Box Folder: https://uofi.box.com/s/kian2fcexf0qg6ngxgsae1areixgulue

IS531 – Theory and Practice of Data Cleaning

Final Project Report

By

Gnanendra Reddy Tugu Yagama Reddy

Samuel John

1. Introduction and Overview

The aim of this project is to take a 'dirty' dataset that would otherwise not be useful in the data analysis process and convert it into an actionable dataset that can be used to derive insightful information. We decided that this insightful information in our case would be meaningful visualizations that would help us identify key properties of objects in the dataset. For this purpose, we looked at several datasets online and zeroed down on the *Flights* dataset that was available on Kaggle (https://kaggle.com/mmetter/flights).

The flights dataset on Kaggle contains information about flights between various airports in North America, for the months of January and May. Our aim was to take the data in this dataset and create visualizations that would answer questions regarding the delays and distances corresponding to various flights in the dataset. These results would be real results and not hypothesized and could help in various fields such as helping customers select the right flight based on flight delay history or in helping Airlines identify which of their planes need better servicing or resources based on flight distance or frequency of flights.

Each of the columns of the dataset would have to be converted into formats that would help answer these questions. For this purpose, while looking at the dataset, we looked at it from an analytical point of view. We then selected value ranges that would specifically suit the use cases that we identified early on in the project.

2. Initial Assessment of the Data Set

The dataset is a public dataset available on Kaggle. The dataset is downloadable as a .txt file and did not have any metadata on the website itself. There was no description of the dataset or a mention of the various column headings and their meanings, or the lengths of the various rows and columns, as seen in Fig 2.1 below. On inspection, it was noted that the data file was a raw text file delimited using the pipe symbol "|". The initial line contained the names of the columns and the subsequent line in the file contained the values for each row.

Data (300 MB)

Data Sources

flights.txt

About this file

No description yet

flights.txt (300.01 MB)

```
TRANSACTIONID|FLIGHTDATE|AIRLINECODE|AIRLINENAME|TAILNUM|FLIGHTNUM|ORIGINAIRPORTCODE|ORIGAIRPORTNAME|ORIGINCITYNAME|OR
54548800|20020101|WN|Southwest Airlines Co.: WN|N103@@|1425|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
55872300|20020101|CO|Continental Air Lines Inc.: CO|N83872|150|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Al
54388800|20020101|WN|Southwest Airlines Co.: WN|N334@@|249|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuqu
54486500|20020101|WN|Southwest Airlines Co.: WN|N699@@|902|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuqu
55878700|20020103|CO|Continental Air Lines Inc.: CO|N58606|234|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Al
54380600|20020103|WN|Southwest Airlines Co.: WN|N501@@|193|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuqu
55117600|20020104|DL|Delta Air Lines Inc.: DL|N37574|262|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuquer
55232700|20020105|DL|Delta Air Lines Inc.: DL|N517D1|1214|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuque
54624700|20020105|WN|Southwest Airlines Co.: WN|N679@@|2122|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
54613800|20020105|WN|Southwest Airlines Co.: WN|N791@@|2038|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
54583300|20020105|WN|Southwest Airlines Co.: WN|N662@@|1737|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
54428000|20020106|WN|Southwest Airlines Co.: WN|N330@@|502|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuqu
54527000|20020106|WN|Southwest Airlines Co.: WN|N328@@|1258|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
54401600|20020107|WN|Southwest Airlines Co.: WN|N507@@|334|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuqu
54642900|20020108|WN|Southwest Airlines Co.: WN|N773@@|2312|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
54660500|20020108|WN|Southwest Airlines Co.: WN|N602@@|2734|ABQ|AlbuquerqueNM: Albuquerque International Sunport|Albuq
55964000|20020108|CO|Continental Air Lines Inc.: CO|N17326|1769|ABQ|AlbuquerqueNM: Albuquerque International Sunport|A
55404900|20020108|HP|America West Airlines Inc.: HP (Merged with US Airways 9/05.Stopped reporting 10/07.)|N316A|763|A
```

Fig 2.1 – Dataset as seen on Kaggle

For further understanding of the data we loaded the data into python and converted the file into a pandas dataframe using the pipe symbol as a delimiter. On inspection of the dataframe, we realized that the dataset contained flight specific data in each column.

The dataset contained 31 columns and 1048576 rows.

	TRANSACTIONID	FLIGHTDATE	AIRLINECODE	AIRLINENAME	TAILNUM	FLIGHTNUM	ORIGINAIRPORTCODE	ORIGAIRPORTNAME	ORIGINCITYNAME	ORIG
0	54548800	20020101	WN	Southwest Airlines Co.: WN	N103@@	1425	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerque	
1	55872300	20020101	CO	Continental Air Lines Inc.: CO	N83872	150	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerque	
2	54388800	20020101	WN	Southwest Airlines Co.: WN	N334@@	249	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerque	
3	54486500	20020101	WN	Southwest Airlines Co.: WN	N699@@	902	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerque	
4	55878700	20020103	CO	Continental Air Lines Inc.: CO	N58606	234	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerque	

Fig 2.2 – Dataset converted to Pandas DataFrame

2.1 Description of the data

The data can be categorized into three sections for better understanding:

Flight Information:

The first column Transaction ID is an identifier for each flight record and contains an 8-digit number which is unique for each record. The second column is the flight date which contains the date of the journey. The next two columns contain information about the Airline (Airline Code and Name). The fifth column contains the tail number for each aircraft. This field contains alphanumeric characters and the field Flight number is a numeric field.

Airport Information:

This section contains all the details about the origin and destination airports for each flight trip. This includes the airport code, airport name, City and State for both origin and destination of the flight trip. All these fields are text with some special characters.

Trip Information:

This section contains information about the CRS arrival and departure times as well as the actual arrival and departure times. This section also contains Taxi, Actual delay, CRS and Actual elapsed time as well as the total distance travelled. We can also find the final status of the flight and whether the flight is diverted from its planned destination or not.

Upon further investigation, we found the below mentioned issues with the dataset.

- FLIGHTDATE is as a blocked number. Should be as a date object
- AIRLINENAME has the airline code concatenated with the airline name
- TAILNUM has the '@' symbol in some of the rows.
- ORIGINAIRPORTNAME and DESAIRPORTNAME have both the state and city concatenated with the airport name
- CRSDEPTIME, DEPTIME, WHEELSOFF, WHEELSON, CRSARRTIME, and ARRTIME all are times as integer format. This is in military time
- CANCELLED and DIVERTED columns have several values which denote false and true.

- DISTANCE is a string with " miles" concatenated onto the numerical value
- There are multiple null value cells in ORIGINSTATE, ORIGINSTATENAME, DESTSTATE and DESTSTATENAME
- There are multiple null value cells present in DEPTIME, DEPDELAY, ARRTIME and ARRDELAY.

2.2 **Use Cases:**

After the initial assessment of the dataset we came up with some use cases for our project. The use cases are:

- Which Airlines appear the most in this dataset – Which flights get booked the most?
- Which Airlines fly larger distances?
- What is the average delay per Airline?
- Which Airlines have had the longest delays?
- Which routes are associated with these Airlines and delays?
- What is the percentage of flight getting delayed or cancelled between two destinations for a specific airline?

We felt that after knowing the information for these questions the user then can decide whether he needs to fly through his preferred airline or any other airline which is rated best for that route alone.

Among the use cases mentioned we feel that the use case "Which Airline appear the most in this dataset – Which flights get booked the most?" requires very less or no cleaning because even though the airline name has airline code concatenated at the end it does not make much difference as the airline will be still same for this use case.

Other use cases require distance flown, average delay and delay times which requires cleaning of delay times and making sure that delay time column doesn't have any "Null" values. And the other use case "Which routes are associated with these Airlines and delays?" requires the origin and destination airport as well as the delay for each flight flown through the route.

3. Data Cleaning methods and Process

For our data cleaning process, we have used Python and OpenRefine. We have used python for the initial cleanup of the dataset. Python was used in restructuring the dataset into a more readable and understandable CSV file. Data cleaning steps often need repeating with multiple files. OpenRefine is perfect for speeding up repetitive tasks by replaying previous actions on multiple datasets. OpenRefine (previously Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data. It is more suitable for large data cleaning operations.

3.1 Python

We have used Python for the initial cleanup of the dataset, which included splitting the data into rows and columns and dropping some rows and columns from the original dataset. After loading the dataset, using pandas we divided the data into rows and columns using "|" as delimiter.

```
import pandas as pd
import numpy as np

flights = pd.read_csv('flights.txt', sep='|')
```

```
flights
```

	TRANSACTIONID	FLIGHTDATE	AIRLINECODE	AIRLINENAME	TAILNUM	FLIGHTNUM	ORIGINAIRPORTCODE	ORIGAIRPORTNAME	ORIGINCITYNAM
0	54548800	20020101	WN	Southwest Airlines Co.: WN	N103@@	1425	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerq
1	55872300	20020101	CO	Continental Air Lines Inc.: CO	N83872	150	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerq
2	54388800	20020101	WN	Southwest Airlines Co.: WN	N334@@	249	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerq
3	54486500	20020101	WN	Southwest Airlines Co.: WN	N699@@	902	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerq
4	55878700	20020103	CO	Continental Air Lines Inc.: CO	N58606	234	ABQ	AlbuquerqueNM: Albuquerque International Sunport	Albuquerq

Fig 3.1 – Loading the file into Python

After dividing the data into rows and columns we found that the data contains more than 1 million records and 31 columns, as seen below in Fig 3.2. When we tried to load the dataset into OpenRefine for further data cleaning, it froze, and the system hung. We tried multiple time by increasing the memory consumption bandwidth of OpenRefine but still we faced the same issue. For this reason, we identified 8 columns that we thought would not help in answering the questions we had put forth as use cases and dropped those columns. We also selected only the latest data, which in this case is data of the flights for the year 2016. This new dataset was much smaller as seen in Fig 3.3, and was of the right size of load into OpenRefine.

```
flights.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1191805 entries, 0 to 1191804
Data columns (total 31 columns):
TRANSACTIONID        1191805 non-null int64
FLIGHTDATE           1191805 non-null int64
AIRLINECODE          1191805 non-null object
AIRLINENAME          1191805 non-null object
TAILNUM              1034988 non-null object
FLIGHTNUM            1191805 non-null int64
ORIGINAIRPORTCODE    1191805 non-null object
ORIGAIRPORTNAME      1191805 non-null object
ORIGINCITYNAME       1191805 non-null object
ORIGINSTATE          1180963 non-null object
ORIGINSTATENAME      1180963 non-null object
DESTAIRPORTCODE      1191805 non-null object
DESTAIRPORTNAME      1191805 non-null object
DESTCITYNAME         1191805 non-null object
DESTSTATE            1180967 non-null object
DESTSTATENAME        1180967 non-null object
CRSDEPTIME           1191805 non-null int64
DEPTIME              1163470 non-null float64
DEPDELAY             1163470 non-null float64
TAXIOUT              1011833 non-null float64
WHEELSOFF            1011791 non-null float64
WHEELSON             1010225 non-null float64
TAXIIN               1010320 non-null float64
CRSARRTIME           1191805 non-null int64
ARRTIME              1161439 non-null float64
ARRDELAY             1160545 non-null float64
CRSELAPSEDTIME       1191383 non-null float64
ACTUALELAPSEDTIME    1160545 non-null float64
CANCELLED            1191805 non-null object
DIVERTED             1191805 non-null object
DISTANCE             1191805 non-null object
dtypes: float64(10), int64(5), object(16)
memory usage: 281.9+ MB
```

Fig 3.2 – Initial dataset information

```
flights = flights.drop("TRANSACTIONID", 1)
flights = flights.drop("TAILNUM", 1)
flights = flights.drop("TAXIOUT", 1)
flights = flights.drop("WHEELSOFF", 1)
flights = flights.drop("TAXIIN", 1)
flights = flights.drop("WHEELSON", 1)
flights = flights.drop("CRSELAPSEDTIME", 1)
flights = flights.drop("ACTUALELAPSEDTIME", 1)

flights.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1191805 entries, 0 to 1191804
Data columns (total 23 columns):
FLIGHTDATE           1191805 non-null int64
AIRLINECODE          1191805 non-null object
AIRLINENAME          1191805 non-null object
FLIGHTNUM            1191805 non-null int64
ORIGINAIRPORTCODE    1191805 non-null object
ORIGAIRPORTNAME      1191805 non-null object
ORIGINCITYNAME       1191805 non-null object
ORIGINSTATE          1180963 non-null object
ORIGINSTATENAME      1180963 non-null object
DESTAIRPORTCODE      1191805 non-null object
DESTAIRPORTNAME      1191805 non-null object
DESTCITYNAME         1191805 non-null object
DESTSTATE            1180967 non-null object
DESTSTATENAME        1180967 non-null object
CRSDEPTIME           1191805 non-null int64
DEPTIME              1163470 non-null float64
DEPDELAY             1163470 non-null float64
CRSARRTIME           1191805 non-null int64
ARRTIME              1161439 non-null float64
ARRDELAY             1160545 non-null float64
CANCELLED            1191805 non-null object
DIVERTED             1191805 non-null object
DISTANCE             1191805 non-null object
dtypes: float64(4), int64(4), object(15)
memory usage: 209.1+ MB
```

Fig 3.3 – Modified dataset information

3.2 OpenRefine

We loaded the newly created dataset into OpenRefine and continued with the data cleaning process.

<u>Transformations done on FLIGHTDATE column:</u>

In the original dataset, the flight date is a blocked text column and we converted the FLIGHTDATE column into a date column using “value.slice(0,4) + "-" + value.slice(4,6) + "-" + value.slice(6,8)”

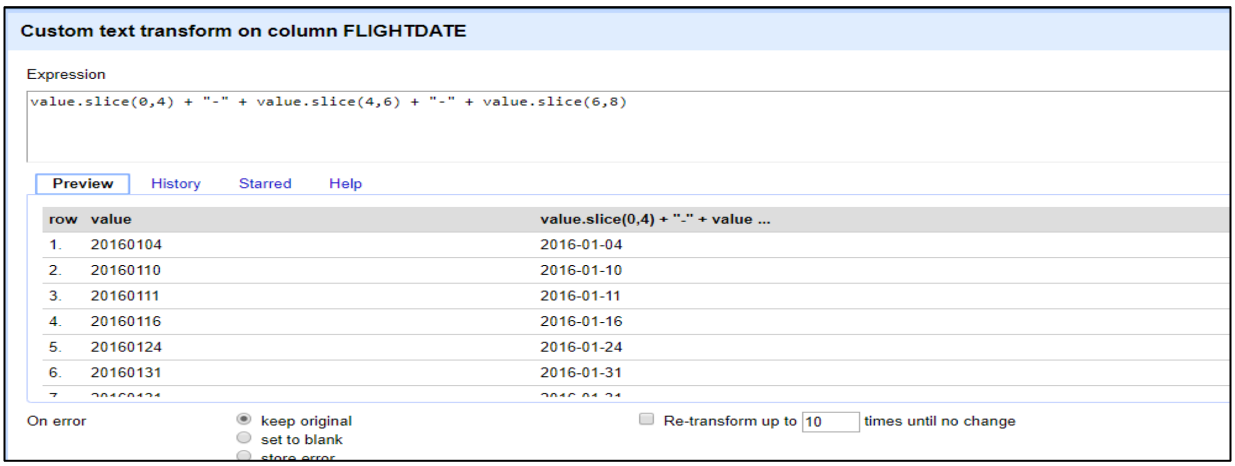


Fig 3.1 – Transformations made to the FlightDate column

<u>Transformations done on AIRLINENAME, ORIGAIRPORTNAME and DESTAIRPORTNAME columns:</u>

In the original dataset, all the data present in the above-mentioned columns contain some unwanted data which is concatenated either front or back of the required data. So, using OpenRefine we split the column into two using a separator and removed the unwanted data column, as seen below:

AIRLINENAME column before making changes

AIRLINECODE	AIRLINENAME	FLIGHTNUM	ORIGINAIRPO
DL	Delta Air Lines Inc.: DL	1079	BIL
DL	Delta Air Lines Inc.: DL	2079	BIL
OO	SkyWest Airlines Inc.: OO	4498	BIL
OO	SkyWest Airlines Inc.: OO	4692	BIL
DL	Delta Air Lines Inc.: DL	1390	BIL
UA	United Air Lines Inc.: UA	536	BIL
DL	Delta Air Lines Inc.: DL	1390	BIL
DL	Delta Air Lines Inc.: DL	1912	BIL
OO	SkyWest Airlines Inc.: OO	6059	BIL
UA	United Air Lines Inc.: UA	377	BIL

Splitting the column into two based on a separator

37007 rows

Show as: rows records Show: 5 10 25 50 rows

All	FLIGHTDATE	AIRLINECODE	AIRLINENAME	FLIGHTNUM	ORIGINAIRPORT	ORIGAIRPORTNAME	ORIGINCITYNAM	ORIGINSTATE
1.	2016-01-04	DL	Delta Air Lines Inc.: DL	1079	BIL	BillingsMT: Billings Logan International	Billings	MT
2.	2016-01-10	DL	Delta Air Lines Inc.: DL	2079	BIL	BillingsMT: Billings Logan International	Billings	MT
3.	2016-01-11	OO	SkyWest Airlines Inc.: OO	4498	BIL	BillingsMT: Billings Logan International	Billings	MT
4.	2016-01-16	OO	SkyWest Airlines Inc.: OO	4692	BIL	BillingsMT: Billings Logan International	Billings	MT
5.	2016-01-24	DL	Delta Air Lines Inc.: DL	1390	BIL	BillingsMT: Billings Logan International	Billings	MT
6.	2016-01-31	UA	United Air Li					
7.	2016-01-31	DL	Delta Air Lin					
8.	2016-05-03	DL	Delta Air Lin					
9.	2016-05-07	OO	SkyWest Air OO					
10.	2016-05-09	UA	United Air Li					

Split column AIRLINENAME into several columns

How to Split Column

by separator

Separator : regular expression

Split into columns at most (leave blank for no limit)

by field lengths

List of integers separated by commas, e.g., 5, 7, 15

After Splitting

Guess cell type

Remove this column

OK Cancel

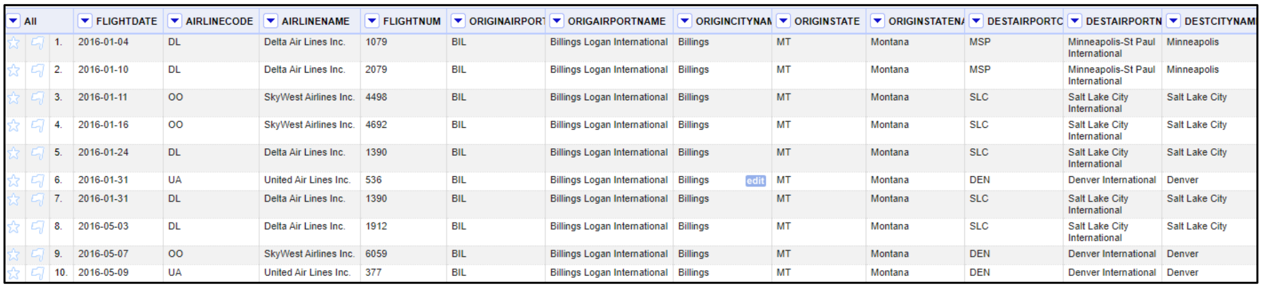
AIRLINENAME column after cleaning the column

37007 rows

Show as: rows records Show: 5 10 25 50 rows

All	FLIGHTDATE	AIRLINECODE	AIRLINENAME 1	AIRLINENAME 2	FLIGHTNU
1.	2016-01-04	DL	Delta Air Lines Inc.	DL	1079
2.	2016-01-10	DL	Delta Air Lines Inc.	DL	2079
3.	2016-01-11	OO	SkyWest Airlines Inc.	OO	4498
4.	2016-01-16	OO	SkyWest Airlines Inc.	OO	4692
5.	2016-01-24	DL	Delta Air Lines Inc.	DL	1390
6.	2016-01-31	UA	United Air Lines Inc.	UA	536
7.	2016-01-31	DL	Delta Air Lines Inc.	DL	1390
8.	2016-05-03	DL	Delta Air Lines Inc.	DL	1912
9.	2016-05-07	OO	SkyWest Airlines Inc.	OO	6059
10.	2016-05-09	UA	United Air Lines Inc.	UA	377

We deleted the AIRLINENAME 2 column and renamed the AIRLINENAME 1 to AIRLINENAME. In a similar manner we have changed the ORGAIRPORTNAME and DESTAIRPORTNAME columns too. All the rows (37007) present in the dataset is changed by removing all the concatenations present.

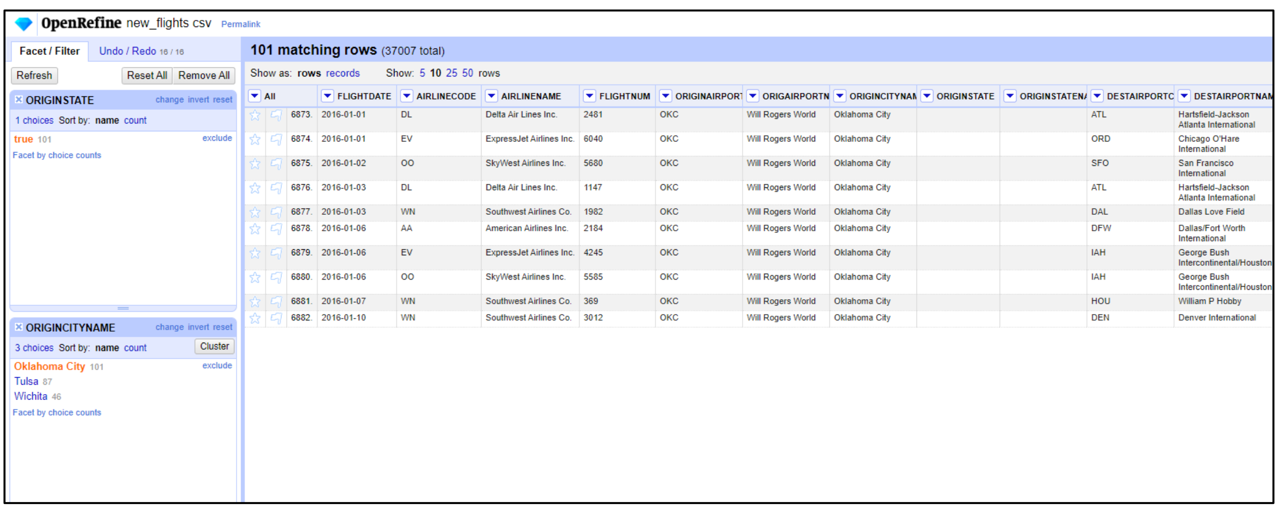


All	FLIGHTDATE	AIRLINECODE	AIRLINENAME	FLIGHTNUM	ORIGINAIRPORT	ORIGAIRPORTNAME	ORIGINCITYNAM	ORIGINSTATE	ORIGINSTATENA	DESTAIRPORTC	DESTAIRPORTN	DESTCITYNAM
1.	2016-01-04	DL	Delta Air Lines Inc.	1079	BIL	Billings Logan International	Billings	MT	Montana	MSP	Minneapolis-St Paul International	Minneapolis
2.	2016-01-10	DL	Delta Air Lines Inc.	2079	BIL	Billings Logan International	Billings	MT	Montana	MSP	Minneapolis-St Paul International	Minneapolis
3.	2016-01-11	OO	SkyWest Airlines Inc.	4498	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City
4.	2016-01-16	OO	SkyWest Airlines Inc.	4692	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City
5.	2016-01-24	DL	Delta Air Lines Inc.	1390	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City
6.	2016-01-31	UA	United Air Lines Inc.	536	BIL	Billings Logan International	Billings edit	MT	Montana	DEN	Denver International	Denver
7.	2016-01-31	DL	Delta Air Lines Inc.	1390	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City
8.	2016-05-03	DL	Delta Air Lines Inc.	1912	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City
9.	2016-05-07	OO	SkyWest Airlines Inc.	6059	BIL	Billings Logan International	Billings	MT	Montana	DEN	Denver International	Denver
10.	2016-05-09	UA	United Air Lines Inc.	377	BIL	Billings Logan International	Billings	MT	Montana	DEN	Denver International	Denver

<u>Transformations done on ORIGINSTATE, ORIGINSTATENAME, DESTSTATE and DESTSTATENAME columns:</u>

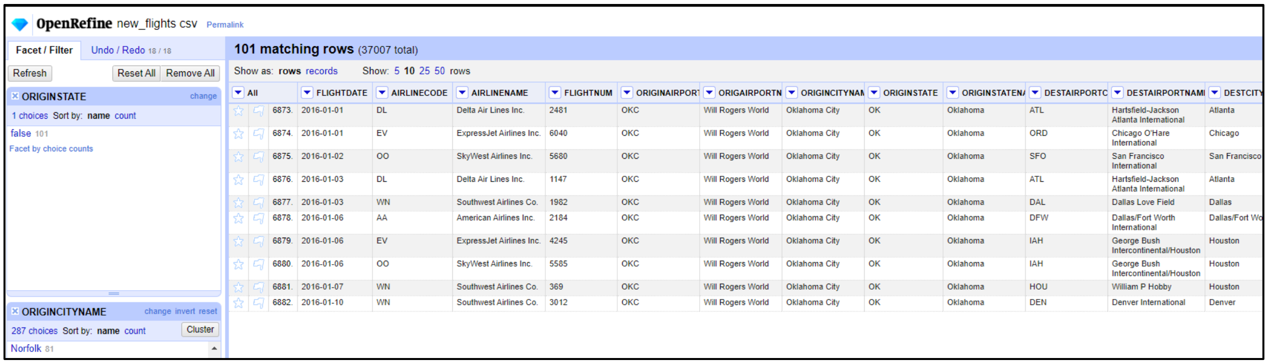
In the original dataset, the above-mentioned columns have blank value which we found by creating a custom facet in “Facet -> Customized facets -> Facet by blank (null or empty string)”.

Using facets to find the blank values in the ORIGINSTATE and ORIGINSTATENAME name columns.



All	FLIGHTDATE	AIRLINECODE	AIRLINENAME	FLIGHTNUM	ORIGINAIRPORT	ORIGAIRPORTN	ORIGINCITYNAM	ORIGINSTATE	ORIGINSTATENA	DESTAIRPORTC	DESTAIRPORTNAM
6873.	2016-01-01	DL	Delta Air Lines Inc.	2481	OKC	Will Rogers World	Oklahoma City			ATL	Hartsfield-Jackson Atlanta International
6874.	2016-01-01	EV	ExpressJet Airlines Inc.	6040	OKC	Will Rogers World	Oklahoma City			ORD	Chicago O'Hare International
6875.	2016-01-02	OO	SkyWest Airlines Inc.	5680	OKC	Will Rogers World	Oklahoma City			SFO	San Francisco International
6876.	2016-01-03	DL	Delta Air Lines Inc.	1147	OKC	Will Rogers World	Oklahoma City			ATL	Hartsfield-Jackson Atlanta International
6877.	2016-01-03	WN	Southwest Airlines Co.	1982	OKC	Will Rogers World	Oklahoma City			DAL	Dallas Love Field
6878.	2016-01-06	AA	American Airlines Inc.	2184	OKC	Will Rogers World	Oklahoma City			DFW	Dallas/Fort Worth International
6879.	2016-01-06	EV	ExpressJet Airlines Inc.	4245	OKC	Will Rogers World	Oklahoma City			IAH	George Bush Intercontinental/Houston
6880.	2016-01-06	OO	SkyWest Airlines Inc.	5585	OKC	Will Rogers World	Oklahoma City			IAH	George Bush Intercontinental/Houston
6881.	2016-01-07	WN	Southwest Airlines Co.	369	OKC	Will Rogers World	Oklahoma City			HOU	William P Hobby
6882.	2016-01-10	WN	Southwest Airlines Co.	3012	OKC	Will Rogers World	Oklahoma City			DEN	Denver International

Filled the blank values in the ORIGINSTATE and ORIGINSTATENAME name columns and in a similar way we filled the blank values present in the DESTSTATE and DESTSTATENAME.



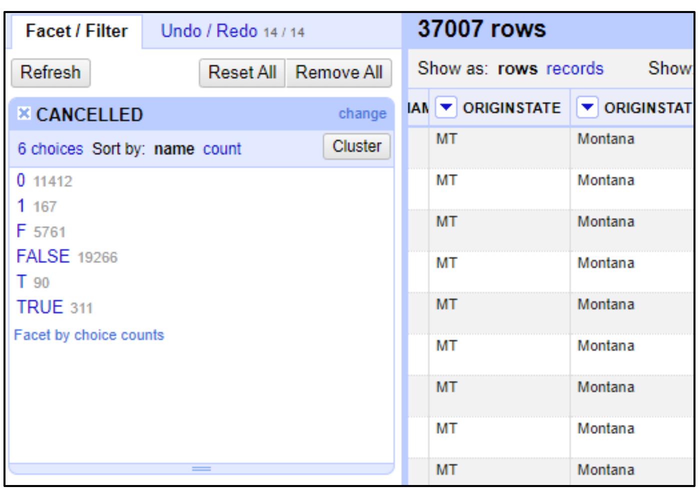
All	FLIGHTDATE	AIRLINECODE	AIRLINENAME	FLIGHTNUM	ORIGINAIRPORT	ORIGAIRPORTN	ORIGINCITYNAM	ORIGINSTATE	ORIGINSTATEN/	DESTAIRPORTC	DESTAIRPORTNAMI	DESTCITY
6873.	2016-01-01	DL	Delta Air Lines Inc.	2481	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	ATL	Hartsfield-Jackson Atlanta International	Atlanta
6874.	2016-01-01	EV	ExpressJet Airlines Inc.	6040	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	ORD	Chicago O'Hare International	Chicago
6875.	2016-01-02	OO	SkyWest Airlines Inc.	5680	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	SFO	San Francisco International	San Francisco
6876.	2016-01-03	DL	Delta Air Lines Inc.	1147	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	ATL	Hartsfield-Jackson Atlanta International	Atlanta
6877.	2016-01-03	WN	Southwest Airlines Co.	1982	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	DAL	Dallas Love Field	Dallas
6878.	2016-01-06	AA	American Airlines Inc.	2184	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	DFW	Dallas/Fort Worth International	Dallas/Fort Wo
6879.	2016-01-06	EV	ExpressJet Airlines Inc.	4245	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	IAH	George Bush Intercontinental/Houston	Houston
6880.	2016-01-06	OO	SkyWest Airlines Inc.	5585	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	IAH	George Bush Intercontinental/Houston	Houston
6881.	2016-01-07	WN	Southwest Airlines Co.	369	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	HOU	William P Hobby	Houston
6882.	2016-01-10	WN	Southwest Airlines Co.	3012	OKC	Will Rogers World	Oklahoma City	OK	Oklahoma	DEN	Denver International	Denver

<u>Transformations done on CANCELLED and DIVERTED columns:</u>

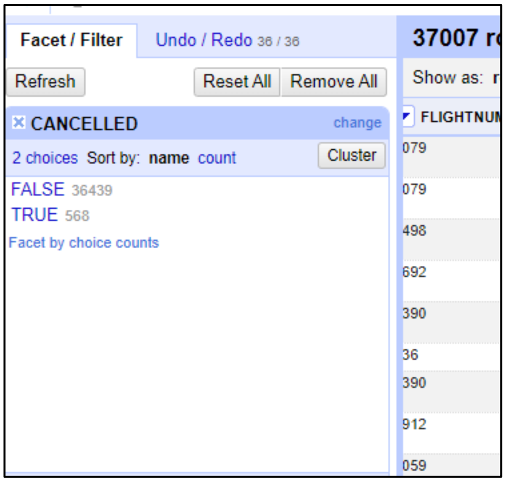
In the original dataset, CANCELLED and DIVERTED columns have multiple values denoting the same context. For example, to denote whether the flight is cancelled in the data multiple values like "T, TRUE and 1" have been used. In a similar way to denote the flight is not cancelled multiple values like "F, FALSE and 0" have been used.

To normalize all the different values which denotes the same into one value we have used "Facet -> Text facet". Form the facets we have edited values that denote true to "TRUE" from "T, 1, TRUE". In a similar way we have changed all the values that denoted false to "FALSE" from "F, 0, FALSE".

Multiple values before changing or normalizing them into a single value.



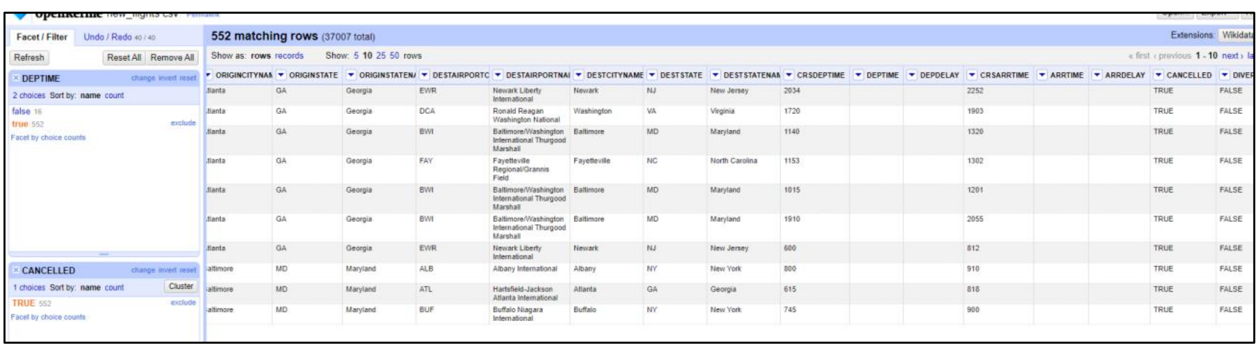
After changing through facets.



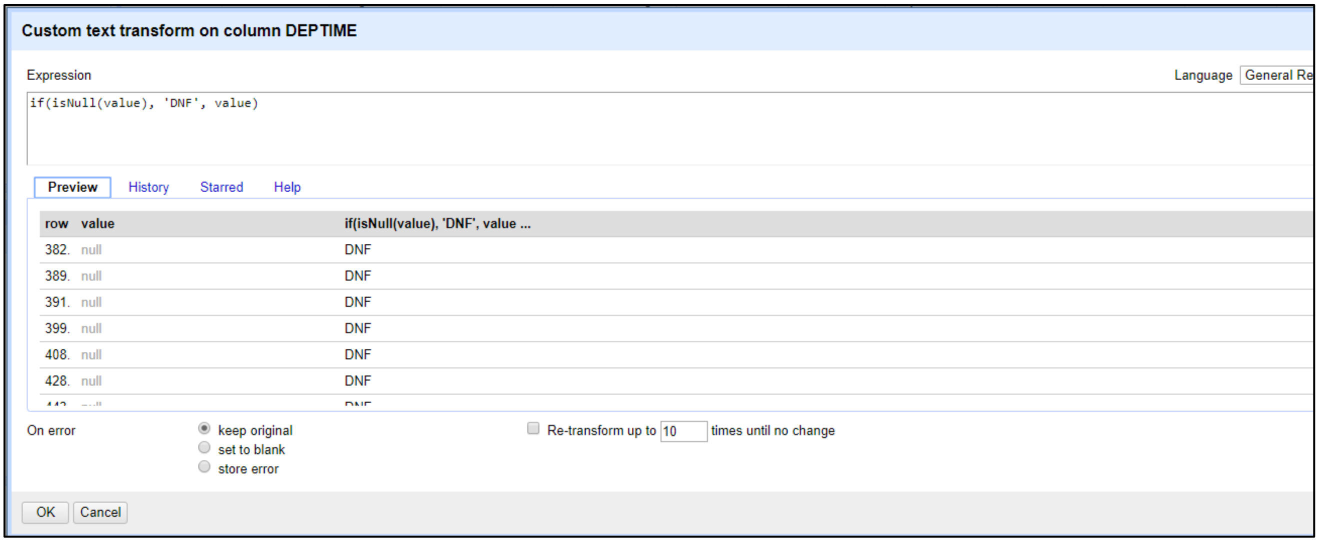
In a similar way we have changed the values present in “DIVERTED” column too.

Transformations done on DEPTIME, DEPDELAY, ARRTIME and ARRDELAY columns:

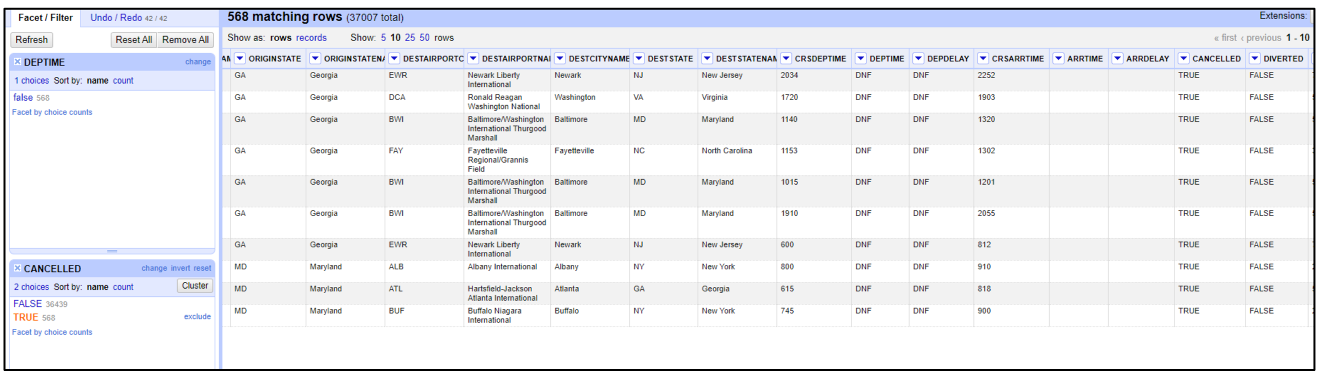
In the original dataset, there are empty or null values in DEPTIME, DEPDELAY, ARRTIME and ARRDELAY columns. For the DEPTIME and DEPDELAY columns we have changed the empty or null value cells into “DNF” when the flight is cancelled.



Now with the help of facets we have identified the null value cells and then by using the transform option we are going to replace the null value cells into “DNF” using the GREL “if(isNull(value), 'DNF', value)”.



After replacing the null value cells. 568 rows have been changed into “DNF”.

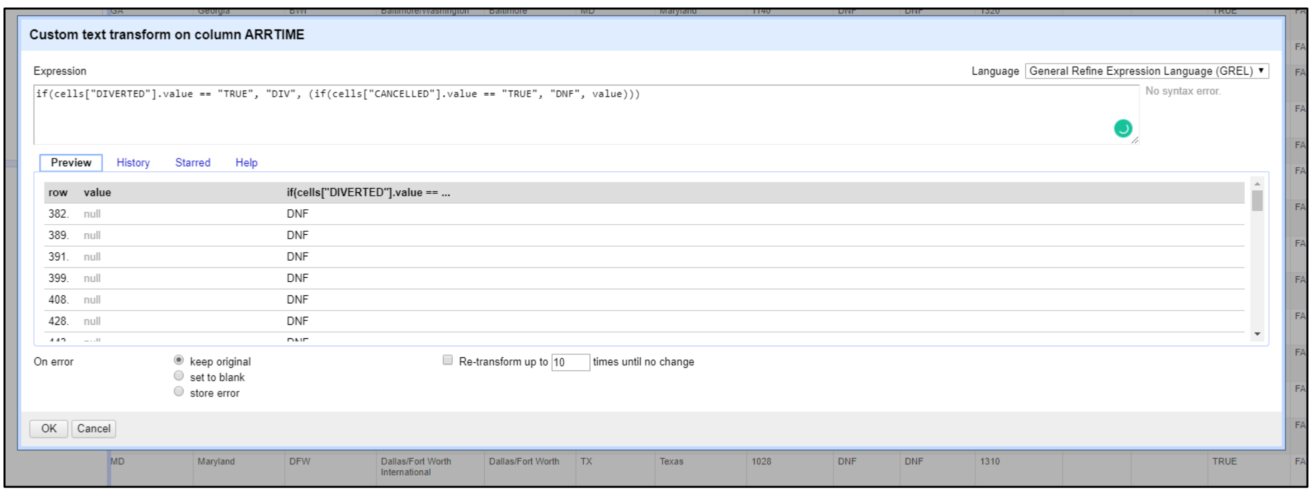


ORIGINSTATE	ORIGINSTATENA	DESTAIRPORTC	DESTAIRPORTNA	DESTCITYNAME	DESTSTATE	DESTSTATENAM	CRSDEPTIME	DEPTIME	DEPDELAY	CRSARRTIME	ARRTIME	ARRDELAY	CANCELLED	DIVERTED
GA	Georgia	EWR	Newark Liberty International	Newark	NJ	New Jersey	2034	DNF	DNF	2252			TRUE	FALSE
GA	Georgia	DCA	Ronald Reagan Washington National	Washington	VA	Virginia	1720	DNF	DNF	1903			TRUE	FALSE
GA	Georgia	BWI	Baltimore/Washington International Thurgood Marshall	Baltimore	MD	Maryland	1140	DNF	DNF	1320			TRUE	FALSE
GA	Georgia	FAY	Fayetteville Regional/Grannis Field	Fayetteville	NC	North Carolina	1153	DNF	DNF	1302			TRUE	FALSE
GA	Georgia	BWI	Baltimore/Washington International Thurgood Marshall	Baltimore	MD	Maryland	1015	DNF	DNF	1201			TRUE	FALSE
GA	Georgia	BWI	Baltimore/Washington International Thurgood Marshall	Baltimore	MD	Maryland	1910	DNF	DNF	2055			TRUE	FALSE
GA	Georgia	EWR	Newark Liberty International	Newark	NJ	New Jersey	600	DNF	DNF	812			TRUE	FALSE
MD	Maryland	ALB	Albany International	Albany	NY	New York	800	DNF	DNF	910			TRUE	FALSE
MD	Maryland	ATL	Hartsfield-Jackson Atlanta International	Atlanta	GA	Georgia	615	DNF	DNF	818			TRUE	FALSE
MD	Maryland	BUF	Buffalo Niagara International	Buffalo	NY	New York	745	DNF	DNF	900			TRUE	FALSE

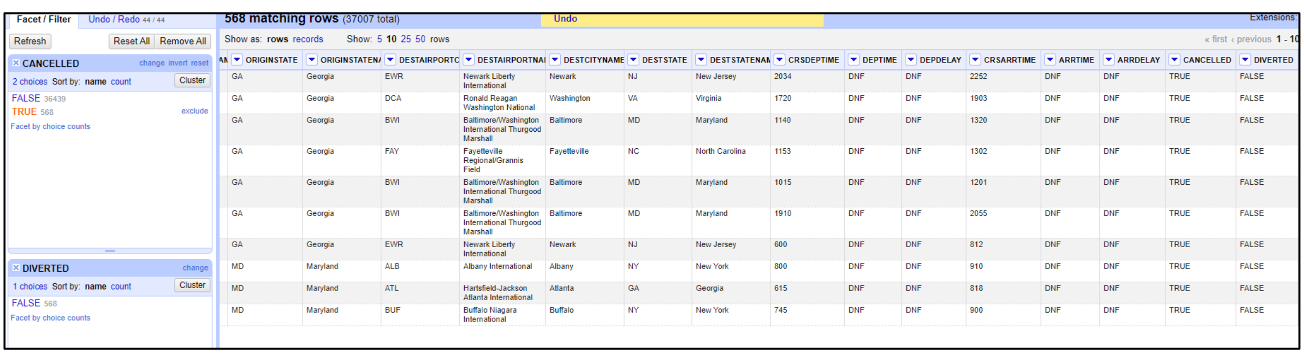
For the ARRTIME and ARRDELAY columns we cannot replace all the null value cells into “DNF” as the flight may be diverted to another location and since it is flown from the one location to another it will have the DEPTIME, DEPDELAY for all the cells and ARRTIME for the cells which have been recorded. So, we are going to replace the empty cells with “DNF” when the flight is “Cancelled” and with “DIV” when the flight is “Diverted”.

When the flight is "Cancelled" before transformation.

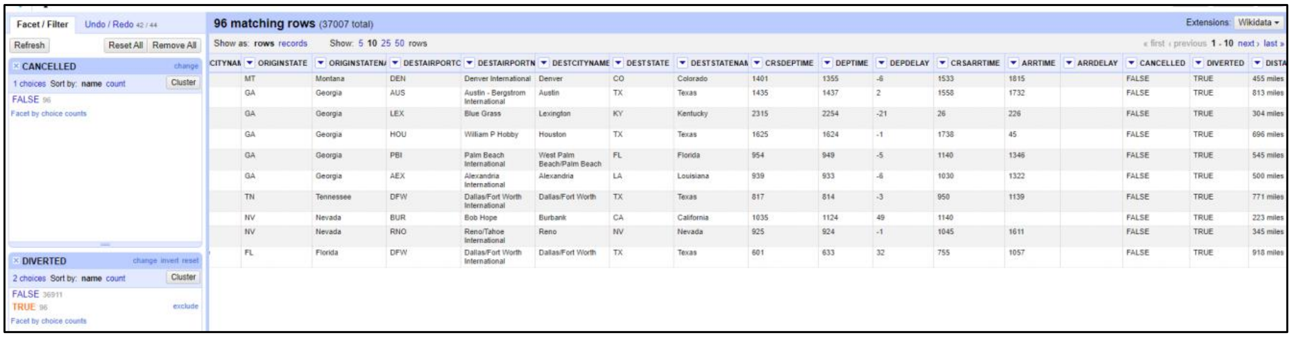
Using the GREL "if(cells["DIVERTED"].value == "TRUE", "DIV", (if(cells["CANCELLED"].value == "TRUE", "DNF", value)))" for transformation.



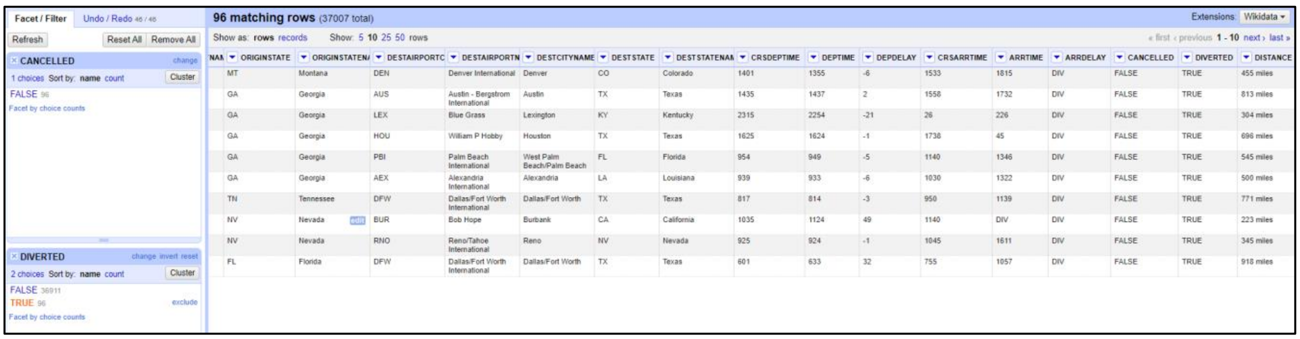
When the flight is "Cancelled" after transformation.



When the flight is “Diverted” before transformation.



When the flight is “Diverted” after transformation.



<u>Transformations done on DISTANCE column:</u>

In the original dataset, the DISTANCE column is in text format with the text “miles” concatenated to the number and we have converted it to number by removing the concatenation.

DEPDELAY	CRSARRTIME	ARRTIME	ARRDELAY	CANCELLED	DIVERTED	DISTANCE
-7	900	844	-16	FALSE	FALSE	748 miles
-9	902	841	-21	FALSE	FALSE	748 miles
-8	1855	1834	-21	FALSE	FALSE	387 miles
-11	1428	1414	-14	FALSE	FALSE edit	387 miles
-3	747	746	-1	FALSE	FALSE	387 miles
-10	1622	1554	-28	FALSE	FALSE	455 miles
0	747	731	-16	FALSE	FALSE	387 miles
-5	745	736	-9	FALSE	FALSE	387 miles
-6	1135	1138	3	FALSE	FALSE	455 miles
-3	843	827	-16	FALSE	FALSE	455 miles

After removing the concatenation

CANCELLED	DIVERTED	DISTANCE
FALSE	FALSE	748
FALSE	FALSE	748
FALSE	FALSE	387
FALSE	FALSE	387
FALSE	FALSE	387
FALSE	FALSE	455
FALSE	FALSE	387
FALSE	FALSE	387
FALSE	FALSE	455
FALSE	FALSE	455

<u>Transformations done to change the text to number in multiple columns:</u>

In the original dataset, there are multiple columns in which the numbers are marked as text but not as number. The columns in which the changes to be made are "FLIGHTNUM", "CRSDEPTIME", "DEPTIME", "DEPDELAY", "CRSARRTIME", "ARRTIME", "ARRDELAY".

37007 rows — Extensions: Wikidata

Show as: rows records — Show: 5 10 25 50 rows — « first ‹ previous 1 - 10 next › last »

'NUM	ORIGINAIRPORT	ORIGAIRPORTNAME	ORIGINCITYNAM	ORIGINSTATE	ORIGINSTATEN	DESTAIRPORTC	DESTAIRPORTN	DESTCITYNAME	DESTSTATE	DESTSTATENAM	CRSDEPTIME	DEPTIME	DEPDELAY	CRSARRTIME	ARRTIME
1079	BIL	Billings Logan International	Billings	MT	Montana	MSP	Minneapolis-St Paul International	Minneapolis	MN	Minnesota	600	553	-7	900	844
2079	BIL	Billings Logan International	Billings	MT	Montana	MSP	Minneapolis-St Paul International	Minneapolis	MN	Minnesota	600	551	-9	902	841
4498	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City	UT	Utah	1730	1722	-8	1855	1834
4692	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City	UT	Utah	1301	1250	-11	1428	1414
1390	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City	UT	Utah	620	617	-3	747	746
536	BIL	Billings Logan International	Billings	MT	Montana	DEN	Denver International	Denver	CO	Colorado	1445	1435	-10	1622	1554
1390	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City	UT	Utah	620	620	0	747	731
1912	BIL	Billings Logan International	Billings	MT	Montana	SLC	Salt Lake City International	Salt Lake City	UT	Utah	630	625	-5	745	736
6059	BIL	Billings Logan International	Billings	MT	Montana	DEN	Denver International	Denver	CO	Colorado	1001	955	-6	1135	1138
377	BIL	Billings Logan International	Billings	MT	Montana	DEN	Denver International	Denver	CO	Colorado	710	707	-3	843	827

4. Results

4.1 Number of changes to the dataset

The following are the number of changes made to each column in the dataset. For further information regarding the changes, please refer to the attached CSV file that shows the original columns in Red and the edited version of the column in Green.

Column_name	No of Rows changed
FLIGHTDATE	37007
AIRLINENAME	37007

ORIGAIRPORTNAME	37007
DESTAIRPORTNAME	37007
ORIGINSTATENAME	245
ORIGINSTATE	245
DESTSTATE	267
DESTSTATENAME	267
CANCELLED	17430
DIVERTED	24328
DEPTIME	552
DEPDELAY	552
ARRTIME	584
ARRDELAY	664
DISTANCE	37007

4.2 IC checks

The IC checks were performed in Python and are present in the attached Jupyter Notebook along with a detailed description for each step. We performed the following IC checks:

1. Check for Null values
2. Check for duplicate rows
3. Primary key check
4. Check for column value lengths
5. Check the values of each column

Please refer to the Jupyter Notebook for further information regarding IC checks.

4.3 Workflow Model

The workflow model represents the workflow of the data and how it changed during the different steps of the process. To create the workflow model of our data cleaning process we have used or2yw tool, YESWorkflow editor and Graphviz. Since we used OpenRefine for most of our data cleaning process we felt that or2yw tool would be better suited for our needs. We have extracted the cleaning recipe from OpenRefine and using the or2yw tool we have created the ".yw" file and used the file to create a workflow model in the YESWorkflow editor.

Firstly, the data root is the new_flights.csv and the output file is the cleaned dataset file. In the workflow we can see all the cleaning operations we performed on the original dataset in a sequential order. In the below image we can see the operations performed on the dataset in a sequential order containing only information about the data.

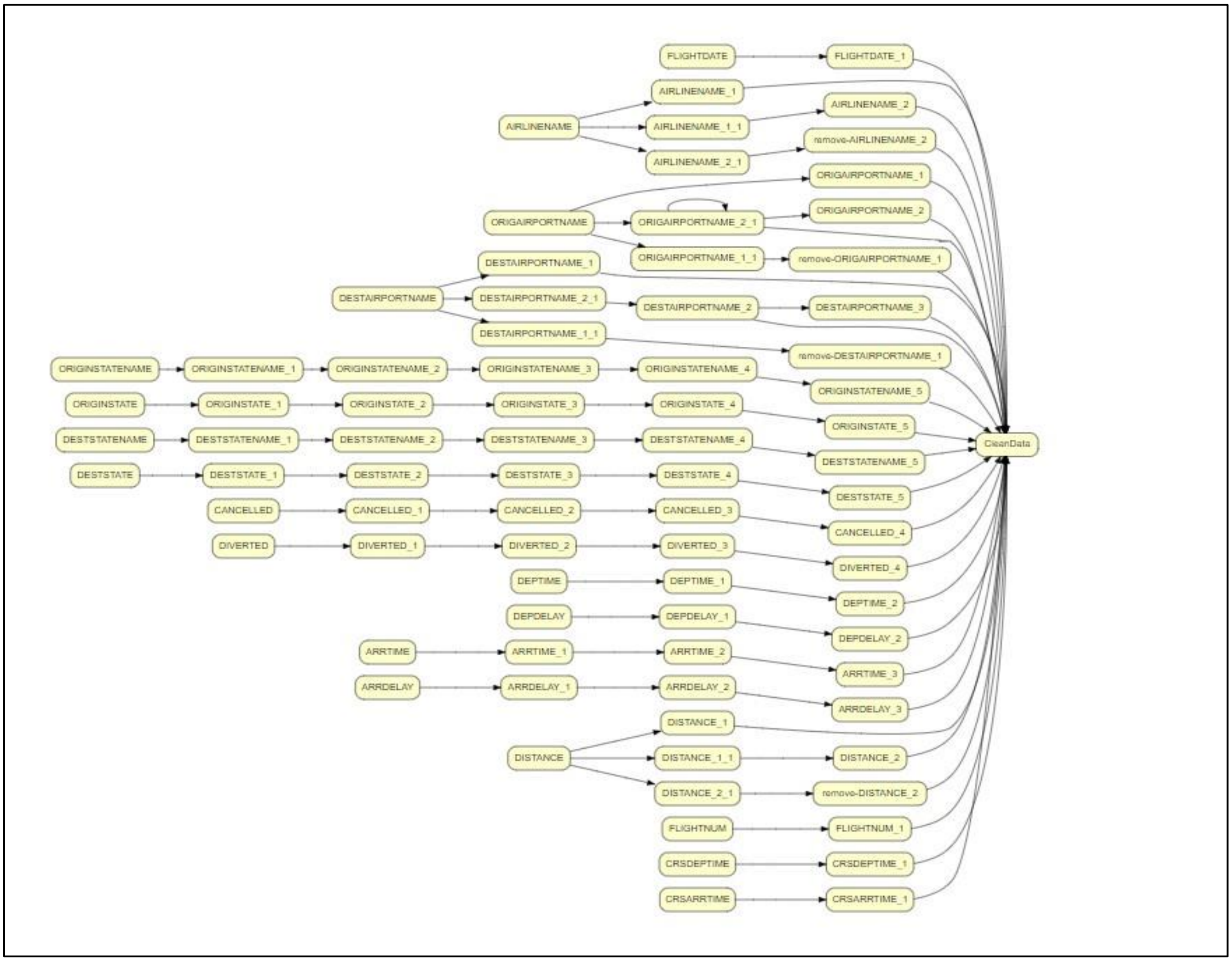


Fig 4.1 – The operations performed

Now, from the above picture we can see all the operations performed and an outline of the data flow which ended at the output file.

The Parallel YESWorkflow of all the operations performed on the dataset.

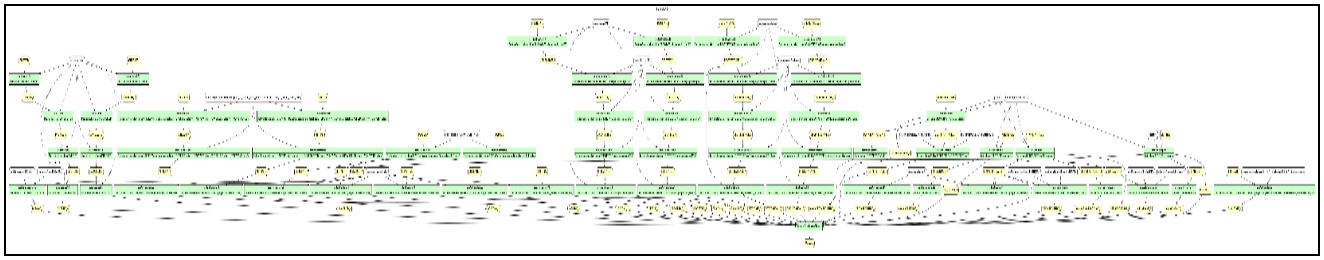


Fig 4.2 – Parallel YESWorkFlow

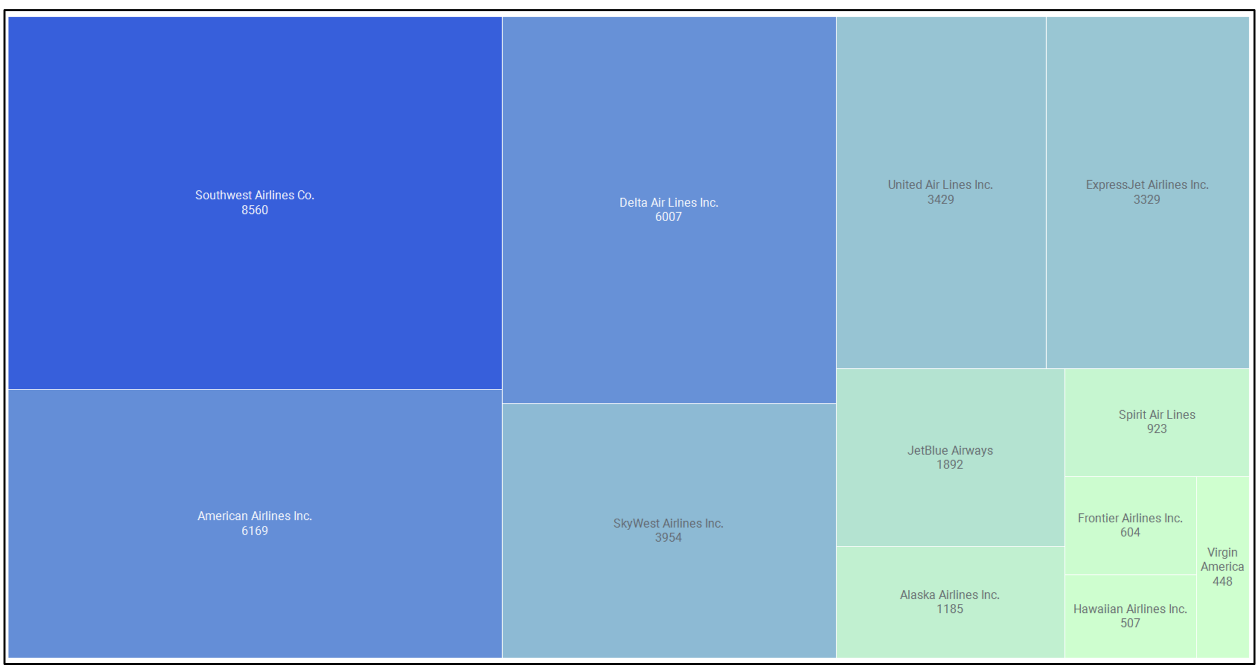
Since the workflow model is very huge and not in a readable position so, we have attached the pdf file for the workflow:

YESWorkflow_parallel.pdf

4.4 Visualizations

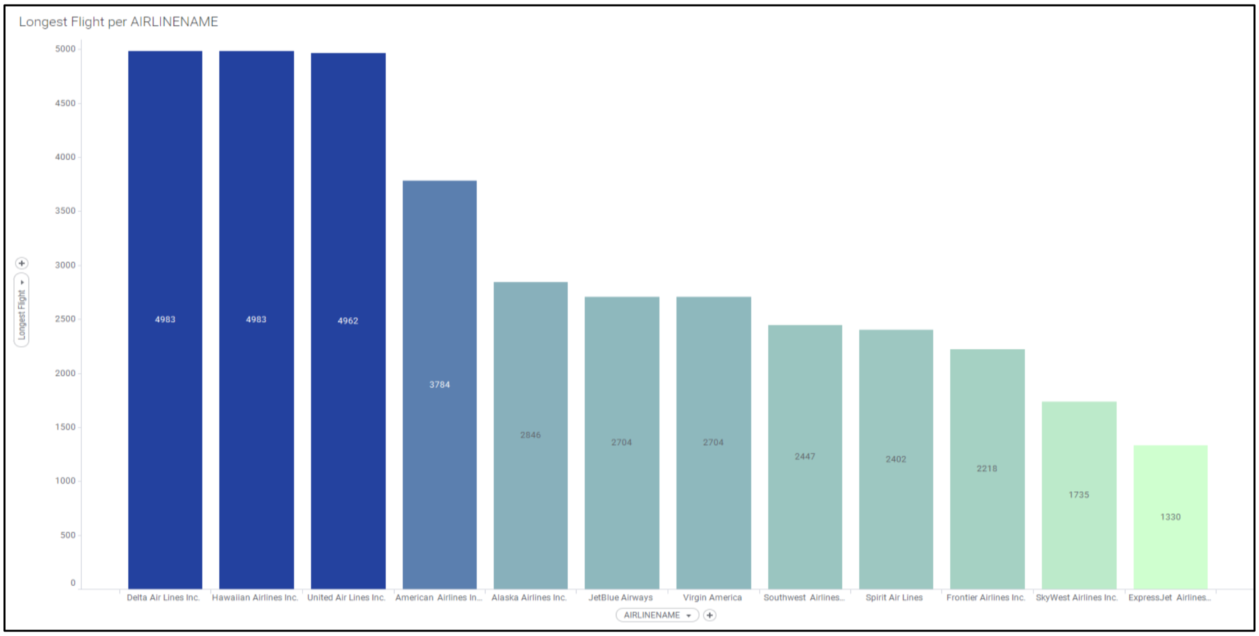
We created visualization in Spotfire using the cleaned data in order to get answers for our use cases. Some of the visualizations were:

1. Which flights appear the most in the dataset – Which flights get booked the most?



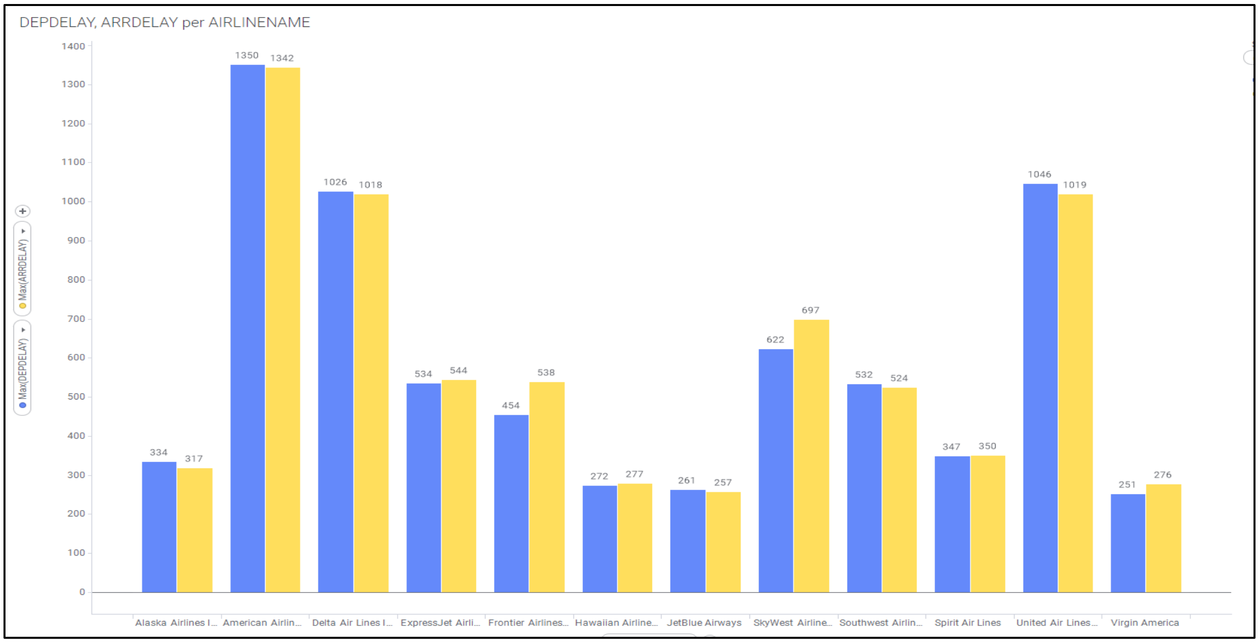
From this we see that Southwest Airlines, American Airlines and Delta Airlines are the top three airlines in terms of number of flights in this time frame. This also means that they were the top booked airlines in this dataset.

2. Which Airlines fly larger distances?



We can see that Delta Airlines, Hawaiin Airlines and United Airlines have the longest flights in terms of distance. We also saw from the details of these flights that these are cross country flights to and from Hawaii.

3. Which Airlines have had the longest delays?



We found that American Airlines has had the maximum delay in term of departure and arrival with a departure delay of 1350 minutes and an arrival delay of 1342 minutes.

5. Conclusion and Future Work

From this project, we learned how to convert a raw file into actionable data, using tools like Python and OpenRefine. We saw that it is difficult to handle larger datasets in OpenRefine and preprocessing in Python is required in order to do so.

Once we were done with the cleaning process, our file was easy to understand, and we were able to successfully load this file into Spotfire to create visualizations that helped us answer the use cases that we had listed out.

We also saw how the cleaning process involves looking at data from the end perspective since we had to deal with columns that had data such as time related data which we did not modify since we realized that the unmodified data could still be used to obtain valid results. In the future, this data could be expanded to cover the rest of the dataset and advanced techniques such as Machine Learning or Deep Learning algorithms could be used to obtain behavioral statistics of each Airline using this data.

6. References

Flights. (n.d.). Retrieved December 14, 2019, from https://kaggle.com/mmetter/flights

Openrefine.github.com. (n.d.). Retrieved December 14, 2019, from https://openrefine.org/

or2ywtool: OR2YW Tool (Version 0.0.16) [Python]. (n.d.). Retrieved from https://github.com/LanLi2017/OR2YWTool