Homework Problem Set H Submission Form

# Overview

| Your Name | **Nick Videtti** |
| --- | --- |
| Your SU Email | **nvidetti@syr.edu** |

# Instructions

Put your name and SU email at the top. Answer these questions all from the lab. When asked to include screenshots, please follow the screenshot guidelines from the first homework.

Remember as you complete the homework that it is not only about getting it right/correct. We will discuss the answers in class so it’s important to articulate anything you would like to contribute to the discussion in your answer:

* If you feel the question is vague, include any assumptions you've made.
* If you feel the answer requires interpretation or justification, provide it.
* If you do not know the answer to the question, articulate what you tried and how you are stuck.
* Highlight any doubts or questions you would like me to review.

This how you receive credit for answering questions that might not be correct. In addition, you must complete the reflection portion of the homework assignment for full credit. Since most answers will be similar this is an important part of your individual submission.

Complete Part II of this document first, then go back and complete the Reflection in Part I.

# Part I: Reflection

Use this section to reflect on your learning. To achieve the highest grade on the assignment, you must be as descriptive and personal as possible with your reflection.

1. As you completed this assignment, identify what you learned.

**This assignment helped me understand the use cases for a Redis database, or really a key-value database in general, and it also gave me a deeper understanding of the different Redis data types.**

1. What barriers or challenges did you encounter while completing this assignment?

**I think the biggest challenge was just remembering the commands, and in some cases having to look them up in the asynchronous work or Google.**

1. How prepared were you to complete this assignment? What can you do to be better prepared?

**I felt pretty prepared for this assignment thanks to the asynchronous work and the quiz. I think knowing what to generally expect after going through past assignments also helps.**

1. Rate your comfort level with this week’s material. Use the rubric provided.

4 ==> I understand this material and can explain it to others.  
**3 ==> I understand this material.**  
2 ==> I somewhat understand the material but sometimes need guidance from others.  
1 ==> I understand very little of this material and need extra help.

# Part II: Questions

**For each question, include a copy of the code required to complete the question along with a screenshot of the code and a screenshot of the output.**

Snapchat clone! Let’s use Redis to create a data model like Snapchat. Basically, users send messages to each other and once the message is accessed by the receiver it expires in 60 seconds. The rules:

1. Each **message** should be keyed by an ID (you can use an integer and control the ID yourself).
2. Each message key should be namespaced, like so: **snap:msg:1** where **1** is the ID in this case.
3. Each **message** has three hash fields:
   1. **To**: username of the recipient, e.g., Bob
   2. **From**: username of the sender, e.g., Mary
   3. **Text**: the message itself
4. When a user ***sends a*** ***message,*** perform these Redis commands:
   1. A new key is added to namespace **snap:msg:*id*** with the fields set in the hash.
   2. Add the ***ID*** of the message to the user’s inbox key, queue, which is a list.   
      For example, Mary’s inbox key is **snap:inbox:mary**
5. When a ***user reads a message,*** we:
   1. Remove it from the end of their inbox key list, a FIFO queue.
   2. Set the message ID key to expire in 60 seconds.

1. Using the Redis CLI, send these messages in the order they are listed with Redis commands. Make sure to perform both steps D.a and D.b as separate commands.

**To From Text**

Bob Art You owe me $50

Che Bob Hello there!!!

Che Dax Is this thing on?

Dax Art When is the meet-up?

Che Art What is Bob doing. OMG.

Bob Dax Who?!?!?

**hmset snap:msg:1 To Bob From Art Text "You owe me $50"**

**lpush snap:inbox:Bob snap:msg:1**

**hmset snap:msg:2 To Che From Bob Text "Hello there!!!"**

**lpush snap:inbox:Che snap:msg:2**

**hmset snap:msg:3 To Che From Dax Text "Is this thing on?"**

**lpush snap:inbox:Che snap:msg:3**

**hmset snap:msg:4 To Dax From Art Text "When is the meet-up?"**

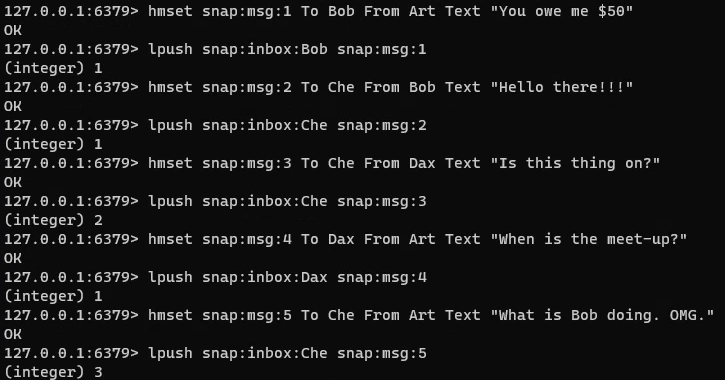
**lpush snap:inbox:Dax snap:msg:4**

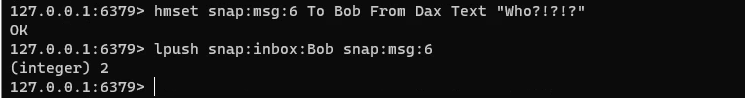
**hmset snap:msg:5 To Che From Art Text "What is Bob doing. OMG."**

**lpush snap:inbox:Che snap:msg:5**

**hmset snap:msg:6 To Bob From Dax Text "Who?!?!?"**

**lpush snap:inbox:Bob snap:msg:6**





2. Using the Redis CLI, read messages for the following users, in the order listed. Make sure to perform both steps E.a and E.b.

Bob

Che

Art

Bob

**rpop snap:inbox:Bob**

**hset snap:msg:1 ex 60**

**rpop snap:inbox:Che**

**hset snap:msg:2 ex 60**

**rpop snap:inbox:Art**

**rpop snap:inbox:Bob**

**hset snap:msg:6 ex 60**













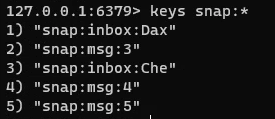


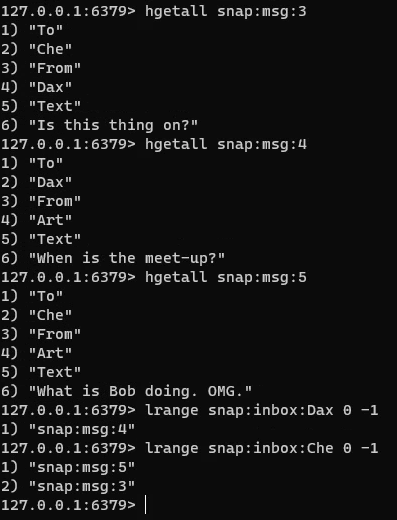
3. Provide a current state of the Redis database after Questions 1 and 2.

Display the current keys under the **snap:** namespace.

Display the messages that have not been read (and therefore have not expired).

Display the message IDs in each users’ inbox.





The Department of Motor Vehicles has hired you to build a queue management system. You have decided the best system for this is Redis (a good choice, BTW). The system needs to manage a single queue of users, by username. Queued users can be served at one of four windows, A, B, C, or D. The structure you build in Redis should support the queue and be able to display who is waiting in the queue. As people go to the window they should be removed from the queue and assigned to one of the four windows. You should be able to display who is at each window at any time.  
Namespace all keys with **dmv:**

**Example:**

Users in queue: Tom, Bill, Bart

Being served at windows: A: Carl, B: Steve, C: Chuck, D: Dave

Event: When Dave is done at the Window D, Bart is served next.

Users in queue: Tom, Bill

Being served at windows: A: Carl, B: Steve, C: Chuck, D: Bart

Event: Mary arrives

Users in queue: Mary, Tom, Bill

Being served at windows: A: Carl, B: Steve, C: Chuck, D: Bart

1. It’s first thing in the morning and eight people are waiting outside for the department to open! Add them to your queue: **Amy, Beth, Chris, Dee, Erin, Fran, Greg, Hela**Provide all the commands required to accomplish this and a view of the queue.  
     
   **lpush dmv:queue "Amy"**

**lpush dmv:queue "Beth"**

**lpush dmv:queue "Chris"**

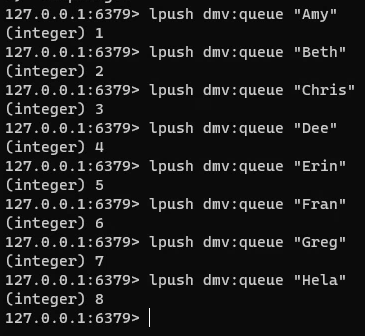
**lpush dmv:queue "Dee"**

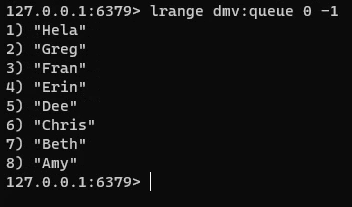
**lpush dmv:queue "Erin"**

**lpush dmv:queue "Fran"**

**lpush dmv:queue "Greg"**

**lpush dmv:queue "Hela"**





1. The department is now open! Assign the first four people to Windows A, B, C, and D, respectively. Oh, and don’t forget to remove them from the queue!  
   Provide all the steps required to accomplish these steps and a view of the queue and windows.

**rpop dmv:queue**

**hmset dmv:windows A "Amy"**

**rpop dmv:queue**

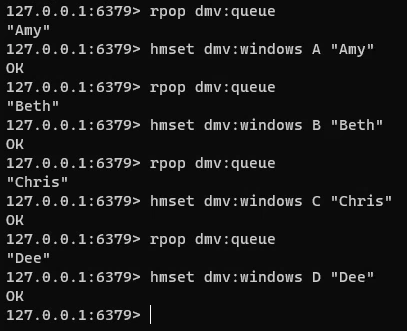
**hmset dmv:windows B "Beth"**

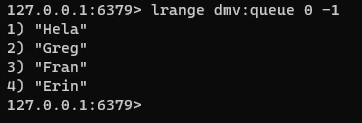
**rpop dmv:queue**

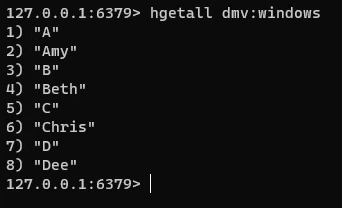
**hmset dmv:windows C "Chris"**

**rpop dmv:queue**

**hmset dmv:windows D "Dee"**







1. Next, the following events occur:  
   a. Iris arrives.   
   b. Window C becomes available—move the next person from the queue to this window!  
   c. Window B becomes available—move the next person from the queue to this window!  
   d. Jake arrives.  
   e. Window C becomes available—move the next person from the queue to this window!  
   Provide all the steps required to accomplish these steps and a view of the queue and windows after the events.

**#a. Iris arrives.**

**lpush dmv:queue "Iris"**

**#b. Window C becomes available—move the next person from the queue to this window!**

**rpop dmv:queue**

**hset dmv:windows C "Erin"**

**#c. Window B becomes available—move the next person from the queue to this window!**

**rpop dmv:queue**

**hset dmv:windows B "Fran"**

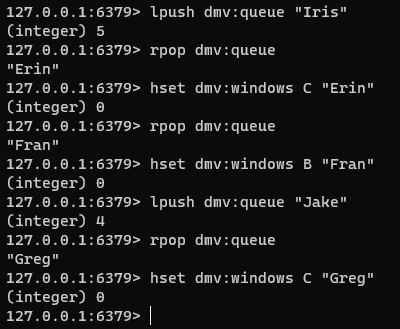
**#d. Jake arrives.**

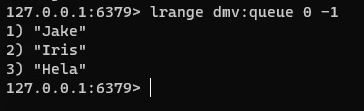
**lpush dmv:queue "Jake"**

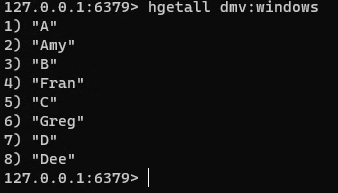
**#e. Window C becomes available—move the next person from the queue to this window!**

**rpop dmv:queue**

**hset dmv:windows C "Greg"**







1. Use Spark to load the exam scores data set `/home/jovyan/datasets/exam-scores/\*.csv` into Redis under the namespace **examscores**. Use Spark to demonstrate the data is there by querying it back out.

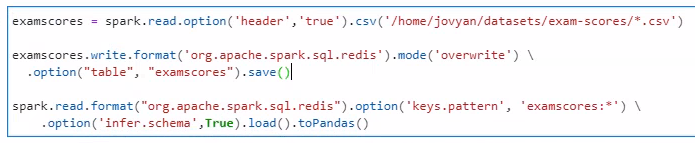
**examscores = spark.read.option('header','true').csv('/home/jovyan/datasets/exam-scores/\*.csv')**

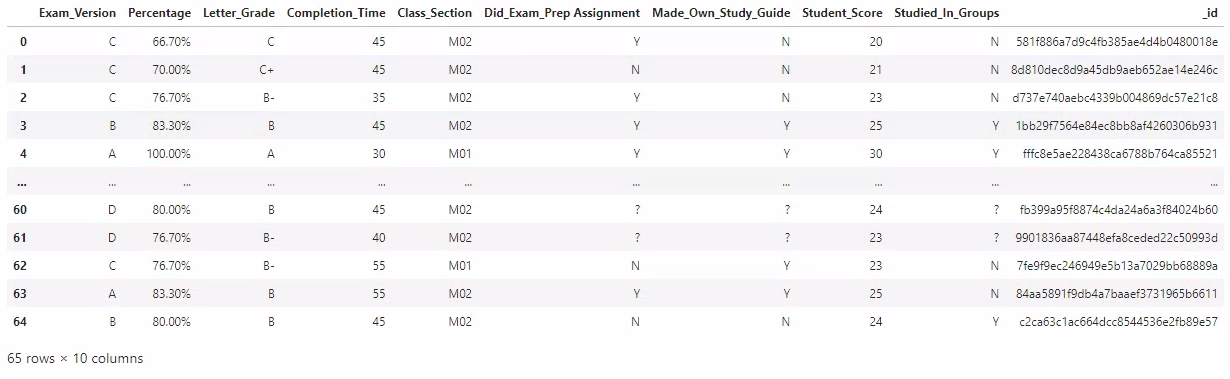
**examscores.write.format('org.apache.spark.sql.redis').mode('overwrite') \**

**.option("table", "examscores").save()**

**spark.read.format("org.apache.spark.sql.redis").option('keys.pattern', 'examscores:\*') \**

**.option('infer.schema',True).load().toPandas()**

****

****

1. In Spark SQL, read the Redis **examscore** data into a temp view and get the min, max, and average exam score across all students. Write the data back out to Redis as **examscoresummary**; finally, query the key in Redis showing all values in the hash!

**exam\_score\_view = examscores.createOrReplaceTempView('exam\_score\_view')**

**examscoresummary = spark.sql('''**

**SELECT**

**MIN(Student\_Score) MINIMUM\_SCORE,**

**MAX(Student\_Score) MAXIMUM\_SCORE,**

**AVG(Student\_Score) AVERAGE\_SCORE**

**FROM**

**EXAM\_SCORE\_VIEW**

**''')**

**examscoresummary.write.format('org.apache.spark.sql.redis').mode('overwrite') \**

**.option("table", "examscoresummary").save()**

**spark.read.format("org.apache.spark.sql.redis").option('keys.pattern', 'examscoresummary:\*') \**

**.option('infer.schema',True).load().show()**

