Batch #13 / Front-End Class

Midterm

2021.05.07 | 9:00 - 18:00

Midterm Exam (Back-End)

We need a Dashboard!

Today, you will be a level-1 data scientist. Your job is to build a dashboard to monitor our sales condition so that our business team and sales team can make better decisions based upon it.

Note:

1. You **can't** discuss with others.
2. You **can** search anything you need online.
3. You **can** ask me for clarification if you still can not understand some parts of the requirements.
4. Build this dashboard on **your own STYLiSH project**. Take a look at "How to Hand In" at the bottom.

Sample Page

Dashboard

Look at this example carefully! Your final result should look like it.

Note:

1. The product id, size, color, qty and price data you get may be different.
2. I will provide you with the data and a front-end library to help you build those charts, you don't need to start from scratch.

How to do it

1. Get and Store Order Data *(20 points)*

Write a script to call the following API and store the data into your database.

Batch #13 / Front-End Class

Midterm

2021.05.07 | 9:00 - 18:00

API: http://13.113.12.180:1234/api/1.0/order/data

you will get a bunch of order data from this API, save these orders into your database according to your table design. You can redesign your database depending on your needs, but make sure your original checkout API can still work.

You may get different values each time you call it, but you only need to save the data **once**, and then, you will use this data from now on.

You **don't** need to force the product id, color and size to be one of the values which you have inserted in other related tables before. If you have any foreign key constraint or null constraint which makes you unable to insert order data, you can remove it for now or add some dummy value.

The data format is like the Order CheckOut API you have seen before without unused keys.

Detail information of the data:

1. 5000 records of orders
2. Details of each key:

- `total`: means the total price of one order (sum of products' `price` * `qty` in the `list` key)
- `id`: product id (random integer number from 1 to 10)
- `color`: product color (random)
- `size`: product size (random)
- `qty`: product quantity (random integer number from 1 to 10)
- `price`: product price (random integer number from 500 to 2000)

2. Build the Dashboard *(70 points)*

Get data from **your database** and finish the following four requirements.

You can use Plot.ly to help you draw those interactive charts. Copy and paste the 3 lines under `plotly.js CDN` part to import the library.

1. Show the sum of `total` value in all orders. It can let us know our total revenue by now. *(15 points)*
2. Draw a Pie Chart which shows the percentage of sold products' quantity divided by color. It can let us know which color customers like the most.

Batch #13 / Front-End Class

Midterm

2021.05.07 | 9:00 - 18:00

(**Note:** The painted color of the Pie Chart need to be the same as its `name`, you can use the color `code` to do it) *(15 points)*

3. Draw a Histogram which shows the quantity of sold products in each price range. It can let us know products in which price range sold the best. (**Note:** same product may still have different prices in different orders, count them as separate no matter they are the same or different) *(20 points)*
4. Draw a Stacked Bar Chart which shows the sold quantity of the top five sold products in each size. Please sort by the total sold quantity from high to low. It can let us know which 5 products sold the best, and in each of these 5 products, which size sold the best. (**Note:** You don't need to take care about product name, use product id is good enough) *(20 points)*

You will need to use Title Setting to set the proper axis title.

Example:

If you are still confused about how to count the number, you can look at the example below:

If you get data like below which contains two orders:

```
[
    {
        total: 510
        list: [
            {id:1, size: 'M', color: {...}, qty: 1, price: 510},
        ]
    },
    {
        total: 7775
        list: [
            {id:1, size: 'S', color: {...}, qty: 3, price: 500},
            {id:1, size: 'L', color: {...}, qty: 5, price: 520},
            {id:2, size: 'S', color: {...}, qty: 7, price: 525},
        ]
    }
]
```

1. We will see that there are 4 products (qty 1 from first order + qty 3 from second order) sold at price range 500~519, and 12 products (qty 5+7 from second order) sold at price range 520~539.

---

2. We will count product id 1 has been sold 9 times (qty 1 from first order and qty 3+5 from second order) and product id 2 has been sold 7 times. So product 1 is the top one sold product.
3. For product 1, it is sold 3 times in size ‘S’, 1 time in size ‘M’ and 5 times in size ‘L’.

3. Auto-Refresh your Dashboard *(10 points)*

Today is the Shopping Festival! We want to see the newest sales status updated on the dashboard all day.

When a new order is created from your original website, the dashboard needs to update the numbers without refreshing the page by hand. Use Socket.io to refresh your page automatically.

Note: I didn’t implement this feature in the sample page. If you finish this, please mention it in the README file. I will test it by creating a new order from your original website, so make sure your checkout process still works.

4. Performance Tuning and Analysis (Advanced Optional) *(10 points)*

Check the following checkpoints and make their performance as good as possible. You can redesign your database but you **don’t** need to consider vertical or horizontal scale up.

1. How much time does your server spend to aggregate the data and render the page?
2. How much additional memory your server (or browser) uses to aggregate the data?
3. How much data do you transfer through the network? (Be careful about browser cache, you should measure it without browser cache)

Note: Describe **why** and **what** you did to improve the performance and **how** you measure these numbers in the README file. Since the time and memory usage may be a little different each time you load your page, you should be careful about how to measure these numbers.

Batch #13 / Front-End Class

Midterm

2021.05.07 | 9:00 - 18:00

How to Hand in

- Like what we did in STYLiSH project

1. Open a new branch called `midterm` and make change on it.
2. Deploy your new change on your own EC2 machine (t2.micro) after finishing.
3. Send me a pull request to branch `<your_name>_develop`.
4. Write your dashboard link in the bottom of README file

Note:

- Your branch name should be `midterm`.
- Your dashboard link should be: `https://[HOST_NAME]/admin/dashboard.html`.
- Make sure I can see the page without login.
- Do **not** upload your database with all the order data to GitHub. You can upload the schema only and generate the data again on EC2 machine, or you can upload your data via `scp` command from your laptop to EC2 machine.
- The auto check on GitHub will only check whether I can access the dashboard link or not, the final grade needs to wait for me to check it manually.

Deadline

- **6:00 pm**
- Please hand in before this time no matter how much you have done.