Lab 4: DynamoDB

DSCI 551 – Spring 2024

Due: 11:59pm, April 10, 2024, Wednesday

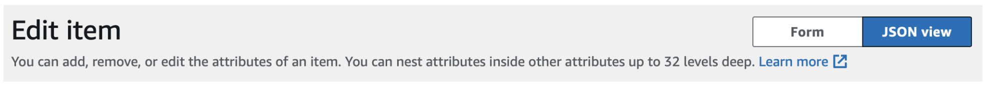
Points: 10

Create a DynamoDB table of your choice (except for the book example shown in class), with the following requirements:

a. The table must have a sort key
b. The table must contain at least one item
c. The item must have attributes of all the following types:
    i. String
    ii. Number
    iii. String set/Number set
    iv. List
    v. Map

Tasks:

1. Explain why you selected the particular attributes as the partition key and sort key.
    [+2] clearly explain the reasons for key selections
    [-1] for each key (partition or sort) explanation missing
2. Insert at least one item into the table that meets all the type requirements mentioned above. Show a screenshot of this item in the DynamoDB console and identify each data type in the item.
    [+3] all requirements met (screenshot of the item and data type identification)
    [-3] missing screenshot of the item
    [-2] the screenshot cannot clearly show each data type or there is no text explanation (some students explain it in Q3, which is also fine)
    [-0.5] for each missing data type
3. Show the `JSON view` of the above item and explain its format (e.g., how each attribute's data type is presented in JSON view)
    a. The `JSON view` can be found in DynamoDB console: DynamoDB > Explore items > Edit item > JSON view


Edit item
You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

    [+3] all requirements met (json view and format explanation)
    [-3] missing json view
    [-2] no explanation of the format
4. Execute the scan and query methods on your table. Show a screenshot for each operation.
    [+2] show screenshots for both scan and query methods
    [-1] for each screenshot missing

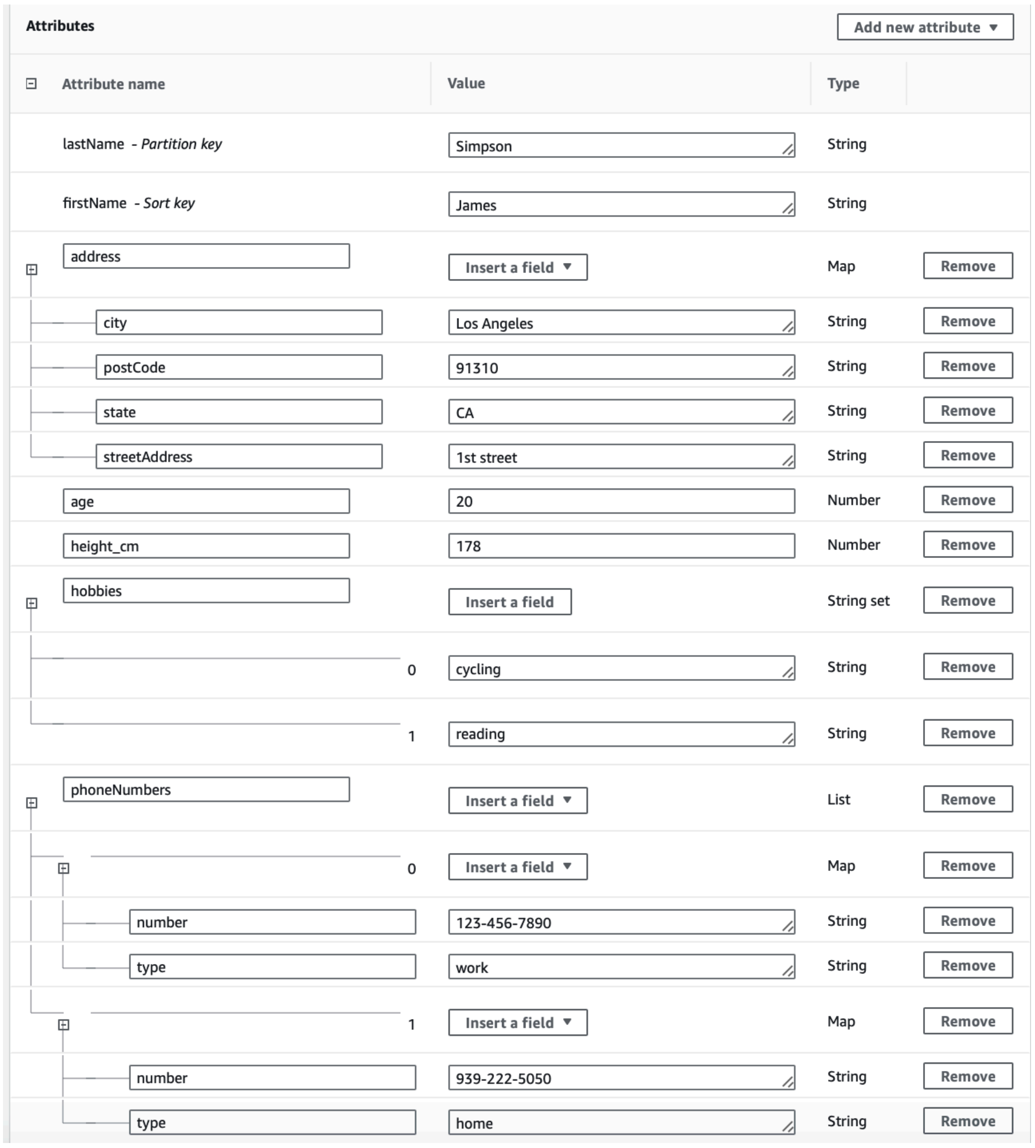
Submission Instructions:

1. Submit one document (word document or PDF) containing all the required answers and screenshots for each task
2. Clearly mark each task with its number.

Sample Solutions: I created a DynamoDB table called `students`

1. In DynamoDB, the partition key determines how data is distributed across the system, while the sort key allows the data within each partition to be ordered and enables efficient range queries. I chose `lastName` as the partition key because it groups individuals by family, which can be useful for queries targeting specific surnames. I chose `firstName` as the sort key because it can make sorting and querying within family groups much easier.

2.



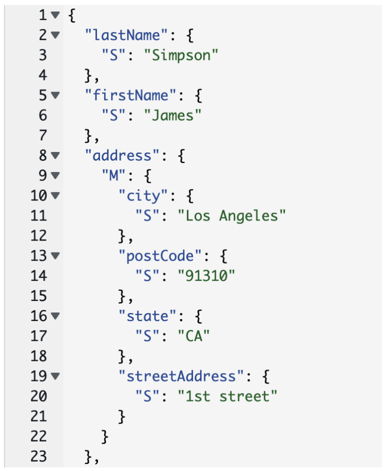
Attributes

Attribute name	Value	Type
lastName - *Partition key*	Simpson	String
firstName - *Sort key*	James	String
address		Map
city	Los Angeles	String
postCode	91310	String
state	CA	String
streetAddress	1st street	String
age	20	Number
height_cm	178	Number
hobbies		String set
0	cycling	String
1	reading	String
phoneNumbers		List
0		Map
number	123-456-7890	String
type	work	String
1		Map
number	939-222-5050	String
type	home	String

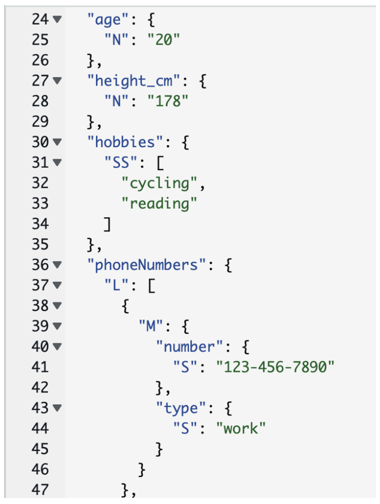
Attributes and Data Types:

- String: ‘firstName’ and ‘lastName’
- Number: ‘age’ and ‘height_cm’
- String set: ‘hobbies’
- List: ‘phoneNumbers’ containing phone number entries
- Map: ‘address’, including street, city, state, postal code

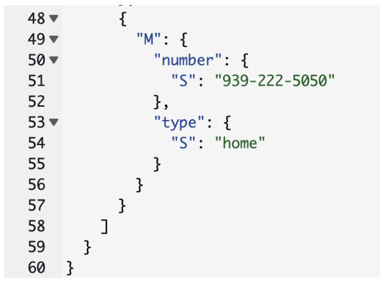
3. JSON view:



```
{
  "lastName": {
    "S": "Simpson"
  },
  "firstName": {
    "S": "James"
  },
  "address": {
    "M": {
      "city": {
        "S": "Los Angeles"
      },
      "postCode": {
        "S": "91310"
      },
      "state": {
        "S": "CA"
      },
      "streetAddress": {
        "S": "1st street"
      }
    }
  },
```



```
  "age": {
    "N": "20"
  },
  "height_cm": {
    "N": "178"
  },
  "hobbies": {
    "SS": [
      "cycling",
      "reading"
    ]
  },
  "phoneNumbers": {
    "L": [
      {
        "M": {
          "number": {
            "S": "123-456-7890"
          },
          "type": {
            "S": "work"
          }
        }
      },
```

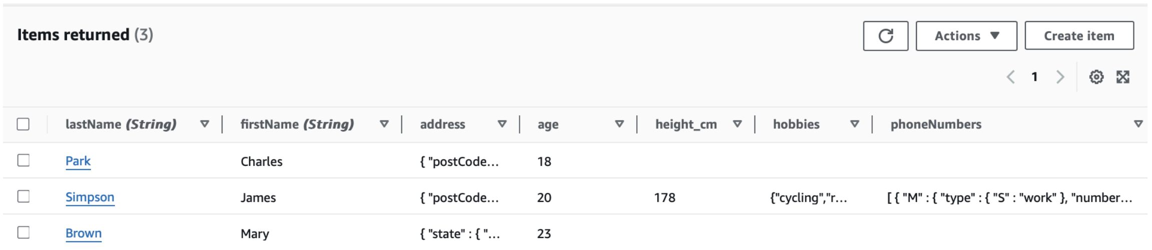


```
      {
        "M": {
          "number": {
            "S": "939-222-5050"
          },
          "type": {
            "S": "home"
          }
        }
      }
    ]
  }
}
```

In the DynamoDB JSON view, each key is an attribute name and the value is another object specifying the type of the attribute and the actual data. Each attribute’s data type is represented as follows:

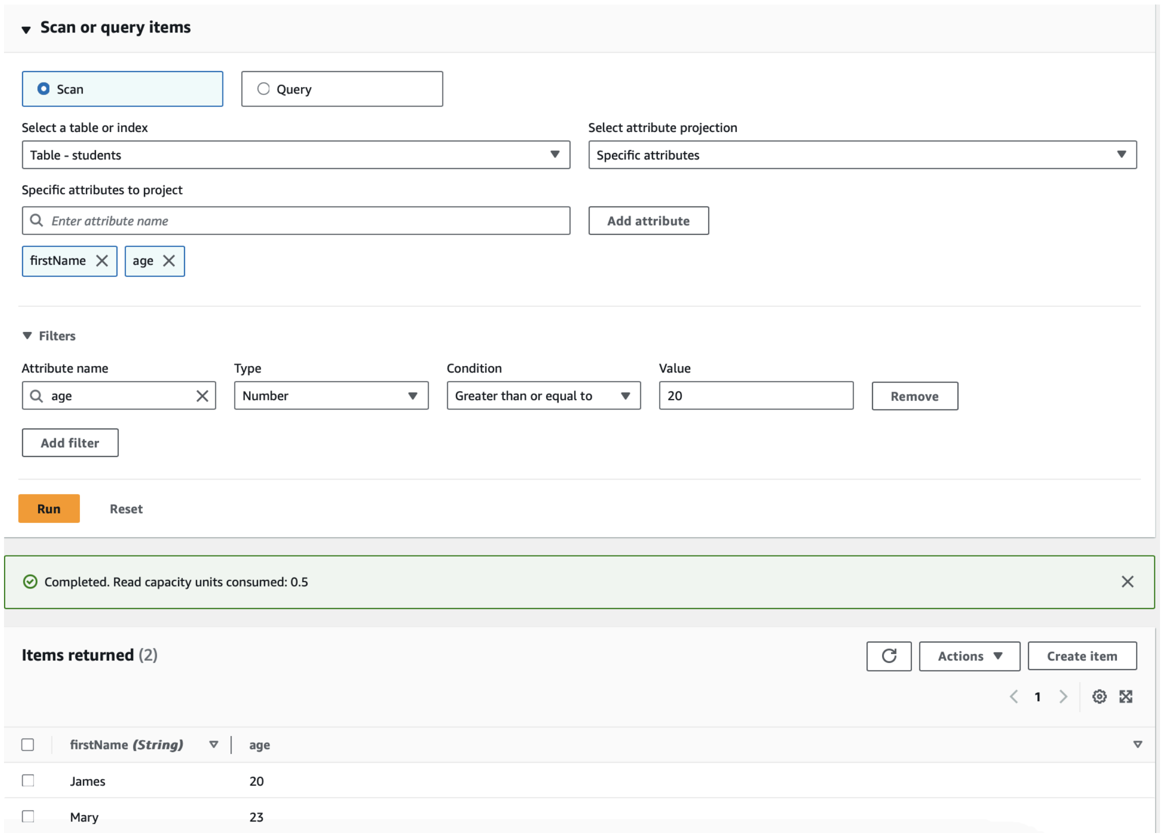
- **String (S):** lastName and firstName are strings, indicated by "S": "value". For example, "lastName": {"S": "Simpson"} means lastName is a string with the value "Simpson".
- **Number (N):** age and height_cm are numbers, shown as "N": "value". For instance, "age": {"N": "20"} means that age is a number with the value 20.
- **String Set (SS):** hobbies is a string set, represented by "SS": ["value1", "value2"], like "hobbies": {"SS": ["cycling", "reading"]} indicating hobbies includes the strings "cycling" and "reading".
- **List (L):** phoneNumbers is a list, denoted by "L": [{...}, {...}], where each item in the list is a map of phone number entries.
- **Map (M):** address is a map, shown as "M": { "key1": {"S": "value1"}, "key2": {"S": "value2"}}. For example, "address": {"M": {"city": {"S": "Los Angeles"}}}, indicating address contains structured data like city, postCode, state, and streetAddress.

4.
- Items in the table


Items returned (3)

	lastName (String)	firstName (String)	address	age	height_cm	hobbies	phoneNumbers
☐	Park	Charles	{ "postCode…	18			
☐	Simpson	James	{ "postCode…	20	178	{"cycling","r…	[{ "M" : { "type" : { "S" : "work" }, "number…
☐	Brown	Mary	{ "state" : { "…	23			

- scan result


Scan or query items

Select a table or index: Table - students

Select attribute projection: Specific attributes

Specific attributes to project: firstName, age

Filters

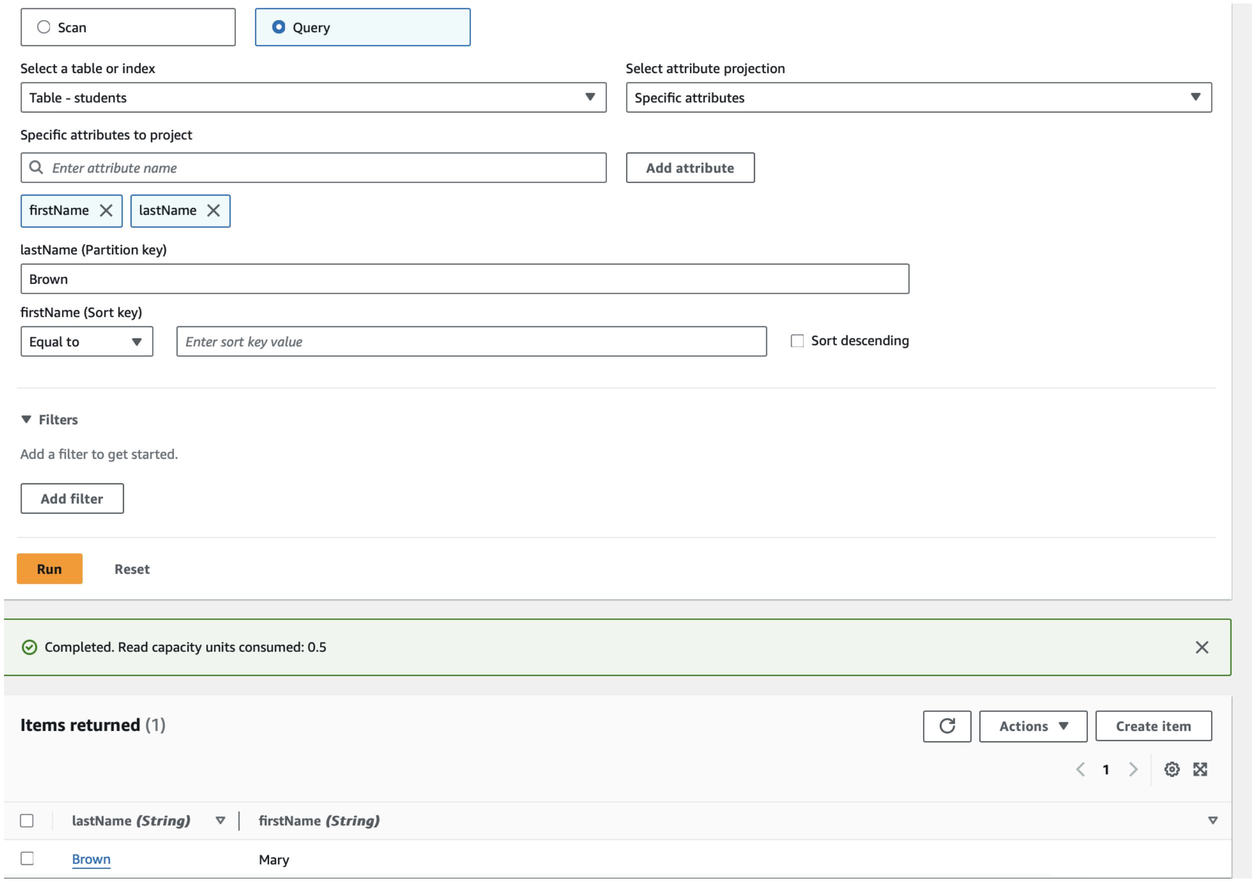
Attribute name	Type	Condition	Value
age	Number	Greater than or equal to	20

Completed. Read capacity units consumed: 0.5

Items returned (2)

	firstName (String)	age
☐	James	20
☐	Mary	23

- query result



Scan [✓ Query]

Select a table or index: Table - students

Select attribute projection: Specific attributes

Specific attributes to project

Enter attribute name

Add attribute

firstName lastName

lastName (Partition key): Brown

firstName (Sort key): Equal to | Enter sort key value | Sort descending

Filters

Add a filter to get started.

Add filter

Run Reset

Completed. Read capacity units consumed: 0.5

Items returned (1)

Actions Create item

lastName *(String)*	firstName *(String)*
Brown	Mary