CS5500 Final Project Documentation

Group 9

Paul Cruz, Madhuri Palanivelu, Alex Yang

About Project Group 9 - Operational Planner:

This product is to help Managers of the MetMarket perform operational planning, amidst the COVID-19 pandemic. With issues like keeping senior citizens away from heavy crowds, keeping customers from waiting in long queues outside the store, etc., this product can help him/her get a better idea of the customer data and plan how they should plan their staffing.

The product will enable the Manager to use a Web Interface to generate customer data based on desired parameters and perform queries on the data as well.

Product Build/ Installation:

To use the web interface, the project needs to be built and certain dependencies need to be installed.

Required Installations:

Java (JDK 8)
MySql (Version 8.0 or higher) *[Keep Server on while running the program]*
Spring Boot
Postman (optional)

Required Dependencies:

Bootstrap
Gson
Json-simple
Mysql-connector-java
Spring-boot-starter-web
Ajax (3.5.1)
Moment

Project Build:

There are two ways the project can be built:

1. The project can be run as an ***executable JAR.***

   To run the project, open up a terminal/command prompt in the project directory and enter:

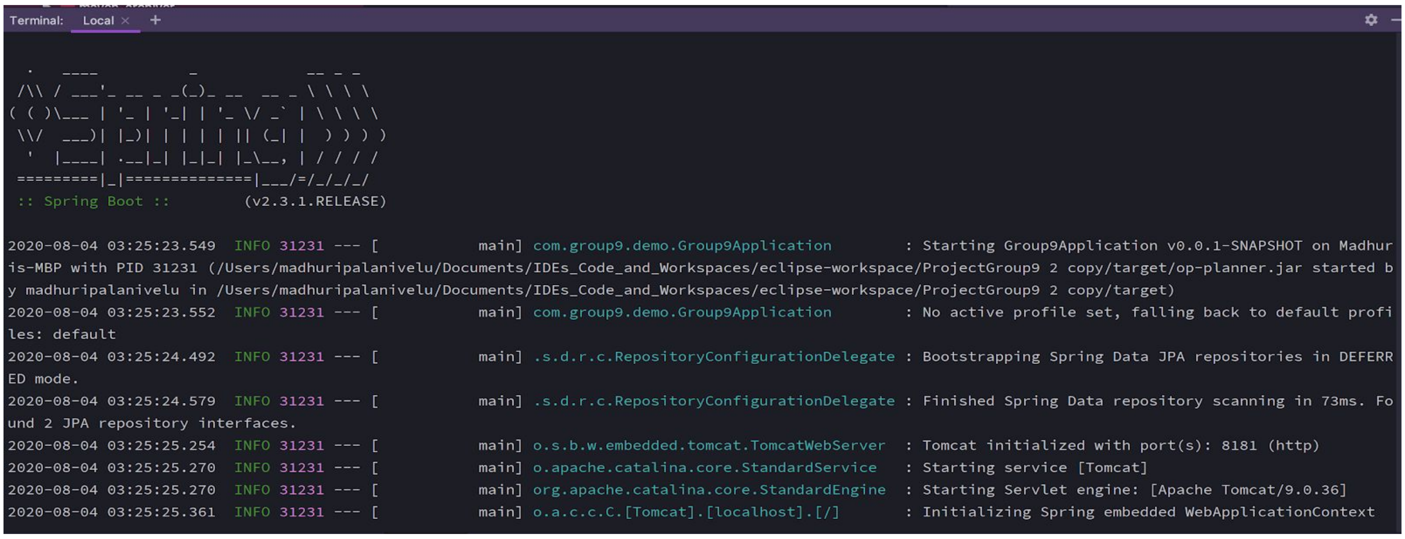
```
$ cd target
$ java -jar op-planner.jar
```

To stop the service while using the JAR, use Ctrl+C on the terminal/command prompt.

2. In case of running the project as a program from an editor, open the project directory and look through the path *src/main/java/com.group9.demo/* and run the file run the file *“Group9Application”.*

   To stop the service, simply use the Stop button of the coding editor in use.

Your terminal will start to display text like this:



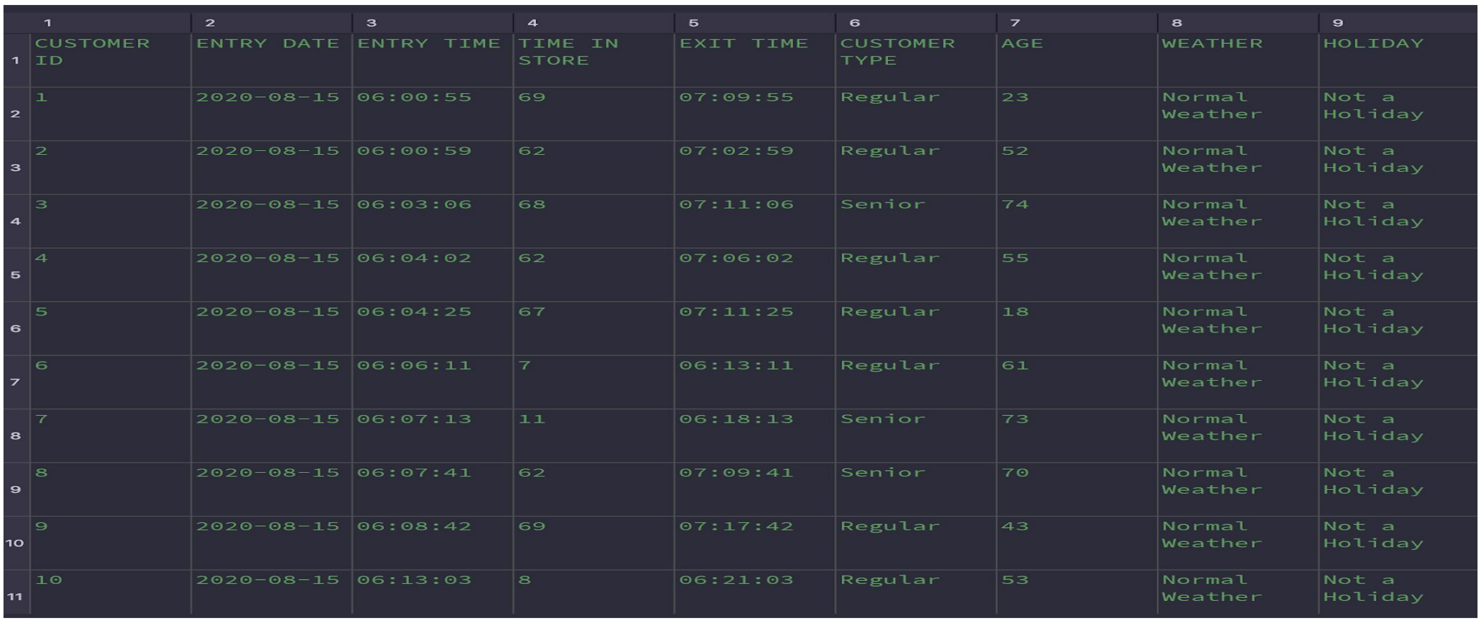
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.1.RELEASE)

2020-08-04 03:25:23.549  INFO 31231 --- [           main] com.group9.demo.Group9Application        : Starting Group9Application v0.0.1-SNAPSHOT on Madhur
is-MBP with PID 31231 (/Users/madhuripalanivelu/Documents/IDEs_Code_and_Workspaces/eclipse-workspace/ProjectGroup9 2 copy/target/op-planner.jar started b
y madhuripalanivelu in /Users/madhuripalanivelu/Documents/IDEs_Code_and_Workspaces/eclipse-workspace/ProjectGroup9 2 copy/target)
2020-08-04 03:25:23.552  INFO 31231 --- [           main] com.group9.demo.Group9Application        : No active profile set, falling back to default profi
les: default
2020-08-04 03:25:24.492  INFO 31231 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFERR
ED mode.
2020-08-04 03:25:24.579  INFO 31231 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 73ms. Fo
und 2 JPA repository interfaces.
2020-08-04 03:25:25.254  INFO 31231 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8181 (http)
2020-08-04 03:25:25.270  INFO 31231 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-08-04 03:25:25.270  INFO 31231 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.36]
2020-08-04 03:25:25.361  INFO 31231 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
```

After this text is displayed, open any browser and enter: [*http://localhost:8181/*](http://localhost:8181/)

CSV Output:

If the User would like to view a CSV file of the data generated, they can view it in the same project directory path, *src/main/java/com.group9.demo/*. The CSV file will contain all the customer records for the range of dates entered by the User.



	1	2	3	4	5	6	7	8	9
1	CUSTOMER ID	ENTRY DATE	ENTRY TIME	TIME IN STORE	EXIT TIME	CUSTOMER TYPE	AGE	WEATHER	HOLIDAY
2	1	2020-08-15	06:00:55	69	07:09:55	Regular	23	Normal Weather	Not a Holiday
3	2	2020-08-15	06:00:59	62	07:02:59	Regular	52	Normal Weather	Not a Holiday
4	3	2020-08-15	06:03:06	68	07:11:06	Senior	74	Normal Weather	Not a Holiday
5	4	2020-08-15	06:04:02	62	07:06:02	Regular	55	Normal Weather	Not a Holiday
6	5	2020-08-15	06:04:25	67	07:11:25	Regular	18	Normal Weather	Not a Holiday
7	6	2020-08-15	06:06:11	7	06:13:11	Regular	61	Normal Weather	Not a Holiday
8	7	2020-08-15	06:07:13	11	06:18:13	Senior	73	Normal Weather	Not a Holiday
9	8	2020-08-15	06:07:41	62	07:09:41	Senior	70	Normal Weather	Not a Holiday
10	9	2020-08-15	06:08:42	69	07:17:42	Regular	43	Normal Weather	Not a Holiday
11	10	2020-08-15	06:13:03	8	06:21:03	Regular	53	Normal Weather	Not a Holiday

Maneuvering through the Web Interface:

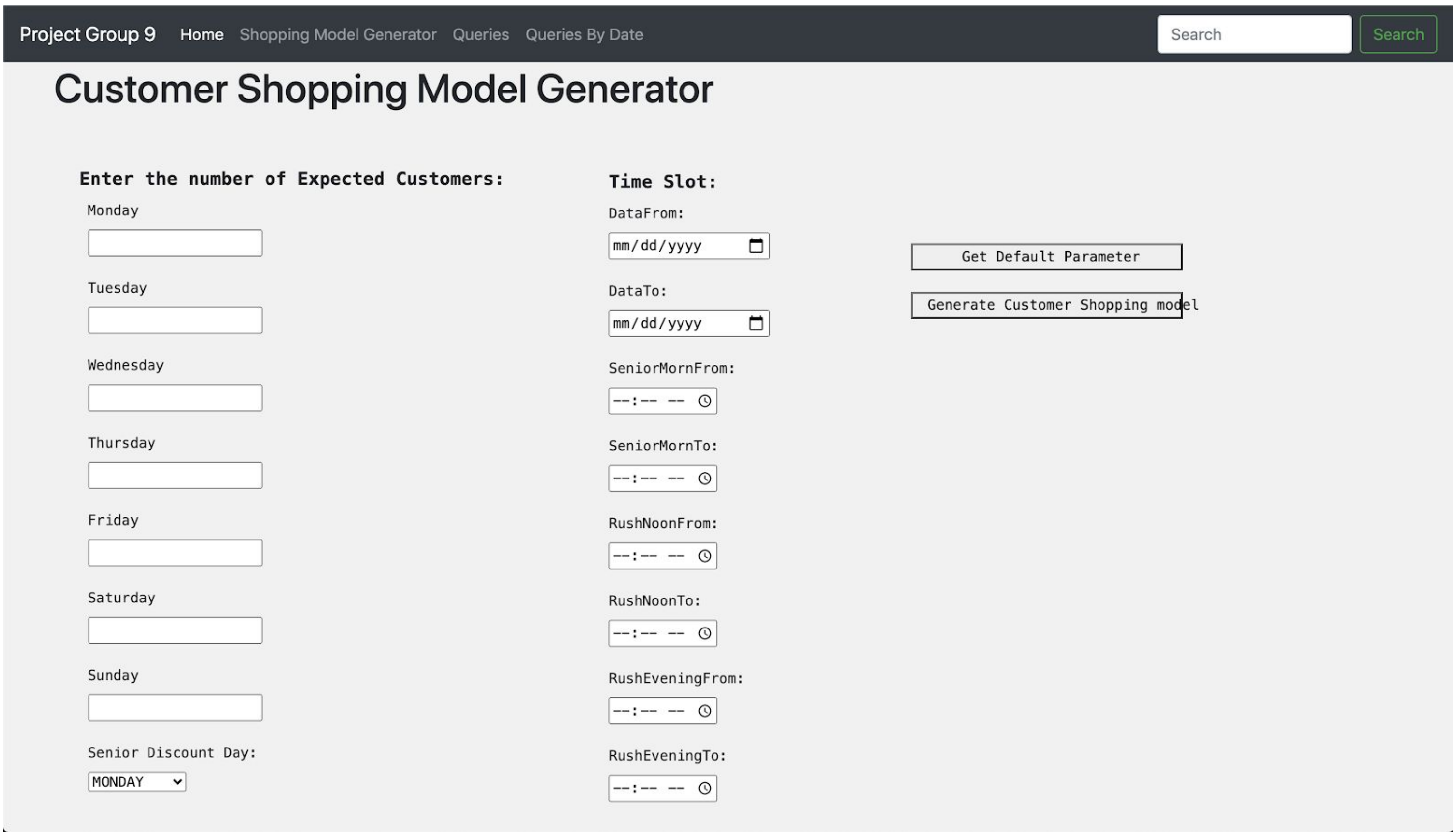
The web interface provides the following features that the User can make use of, which were promised in the final deliverable:

1. Customer Shopping Model Generator:

This generator can be found under the Shopping Model Generator tab in the Navigation Bar. It allows the User to modify parameters to fine-tune the customer data generated. These parameters include:

- Expected number of customers per day of the week
- Day of week selected for providing Senior Discounts
- Range of dates that the User wants to generate data for
- Time slots (from/to) for senior hours and the rush hours in afternoon and evening

The User can also set a default set of parameters which will automatically fill all the parameter fields with default data (as specified by the customer in the requirements stage).



Customer Shopping Model Generator

Enter the number of Expected Customers:

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

Sunday

Senior Discount Day: MONDAY

Time Slot:

DataFrom: mm/dd/yyyy

DataTo: mm/dd/yyyy

SeniorMornFrom: --:-- --

SeniorMornTo: --:-- --

RushNoonFrom: --:-- --

RushNoonTo: --:-- --

RushEveningFrom: --:-- --

RushEveningTo: --:-- --

Customer Shopping Model Generator Page

2. Query By Date Viewer:

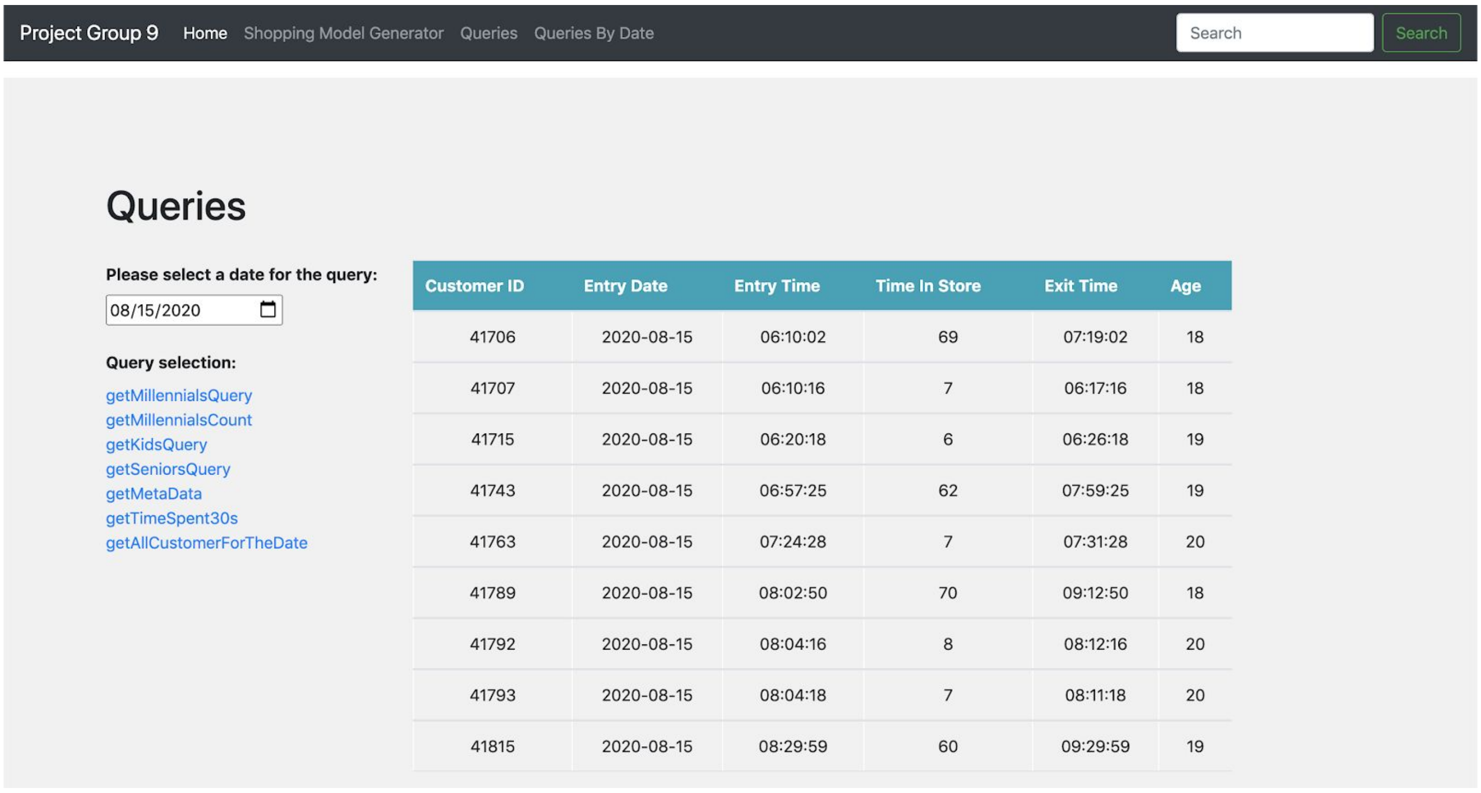
This page can be found under the Queries By Date tab in the Navigation tab. The user has the ability to perform various queries on the data generated by the generator. They can enter the date table they'd like to perform queries on, and use any of the following queries available in the interface:

- getMillennialsQuery
- getMillennialsCount
- getKidsQuery
- getSeniorsQuery
- getMetaData
- getTimeSpent30s
- getAllCustomerForTheDate

Another extra query that is available via Postman is:

- getCustomerById

The use of these queries will be described in detail in the following section.



Queries

Please select a date for the query:

08/15/2020

Query selection:

getMillennialsQuery
getMillennialsCount
getKidsQuery
getSeniorsQuery
getMetaData
getTimeSpent30s
getAllCustomerForTheDate

Customer ID	Entry Date	Entry Time	Time In Store	Exit Time	Age
41706	2020-08-15	06:10:02	69	07:19:02	18
41707	2020-08-15	06:10:16	7	06:17:16	18
41715	2020-08-15	06:20:18	6	06:26:18	19
41743	2020-08-15	06:57:25	62	07:59:25	19
41763	2020-08-15	07:24:28	7	07:31:28	20
41789	2020-08-15	08:02:50	70	09:12:50	18
41792	2020-08-15	08:04:16	8	08:12:16	20
41793	2020-08-15	08:04:18	7	08:11:18	20
41815	2020-08-15	08:29:59	60	09:29:59	19

getMillennialsQuery Results for August 15th, 2020

In case the User would like to perform the queries in their Postman environments, they can use the following URLs to perform the following query operations instead.

Available URLs:

To begin with, the user will have to edit the JSON (JavaScript Object Notation) file provided, in order to generate the data that they require. The available URLs that can be tested through Postman allow the user to view the default set of parameters that can be used (1), create the database of customer records for the range of dates specified in the JSON (2), get all the records of a particular date (3) and get a single customer record based on customer ID, on the specified date (4).

We have also specified two example URLs that perform SQL queries on the data retrieved on the specified date (5, 6)

1. GETDefaultParameters:

   URL Form: http://localhost:8080/defaultParameters

   This API call is used to display the set of parameters, in JSON format, that will be used by default. Whatever set is displayed here, will be used for creating the dataset, if the user does not change any parameters in the input JSON in the POST call below.

2. POSTDatabaseCreation:

   URL Form: http://localhost:8080/createData

   This API call is used to create the entire database from scratch. The database will be created based on the JSON output that the user must provide in the *Body* à*Raw* space in Postman, as input. Based on the from and to date range given, the dataset will be formed accordingly.

3. GETAllRecordsofaDateTable:

   URL Form: http://localhost:8080/2020_JANUARY_29

   This API call is for retrieving all the records of a particular date table, provided that it exists in the database.

4. GETSingleRecordfromADateTable:

   URL Form: http://localhost:8080/2020_JANUARY_29/{id}

   This API call is for retrieving a single record from the specified date table. The record is retrieved based on the customer ID given as a parameter in the URL. If the customer ID does not exist, a 404 Not Found error will be thrown.

5. GETRecordsofPeoplebetweenAges18-25:

   URL Form: http://localhost:8080/getMillennialsQuery/2020_JANUARY_29

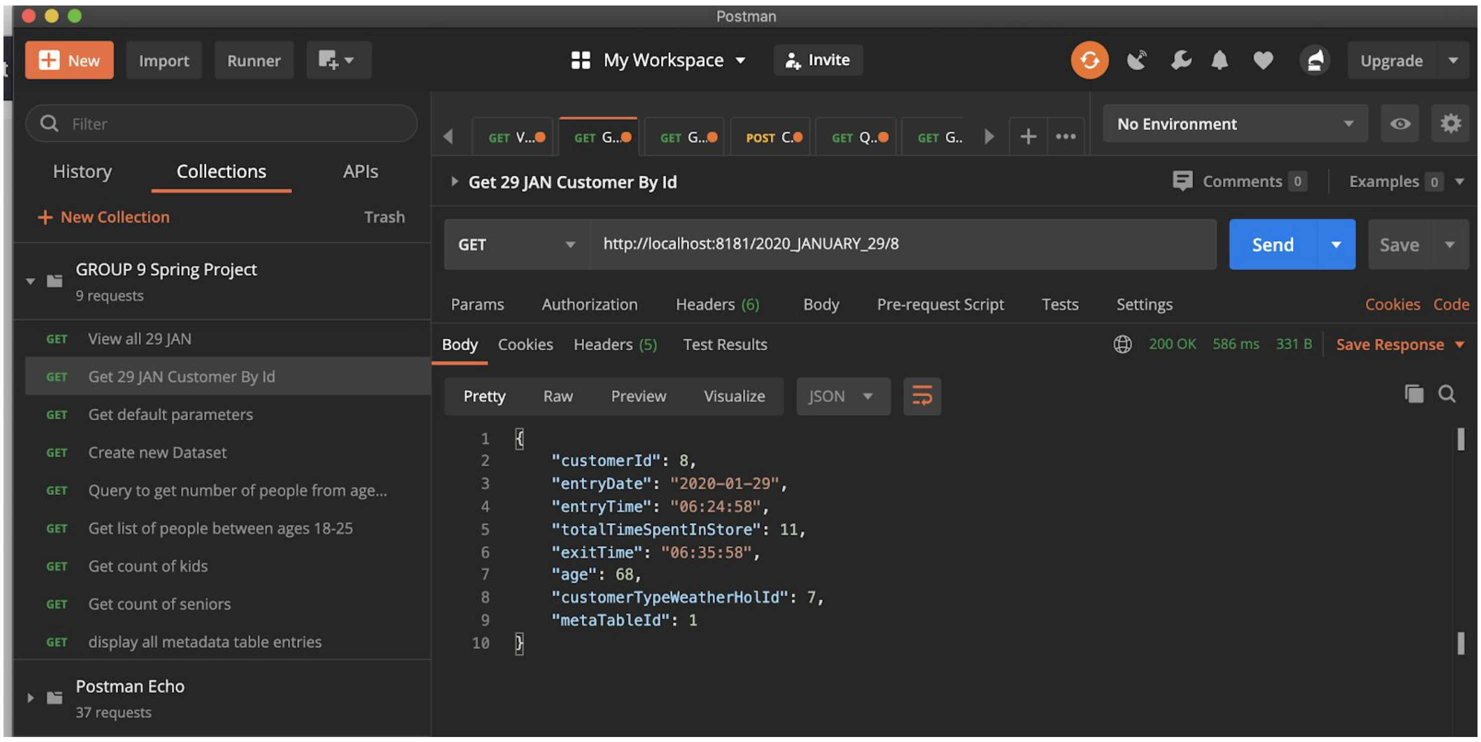
   This API call performs a SQL type query on the date table specified, in order to get some useful insights. For this particular call, the user can retrieve customer records where the customers are between the ages 18 to 25.

6. GETCountofaboveAge50:

   URL Form: http://localhost:8080/getCountQuery/2020_JANUARY_29

For this particular call, the user can retrieve the number of customer records where the customer is greater than or equal to the age of 50.

The results for all these queries, will be shown as a resultant set of JSON objects. In case a table or value doesn't exist, a user understandable JSON error message will be thrown as an exception.



Get 29 JAN Customer By Id

GET http://localhost:8181/2020_JANUARY_29/8

GROUP 9 Spring Project
9 requests
- GET View all 29 JAN
- GET Get 29 JAN Customer By Id
- GET Get default parameters
- GET Create new Dataset
- GET Query to get number of people from age...
- GET Get list of people between ages 18-25
- GET Get count of kids
- GET Get count of seniors
- GET display all metadata table entries

Postman Echo
37 requests

```
{
    "customerId": 8,
    "entryDate": "2020-01-29",
    "entryTime": "06:24:58",
    "totalTimeSpentInStore": 11,
    "exitTime": "06:35:58",
    "age": 68,
    "customerTypeWeatherHolId": 7,
    "metaTableId": 1
}
```

getCustomerById for January 29th, 2020

Issues to Look Out For:

- While working on the Queries By Date Tab, the results will display slightly late by a couple seconds based on the size of the data requested.
- The Search Bar may not function. Use other tabs on the navigation bar directly.
- Queries Tab is non-functional at this time. Please use the Queries By Date tab to perform database queries.
- If faced with connection errors, make sure MySQL server is turned on with the username and password mentioned in the application.properties file.