# COVID-19 Data Analysis Notebook Documentation

## Overview

This Jupyter Notebook performs exploratory data analysis on a COVID-19 dataset containing information about confirmed cases, deaths, and recoveries by region as of April 29, 2020. The dataset is a smaller subset for educational purposes, while the original contains approximately 19,000 rows.

## Dataset Information

- **Source**: Kaggle
- **File**: `Covid_19_Dataset.csv`
- **Data Period**: Up to April 29, 2020
- **Columns**: Region, Confirmed, Deaths, Recovered
- **Purpose**: Educational analysis of COVID-19 statistics

---

## Section 1: Data Loading and Initial Setup

### 1.1 Import Required Libraries

python

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

**Purpose**: Import essential libraries for data manipulation, visualization, and statistical analysis.

### 1.2 Load the Dataset

python

```
data = pd.read_csv(r"Covid_19_Dataset.csv")
```

**Purpose**: Load the COVID-19 dataset from CSV file into a pandas DataFrame.

### 1.3 Display the Dataset

python

```
data
```

**Expected Output**: Complete DataFrame showing all rows and columns with Region, Confirmed, Deaths, and Recovered cases.

# Section 2: Data Quality Assessment

## 2.1 Check Data Completeness

python

```
data.count()
```

**Purpose**: Count non-null values in each column to assess data completeness.

**Expected Output**:

```
Region       XXX
Confirmed    XXX
Deaths       XXX
Recovered    XXX
dtype: int64
```

## 2.2 Identify Missing Values

python

```
data.isnull().sum()
```

**Purpose**: Count null/missing values in each column.

**Expected Output**:

```
Region       0
Confirmed    0
Deaths       X
Recovered    X
dtype: int64
```

## 2.3 Visualize Missing Data Pattern

python

```
sns.heatmap(data.isnull())
plt.show()
```

**Purpose**: Create a heatmap visualization to identify patterns in missing data.

**Expected Output**: A heatmap where:

- White areas represent missing values

- Colored areas represent present values
- Helps identify systematic patterns in missing data

---

## Section 3: Analysis Questions and Solutions

### Q1: Show the number of Confirmed, Deaths and Recovered cases in each Region

**Code Implementation:**

```python
# Display first 2 rows for reference
data.head(2)

# Group by Region and sum Confirmed and Recovered cases
data.groupby('Region')[['Confirmed', 'Recovered']].sum()
```

**Alternative Approaches:**

```python
# Sum all numeric columns by region
data.groupby('Region').sum().head(20)

# Get top 10 regions by confirmed cases
data.groupby('Region')['Confirmed'].sum().sort_values(ascending=False).head(10)
```

**Expected Output**: DataFrame showing total confirmed and recovered cases for each region.

### Q2: Remove all records where Confirmed Cases is Less Than 10

**Code Implementation:**

```python
# Remove records with confirmed cases < 10
data = data[~(data.Confirmed < 10)]
```

**Explanation:**

- `data.Confirmed < 10` creates a boolean mask
- `~` operator negates the condition (keeps records with >= 10 cases)
- Updates the original DataFrame

**Expected Output**: Filtered DataFrame with only records having 10 or more confirmed cases.

## Q3: In which Region, maximum number of Confirmed cases were recorded?

**Code Implementation:**

```python
data.groupby('Region').Confirmed.sum().sort_values(ascending=False).head(20)
```

**Purpose**: Identify regions with highest total confirmed cases.

**Expected Output**: Series showing regions ranked by total confirmed cases (descending order).

## Q4: In which Region, minimum number of Deaths cases were recorded?

**Code Implementation:**

```python
data.groupby('Region').Deaths.sum().sort_values(ascending=True).head(50)
```

**Purpose**: Identify regions with lowest total death cases.

**Expected Output**: Series showing regions ranked by total deaths (ascending order).

## Q5: How many Confirmed, Deaths & Recovered cases were reported from India till 29 April 2020?

**Code Implementation:**

```python
# Filter data for India
data[data.Region == 'India']

# Examples for other countries
data[data.Region == 'Yemen']
data[data.Region == 'US']
```

**Purpose**: Extract specific country data for detailed analysis.

**Expected Output**: DataFrame rows containing all records for the specified country.

## Q6-A: Sort the entire data w.r.t No. of Confirmed cases in ascending order

**Code Implementation:**

python

```
data.sort_values(by=['Confirmed'], ascending=True).head(50)
```

**Purpose**: Arrange data from lowest to highest confirmed cases.

**Expected Output**: DataFrame sorted by confirmed cases (ascending), showing regions with fewest cases first.

### Q6-B: Sort the entire data w.r.t No. of Recovered cases in descending order

**Code Implementation:**

python

```
data.sort_values(by=['Recovered'], ascending=False).head(50)
```

**Purpose**: Arrange data from highest to lowest recovered cases.

**Expected Output**: DataFrame sorted by recovered cases (descending), showing regions with most recoveries first.

---

## Key Pandas Operations Used

### 1. Data Filtering

python

```
# Boolean indexing
data[data.Confirmed >= 10]
data[data.Region == 'India']

# Negation operator
data[~(data.Confirmed < 10)]
```

### 2. Grouping and Aggregation

python

```
# Group by single column
data.groupby('Region').sum()

# Group by with specific columns
data.groupby('Region')[['Confirmed', 'Recovered']].sum()

# Group with single column aggregation
data.groupby('Region').Confirmed.sum()
```

### 3. Sorting

python

```
# Sort by single column
data.sort_values(by=['Confirmed'], ascending=True)

# Sort aggregated results
data.groupby('Region').Confirmed.sum().sort_values(ascending=False)
```

### 4. Data Inspection

python

```
# Display first n rows
data.head(n)

# Check data types and info
data.info()

# Statistical summary
data.describe()
```

---

## Expected Insights

1. **Regional Distribution**: Identification of most and least affected regions
2. **Data Quality**: Assessment of missing values and data completeness
3. **Filtering Impact**: Understanding how data filtering affects analysis results
4. **Country-Specific Analysis**: Detailed view of individual country statistics
5. **Ranking Analysis**: Comparative analysis of regions based on different metrics

---

## Best Practices Demonstrated

1. **Data Validation**: Checking for missing values before analysis
2. **Data Cleaning**: Removing low-quality records (< 10 confirmed cases)
3. **Exploratory Analysis**: Using multiple approaches to understand data
4. **Visualization**: Using heatmaps for missing data patterns
5. **Comparative Analysis**: Sorting and ranking for insights

---

## Limitations and Considerations

1. **Temporal Scope**: Data only up to April 29, 2020
2. **Sample Size**: Subset of original 19,000-row dataset
3. **Data Currency**: Analysis reflects early pandemic period
4. **Missing Data**: Some regions may have incomplete reporting
5. **Reporting Variations**: Different countries may have different reporting standards

---

## Next Steps for Extended Analysis

1. **Time Series Analysis**: Analyze trends over time
2. **Mortality Rate Calculation**: Calculate death rates by region
3. **Recovery Rate Analysis**: Analyze recovery patterns
4. **Correlation Analysis**: Examine relationships between variables
5. **Predictive Modeling**: Develop forecasting models
6. **Comparative Studies**: Compare with current data for longitudinal analysis