5.2.4 Explore the Data in Pandas

Often you'll be given more than one CSV file to work with and be asked to merge them to perform the analysis. Before you merge them, you should explore the data.

Missing, malformed, or incorrect data could lead to a poor or incorrect analysis, and the last thing we want is to be standing in front of the CEO with faulty information!

We'll start off by inspecting each DataFrame. When we're satisfied that there is no missing, malformed, or incorrect data, we'll merge the DataFrames.

Inspect the City Data DataFrame

For the `city_data_df` DataFrame, we need to:

1. Get all the rows that contain null values.
2. Make sure the driver_count column has an integer data type.
3. Find out how many data points there are for each type of city.

First, let's get all the rows that are not null.

REWIND

To get the name of each column and the number of rows that are not null, we can use the `df.count()` method.

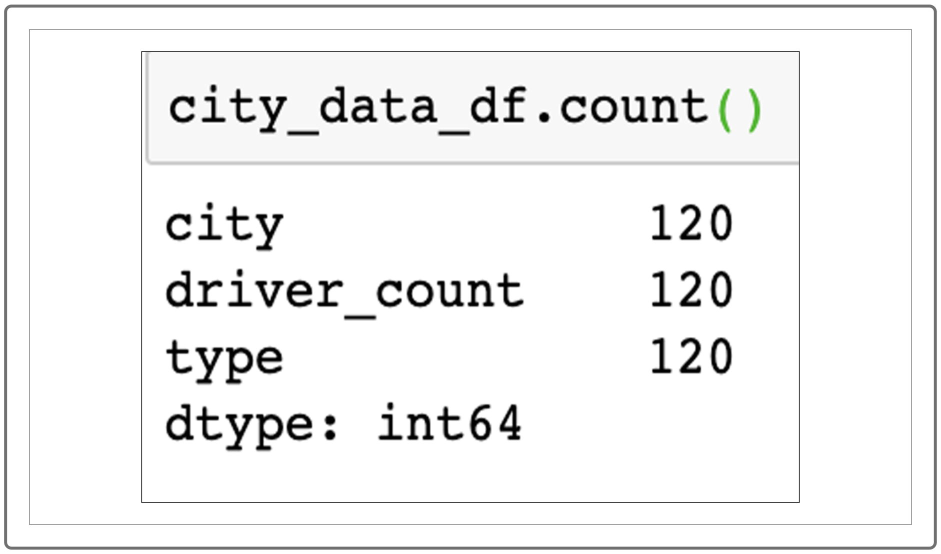
Another option is to use `df.isnull().sum()` method chaining.

We'll use the `df.count()` method to find the names of our columns and the number of rows that are not null.

Add the following code to a new cell:

```
# Get the columns and the rows that are not null.
city_data_df.count()
```

When you run the cell, the output shows 120 rows for each column:



```
city_data_df.count()
```

```
city            120
driver_count    120
type            120
dtype: int64
```

And to make sure there are no null values, we can type and run the following code:

```
# Get the columns and the rows that are not null.
city_data_df.isnull().sum()
```

The output from this code shows that there are zero null values in all three columns.

Next, we need to see if the driver_count column has a numerical data type because we plan to perform mathematical calculations on that column.

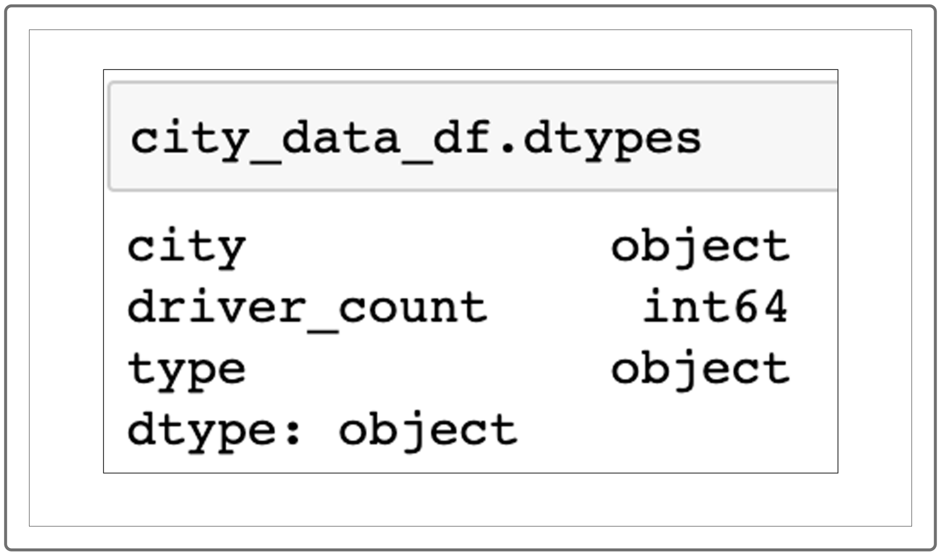
REWIND

To get the data types of each column, we use the `dtypes` on the DataFrame.

Add the following code to a new cell:

```
# Get the data types of each column.
city_data_df.dtypes
```

Run the cell, and you'll see that the output shows the data types for each column. Most importantly, the data type for the driver_count column is an integer:



```
city_data_df.dtypes

city               object
driver_count        int64
type               object
dtype: object
```

Finally, we'll check to see how many data points there are for each type of city. To do this, we'll use the `sum()` method on the `city_data_df` for the type column where the condition equals each city in the DataFrame.

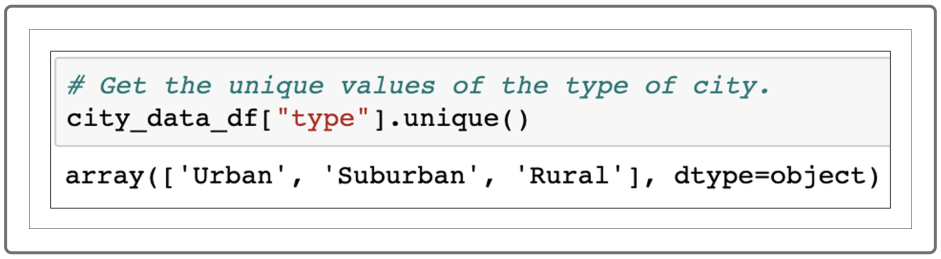
REWIND

We can use the `unique()` method on a specific column, which will return an array, or list, of all the unique values of that column

Add the following code to a new cell:

```
# Get the unique values of the type of city.
city_data_df["type"].unique()
```

Run the cell, and you'll see that the output will be an array of different types of cities.



```
# Get the unique values of the type of city.
city_data_df["type"].unique()
```

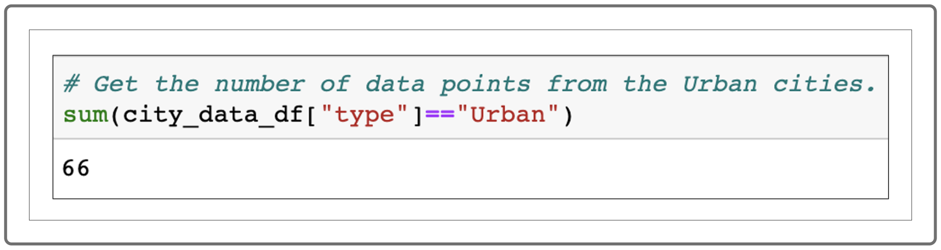
```
array(['Urban', 'Suburban', 'Rural'], dtype=object)
```

Now we can use the `sum()` method on the `city_data_df` for the type column where the condition equals either Urban, Suburban, or Rural.

To get the number of data points for the Urban cities, add the following code to a new cell:

```
# Get the number of data points from the Urban cities.
sum(city_data_df["type"]=="Urban")
```

When you run this cell, the output will show that the number of data points for the Urban cities is 66.



```
# Get the number of data points from the Urban cities.
sum(city_data_df["type"]=="Urban")
```

```
66
```

Inspect Ride Data DataFrame

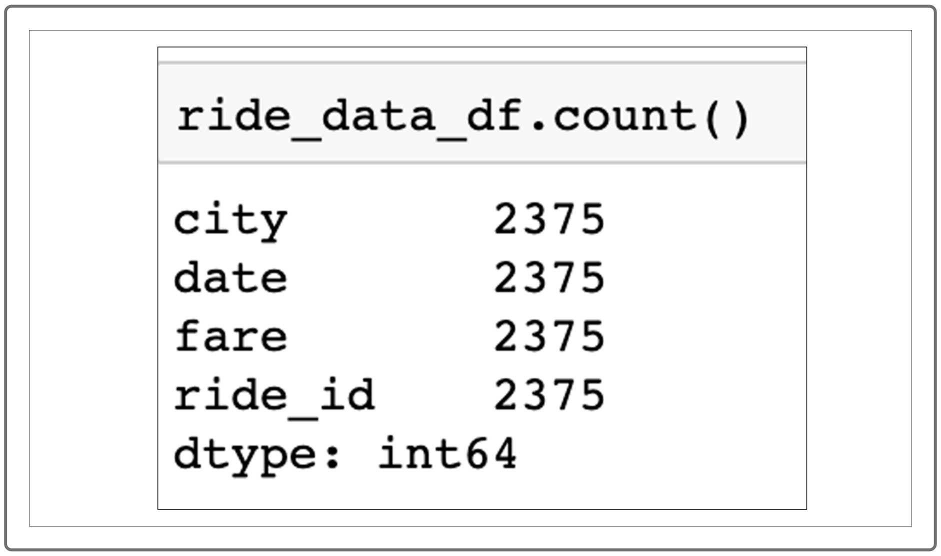
For the `ride_data_df` DataFrame, we need to:

1. Get all the rows that contain null values.
2. Make sure the fare and ride_id columns are numerical data types.

First, let's get all the rows that are not null. Add the following code:

```
# Get the columns and the rows that are not null.
ride_data_df.count()
```

When you run the cell, you'll see that the output is 2,375 rows for each column, as shown here:



```
ride_data_df.count()

city        2375
date        2375
fare        2375
ride_id     2375
dtype: int64
```

And to make sure there are no null values, we can type and run the following code:

```
# Get the columns and the rows that are not null.
ride_data_df.isnull().sum()
```

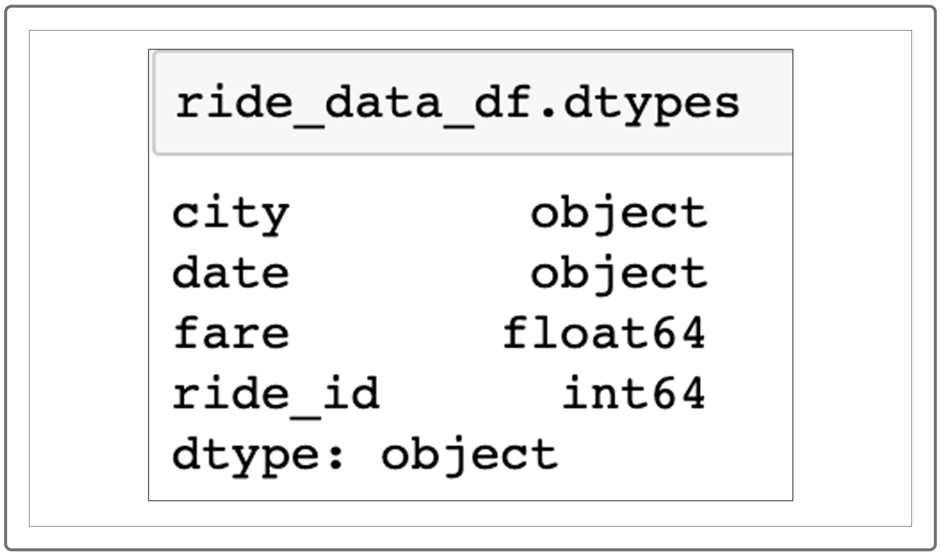
The output from this code shows that there are zero null values in all three columns.

Next, we need to determine if the fare and ride_id columns are numerical data types so that we can perform mathematical calculations on those columns.

Add the following code to a new cell:

```
# Get the data types of each column.
ride_data_df.dtypes
```

Run the cell. The output shows the data types for each column. And more importantly, the data types for the fare and ride_id columns are floating-point decimal, and integer, respectively.



```
ride_data_df.dtypes
```

```
city           object
date           object
fare          float64
ride_id         int64
dtype: object
```

Both of the DataFrames look good and can now be merged.

Merge DataFrames

Before we merge the DataFrames, let's review each DataFrame.

The columns in the `city_data_df` DataFrame are:

- city
- driver_count
- type

The columns in the `ride_data_df` are:

- city
- date
- fare
- ride_id

REWIND

When we merge two DataFrames, we merge on a column with the same data, and the same column name, in both DataFrames. We use the following syntax to do that:

`new_df = pd.merge(leftdf, rightdf, on=["column_leftdf", "column_rightdf"])`

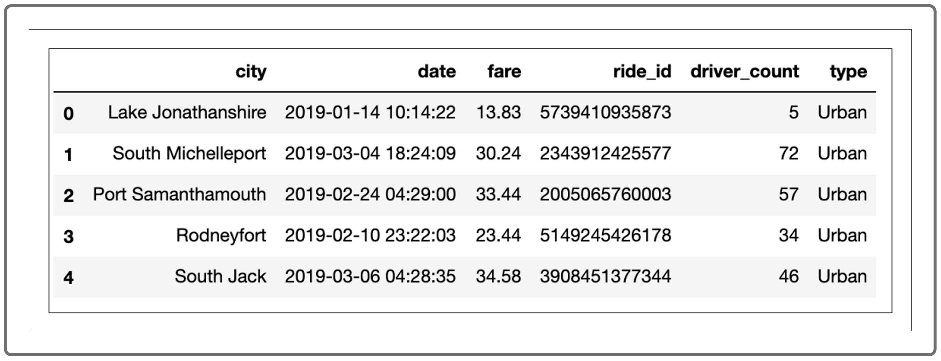
We may have to merge the DataFrames using the `how=` parameter either left, right, inner, or outer depending how we want to merge the DataFrames. The default is inner.

Looking at the columns in the two DataFrames, we can see that the column the DataFrames have in common is `city`. Therefore, we will merge the two DataFrames on the `city` column, and then add the `city_data_df` to the end of the `ride_data_df` DataFrame with the constraint `how="left"`.

Add the following code to a new cell and run the cell to merge the two DataFrames.

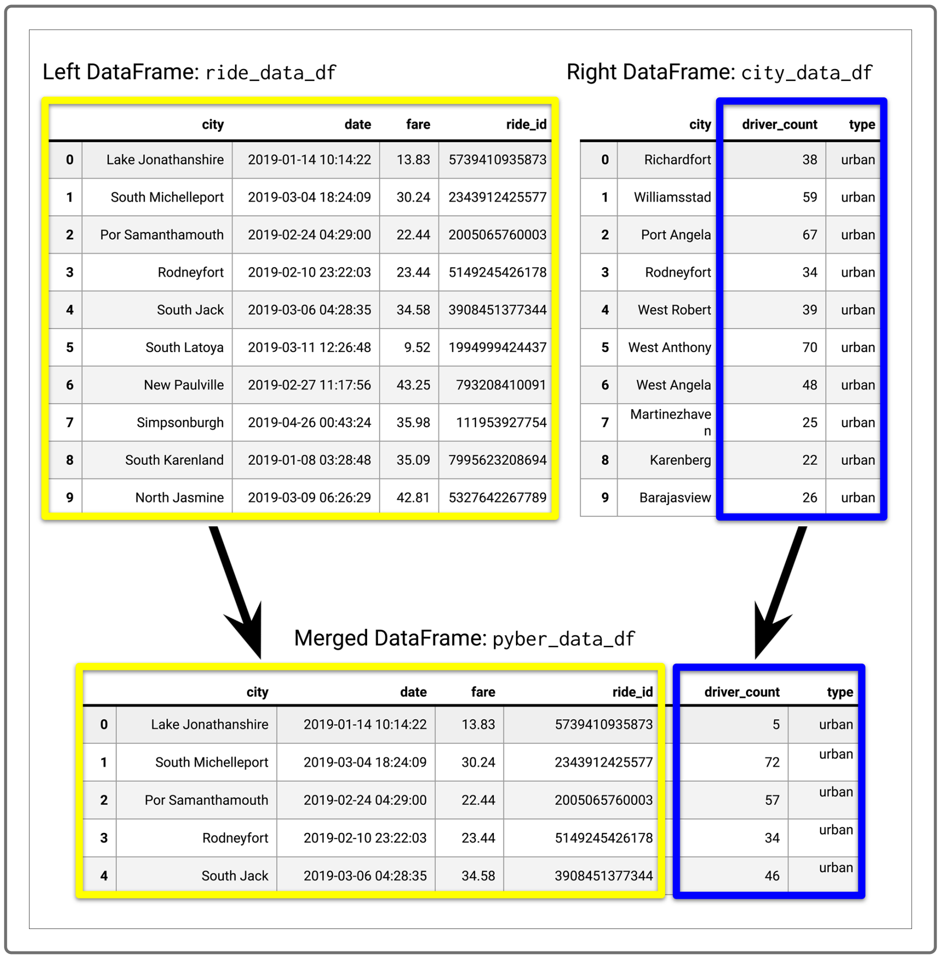
```
# Combine the data into a single dataset
pyber_data_df = pd.merge(ride_data_df, city_data_df, how="left", on=["city",

# Display the DataFrame
pyber_data_df.head()
```

The merged DataFrame, `pyber_data_df`, should look like this:



	city	date	fare	ride_id	driver_count	type
0	Lake Jonathanshire	2019-01-14 10:14:22	13.83	5739410935873	5	Urban
1	South Michelleport	2019-03-04 18:24:09	30.24	2343912425577	72	Urban
2	Port Samanthamouth	2019-02-24 04:29:00	33.44	2005065760003	57	Urban
3	Rodneyfort	2019-02-10 23:22:03	23.44	5149245426178	34	Urban
4	South Jack	2019-03-06 04:28:35	34.58	3908451377344	46	Urban

In the `pyber_data_df` DataFrame, all the columns from the `city_data_df` are the first four columns after the index. The driver_count and type columns from the `ride_data_df` are added at the end, as shown in the following image:



Left DataFrame: `ride_data_df`

	city	date	fare	ride_id
0	Lake Jonathanshire	2019-01-14 10:14:22	13.83	5739410935873
1	South Michelleport	2019-03-04 18:24:09	30.24	2343912425577
2	Por Samanthamouth	2019-02-24 04:29:00	22.44	2005065760003
3	Rodneyfort	2019-02-10 23:22:03	23.44	5149245426178
4	South Jack	2019-03-06 04:28:35	34.58	3908451377344
5	South Latoya	2019-03-11 12:26:48	9.52	1994999424437
6	New Paulville	2019-02-27 11:17:56	43.25	793208410091
7	Simpsonburgh	2019-04-26 00:43:24	35.98	111953927754
8	South Karenland	2019-01-08 03:28:48	35.09	7995623208694
9	North Jasmine	2019-03-09 06:26:29	42.81	5327642267789

Right DataFrame: `city_data_df`

	city	driver_count	type
0	Richardfort	38	urban
1	Williamsstad	59	urban
2	Port Angela	67	urban
3	Rodneyfort	34	urban
4	West Robert	39	urban
5	West Anthony	70	urban
6	West Angela	48	urban
7	Martinezhaven	25	urban
8	Karenberg	22	urban
9	Barajasview	26	urban

Merged DataFrame: `pyber_data_df`

	city	date	fare	ride_id	driver_count	type
0	Lake Jonathanshire	2019-01-14 10:14:22	13.83	5739410935873	5	urban
1	South Michelleport	2019-03-04 18:24:09	30.24	2343912425577	72	urban
2	Por Samanthamouth	2019-02-24 04:29:00	22.44	2005065760003	57	urban
3	Rodneyfort	2019-02-10 23:22:03	23.44	5149245426178	34	urban
4	South Jack	2019-03-06 04:28:35	34.58	3908451377344	46	urban