CS 122A: Introduction to Data Management – Spring 2019

Homework 8: NoSQL (100 points)

Due Date: **Thursday, June 6 (5:00 PM)**

Submission

All HW assignments should contain both your student ID, your name, and a suffix ("**HW8**") and must be submitted online, formatted in PDF document (e.g., 12345678_John_Doe_HW8.pdf), through the associated dropbox on Gradescope. **Also, please submit a zipped file (12345678_John_Doe.zip) of your SQL++ queries via EEE.** Within the zip file, the query files should be named properly by following the specified naming convention (i.e., **Question_4.txt**, **Question_5.txt**, **Question_6.txt**, **Question_7.txt**, and **Question_8.txt**). The submitted queries will be evaluated by an automated script, so please make sure your output uses the same schema as the sample output. Extra or missing attributes in your query results may lead to a lower grade.

See the table below for the HW submission opportunities. Note that after 5:00 PM on Thursday no further HW submissions will be accepted. (We will be releasing the solution at that time so you can study for the final midterm exam.) Please strive to get all your work in on time!

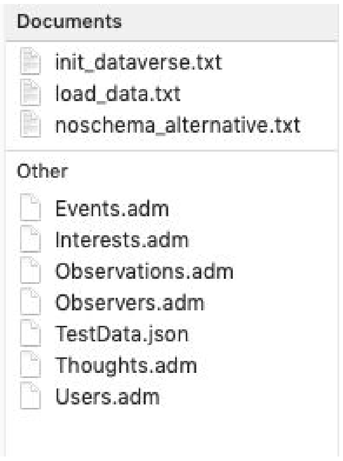
Date / Time	**Grade Implications**
Thursday, Jun 6 (5:00 PM)	Full credit will be available

NoSQL (SQL) [100 pts]

Congratulations! If you are reading this, you **must have successfully finished reading and running the exercises in the SQL++ Primer**. Business is booming, and PHLogger.com needs to scale beyond MySQL's capabilities. In this assignment, you will explore some of the virtues of "NoSQL" systems in situations like this. Enter AsterixDB…! As you read through the assignment, answer its questions, and write queries, you will want to refer to two other sources of information -- namely, the solution to HW #1 (the E-R schema) and the corresponding translated relational schema that you used for the SQL-based HW assignments. (We will specify the number of the expected answer for each query and a sample output record, which has the right schema but synthetic values, to help you with that.)

Step 0. Load dataset

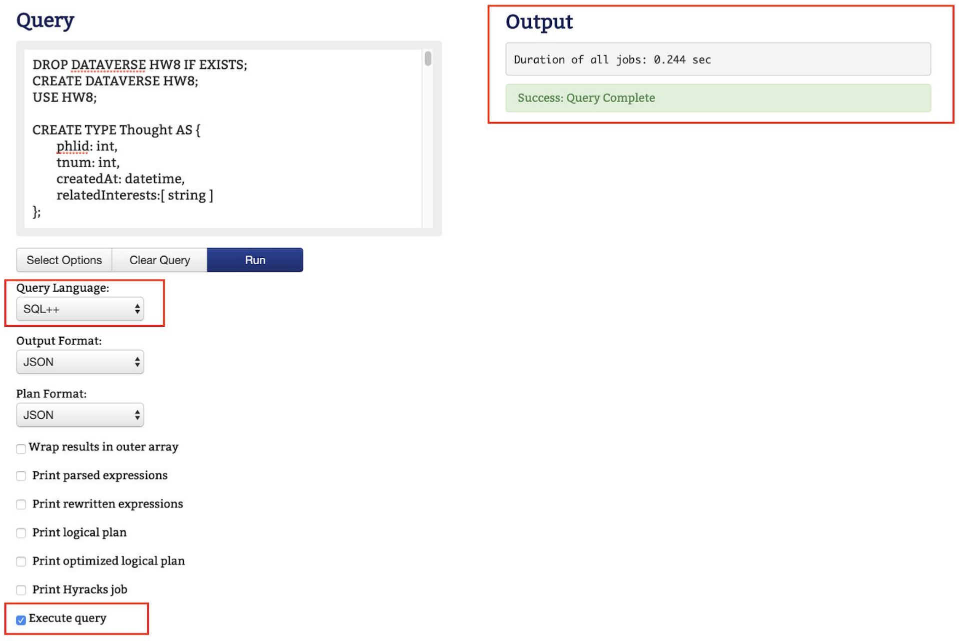
To save you time, we have transformed the relational PHLogger database schema into ADM (the Asterix Data Model). You need to download our scripts and populate the dataset. Unzip the 'HW8.zip' file; there should be 11 files inside the folder like the following picture.



Documents
- init_dataverse.txt
- load_data.txt
- noschema_alternative.txt

Other
- Events.adm
- Interests.adm
- Observations.adm
- Observers.adm
- TestData.json
- Thoughts.adm
- Users.adm

After you've started your sample AsterixDB cluster, the next thing you will need to do is to create the dataverse, datatypes, and datasets for this NoSQL translation of the schema. To do so, you should open `init_dataverse.txt` and copy its contents into the AsterixDB web-based query interface. Make sure that the `Query Language` selected there is "SQL++" (the default setting) and then click 'Run'. After execution, you should see something that resembles the following result:



Query

```
DROP DATAVERSE HW8 IF EXISTS;
CREATE DATAVERSE HW8;
USE HW8;

CREATE TYPE Thought AS {
      phlid: int,
      tnum: int,
      createdAt: datetime,
      relatedInterests:[ string ]
};
```

Query Language: SQL++

Output Format: JSON

Plan Format: JSON

- [ ] Wrap results in outer array
- [ ] Print parsed expressions
- [ ] Print rewritten expressions
- [ ] Print logical plan
- [ ] Print optimized logical plan
- [ ] Print Hyracks job
- [x] Execute query

Output

```
Duration of all jobs: 0.244 sec
```

Success: Query Complete

If the Output doesn't show "Success", please post your issue on **Piazza**. Don't wait until you finish writing your script. Now we will populate all datasets using the bulk loading technique. However, before we load the actual data into our datasets, let's try loading a small dataset from a file. ***Under no circumstances should you try to load the full datasets until you get loading working perfectly here first!***

Edit the path's value (***path_to_hw8_folder***) in the DDL below to point to the TestData.json file inside '**HW8**' folder. **Note**: After you change the path, copy and paste the DDL you just edited into the Query box and hit 'Run'.

Windows Users: There is **NO** need to change '/'. The path you would use will look like this: ("path"=":///C:/Users/XXX/Downloads/HW8/TestData.json")

OSX, Linux Users: Change the *path_to_hw8_folder* to match with the destination you have placed the HW8 folder. (e.g., "path"=":///Users/XXX/Downloads/HW8/TestData.json")

```
DROP DATAVERSE cs122aTest IF EXISTS;
CREATE DATAVERSE cs122aTest;
USE cs122aTest;

CREATE TYPE TestType AS {id: int};
CREATE DATASET TestData(TestType) PRIMARY KEY id;
LOAD DATASET TestData USING localfs
        (("path"=":///path_to_hw8_folder/HW8/TestData.json"),("format"="adm"));
SELECT * FROM TestData;
```

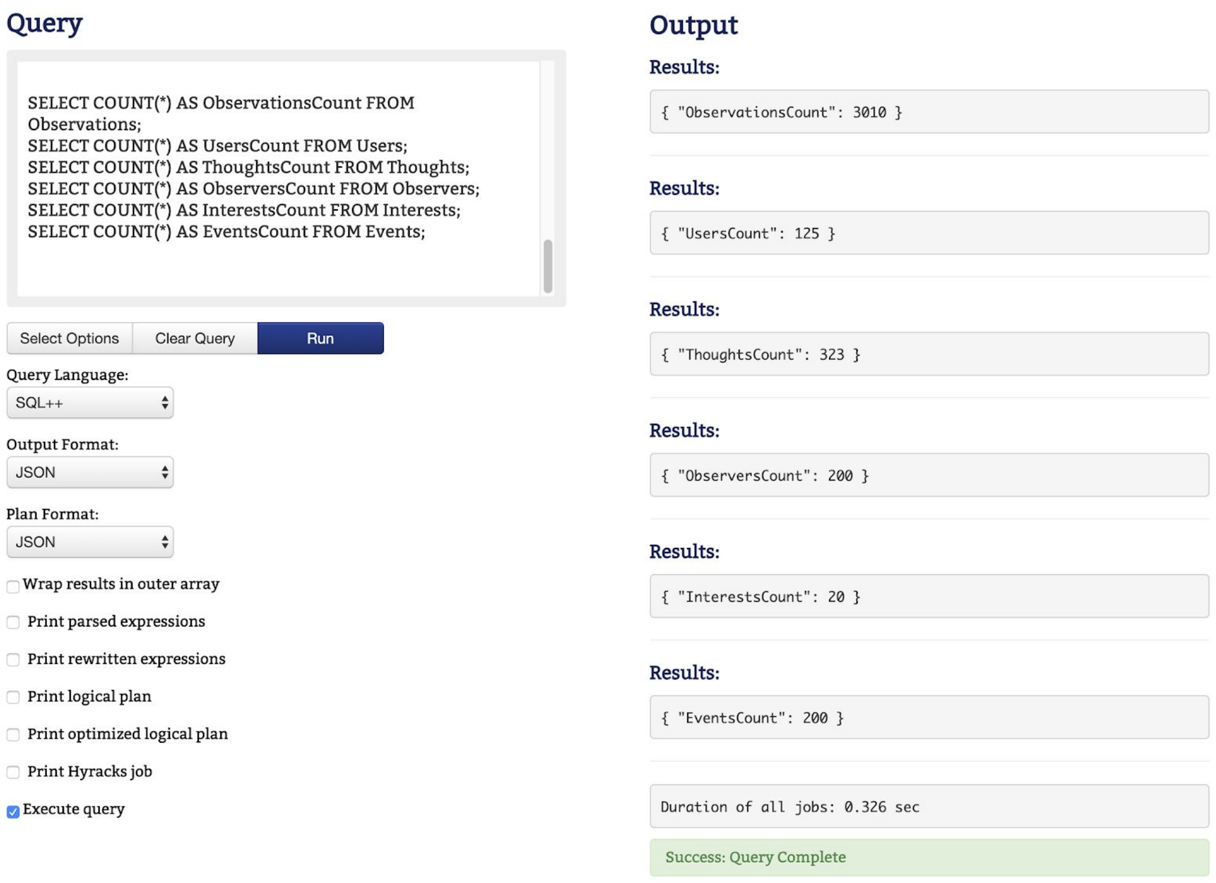
```
        Result:  { "TestData": { "id": 1, "text": "Digital Media: Exhibition" } }
                 { "TestData": { "id": 2, "text": "Changing Creativity" } }
                 { "TestData": { "id": 3, "text": "Introduction To Data Management" } }
```

If this is executed successfully, great! You are now ready to populate the actual datasets. Open `**load_data.txt**` and change the path's value to your downloaded 'HW8' folder just like what you did with the test dataset. Make sure this is the absolute path and ***very carefully*** change **ALL** 6 occurrences of the path variable like you did above. Each file will populate a dataset we previously created. (Hint: If you get the path variable replacement wrong, you will probably get an error message like `ASX3077: /wrong_path/HW8/TestData.json: path not found [HyracksDataException]`.)

Change *path_to_hw8_folder* to the absolute path where you have placed the HW8 folder. (See an example below).

```
        LOAD DATASET Users USING localfs
        (("path"=":///Users/XXX/Downloads/HW8/Users.adm"),("format"="adm"));
```

After changing all 6 path's value, copy and paste the **ENTIRE** content of load_data.txt file into the query interface. As before, execute it by clicking `Run`. You should see something similar to the following result:



```
SELECT COUNT(*) AS ObservationsCount FROM
Observations;
SELECT COUNT(*) AS UsersCount FROM Users;
SELECT COUNT(*) AS ThoughtsCount FROM Thoughts;
SELECT COUNT(*) AS ObserversCount FROM Observers;
SELECT COUNT(*) AS InterestsCount FROM Interests;
SELECT COUNT(*) AS EventsCount FROM Events;
```

Query Language: SQL++
Output Format: JSON
Plan Format: JSON

- [ ] Wrap results in outer array
- [ ] Print parsed expressions
- [ ] Print rewritten expressions
- [ ] Print logical plan
- [ ] Print optimized logical plan
- [ ] Print Hyracks job
- [x] Execute query

```
Results:
{ "ObservationsCount": 3010 }

Results:
{ "UsersCount": 125 }

Results:
{ "ThoughtsCount": 323 }

Results:
{ "ObserversCount": 200 }

Results:
{ "InterestsCount": 20 }

Results:
{ "EventsCount": 200 }

Duration of all jobs: 0.326 sec

Success: Query Complete
```

Again, if the Output doesn't show "Success", please post your issue on **Piazza** after first trying to resolve the problem yourself. Don't wait until you finish your script! (And if you later happen to see other students struggling on Piazza at this step - students making the same mistake you did - please chime in and give them a hand! Loading the data successfully isn't part of the graded exercise, so helping them with loading specifics will be lauded rather than disapproved of as being inappropriate collaboration. :-))

The actual query output results should be:

{ "ObservationsCount": 3010 }

{ "UsersCount": 125 }

{ "ThoughtsCount": 323 }

{ "ObserversCount": 200 }

{ "InterestsCount": 20 }

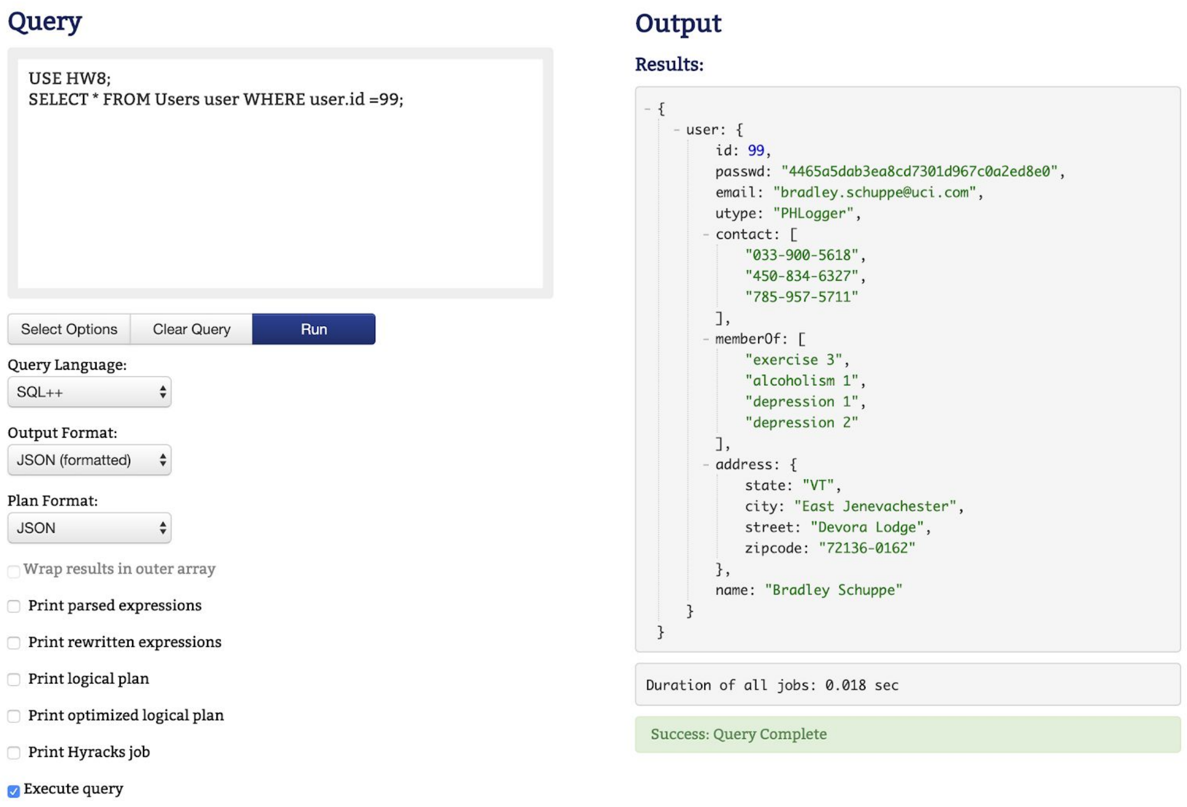
{ "EventsCount": 200 }

Compare these counts with your own output to make sure that you that have the ***complete*** set of datasets. All datasets reside in the **‘HW8’** dataverse, so be sure to put **‘USE HW8;’** in front of your queries when you run them. Here is an example of how that looks:

```
USE HW8;

SELECT COUNT(*) AS Observations FROM Observations;
```

We know looking at a long JSON string can be quite overwhelming so AsterixDB provides a ‘formatted’ output view. In the ‘Output Format’ section, change the format to ‘**JSON (formatted)**’ if you want.



```
USE HW8;
SELECT * FROM Users user WHERE user.id =99;
```

Query Language: SQL++

Output Format: JSON (formatted)

Plan Format: JSON

- [ ] Wrap results in outer array
- [ ] Print parsed expressions
- [ ] Print rewritten expressions
- [ ] Print logical plan
- [ ] Print optimized logical plan
- [ ] Print Hyracks job
- [x] Execute query

Results:

```
- {
  - user: {
      id: 99,
      passwd: "4465a5dab3ea8cd7301d967c0a2ed8e0",
      email: "bradley.schuppe@uci.com",
      utype: "PHLogger",
    - contact: [
        "033-900-5618",
        "450-834-6327",
        "785-957-5711"
      ],
    - memberOf: [
        "exercise 3",
        "alcoholism 1",
        "depression 1",
        "depression 2"
      ],
    - address: {
        state: "VT",
        city: "East Jenevachester",
        street: "Devora Lodge",
        zipcode: "72136-0162"
      },
      name: "Bradley Schuppe"
    }
  }
```

```
Duration of all jobs: 0.018 sec
```

Success: Query Complete

It’s now question-answering time! Answer each of the following questions after first following the instructions (if any) that they contain.

Note: For this assignment, you must turn in a PDF that you have created by copying/pasting from the query interface into a copy of HW8 template and **a zipped file (12345678_John_Doe.zip) that contains all queries and their outputs with specified file names to EEE**. Please choose “JSON” as the output format. When you copy and paste your query and result, do ***not*** take a screenshot. Instead, use the text copy and paste feature.

NOTE: Check the DDL of the dataset to determine the attribute names and structures.

1. [10 pts] Looking at the **Schema** file (init_dataverse.txt), which dataset/s in AsterixDB can be classified as being in 1NF based on the DDL you've been given? The commented section describes what records could be stored in those datasets. For this question only, we consider data stored in datasets with open data types has no extra fields. Write down the dataset name(s) and briefly explain your answer(s).

In 1NF:

Not in 1NF:

2. [10pts] There is another **NoSchema** file named "noschema_alternative.txt", which can also be used to create datatypes and datasets. Repeat **step 0** but this time use the **NoSchema** version. You can use the same load dataset as in step 0, but you need to change the 'USE' statement to be '**USE NoSchema**'. Like this:

```
USE NoSchema;

LOAD DATASET Observations USING localfs...
```

Compare the data type definitions in both the **Schema** and **NoSchema**. How is the **NoSchema** different from **Schema** in terms of number/type of attributes? Is there any difference in the way the data is showing in both versions? (You can make some queries to verify your answer, but you don't have to attach them to the answer.) **Note**: the dataverse that contains the schema version is 'HW8' and the non-schema version is 'NoSchema'. You only need to use 'NoSchema' for this question.

3. [10 pts] Looking back at the E-R diagram from the PHLogger conceptual design, compare and contrast the MySQL schema from the past SQL HW assignments with the AsterixDB schema (given in the Schema DDL version). You will find that we have made some different design decisions here in the NoSQL database case. Very briefly explain, after looking at the schema and also exploring the data (e.g., by looking at the DDL statements in the script and by running "SELECT VALUE u FROM Users u LIMIT 10;" and similarly for Events), how we have captured the information from the following E-R entities differently in AsterixDB and what the benefit(s) of our new design probably are:

Users:

Events:

4. [12 pts] For the user whose id is “97”, print his/her **user’s id**, **name**, **email address**, **most recent heart rate**, **the kind of the observer that recorded this reading**, **the start timestamp of this reading**. The recentness of a heart rate reading is determined by its start timestamp. [Result size: 1]

Sample output: { "id": 97, "name": "Gwenn Dooley", "email": "gwenn.dooley@uci.com", "rate": 21, "kind": "watch", "startTimestamp": "2019-04-15T13:00:00.000Z" }

[8 pts] Query (Please also submit this query in Question_4.txt within your zip file to EEE):

[4 pts] Result:

5. [14 pts] List **all the attributes** of the top **three** users who have joined the most number of interest groups (**Hint**: You might want to check out the len() function here). [Result size: 3]
Sample output: { "id": 2, "utype": "PHLogger", "contact": [ ], "memberOf": ["alcoholism 1", "exercise 2", "asthma 1"], "address": { "state": "NC", "city": "Murazikfort", "street": "Timmy Club", "zipcode": "61183" }, "passwd": "aa8ed233b0aa9b7492bd58910c5f1392", "email": "bethany.macgyver@uci.com", "name": "Bethany MacGyver" }...

[7 pts] Query (Please also submit this query in Question_5.txt within your zip file to EEE):

[4 pts] Result:

[3 pts] Now try the same query on the **NoSchema** version of the PHLogger dataverse (**‘NoSchema’**).

a) Did it work?
b) Were the results different?
c) What does this tell you?

6. [14 pts] List the **ids** of the users who are members of groups with the topic of “diabetes” and have an average heart rate greater than 85.
(HINT: You might find ‘IN’ useful for checking the existence of a nested array element.) [Result size: 4]
Sample output: { "id": 9 }...

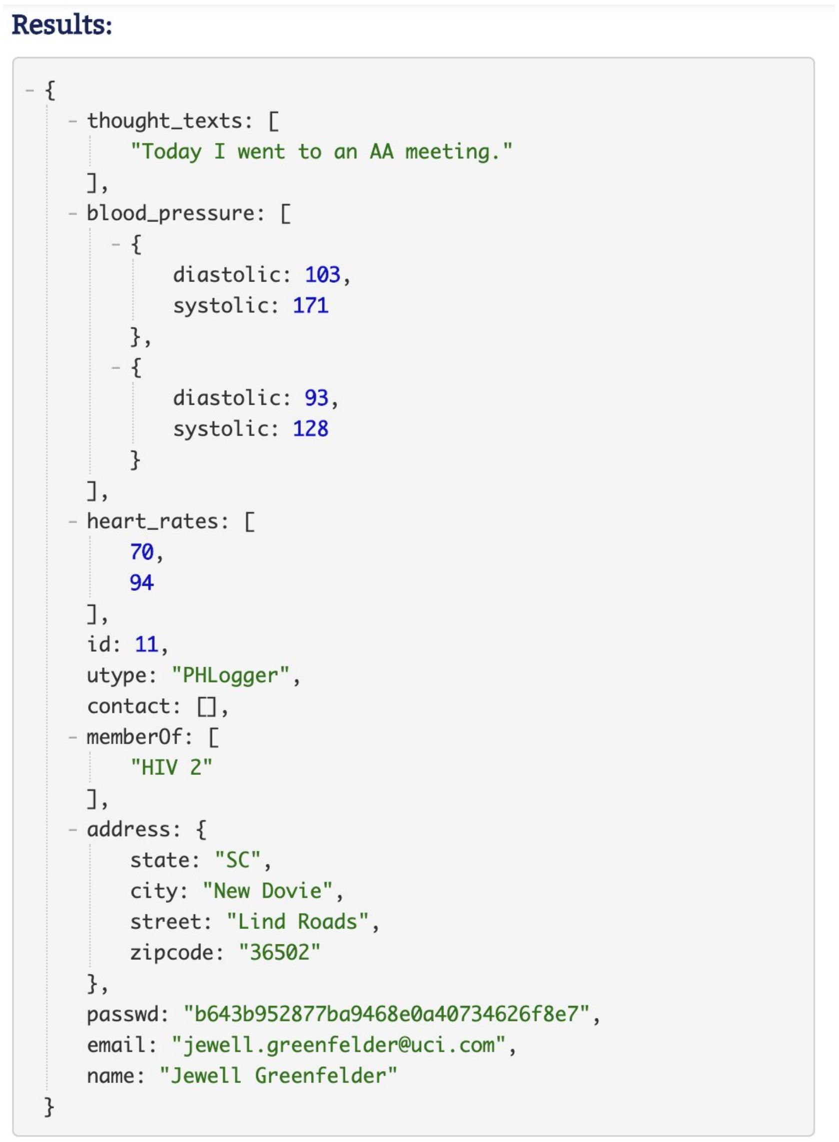
[10 pts] Query (Please also submit this query in Question_6.txt within your zip file to EEE):

[4 pts] Result:

7. [18 pts] Write a query to return **the attributes of the user** with id “85”, together with **the texts of all thoughts** posted by this users, **all blood pressure readings** (including both diastolic and systolic) and **heart rate readings** recorded by observers owned by this user. (You can use user.* to return all attributes from a user. You might also want to read the SELECT VALUE info here and review the SQL++ tutorial info on how to form a nested array of values.) [Result size: 1]

Sample output: { "thought_texts": ["Today I went to an AA meeting."], "blood_pressure": [{ "diastolic": 93, "systolic": 128 }, { "diastolic": 103, "systolic": 171 }], "heart_rates": [70, 94], "id": 11, "utype": "PHLogger", "contact": [ ], "memberOf": ["HIV 2"], "address": { "state": "SC", "city": "New Dovie", "street": "Lind Roads", "zipcode": "36502" }, "passwd": "b643b952877ba9468e0a40734626f8e7", "email": "jewell.greenfelder@uci.com", "name": "Jewell Greenfelder" }

To help you understand, here is a formatted version of the answer:



Results:

```
- {
  - thought_texts: [
        "Today I went to an AA meeting."
    ],
  - blood_pressure: [
      - {
            diastolic: 103,
            systolic: 171
        },
      - {
            diastolic: 93,
            systolic: 128
        }
    ],
  - heart_rates: [
        70,
        94
    ],
    id: 11,
    utype: "PHLogger",
    contact: [],
  - memberOf: [
        "HIV 2"
    ],
  - address: {
        state: "SC",
        city: "New Dovie",
        street: "Lind Roads",
        zipcode: "36502"
    },
    passwd: "b643b952877ba9468e0a40734626f8e7",
    email: "jewell.greenfelder@uci.com",
    name: "Jewell Greenfelder"
}
```

[10 pts] Query (Please also submit this query in Question_7.txt within your zip file to EEE):

[4 pts] Result:

[4 pts] As NoSQL provides nested attributes, we could have pushed all the thoughts, observations into the Users dataset. Briefly explain what could be the reason(s) that we didn't do that.

8. [12 pts] Write a query to print the **user id** and **the number of thoughts** of users who have posted at least 9 thoughts. [Result size: 6]
Sample output: { "id": 1, "cnt": 1 }...

[8 pts] Query (Please also submit this query in Question_8.txt within your zip file to EEE):

[4 pts] Result: