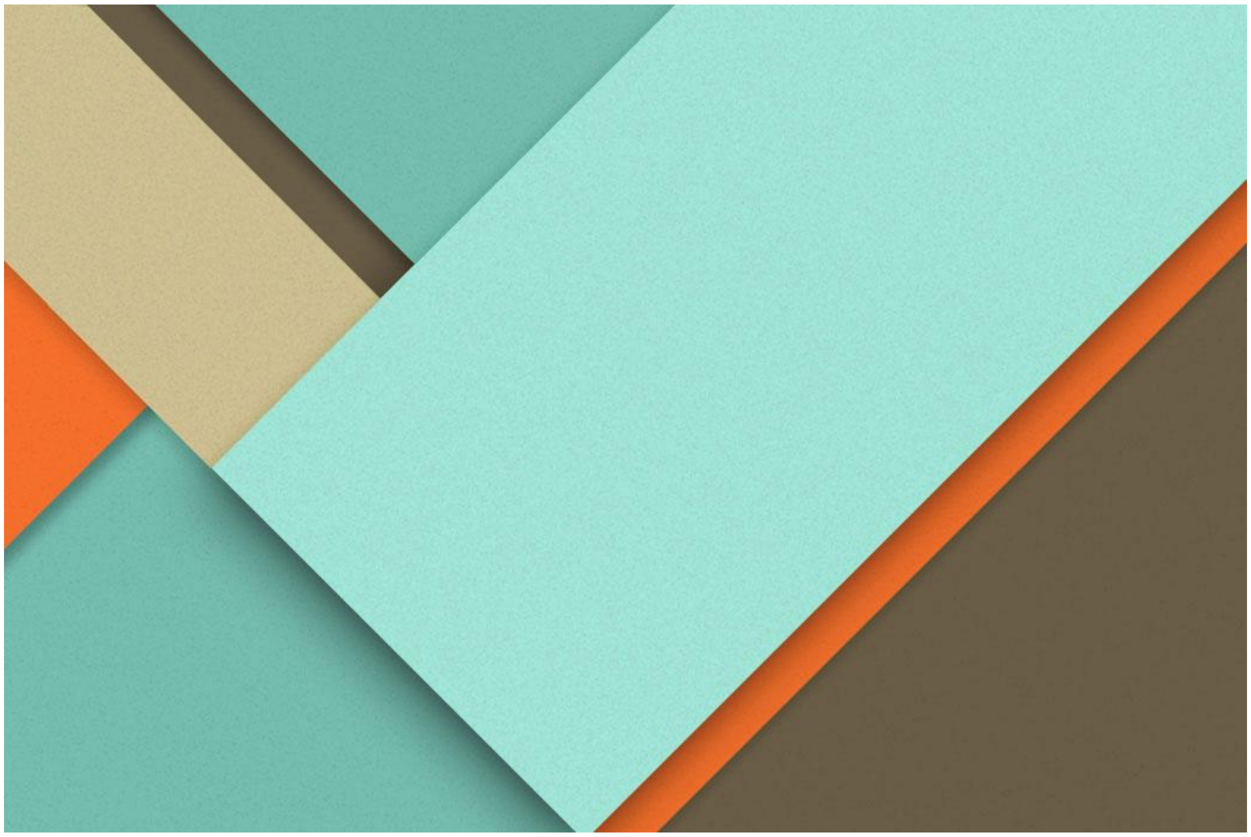


Cadence Take Home Coding Assignment

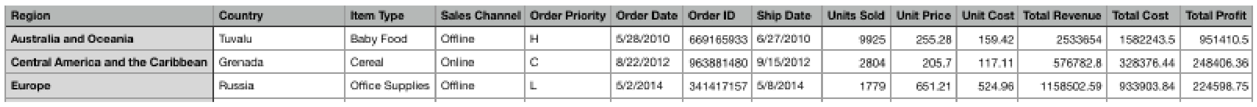
06.03.2021

—

Prachal Jitendrakumar Patel
San Jose State University
San Jose, CA

Overview

I was given a csv file which had the attributes as given below:



Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	9925	255.28	159.42	2533654	1582243.5	951410.5
Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	2804	205.7	117.11	576782.8	328376.44	248406.36
Europe	Russia	Office Supplies	Offline	L	5/2/2014	341417157	5/8/2014	1779	651.21	524.96	1158502.59	933903.84	224598.75

Goals

1. Use any SQL/NOSQL (MongoDB/Redis) database of your choice to load CSV to database
    a. If you are using SQL explain ER /normalization database design
2. Build a JSON endpoint for data to be consumed by scripts
    a. How do you validate json generated is a valid json data in your script ?
3. Build a CURD app to
    a. Accept new entry for all columns listed above from user using REST APIs
    b. Display Existing Data from database on a webpage
    c. Can you make each of columns sortable?
    d. Can you implement search on any data field ?

Specifications

Tech Stack Used

Database: MySQL RDS (AWS)

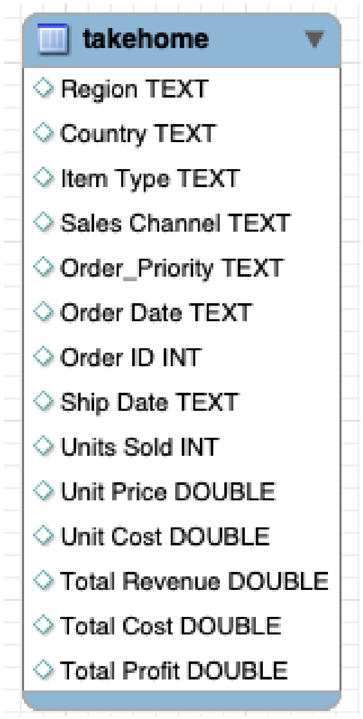
Frontend: React Js, HTML, CSS

Backend: Node Js, Express JS

Milestones

I. Database

The database schema is shown below.



```
takehome
Region TEXT
Country TEXT
Item Type TEXT
Sales Channel TEXT
Order_Priority TEXT
Order Date TEXT
Order ID INT
Ship Date TEXT
Units Sold INT
Unit Price DOUBLE
Unit Cost DOUBLE
Total Revenue DOUBLE
Total Cost DOUBLE
Total Profit DOUBLE
```

The original input table was in the above format. It is not a good idea to keep everything as

this because there is a lot of redundancy in the data. The schema must be designed in

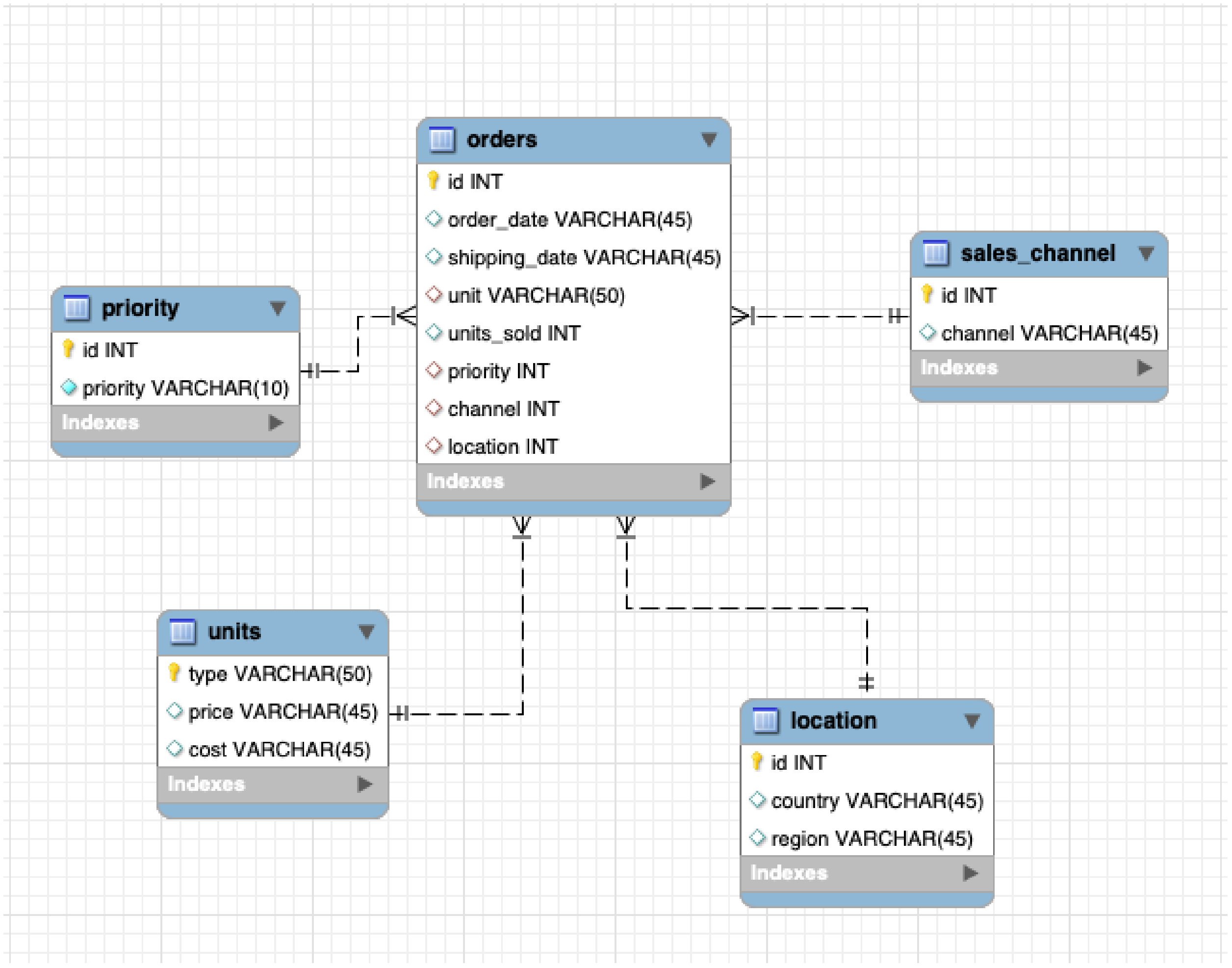
such a way that:

1. The data must be least redundant
2. The responsibilities are separated
3. The fields can be easily modified
4. The search is efficient

Thus the revised schema for this application is shown below.

There are 5 different tables in the schema for sales_channel, priority, orders, location and units. There is a primary key - foreign key relationship between the fields in the tables.

The main table in our schema is orders table which stores all the necessary information for a particular order. In the dashboard, the result is being sent after multiple joins and aggregations from all the tables.



1. There were 3 fields (revenue, total cost and profit) which could be derived from the cost price, selling price and number of units so I removed those fields in the new schema.
2. The individual tables are loosely coupled, easier to maintain and modify.
3. While gathering the details for the dropdown form, only the unique values will be taken from the separate tables which is a very lightweight operation as compared to searching the whole old bulky schema many times to find unique fields.
4. This will make administration tasks easy. If we need to add new options, then we can simply add them in the separate table without altering other tables. This will ensure separation of concern and responsibility segregation.

II. Backend routes

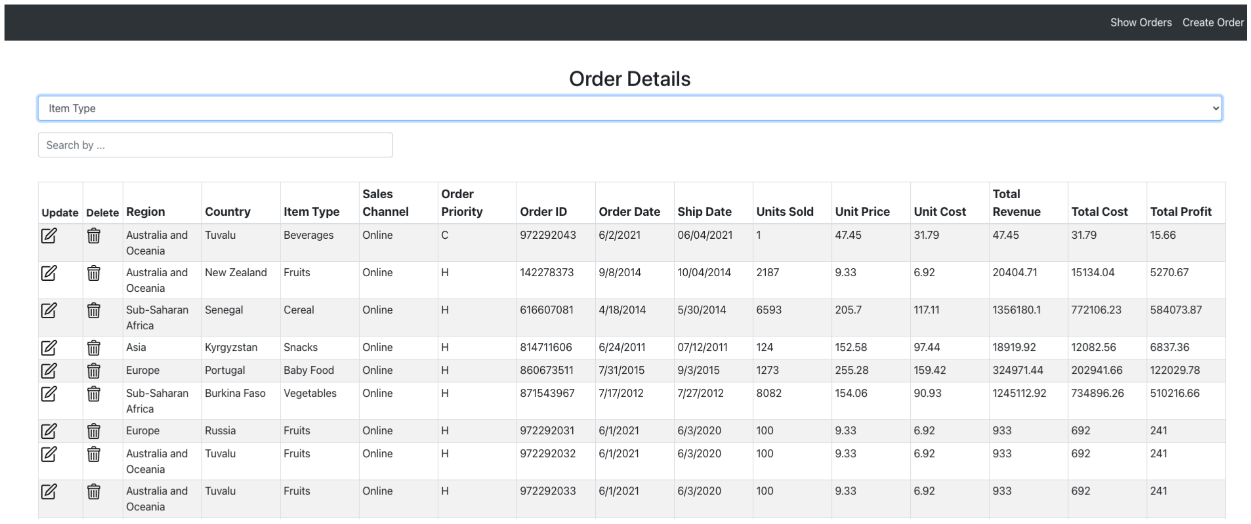
There are 4 main routes in the system that support CRUD operations.

1. GET
2. POST
3. UPDATE
4. DELETE

Apart from that, there are other utility operations that are taking care of the 4 different tables' unique fields for the create and update forms. Everytime the update/Create page is loaded, these utility get calls will be made to the database and it will save the result in the state.

III. Dashboard

The dashboard



Show Orders Create Order

Order Details

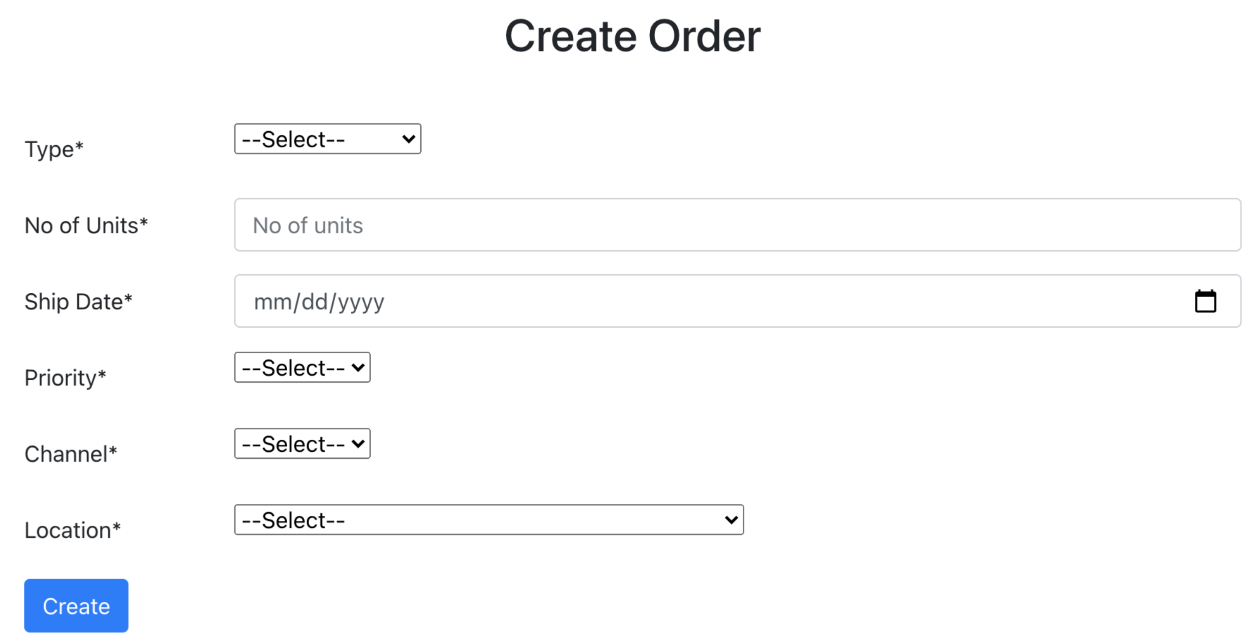
Item Type

Search by ...

Update	Delete	Region	Country	Item Type	Sales Channel	Order Priority	Order ID	Order Date	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
✎	🗑	Australia and Oceania	Tuvalu	Beverages	Online	C	972292043	6/2/2021	06/04/2021	1	47.45	31.79	47.45	31.79	15.66
✎	🗑	Australia and Oceania	New Zealand	Fruits	Online	H	142278373	9/8/2014	10/04/2014	2187	9.33	6.92	20404.71	15134.04	5270.67
✎	🗑	Sub-Saharan Africa	Senegal	Cereal	Online	H	616607081	4/18/2014	5/30/2014	6593	205.7	117.11	1356180.1	772106.23	584073.87
✎	🗑	Asia	Kyrgyzstan	Snacks	Online	H	814711606	6/24/2011	07/12/2011	124	152.58	97.44	18919.92	12082.56	6837.36
✎	🗑	Europe	Portugal	Baby Food	Online	H	860673511	7/31/2015	9/3/2015	1273	255.28	159.42	324971.44	202941.66	122029.78
✎	🗑	Sub-Saharan Africa	Burkina Faso	Vegetables	Online	H	871543967	7/17/2012	7/27/2012	8082	154.06	90.93	1245112.92	734896.26	510216.66
✎	🗑	Europe	Russia	Fruits	Online	H	972292031	6/1/2021	6/3/2020	100	9.33	6.92	933	692	241
✎	🗑	Australia and Oceania	Tuvalu	Fruits	Online	H	972292032	6/1/2021	6/3/2020	100	9.33	6.92	933	692	241
✎	🗑	Australia and Oceania	Tuvalu	Fruits	Online	H	972292033	6/1/2021	6/3/2020	100	9.33	6.92	933	692	241

IV. Add New Order

The user can add new orders to the table by filling a form and the value will be added to the database.



Create Order

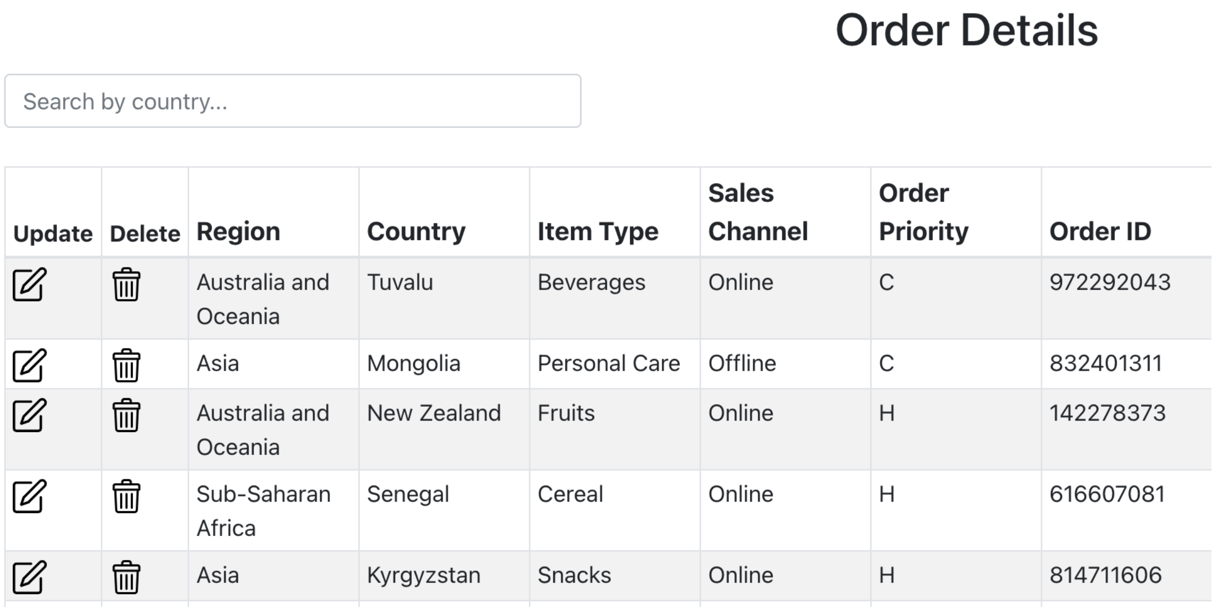
Type*	--Select--
No of Units*	No of units
Ship Date*	mm/dd/yyyy
Priority*	--Select--
Channel*	--Select--
Location*	--Select--

In this form, Most of the fields are dropdown select except the date and number of units. As per the database scheme, the form will fetch all the values from the database and the user can select all the values from dropdown. There are constraints in 2 fields.

1. No. of units: It should be more than 0.
2. Ship date: It should be after the order date.

V. Update Existing Order

To update a particular order, there is an edit icon at the beginning of each row under the update column. The order can be updated using the same form which is used for create but here the fields will be pre populated accordingly. The user can change the values from the table and update the order in the database.



Order Details

Search by country...

Update	Delete	Region	Country	Item Type	Sales Channel	Order Priority	Order ID
✎	🗑	Australia and Oceania	Tuvalu	Beverages	Online	C	972292043
✎	🗑	Asia	Mongolia	Personal Care	Offline	C	832401311
✎	🗑	Australia and Oceania	New Zealand	Fruits	Online	H	142278373
✎	🗑	Sub-Saharan Africa	Senegal	Cereal	Online	H	616607081
✎	🗑	Asia	Kyrgyzstan	Snacks	Online	H	814711606

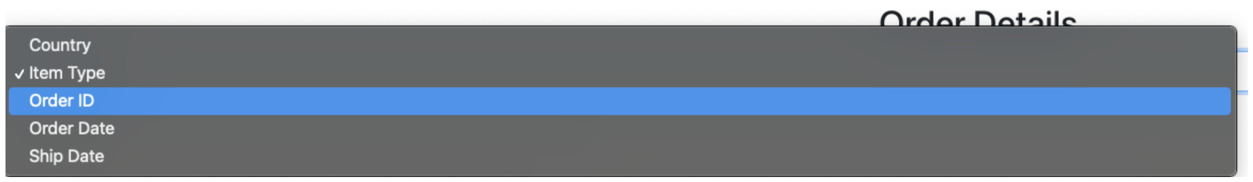
VI. Delete Existing Order

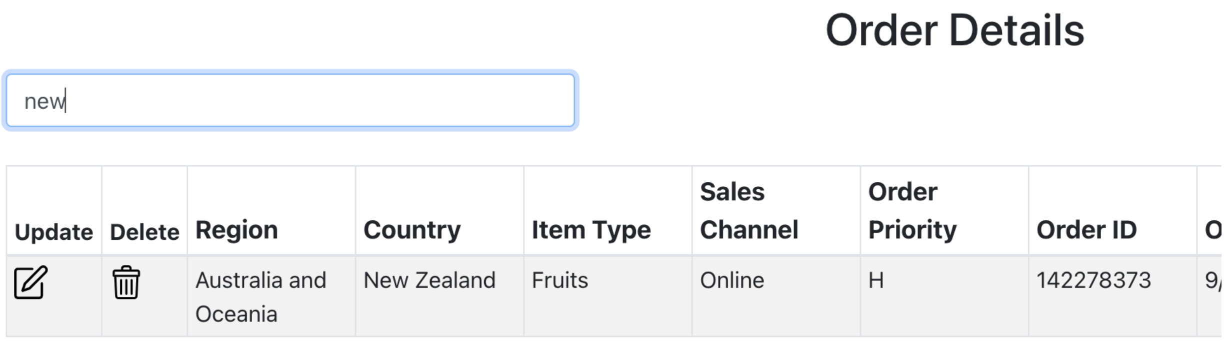
To delete the order, there is an icon like a trash bin at each row in the beginning. Once the user clicks on the delete icon, the row will be deleted from the table.

VII. Search Existing Order

For the search functionality, there is a search bar on the top of the dashboard and it will filter out the results as per the search criteria.

The search criteria is implemented for below fields





Order Details

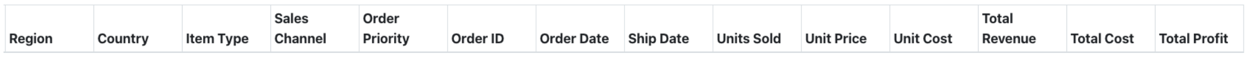
new

Update	Delete	Region	Country	Item Type	Sales Channel	Order Priority	Order ID	O
✎	🗑	Australia and Oceania	New Zealand	Fruits	Online	H	142278373	9/

VIII. Sorting individual columns

The dashboard has all the fields sortable as per the requirements.

The user can click on the Column headers and can sort the columns in ascending as well as descending order. The field header is shown below:



Region	Country	Item Type	Sales Channel	Order Priority	Order ID	Order Date	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit

There are 3 categories in sorting:

1. Strings
2. Decimals
3. Dates

Each and every category is handled individually in the frontend code.

IX. Validating JSON documents

For validating the json document, I am first fetching the data from one of the GET endpoints and saving it in a json file. Then the json file will be validated using python script.

The execution is like below:

```
Prachals-MacBook-Pro:marketplace prachal$ curl -X GET
http://localhost:3001/orders -o hello.json

  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload   Total   Spent    Left  Speed

100  7429  100  7429    0     0   122k      0 --:--:-- --:--:--
--:--:--  122k

Prachals-MacBook-Pro:marketplace prachal$ python -mjson.tool
hello.json > /dev/null

Expecting : delimiter: line 1 column 7419 (char 7418)

Prachals-MacBook-Pro:marketplace prachal$ python -mjson.tool
hello.json > /dev/null
```