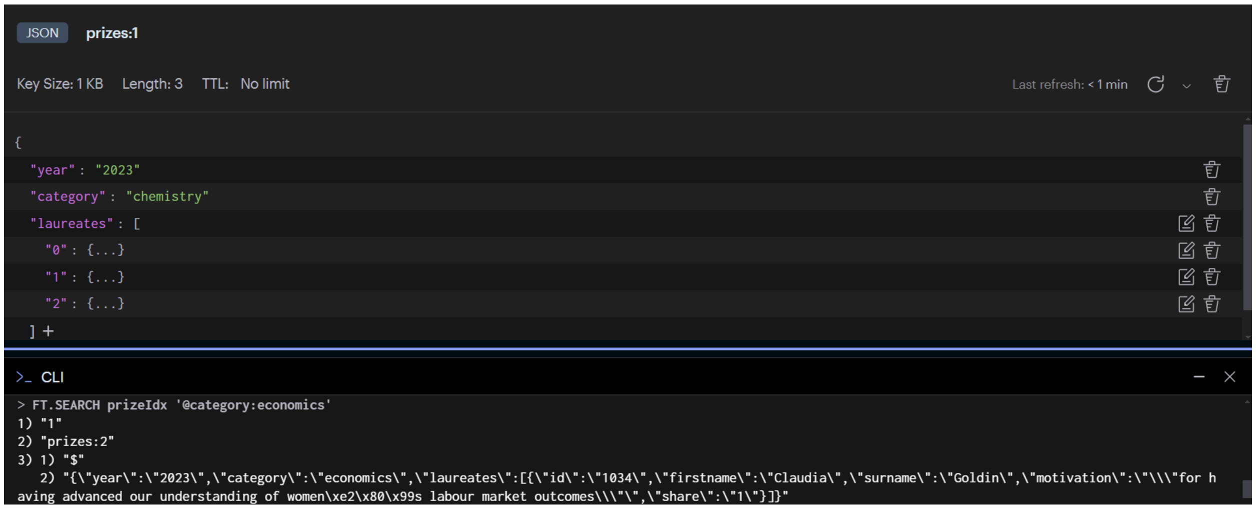
# COEN 424/6313 Assignment1 Fall 2024

Individual or Group of 2 or 3 Assignment due by October 26 23:59

This assignment is designed to practise data model design, queries and communication through binary serialization and deserialization (gRPC).

The dataset is in the JSON form for Novel Prizes since year 1901. api.nobelprize.org/v1/prize.json

To illustrate an data sample in Redis, one prize data sample from the prize.json is added to the Redis database with a key called prizes: 1, type as JSON. After creating an index, the 'category:economics' can be searched using the FT.SEARCH (index search).



```
{
  "year" : "2023"
  "category" : "chemistry"
  "laureates" : [
    "0" : {...}
    "1" : {...}
    "2" : {...}
  ] +
```

```
> FT.SEARCH prizeIdx '@category:economics'
1) "1"
2) "prizes:2"
3) 1) "$"
   2) "{\"year\":\"2023\",\"category\":\"economics\",\"laureates\":[{\"id\":\"1034\",\"firstname\":\"Claudia\",\"surname\":\"Goldin\",\"motivation\":\"\\\"for h
aving advanced our understanding of women\xe2\x80\x99s labour market outcomes\\\"\",\"share\":\"1\"}]}"
```

Task 1 Data Storage and Query in Redis.

1.1 Load this dataset into Redis Cloud using the free tier including at least data in year 2013 to year 2023 inclusive. The keys' type should be JSON.

1.2 Create an index that includes the key 'year' and 'category' at least. **[bonus point: the index includes the key 'laureates' as vector type].**
A reference on Redis index is accessible here. Indexing | Docs (redis.io)

1.3 Program a client application code with a language preferred to respond the following queries:

- Given a category value, return the total number of laureates between a certain year range within the span from year 2013 to year 2023. The query should use the index.
- Given a keyword, return the total number of laureates that have motivations covering the keyword.

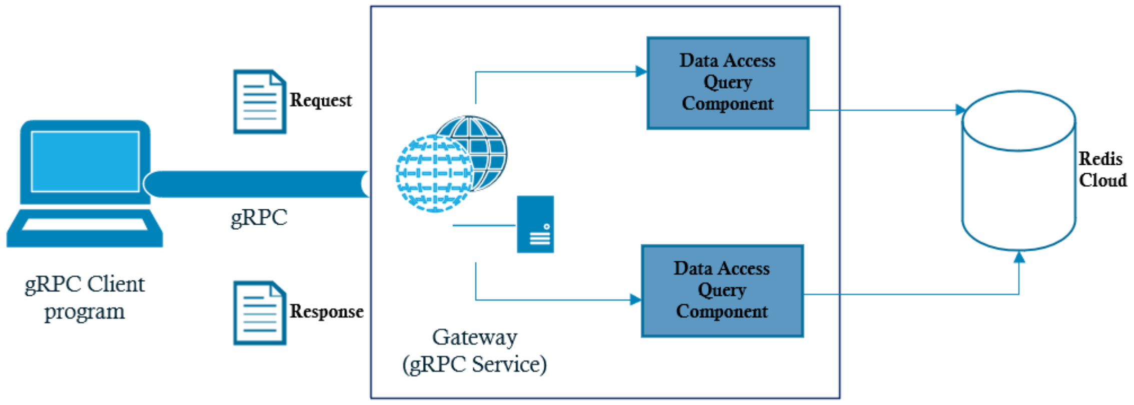
- Given the first name and last name, return the year, category and motivation of the laureate.

This client program or application can run on the local computer that connects to Redis cloud for queries. The application can be run as an application from command line or IDE without the need of any UI.

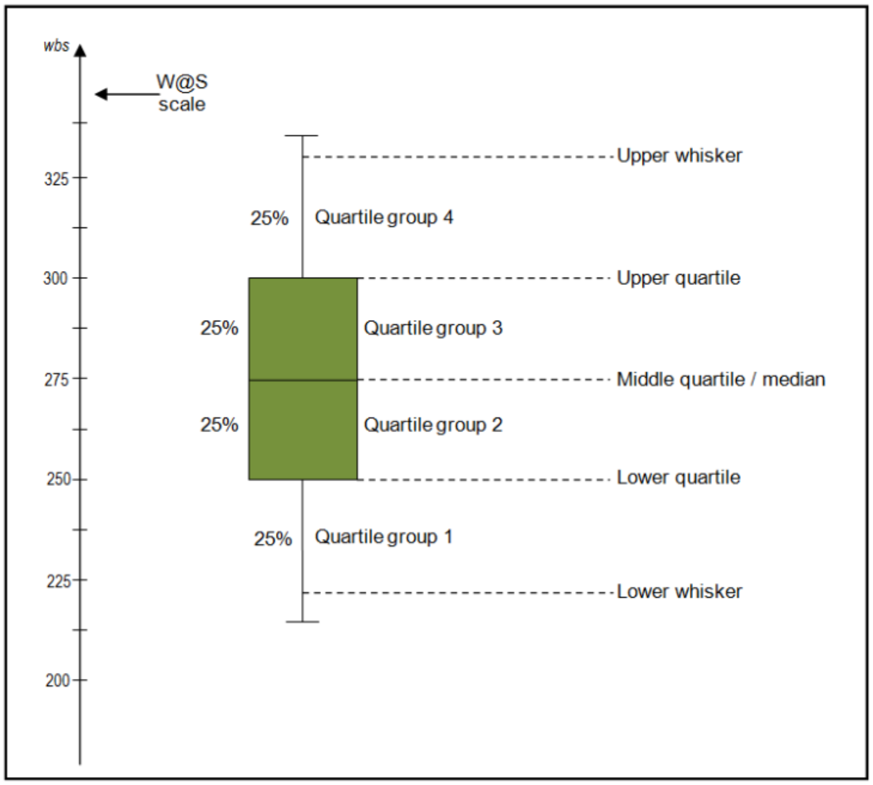
Task 2 - Data Model Communication through gRPC or any Protobuf based binary serialization and deserialization). The communication involves two parts – client and service gateway. A client sends a gRPC Request with parameters to the service gateway. The service gateway is a gRPC service. It invokes the data access components (similar to Task 1' programs) that retrieve data from the Redis cloud. The service gateway generates the gRPC Response and sends the response back to the client.

The reference to gRPC is accessible from gRPC. A reference below on programing null object as ***Empty*** request or response.

Google.Protobuf.WellKnownTypes.Empty Class Reference [ https://protobuf.dev/reference/csharp/api-docs/class/google/protobuf/well-known-types/empty ]



2.1 Define the three queries in Task 1.3 into three gRPC services with. proto files.
2.2 Define the JSON data model for Request and Response for each query. Use a json to proto converter to generate the Response and Request Protobuf messages.
2.3 Choose the programming language preferred, program the gRPC client and gRPC service or gateway. This gateway includes data access query components refactored or reused from Task 1 to query the results from the Redis cloud.
2.4 Deploy the gRPC service to a cloud endpoint . The gRPC client can run on local computer.
2.5 Run the client from a local computer for 100 times to each of three queries and measure the end-to-end delay.
2.6 Plot the box plots for each query's 100 measured end-to-end delays. An example is shown below.



Task 3: Write up the solution report following the Template IEEE - Manuscript Templates for Conference Proceedings   https://www.ieee.org/conferences/publishing/templates.html

The report content outline

Section 1. Redis Cloud Query Solution

Describing the technical solutions for Task 1.1 and 1.2. Attach screenshots for 1.3's queries and outputs for each query.

Section 2. Data Model Design Service Development

2.1 The overview of serialization and deserialization model (gRPC or protobuf based framework)

2.2  Present the .proto for each gRPC service and messages defined.

2.3  For each service definition, screenshot of the code implementation with brief description

Section 3. Cloud Deployment and Run

Present a step by step description on the deployment to a cloud at your choice and present how the endpoint of the cloud is set or configured.

Attach a screenshot of the running cloud service, and returning responses to the client's requests.

Section 4.

4.1 The settings to run 100 times of each service

4.2 The box plots with measurement of three gRPC services.

Section 5 Member contribution list of each member's contribution according to the checklist below.

| Name (SID) and Signature | Task List | | Contribution Role and Percentage ( X % )<br>(If Y for the task list, write the contribution role and percentage counted for the completeness of this task) |
|---|---|---|---|
| Member 1 | Task 1.1 | Y / N | |
| | Task 1.2 | Y / N | |
| | Task 1.3 | Y / N | |
| | Task 2.1 | Y / N | |
| | Task 2.2 | Y / N | |
| | Task 2.3 | Y / N | |
| | Task 2.4 | Y / N | |
| | Task 2.5 | Y / N | |
| | Task 2.6 | Y / N | |
| | Task 3 | Y / N | |

Create **[SID_1]_[SID_2]_[SID_3]A1.zip** (.gz, .tar, .zip are acceptable. .rar file is NOT acceptable) file contains a folder named with your student ID. This folder contains all the source code + report.pdf for the assignment. One submission for the group is sufficient.