```
-- SET command to add a key-value pair
SET ds4300 "LargeScale Info Retrieval"   -- Sets the key 'ds4300' with the value "LargeScale Info Retrieval"
SET cs3200 "Intro to DBs"              -- Sets the key 'cs3200' with the value "Intro to DBs"

-- KEYS command lists all keys in the database
KEYS *                          -- Lists all the keys currently in the Redis database

-- GET command retrieves the value of a key
GET ds4300                      -- Retrieves the value of key 'ds4300' ("LargeScale Info Retrieval")
GET cs3200                      -- Retrieves the value of key 'cs3200' ("Intro to DBs")

-- DEL command removes a key
DEL ds4300                      -- Deletes the key 'ds4300' from the Redis database

-- KEYS again to check keys after deletion
KEYS *                          -- Lists all keys after deleting 'ds4300'

-- SELECT command switches between Redis databases
SELECT 5                        -- Switches to the Redis database index 5

-- KEYS in new database (after SELECT)
KEYS *                          -- Lists all the keys in database 5

-- SET command to store a numeric value
SET ds4300 10                    -- Sets the key 'ds4300' to the value 10

-- INCR command increments the value of a numeric key by 1
INCR ds4300                     -- Increments the value of 'ds4300' by 1 (from 10 to 11)

-- GET command to fetch the updated value
GET ds4300                      -- Retrieves the new value of 'ds4300' (now 11)

-- KEYS command to list keys again
KEYS *                          -- Lists all keys in the current database

-- SETNX command sets a value only if the key doesn't exist
SET winston 10                   -- Sets the key 'winston' to 10
GET winston                     -- Retrieves the value of 'winston' (10)

SET winston 11                   -- Updates 'winston' to 11
GET winston                     -- Retrieves the updated value of 'winston' (11)
```

```
SETNX winston 12                    -- Sets 'winston' to 12 only if it doesn't already exist (no effect since 'winston' exists)
-- This will not change the value because 'winston' already exists with value 11.

-- HSET command to create and store fields in a hash
HSET bike:1 model Demios brand Ergonom price 1971  -- Sets fields in the hash 'bike:1' with model, brand, and price

-- HGETALL command to fetch all fields of a hash
HGETALL bike:1                      -- Retrieves all fields and values in the hash 'bike:1'

-- HGET command to retrieve a specific field from a hash
HGET bike:1 model                   -- Retrieves the 'model' field from 'bike:1' (Demios)

-- SADD command to add members to a set
SADD ds4300 "Mark"                  -- Adds "Mark" to the set 'ds4300'
SADD ds4300 "Sam"                   -- Adds "Sam" to the set 'ds4300'
SADD cs3200 "Nick"                  -- Adds "Nick" to the set 'cs3200'
SADD cs3200 "Sam"                   -- Adds "Sam" to the set 'cs3200' (no effect because "Sam" already exists)

-- SCARD command to get the number of elements in a set
SCARD ds4300                        -- Returns the number of elements in the set 'ds4300' (2: "Mark" and "Sam")

-- SINTER command to find the intersection of two sets
SINTER ds4300 cs3200                -- Finds common elements between 'ds4300' and 'cs3200' (will return "Sam")

-- SDIFF command to find the difference between two sets
SDIFF ds4300 cs3200                 -- Finds elements in 'ds4300' but not in 'cs3200' (will return "Mark")

-- SREM command to remove an element from a set
SREM ds4300 "Mark"                  -- Removes "Mark" from the set 'ds4300'

-- SRANDMEMBER command to get a random element from a set
SRANDMEMBER ds4300                  -- Retrieves a random element from 'ds4300' (will return "Sam" since it's the only remaining member)
```

Summary of Commands:

- **SET**: Assigns a value to a key.

- **GET**: Retrieves the value of a key.
- **DEL**: Deletes a key.
- **SELECT**: Switches between Redis databases.
- **INCR**: Increments the value of a numeric key.
- **SETNX**: Sets a value only if the key doesn't exist.
- **HSET/HGET/HGETALL**: Works with hash data types (field-value pairs).
- **SADD**: Adds elements to a set.
- **SCARD**: Gets the size (cardinality) of a set.
- **SINTER**: Returns the intersection of two sets.
- **SDIFF**: Returns the difference between two sets.
- **SREM**: Removes an element from a set.
- **SRANDMEMBER**: Retrieves a random element from a set.

Redis: A Beginner's Guide

Introduction

Redis (Remote Dictionary Server) is an open-source, in-memory data structure store used as a database, cache, and message broker. It supports various data structures like strings, hashes, lists, sets, and sorted sets, making it highly versatile for different use cases.

Installing Redis

Redis can be installed on different operating systems using package managers. Below are some common installation methods:

On Linux (Ubuntu/Debian):

sudo apt update

sudo apt install redis-server

On macOS (using Homebrew):

brew install redis

On Windows:

Redis does not natively support Windows, but you can use WSL (Windows Subsystem for Linux) to run Redis:

wsl --install -d Ubuntu

sudo apt update

sudo apt install redis-server

Starting Redis:

redis-server

You can check if Redis is running by executing:

redis-cli ping

If it responds with `PONG`, Redis is running.

Redis Data Types

1. Strings

SET key "value"

GET key

2. Hashes

HSET user:1 name "Alice" age 30

HGET user:1 name

3. Lists

LPUSH queue "task1"

RPUSH queue "task2"

LPOP queue

4. Sets

SADD myset "apple"

SADD myset "banana"

SMEMBERS myset

5. Sorted Sets

ZADD scores 100 "Alice"

ZADD scores 200 "Bob"

ZRANGE scores 0 -1 WITHSCORES

Basic Query Writing in Redis

Key-Value Operations

SET name "John"

GET name

DEL name

EXISTS name

Working with Multiple Keys

MSET key1 "value1" key2 "value2"

MGET key1 key2

Expiring Keys

SET session "xyz" EX 60   # Expires in 60 seconds

TTL session # Check time-to-live

Aggregation in Redis

Aggregation refers to performing calculations on stored data. Redis provides commands for counting, summing, and filtering data.

Counting Elements

SCARD myset # Count elements in a set

LLEN queue   # Count elements in a list

Summing and Averaging Data

INCR counter # Increment counter

DECR counter # Decrement counter

For more complex aggregation, you can use Lua scripting or Redis Streams.

Advanced Query Writing

Pattern Matching with Keys

KEYS user:* # Find all keys matching a pattern

Transactions

Transactions allow multiple commands to be executed atomically.

MULTI

SET key1 "value1"

SET key2 "value2"

EXEC

Lua Scripting

Lua scripts allow running complex queries directly on Redis.

EVAL "return redis.call('GET', KEYS[1])" 1 key1

Example Queries for Common Tasks

1. Caching API Responses

SET api:response "{\"data\": \"value\"}" EX 300 # Cache expires in 5 minutes

GET api:response

2. Implementing a Rate Limiter

INCR user:123:requests

EXPIRE user:123:requests 60 # Limit resets every 60 seconds

GET user:123:requests

3. Creating a Simple Leaderboard

ZADD leaderboard 500 "Alice"

ZADD leaderboard 600 "Bob"

ZRANGE leaderboard 0 -1 WITHSCORES # Retrieve sorted scores

4. Managing a Queue

LPUSH job_queue "task1"

LPUSH job_queue "task2"

RPOP job_queue # Process the last added task

5. Storing and Retrieving User Sessions

HSET session:abc123 user_id 42 status "active"

HGETALL session:abc123

EXPIRE session:abc123 1800 # Session expires in 30 minutes

Conclusion

Redis is a powerful in-memory database with fast performance and rich data structures. Learning its commands and aggregation techniques will help you efficiently store and query data. Experiment with different data structures and use cases to get the most out of Redis.