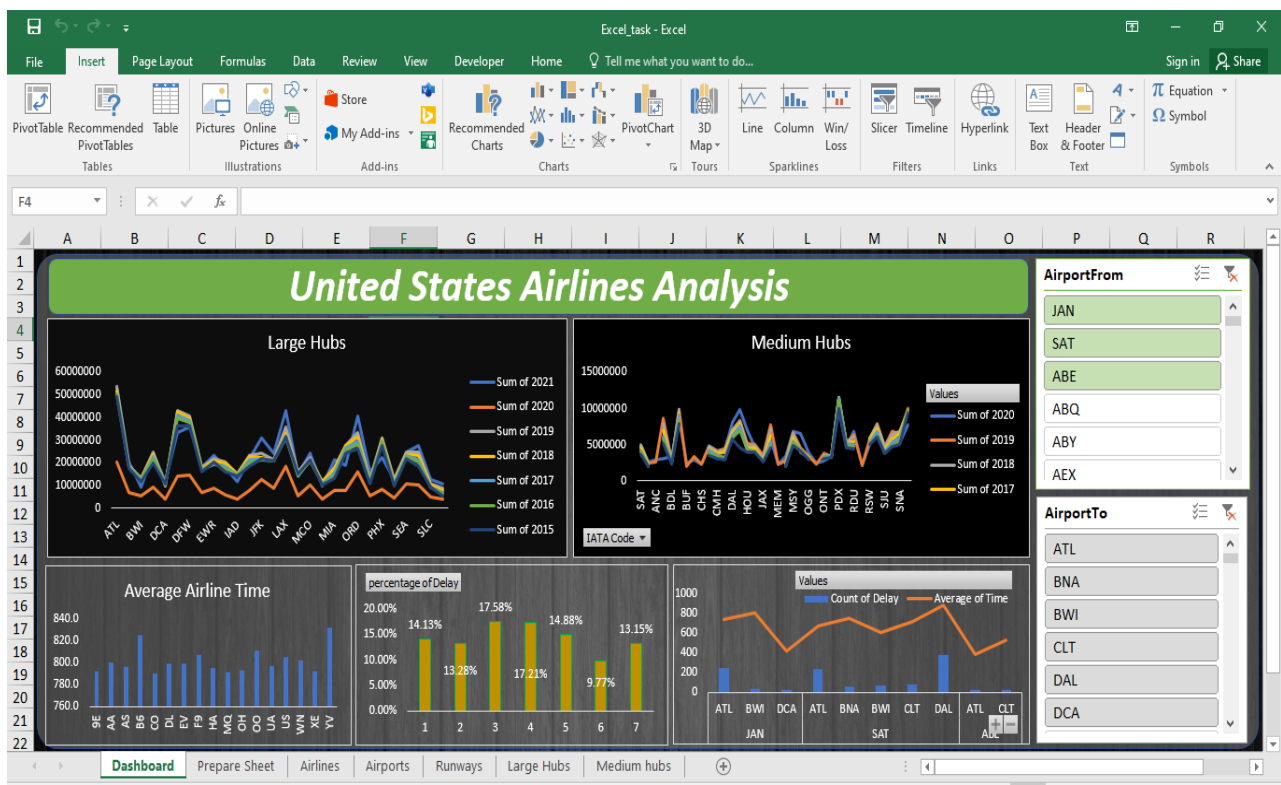






Excel

1. Create an Excel dashboard showcasing the following (use form controls to make a dynamic chart):
 - a. Compare different airlines based on their on-time performance
 - b. Compare the percentage of delayed flights for different days of the week
 - c. Create a trend chart for the number of passenger's at large and medium hubs
 - d. Visualize the count of delayed and on-time flights for different pairs of source and destination airports
- Create a dynamic chart that allows users to select a source and destination airport.



```

/* Question No1:- Determine the number of flights that are delayed on various days of the week
*/

-- First calling the database to import the data.
use job_readiness;
-- Import the data set
to perform further operation.
select * from airline;
select * from airports;
select * from
runways;

select DayOfWeek, count(Flight), Delay from airline where Delay=1 group by
DayOfWeek;

/* Question No2:- Determine the number of delayed flights for various airlines
*/

select Airline, count(Flight) from airline where Delay=1 group by Airline;

/* Question
No3:- Determine how many delayed flights land at airports with at least 10 runways */

select
AirportTo, Flight, Delay from airline where Delay=1 group by AirportTo;

/* Question No4:-
Compare the number of delayed flights at airports higher than average elevation and
those that
are lower than average elevation for both source and destination airports */

-- Lets first
compare for the source airport
select l.AirportFrom, count(l.Flight), avg(p.elevation_ft) as
avg_elevation, p.elevation_ft
from airline as l
inner join airports as p
on p.iata_code =
l.AirportFrom
where p.elevation_ft >1037.25 and l.Delay=1
group by l.AirportFrom;

select
l.AirportFrom, count(l.Flight), avg(p.elevation_ft) as avg_elevation, p.elevation_ft
from
airline as l
inner join airports as p
on p.iata_code = l.AirportFrom
where p.elevation_ft<
1037.25 and l.Delay=1
group by l.AirportFrom;

-- Lets now compare for the destination
airport
select l.AirportTo, count(l.Flight), avg(p.elevation_ft) as avg_elevation,
p.elevation_ft
from airline as l
inner join airports as p
on p.iata_code = l.AirportFrom
where
p.elevation_ft >1037.25 and l.Delay=1
group by l.AirportTo;

select l.AirportTo,
count(l.Flight), avg(p.elevation_ft) as avg_elevation, p.elevation_ft
from airline as l
inner
join airports as p
on p.iata_code = l.AirportFrom
where p.elevation_ft <1037.25 and
l.Delay=1
group by l.AirportTo;

```