SHIPT - API

Overview:

This application was built /implemented using **Java (Spring Boot, JPA)** and **MySQL**. I have used **Eclipse** as my development tool.

Assumptions:

1. A product belongs to a particular category
2. Inventory is infinite.
3. Given a date range and interval(day/Month/Week), the query/api results the count of a each product per interval within the range.

   Given more time I would complete all the basic API's with custom validation and exceptions.

Setup guide:

1. Download the zip file and extract the contents .

2. Import the project into your workspace.

3. Install MySQL (if required) https://www.youtube.com/watch?v=UcpHkYfWarM&t=837s

4. Open terminal and type ***mysql –u root –p***

5. Enter your root password(same password while installing mysql)

6. Create a database using ***CREATE database <db_name>***

7. To use the database ***USE <db_name>***

8. Spring boot Application by default runs on port 8080. To change the port navigate to src-> main -> resources -> application.properties and enter **server.port=<port_number> (eg: server.port = 8091)**

9. Update the following in your application.properties file

   `spring.`datasource.url = jdbc:mysql://localhost:3306**/<db_name>?**useSSL=false
   spring.datasource.username = root
   spring.datasource.password = **<your root password>**
   spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
   spring.jpa.hibernate.ddl-auto = update

10. Update the project to down load all the dependencies
    Right click on project -> Maven -> update project

11. Run the project

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.1.RELEASE)
Tomcat started on port(s): 8091 (http) with context path '
```

12. Now entities/tables would be created in your db.

```
mysql> show tables;
+------------------+
| Tables_in_shipt  |
+------------------+
| categories       |
| category_products |
| customer_orders  |
| customers        |
| orderdetails     |
| orders           |
| orders_products  |
| products         |
+------------------+
```

13. Insert data by pasting the Insert queries from **data.sql** present in **src /main/resources**

Url end-points:

1. An API end point that accepts a date range and a day, week, or month and returns a breakdown of products sold by quantity per day/week/month

/order/order-range/{startDate}/{endDate}/Month
Eg: http://localhost:8091/order/order-range/2019-01-01/2019-01-20/Month

Output:


```
[{"month":1,"year":2019,"quantity":3,"productName":"Whole Milk"},
{"month":1,"year":2019,"quantity":10,"productName":"Carrot"},
{"month":1,"year":2019,"quantity":8,"productName":"Organic Eggs"}]
```


Similarly for Week / Day

/order/order-range/{startDate}/{endDate}/Week
Eg: http://localhost:8091/order/order-range/2019-01-01/2019-01-20/Week

Output:


```
[{"year":2019,"quantity":3,"productName":"Whole Milk","week":0},
{"year":2019,"quantity":10,"productName":"Carrot","week":0},
{"year":2019,"quantity":5,"productName":"Organic Eggs","week":0},
{"year":2019,"quantity":3,"productName":"Organic Eggs","week":2}]
```

/order/order-range/{startDate}/{endDate}/Day
Eg: http://localhost:8091/order/order-range/2019-01-01/2019-01-20/Day

Output:


```
[{"day":1,"month":1,"year":2019,"quantity":3,"productName":"Whole Milk"},
{"day":1,"month":1,"year":2019,"quantity":10,"productName":"Carrot"},
{"day":1,"month":1,"year":2019,"quantity":5,"productName":"Organic Eggs"},
{"day":17,"month":1,"year":2019,"quantity":3,"productName":"Organic Eggs"}]
```


Note: format for date should be yyyy-mm-dd

2. An API end point that returns the orders for a customer.

/customer-api/customerOrders/{email}

Eg: http://localhost:8091/customer-api/customerOrders/thejussingh.13@gmail.com

Output:


```
{
"email":"thejussingh.13@gmail.com",
"fname":"Thejus Singh",
"lname":"Jagadish",
"orders":[{
              "id":2,
              "orderdate":"2019-01-17T08:00:00.000+0000",
              "weeknum":3, day":"Thursday",month":"January",
              "status":"DELIVERED",
              "orders":[
                     {"id":4,
                     "product":{"id":4,
                            "name":"Organic Eggs",
                            "price":10.0},
                     "quantity":3.0}]
              {"id":1,
              "orderdate":"2019-01-01T08:00:00.000+0000",
              "weeknum":1, "day":"Tuesday",month":"January",
              "status":"DELIVERED",
              "orders":[
                     {"id":3,
                     "product":{"id":4, name":"Organic Eggs","price":10.0},
                     "quantity":5.0},
                     {"id":1,
                     "product":{"id":3, "name":"Whole Milk","price":5.99},
                     "quantity":3.0},
                     {"id":2,
                     "product":{"id":1, "name":"Carrot","price":2.99},
                     "quantity":10.0}]},
              {"id":3,
              "orderdate":"2019-02-24T08:00:00.000+0000",
              "weeknum":9, day":"Sunday",month":"February",
              "status":"ON THE WAY",
              "orders":[
                     {"id":6,
                     "product":{"id":2, "name":"Cherry Tomato","price":3.99},
                     "quantity":4.0},
                     {"id":7,
                     "product":{"id":1,"name":"Carrot","price":2.99},
                     "quantity":3.0},
                     {"id":5,
```

```
                    "product":{"id":4,"name":"Organic Eggs","price":10.0},
                    "quantity":2.0}]
                }]
    }
```


Queries:

1. Write a SQL query to return the results as display below:

Example

customer_id	customer_first_name	category_id	category_name	number_purchased
1	John	1	Bouquets	15

Solution 1: Gets customer information, category information and number of products purchased per category by each customer.

```
SELECT customers.email, customers.fname AS first_name, customers.lname AS
 last_name, categories.id, categories.name AS category_name,
 SUM(orderdetails.quantity) AS quantity FROM customers, orders, customer_orders,
 orderdetails, orders_products, products, categories, category_products WHERE
 customers.email = customer_orders.email AND  customer_orders.orders_id =
 orders.id AND orders.id = orders_products.orders_id AND
 orders_products.orderdetails_id = orderdetails.id AND orderdetails.product_id =
 products.id AND categories.id = category_products.category_id AND
 category_products.product_id = products.id GROUP BY customers.email,
 categories.id;
```

```
+--------------------------+--------------+----------+---+--------------+---------+
| email                    | first_name   |last_name |id | product_name |quantity |
+--------------------------+--------------+----------+---+--------------+---------+
| thejussingh.13@gmail.com |Thejus Singh  | Jagadish | 1 | Dairy Products| 13     |
| thejussingh.13@gmail.com | Thejus Singh | Jagadish | 2 | Vegetables    | 17     |
+--------------------------+--------------+----------+---+--------------+---------+
```

Solution 2: Gets customer information, product information and number of products purchased by each customer.

```
SELECT customers.email, customers.fname AS first_name, customers.lname AS
last_name, products.id, products.name AS product_name,
SUM(orderdetails.quantity) AS quantity FROM customers, orders, customer_orders,
orderdetails, orders_products, products WHERE customers.email =
customer_orders.email AND  customer_orders.orders_id = orders.id AND orders.id
= orders_products.orders_id AND orders_products.orderdetails_id =
orderdetails.id AND orderdetails.product_id = products.id GROUP BY
customers.email, products.id;
```

```
+--------------------------+--------------+-----------+---+--------------+---------+
| email                    | first_name   | last_name |id | product_name |quantity |
+--------------------------+--------------+-----------+---+--------------+---------+
| thejussingh.13@gmail.com | Thejus Singh | Jagadish  | 3 | Whole Milk   | 3       |
| thejussingh.13@gmail.com | Thejus Singh | Jagadish  | 1 | Carrot       | 13      |
| thejussingh.13@gmail.com | Thejus Singh | Jagadish  | 4 | Organic Eggs | 10      |
| thejussingh.13@gmail.com | Thejus Singh | Jagadish  | 2 | Cherry Tomato| 4       |
+--------------------------+--------------+-----------+---+--------------+---------+
```

Solution 3: Gets customer information, category information and number of products purchased for a particular customer.

```
SELECT customers.email, customers.fname AS first_name, customers.lname AS
last_name, categories.id, categories.name AS category_name,
SUM(orderdetails.quantity) AS quantity
FROM customers, orders, customer_orders, orderdetails, orders_products,
products, categories, category_products
WHERE customers.email = customer_orders.email AND  customer_orders.orders_id =
orders.id AND orders.id = orders_products.orders_id AND
orders_products.orderdetails_id = orderdetails.id AND orderdetails.product_id =
products.id AND categories.id = category_products.category_id AND
category_products.product_id = products.id
AND customers.email = "thejussingh.13@gmail.com"
GROUP BY customers.email, categories.id;
```

Note: Queries can be found in src/main/resources/query.sql

Other endPoints:

1. Categories

   Get all categories(GET):

   `/category-api/categories`

   Get category by id(GET):

   `/category-api/category/{id}`

   Creat a new category(POST):

   `/category-api/category`

   Update a category or create a new category(PUT):

   `/category-api/category/{id}`

   Delete a category(DELETE):

   `/category-api/category/{id}`

2. Customer

   Get all customer information(GET):

   `/customer-api/customers`

3. Order

   Get all order related information(GET):

   `/customer-api/customers`

4. Product

   Get all product information(GET):

   `/product-api/products`

Additional questions:

1. We want to give customers the ability to create lists of products for one-click ordering of bulk items. How would you design the tables, what are the pros and cons of your approach?

   Soln: I would have an additional table called order_List which will map to both customers as well as products to this table.

   Pros: Accessing data is simpler.

   Cons: order_List would have many records, one per each product for a particular customer. If the transaction fails then we need to rollback all the updates in that table.

2. If Shipt knew exact inventory of stores, and when facing a high traffic and limited supply of particular item, how do you distribute the inventory among customers checking out?

   Soln: I would use a queue data structure. Every customer can add the product to the cart but at the time of check out, I would check the database again to check if the order is satisfied and deducted count of those purchased items.