CODE:

```
rdd1 = sc.textFile("pulsar.dat")
rdd2 = rdd1.map(lambda x:x.split())
rdd3 = rdd2.map(lambda x: [float(x[0]) , float(x[1]) , float(x[2]) , float(x[3])])
```

 # Group signals based on (ascension, declination, frequency), rounding to the nearest integer

```
grouped_rdd = rdd3.map(lambda signal: ((round(signal[0], 0), round(signal[1], 0), round(signal[3], 0)),  #
(ascension, declination, frequency)signal[2]  # Keep the time for blip detection))
```

 # Group by (ascension, declination, frequency), and aggregate signal times

```
grouped_signals = grouped_rdd.groupByKey().mapValues(list)
```

 # Use lambda for periodic blip calculations

```
calculate_periodic_blips = lambda times: (None if len(times) < 2 else (round(sum([y - x for x, y in
zip(sorted(times)[:-1], sorted(times)[1:])]) / (len(times) - 1)), len(times)))
```

 # Apply the function to calculate the period and blip count

```
 blip_periods = grouped_signals.mapValues(calculate_periodic_blips).filter(lambda x: x[1] is not None)
```

 # Find the top 10 groups with the most blips

```
top_10_blips = blip_periods.takeOrdered(10, key=lambda x: -x[1][1])
```

 # Create an RDD to format the output as strings

```
 formatted_output = sc.parallelize(top_10_blips).map(lambda x: f"There are {x[1][1]} blips at location
{x[0][0]:.1f}, {x[0][1]:.1f} degrees with a frequency of {x[0][2]:.1f} MHz and a period of {x[1][0]} seconds.")
```

 # Collect and print the formatted results

```
 formatted_output.collect()
```

OUTPUT:

```
Out[13]:
['There are 18 blips at location 87.0, 68.0 degrees with a frequency of 4448.0 MHz and a period of 1 seconds.',
 'There are 18 blips at location 86.0, 68.0 degrees with a frequency of 4448.0 MHz and a period of 1 seconds.',
 'There are 16 blips at location 105.0, 111.0 degrees with a frequency of 3031.0 MHz and a period of 4 seconds.',
 'There are 15 blips at location 57.0, 115.0 degrees with a frequency of 3509.0 MHz and a period of 3 seconds.',
 'There are 14 blips at location 59.0, 58.0 degrees with a frequency of 3782.0 MHz and a period of 8 seconds.',
 'There are 13 blips at location 82.0, 96.0 degrees with a frequency of 4631.0 MHz and a period of 8 seconds.',
 'There are 12 blips at location 77.0, 89.0 degrees with a frequency of 8245.0 MHz and a period of 26 seconds.',
 'There are 10 blips at location 77.0, 89.0 degrees with a frequency of 8246.0 MHz and a period of 32 seconds.',
 'There are 8 blips at location 67.0, 58.0 degrees with a frequency of 3073.0 MHz and a period of 15 seconds.',
 'There are 8 blips at location 91.0, 94.0 degrees with a frequency of 6477.0 MHz and a period of 10 seconds.']
```

INFERENCE:

- Looking at this results there might be a possibility that the 18 blips at 87,68 and 18 blips at 86,68 all belong to a same cluster but diving further into analysis

- I looked at two clusters with the highest number of blips. The first cluster includes data points with ascension between 85 and 87 degrees, declination from 67 to 69 degrees, and frequency between 4447 and 4449 MHz.The second cluster has ascension between 103 and 105 degrees, declination between 110 and 112 degrees, and frequency from 3030 to 3032 MHz. We created a text file that contains all the data points within these ranges for further review.

For Signal With Frequency 4448MHz

CODE:

```
 import shutil
 import os

 # Path to the output file
output_path = "filtered_signal4448.txt"

# Remove the existing output directory if it exists
 if os.path.exists(output_path):
 shutil.rmtree(output_path)

 rdd1 = sc.textFile("pulsar.dat")
rdd2 = rdd1.map(lambda x: x.split())
 rdd3 = rdd2.map(lambda x: [float(x[0]), float(x[1]), float(x[2]), float(x[3])])

 # Group signals based on (ascension, declination, frequency), rounding to the nearest decimal
 grouped_rdd = rdd3.map(lambda signal: (
 (round(signal[0], 1), round(signal[1], 1), round(signal[3], 1)),  # (ascension, declination, frequency)
 signal[2]  # Keep the time for blip detection))

 # Filter for the specified conditions (ascension: 85-87, declination: 67-69, frequency: 4447-4449)
 filtered_signals = grouped_rdd.filter(lambda x: 85 <= x[0][0] <= 87 and 67 <= x[0][1] <= 69 and 4447 <=
x[0][2] <=    4449)

 # Sort the filtered signals based on time period (the second element)
 sorted_filtered_signals = filtered_signals.sortBy(lambda x: x[1])

  # Convert the sorted filtered RDD back to a format suitable for saving
 formatted_filtered_signals = sorted_filtered_signals.map(lambda x: f"Ascension: {x[0][0]}, Declination:
{x[0][1]},Frequency: {x[0][2]}, Time: {x[1]}")

  # Save the sorted filtered results as a text file
  formatted_filtered_signals.saveAsTextFile(output_path)
```

OUTPUT:

```
Ascension: 86.4, Declination: 68.1, Frequency: 4448.1, Time: 2799.9995391021644
Ascension: 86.5, Declination: 68.1, Frequency: 4448.0, Time: 2801.1015620642256
Ascension: 86.4, Declination: 68.0, Frequency: 4448.0, Time: 2802.1992164117432
Ascension: 86.4, Declination: 68.1, Frequency: 4448.3, Time: 2803.300508136213
Ascension: 86.5, Declination: 67.9, Frequency: 4448.0, Time: 2804.4002043765263
Ascension: 86.4, Declination: 68.2, Frequency: 4447.9, Time: 2805.4997770619098
Ascension: 86.4, Declination: 68.1, Frequency: 4448.0, Time: 2806.5988255784805
Ascension: 86.4, Declination: 68.1, Frequency: 4448.1, Time: 2807.6995732951605
Ascension: 86.4, Declination: 68.0, Frequency: 4448.0, Time: 2808.799192556669
Ascension: 86.3, Declination: 68.0, Frequency: 4447.9, Time: 2809.9014342844685
Ascension: 86.4, Declination: 68.1, Frequency: 4448.0, Time: 2810.999136556827
Ascension: 86.4, Declination: 68.2, Frequency: 4448.0, Time: 2812.1001139407
Ascension: 86.3, Declination: 68.1, Frequency: 4448.2, Time: 2813.199764930366
Ascension: 86.4, Declination: 68.1, Frequency: 4448.0, Time: 2814.3006596393866
Ascension: 86.4, Declination: 68.1, Frequency: 4448.0, Time: 2815.399459366674
Ascension: 86.4, Declination: 68.2, Frequency: 4448.2, Time: 2816.499450662511
Ascension: 86.4, Declination: 68.1, Frequency: 4448.1, Time: 2817.600667619794
Ascension: 86.4, Declination: 68.0, Frequency: 4448.1, Time: 2818.7000708226615
Ascension: 86.5, Declination: 68.2, Frequency: 4448.1, Time: 2819.8014903605313
Ascension: 86.7, Declination: 67.9, Frequency: 4447.8, Time: 2820.8997594082334
Ascension: 86.6, Declination: 68.1, Frequency: 4448.0, Time: 2822.0016885808036
Ascension: 86.6, Declination: 68.2, Frequency: 4447.9, Time: 2823.098658583156
Ascension: 86.7, Declination: 68.2, Frequency: 4448.1, Time: 2824.1994913737376
Ascension: 86.5, Declination: 68.3, Frequency: 4448.1, Time: 2825.2999328149576
Ascension: 86.6, Declination: 68.1, Frequency: 4448.0, Time: 2826.4001997684386
Ascension: 86.8, Declination: 68.1, Frequency: 4448.0, Time: 2827.501043401596
Ascension: 86.5, Declination: 68.1, Frequency: 4448.1, Time: 2828.6005715448173
Ascension: 86.6, Declination: 68.0, Frequency: 4448.1, Time: 2829.7011234664615
Ascension: 86.5, Declination: 68.4, Frequency: 4448.1, Time: 2830.8008827349418
Ascension: 86.5, Declination: 68.0, Frequency: 4448.0, Time: 2831.901136426528
Ascension: 86.5, Declination: 68.2, Frequency: 4448.1, Time: 2833.001017850788
Ascension: 86.5, Declination: 68.3, Frequency: 4447.9, Time: 2834.1005728218825
Ascension: 86.6, Declination: 68.1, Frequency: 4448.1, Time: 2835.200032651227
Ascension: 86.5, Declination: 68.3, Frequency: 4448.1, Time: 2836.3002041294308
Ascension: 86.6, Declination: 68.0, Frequency: 4448.0, Time: 2837.4010138701383
Ascension: 86.6, Declination: 68.0, Frequency: 4447.9, Time: 2838.500419586209
```

INFERENCE:

- From the output, we can see that for ascension values ranging from 86.3 to 86.7, with 86.5 as the center, all coordinates fall within three standard deviations. For declination, the values range from 67.9 to 68.4, with 68.1 as the center, and again, all coordinates are within three standard deviations. A similar pattern is observed for frequency.
- Since the data is sorted by time, we can conclude that all the data points are within a time period of 1.

- **This supports our assumption that all 36 blips are part of the same cluster.**

For Signal With Frequency 3030MHz

CODE:

```
import shutil
import os

# Path to the output file
output_path = "filtered_signal3030.txt"

# Remove the existing output directory if it exists
if os.path.exists(output_path):
shutil.rmtree(output_path)

rdd1 = sc.textFile("pulsar.dat")
rdd2 = rdd1.map(lambda x: x.split())
rdd3 = rdd2.map(lambda x: [float(x[0]), float(x[1]), float(x[2]), float(x[3])])

# Group signals based on (ascension, declination, frequency), rounding to the nearest decimal
grouped_rdd = rdd3.map(lambda signal: (
(round(signal[0], 1), round(signal[1], 1), round(signal[3], 1)),  # (ascension, declination, frequency)
signal[2]  # Keep the time for blip detection))

# Filter for the specified conditions (ascension: 103-105, declination: 110-112, frequency: 3030-3032)
filtered_signals = grouped_rdd.filter(lambda x: 103 <= x[0][0] <= 105 and 110 <= x[0][1] <= 112 and 3030
<= x[0][2] <=    3032)

# Sort the filtered signals based on time period (the second element)
sorted_filtered_signals = filtered_signals.sortBy(lambda x: x[1])

# Convert the sorted filtered RDD back to a format suitable for saving
formatted_filtered_signals = sorted_filtered_signals.map(lambda x: f"Ascension: {x[0][0]}, Declination:
{x[0][1]},Frequency: {x[0][2]}, Time: {x[1]}")

# Save the sorted filtered results as a text file
formatted_filtered_signals.saveAsTextFile(output_path)
```

OUTPUT:

filtered_signals3030.txt > ! part-00000

```
Ascension: 104.5, Declination: 111.4, Frequency: 3030.7, Time: 869.9995391021644
Ascension: 104.6, Declination: 111.4, Frequency: 3030.6, Time: 872.2015620642259
Ascension: 104.5, Declination: 111.3, Frequency: 3030.7, Time: 874.3992164117436
Ascension: 104.5, Declination: 111.4, Frequency: 3030.9, Time: 876.6005081362132
Ascension: 104.6, Declination: 111.2, Frequency: 3030.6, Time: 878.800204376527
Ascension: 104.4, Declination: 111.5, Frequency: 3030.5, Time: 880.9997770619104
Ascension: 104.5, Declination: 111.4, Frequency: 3030.6, Time: 883.1988255784812
Ascension: 104.4, Declination: 111.4, Frequency: 3030.7, Time: 885.3995732951612
Ascension: 104.7, Declination: 111.3, Frequency: 3030.7, Time: 887.5991925566699
Ascension: 104.6, Declination: 111.3, Frequency: 3030.6, Time: 889.8014342844698
Ascension: 104.4, Declination: 111.4, Frequency: 3030.6, Time: 891.9991365568286
Ascension: 104.5, Declination: 111.5, Frequency: 3030.7, Time: 894.2001139407015
Ascension: 104.6, Declination: 111.4, Frequency: 3030.8, Time: 896.399764930368
Ascension: 104.4, Declination: 111.4, Frequency: 3030.7, Time: 898.6006596393883
Ascension: 104.4, Declination: 111.4, Frequency: 3030.6, Time: 900.7994593666759
Ascension: 104.4, Declination: 111.5, Frequency: 3030.8, Time: 902.999450662513
Ascension: 104.5, Declination: 111.4, Frequency: 3030.7, Time: 905.200667619796
Ascension: 104.5, Declination: 111.3, Frequency: 3030.8, Time: 907.4000708226637
Ascension: 104.5, Declination: 111.5, Frequency: 3030.7, Time: 909.6014903605335
Ascension: 104.7, Declination: 111.2, Frequency: 3030.4, Time: 911.7997594082358
Ascension: 104.6, Declination: 111.4, Frequency: 3030.7, Time: 914.0016885808064
Ascension: 104.3, Declination: 111.5, Frequency: 3030.6, Time: 916.1986585831588
Ascension: 104.5, Declination: 111.5, Frequency: 3030.7, Time: 918.3994913737407
Ascension: 104.3, Declination: 111.6, Frequency: 3030.7, Time: 920.5999328149609
Ascension: 104.6, Declination: 111.4, Frequency: 3030.6, Time: 922.800199768442
Ascension: 104.5, Declination: 111.4, Frequency: 3030.7, Time: 925.0010434015992
Ascension: 104.5, Declination: 111.4, Frequency: 3030.7, Time: 927.2005715448211
Ascension: 104.5, Declination: 111.3, Frequency: 3030.7, Time: 929.401123466465
```

INFERENCE:

- From the output, we can see that for ascension values ranging from 104.3 to 104.7, with 104.5 as the center, all coordinates fall within three standard deviations. For declination, the values range from 111.2 to 111.6, with 111.4 as the center, and again, all coordinates are within three standard deviations. A similar pattern is observed for frequency.
- Since the data is sorted by time, we can conclude that all the data points are within a time period of 2.

CONCLUSION:

From the above codes I conclude that 36 blips at 4448MHz is the biggest cluster and second being in 3030MHz with 28 blips .