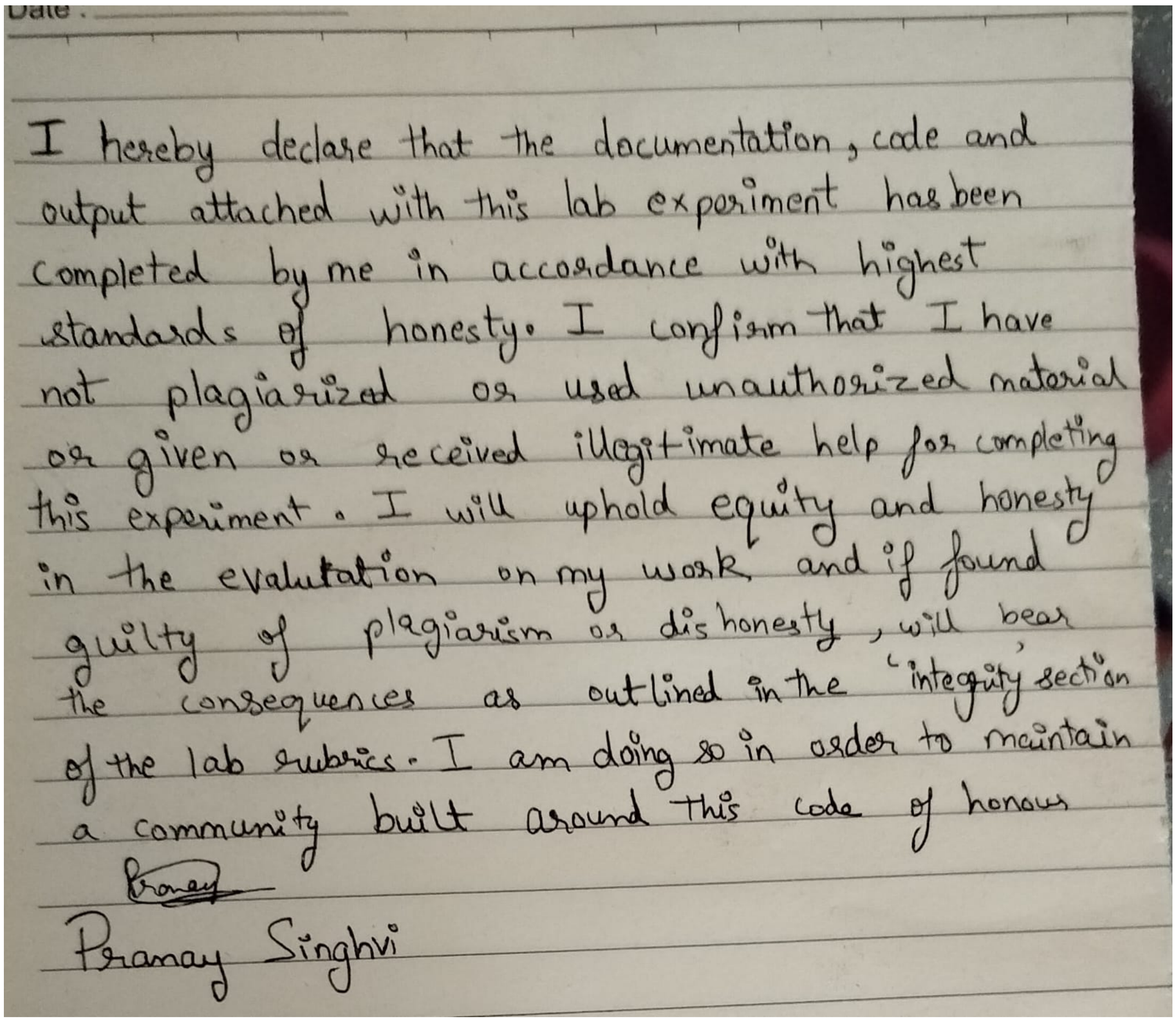
Experiment 5

Name	Pranay Singhvi
UID	2021300126

HONOR PLEDGE	

PROBLEM STATEMENT	**Data Quality Assurance :** Answer the following question first: 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? ● Pick a time-series dataset from either of these websites based on batch majority: 1) https://data.gov.in 2) https://data.oecd.org 3) Kaggle ● Assess the quality of this dataset on all the dimensions of quality as identified by you. ● Identify and handle inconsistent or erroneous data ● 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** 1. Accuracy: Definition: Refers to the correctness of the data. Accurate data is free from errors and represents the real-world values it is supposed to portray. Example: In a database tracking patient information, accurate data would mean that each patient's age, weight, and other details are recorded correctly without any mistakes. 2. Completeness: Definition: Indicates whether all required data is present. Incomplete data can lead to gaps in analysis and decision-making. Example: In an online order system, completeness would mean that all necessary fields (e.g., customer name, address, product details) are filled out for every order. 3. Consistency: Definition: Ensures uniformity and agreement in data across various databases or data sources. Example: In a multinational corporation's database, consistency would be maintained if the currency format is the same across all financial records, regardless of the country of origin. 4. Timeliness: Definition: Reflects the relevance and currency of the data. Timely data is up-to-date and aligns with the required time frame for decision-making. Example: In a stock market analysis system, timely data is crucial to accurately assess the current state of the market and make informed investment decisions. 5. Validity: Definition: Refers to the conformity of data to the defined rules or constraints. Valid data adheres to the specified formats and standards. Example: In a customer database, validity would ensure that email addresses follow a standard format and adhere to any domain restrictions. 6. Reliability: Definition: Indicates the trustworthiness and consistency of data over time. Reliable data can be depended upon for making decisions and conducting analyses. Example: In a scientific research database, reliable data would have consistent results when experiments are repeated under similar conditions.

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.

1.Import Library

In [ ]:
```
import pandas as pd
from datetime import datetime
import numpy as np
```

In [ ]:
```
df = pd.read_csv('Flight_Schedule.csv')
df
```

```
/var/folders/fc/b_3ntmtx3kv_43ckpdzjd47r0000gn/T/ipykernel_88074/4254981502.py:1: DtypeWarning: Columns (5) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('Flight_Schedule.csv')
```

Out[ ]:

	airline	flightNumber	origin	destination	daysOfWeek	scheduledDepartureTime	scheduledArrivalTime	timezone	validFrom	validTo	lastUpdated
0	GoAir	425	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Frida...	05:45	NaN	2019-03-30	2018-10-28	2019-03-30	2023-11-05
1	GoAir	423	Delhi	Hyderabad	Saturday	07:30	NaN	2018-10-28	2018-10-28	2018-10-28	2023-11-05
2	GoAir	423	Delhi	Hyderabad	Friday	07:30	NaN	2018-12-01	2018-11-03	2018-12-01	2023-11-05
3	GoAir	423	Delhi	Hyderabad	Friday	07:30	NaN	2019-03-30	2019-02-02	2019-03-30	2023-11-05
4	GoAir	423	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Saturday	07:30	NaN	2018-11-30	2018-10-29	2018-11-30	2023-11-05
...	...	...	...	...	...	...	...	...	...	...	...
88977	SpiceJet	2923	Hyderabad	Shirdi Airport	Sunday,Tuesday,Thursday,Saturday	NaN	16:35	2023-10-28	2023-03-26	2023-10-28	2023-11-05
88978	SpiceJet	2923	Tirupati	Shirdi Airport	Monday,Wednesday,Friday	NaN	16:50	2023-10-28	2023-03-26	2023-10-28	2023-11-05
88979	SpiceJet	0329	Chennai	Shirdi Airport	Sunday,Monday,Tuesday,Wednesday,Thursday,Frida...	NaN	16:20	2023-10-28	2023-03-26	2023-10-28	2023-11-05
88980	SpiceJet	2950	Delhi	Kangra	Sunday,Monday,Tuesday,Wednesday,Thursday,Frida...	NaN	07:30	2023-10-28	2023-03-26	2023-10-28	2023-11-05
88981	SpiceJet	2345	Delhi	Kangra	Sunday,Monday,Tuesday,Wednesday,Thursday,Frida...	NaN	10:00	2023-10-28	2023-03-26	2023-10-28	2023-11-05

88982 rows × 11 columns

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

Out[ ]:

	airline	flightNumber	origin	destination	daysOfWeek	scheduledDepartureTime	scheduledArrivalTime	validFrom	validTo	lastUpdated
0	GoAir	425	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Frida...	05:45	NaN	2018-10-28	2019-03-30	2023-11-05
1	GoAir	423	Delhi	Hyderabad	Saturday	07:30	NaN	2018-10-28	2018-10-28	2023-11-05
2	GoAir	423	Delhi	Hyderabad	Friday	07:30	NaN	2018-11-03	2018-12-01	2023-11-05
3	GoAir	423	Delhi	Hyderabad	Friday	07:30	NaN	2019-02-02	2019-03-30	2023-11-05
4	GoAir	423	Delhi	Hyderabad	Sunday,Monday,Tuesday,Wednesday,Thursday,Saturday	07:30	NaN	2018-10-29	2018-11-30	2023-11-05

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

```
41
```

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 'Daily'
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: x.lower())
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: 'daily' if 'sunday' in x and 'monday' in x and 'tuesday' in x and 'wednesday' in x and 'thursday' in x and 'friday' in x and 'saturday'

# for the daysOfWeek column, if a column has 6 days then replace it with "All except {day}" day being the one not in the list
df['daysOfWeek'] = df['daysOfWeek'].apply(lambda x: "all except " + [i for i in ['sunday', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday'] if i not in x][0] if len(x.s
df.head()
```

Out[ ]:

	airline	flightNumber	origin	destination	daysOfWeek	scheduledDepartureTime	scheduledArrivalTime	validFrom	validTo	lastUpdated
0	GoAir	425	Delhi	Hyderabad	daily	05:45	NaN	2018-10-28	2019-03-30	2023-11-05
1	GoAir	423	Delhi	Hyderabad	saturday	07:30	NaN	2018-10-28	2018-10-28	2023-11-05
2	GoAir	423	Delhi	Hyderabad	friday	07:30	NaN	2018-11-03	2018-12-01	2023-11-05
3	GoAir	423	Delhi	Hyderabad	friday	07:30	NaN	2019-02-02	2019-03-30	2023-11-05
4	GoAir	423	Delhi	Hyderabad	all except friday	07:30	NaN	2018-10-29	2018-11-30	2023-11-05

Check Data Quality Metrics

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
airline                   642
flightNumber              307
origin                    484
destination               521
daysOfWeek                  0
scheduledDepartureTime    30387
scheduledArrivalTime      30697
validFrom                   0
validTo                     0
lastUpdated                 0
dtype: int64
```

In [ ]:
```
def check_accuracy(df):
    # Check if categorical columns have accurate and valid values
    valid_airlines = ['GoAir', 'Air India', 'AirAsia India', 'Jet Airways','Alliance Air (India)', 'Jetlite', 'Vistara', 'IndiGo', 'TruJet', 'SpiceJet', 'FlyBig', 'Star Air', 'Akasa Air',
    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']  # Add valid origin locations if needed
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']  # Add valid destination locations if needed

    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: 98.13%
```

In [ ]:
```
def check_completeness(df):
    # Check for missing values in essential columns
    completeness_checks = ~df[['flightNumber', 'origin', 'destination', 'daysOfWeek',
                               'scheduledDepartureTime', 'scheduledArrivalTime', 'validFrom', 'validTo', 'lastUpdated']].isnull().any(axis=1)
    return completeness_checks
```

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

```
Completness Percentage: 30.33%
```

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

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

    return consistency_checks
```

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

```
Consistency Percentage: 44.91%
```

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
```

In [ ]:
```
df['timeliness'] = check_timeliness(df)
Timeliness_percentage = (df['timeliness'].sum() / len(df)) * 100
print(f"Timeliness Percentage: {Timeliness_percentage:.2f}%")
```

```
Timeliness Percentage: 100.00%
```

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='coerce').notna() &
        pd.to_datetime(df['scheduledArrivalTime'], format=time_format, errors='coerce').notna()
    )

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

    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: 8.41%
```

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
    reliability_percentage = (reliability_checks.sum() / len(df)) * 100

    return reliability_percentage
```

In [ ]:
```
reliability_percentage = check_reliability(df)
print(f"Reliability Percentage: {reliability_percentage:.2f}%")
```

```
Reliability Percentage: 91.66%
```

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

```
Reliability Percentage: 91.66%
Consistency Percentage: 44.91%
Accuracy Percentage: 98.13%
Timeliness Percentage: 100.00%
Completness Percentage: 30.33%
Validity Percentage: 8.41%
```

CONCLUSION	In this experiment, I conducted a comprehensive assessment of a flight schedule dataset, evaluating dimensions like accuracy, completeness, consistency, timeliness, and validity. Utilizing Python and pandas, I implemented data quality metrics, identifying and handling errors. The analysis highlighted areas for improvement in data quality, aiding future enhancements.