**Answer Questions With Explanation**

# Theory Questions And Answers:

1. What is meant by the REST API?

* It is an architectural style for an application program interface (API) that uses HTTP requests to access
* **API** is an application programming interface. It is a set of rules that allow programs to talk to each other. The developer creates the API on the server and allows the client to talk to it
* **REST** determines how the API looks like. It stands for “Representational State Transfer”. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL
* A request is made up of four things:  
  + The endpoint

- The root-endpoint is the starting point of the API you’re requesting from. For example: It is the starting point of the API local server

* + The method

- The method is the type of request you send to the server.

- It always has the main methods which are GET (the server looks for the data you requested and sends it back to you), POST (the server creates a new entry in the database and tells you whether the creation is successful), PUT/PATCH (the server updates an entry in the database and tells you whether the update is successful), DELETE (the server deletes an entry in the database and tells you whether the deletion is successful) to implement communication with client for transfer data

* + The headers  
      
    - Headers are used to provide information to both the client and server. It can be used for many purposes, such as authentication and providing information about the body content.
  + The data (or body)  
      
    - The data (sometimes called “body” or “message”) contains information you want to be sent to the server. This option is only used with POST, PUT, PATCH or DELETE requests.

1. What is a SOAP service?

* It is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to provide extensibility, neutrality, verbosity and independence.
* The main format for exchanging structured information is XML (Extensible Markup Language)

1. What is a relational database?

* It is a type of database. It uses a structure that allows us to identify and access data in relation to another piece of data in the database. Often, data in a relational database is organized into tables
* Tables can have hundreds, thousands, sometimes even millions of rows of data. These rows are often called records.
* Tables can also have many columns of data. Columns are labeled with a descriptive name (say, age for example) and have a specific data type
* Columns store string data types, whereas age stores integer data types. The set of columns and data types make up the schema of this table.

1. What is a NoSQL database?

* It is non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.
* NoSQL databases are databases that store data in a format other than relational tables
* A common misconception is that NoSQL databases or non-relational databases don’t store relationship data well. NoSQL databases can store relationship data—they just store it differently than relational databases do. In fact, [when compared with SQL databases](https://www.mongodb.com/nosql-explained/nosql-vs-sql), many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn’t have to be split between tables.
* NoSQL data models allow related data to be nested within a single data structure.

1. What is an ORM? Give at least one example.

* Object Relational Mapping (ORM) is a functionality which is used to develop and maintain a relationship between an object and relational database by mapping an object state to database column. It is capable to handle various database operations easily such as inserting, updating, deleting
* We have the various frameworks that function on ORM mechanism such as: Hibernate, TopLink, ORMLite, iBATIS, JPOX. The most popular is Hibernate.
* Hibernate is supporting for Object/Relational Mapping
* Hibernate ORM enables developers to more easily write applications whose data outlives the application process. As an Object/Relational Mapping (ORM) framework, Hibernate is concerned with data persistence as it applies to relational databases (via JDBC). Unfamiliar with the notion of ORM? Read here.
* Hibernate is supporting for JPA Provider
* Its own "native" API, Hibernate is also an implementation of the Java Persistence API (JPA) specification. As such, it can be easily used in any environment supporting JPA including Java SE applications, Java EE application servers, Enterprise OSGi containers, etc.
* Hibernate is supporting for Idiomatic persistence
* Hibernate enables you to develop persistent classes following natural Object-oriented idioms including inheritance, polymorphism, association, composition, and the Java collections framework. Hibernate requires no interfaces or base classes for persistent classes and enables any class or data structure to be persistent.
* Hibernate is supporting for High Performance
* Hibernate supports lazy initialization, numerous fetching strategies and optimistic locking with automatic versioning and time stamping. Hibernate requires no special database tables or fields and generates much of the SQL at system initialization time instead of at runtime. Hibernate consistently offers superior performance over straight JDBC code, both in terms of developer productivity and runtime performance.
* Hibernate is supporting for Scalability
* Hibernate was designed to work in an application server cluster and deliver a highly scalable architecture. Hibernate scales well in any environment: Use it to drive your in-house Intranet that serves hundreds of users or for mission-critical applications that serve hundreds of thousands.
* Hibernate is supporting for Reliable
* Hibernate is well known for its excellent stability and quality, proven by the acceptance and use by tens of thousands of Java developers.
* Hibernate is supporting for Extensibility
* Hibernate is highly configurable and extensible.

1. What is SQL Injection?

* SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.
* Normally, attacker will attack to query string in link box on browser and change input data for making some changes of query data to steal data or break system. For example, they used some rules to be used in query string. For example, retrieving hidden data (form: https://host\_domain/products?category=electrolic'--), subverting application logic (submitting the username administrator'-- and a blank password results in the following query: SELECT \* FROM users WHERE username = 'administrator'--' AND password = ''), retrieving data from other database tables (attacker can submit the input:' UNION SELECT username, password FROM users--) etc.

1. What is 2-factor authentication? Briefly describe an example.

* It is an extra layer of security used to make sure that people trying to gain access to an online account are who they say they are. First, a user will enter their username and a password. Then, instead of immediately gaining access, they will be required to provide another piece of information. This second factor could come from one of the following categories:
* Something you know: This could be a personal identification number (PIN), a password, answers to “secret questions” or a specific keystroke pattern
* Something you have: Typically, a user would have something in their possession, like a credit card, a smartphone, or a small hardware token
* Something you are: This category is a little more advanced, and might include biometric pattern of a fingerprint, an iris scan, or a voice print
* It is a potential compromise of just one of these factors won’t unlock the account. So, even if your password is stolen or your phone is lost, the chances of someone else having your second-factor information is highly unlikely. Looking at it from another angle, if a consumer uses 2FA correctly, websites and apps can be more confident of the user’s identity, and unlock the account.
* For example, 2-factor authentication to be built by Google. When you sign in to Google system, Google system will suggest you to provide more information about yourself for setting up 2-factor authentication function. After you completed it and login to the system, the system will require one more authentication by some authentication method as input PIN, input your number or input your second mail. If the information is completely exact, you will be forwarded to the system.

1. Briefly describe a secure method for saving user passwords on the DB.

* As I’ve known, before you save user passwords to DB, this password will be transformed by a non-reversible cryptographic function like outdated irreversible cryptographic functions (md5, sha1…), brute force, using of a Salt etc. With respective cryptographic functions, they will have solution to encryption and decryption respectively. After all that user still login successfully with their password to be secured. Especially, cryptographic function mechanism ensure that only system is possible to handle it and make security to user.
* For example, user’s password is entered with text/plain is “todo”. The text/plain value will be encrypted by cryptographic function into “HZBD^@gL\*wvoExo6yJ7hVB” and stored to DB. When user login to the system, the system will decrypt the password stored and check it to allow user login if it was valid and mapping with user’s password.

1. What is a Sticky Session in a Load Balanced System?

* It is a process in which a load balancer creates an affinity between a client and a specific network server for the duration of a session, (i.e., the time a specific IP spends on a website). Using sticky sessions can help improve user experience and optimize network resource usage.
* With sticky sessions, a load balancer assigns an identifying attribute to a user, typically by issuing a cookie or by tracking their IP details. Then, according to the tracking ID, a load balancer can start routing all of the requests of this user to a specific server for the duration of the session.

# Practice Questions And Answers:

## Write a query that extracts the number of people under the age of 30 and earning more than $ 50,000 a year

**Query Command:**

**select** *\** **from** records **where age**<30 **and over\_50k**=**true**

## Write a query that reports the average capital gain for each occupation category:

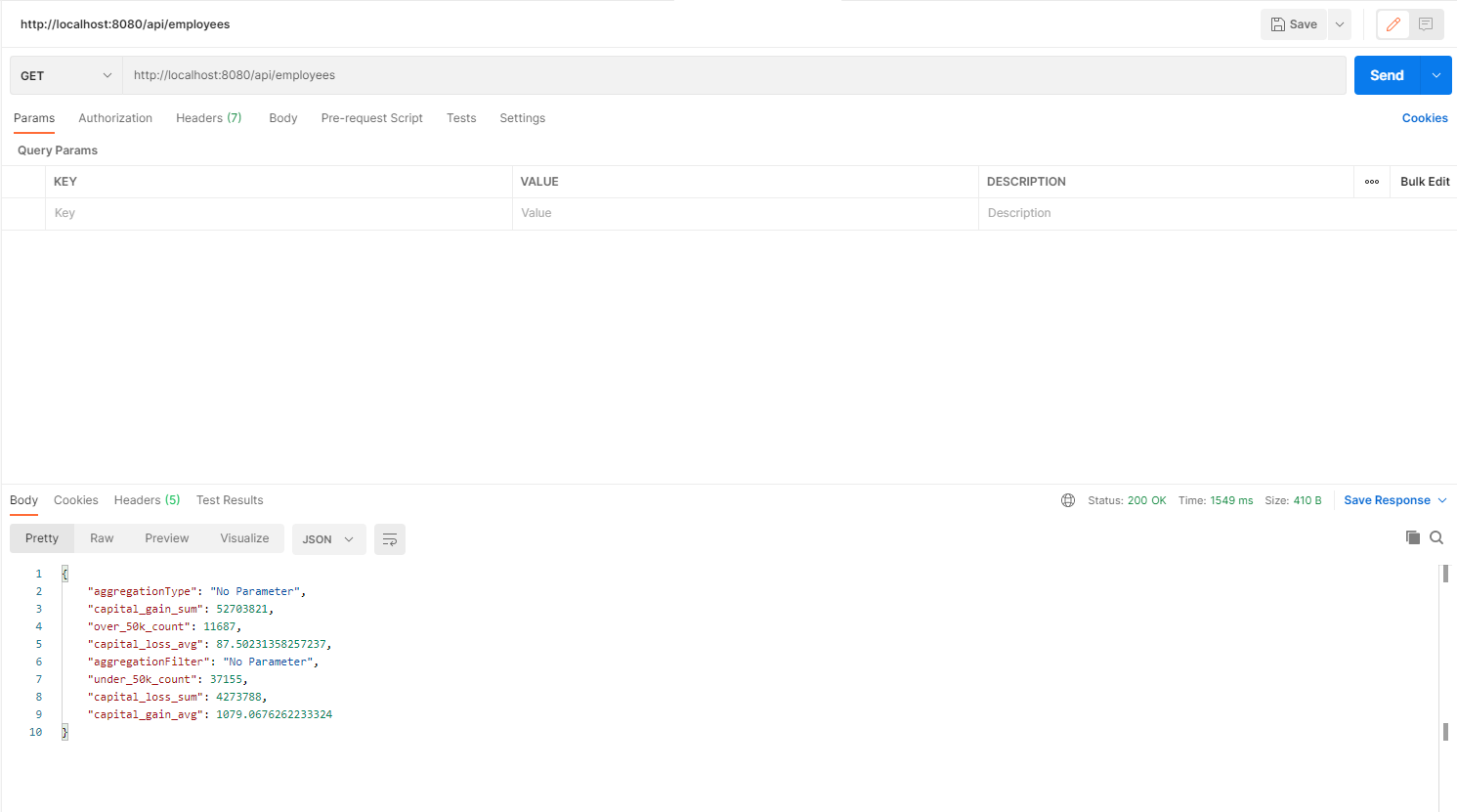
**Query Command:**

**select** main.occupations.**id**, main.occupations.**name**, (**select** *AVG*(main.records.**capital\_gain**) **from** records **where** records.**occupation\_id** = main.occupations.**id**) **AS** AverageCapitalGain **from** occupations

## The back-end system was built up successfully,

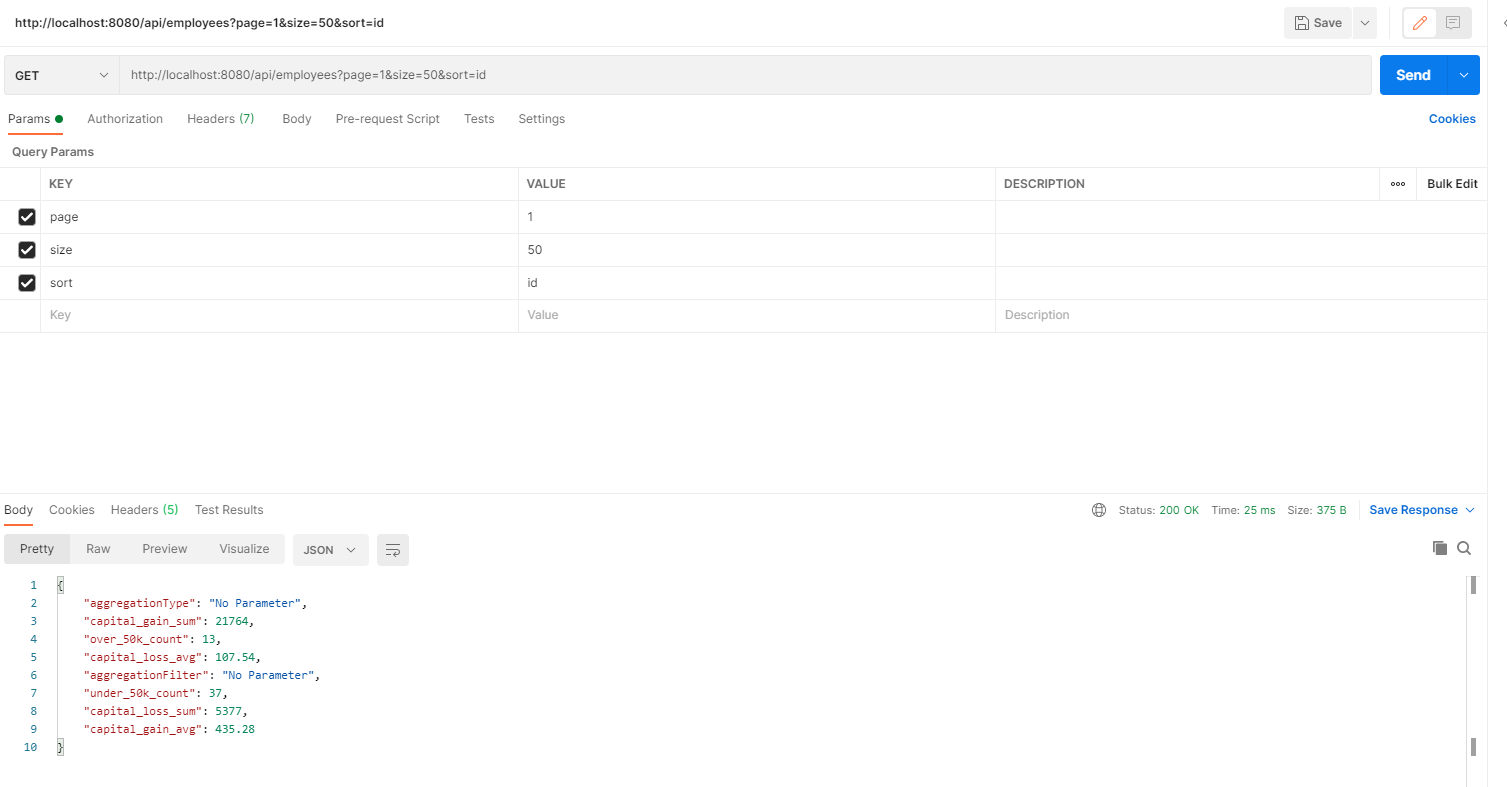
* You could access to local host address after you downloaded source code and built it in local server environment.
* Particularly, you can access to local host address links below to get expected result as requirement:
* Get response data based on all employees in records table without pagination and sort :

<http://localhost:8080/api/employees>



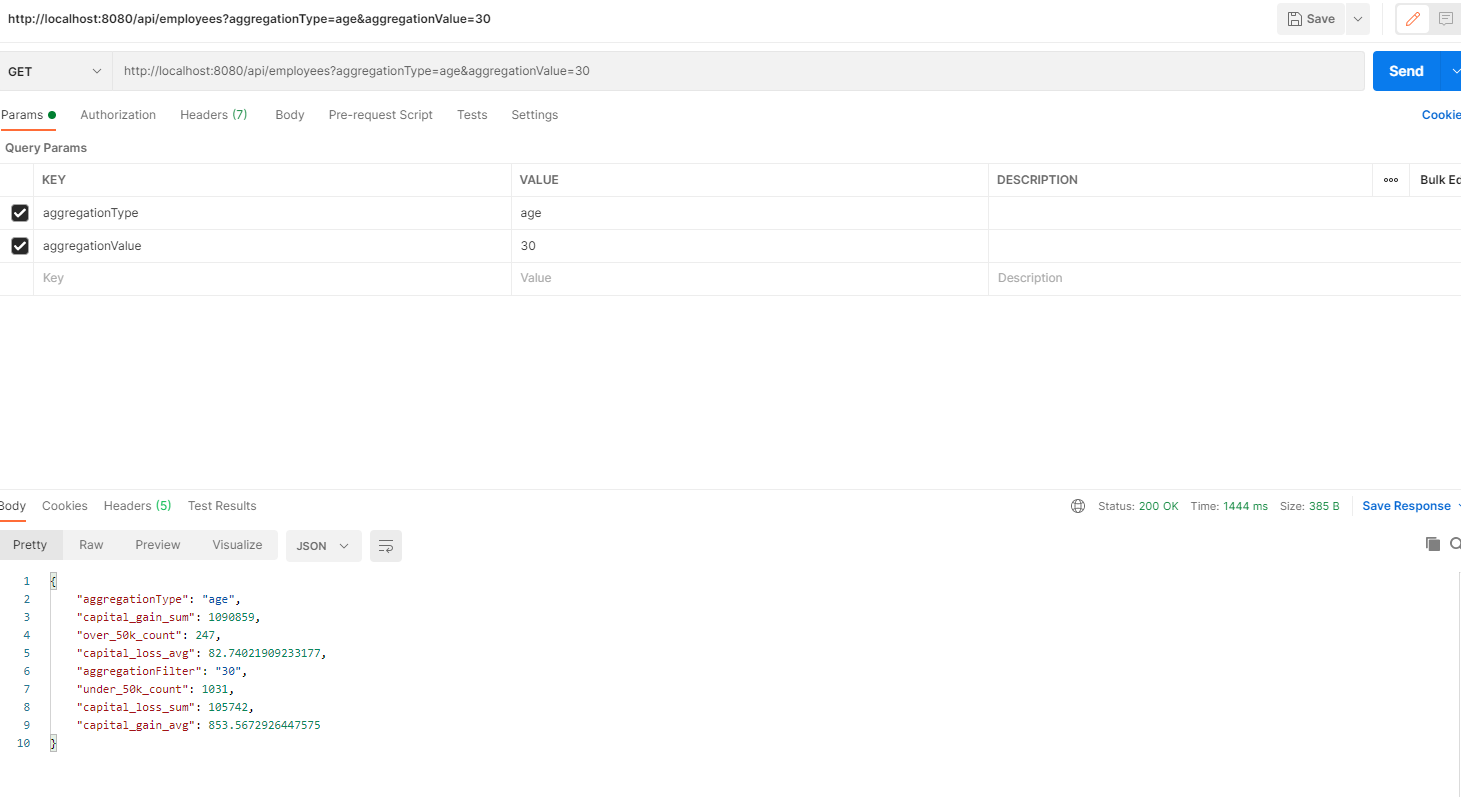
* Get response data based on employees in records table paginated and sorted:

<http://localhost:8080/api/employees?page=1&size=50&sort=id>



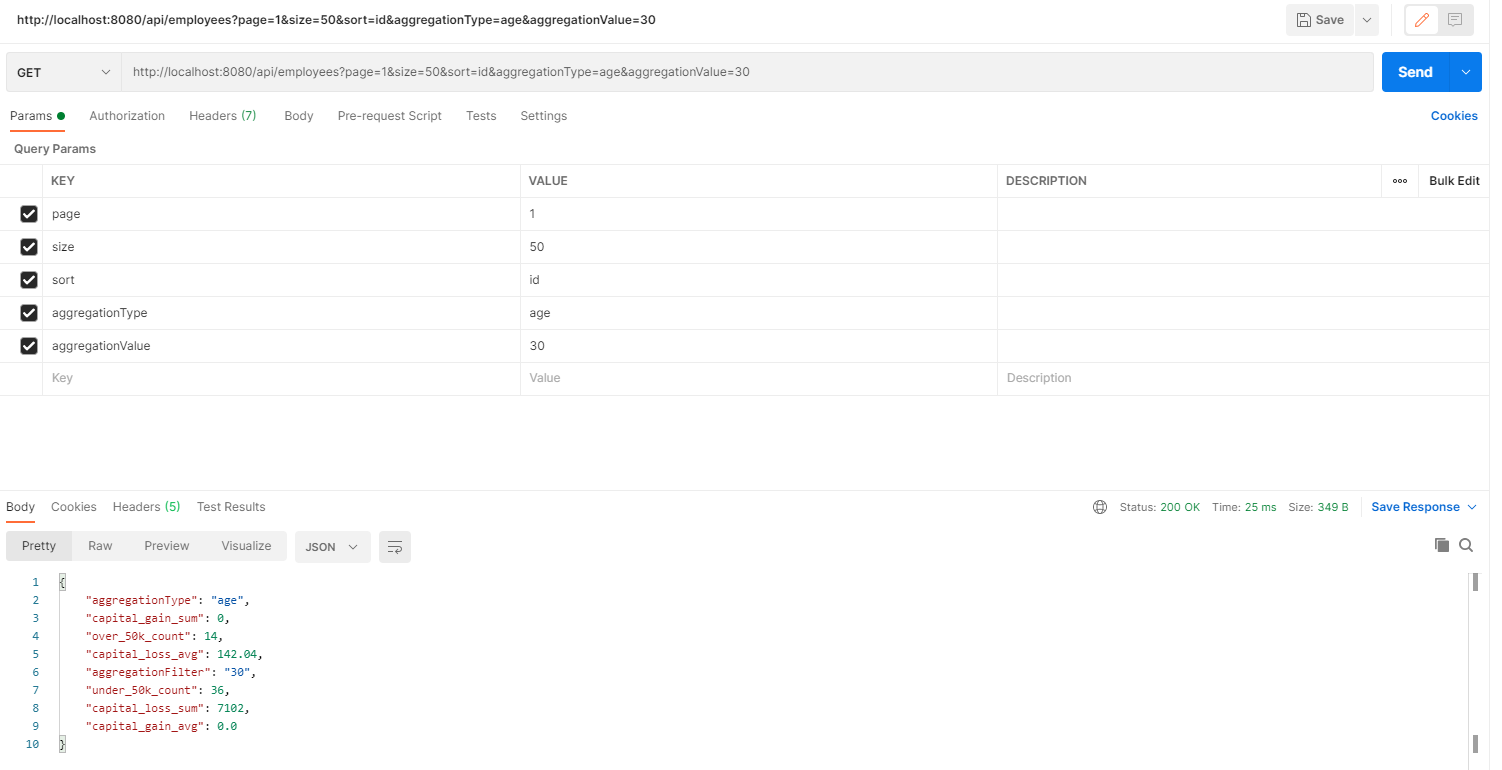
* Get response data based on employees in records table without pagination and sort and parameter inputs on query string :

<http://localhost:8080/api/employees?aggregationType=age&aggregationValue=30>



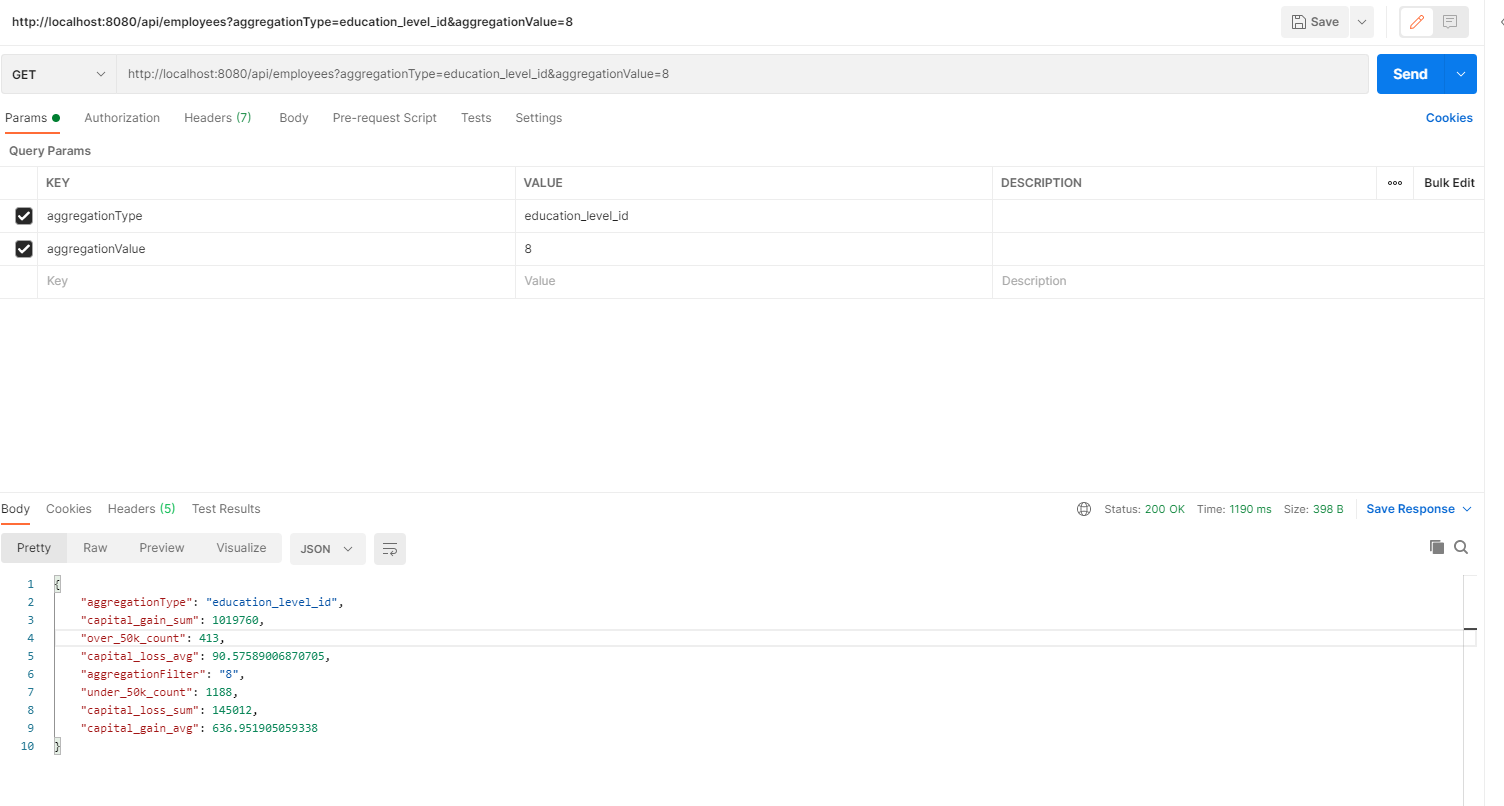
* Get response data based on employees in records table with pagination and sort and parameter inputs on query string :

<http://localhost:8080/api/employees?page=1&size=50&sort=id&aggregationType=age&aggregationValue=30>

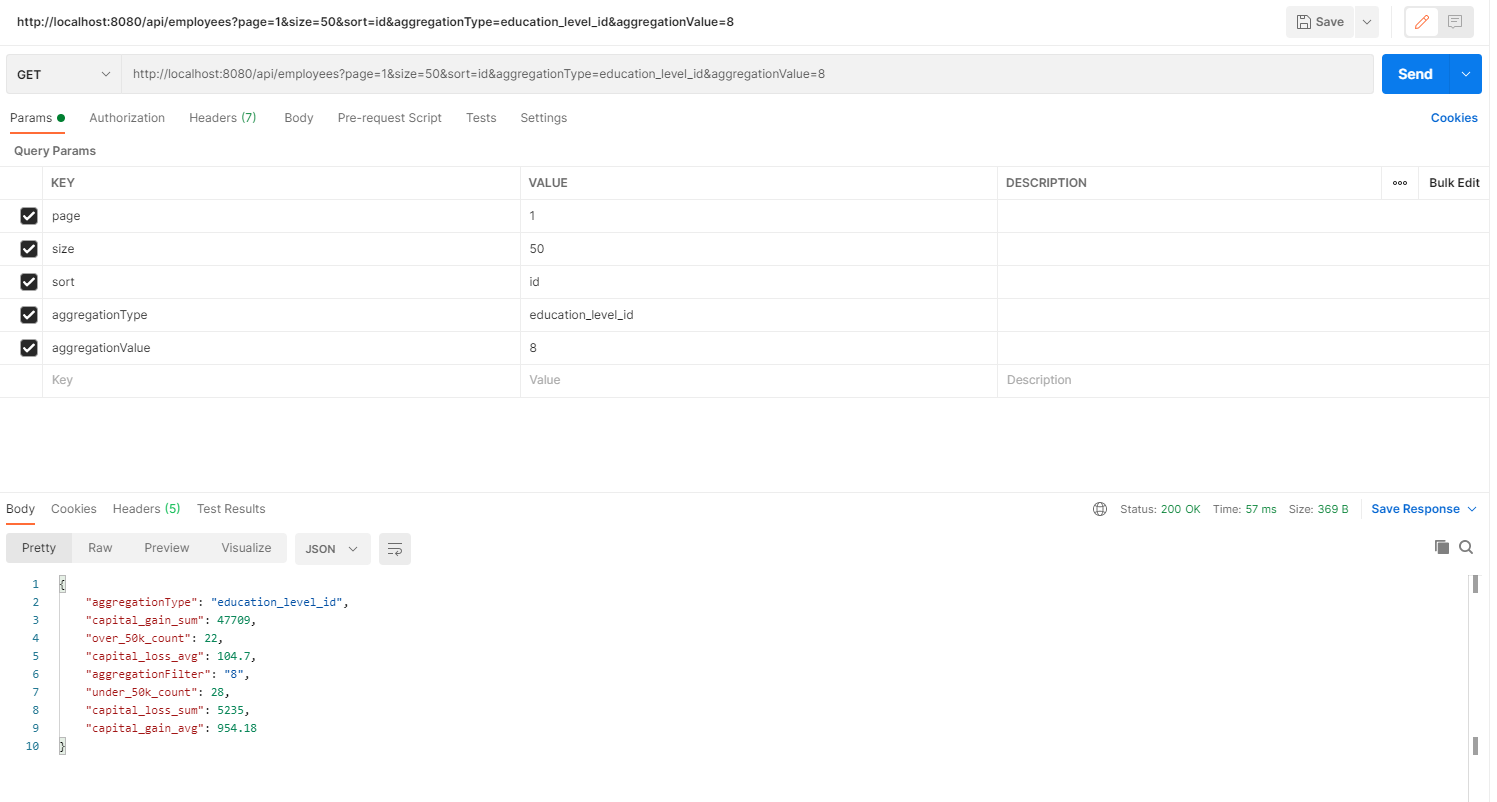


* To do the same with above case and change value for parameter inputs on query string. We will have the result following the below images:

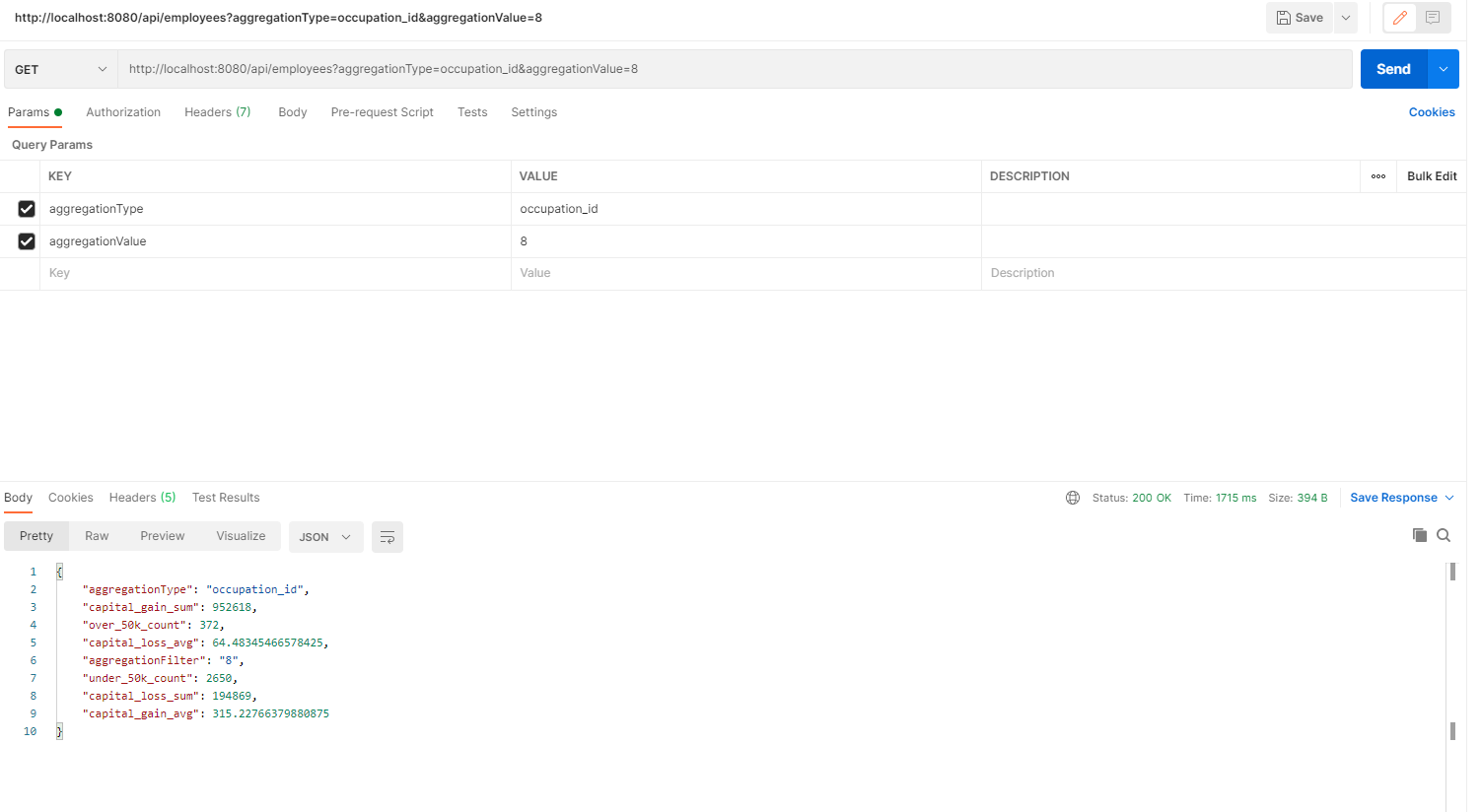
<http://localhost:8080/api/employees?aggregationType=education_level_id&aggregationValue=8>



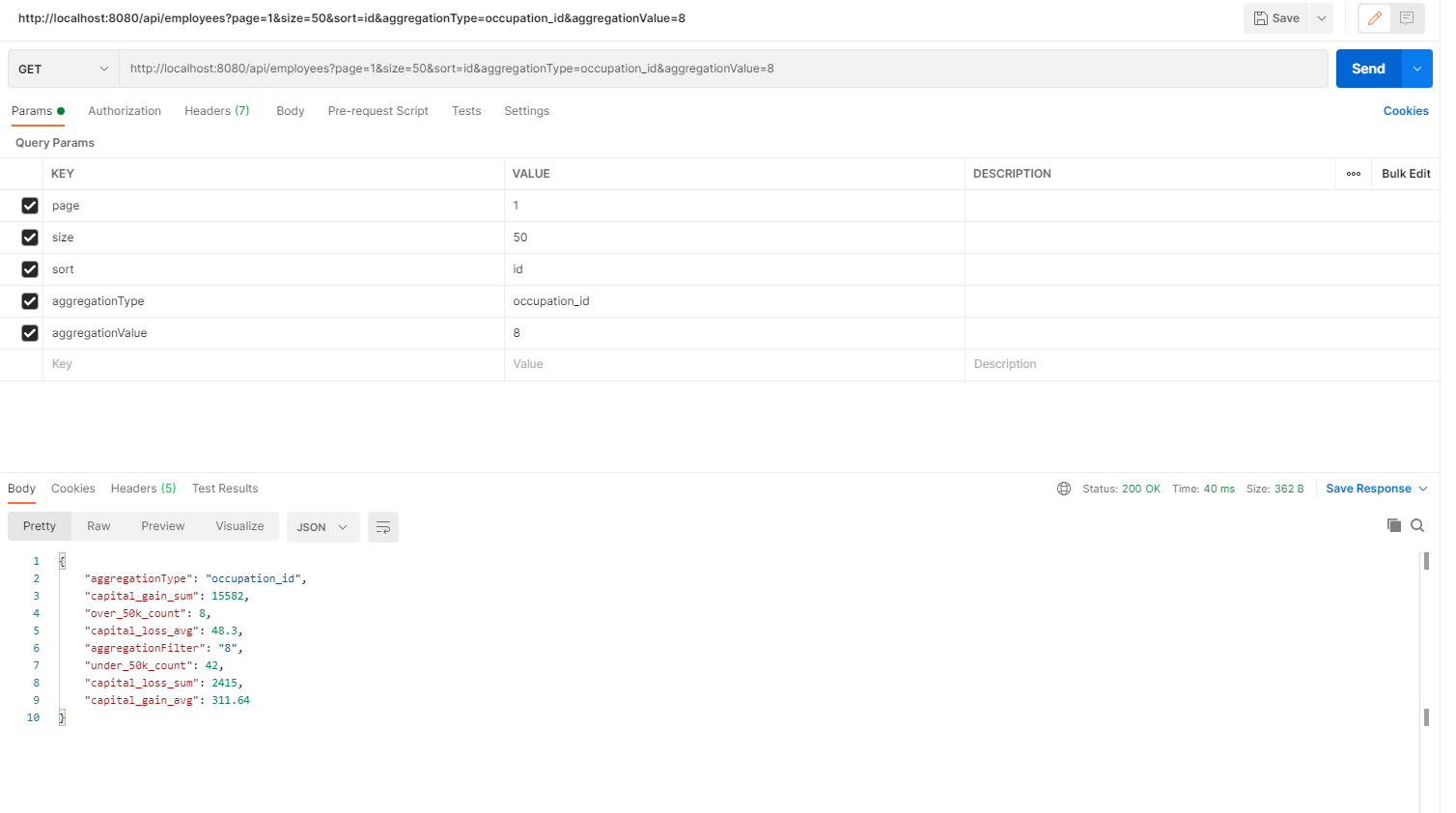
<http://localhost:8080/api/employees?page=1&size=50&sort=id&aggregationType=education_level_id&aggregationValue=8>



<http://localhost:8080/api/employees?aggregationType=occupation_id&aggregationValue=8>



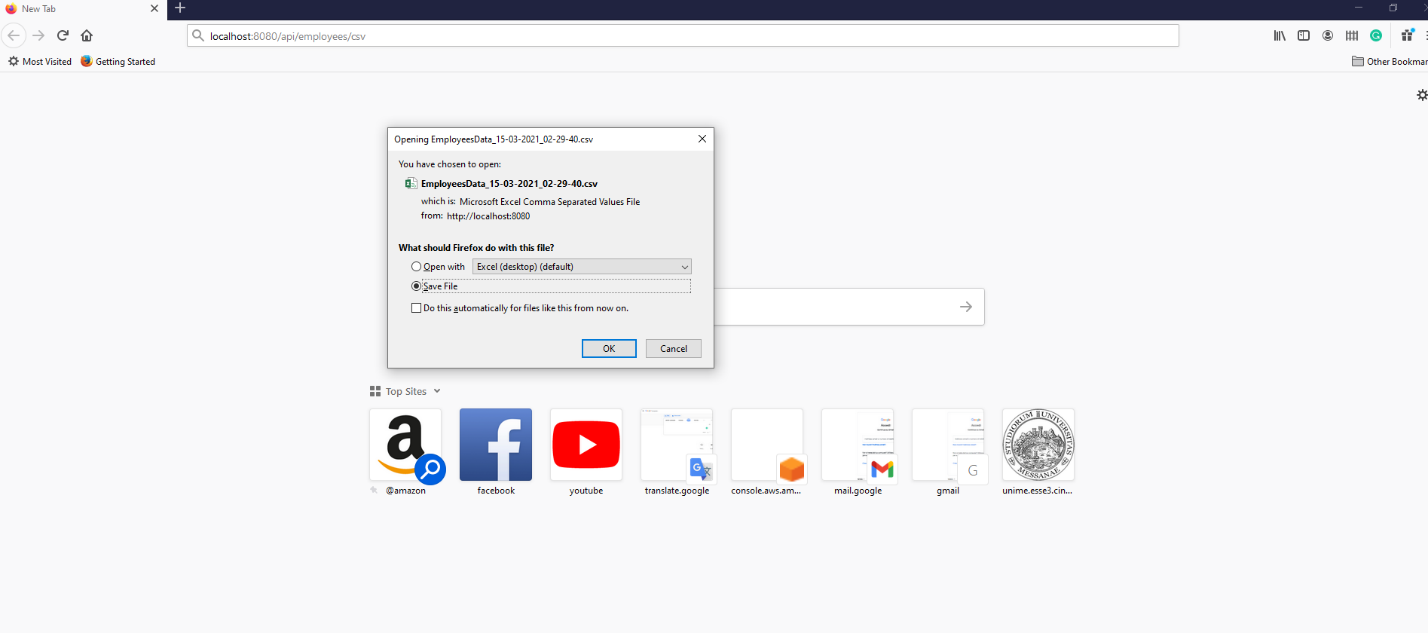
<http://localhost:8080/api/employees?page=1&size=50&sort=id&aggregationType=occupation_id&aggregationValue=8>

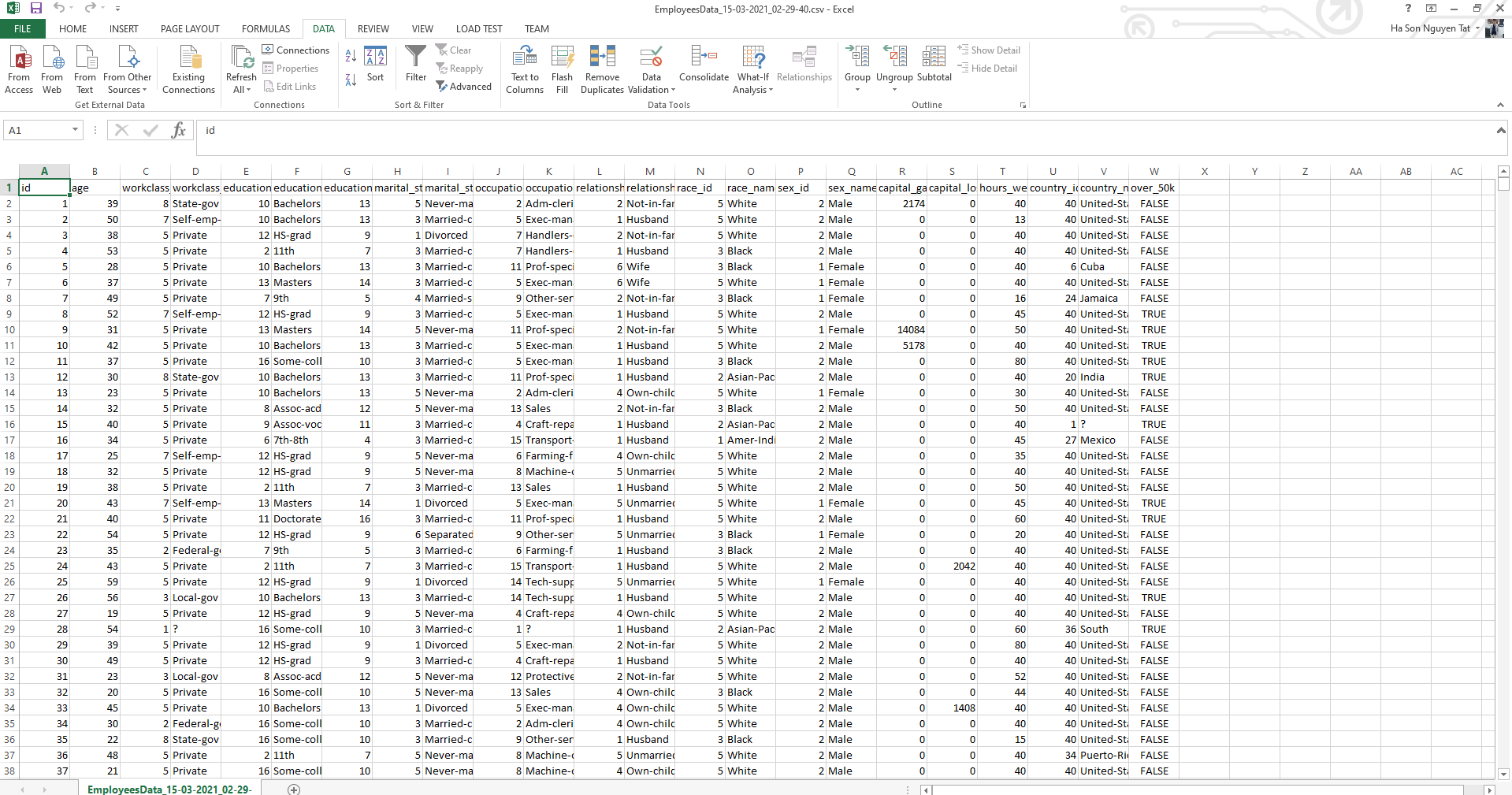


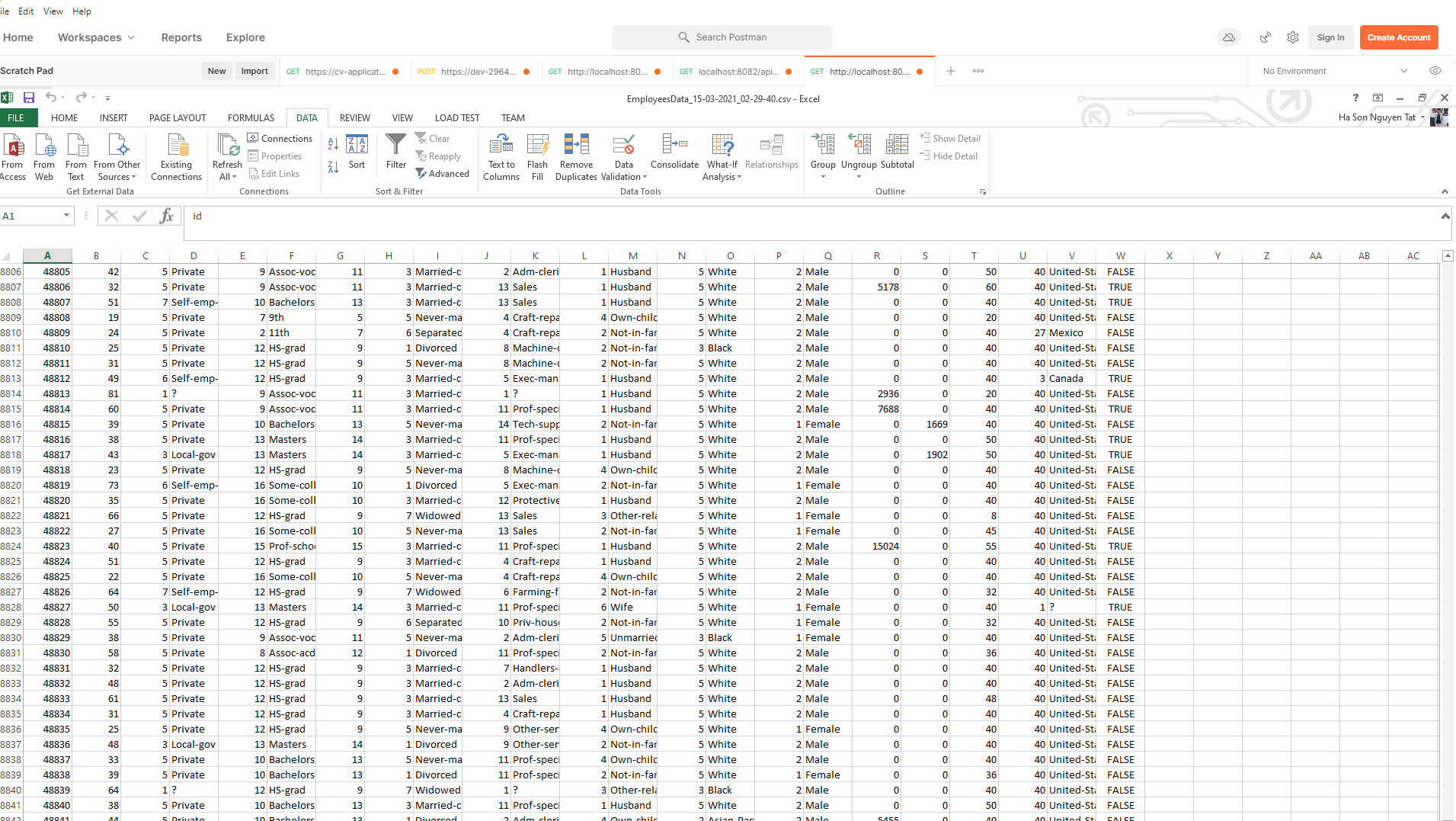
## ­­­The back-end system was built up successfully.

* You could access to local host address after you downloaded source code and built it in local server environment.
* Particularly, you can access to local host address links below to get expected result as requirement:
* Export data to CSV and download based on all employees in records table without pagination and sort :

<http://localhost:8080/api/employees/csv>







* Export data to CSV and download based on all employees in records table with pagination and sort :

<http://localhost:8080/api/employees/csv?page=1&size=50&sort=id>

