Name	Hatim Sawai
UID No.	2021300108

Experiment 5

HONOR PLEDGE	I hereby declare that the documentation, code & output attached with this lab experiment has been completed by me in accordance with the highest standards of honesty. I confirm that I have not plagiarized OR used unauthorized materials OR given or recieved illegitimate help for completing this experiment. I will uphold equity & honesty in the evaluation of my work & if found guilty of plagiarism or dishonesty, will bear consequences as outlined in the 'integrity' section of the lab rubrics. I am doing so to maintain a community built around this code of honor'. Name: Hatim Sawai Sign:
PROBLEM STATEMENT	**Data Quality Assurance** 1. Pick a time-series dataset from either of the websites based on batch majority 2. Assess the quality of this dataset on all the dimensions of quality as identified by you. 3. Identify and handle inconsistent or erroneous data 4. Calculate data quality metrics for each data quality dimension for your dataset
THEORY	**Q. What are the 6 core dimensions of data quality? Elaborate on each with an example. Is there any other dimension that you can think of apart from these 6 for measuring quality?** Ans. The six core dimensions of data quality are: 1. **Accuracy**: It is the degree to which data correctly represents the real-world scenario. For example, a dataset containing the age of people should not contain any negative values. 2. **Completeness**: It is the degree to which all the required data is present. For example, a dataset containing the age of people should not contain any missing values. 3. **Consistency**: It is the degree to which data is consistent within itself. For example, a dataset containing the age of people should not contain any inconsistent values like a person with age 200 years.

	4. **Timeliness**: It is the degree to which data is available within the required time frame. For example, a dataset containing the age of people should not contain any future dates. 5. **Uniqueness**: It is the degree to which data is unique. For example, a dataset containing the age of people should not contain any duplicate values. 6. **Validity**: It is the degree to which data conforms to the defined schema. For example, a dataset containing the age of people should not contain any non-numeric values. Apart from these 6 dimensions, another dimension that can be used for measuring quality is **Reliability**. It is the degree to which data is reliable. For example, a dataset containing the age of people should not contain any values that are not reliable.

Importing Libraries & Dataset

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
# read csv file
df = pd.read_csv('../Datasets/Flight_Schedule.csv', low_memory=False)
df.head()
```

Out[ ]:

	airline	flightNumber	origin	destination	daysOfV
0	GoAir	425	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Fr
1	GoAir	423	Delhi	Hyderabad	Satu
2	GoAir	423	Delhi	Hyderabad	F
3	GoAir	423	Delhi	Hyderabad	F
4	GoAir	423	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Satu

1. Assesing the quality of the dataset

1. Accuracy

- The dataset is a goverment dataset and is expected to be accurate. However, we can check for accuracy by checking if the airlines are correct.

2. Completeness

- The dataset has many null/missing values. We can check for completeness by checking the number of missing values. Roughly 25% of the data in some columns is missing.

3. Consistency

- The dataset appears to be consistent by the fact that all the time and date values have the same and correct format across all the different datetime columns.

4. Timeliness

- The dataset is appearing to be timely as it is from 2018 to 2023, which is within 5 yrs of the current date. The dataset being from the goverment is expected to be timely.

5. Uniqueness

- The dataset is mostly containing unique values. However, we can check for uniqueness by checking for duplicate values. There appear to be some duplicate flight numbers with the same info in the dataset. Uniqueness for the full set is there but for some subsets it does not hold.

6. Validity

- The dataset is valid as it conforms to the defined schema. However, we can check for validity by checking if the data types are correct and if the data is in the correct format. The data types are correct and the data is in the correct format as can be seen from the data types and the data itself.

2. Identifying and Handling Inconsistent or Erroneous Data

In [ ]:
```
# drop timezone column as its the same as validTo
df.drop(["timezone"], axis=1, inplace=True)
df.head()
```

Out[ ]:

	airline	flightNumber	origin	destination	daysOfV
0	GoAir	425	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Fr
1	GoAir	423	Delhi	Hyderabad	Satu
2	GoAir	423	Delhi	Hyderabad	F
3	GoAir	423	Delhi	Hyderabad	F
4	GoAir	423	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Satu

In [ ]:
```
# print rows with more than 2 missing values
print((df.isnull().sum(axis=1) >= 3).sum())
```

```
0
```

In [ ]:
```
# drop duplicate rows
df.drop_duplicates(inplace=True)

# drop rows with more than 2 missing values
df.dropna(thresh=8, inplace=True)

# reset index
df.reset_index(drop=True, inplace=True)
```

In [ ]:
```
# for the daysOfWeek column, if a column has all the days then replace it with 'Dai
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: x.lower())
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: 'daily' if 'sunday' in x and 'm

# for the daysOfWeek column, if a column has 6 days then replace it with "All excep
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: "all except " + [i for i in ['s
df.head()
```

Out[ ]:

	airline	flightNumber	origin	destination	daysOfWeek	scheduledDepartureTime	sched
0	GoAir	425	Delhi	Hyderabad	daily	05:45	
1	GoAir	423	Delhi	Hyderabad	saturday	07:30	
2	GoAir	423	Delhi	Hyderabad	friday	07:30	
3	GoAir	423	Delhi	Hyderabad	friday	07:30	
4	GoAir	423	Delhi	Hyderabad	all except friday	07:30	

3. Calculating Data Quality Metrics

In [ ]:
```
# assess the accuracy of the data
df.info()
missing = df.isnull().sum()
dups = df.duplicated().sum()
uns = df.nunique()

print("\nSummary of the data:")
print("Missing values: ", missing)
print("Duplicates: ", dups)
print("Unique values: ", uns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88982 entries, 0 to 88981
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   airline                88307 non-null  object
 1   flightNumber           88659 non-null  object
 2   origin                 88482 non-null  object
 3   destination            88435 non-null  object
 4   daysOfWeek             88981 non-null  object
 5   scheduledDepartureTime  58296 non-null  object
 6   scheduledArrivalTime    58014 non-null  object
 7   timezone               88982 non-null  object
 8   validFrom              88982 non-null  object
 9   validTo                88982 non-null  object
 10  lastUpdated            88982 non-null  object
dtypes: object(11)
memory usage: 7.5+ MB

Summary of the data:
Missing values:  airline                     675
flightNumber              323
origin                    500
destination               547
daysOfWeek                  1
scheduledDepartureTime    30686
scheduledArrivalTime      30968
timezone                    0
validFrom                   0
validTo                     0
lastUpdated                 0
dtype: int64
Duplicates:  1084
Unique values:  airline                     14
flightNumber              4877
origin                     112
destination                112
daysOfWeek                 126
scheduledDepartureTime     288
scheduledArrivalTime       288
timezone                  1255
validFrom                 1266
validTo                   1255
lastUpdated                  1
dtype: int64
```

3.1. Dimension: Accuracy

In [ ]:
```python
def check_accuracy(df):
    valid_airlines = ['GoAir', 'Air India', 'AirAsia India', 'Jet Airways','Allianc
    valid_origins = ['Delhi', 'Lucknow', 'Kochi', 'Ahmedabad', 'Jaipur', 'Bengaluru
       'Guwahati', 'Goa', 'Kolkata', 'Hyderabad', 'Nagpur', 'Bagdogra',
       'MIHAN', 'Mumbai', 'Leh', 'Patna', 'Ranchi', 'Pune', 'Jammu',
       'Srinagar', 'Chennai', 'Bhubaneswar', 'Port Blair', 'Chandigarh',
       'Visakhapatnam', 'Vijayawada', 'Tirupati', 'Varanasi',
```

```
        'Aurangabad', 'Rajkot', 'Amritsar', 'Imphal', 'Jodhpur', 'Indore',
        'Vadodara', 'Raipur', 'Udaipur', 'Surat', 'Bhopal', 'Gaya',
        'Khajuraho', 'Thiruvananthapuram', 'Mangalore', 'Calicut', 'Hubli',
        'Coimbatore', 'Jamnagar', 'Nanded', 'Aizwal', 'Dibrugarh',
        'Madurai', 'Agra', 'Dimapur', 'Silchar', 'Agartala', 'Nasik',
        'Dehradun', 'Allahabad', 'Jorhat', 'Tiruchirappalli',
        'Kolhapur', 'Shirdi Airport', 'Kullu', 'Gorakhpur', 'Ludhiana',
        'Shimla', 'Gwalior', 'Pantnagar', 'Bikaner', 'Bathinda',
        'Jabalpur', 'Pathankot', 'Kangra', 'Agatti', 'Diu', 'Bhavnagar',
        'Kandla', 'Belgaum', 'Lilabari', 'Tezpur', 'Passighat', 'Shillong',
        'Rajahmundry', 'Tuticorin', 'Vidyanagar', 'Jalgaon', 'Keshod',
        'Porbandar', 'Salem', 'Mysore', 'Pondicherry', 'Adampur',
        'Jaisalmer', 'Kanpur', 'Kishangarh', 'Pakyong', 'Tezu',
        'Kannur International Airport', 'Bhuj', 'Kadapa', 'Jharsuguda',
        'Pithoragarh', 'Kalaburgi (Gulbarga)', 'Bidar', 'Thoise', 'Rupsi',
        'Durgapur', 'Darbhanga', 'Bilaspur', 'Mundra', 'Hindon Airport',
        'Cooch-Behar']
    valid_destinations = ['Hyderabad', 'Delhi', 'Varanasi', 'Bhubaneswar', 'Nagpur'
        'Bagdogra', 'MIHAN', 'Lucknow', 'Kochi', 'Mumbai', 'Leh', 'Patna',
        'Port Blair', 'Kannur International Airport', 'Ahmedabad',
        'Jaipur', 'Bengaluru', 'Ranchi', 'Chandigarh', 'Pune', 'Guwahati',
        'Chennai', 'Jammu', 'Goa', 'Kolkata', 'Srinagar',
        'Thiruvananthapuram', 'Mangalore', 'Calicut', 'Aurangabad',
        'Madurai', 'Rajkot', 'Gaya', 'Dimapur', 'Visakhapatnam',
        'Amritsar', 'Imphal', 'Agra', 'Jodhpur', 'Vijayawada', 'Indore',
        'Vadodara', 'Silchar', 'Raipur', 'Udaipur', 'Tirupati', 'Belgaum',
        'Hubli', 'Aizwal', 'Surat', 'Coimbatore', 'Khajuraho', 'Bhopal',
        'Dibrugarh', 'Agartala', 'Nanded', 'Jamnagar', 'Nasik',
        'Dehradun', 'Bhuj', 'Allahabad', 'Jorhat', 'Tiruchirappalli',
        'Thoise', 'Shirdi Airport', 'Kandla', 'Kullu', 'Passighat',
        'Shillong', 'Kolhapur', 'Rajahmundry', 'Diu', 'Gorakhpur',
        'Ludhiana', 'Shimla', 'Gwalior', 'Pantnagar', 'Agatti', 'Lilabari',
        'Tezpur', 'Bikaner', 'Bathinda', 'Jabalpur', 'Pathankot',
        'Bhavnagar', 'Kangra', 'Tuticorin', 'Kadapa', 'Salem', 'Jaisalmer',
        'Vidyanagar', 'Adampur', 'Pondicherry', 'Kanpur', 'Kishangarh',
        'Pakyong', 'Jalgaon', 'Mundra', 'Tezu', 'Jharsuguda', 'Mysore',
        'Darbhanga', 'Pithoragarh', 'Kalaburgi (Gulbarga)', 'Keshod',
        'Bidar', 'Hindon Airport', 'Rupsi', 'Bilaspur', 'Cooch-Behar',
        'Porbandar', 'Durgapur']
    accuracy_checks = (
        df["airline"].isin(valid_airlines)
        & df["origin"].isin(valid_origins)
        & df["destination"].isin(valid_destinations)
    )
    return accuracy_checks
```

In [ ]:

```
# Calculate accuracy percentage
df["accuracy"] = check_accuracy(df)
accuracy_percentage = (df["accuracy"].sum() / len(df)) * 100
print(f"Accuracy Percentage: {accuracy_percentage:.2f}%")
```

```
Accuracy Percentage: 98.10%
```

3.2. Dimension: Completeness

In [ ]:
```
def check_completeness(df):
    completeness_checks = ~df[['flightNumber', 'origin', 'destination', 'daysOfWeek
                               'scheduledDepartureTime', 'scheduledArrivalTime',
                               'timezone', 'validFrom', 'validTo', 'lastUpdated']].
    return completeness_checks

df['completeness'] = check_completeness(df)
completeness_percentage = (df["completeness"].sum() / len(df)) * 100
print(f"Completeness Percentage: {completeness_percentage:.2f}%")
```

```
Completeness Percentage: 30.57%
```

3.3. Dimension: Consistency

In [ ]:
```
def check_consistency(df):
    # Check consistency in the timezone column
    consistency_checks = pd.to_datetime(df['timezone'], errors='coerce').notna()

    # Check consistency in the daysOfWeek column
    days_of_week_validity = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday'
    consistency_checks &= df['daysOfWeek'].apply(lambda x: all(day.lower() in days_
    return consistency_checks

df["consistency"] = check_consistency(df)
consistency_percentage = (df["consistency"].sum() / len(df)) * 100
print(f"Consistency Percentage: {consistency_percentage:.2f}%")
```

```
Consistency Percentage: 100.00%
```

3.4. Dimension: Timeliness

In [ ]:
```
def check_timeliness(df):
    # Check if validFrom and validTo dates are within a reasonable time frame
    timeliness_checks = (
        pd.to_datetime(df["validFrom"], errors="coerce")
        <= pd.to_datetime(df["validTo"], errors="coerce")
    ) & (pd.to_datetime(df["validTo"], errors="coerce") <= datetime.now())
    # Check if lastUpdated date is recent
    timeliness_checks &= (
        pd.to_datetime(df["lastUpdated"], errors="coerce") <= datetime.now()
    )
    return timeliness_checks

df["timeliness"] = check_timeliness(df)
Timeliness_percentage = (df["timeliness"].sum() / len(df)) * 100
print(f"Timeliness Percentage: {Timeliness_percentage}%")
```

```
Timeliness Percentage: 99.99887617720438%
```

3.5. Dimension: Uniqueness

In [ ]:
```
def check_reliability(df):
    # Check reliability by examining consistency of data for the same flight
    reliability_checks = df.duplicated(
        subset=["flightNumber", "origin", "destination"], keep=False
```

```
    )
    # Calculate the percentage of reliability
    unique_percentage = (reliability_checks.sum() / len(df)) * 100
    return unique_percentage

# Assuming df is your DataFrame
unique_percentage = check_reliability(df)
print(f"Uniqueness Percentage: {unique_percentage:.2f}%")
```

```
Uniqueness Percentage: 91.89%
```

3.6. Dimension: Validity

In [ ]:
```
def check_validity(df):
    # Check validity of scheduledDepartureTime and scheduledArrivalTime
    time_format = '%H:%M'
    validity_checks = (
        pd.to_datetime(df['scheduledDepartureTime'], format=time_format, errors='co
        pd.to_datetime(df['scheduledArrivalTime'], format=time_format, errors='coer
    )

    # Check validity of daysOfWeek
    days_of_week_validity = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday'
    validity_checks &= df['daysOfWeek'].apply(lambda x: all(day.lower() in days_of_

    return validity_checks
```

In [ ]:
```
df["validity"] = check_validity(df)
Validity_percentage = (df["validity"].sum() / len(df)) * 100
print(f"Validity Percentage: {Validity_percentage:.2f}%")
```

```
Validity Percentage: 30.71%
```

2.7. Data Quality Metrics for the Dataset

In [ ]:
```
print(f"Accuracy Percentage: {accuracy_percentage:.2f}%")
print(f"Completness Percentage: {completeness_percentage:.2f}%")
print(f"Consistency Percentage: {consistency_percentage:.2f}%")
print(f"Timeliness Percentage: {Timeliness_percentage:.2f}%")
print(f"Uniqueness Percentage: {unique_percentage:.2f}%")
print(f"Validity Percentage: {Validity_percentage:.2f}%")
```

```
Accuracy Percentage: 98.10%
Completness Percentage: 30.57%
Consistency Percentage: 100.00%
Timeliness Percentage: 100.00%
Uniqueness Percentage: 91.89%
Validity Percentage: 30.71%
```

CONCLUSION	In this experiment we learned about the 6 core dimensions of data quality and calculated the data quality metrics for each data quality dimension for the dataset. We also identified and handled inconsistent or erroneous data.