REDIS CLI

REDIS-SERVER :
→ sudo apt update – update repository
→ sudo apt install redis-server – install redis server
→redis-server – start redis server
→ps -ef | grep redis –check redis server status

REDIS-CLI :
→ sudo apt install redis - install redis cli
→ redis-cli --version -check version
→ sudo systemctl status redis -check status of redis-server
→ sudo systemctl start redis-server -start redis server
→ sudo systemctl stop redis -stop redis server

Use redis-cli
→ redis-cli +enter
127.0.0.1:6379>write command here

String command →
Set key value → set name chail → it will set value of name as chail
Get name → return the value of key
set email chail@gmail.com
Get email
Getrange email 0 4 →return first 5 charactor of email
Mset key value key value key value → set multiple key value pair
Mset name chail email chail@gmail.com
Mget name email
Strlen name → return length of string
Set name "awantika" → update name from chail to awantika
Strlen key → if key is not present then length will be return as 0

Number →
Set count 1
Get count
Incr count → increment by 1
Incrby count 10 → increment by 10
decr count → decrement by 1
decrby count 10 → decrement by 10
Set pi 3.14
Incrbyfloat pi 0.001
Delete or expire value after some time → Set a 1
→ Get a
Expire a 10 → after 10 sec valueof a will be deleted.
Ttl a → return how much time remains to deleted.
Get a → null after 10 sec
Setex b 10 anyvalue → valueof b would be expired after 10 sec

Redis List →

→ keys * – return all key
→ flushall – remove all the key-value pair from redis
→ lpush country india –add key value in list from top
→ lpush country usa
→ lrange country 0 -1 –return all the list value
→ lpush country uk
→ lrange country 0 1 –return only 0th to 1st position list value
→ rpush country india –add key value in list from bottom
→ llen county –length of list
→ llen key – if the key is not present then it will return 0

→ lpop country –remove key value in list from top
→ rpop country –remove key value in list from bottom
→ lrange country 0 -1 –return all the list value
→ lset country 1 srilanka → 1st index element update as srilanka
→ linsert country before china "india" – set india before china
→ linsert country after china "india" – set india after china
→ lindex country 3 → return country at index 3
→ lpushx movies avenger – since movies list is not present so return 0
→ lpush country "korea" –return length of the list
→ rpushx movies avenger – since movies list is not present so return 0
→ rpush country "korea" –return length of the list
→ sort country ALPHA – sort data alphabetical
→ sort country desc ALPHA – sort data alphabetical inn desc order
→ blpop movies 20 – it will wait for 20 sec to remove the element.
→ brpop movies 20 – it will wait for 20 sec to remove the element.

Redis Set →unique value

→ sadd key member – add member in the set
→ sadd key m1 m2 m3 – add multiple member
→ smembers key – return all members in the set
→ scard key – count member in the key set
→ sismember key member – if return 1 them member is available else return 0
→ sdiff key1 key2 – return difference between two set key1 and key2
→ sdiffstore newset key1 key2 → difference of both set would be stored in newset
→ sinter key1 key2 – return intersection of both set or common member return
→ sinterstore newset key1 key2 → common member of both set would be stored in newset
→ sunion key1 key2 – return all data from two set key1 and key2
→ sunionstore newset key1 key2 – store all data from both set into newset

Sorted Set →
→ zadd set score value –value sorted according to given score.
→ zadd users 1 chail
→ zadd users 2 michel
→ zadd users 3 komal 4 gaju
→ zrange users 0 -1
1 chail
2 michel
3 komal
4 gaju

→ zrange users 0 -1 withscores –return value with scores
→ zcard users – available users count
→ zcount users-inf +inf –return count from score -infinity to +infinity
→ zcount users 0 3 –return count
→ zrem users chail –chail will be removed from users set
→ zrevrange users 0 -1 with scores – return value from reverse
→ zscore users michel – return score of michel
→ zrevrangebyscore users 5 0 withscores – reverse the set with score based on the provided limit
→ zincrby users 2 komal – increment score of komal
→ zremrangebyscore users 3 4 – 3 and 4 would be removed.
→ zremrangebyrank users 0 2 –from score 0 to 2 value would be removed.

HyperLogLog - Probalastic ds used to count unique values. These value can be anything eg.ip,visitors of website ,email,unique count of locations.
→ pfadd key element
→ pfadd hll a → value a added in hll
→ pfadd hll b c d e f g
→ pfcount hll – return count of value
→ pfadd hll2 1 2 3 4 5 6 7
→ pfcount hll2 – 7
→ pfcount hll hll2 – return total count of both element - 14
→ pfmerge mergedhll hll hll2 – merge into hll
→ pfcount mergehll

Hashes:
Hashes is the map of string keys and string values , that is perfect example of store the object.

→ hset myhash name chail
→ hset myhash email chail@gmail.com
→ hmset myhash country india phone 87654654654 age 14
→ hkeys myhash – return all the keys available in hash
→ hvals myhash - return all the values available in hash

→ hgetall myhash - return all key values
→ hexist myhash name - if val.available then return 1 else 0
→ hlen myhash – return length of hash
→ hmget myhash name email phone country –add multiple
→ hincrby myhash age 2 – increase age by 2
→ hincrbyfloat myhash age 2.5 – increase float value in age
→ hdel myhash age – delete age from myhash
→ hexists myhash age -if age is aval.in myhash then return 1 else 0
→ hstrlen myhash name -return length of name in myhash
→ hsetnx myhash name chail - check if name field is aval.then it won't add else add new field
→ hsetnx myhash lastname rathore – add lastname field bcz it wasn't aval.before.

Redis Transaction : executing bunch of command in single go.all command should be atomic ,if any error then none of command would be executing.
→ **enable transaction by enter multi+enter**
Now you are in transcatin mode , enter multiple command then once you have done just write exec+enter to execute all the command in a single go.
→ **multi**
→ set name chail
→ get name
→ set a 1
→ set b 2
→ **exec**
Now all command executed in a single go.
If you press **discard** before exec then all command would be discarded.
→ **watch** command - if you watch any particular key then we create a transaction then transaction will be executing based on condition provided.
If no change in key then trans. Would be completed else return nil.
also Change of watch must be from different client.
Change of watch from same cli would block the transaction

redis-pubsub(publisher subscriber model)- we will be having a different channel where any of the client can subscribe to it and publish the message to it.
We will subscribe couple on channel and within one client we will push the message to those channel and see how the message is displayed.
Three client for redis - subscriber,subscriber,publisher
Note- we can subscribe first and then create the channel also.
→ subscribe news
→ subscribe news
→ publish news "breaking news"
Now "breaking news" message will be shown on both the subscriber.
Now terminate 2nd client and subscribe two new channel news and broadcast.
→ subscribe news broadcast
→ publish broadcast "broadcast breaking news"

Pattern based subscribe
→ psubscribe news* –all channel start with news would be subscribe
→ psubscribe h?llo – all pattern made from h and llo
→ psubscribe b[ai]ll –ball,bill

→ Psubscribe news* h?llo b[ai]ll – subscribe all related pattern channel
→ publish news "cold war"
→ publish newsone "hotwar"
→ publish hello "world"
→ publish hello "duniya"
→ publish ball "ball"
→ publish bill "bill"

pubsub channels -return all without pattern based channel which have subscribed
pubsub numsub news - return no.of subscriber who has subscribe to news channel
pubsub numpat - return all pattern based subscriber of channel

Redis scripts - all steps in the script runs in atomic way.

→ eval "redis.call('set',KEYS[1],ARGV[1])" 1 name chail
→ get name
→ eval "redis.call('mset',KEYS[1],ARGV,KEYS[2],ARGV[2])" 2 name last_name chail singh
→ get last_name
→ hmset county_cap india "new delhi" usa "wdc" russia "mosscow" germany "berlin" japan "tokyo" italy "rome"
→ zadd country 1 italy 2 india 3 usa
→ zrange country 0 -1
Now i want the capital from hset based on value aval.on country sorted set
For that we will write the script
→ eval local order = "redis.call('zrange', KEYS[1],0 , -1); return redis.call('hmget',KEYS[2],unpack(order));" 2 country country_cap

Rather we write all script we can store then call
So first load script it will return the hashcode and run

→ Script load "local order = "redis.call('zrange', KEYS[1],0 , -1); return redis.call('hmget',KEYS[2],unpack(order));"
This return hashcode 464656gd6sf9

Now run
→ evalsha 464656gd6sf9 2 country country_cap
→ scripts exists 464656gd6sf9 –return 1 because this script is available
→ script flush

Note:we can create an entire lua script and run the entire file but remember to always use script when it results in a better performance.
Script should not be too long because while script is running everything is waiting to be finished
Script default timeout= 5second.

Connection and security :

Once you have connected to redis client,is your connection is successfull or not to check that use ping command.
→ ping - return PONG if success
→ echo message
Redis have many databases index of default database is 0
→ select 0
→ select 1 -switch to 1th index database
→ all the data of different database is stored into single rdb or aof file in the redis.
In singleton architecture if cluster is available then by default everytime database will be 0th index you can’t change.
Keys would be unique for different database.
→ select 0
→ set name chail
→ get name
→ select 1
→ set name komal
→ get name – return komal
→ select 0
→ get name – return chail
→ client list - return all client
→ client setname clientname
→ client getname – clientname
→ client kill id 3
→ config set requirepass alkcgr –set password ,now you need to enter password to execute command
→ auth alkcgr

Redis-GeoSpatial :
Data has been added based on latitude and longitude -generate 52bit int value based on that geohash generated based on location.
Earth in redis is modelized as a sphere. So data may have 0.5% error.

→ GEOADD key longitude latitude location
→ GEOADD maps 27.2046 77.4977 Ahmedabad
→ GEOADD maps 67.24466 57.4677 Mumbai 77.246766 52.45377 Banglore
→ zrange maps 0 -1
→ GEOHASH maps Ahmedabad – tssse48afj0
→ to check go to google - geohash.org ,enter geohash value it will give location
→ GEOPOS maps Ahmedabad - return longitude and latitude value.
→ GEODIST maps Ahmdbad Mumbai – give distance in meter
→ GEODIST maps Ahmdabad Mumbai km - give distance in km
→ GEODIST maps Ahmdabad Mumbai mi - give distance in miles
→ GEODIST maps Ahmdabad Mumbai m - give distance in meter
→ GEORADIUS maps 27.2046 77.4977 500km – within 500km from given coordinates all the city will be displayed.
→ GEORADIUS maps 27.2046 77.4977 500km withcoord
→ GEORADIUS maps 27.2046 77.4977 500km withdist
→ GEORADIUS maps 27.2046 77.4977 500km withcoord withdist withhash – in this case it will not return geohash value it will return 52bit integer value.
→ GEORADIUSBYMEMBER maps Ahmedabad 500km – get value from member
→ GEORADIUSBYMEMBER maps Ahmedabad 500km desc
→ GEORADIUSBYMEMBER maps Ahmedabad 500km asc

Benchmark : radis has it’s inbuild tool that is redis benchmark to check how radius is performed on single node and cluster.

→ redis-cli -h 127.0.0.1 -p 6379 – connect with remote redis server with host name and port
→ redis-benchmark - if running on local system
→ redis-benchmark -h ip -p port -if running on other server
→ redis-benchmark -n 1000 -runs on 1K command
→ redis-benchmark -n 1000 -d 1000000 - benchmark run for 1K command with 1000kb data
→ redis-benchmark -n 1000 -d 1000000 -c 200

Defaul client = 50
Default data = 3 bytes
Default command = all