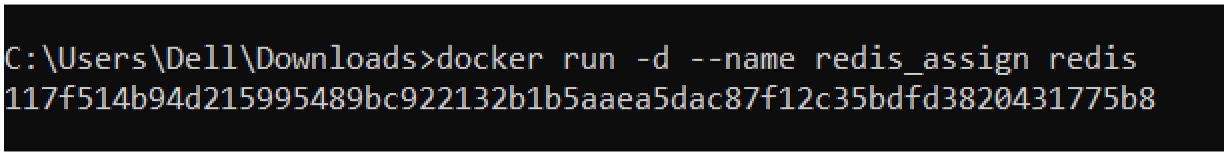
Redis – Assignment

Submitted by: Muhammad Amin Ghias

ERP: 25366

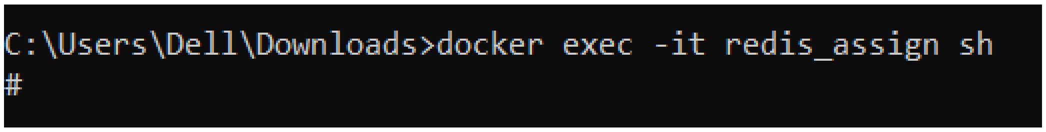
Commands

1) Creating container of redis and running it



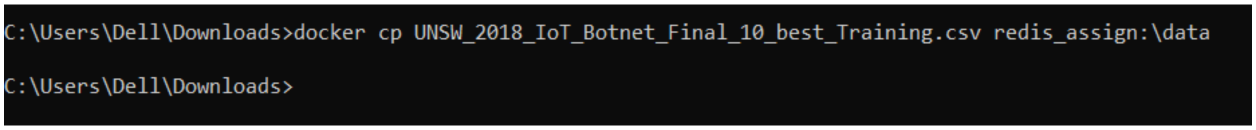
```
C:\Users\Dell\Downloads>docker run -d --name redis_assign redis
117f514b94d215995489bc922132b1b5aaea5dac87f12c35bdfd3820431775b8
```

2) Executing conatiner



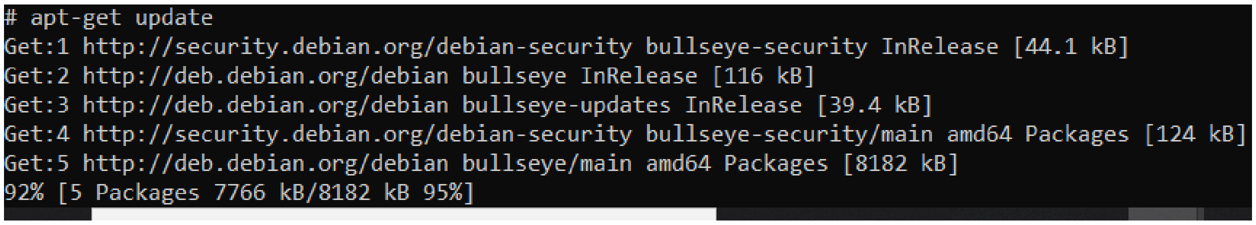
```
C:\Users\Dell\Downloads>docker exec -it redis_assign sh
#
```

3) Copying csv into the container (**the size of csv file is 360 MB)**



```
C:\Users\Dell\Downloads>docker cp UNSW_2018_IoT_Botnet_Final_10_best_Training.csv redis_assign:\data

C:\Users\Dell\Downloads>
```

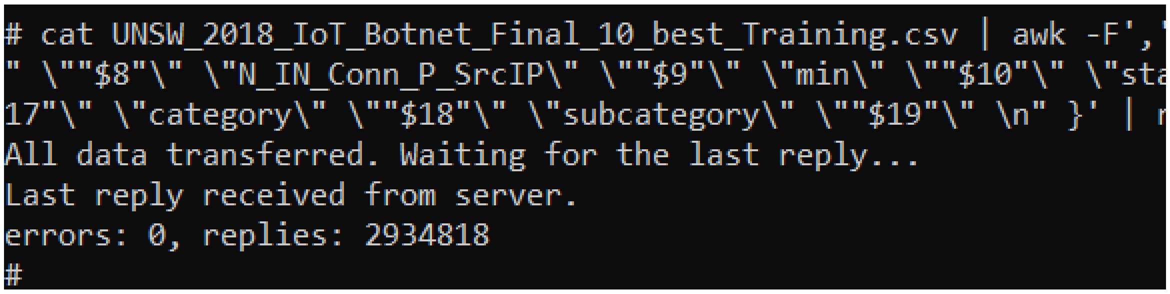
4) Updating the redis container shell



```
# apt-get update
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [44.1 kB]
Get:2 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [39.4 kB]
Get:4 http://security.debian.org/debian-security bullseye-security/main amd64 Packages [124 kB]
Get:5 http://deb.debian.org/debian bullseye/main amd64 Packages [8182 kB]
92% [5 Packages 7766 kB/8182 kB 95%]
```

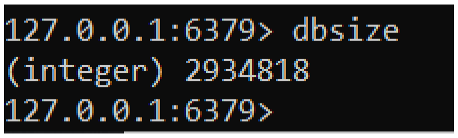
5) Importing the csv file as a hash

- **cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " HSET \""$1"\" \"proto\" \""$2"\" \"saddr\" \""$3"\" \"sport\" \""$4"\" \"daddr\" \""$5"\" \"dport\" \""$6"\" \"seq\" \""$7"\" \"stddev\" \""$8"\" \"N_IN_Conn_P_SrcIP\" \""$9"\" \"min\" \""$10"\" \"state_number\" \""$11"\" \"mean\" \""$12"\" \"N_IN_Conn_P_DstIP\" \""$13"\" \"drate\" \""$14"\" \"srate\" \""$15"\" \"max\" \""$16"\" \"attack\" \""$17"\" \"category\" \""$18"\" \"subcategory\" \""$19"\" \n" }' | redis-cli --pipe**



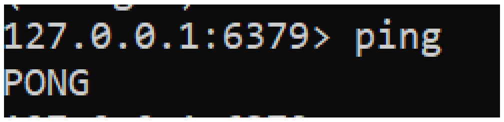
```
# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F','
" \""$8"\" \"N_IN_Conn_P_SrcIP\" \""$9"\" \"min\" \""$10"\" \"sta
17"\" \"category\" \""$18"\" \"subcategory\" \""$19"\" \n" }' | r
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 2934818
#
```

6) Running the redis db and checking dbsize



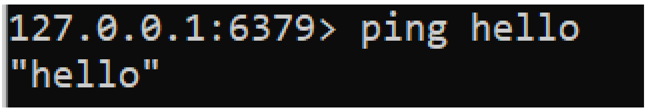
```
127.0.0.1:6379> dbsize
(integer) 2934818
127.0.0.1:6379>
```

7) Trying ping command



```
127.0.0.1:6379> ping
PONG
```

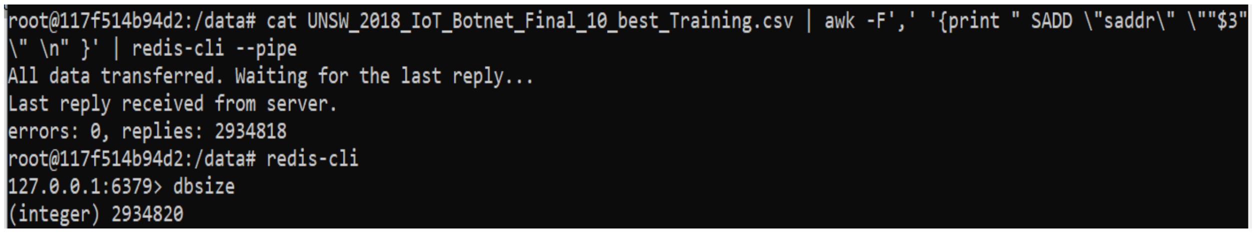
8) Getting response from ping



```
127.0.0.1:6379> ping hello
"hello"
```

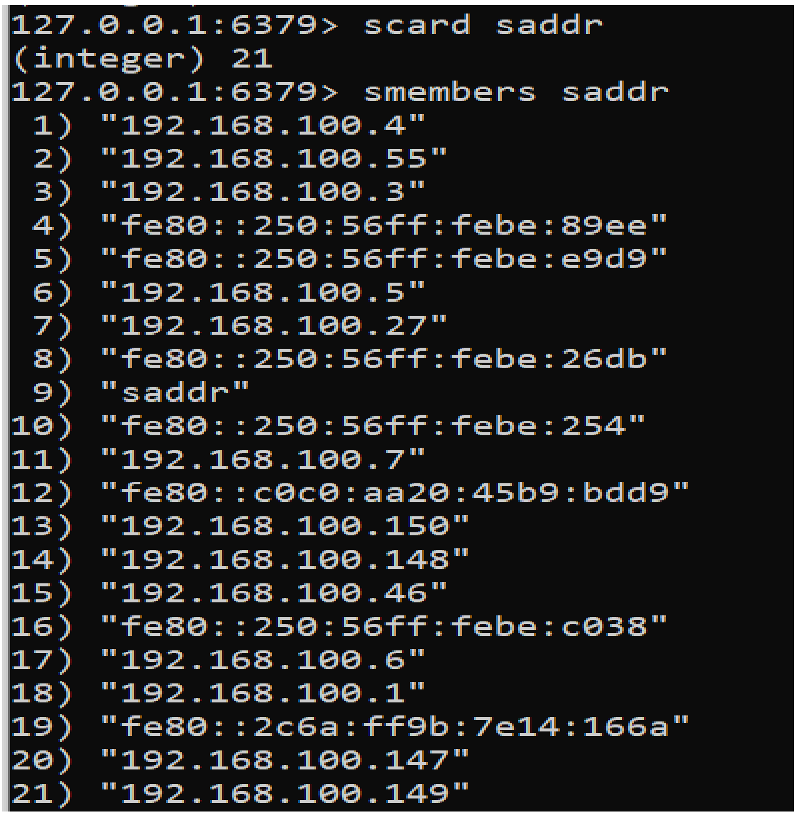
SETS

9) Loading csv column as a set



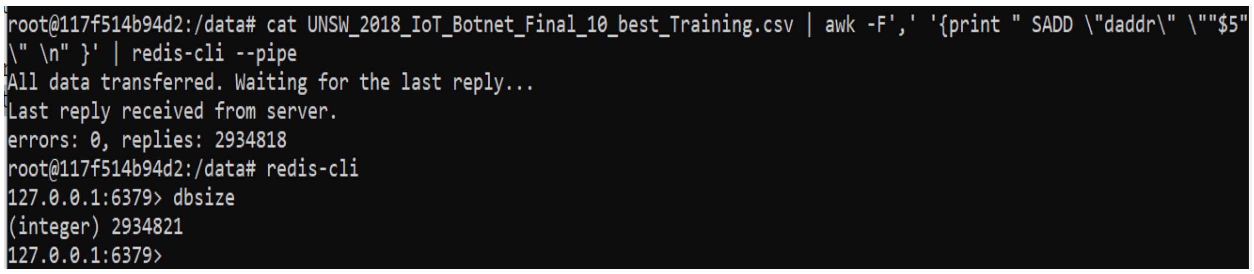
```
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " SADD \"saddr\" \""$3"
\" \n" }' | redis-cli --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 2934818
root@117f514b94d2:/data# redis-cli
127.0.0.1:6379> dbsize
(integer) 2934820
```

10) Checking the number of members and size of set

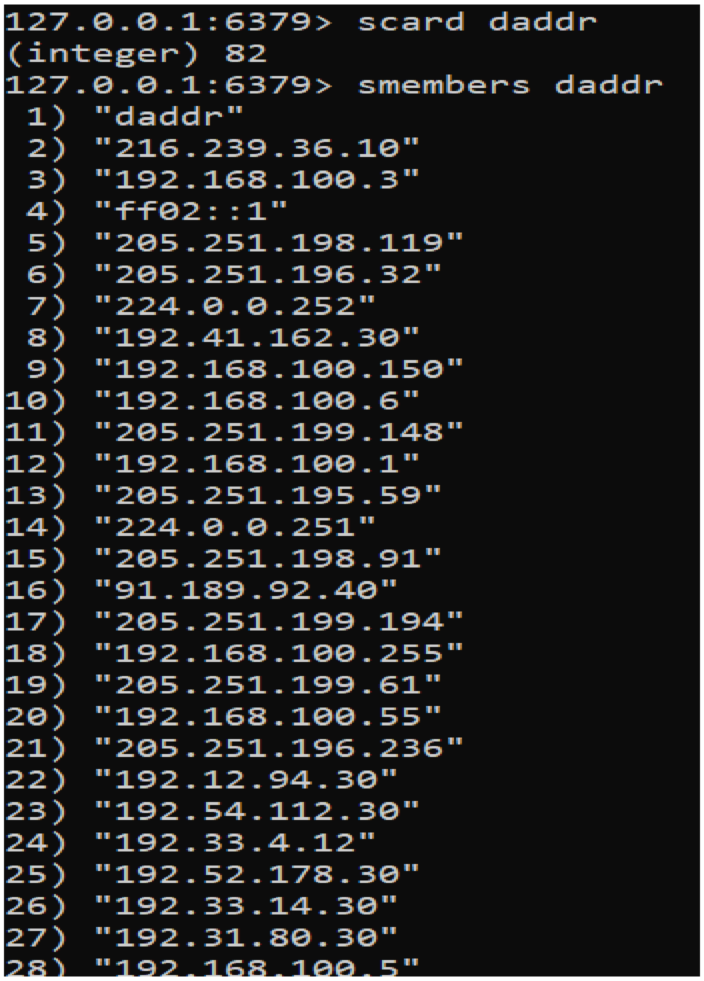


```
127.0.0.1:6379> scard saddr
(integer) 21
127.0.0.1:6379> smembers saddr
 1) "192.168.100.4"
 2) "192.168.100.55"
 3) "192.168.100.3"
 4) "fe80::250:56ff:febe:89ee"
 5) "fe80::250:56ff:febe:e9d9"
 6) "192.168.100.5"
 7) "192.168.100.27"
 8) "fe80::250:56ff:febe:26db"
 9) "saddr"
10) "fe80::250:56ff:febe:254"
11) "192.168.100.7"
12) "fe80::c0c0:aa20:45b9:bdd9"
13) "192.168.100.150"
14) "192.168.100.148"
15) "192.168.100.46"
16) "fe80::250:56ff:febe:c038"
17) "192.168.100.6"
18) "192.168.100.1"
19) "fe80::2c6a:ff9b:7e14:166a"
20) "192.168.100.147"
21) "192.168.100.149"
```

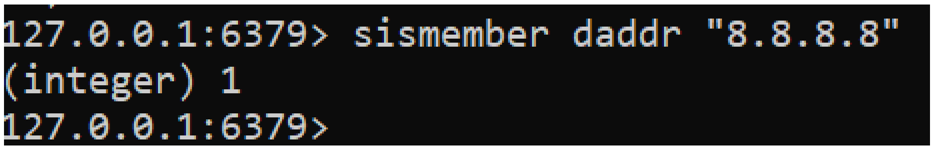
11) Loading csv column to make a second set


```
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " SADD \"daddr\" \""$5"
\" \n" }' | redis-cli --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 2934818
root@117f514b94d2:/data# redis-cli
127.0.0.1:6379> dbsize
(integer) 2934821
127.0.0.1:6379>
```

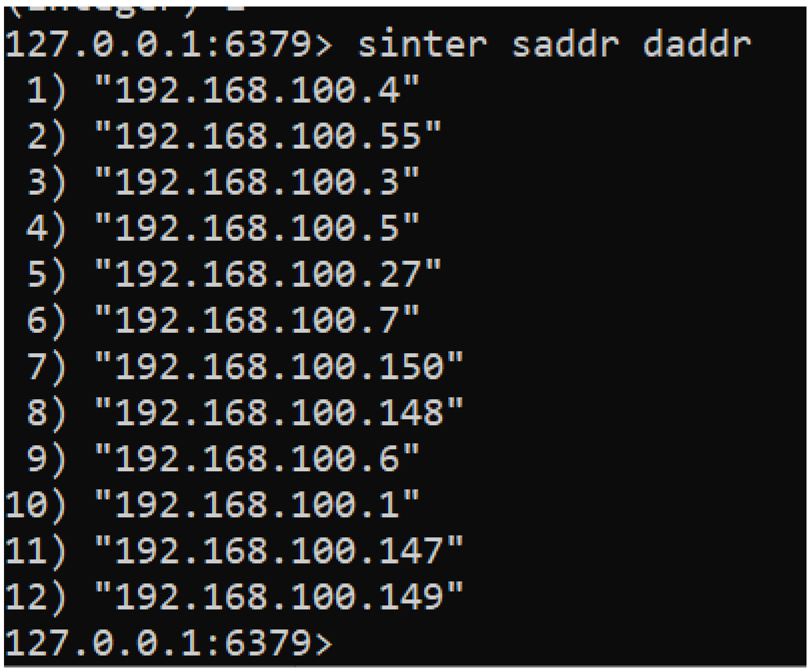
12) Checking the number of members and size of set


```
127.0.0.1:6379> scard daddr
(integer) 82
127.0.0.1:6379> smembers daddr
 1) "daddr"
 2) "216.239.36.10"
 3) "192.168.100.3"
 4) "ff02::1"
 5) "205.251.198.119"
 6) "205.251.196.32"
 7) "224.0.0.252"
 8) "192.41.162.30"
 9) "192.168.100.150"
10) "192.168.100.6"
11) "205.251.199.148"
12) "192.168.100.1"
13) "205.251.195.59"
14) "224.0.0.251"
15) "205.251.198.91"
16) "91.189.92.40"
17) "205.251.199.194"
18) "192.168.100.255"
19) "205.251.199.61"
20) "192.168.100.55"
21) "205.251.196.236"
22) "192.12.94.30"
23) "192.54.112.30"
24) "192.33.4.12"
25) "192.52.178.30"
26) "192.33.14.30"
27) "192.31.80.30"
28) "192.168.100.5"
```

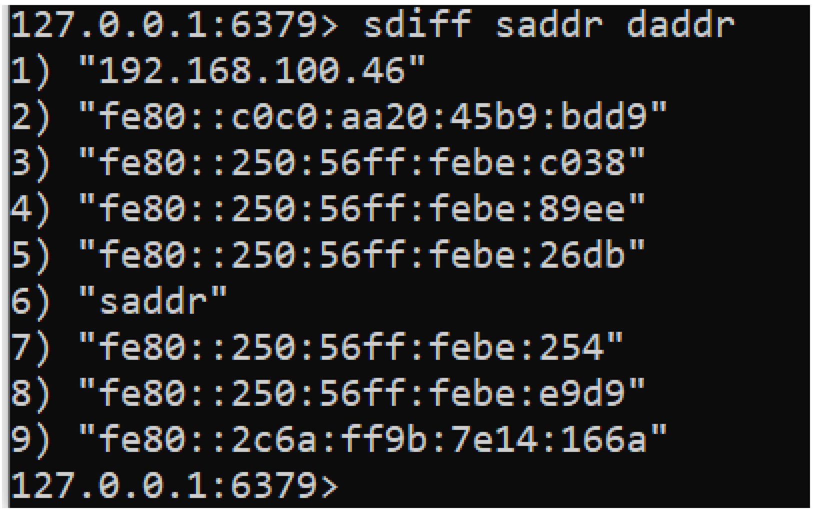
13) Checking if 8.8.8.8 is in set daddr


```
127.0.0.1:6379> sismember daddr "8.8.8.8"
(integer) 1
127.0.0.1:6379>
```

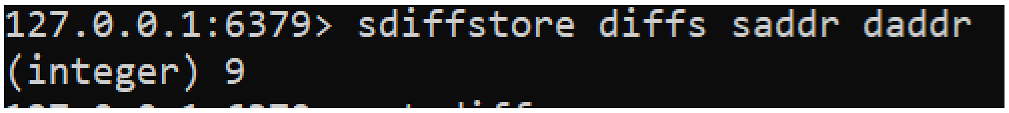
14) Doing intersection of 2 sets


```
127.0.0.1:6379> sinter saddr daddr
 1) "192.168.100.4"
 2) "192.168.100.55"
 3) "192.168.100.3"
 4) "192.168.100.5"
 5) "192.168.100.27"
 6) "192.168.100.7"
 7) "192.168.100.150"
 8) "192.168.100.148"
 9) "192.168.100.6"
10) "192.168.100.1"
11) "192.168.100.147"
12) "192.168.100.149"
127.0.0.1:6379>
```

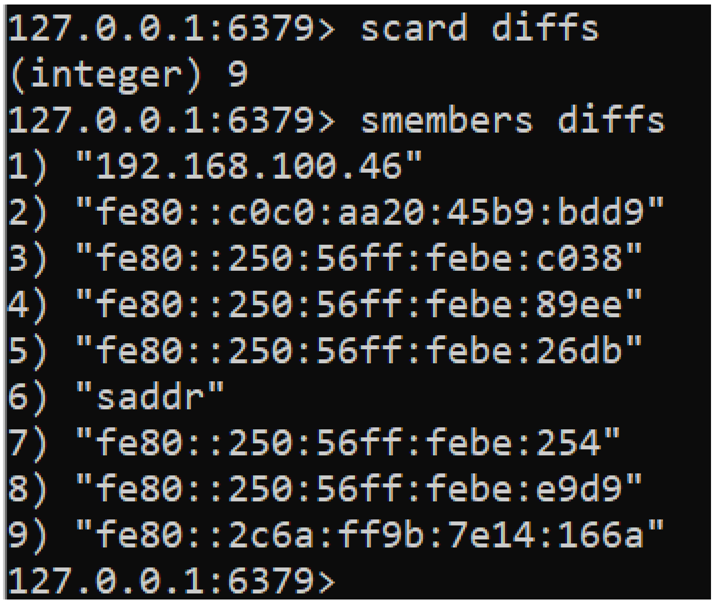
15) Taking difference of 2 sets


```
127.0.0.1:6379> sdiff saddr daddr
1) "192.168.100.46"
2) "fe80::c0c0:aa20:45b9:bdd9"
3) "fe80::250:56ff:febe:c038"
4) "fe80::250:56ff:febe:89ee"
5) "fe80::250:56ff:febe:26db"
6) "saddr"
7) "fe80::250:56ff:febe:254"
8) "fe80::250:56ff:febe:e9d9"
9) "fe80::2c6a:ff9b:7e14:166a"
127.0.0.1:6379>
```

16) Taking difference and storing result in a new key

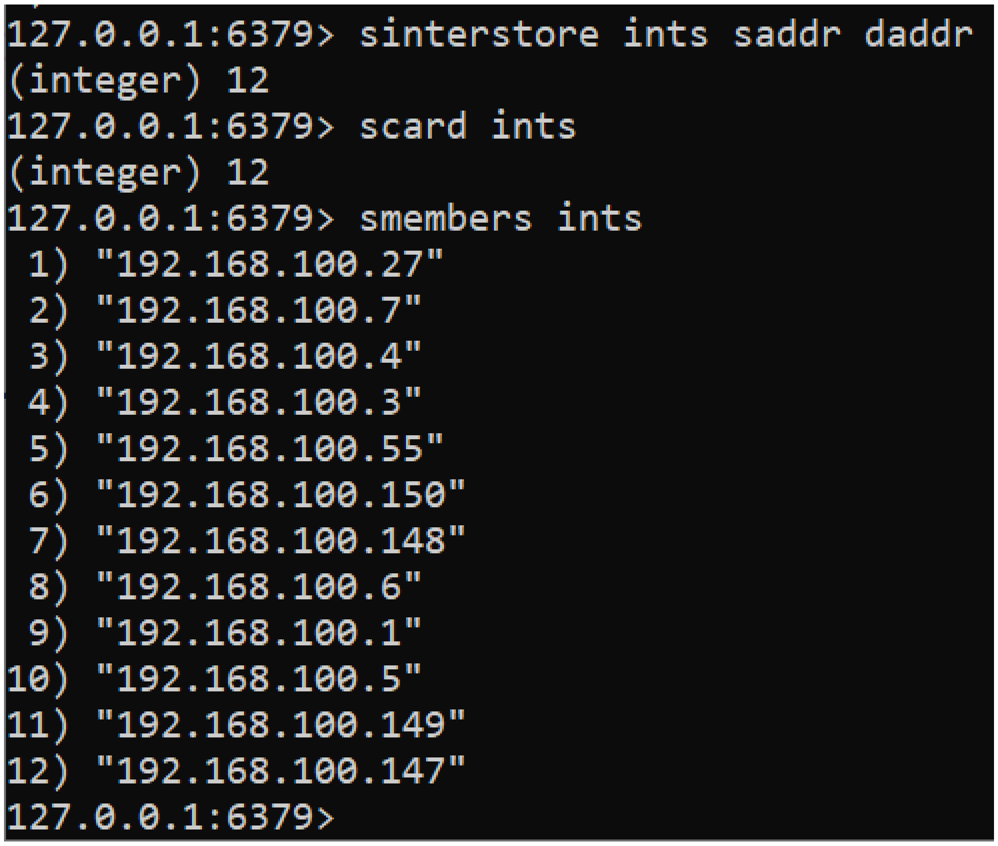

```
127.0.0.1:6379> sdiffstore diffs saddr daddr
(integer) 9
```

17) Finding cardinality and members of new key



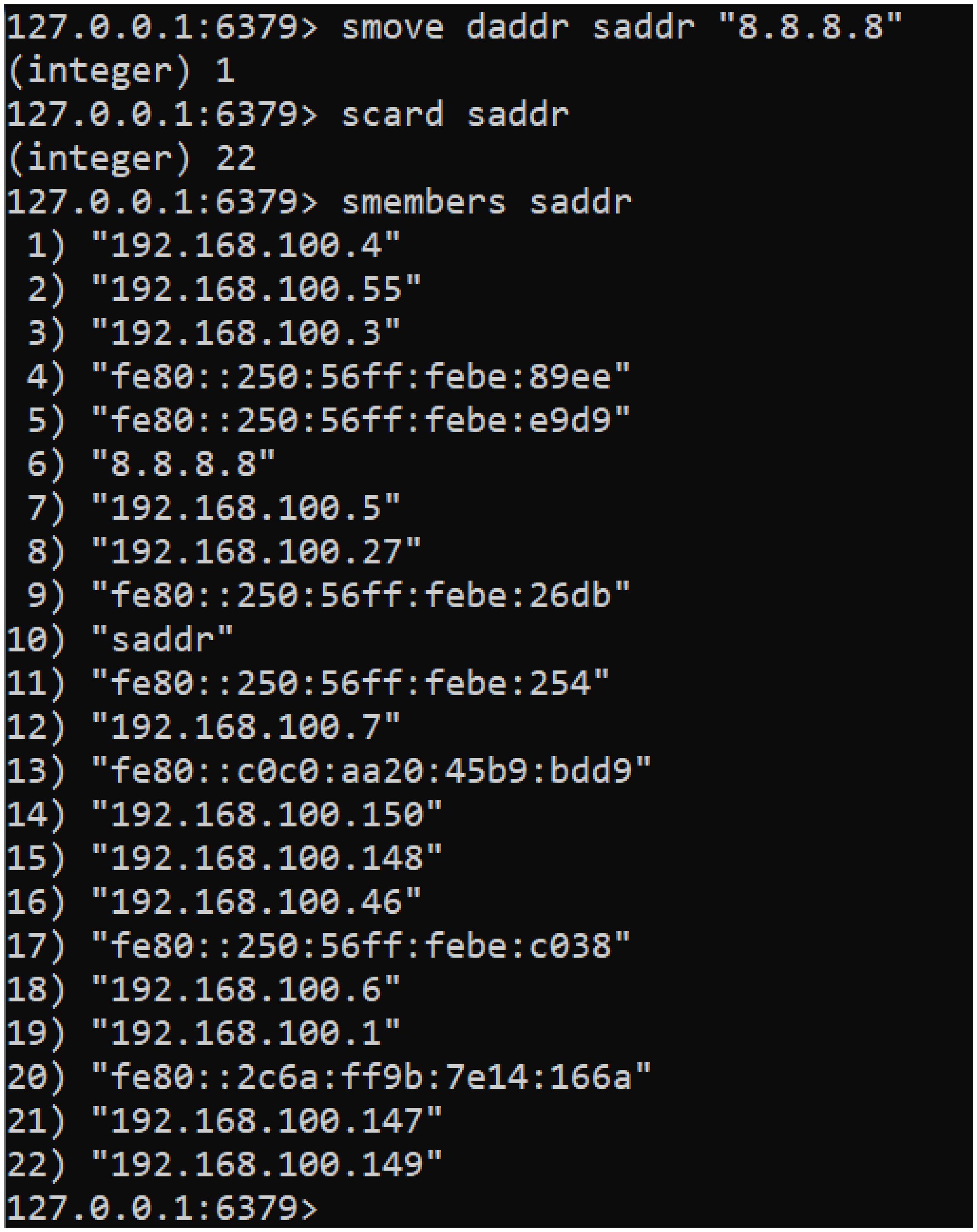
```
127.0.0.1:6379> scard diffs
(integer) 9
127.0.0.1:6379> smembers diffs
1) "192.168.100.46"
2) "fe80::c0c0:aa20:45b9:bdd9"
3) "fe80::250:56ff:febe:c038"
4) "fe80::250:56ff:febe:89ee"
5) "fe80::250:56ff:febe:26db"
6) "saddr"
7) "fe80::250:56ff:febe:254"
8) "fe80::250:56ff:febe:e9d9"
9) "fe80::2c6a:ff9b:7e14:166a"
127.0.0.1:6379>
```

18) Storing the intersection result in new key



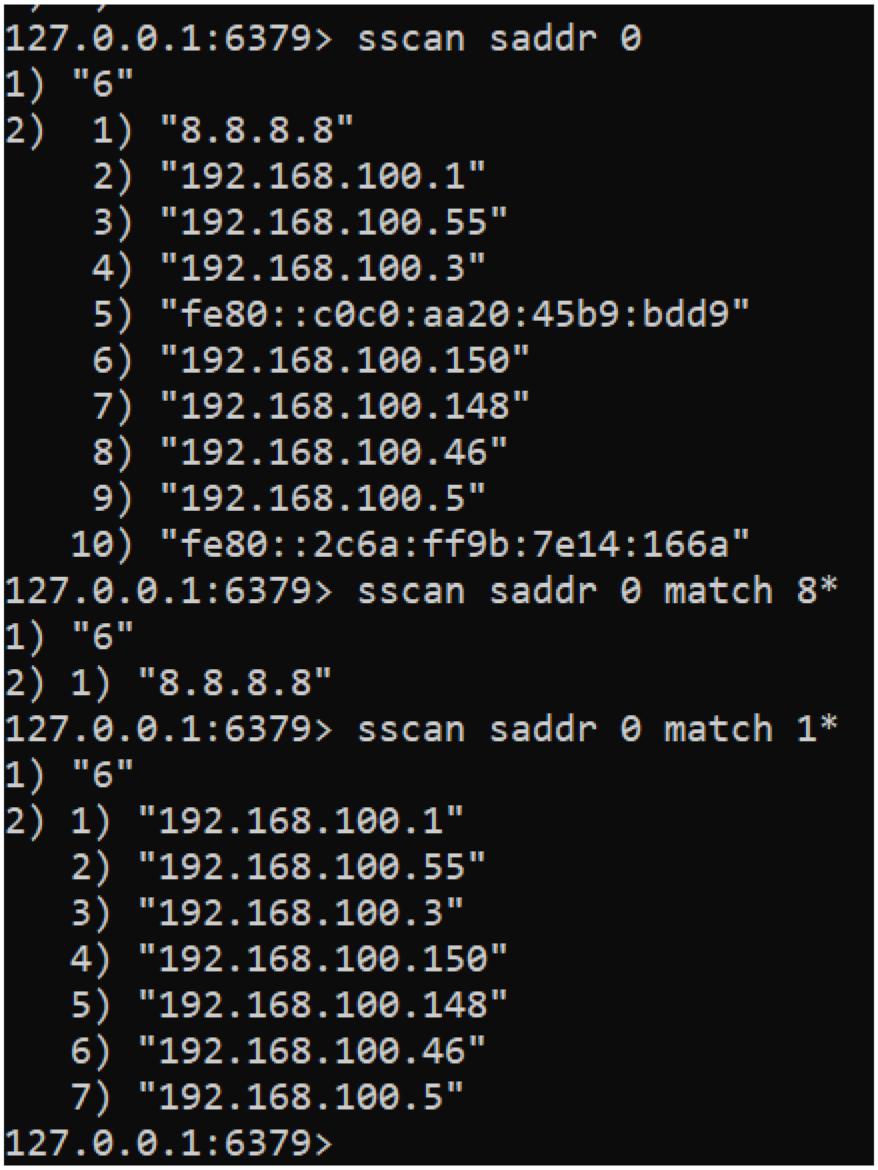
```
127.0.0.1:6379> sinterstore ints saddr daddr
(integer) 12
127.0.0.1:6379> scard ints
(integer) 12
127.0.0.1:6379> smembers ints
 1) "192.168.100.27"
 2) "192.168.100.7"
 3) "192.168.100.4"
 4) "192.168.100.3"
 5) "192.168.100.55"
 6) "192.168.100.150"
 7) "192.168.100.148"
 8) "192.168.100.6"
 9) "192.168.100.1"
10) "192.168.100.5"
11) "192.168.100.149"
12) "192.168.100.147"
127.0.0.1:6379>
```

19) Moving a key from one set to another



```
127.0.0.1:6379> smove daddr saddr "8.8.8.8"
(integer) 1
127.0.0.1:6379> scard saddr
(integer) 22
127.0.0.1:6379> smembers saddr
 1) "192.168.100.4"
 2) "192.168.100.55"
 3) "192.168.100.3"
 4) "fe80::250:56ff:febe:89ee"
 5) "fe80::250:56ff:febe:e9d9"
 6) "8.8.8.8"
 7) "192.168.100.5"
 8) "192.168.100.27"
 9) "fe80::250:56ff:febe:26db"
10) "saddr"
11) "fe80::250:56ff:febe:254"
12) "192.168.100.7"
13) "fe80::c0c0:aa20:45b9:bdd9"
14) "192.168.100.150"
15) "192.168.100.148"
16) "192.168.100.46"
17) "fe80::250:56ff:febe:c038"
18) "192.168.100.6"
19) "192.168.100.1"
20) "fe80::2c6a:ff9b:7e14:166a"
21) "192.168.100.147"
22) "192.168.100.149"
127.0.0.1:6379>
```

20) Using scan to show members and find keys with matching pattern here keys starting with 8

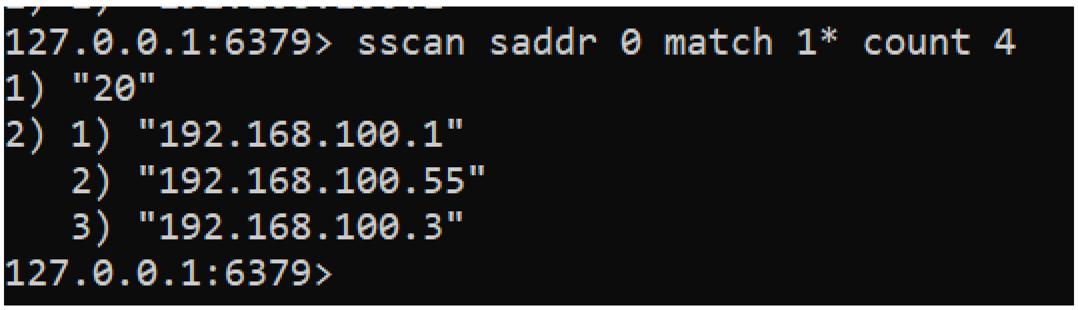


```
127.0.0.1:6379> sscan saddr 0
1) "6"
2)  1) "8.8.8.8"
    2) "192.168.100.1"
    3) "192.168.100.55"
    4) "192.168.100.3"
    5) "fe80::c0c0:aa20:45b9:bdd9"
    6) "192.168.100.150"
    7) "192.168.100.148"
    8) "192.168.100.46"
    9) "192.168.100.5"
   10) "fe80::2c6a:ff9b:7e14:166a"
127.0.0.1:6379> sscan saddr 0 match 8*
1) "6"
2) 1) "8.8.8.8"
127.0.0.1:6379> sscan saddr 0 match 1*
1) "6"
2) 1) "192.168.100.1"
   2) "192.168.100.55"
   3) "192.168.100.3"
   4) "192.168.100.150"
   5) "192.168.100.148"
   6) "192.168.100.46"
   7) "192.168.100.5"
127.0.0.1:6379>
```

Where 0 is cursor and gets all the values staring from 1 and 8, scan command gets 10 key values

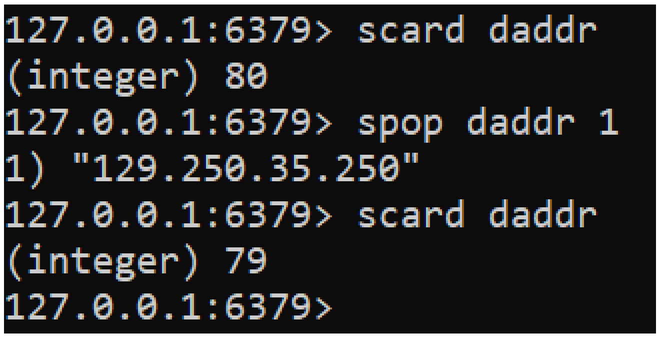
Scan starts from zero cursor

21) Finding values of set starting from 1 using scan and giving 4 results



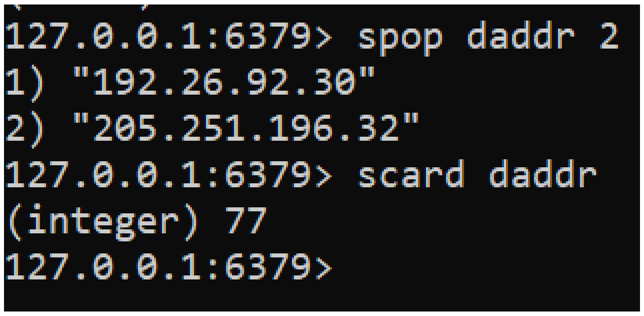
```
127.0.0.1:6379> sscan saddr 0 match 1* count 4
1) "20"
2) 1) "192.168.100.1"
   2) "192.168.100.55"
   3) "192.168.100.3"
127.0.0.1:6379>
```

22) Popping a value from set



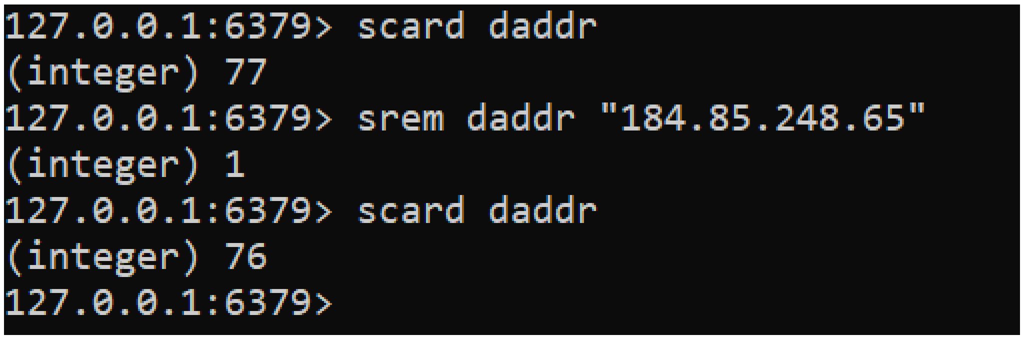
```
127.0.0.1:6379> scard daddr
(integer) 80
127.0.0.1:6379> spop daddr 1
1) "129.250.35.250"
127.0.0.1:6379> scard daddr
(integer) 79
127.0.0.1:6379>
```

23) Removing 2 values from set



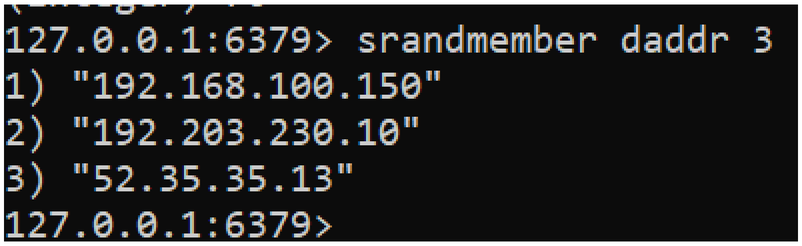
```
127.0.0.1:6379> spop daddr 2
1) "192.26.92.30"
2) "205.251.196.32"
127.0.0.1:6379> scard daddr
(integer) 77
127.0.0.1:6379>
```

24) Removing value from set



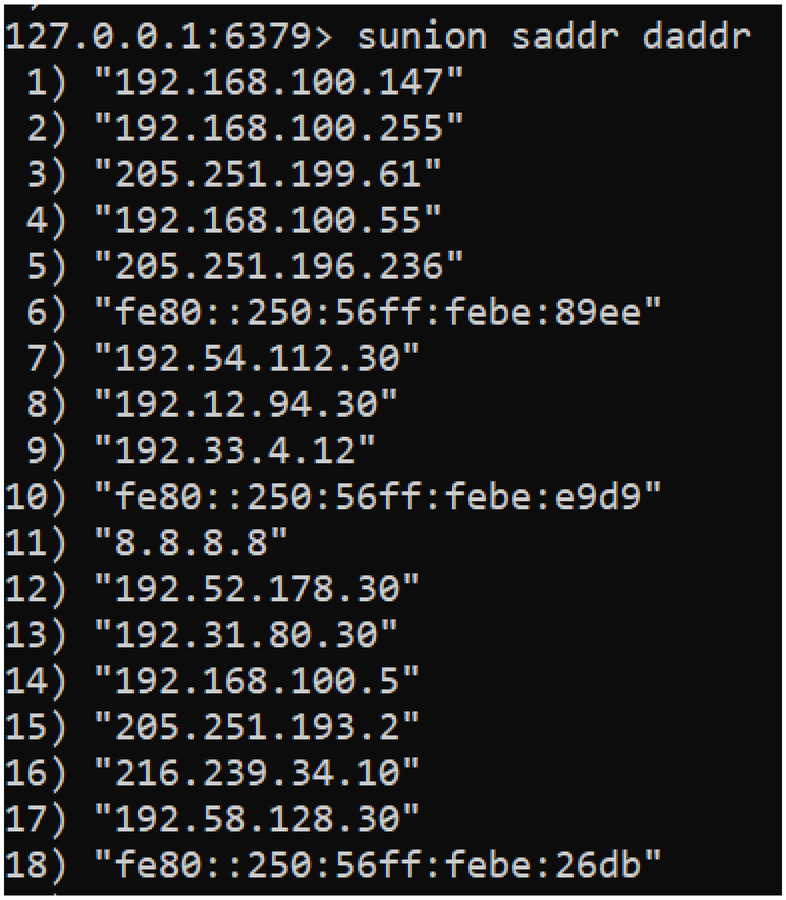
```
127.0.0.1:6379> scard daddr
(integer) 77
127.0.0.1:6379> srem daddr "184.85.248.65"
(integer) 1
127.0.0.1:6379> scard daddr
(integer) 76
127.0.0.1:6379>
```

25) Getting 3 random members of set



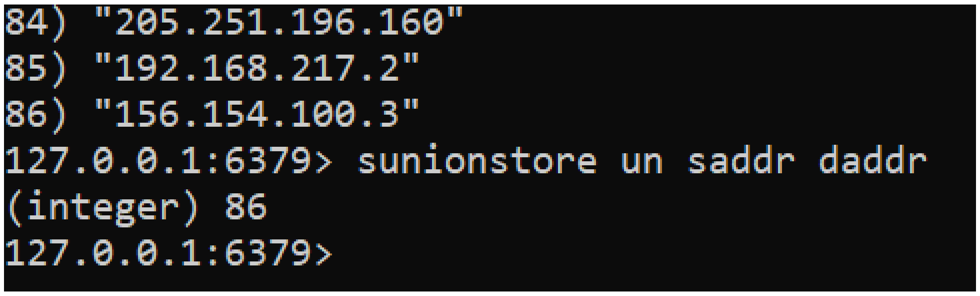
```
127.0.0.1:6379> srandmember daddr 3
1) "192.168.100.150"
2) "192.203.230.10"
3) "52.35.35.13"
127.0.0.1:6379>
```

26) Taking union of 2 sets



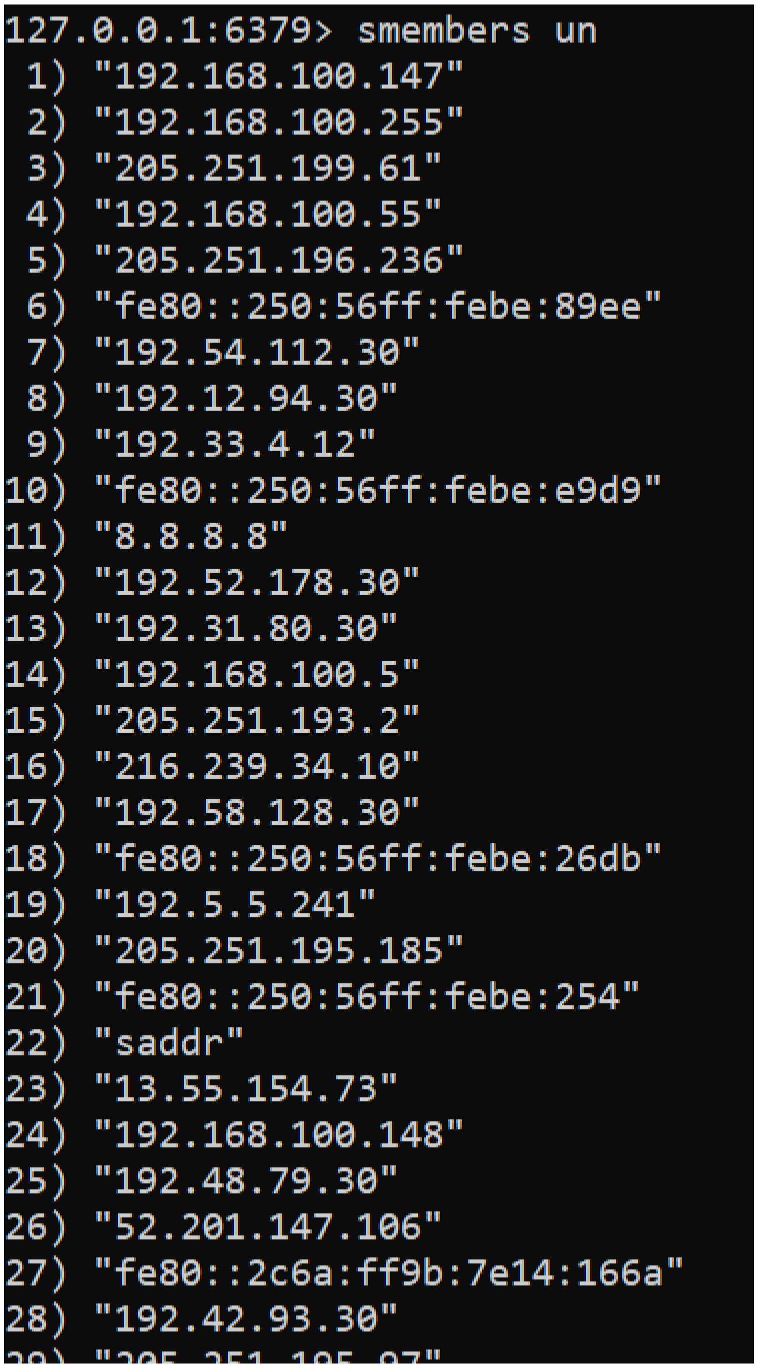
```
127.0.0.1:6379> sunion saddr daddr
 1) "192.168.100.147"
 2) "192.168.100.255"
 3) "205.251.199.61"
 4) "192.168.100.55"
 5) "205.251.196.236"
 6) "fe80::250:56ff:febe:89ee"
 7) "192.54.112.30"
 8) "192.12.94.30"
 9) "192.33.4.12"
10) "fe80::250:56ff:febe:e9d9"
11) "8.8.8.8"
12) "192.52.178.30"
13) "192.31.80.30"
14) "192.168.100.5"
15) "205.251.193.2"
16) "216.239.34.10"
17) "192.58.128.30"
18) "fe80::250:56ff:febe:26db"
```

27) Storing union of sets to a key



```
84) "205.251.196.160"
85) "192.168.217.2"
86) "156.154.100.3"
127.0.0.1:6379> sunionstore un saddr daddr
(integer) 86
127.0.0.1:6379>
```

28) Getting all members of union key

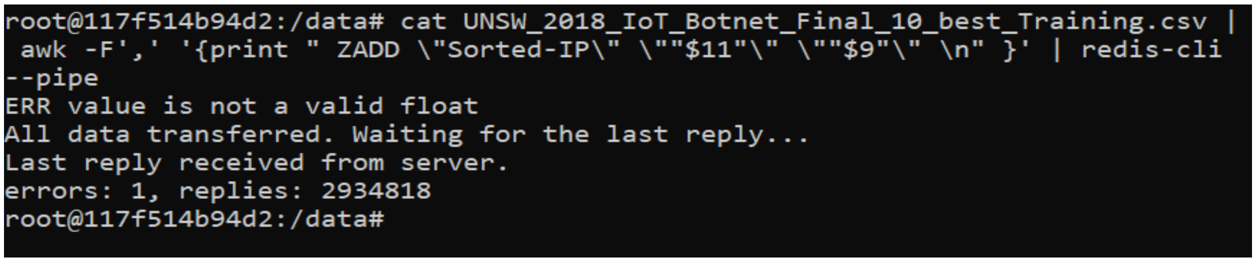


```
127.0.0.1:6379> smembers un
 1) "192.168.100.147"
 2) "192.168.100.255"
 3) "205.251.199.61"
 4) "192.168.100.55"
 5) "205.251.196.236"
 6) "fe80::250:56ff:febe:89ee"
 7) "192.54.112.30"
 8) "192.12.94.30"
 9) "192.33.4.12"
10) "fe80::250:56ff:febe:e9d9"
11) "8.8.8.8"
12) "192.52.178.30"
13) "192.31.80.30"
14) "192.168.100.5"
15) "205.251.193.2"
16) "216.239.34.10"
17) "192.58.128.30"
18) "fe80::250:56ff:febe:26db"
19) "192.5.5.241"
20) "205.251.195.185"
21) "fe80::250:56ff:febe:254"
22) "saddr"
23) "13.55.154.73"
24) "192.168.100.148"
25) "192.48.79.30"
26) "52.201.147.106"
27) "fe80::2c6a:ff9b:7e14:166a"
28) "192.42.93.30"
```

SORTED SETS

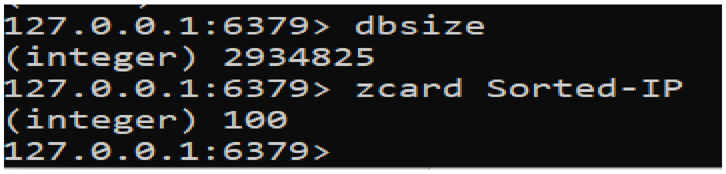
Every element has a score associated with it. Sorting will be based on score

29) Uploading a column as a sorted set. Making 1st sorted set

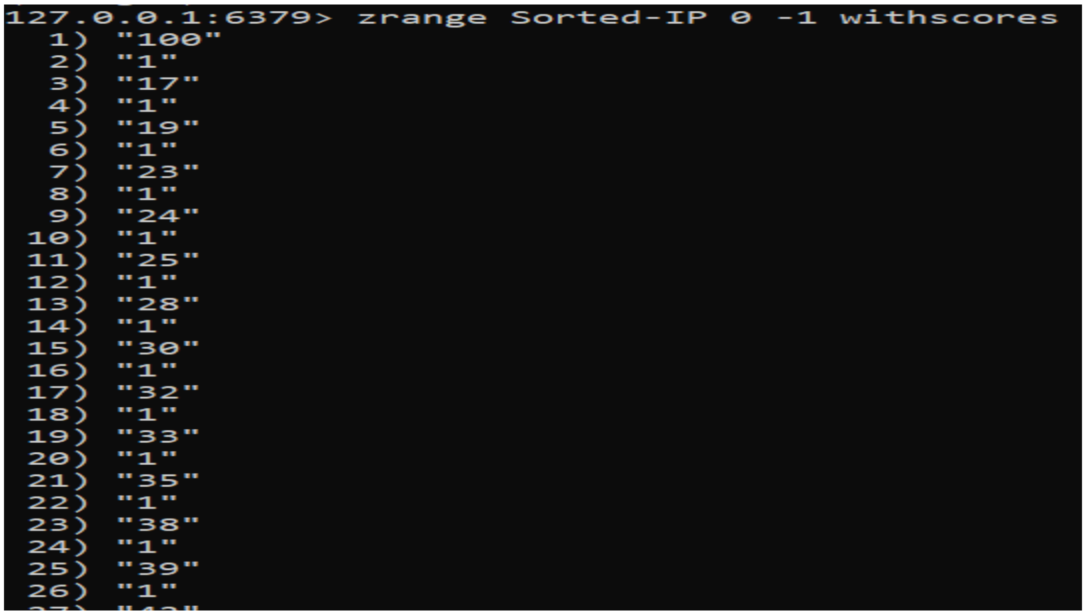


```
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv |
 awk -F',' '{print " ZADD \"Sorted-IP\" \""$11"\" \""$9"\" \n" }' | redis-cli
--pipe
ERR value is not a valid float
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 1, replies: 2934818
root@117f514b94d2:/data#
```

30) Checking members of the sorted set



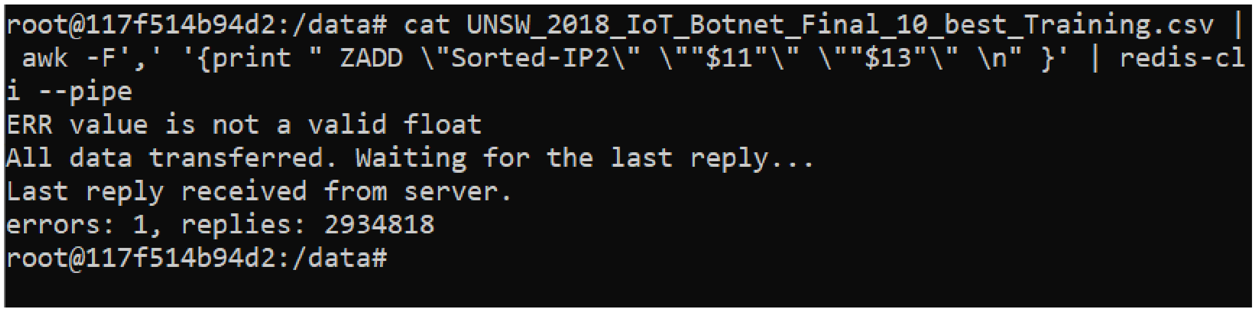
```
127.0.0.1:6379> dbsize
(integer) 2934825
127.0.0.1:6379> zcard Sorted-IP
(integer) 100
127.0.0.1:6379>
```



```
127.0.0.1:6379> zrange Sorted-IP 0 -1 withscores
  1) "100"
  2) "1"
  3) "17"
  4) "1"
  5) "19"
  6) "1"
  7) "23"
  8) "1"
  9) "24"
 10) "1"
 11) "25"
 12) "1"
 13) "28"
 14) "1"
 15) "30"
 16) "1"
 17) "32"
 18) "1"
 19) "33"
 20) "1"
 21) "35"
 22) "1"
 23) "38"
 24) "1"
 25) "39"
 26) "1"
```

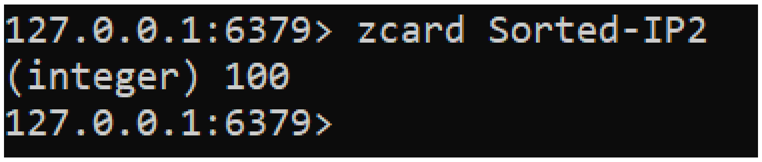
Sorting based on scores, if score same then values sorted based on lexicographical order

31) Uploading a column as a sorted set. Making 2nd sorted set

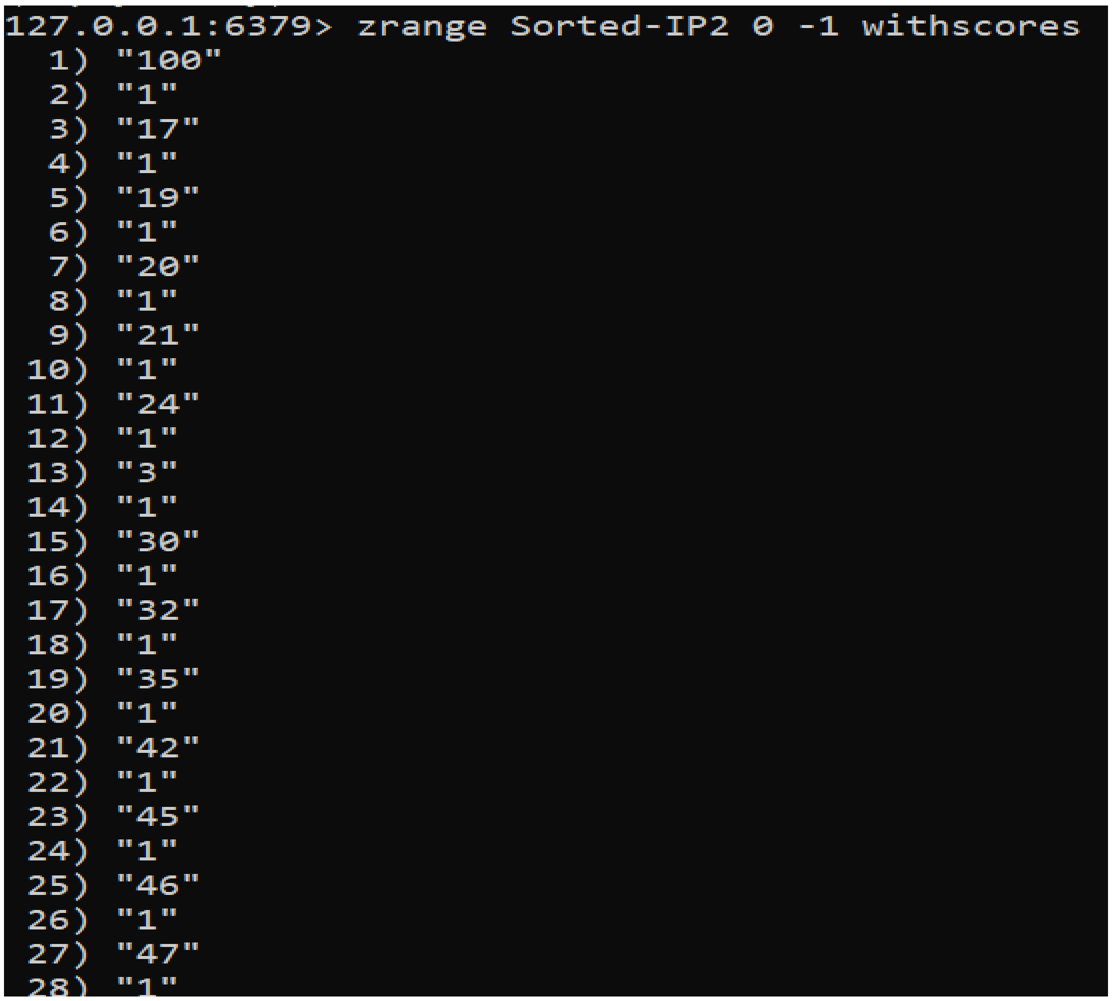


```
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv |
 awk -F',' '{print " ZADD \"Sorted-IP2\" \""$11"\" \""$13"\" \n" }' | redis-cl
i --pipe
ERR value is not a valid float
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 1, replies: 2934818
root@117f514b94d2:/data#
```

32) Checking the size and members in the 2nd sorted set

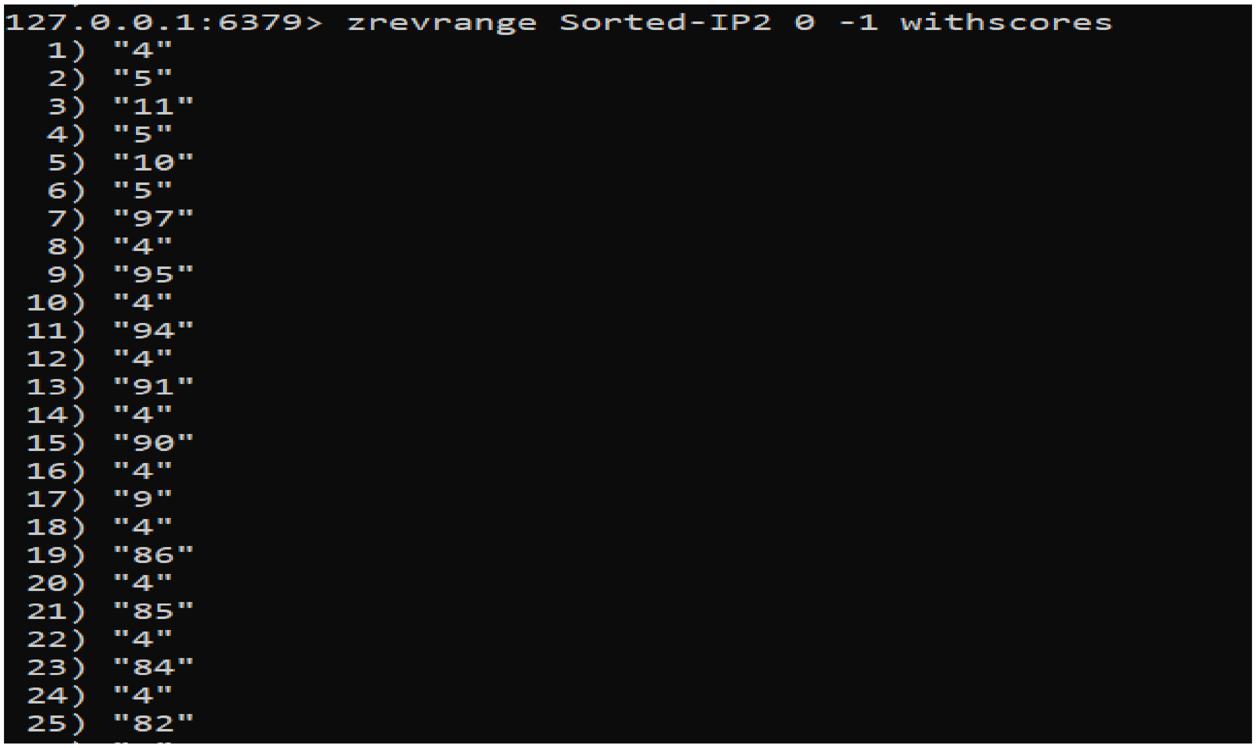


```
127.0.0.1:6379> zcard Sorted-IP2
(integer) 100
127.0.0.1:6379>
```

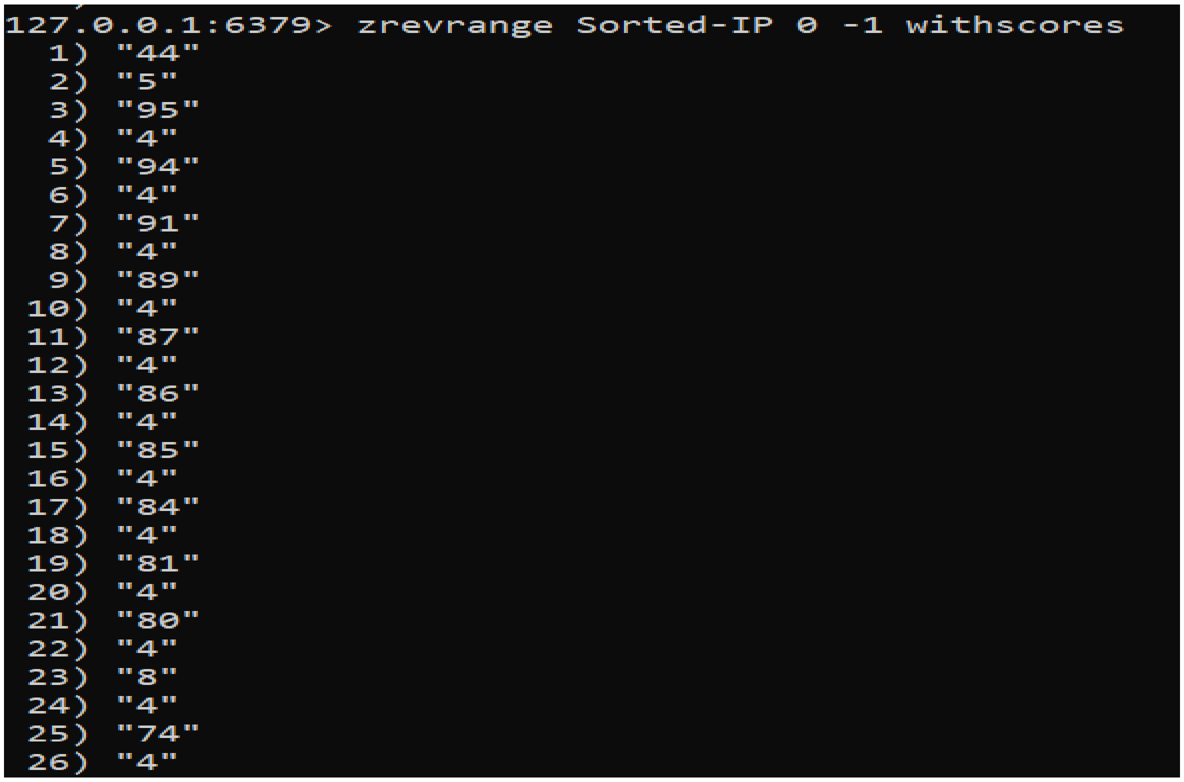


```
127.0.0.1:6379> zrange Sorted-IP2 0 -1 withscores
  1) "100"
  2) "1"
  3) "17"
  4) "1"
  5) "19"
  6) "1"
  7) "20"
  8) "1"
  9) "21"
 10) "1"
 11) "24"
 12) "1"
 13) "3"
 14) "1"
 15) "30"
 16) "1"
 17) "32"
 18) "1"
 19) "35"
 20) "1"
 21) "42"
 22) "1"
 23) "45"
 24) "1"
 25) "46"
 26) "1"
 27) "47"
 28) "1"
```

33) Getting all members of sorted sets in reverse order with scores

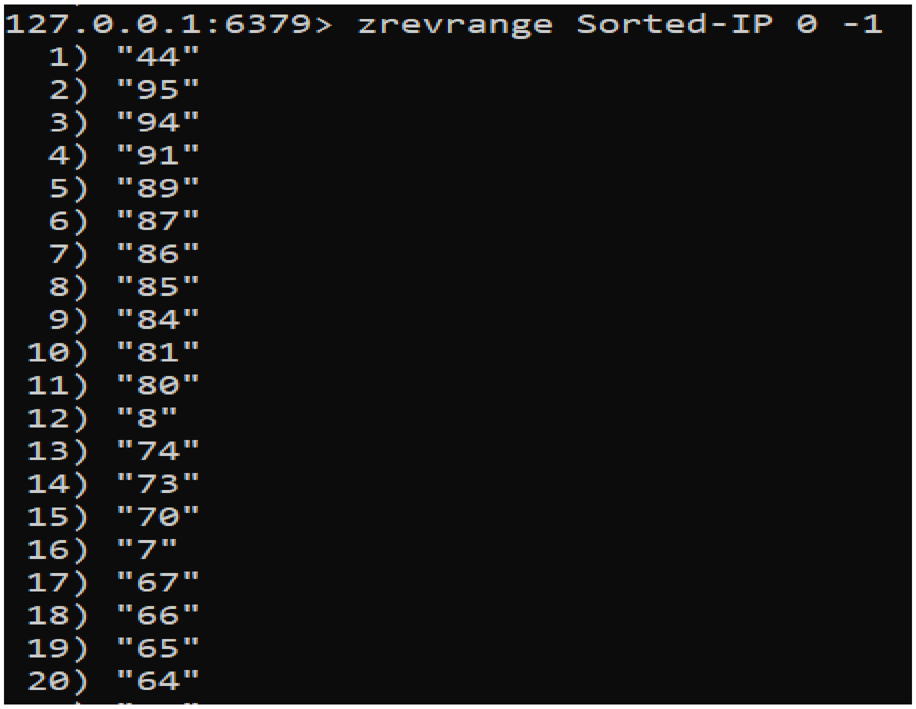


```
127.0.0.1:6379> zrevrange Sorted-IP2 0 -1 withscores
  1) "4"
  2) "5"
  3) "11"
  4) "5"
  5) "10"
  6) "5"
  7) "97"
  8) "4"
  9) "95"
 10) "4"
 11) "94"
 12) "4"
 13) "91"
 14) "4"
 15) "90"
 16) "4"
 17) "9"
 18) "4"
 19) "86"
 20) "4"
 21) "85"
 22) "4"
 23) "84"
 24) "4"
 25) "82"
```

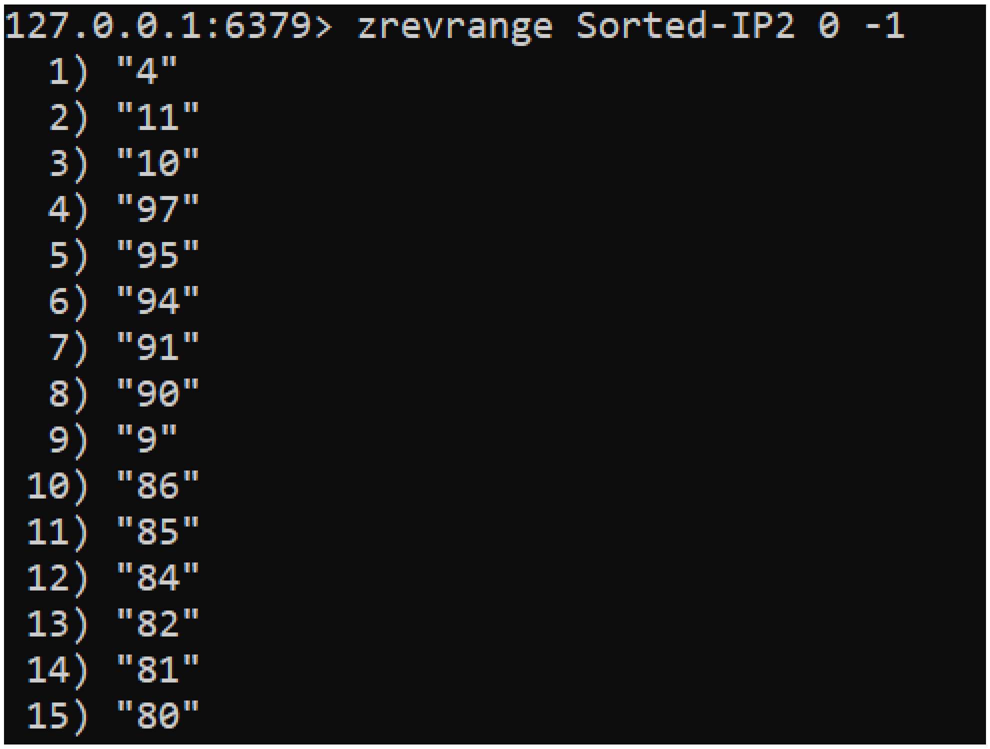


```
127.0.0.1:6379> zrevrange Sorted-IP 0 -1 withscores
  1) "44"
  2) "5"
  3) "95"
  4) "4"
  5) "94"
  6) "4"
  7) "91"
  8) "4"
  9) "89"
 10) "4"
 11) "87"
 12) "4"
 13) "86"
 14) "4"
 15) "85"
 16) "4"
 17) "84"
 18) "4"
 19) "81"
 20) "4"
 21) "80"
 22) "4"
 23) "8"
 24) "4"
 25) "74"
 26) "4"
```

34) Getting all members of sorted sets in reverse order without scores



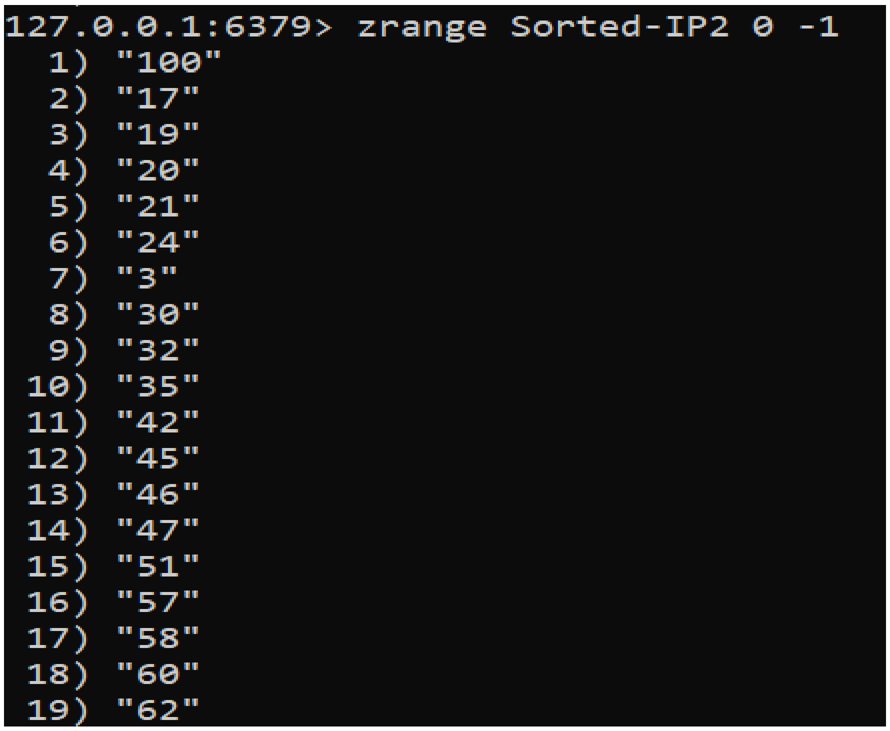
```
127.0.0.1:6379> zrevrange Sorted-IP 0 -1
  1) "44"
  2) "95"
  3) "94"
  4) "91"
  5) "89"
  6) "87"
  7) "86"
  8) "85"
  9) "84"
 10) "81"
 11) "80"
 12) "8"
 13) "74"
 14) "73"
 15) "70"
 16) "7"
 17) "67"
 18) "66"
 19) "65"
 20) "64"
```



```
127.0.0.1:6379> zrevrange Sorted-IP2 0 -1
  1) "4"
  2) "11"
  3) "10"
  4) "97"
  5) "95"
  6) "94"
  7) "91"
  8) "90"
  9) "9"
 10) "86"
 11) "85"
 12) "84"
 13) "82"
 14) "81"
 15) "80"
```

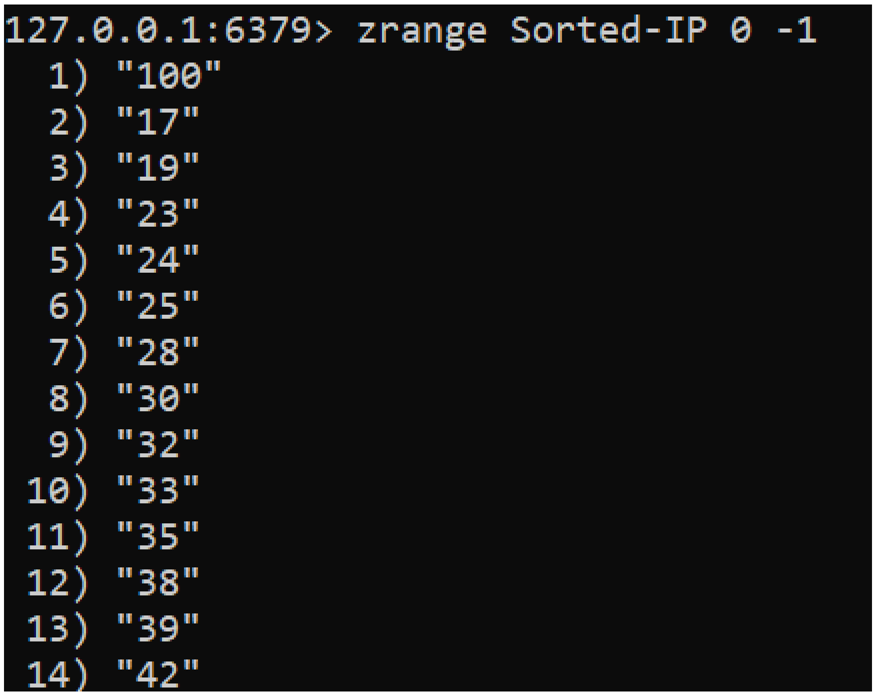
35) Getting All elements of sorted set

- **zrange Sorted-IP2 0 -1**



```
127.0.0.1:6379> zrange Sorted-IP2 0 -1
  1) "100"
  2) "17"
  3) "19"
  4) "20"
  5) "21"
  6) "24"
  7) "3"
  8) "30"
  9) "32"
 10) "35"
 11) "42"
 12) "45"
 13) "46"
 14) "47"
 15) "51"
 16) "57"
 17) "58"
 18) "60"
 19) "62"
```

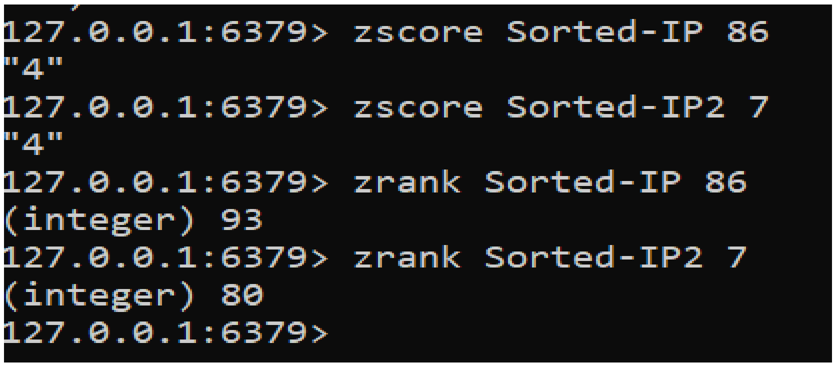
- **zrange Sorted-IP 0 -1**



```
127.0.0.1:6379> zrange Sorted-IP 0 -1
  1) "100"
  2) "17"
  3) "19"
  4) "23"
  5) "24"
  6) "25"
  7) "28"
  8) "30"
  9) "32"
 10) "33"
 11) "35"
 12) "38"
 13) "39"
 14) "42"
```

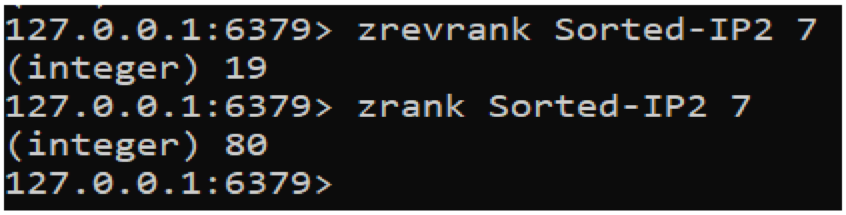
36) finding the score and rank of a values of a set with given key and its value

- **zscore Sorted-IP 86**
- **zrank Sorted-IP2 7**



```
127.0.0.1:6379> zscore Sorted-IP 86
"4"
127.0.0.1:6379> zscore Sorted-IP2 7
"4"
127.0.0.1:6379> zrank Sorted-IP 86
(integer) 93
127.0.0.1:6379> zrank Sorted-IP2 7
(integer) 80
127.0.0.1:6379>
```

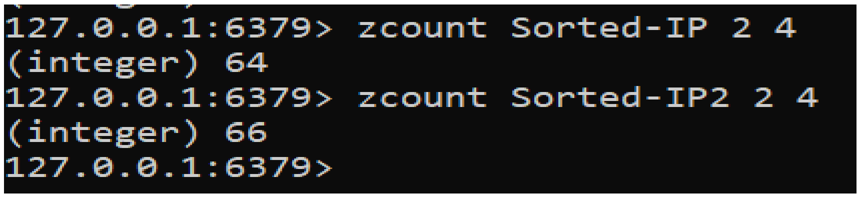
37) Getting rank in reverse order



```
127.0.0.1:6379> zrevrank Sorted-IP2 7
(integer) 19
127.0.0.1:6379> zrank Sorted-IP2 7
(integer) 80
127.0.0.1:6379>
```

38) Getting the counts of all the values between score ranges 2 and 4

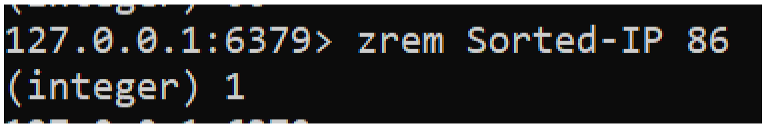
- **zcount Sorted-IP 2 4**



```
127.0.0.1:6379> zcount Sorted-IP 2 4
(integer) 64
127.0.0.1:6379> zcount Sorted-IP2 2 4
(integer) 66
127.0.0.1:6379>
```

39) Removing value from a sorted set

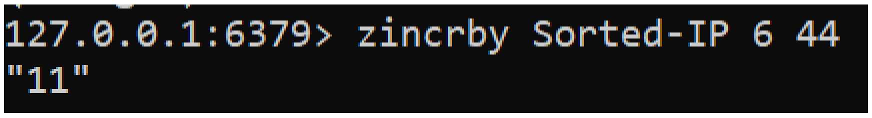
- **zrem Sorted-IP 86**



```
127.0.0.1:6379> zrem Sorted-IP 86
(integer) 1
```

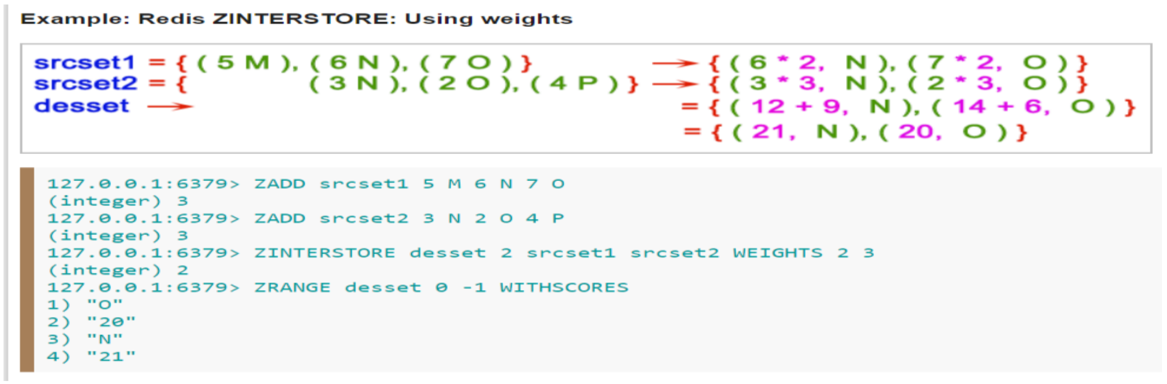
40) Increasing value of a sorted set

- **zincrby Sorted-IP 6 44**



```
127.0.0.1:6379> zincrby Sorted-IP 6 44
"11"
```

41) Intersecting 2 sorted sets and storing it in a new key,
where 3 is score of sorted set multiplied by 3 and 4 is score of sorted set multiplied by 4 and after intersection both scores are added like and 2 tell number of keys to be intersected

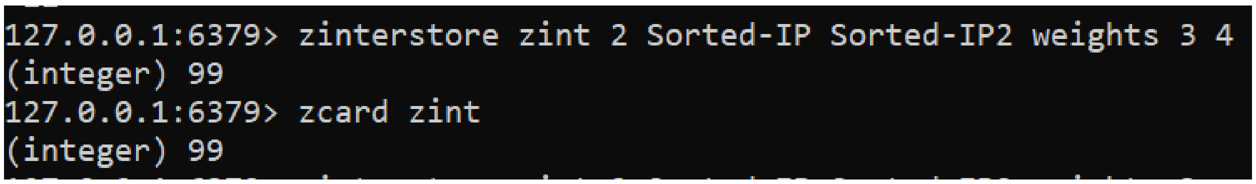


Example: Redis ZINTERSTORE: Using weights

```
srcset1 = { ( 5 M ), ( 6 N ), ( 7 O ) }            → { ( 6 * 2,  N ), ( 7 * 2,  O ) }
srcset2 = {          ( 3 N ), ( 2 O ), ( 4 P ) }   → { ( 3 * 3,  N ), ( 2 * 3,  O ) }
desset  →                                          = { ( 12 + 9,  N ), ( 14 + 6,  O ) }
                                                   = { ( 21,  N ), ( 20,  O ) }
```

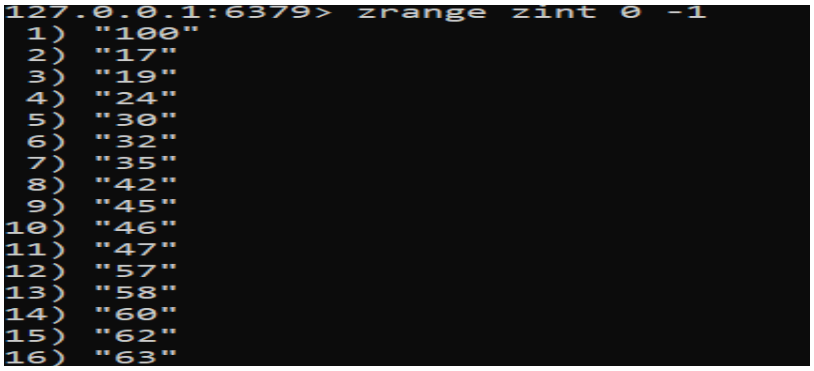
```
127.0.0.1:6379> ZADD srcset1 5 M 6 N 7 O
(integer) 3
127.0.0.1:6379> ZADD srcset2 3 N 2 O 4 P
(integer) 3
127.0.0.1:6379> ZINTERSTORE desset 2 srcset1 srcset2 WEIGHTS 2 3
(integer) 2
127.0.0.1:6379> ZRANGE desset 0 -1 WITHSCORES
1) "O"
2) "20"
3) "N"
4) "21"
```

- **zinterstore zint 2 Sorted-IP Sorted-IP2 weights 3 4**



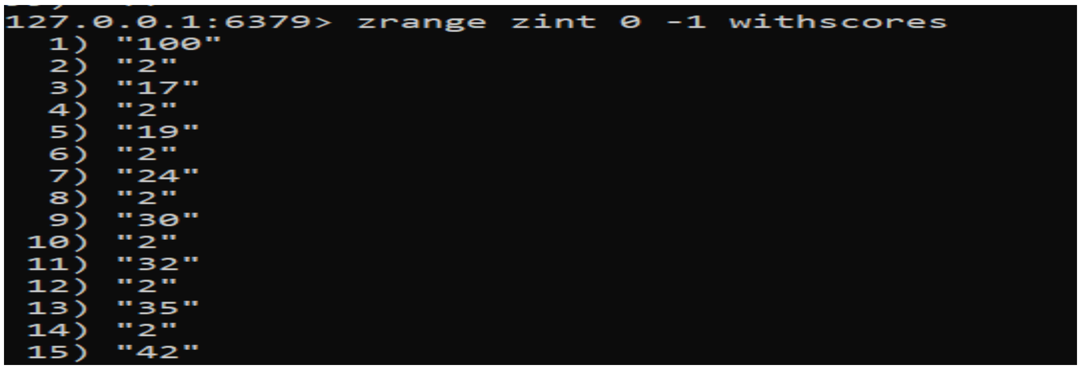
```
127.0.0.1:6379> zinterstore zint 2 Sorted-IP Sorted-IP2 weights 3 4
(integer) 99
127.0.0.1:6379> zcard zint
(integer) 99
```

- **zrange zint 0 -1**



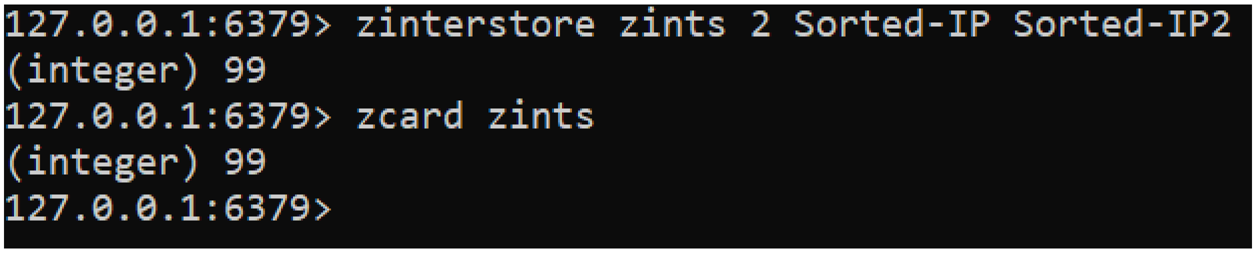
```
127.0.0.1:6379> zrange zint 0 -1
 1) "100"
 2) "17"
 3) "19"
 4) "24"
 5) "30"
 6) "32"
 7) "35"
 8) "42"
 9) "45"
10) "46"
11) "47"
12) "57"
13) "58"
14) "60"
15) "62"
16) "63"
```

- **zrange zint 0 -1 withscores**



```
127.0.0.1:6379> zrange zint 0 -1 withscores
  1) "100"
  2) "2"
  3) "17"
  4) "2"
  5) "19"
  6) "2"
  7) "24"
  8) "2"
  9) "30"
 10) "2"
 11) "32"
 12) "2"
 13) "35"
 14) "2"
 15) "42"
```

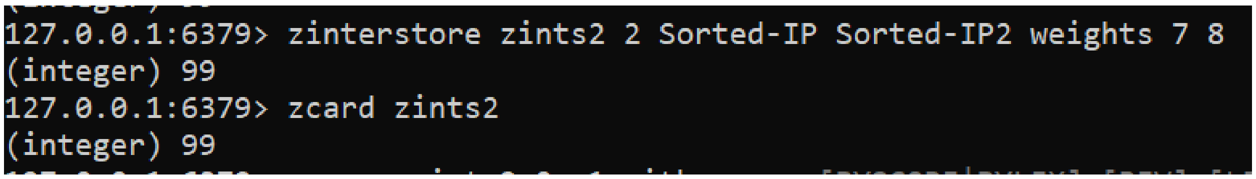
42) Intersecting 2 sorted sets without weights



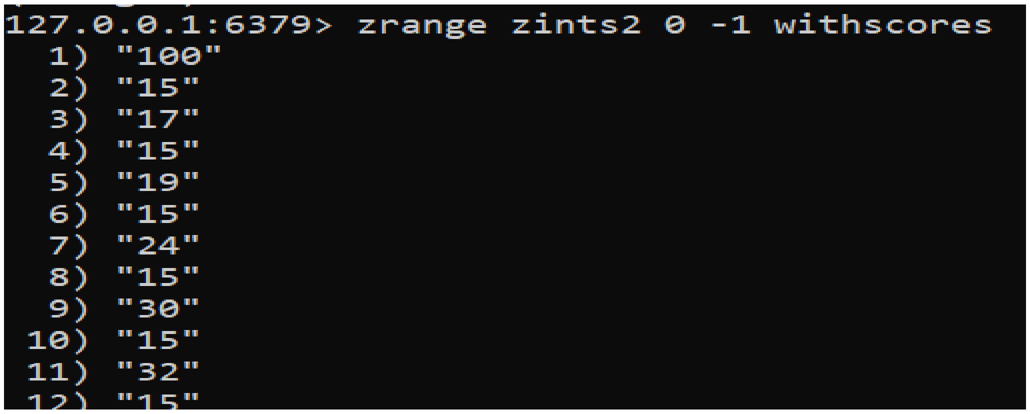
```
127.0.0.1:6379> zinterstore zints 2 Sorted-IP Sorted-IP2
(integer) 99
127.0.0.1:6379> zcard zints
(integer) 99
127.0.0.1:6379>
```

Here 2 tell number of keys to be intersected

43) Intersecting 2 sorted sets with score

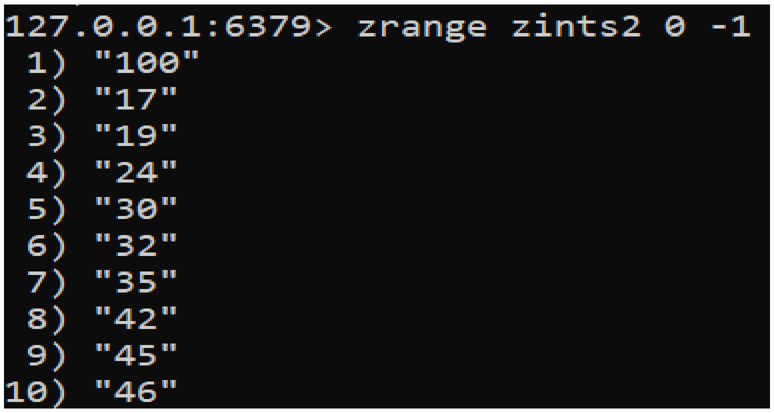


```
127.0.0.1:6379> zinterstore zints2 2 Sorted-IP Sorted-IP2 weights 7 8
(integer) 99
127.0.0.1:6379> zcard zints2
(integer) 99
```



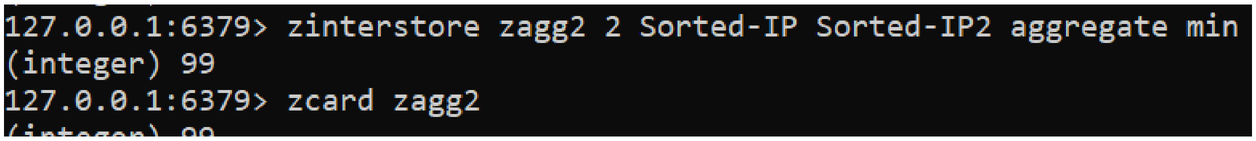
```
127.0.0.1:6379> zrange zints2 0 -1 withscores
  1) "100"
  2) "15"
  3) "17"
  4) "15"
  5) "19"
  6) "15"
  7) "24"
  8) "15"
  9) "30"
 10) "15"
 11) "32"
 12) "15"
```

Where weights 7 multiplies score of Sorted-IP by 7 and similiary weights 8 multiplies score of Sorted-IP2 by 8, and final score added



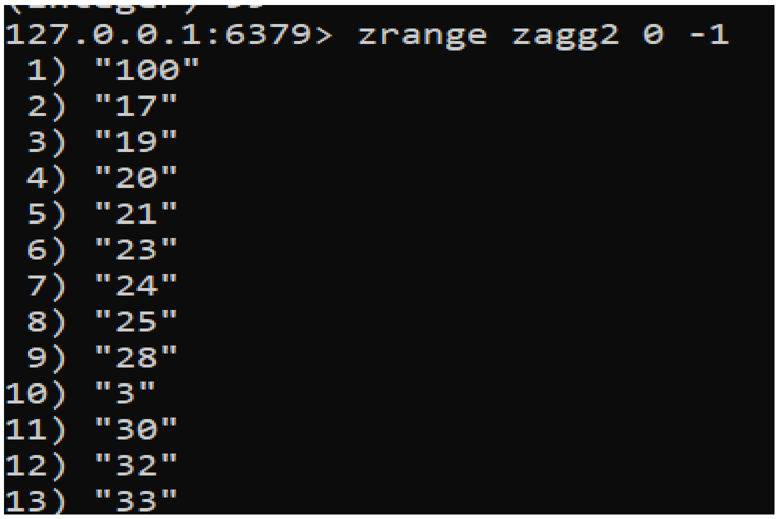
```
127.0.0.1:6379> zrange zints2 0 -1
 1) "100"
 2) "17"
 3) "19"
 4) "24"
 5) "30"
 6) "32"
 7) "35"
 8) "42"
 9) "45"
10) "46"
```

44) Intersecting 2 sorted sets and with aggregate using the min score of the common values and storing the result in a key



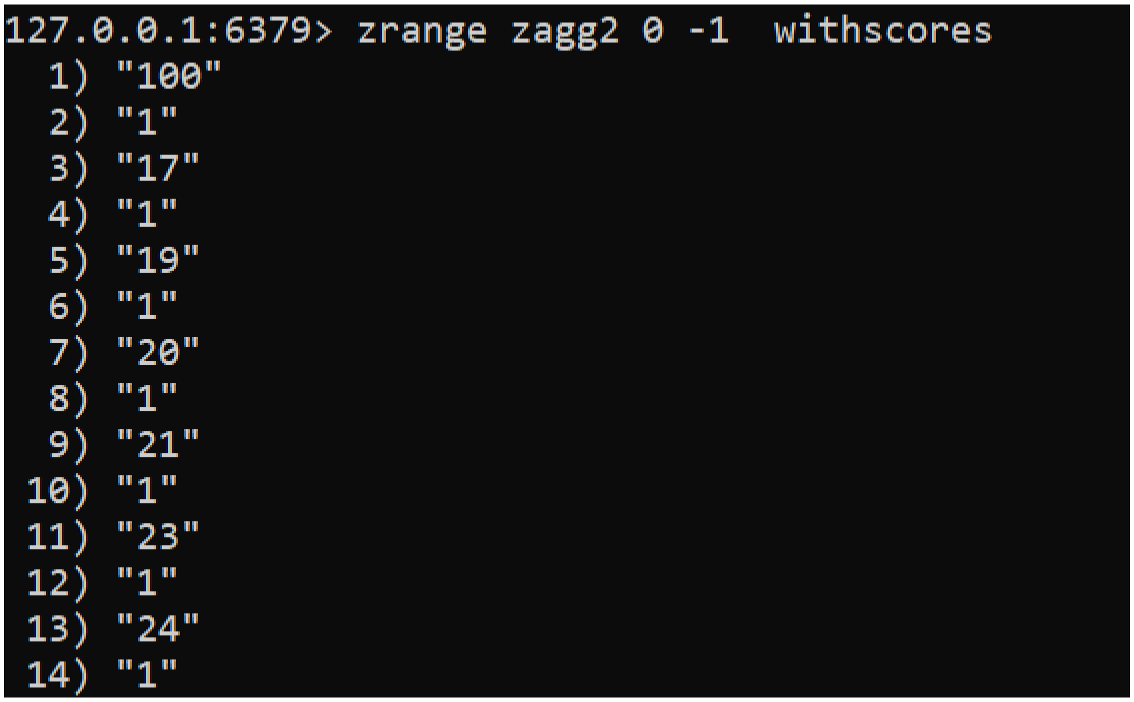
```
127.0.0.1:6379> zinterstore zagg2 2 Sorted-IP Sorted-IP2 aggregate min
(integer) 99
127.0.0.1:6379> zcard zagg2
```

- **zrange zagg2 0 -1**



```
127.0.0.1:6379> zrange zagg2 0 -1
 1) "100"
 2) "17"
 3) "19"
 4) "20"
 5) "21"
 6) "23"
 7) "24"
 8) "25"
 9) "28"
10) "3"
11) "30"
12) "32"
13) "33"
```

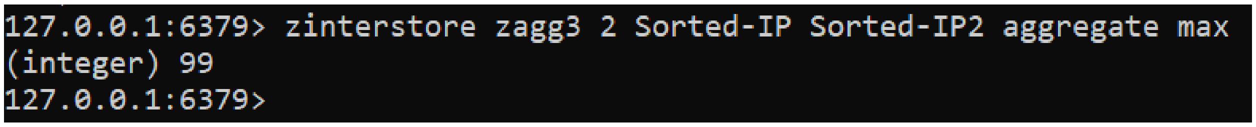
- **zrange zagg2 0 -1 withscores**



```
127.0.0.1:6379> zrange zagg2 0 -1  withscores
  1) "100"
  2) "1"
  3) "17"
  4) "1"
  5) "19"
  6) "1"
  7) "20"
  8) "1"
  9) "21"
 10) "1"
 11) "23"
 12) "1"
 13) "24"
 14) "1"
```

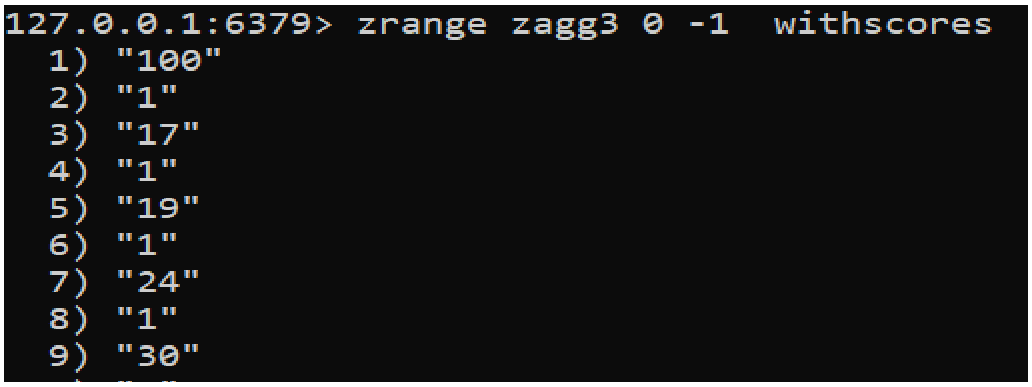
45) Intersecting 2 sorted sets and with aggregate using the max score of the common values and storing the result in a key

- **zinterstore zagg3 2 Sorted-IP Sorted-IP2 aggregate max**



```
127.0.0.1:6379> zinterstore zagg3 2 Sorted-IP Sorted-IP2 aggregate max
(integer) 99
127.0.0.1:6379>
```

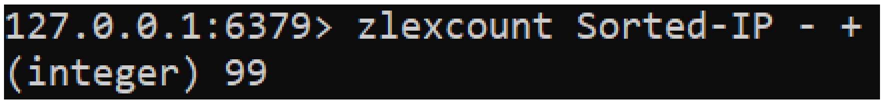
- **zrange zagg3 0 -1 withscores**



```
127.0.0.1:6379> zrange zagg3 0 -1  withscores
  1) "100"
  2) "1"
  3) "17"
  4) "1"
  5) "19"
  6) "1"
  7) "24"
  8) "1"
  9) "30"
```

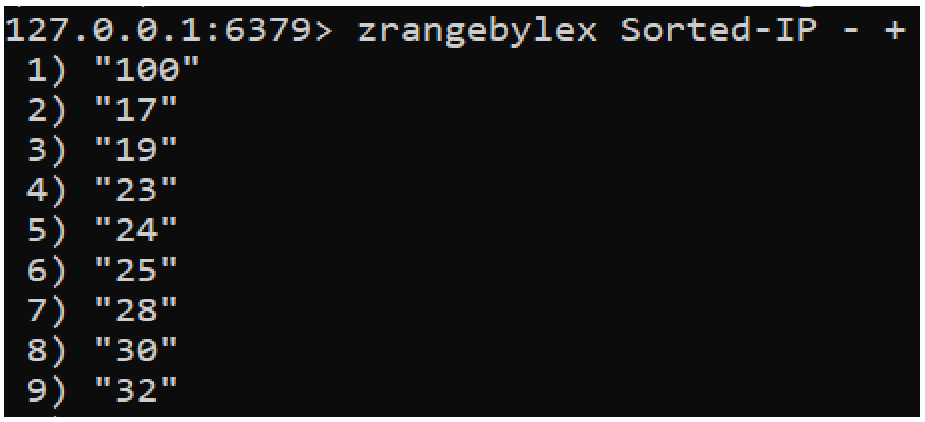
46) Using Zlex commands count and range zlex is used to sort values with same score.
This Zlexcount command is used to return the number of elements in the sorted set at the key with a value between minimum and maximum when all the elements in a sorted set are inserted with the same score, in order to force lexicographical ordering.

- **zlexcount Sorted-IP - +**



```
127.0.0.1:6379> zlexcount Sorted-IP - +
(integer) 99
```

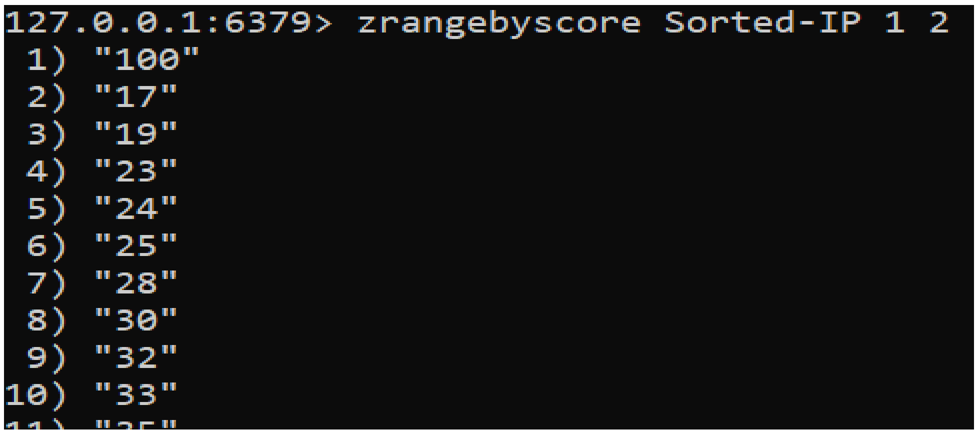
- **zrangebylex Sorted-IP - +**



```
127.0.0.1:6379> zrangebylex Sorted-IP - +
 1) "100"
 2) "17"
 3) "19"
 4) "23"
 5) "24"
 6) "25"
 7) "28"
 8) "30"
 9) "32"
```

47) Getting values ranged by score limits

- **zrangebyscore Sorted-IP 1 2**

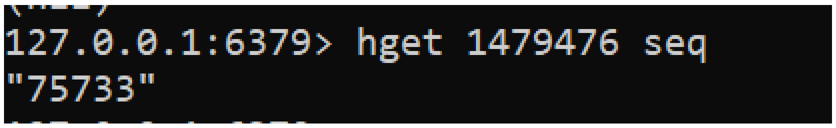


```
127.0.0.1:6379> zrangebyscore Sorted-IP 1 2
 1) "100"
 2) "17"
 3) "19"
 4) "23"
 5) "24"
 6) "25"
 7) "28"
 8) "30"
 9) "32"
10) "33"
```

HASHES

48) getting the value of key with specified field

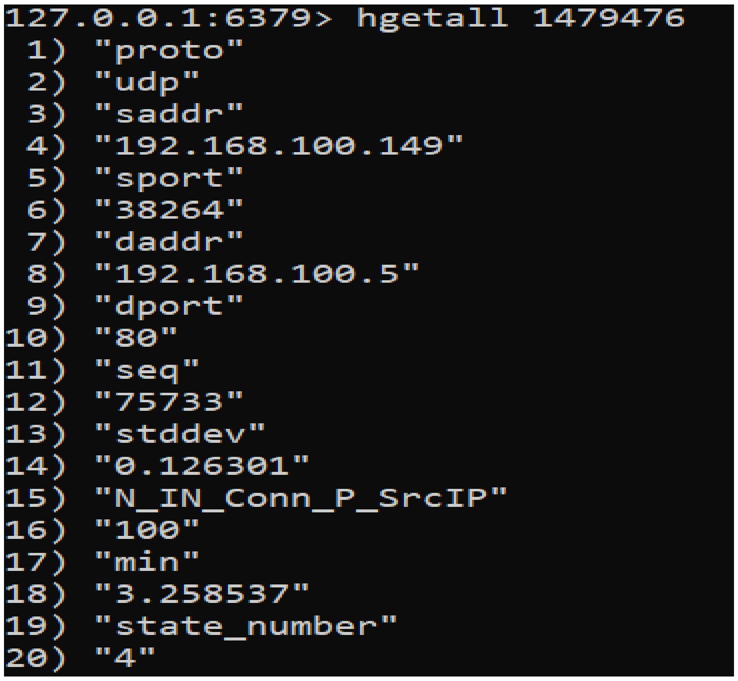
- **hget 1479476 seq**



```
127.0.0.1:6379> hget 1479476 seq
"75733"
```

49) Getting all fields of a hash key

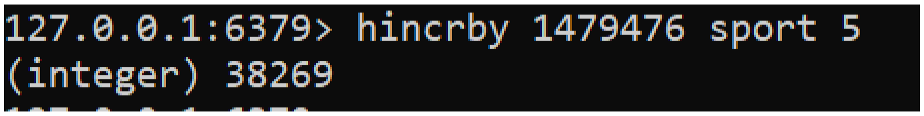
- hgetall 1479476



```
127.0.0.1:6379> hgetall 1479476
 1) "proto"
 2) "udp"
 3) "saddr"
 4) "192.168.100.149"
 5) "sport"
 6) "38264"
 7) "daddr"
 8) "192.168.100.5"
 9) "dport"
10) "80"
11) "seq"
12) "75733"
13) "stddev"
14) "0.126301"
15) "N_IN_Conn_P_SrcIP"
16) "100"
17) "min"
18) "3.258537"
19) "state_number"
20) "4"
```

50) Increasing the field value of a particular key

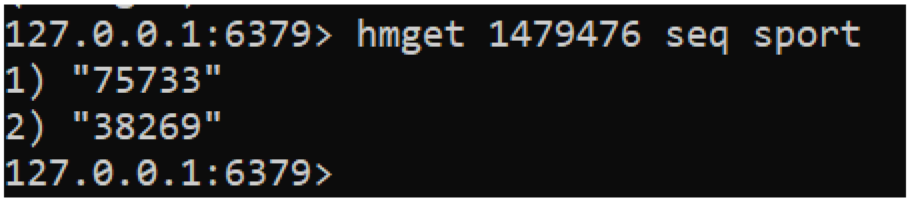
- **hincrby 1479476 sport 5**



```
127.0.0.1:6379> hincrby 1479476 sport 5
(integer) 38269
```

51) Using hmget to get values of 2 fields

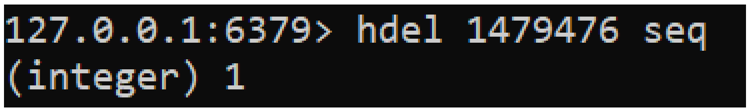
- **hmget 1479476 seq sport**



```
127.0.0.1:6379> hmget 1479476 seq sport
1) "75733"
2) "38269"
127.0.0.1:6379>
```

52) Deleting a field

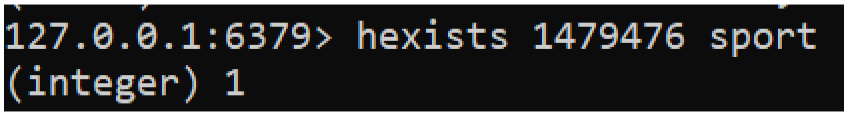
- **hdel 1479476 seq**



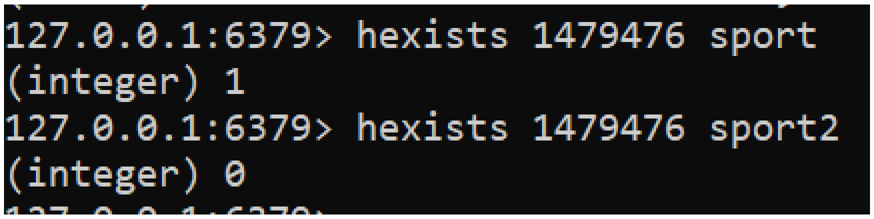
```
127.0.0.1:6379> hdel 1479476 seq
(integer) 1
```

53) Checking is field exists

- **hexists 1479476 sport**



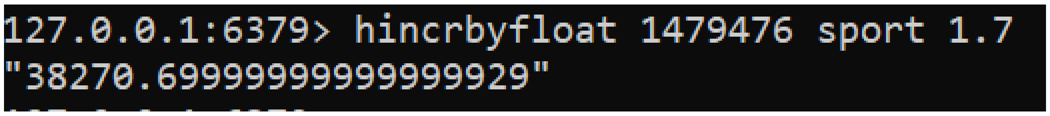
```
127.0.0.1:6379> hexists 1479476 sport
(integer) 1
```



```
127.0.0.1:6379> hexists 1479476 sport
(integer) 1
127.0.0.1:6379> hexists 1479476 sport2
(integer) 0
```

54) Increaing field value by floadt

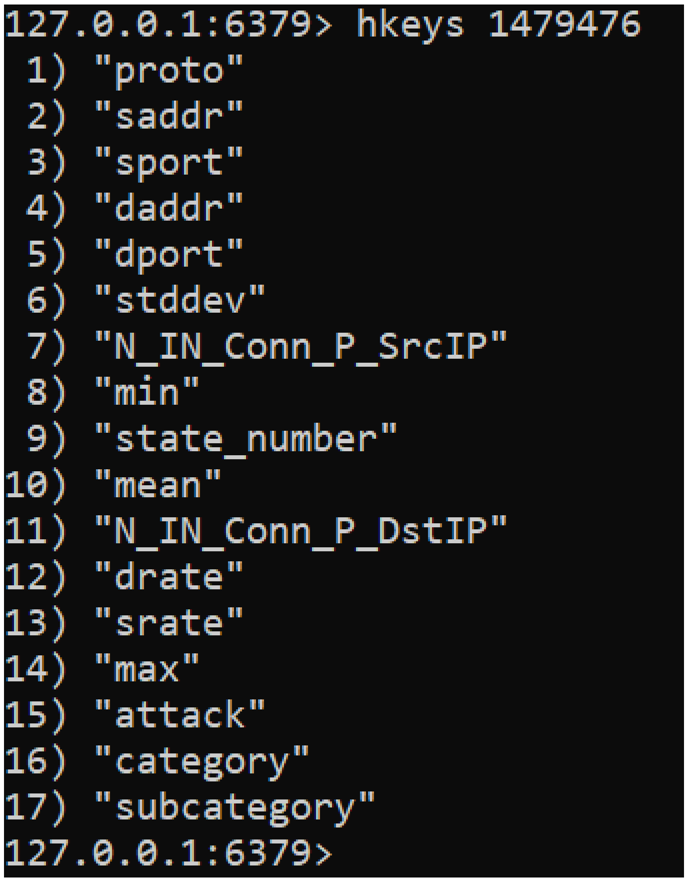
- **hincrbyfloat 1479476 sport 1.7**



```
127.0.0.1:6379> hincrbyfloat 1479476 sport 1.7
"38270.69999999999999929"
```

55) getting all the fields names of a hash key

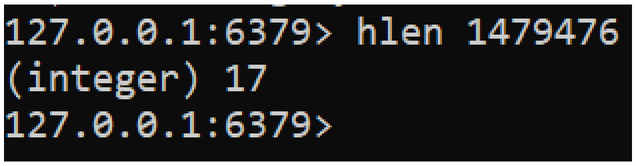
- **hkeys 1479476**



```
127.0.0.1:6379> hkeys 1479476
 1) "proto"
 2) "saddr"
 3) "sport"
 4) "daddr"
 5) "dport"
 6) "stddev"
 7) "N_IN_Conn_P_SrcIP"
 8) "min"
 9) "state_number"
10) "mean"
11) "N_IN_Conn_P_DstIP"
12) "drate"
13) "srate"
14) "max"
15) "attack"
16) "category"
17) "subcategory"
127.0.0.1:6379>
```

56) Finding length(total count) of fields of a key

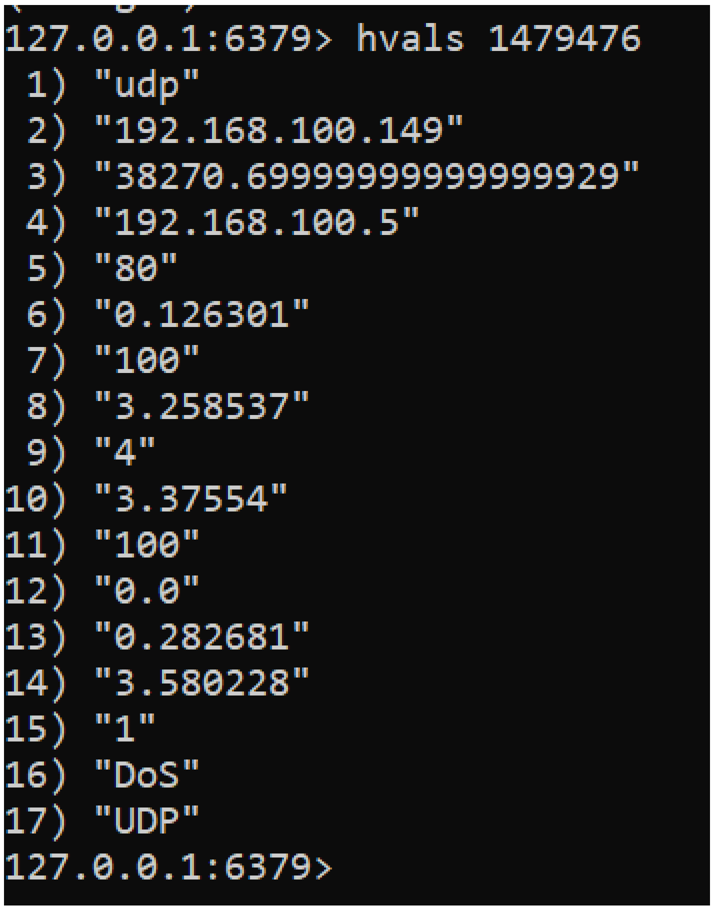
- **hlen 1479476**



```
127.0.0.1:6379> hlen 1479476
(integer) 17
127.0.0.1:6379>
```

57) Getting values of all fields of a key

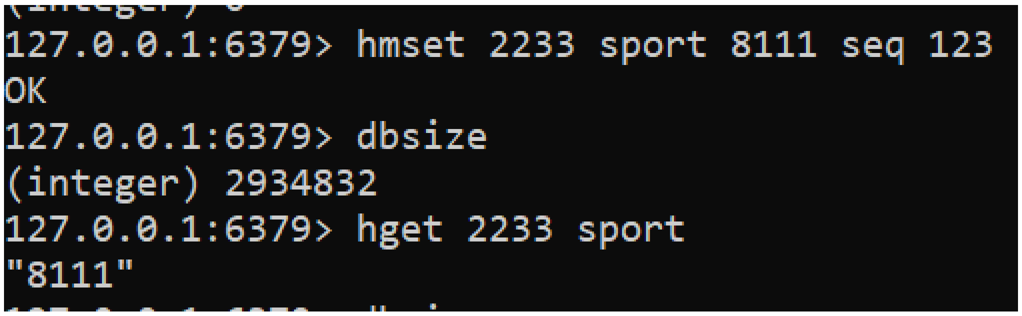
- **hvals 1479476**



```
127.0.0.1:6379> hvals 1479476
 1) "udp"
 2) "192.168.100.149"
 3) "38270.6999999999999929"
 4) "192.168.100.5"
 5) "80"
 6) "0.126301"
 7) "100"
 8) "3.258537"
 9) "4"
10) "3.37554"
11) "100"
12) "0.0"
13) "0.282681"
14) "3.580228"
15) "1"
16) "DoS"
17) "UDP"
127.0.0.1:6379>
```

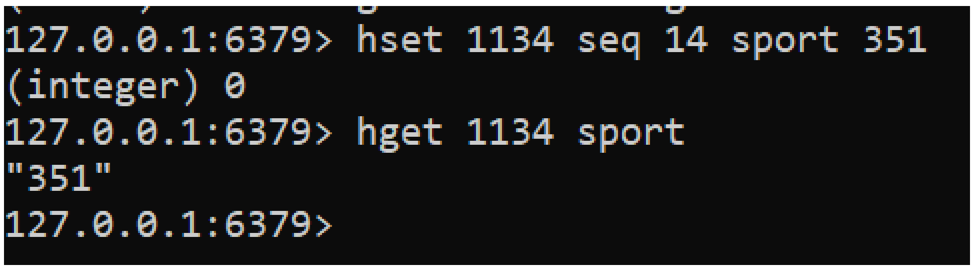
58) Making new multiple fields of a new key with hmset

- **hmset 2233 sport 8111 seq 123**



```
127.0.0.1:6379> hmset 2233 sport 8111 seq 123
OK
127.0.0.1:6379> dbsize
(integer) 2934832
127.0.0.1:6379> hget 2233 sport
"8111"
```

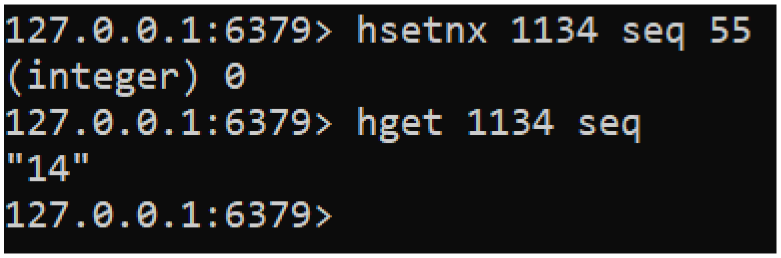
- **hset 1134 seq 14 sport 351**



```
127.0.0.1:6379> hset 1134 seq 14 sport 351
(integer) 0
127.0.0.1:6379> hget 1134 sport
"351"
127.0.0.1:6379>
```

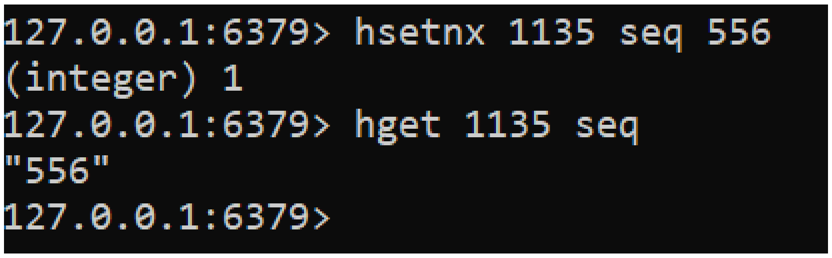
59) Making new key if key with same name does not exists

- **hsetnx 1134 seq 55**



```
127.0.0.1:6379> hsetnx 1134 seq 55
(integer) 0
127.0.0.1:6379> hget 1134 seq
"14"
127.0.0.1:6379>
```

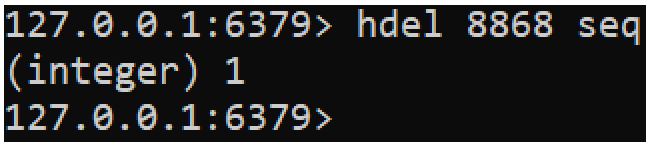
- **hsetnx 1135 seq 556**



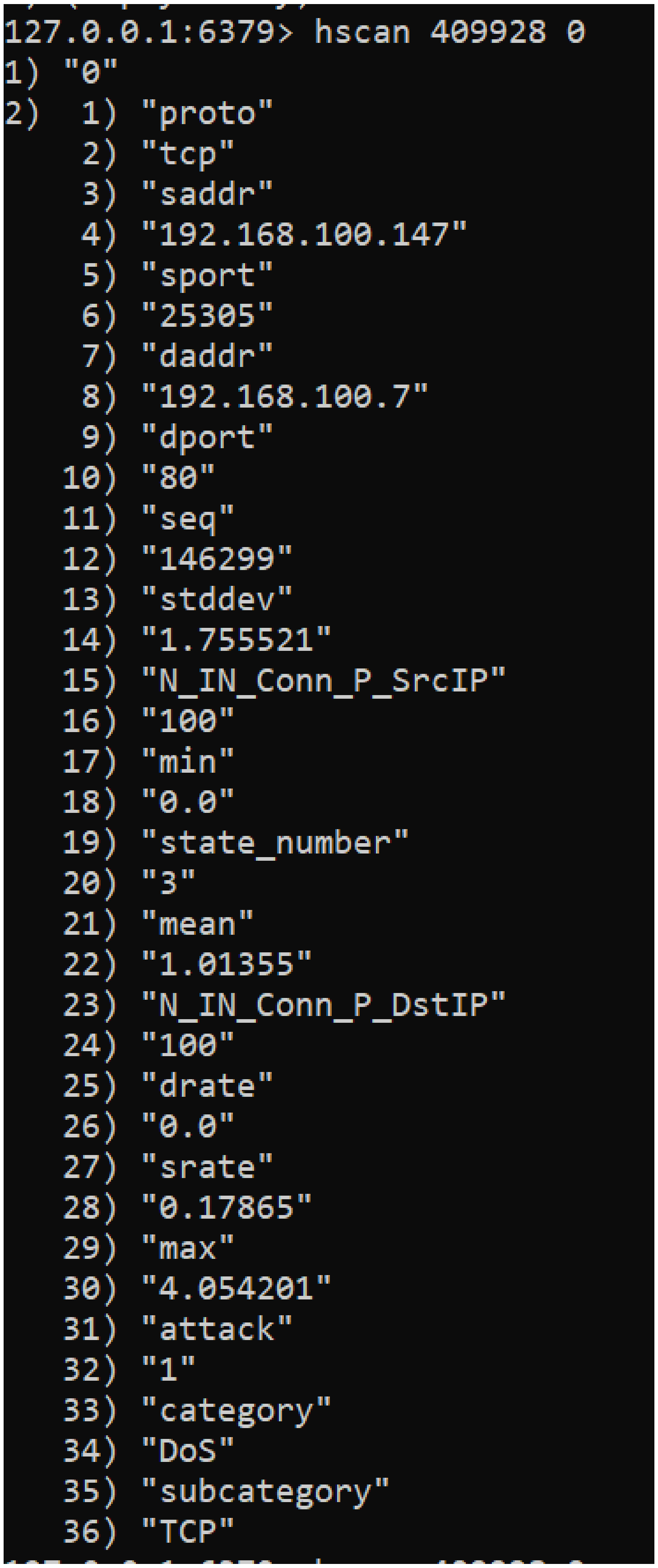
```
127.0.0.1:6379> hsetnx 1135 seq 556
(integer) 1
127.0.0.1:6379> hget 1135 seq
"556"
127.0.0.1:6379>
```

60) Deleting a field

- **hdel 8868 seq**



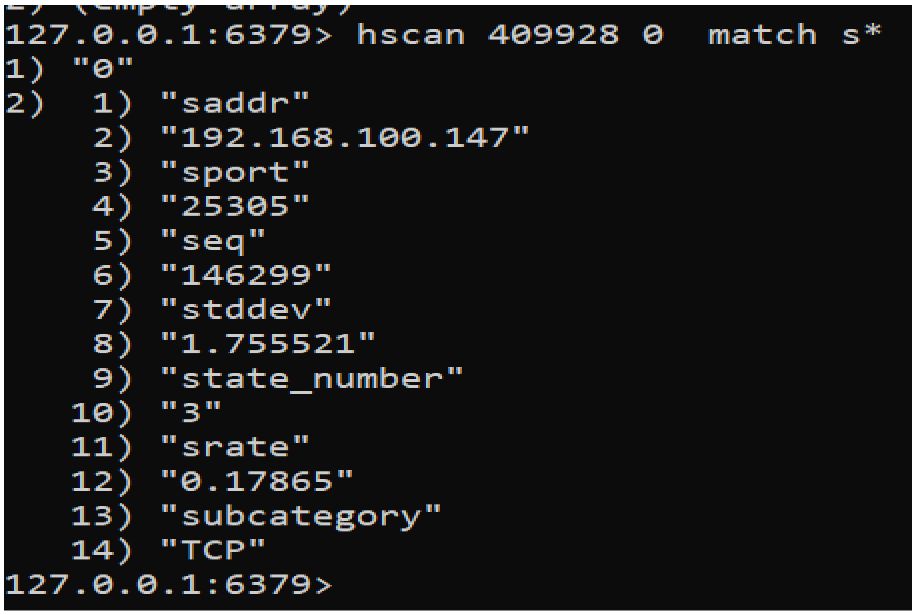
```
127.0.0.1:6379> hdel 8868 seq
(integer) 1
127.0.0.1:6379>
```



```
127.0.0.1:6379> hscan 409928 0
1) "0"
2)  1) "proto"
    2) "tcp"
    3) "saddr"
    4) "192.168.100.147"
    5) "sport"
    6) "25305"
    7) "daddr"
    8) "192.168.100.7"
    9) "dport"
   10) "80"
   11) "seq"
   12) "146299"
   13) "stddev"
   14) "1.755521"
   15) "N_IN_Conn_P_SrcIP"
   16) "100"
   17) "min"
   18) "0.0"
   19) "state_number"
   20) "3"
   21) "mean"
   22) "1.01355"
   23) "N_IN_Conn_P_DstIP"
   24) "100"
   25) "drate"
   26) "0.0"
   27) "srate"
   28) "0.17865"
   29) "max"
   30) "4.054201"
   31) "attack"
   32) "1"
   33) "category"
   34) "DoS"
   35) "subcategory"
   36) "TCP"
```

61) finding all the fields which start with letter s and showing its value

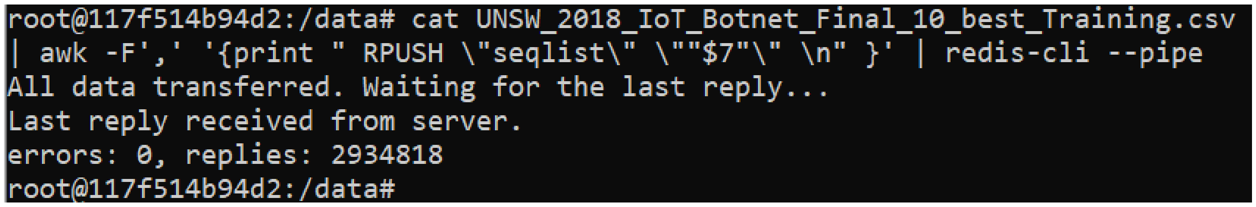
- **hscan 409928 0 match s***



```
127.0.0.1:6379> hscan 409928 0  match s*
1) "0"
2)  1) "saddr"
    2) "192.168.100.147"
    3) "sport"
    4) "25305"
    5) "seq"
    6) "146299"
    7) "stddev"
    8) "1.755521"
    9) "state_number"
   10) "3"
   11) "srate"
   12) "0.17865"
   13) "subcategory"
   14) "TCP"
127.0.0.1:6379>
```

LISTS

62) Upload a column from csv to make a list



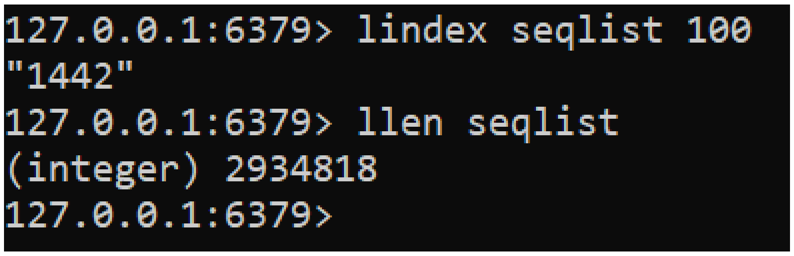
```
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv
| awk -F',' '{print " RPUSH \"seqlist\" \""$7"\" \n" }' | redis-cli --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 2934818
root@117f514b94d2:/data#
```

63) Finding the index of the value of a list

- **lindex seqlist 100**

64) Finding the length of the list

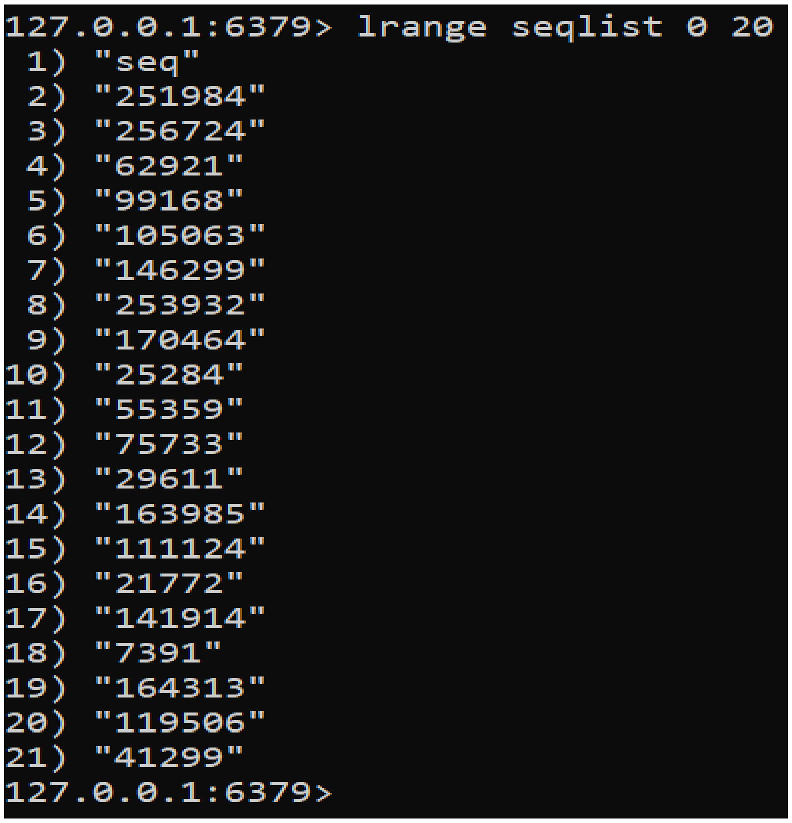
- **llen seqlist**



```
127.0.0.1:6379> lindex seqlist 100
"1442"
127.0.0.1:6379> llen seqlist
(integer) 2934818
127.0.0.1:6379>
```

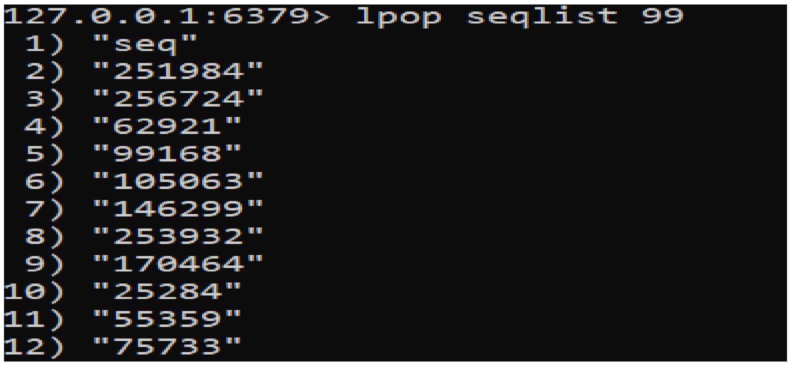
65) Getting all the values of list from index 0 to 20

- **lrange seqlist 0 20**



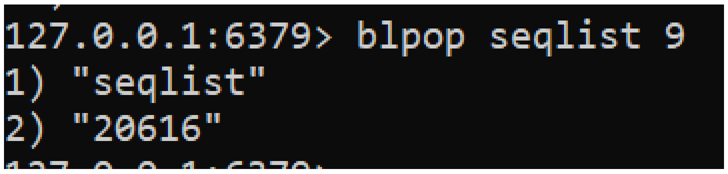
```
127.0.0.1:6379> lrange seqlist 0 20
 1) "seq"
 2) "251984"
 3) "256724"
 4) "62921"
 5) "99168"
 6) "105063"
 7) "146299"
 8) "253932"
 9) "170464"
10) "25284"
11) "55359"
12) "75733"
13) "29611"
14) "163985"
15) "111124"
16) "21772"
17) "141914"
18) "7391"
19) "164313"
20) "119506"
21) "41299"
127.0.0.1:6379>
```

66) Popping removing from left side 99 values



```
127.0.0.1:6379> lpop seqlist 99
 1) "seq"
 2) "251984"
 3) "256724"
 4) "62921"
 5) "99168"
 6) "105063"
 7) "146299"
 8) "253932"
 9) "170464"
10) "25284"
11) "55359"
12) "75733"
```

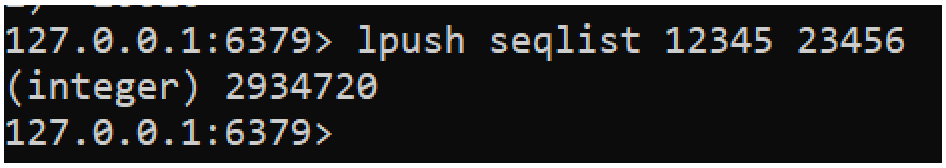
- **blpop seqlist 9**



```
127.0.0.1:6379> blpop seqlist 9
1) "seqlist"
2) "20616"
```

67) pushing(adding) value in the list from left side

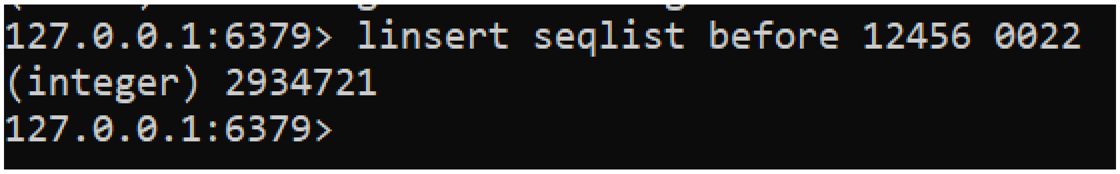
- **lpush seqlist 12345 23456**



```
127.0.0.1:6379> lpush seqlist 12345 23456
(integer) 2934720
127.0.0.1:6379>
```

68) Inserting value in list

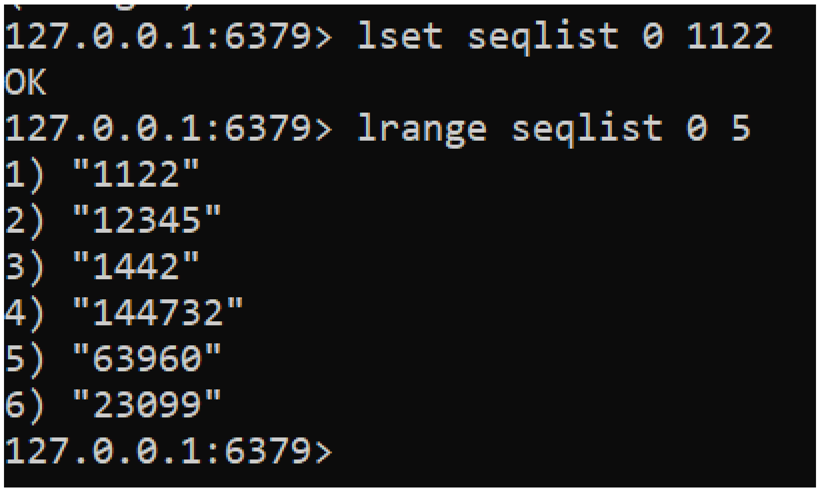
- **linsert seqlist before 12456 0022**



```
127.0.0.1:6379> linsert seqlist before 12456 0022
(integer) 2934721
127.0.0.1:6379>
```

69) Setting/adding a value at 0 index in list

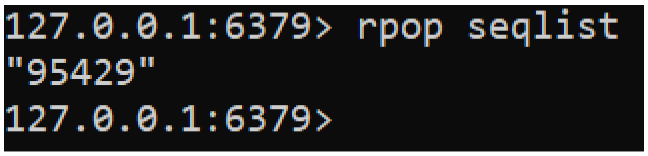
- **lset seqlist 0 1122**



```
127.0.0.1:6379> lset seqlist 0 1122
OK
127.0.0.1:6379> lrange seqlist 0 5
1) "1122"
2) "12345"
3) "1442"
4) "144732"
5) "63960"
6) "23099"
127.0.0.1:6379>
```

70) popping a value from right side of list

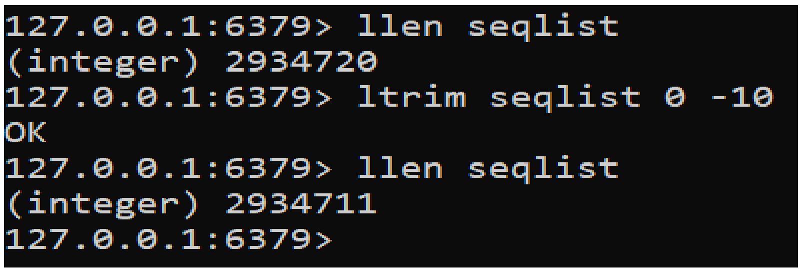
- **rpop seqlist**



```
127.0.0.1:6379> rpop seqlist
"95429"
127.0.0.1:6379>
```

71) removing values of list with the index range from 0 to -10

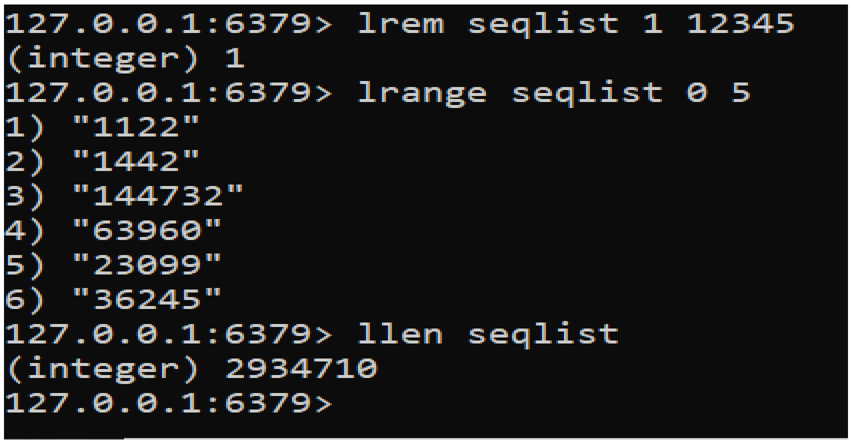
- **ltrim seqlist 0 -10**



```
127.0.0.1:6379> llen seqlist
(integer) 2934720
127.0.0.1:6379> ltrim seqlist 0 -10
OK
127.0.0.1:6379> llen seqlist
(integer) 2934711
127.0.0.1:6379>
```

72) Removing value from a list

- **lrem seqlist 1 12345**

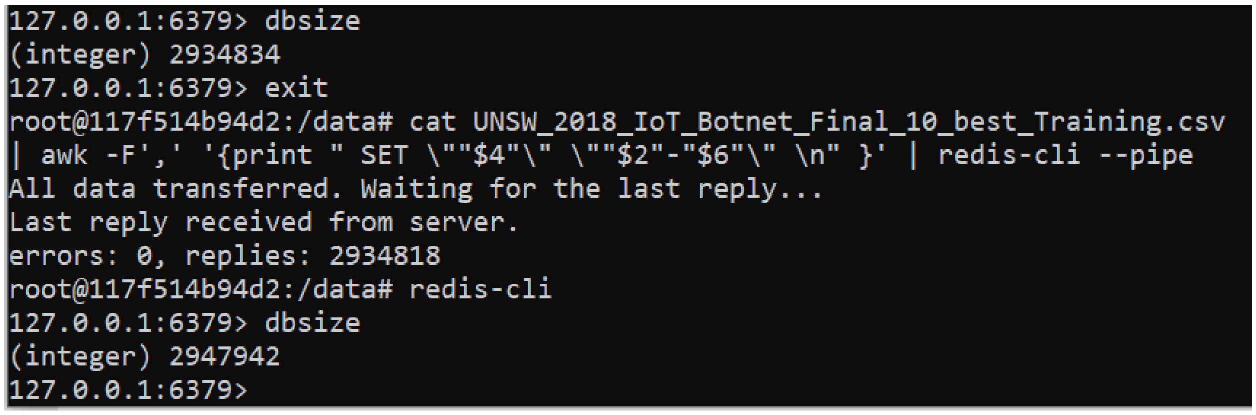


```
127.0.0.1:6379> lrem seqlist 1 12345
(integer) 1
127.0.0.1:6379> lrange seqlist 0 5
1) "1122"
2) "1442"
3) "144732"
4) "63960"
5) "23099"
6) "36245"
127.0.0.1:6379> llen seqlist
(integer) 2934710
127.0.0.1:6379>
```

STRINGS

73) Upload the columns of csv to make a string

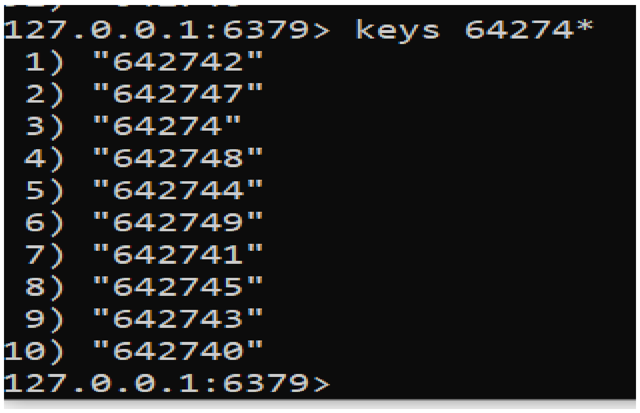
- **cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " SET \""$4"\" \""$2"-"$6"\" \n" }' | redis-cli --pipe**



```
127.0.0.1:6379> dbsize
(integer) 2934834
127.0.0.1:6379> exit
root@117f514b94d2:/data# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv
| awk -F',' '{print " SET \""$4"\" \""$2"-"$6"\" \n" }' | redis-cli --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 2934818
root@117f514b94d2:/data# redis-cli
127.0.0.1:6379> dbsize
(integer) 2947942
127.0.0.1:6379>
```

74) finding all keys starting with 64274

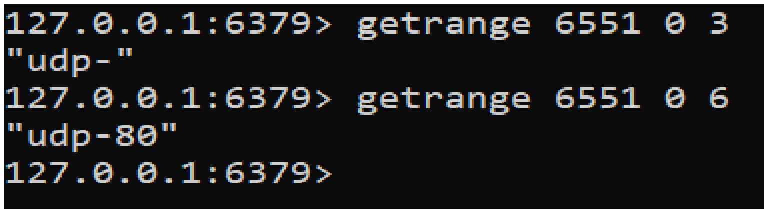
- **keys 64274***



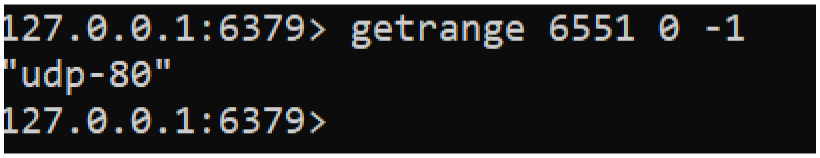
```
127.0.0.1:6379> keys 64274*
 1) "642742"
 2) "642747"
 3) "64274"
 4) "642748"
 5) "642744"
 6) "642749"
 7) "642741"
 8) "642745"
 9) "642743"
10) "642740"
127.0.0.1:6379>
```

75) getting the string from 0 to 3 of a particular key

- **getrange 6551 0 3**



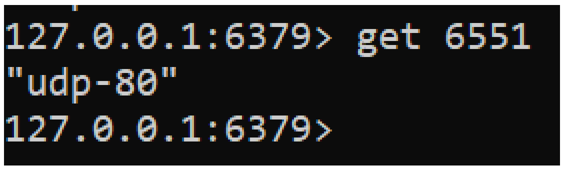
```
127.0.0.1:6379> getrange 6551 0 3
"udp-"
127.0.0.1:6379> getrange 6551 0 6
"udp-80"
127.0.0.1:6379>
```



```
127.0.0.1:6379> getrange 6551 0 -1
"udp-80"
127.0.0.1:6379>
```

76) getting value of a key

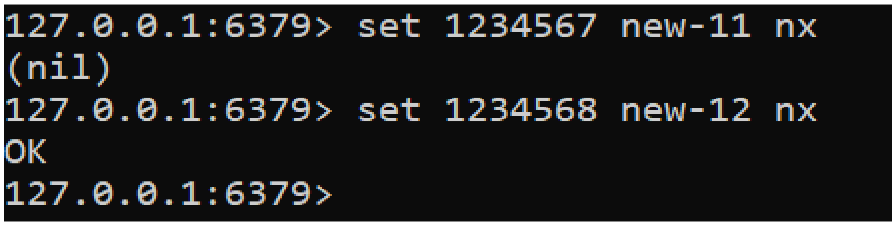
- **get 6551**



```
127.0.0.1:6379> get 6551
"udp-80"
127.0.0.1:6379>
```

77) Making new key if key with same name does not exists

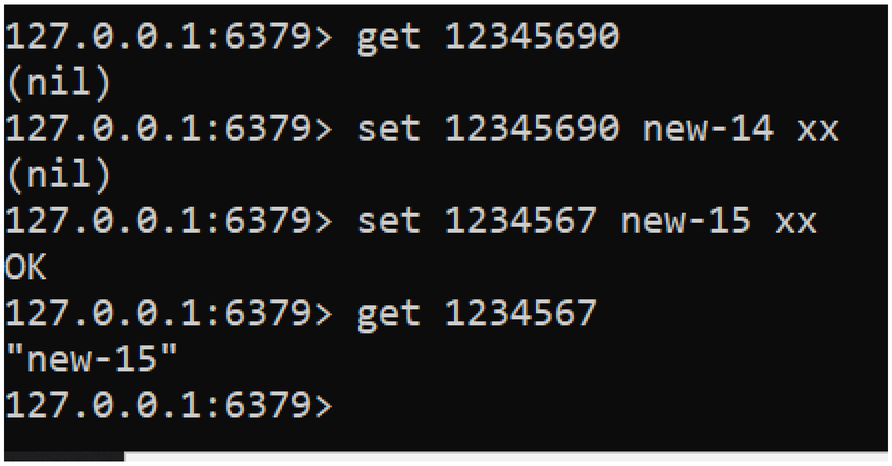
- **set 1234568 new-12 nx**



```
127.0.0.1:6379> set 1234567 new-11 nx
(nil)
127.0.0.1:6379> set 1234568 new-12 nx
OK
127.0.0.1:6379>
```

78) making new key even if key with same name does exists

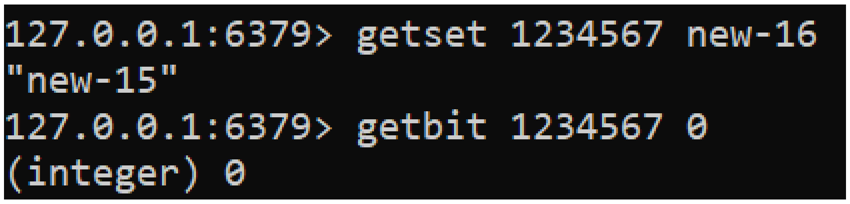
- **set 1234567 new-15 xx**



```
127.0.0.1:6379> get 12345690
(nil)
127.0.0.1:6379> set 12345690 new-14 xx
(nil)
127.0.0.1:6379> set 1234567 new-15 xx
OK
127.0.0.1:6379> get 1234567
"new-15"
127.0.0.1:6379>
```

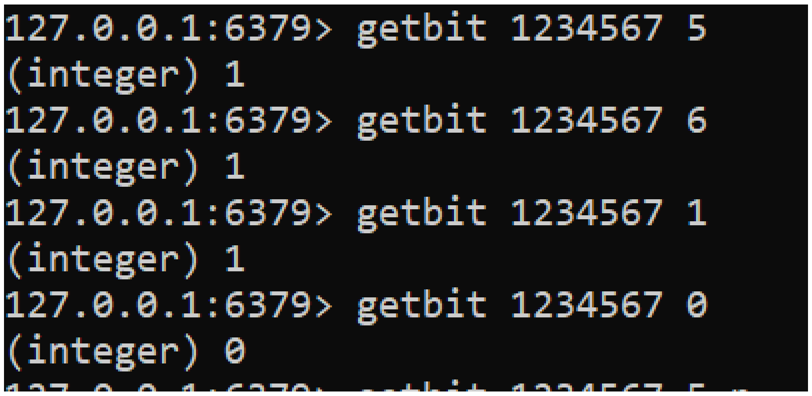
79) getting old value of key and setting new value

- **getset 1234567 new-16**



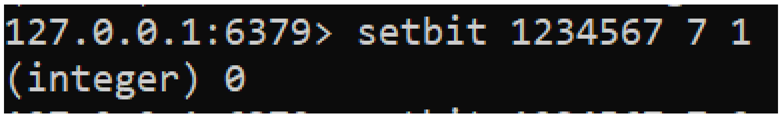
```
127.0.0.1:6379> getset 1234567 new-16
"new-15"
127.0.0.1:6379> getbit 1234567 0
(integer) 0
```

- **getbit 1234567 5**



```
127.0.0.1:6379> getbit 1234567 5
(integer) 1
127.0.0.1:6379> getbit 1234567 6
(integer) 1
127.0.0.1:6379> getbit 1234567 1
(integer) 1
127.0.0.1:6379> getbit 1234567 0
(integer) 0
```

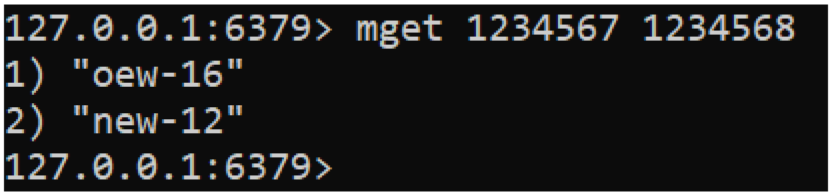
- **setbit 1234567 7 1**



```
127.0.0.1:6379> setbit 1234567 7 1
(integer) 0
```

80) Getting values of multiple keys

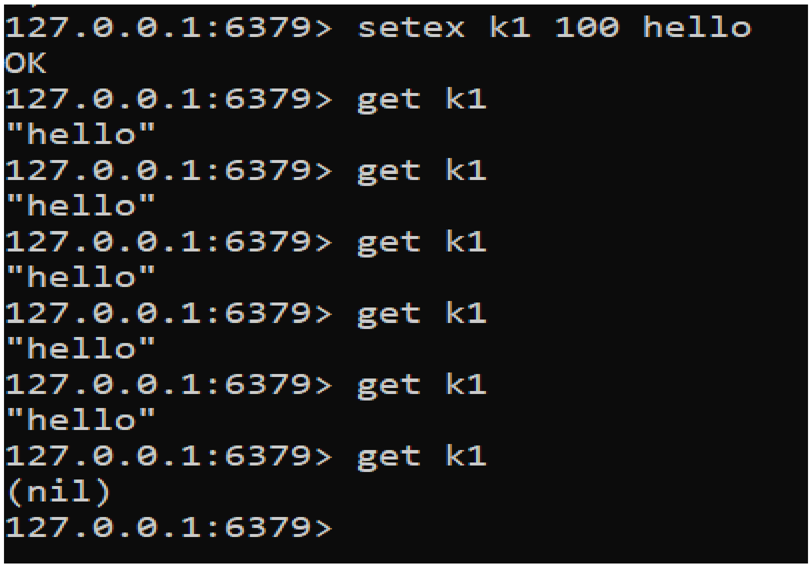
- **mget 1234567 1234568**



```
127.0.0.1:6379> mget 1234567 1234568
1) "oew-16"
2) "new-12"
127.0.0.1:6379>
```

81) making key with expiration time of 100 seconds

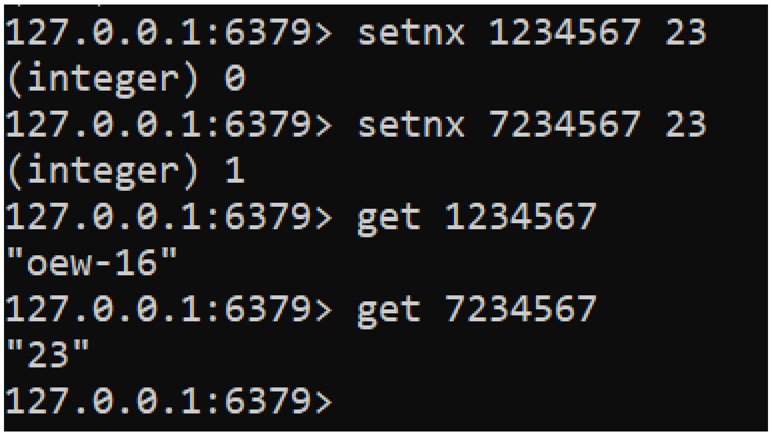
- **setex k1 100 hello**



```
127.0.0.1:6379> setex k1 100 hello
OK
127.0.0.1:6379> get k1
"hello"
127.0.0.1:6379> get k1
"hello"
127.0.0.1:6379> get k1
"hello"
127.0.0.1:6379> get k1
"hello"
127.0.0.1:6379> get k1
"hello"
127.0.0.1:6379> get k1
(nil)
127.0.0.1:6379>
```

82) Making key if it does not exists

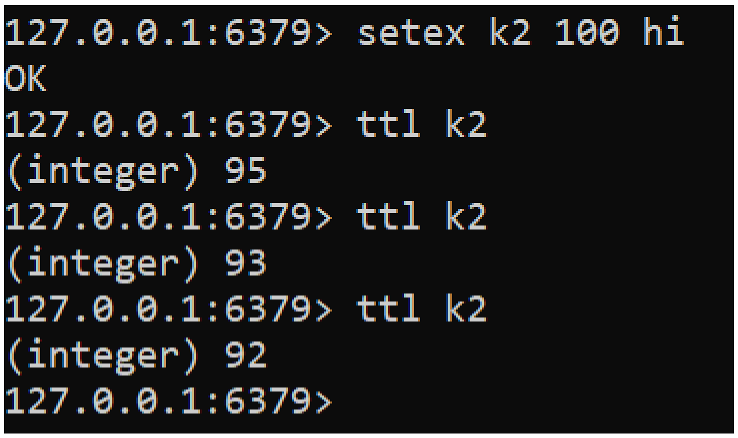
- **setnx 7234567 23**



```
127.0.0.1:6379> setnx 1234567 23
(integer) 0
127.0.0.1:6379> setnx 7234567 23
(integer) 1
127.0.0.1:6379> get 1234567
"oew-16"
127.0.0.1:6379> get 7234567
"23"
127.0.0.1:6379>
```

83) Finding the remaining expiration time of key

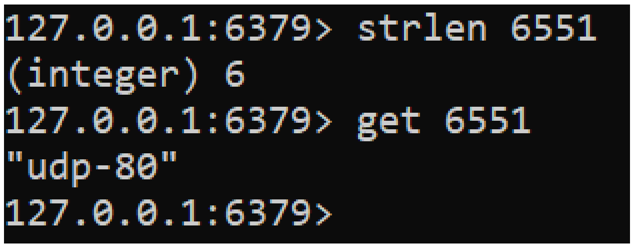
- **ttl k2**



```
127.0.0.1:6379> setex k2 100 hi
OK
127.0.0.1:6379> ttl k2
(integer) 95
127.0.0.1:6379> ttl k2
(integer) 93
127.0.0.1:6379> ttl k2
(integer) 92
127.0.0.1:6379>
```

84) Finding the length of the string of a particular key

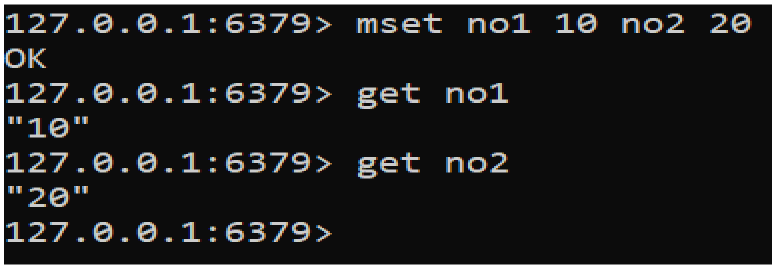
- **strlen 6551**



```
127.0.0.1:6379> strlen 6551
(integer) 6
127.0.0.1:6379> get 6551
"udp-80"
127.0.0.1:6379>
```

85) making multiple keys at same time

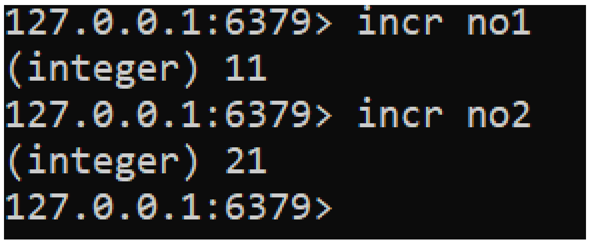
- **mset no1 10 no2 20**



```
127.0.0.1:6379> mset no1 10 no2 20
OK
127.0.0.1:6379> get no1
"10"
127.0.0.1:6379> get no2
"20"
127.0.0.1:6379>
```

86) Increasing the key value by 1

- **incr no1**



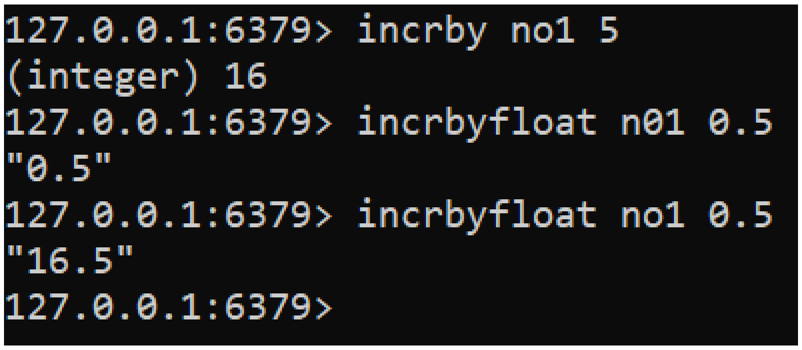
```
127.0.0.1:6379> incr no1
(integer) 11
127.0.0.1:6379> incr no2
(integer) 21
127.0.0.1:6379>
```

87) Increase key value by float

- **incrbyfloat no1 0.5**

88) Increase key value by 15

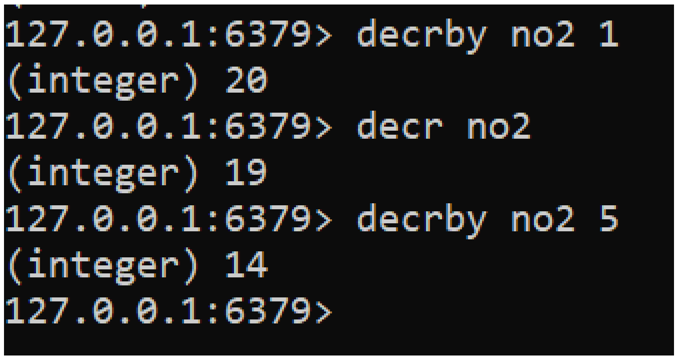
- **incrby no1 5**



```
127.0.0.1:6379> incrby no1 5
(integer) 16
127.0.0.1:6379> incrbyfloat n01 0.5
"0.5"
127.0.0.1:6379> incrbyfloat no1 0.5
"16.5"
127.0.0.1:6379>
```

89) decreasing key value by 1

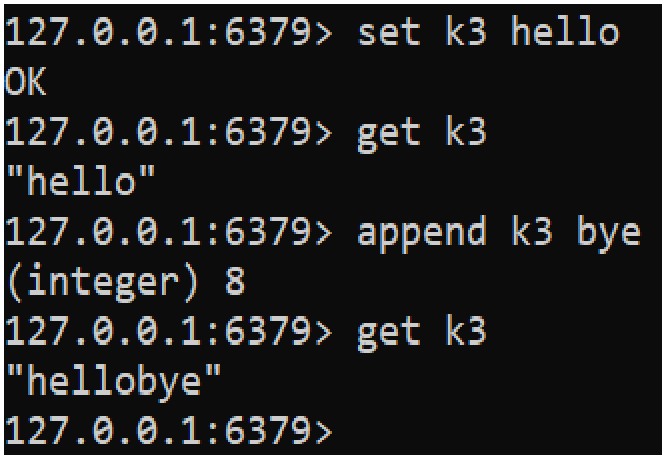
- **decrby no2 1**



```
127.0.0.1:6379> decrby no2 1
(integer) 20
127.0.0.1:6379> decr no2
(integer) 19
127.0.0.1:6379> decrby no2 5
(integer) 14
127.0.0.1:6379>
```

90) Appending the value in existing key

- **append k3 bye**



```
127.0.0.1:6379> set k3 hello
OK
127.0.0.1:6379> get k3
"hello"
127.0.0.1:6379> append k3 bye
(integer) 8
127.0.0.1:6379> get k3
"hellobye"
127.0.0.1:6379>
```

TRANSACTIONS

MULTI & EXEC

Command run in 2 terminals

91) Creating a key with its value

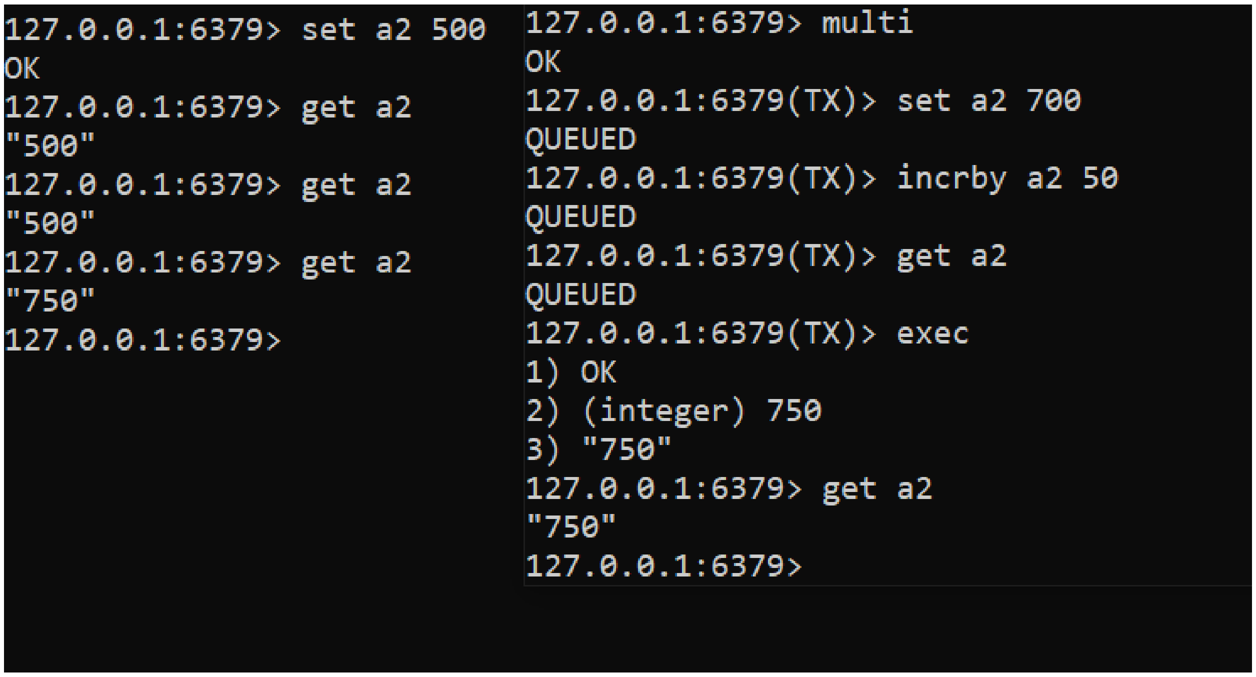
Terminal 1

- **127.0.0.1:6379> set a2 500**
- **OK**
- **127.0.0.1:6379> get a2**
- **"500"**
- **127.0.0.1:6379> get a2**
- **"500"**
- **127.0.0.1:6379> get a2**
- **"750"**
- **127.0.0.1:6379>**

92) Creating a transaction

Terminal 2

- **127.0.0.1:6379> multi**
- **OK**
- **127.0.0.1:6379(TX)> set a2 700**
- **QUEUED**
- **127.0.0.1:6379(TX)> incrby a2 50**
- **QUEUED**
- **127.0.0.1:6379(TX)> get a2**
- **QUEUED**
- **127.0.0.1:6379(TX)> exec**
- **OK**
- **(integer) 750**
- **3) "750"**
- **127.0.0.1:6379> get a2**
- **"750"**
- **127.0.0.1:6379>**



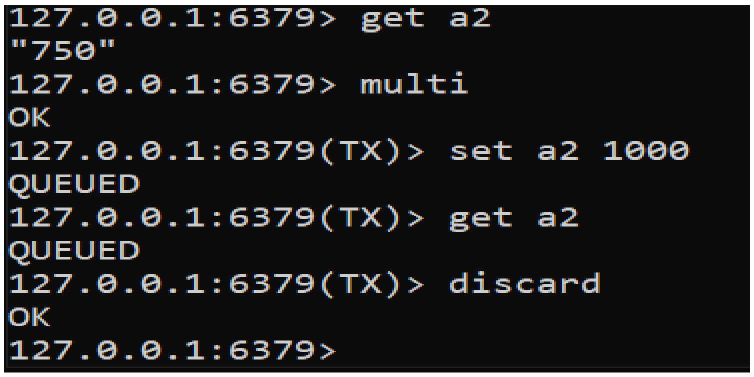
```
127.0.0.1:6379> set a2 500
OK
127.0.0.1:6379> get a2
"500"
127.0.0.1:6379> get a2
"500"
127.0.0.1:6379> get a2
"750"
127.0.0.1:6379>
```

```
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> set a2 700
QUEUED
127.0.0.1:6379(TX)> incrby a2 50
QUEUED
127.0.0.1:6379(TX)> get a2
QUEUED
127.0.0.1:6379(TX)> exec
1) OK
2) (integer) 750
3) "750"
127.0.0.1:6379> get a2
"750"
127.0.0.1:6379>
```

Discard

93) Now using discard which can end a transaction

- **127.0.0.1:6379> get a2**
- **"750"**
- **127.0.0.1:6379> multi**
- **OK**
- **127.0.0.1:6379(TX)> set a2 1000**
- **QUEUED**
- **127.0.0.1:6379(TX)> get a2**
- **QUEUED**
- **127.0.0.1:6379(TX)> discard**
- **OK**
- **127.0.0.1:6379>**



```
127.0.0.1:6379> get a2
"750"
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> set a2 1000
QUEUED
127.0.0.1:6379(TX)> get a2
QUEUED
127.0.0.1:6379(TX)> discard
OK
127.0.0.1:6379>
```

WATCH

94) Creating 2 keys to be watched

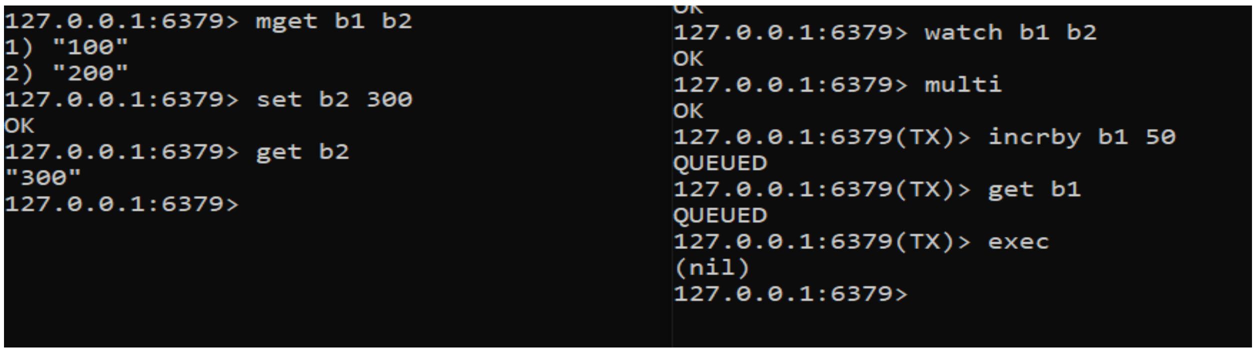
Terminal 1

- **127.0.0.1:6379> mset b1 100 b2 200**
- **OK**
- **127.0.0.1:6379> mget b1 b2**
- **"100"**
- **"200"**
- **127.0.0.1:6379> set b2 300**
- **OK**
- **127.0.0.1:6379> get b2**
- **"300"**
- **127.0.0.1:6379>**

95) Executing the watch in transaction

Terminal 2

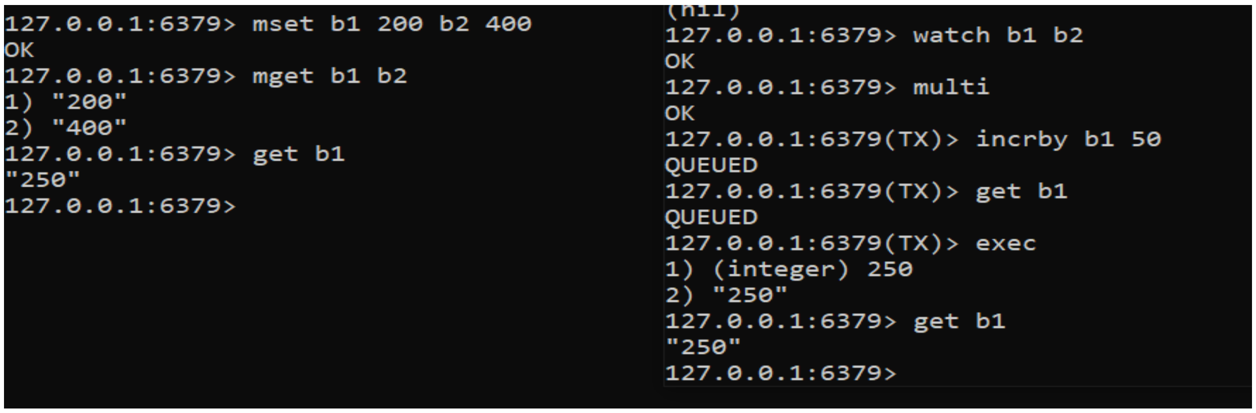
- **127.0.0.1:6379> watch b1 b2**
- **OK**
- **127.0.0.1:6379> multi**
- **OK**
- **127.0.0.1:6379(TX)> incrby b1 50**
- **QUEUED**
- **127.0.0.1:6379(TX)> get b1**
- **QUEUED**
- **127.0.0.1:6379(TX)> exec**
- **(nil)**
- **127.0.0.1:6379>**



```
127.0.0.1:6379> mget b1 b2
1) "100"
2) "200"
127.0.0.1:6379> set b2 300
OK
127.0.0.1:6379> get b2
"300"
127.0.0.1:6379>
```

```
127.0.0.1:6379> watch b1 b2
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> incrby b1 50
QUEUED
127.0.0.1:6379(TX)> get b1
QUEUED
127.0.0.1:6379(TX)> exec
(nil)
127.0.0.1:6379>
```

Transaction not worked/executed as b2 value was changed during transaction



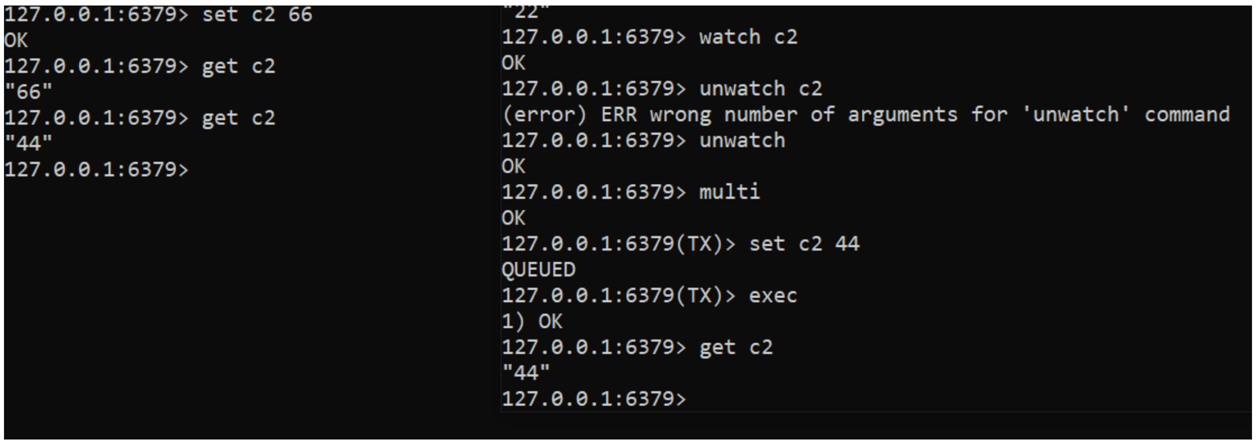
```
127.0.0.1:6379> mset b1 200 b2 400
OK
127.0.0.1:6379> mget b1 b2
1) "200"
2) "400"
127.0.0.1:6379> get b1
"250"
127.0.0.1:6379>
```

```
127.0.0.1:6379> watch b1 b2
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> incrby b1 50
QUEUED
127.0.0.1:6379(TX)> get b1
QUEUED
127.0.0.1:6379(TX)> exec
1) (integer) 250
2) "250"
127.0.0.1:6379> get b1
"250"
127.0.0.1:6379>
```

Now executed as no changes in b1 and b2

Unwatch

96) Executing unwatch



```
127.0.0.1:6379> set c2 66
OK
127.0.0.1:6379> get c2
"66"
127.0.0.1:6379> get c2
"44"
127.0.0.1:6379>
```

```
127.0.0.1:6379> watch c2
OK
127.0.0.1:6379> unwatch c2
(error) ERR wrong number of arguments for 'unwatch' command
127.0.0.1:6379> unwatch
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> set c2 44
QUEUED
127.0.0.1:6379(TX)> exec
1) OK
127.0.0.1:6379> get c2
"44"
127.0.0.1:6379>
```

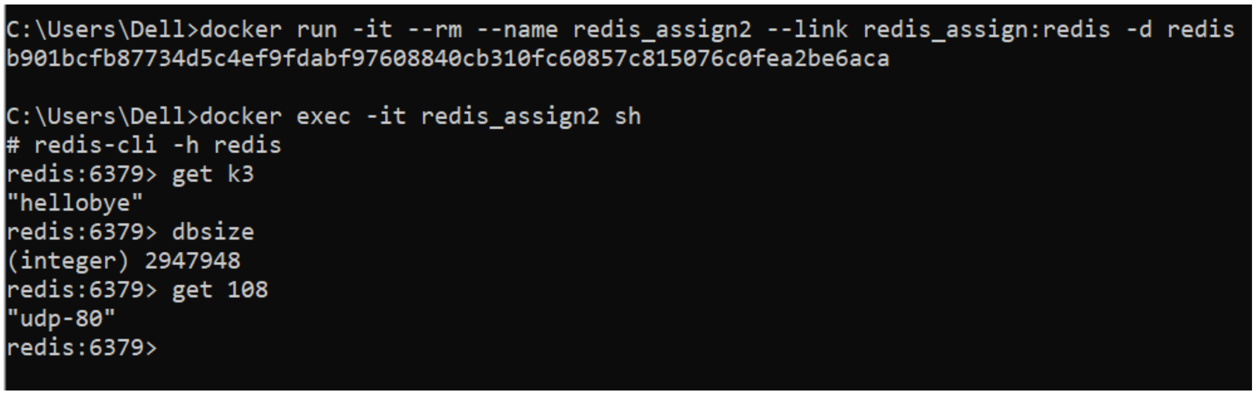
LINKING TWO REDIS CONTAINERS

97) Making a new redis container

- **docker run -it --rm --name redis_assign2 --link redis_assign:redis -d redis**
- **docker exec -it redis_assign2 sh**

98) Connecting container to old conatiner

- **redis-cli -h redis**
- **get k3**



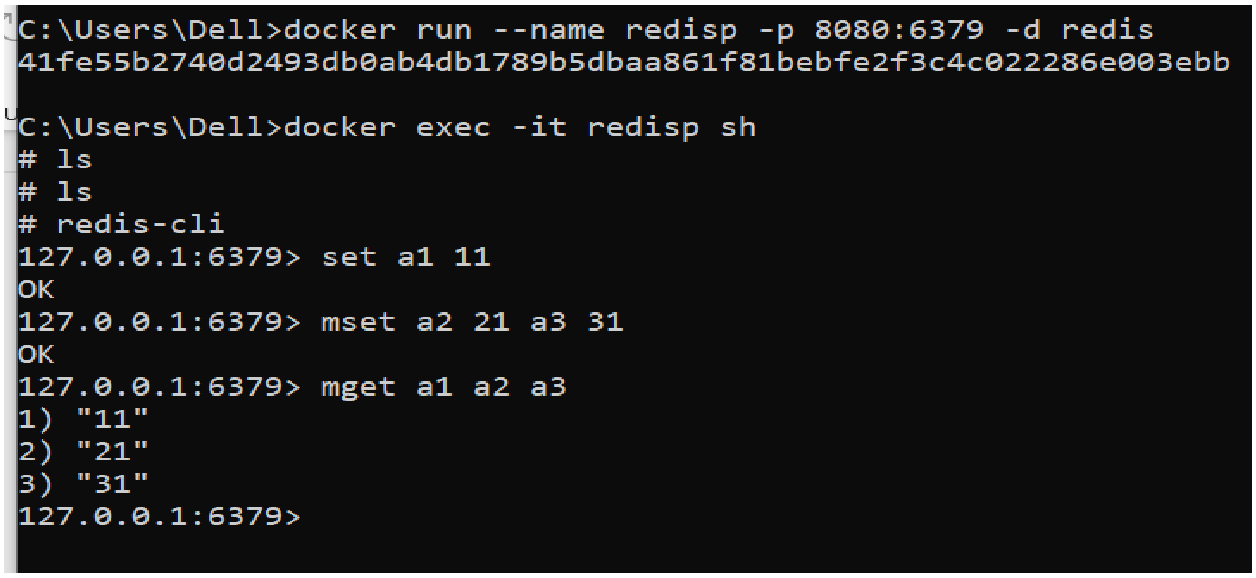
```
C:\Users\Dell>docker run -it --rm --name redis_assign2 --link redis_assign:redis -d redis
b901bcfb87734d5c4ef9fdabf97608840cb310fc60857c815076c0fea2be6aca

C:\Users\Dell>docker exec -it redis_assign2 sh
# redis-cli -h redis
redis:6379> get k3
"hellobye"
redis:6379> dbsize
(integer) 2947948
redis:6379> get 108
"udp-80"
redis:6379>
```

Access Redis from Remote Server

99) Making a new redis container with the port number specified

Terminal 1

- **docker run --name redisp -p 8080:6379 -d redis**

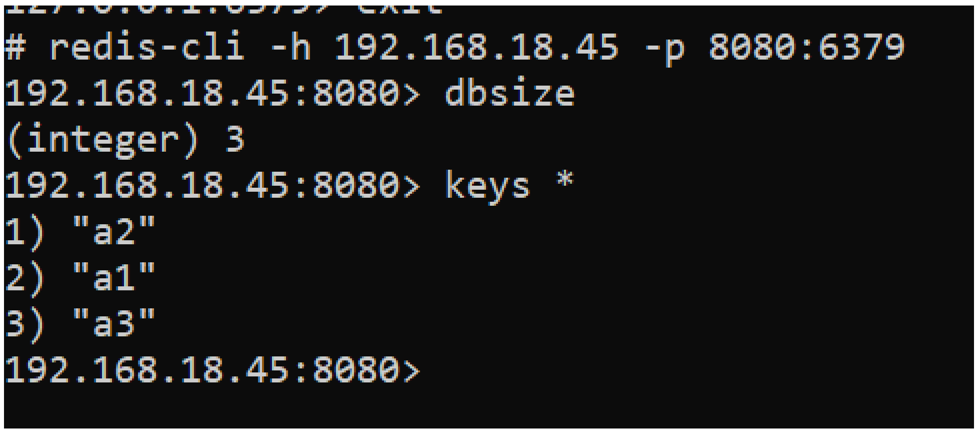


```
C:\Users\Dell>docker run --name redisp -p 8080:6379 -d redis
41fe55b2740d2493db0ab4db1789b5dbaa861f81bebfe2f3c4c022286e003ebb

C:\Users\Dell>docker exec -it redisp sh
# ls
# ls
# redis-cli
127.0.0.1:6379> set a1 11
OK
127.0.0.1:6379> mset a2 21 a3 31
OK
127.0.0.1:6379> mget a1 a2 a3
1) "11"
2) "21"
3) "31"
127.0.0.1:6379>
```

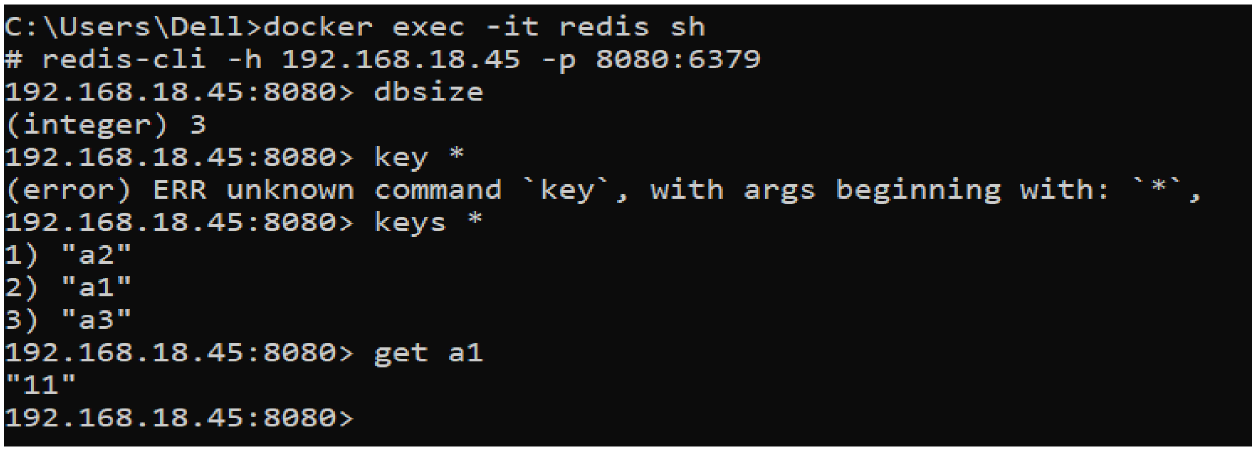
100) Connecting the old container to a remote redis database

Terminal 2

- redis-cli -h 192.168.18.45 -p 8080:6379



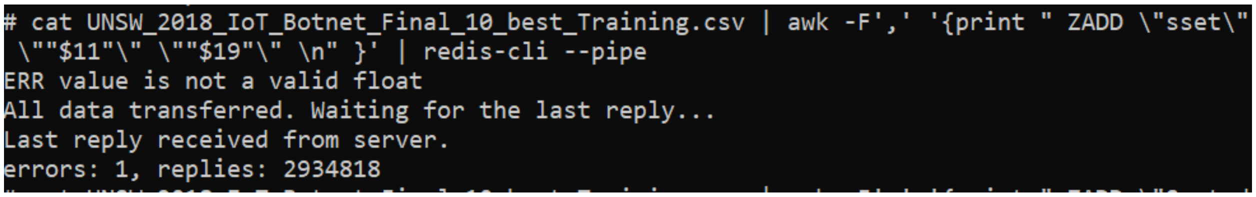
```
# redis-cli -h 192.168.18.45 -p 8080:6379
192.168.18.45:8080> dbsize
(integer) 3
192.168.18.45:8080> keys *
1) "a2"
2) "a1"
3) "a3"
192.168.18.45:8080>
```



```
C:\Users\Dell>docker exec -it redis sh
# redis-cli -h 192.168.18.45 -p 8080:6379
192.168.18.45:8080> dbsize
(integer) 3
192.168.18.45:8080> key *
(error) ERR unknown command `key`, with args beginning with: `*`,
192.168.18.45:8080> keys *
1) "a2"
2) "a1"
3) "a3"
192.168.18.45:8080> get a1
"11"
192.168.18.45:8080>
```

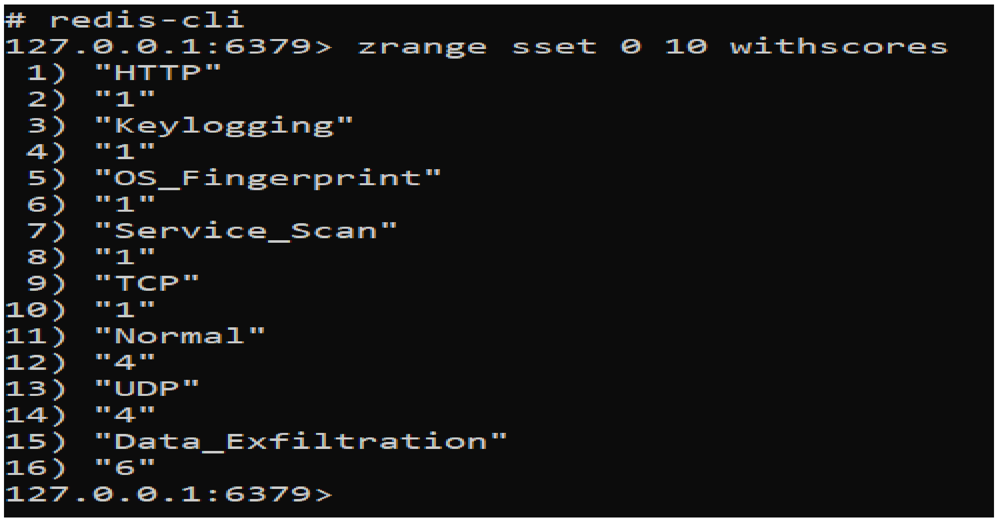
101) Uplaoding a column to make a new sorted set

- **cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " ZADD \"sset\" \""$11"\" \""$19"\" \n" }' | redis-cli –pipe**



```
# cat UNSW_2018_IoT_Botnet_Final_10_best_Training.csv | awk -F',' '{print " ZADD \"sset\"
 \""$11"\" \""$19"\" \n" }' | redis-cli --pipe
ERR value is not a valid float
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 1, replies: 2934818
```

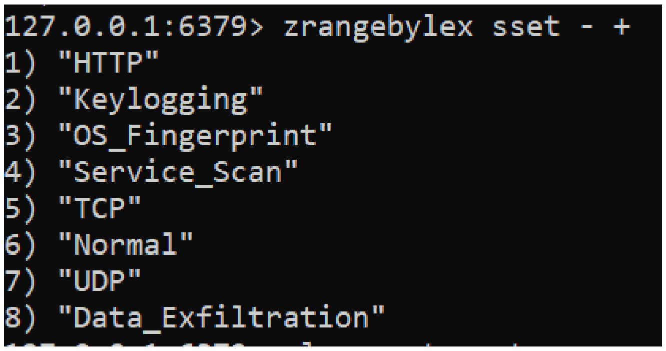
102) Checking the new sorted set



```
# redis-cli
127.0.0.1:6379> zrange sset 0 10 withscores
 1) "HTTP"
 2) "1"
 3) "Keylogging"
 4) "1"
 5) "OS_Fingerprint"
 6) "1"
 7) "Service_Scan"
 8) "1"
 9) "TCP"
10) "1"
11) "Normal"
12) "4"
13) "UDP"
14) "4"
15) "Data_Exfiltration"
16) "6"
127.0.0.1:6379>
```

103) Using Zrangelex to sort same score values

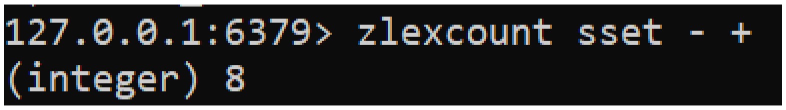
- **zrangebylex sset - +**



```
127.0.0.1:6379> zrangebylex sset - +
1) "HTTP"
2) "Keylogging"
3) "OS_Fingerprint"
4) "Service_Scan"
5) "TCP"
6) "Normal"
7) "UDP"
8) "Data_Exfiltration"
```

104) Using zlex count

- **zlexcount sset - +**



```
127.0.0.1:6379> zlexcount sset - +
(integer) 8
```

PUBLISH & SUBSCRIBE

105) Subscribing to ch-1

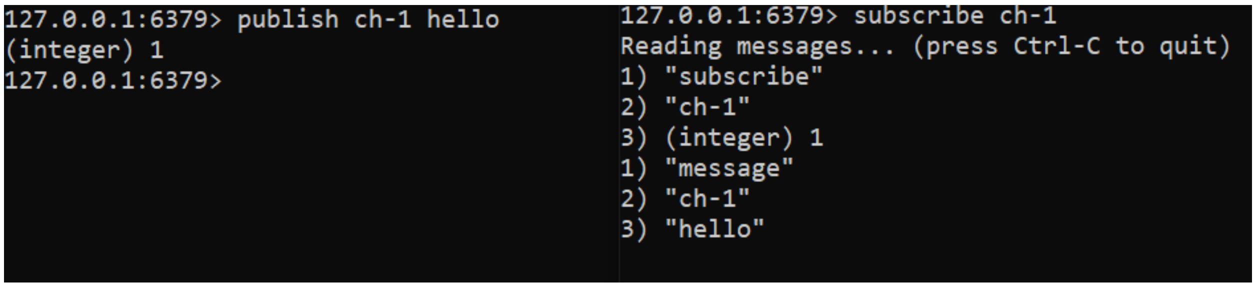
Terminal 1

- **subscribe ch-1**

106) Publishing to channel 1

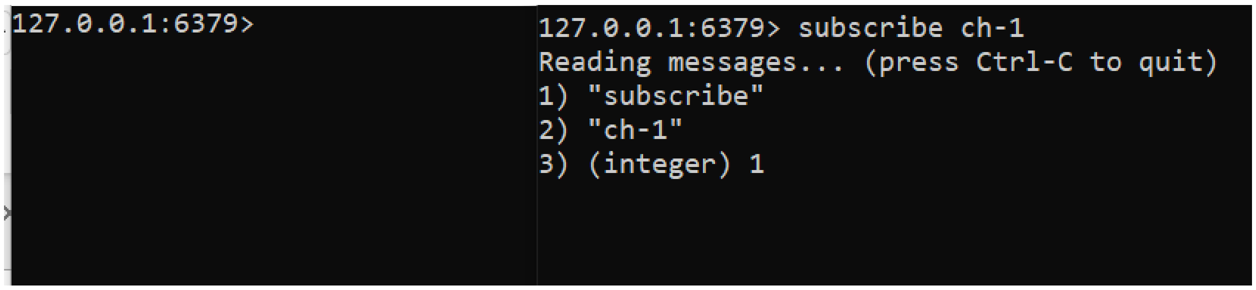
Terminal 2

- **publish ch-1 hello**



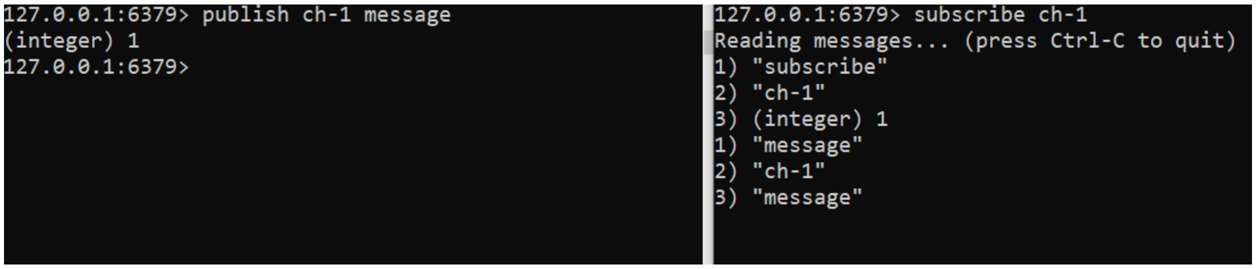
```
127.0.0.1:6379> publish ch-1 hello
(integer) 1
127.0.0.1:6379>
```

```
127.0.0.1:6379> subscribe ch-1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "ch-1"
3) (integer) 1
1) "message"
2) "ch-1"
3) "hello"
```



```
127.0.0.1:6379>
```

```
127.0.0.1:6379> subscribe ch-1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "ch-1"
3) (integer) 1
```

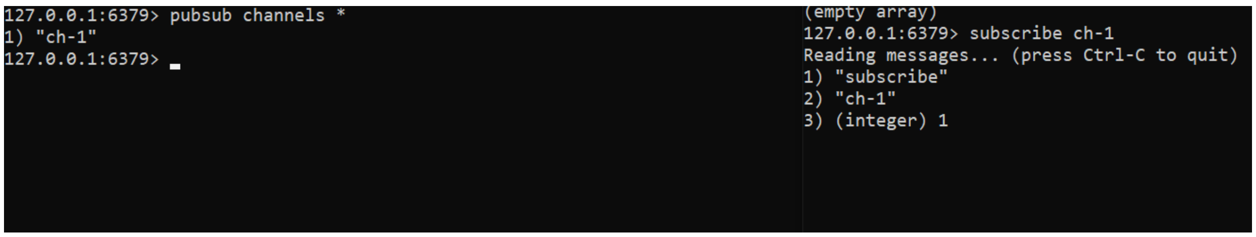


```
127.0.0.1:6379> publish ch-1 message
(integer) 1
127.0.0.1:6379>
```

```
127.0.0.1:6379> subscribe ch-1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "ch-1"
3) (integer) 1
1) "message"
2) "ch-1"
3) "message"
```

107) Finding how many channels currently subscribed and active

- **pubsub channels ***



```
127.0.0.1:6379> pubsub channels *
1) "ch-1"
127.0.0.1:6379>
```

```
(empty array)
127.0.0.1:6379> subscribe ch-1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "ch-1"
3) (integer) 1
```