```
# ****************************************************************************
# BASICS
# ****************************************************************************

redis-server /path/redis.conf        # start redis with the related configuration file
redis-cli                            # opens a redis prompt
sudo systemctl restart redis.service # Restart Redis
sudo systemctl status redis          # Check Redis status

# ****************************************************************************
# STRINGS
# ****************************************************************************

APPEND key value                  # append a value to a key
BITCOUNT key [start end]          # count set bits in a string
SET key value                     # set value in key
SETNX key value                   # set if not exist value in key
SETRANGE key offset value         # overwrite part of a string at key starting at the specified
offset
STRLEN key                        # get the length of the value stored in a key
MSET key value [key value ...]    # set multiple keys to multiple values
MSETNX key value [key value ...]  # set multiple keys to multiple values, only if none of the keys
exist
GET key                           # get value in key
GETRANGE key start end            # get substring of stored value from start to end offsets (both
inclusive)
MGET key [key ...]                # get the values of all the given keys
INCR key                          # increment value in key
INCRBY key increment              # increment the integer value of a key by the given amount
INCRBYFLOAT key increment         # increment the float value of a key by the given amount
DECR key                          # decrement the integer value of key by one
DECRBY key decrement              # decrement the integer value of a key by the given number
DEL key                           # delete key

EXPIRE key 120                    # key will be deleted in 120 seconds
TTL key                           # returns the number of seconds until a key is deleted


# ****************************************************************************
# LISTS
# A list is a series of ordered values
# ****************************************************************************

RPUSH key value [value ...]          # put the new value at the end of the list
RPUSHX key value                     # append a value at the end of the list, only if it exists
LPUSH key value [value ...]          # put the new value at the start of the list
LPUSHX key value                     # append a value at the start of the list, only if it exists
LRANGE key start stop                # give a subset of the list
LINDEX key index                     # get an element from a list by its index
LINSERT key BEFORE|AFTER pivot value  # insert an element before or after another element
in a list
```

```
LLEN key                        # return the current length of the list
LPOP key                        # remove the first element from the list and returns it
LSET key index value            # set the value of an element in a list by its index
LREM key number_of_occurrences value  # delete occurrences of value if the list stored in key
LTRIM key start stop            # trim a list to the specified range
RPOP key                        # remove the last element from the list and returns it
RPOPLPUSH source destination    # remove the last element in a list, prepend it to another
list and return it
BLPOP key [key ...] timeout     # remove and get the first element in a list, or block until one
is available
BRPOP key [key ...] timeout     # remove and get the last element in a list, or block until one
is available


# ********************************************************************************
# SETS
# A set is similar to a list, except it does not have a specific order
# and each element may only appear once.
# ********************************************************************************


SADD key member [member ...]    # add the given value to the set
SCARD key                       # get the number of members in a set
SREM key member [member ...]    # remove the given value from the set
SISMEMBER myset value           # test if the given value is in the set.
SMEMBERS myset                  # return a list of all the members of this set
SUNION key [key ...]            # combine two or more sets and returns the list of all elements
SINTER key [key ...]            # intersect multiple sets
SMOVE source destination member  # move a member from one set to another
SPOP key [count]                # remove and return one or multiple random members from a set


# ********************************************************************************
# SORTED SETS
# A sorted set is similar to a regular set, but now each value has an associated score.
# This score is used to sort the elements in the set.
# ********************************************************************************


ZADD key [NX|XX] [CH] [INCR] score member [score member ...]  # add one or more members
to a sorted set, or update its score if it already exists

ZCARD key                       # get the number of members in a sorted set
ZCOUNT key min max              # count the members in a sorted set with scores within the
given values
ZINCRBY key increment member    # increment the score of a member in a sorted set
ZRANGE key start stop [WITHSCORES]  # returns a subset of the sorted set
ZRANK key member                # determine the index of a member in a sorted set
ZREM key member [member ...]    # remove one or more members from a sorted set
ZREMRANGEBYRANK key start stop  # remove all members in a sorted set within the given
indexes
ZREMRANGEBYSCORE key min max    # remove all members in a sorted set, by index, with
scores ordered from high to low
```

```
ZSCORE key member                  # get the score associated with the given mmeber in a sorted set

ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]  # return a range of members in a sorted set, by score


# ********************************************************************************
# HASHES
# Hashes are maps between string fields and string values,
# so they are the perfect data type to represent objects.
# ********************************************************************************


HGET key field          # get the value of a hash field
HGETALL key             # get all the fields and values in a hash
HSET key field value    # set the string value of a hash field
HSETNX key field value  # set the string value of a hash field, only if the field does not exists

HMSET key field value [field value ...]  # set multiple fields at once

HINCRBY key field increment  # increment value in hash by X
HDEL key field [field ...]   # delete one or more hash fields
HEXISTS key field            # determine if a hash field exists
HKEYS key                    # get all the fields in a hash
HLEN key                     # get the number of fields in a hash
HSTRLEN key field            # get the length of the value of a hash field
HVALS key                    # get all the values in a hash


# ********************************************************************************
# HYPERLOGLOG
# HyperLogLog uses randomization in order to provide an approximation of the number
# of unique elements in a set using just a constant, and small, amount of memory
# ********************************************************************************


PFADD key element [element ...]  # add the specified elements to the specified HyperLogLog
PFCOUNT key [key ...]            # return the approximated cardinality of the set(s) observed by the HyperLogLog at key's)

PFMERGE destkey sourcekey [sourcekey ...]  # merge N HyperLogLogs into a single one


# ********************************************************************************
# PUBLICATION & SUBSCRIPTION
# ********************************************************************************


PSUBSCRIBE pattern [pattern ...]           # listen for messages published to channels matching the given patterns
PUBSUB subcommand [argument [argument ...]]  # inspect the state of the Pub/Sub subsystem
PUBLISH channel message                    # post a message to a channel
```

```
PUNSUBSCRIBE [pattern [pattern ...]]        # stop listening for messages posted to channels matching the given patterns
SUBSCRIBE channel [channel ...]             # listen for messages published to the given channels
UNSUBSCRIBE [channel [channel ...]]         # stop listening for messages posted to the given channels


# ************************************************************************
# OTHER COMMANDS
# ************************************************************************


KEYS pattern  # find all keys matching the given pattern
```