```
In [3]: df.describe(['air_temp_9am']).show()
+-------+-----------------+
|summary|      air_temp_9am|
+-------+-----------------+
|  count|             1090|
|   mean| 64.93300141287075|
| stddev|11.175514003175877|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+
```

This says that there are 1090 rows. The total number of rows in the DataFrame is 1095:

```
In [4]: df.count()
Out[4]: 1095
```

This means that 5 of the rows in the *air_temp_9am* column are missing values.

Step 4. **Remove missing values.** We can drop all the rows missing a value in any calling using *na.drop():*

```
In [5]: removeAllDF = df.na.drop()
```

Let's look at the summary statistics for *air_temp_9am* with the missing values dropped:

```
In [6]: removeAllDF.describe(['air_temp_9am']).show()
+-------+-----------------+
|summary|      air_temp_9am|
+-------+-----------------+
|  count|             1064|
|   mean| 65.02260949558739|
| stddev|11.168033449415699|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+
```

We can see that the mean and standard deviation is close to the original values: mean is 64.933 vs. 65.022, and standard deviation is 11.175 vs. 11.168.

The count is 1064, which means that 1095 - 1064 = 31 rows were dropped. We can see this agrees with the total number of rows in the new DataFrame:

```
In [7]: removeAllDF.count()
Out[7]: 1064
```

Step 5. **Impute missing values.** Instead of removing rows containing missing values, let's replace the values with the mean value for that column. First, we'll load the *avg* function and make a copy of the original DataFrame:

```
In [8]: from pyspark.sql.functions import avg

        imputeDF = df
```

Next, we'll iterate through each column in the DataFrame: compute the mean value for that column and then replace any missing values in that column with the mean.

```
In [9]: for x in imputeDF.columns:
            meanValue = removeAllDF.agg(avg(x)).first()[0]
            print(x, meanValue)
            imputeDF = imputeDF.na.fill(meanValue, [x])
```

```
number 545.0018796992481
air_pressure_9am 918.9031798641055
air_temp_9am 65.02260949558739
avg_wind_direction_9am 142.30675564934032
avg_wind_speed_9am 5.485793050713691
max_wind_direction_9am 148.48042413321312
max_wind_speed_9am 6.9997136588756925
rain_accumulation_9am 0.18202347650615522
rain_duration_9am 266.3936973996038
relative_humidity_9am 34.07743985327712
relative_humidity_3pm 35.14838093290537
```

The *agg()* function performs an aggregate calculation on the DataFrame and *avg(x)* specifies to compute the mean on column *x*. The *agg()* function returns a DataFrame, *first()* returns the first *Row*, and *[0]* gets the first value.

The last line of code uses *na.fill()* to replace the missing values with the mean value (first argument) in column *x* (second argument).

The output of executing this cell prints the mean values for each column and we can see the mean value for *air_temp_9am* is the same as the mean when we removed all the missing values in step 4, *i.e.*, 65.022.

Step 6. **Print imputed data summary statistics.** Let's call *describe()* to show the summary statistics for the original and imputed *air_temp_9am:*

```
In [10]: df.describe(['air_temp_9am']).show()
         imputeDF.describe(['air_temp_9am']).show()
```

```
+-------+-----------------+
|summary|      air_temp_9am|
+-------+-----------------+
|  count|             1090|
|   mean| 64.93300141287075|
| stddev|11.175514003175877|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+

+-------+-----------------+
|summary|      air_temp_9am|
+-------+-----------------+
|  count|             1095|
|   mean| 64.93341058219822|
| stddev|11.149948199920226|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+
```

The count for the imputed data is larger since the 5 rows with missing data have replaced with real values. Additionally, we can see that the means are close, but not equal, and this is probably due to round-off error.