CIS5200 Term Project Tutorial

Authors: Jose Ramirez; Jonathan B. Reyes;

Instructor: Jongwook Woo

Date: 12/15/2020

Lab Tutorial

Jose Ramirez (jramire3@calstatela.edu)
Jonathan B. Reyes (jreyes175@calstatela.edu)
12/15/2020

E-Commerce Behavior Analysis using HiveQL

Objectives

In this hands-on lab, you will learn how to:

- Download data from Kaggle.com and Google Drive
- Upload files into OBDCE
- Move files from Oracle to HDFS
- Create HiveQL Table to query data
- Analyze data using HiveQL
- Download data from Oracle to local computer
- Visualization of data generated from analysis

Platform Spec

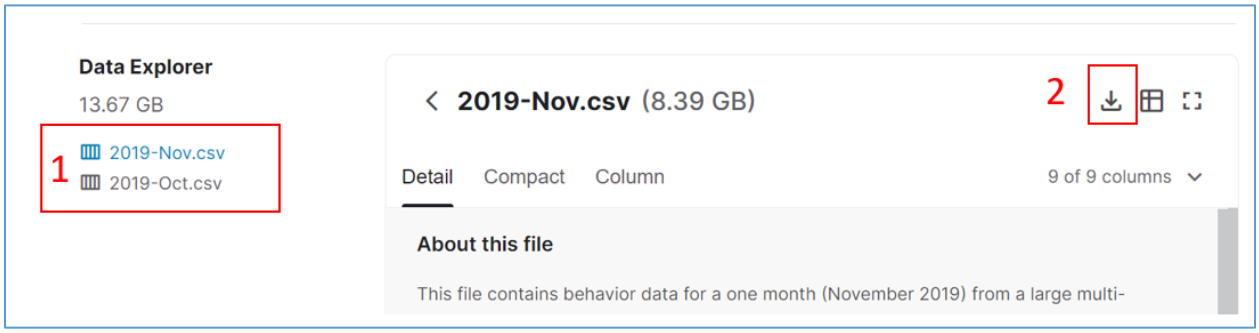
- Oracle BDCE
- CPU Speed: 2.195Ghz
- # of CPU cores: 12
- # of nodes: 3
- Total Memory Size: 180GB
- Storage: 957GB

Step 1: Download Data from Kaggle and Google Drive

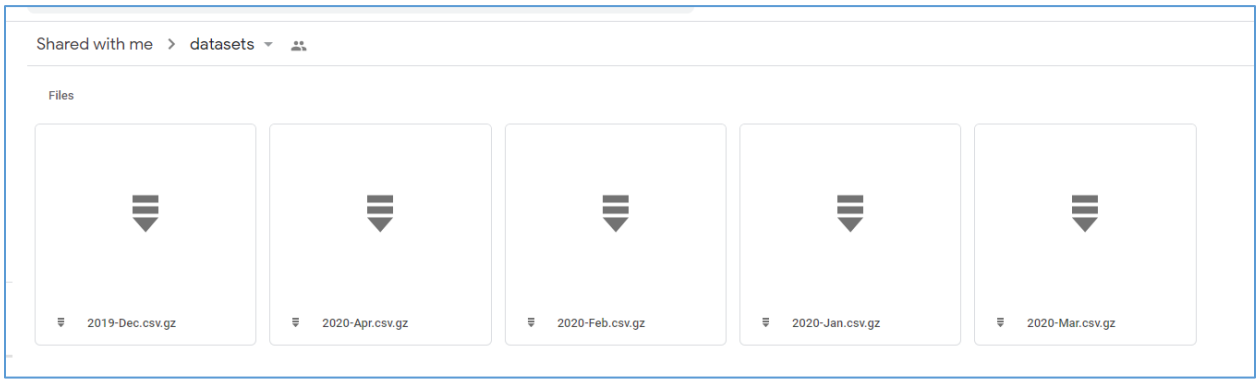
In this step, you will navigate to Kaggle.com and Google Drive to download ecommerce data needed to complete this lab.

1. Using a web browser, navigate to: https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store

2. You need to download both the *2019-Nov.csv* and *2019-Oct.csv* files. To download the files, scroll to where a preview of the data is provided. Then, click on one of the files listed under the Data Explorer heading followed by the download button, as demonstrated in the following image:



3. Then, using a web browser, navigate to: https://drive.google.com/drive/folders/1Nan8X33H8xrXS5XhCKZmSpClFTCJsSpE
4. In the Google Drive, you will see four files: 1) *2019-Dec.csv.gz*, 2) *2020-Apr.csv.gz*, 3) *2020-Feb.csv.gz*, 4) *2020-Jan.csv.gz*, and 5) *2020-Mar-csv.gz*. Download all files.



Step 2: Upload files to OBDCE and move to HDFS

Before data can be analyzed using Hive, you will need to first upload the files downloaded in Step 1 to OBDCE before moving them to HDFS.

The files you downloaded from Google Drive will be compressed, and the files from Kaggle will not be compressed. To speed up the upload process, you should also compress the files from Kaggle to reduce their file size.

NOTE: The size of the files may be too large to upload to OBDCE due to limited disk space. As a result, you may need to split the files into smaller sized files. To split a file, you will need to first un-compress it, and use software such as CSV Splitter to split the file. You can download the software from: https://download.cnet.com/CSV-Splitter/3000-2074_4-75910188.htmlnload . If you split the files, you will need ensure each file contains the field headings.

1. As mentioned, you will need to compress your files to speed up the upload process. To upload a file, open the command line and enter the following. You will need replace [username] with your username.

   ```
   scp 2019-Oct.zip [username]@129.150.69.91:/home/[username]
   ```

   You will be prompt to enter your password.

2. You will then need to SSH into OBDCE after the file is uploaded. Enter the following into your command line, ensuring you replace [username] with your username:

   ```
   ssh [username]@129.150.69.91
   ```

3. You will need to unzip it prior to moving to HDFS. To unzip a file, enter the following command:

   ```
   unzip 2019-Oct.zip
   ```

   You should now see a file in Oracle titled 2019-oct.csv.

4. Before moving this file to HDFS, we will create a directory where all uploaded files will be stored. This directory will be referenced in the HiveQL table schema you will create in Step 3. To create a directory in HDFS, enter the following command in your terminal and execute:

   ```
   hdfs dfs -mkdir Ecommerce_Behaviour
   ```

   You can confirm if the directory was successfully created by executing the following command:

   ```
   hdfs dfs -ls
   ```

5. You are now ready to move the 2019-Oct.csv file. To move the file into the directory you just created, enter the following:

   ```
   hdfs dfs -put 2019-Oct.csv Ecommerce_Behaviour/
   ```

   You can confirm the file was moved to the Ecommerce_Behaviour directory by running the following command:

```
hdfs dfs -ls Ecommerce_Behaviour/
```

You should see the 2019-Oct.csv file in the results.

6. To save space, you can delete both the 2019-Oct.csv and 2019-Oct.zip files stored in Oracle. Run the following to delete both files:

```
rm 2019-Oct*
```

You will need to repeat Step 2 for each file you need to upload. If you used CSV Splitter, you may have several files. All files downloaded from Kaggle and Google will need to be stored in the `Ecommerce_Behaviour/` directory.

Step 3: Create HiveQL Table to Query Data

In this step, you will connect to Hive using the beeline CLI. While in the beeline CLI, you will create a HiveQL table to query the data you moved to HDFS directory titled `Ecommerce_Behaviour/`. You will need to SSH to Oracle before continuing, i.e. `ssh [username]@129.150.69.91.`

1. Before running the beeline command, you will need to execute the following command to ensure you do not run into any permissions issues:

```
hdfs dfs -chmod -R o+w .
```

2. After connecting to Oracle, enter the following command to connect to Hive:

```
beeline
```

You will see the following output on your terminal:

```
-bash-4.1$ beeline
WARNING: Use "yarn jar" to launch YARN applications.
Beeline version 1.2.1000.2.4.2.0-258 by Apache Hive
```

3. Then, run the following:

```
!connect jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigdai-nov-
bdcsce-2:2181,bigdai-nov-bdcsce-
3:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hive
server2?tez.queue.name=interactive  bdcsce_admin
```

You will be prompt for a password. To bypass it, press the Enter key. You will see the following string on your screen, indicating you have successfully connected:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd>
```

4. Hive contains multiple databases. Use the following command to ensure your table is created in your database. Replace [database_name] with the name of your database:

```
USE [database_name];
```

5. You will now create the table needed to query the ecommerce data. Enter the following command and execute it, ensuring you replace [user_name] with your username:

```
CREATE EXTERNAL TABLE ecommerce_behavior (
   event_time   STRING,
   event_type    STRING,
   product_id    BIGINT,
   category_id   BIGINT,
   category_code STRING,
   brand         STRING,
   price         FLOAT,
   user_id       BIGINT,
   user_session  STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/[user_name]/Ecommerce_Behaviour/'
TBLPROPERTIES ('skip.header.line.count'='1');
```

To confirm the table was successfully created, run the following command:

```
SHOW TABLES;
```

You should see the following output:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> SHOW TABLES;
+----------------------+--+
|      tab_name        |
+----------------------+--+
| ecommerce_behavior   |
+----------------------+--+
```

Lastly, run the following query to ensure the data is correctly queried:

```
SELECT event_time, event_type FORM ecommerce_behavior LIMIT 5;
```

You will see the following output. The records returned may be different.

```
+--------------------------+-------------+--+
|        event_time        | event_type  |
```

```
+--------------------------+-------------+--+
| 2019-12-01 00:00:00 UTC  | view        |
| 2019-12-01 00:00:00 UTC  | view        |
| 2019-12-01 00:00:01 UTC  | view        |
| 2019-12-01 00:00:02 UTC  | purchase    |
| 2019-12-01 00:00:02 UTC  | view        |
+--------------------------+-------------+--+
```

Step 4: Query Data and Insert into Tables

In this step, you will query the ecommerce data using the table that was created in the previous step. The output of your queries will be inserted into new tables. The tables will have an associated directory where the output of your queries will be stored. In Step 5, you will download the output into your desktop.

1. You will create a table that contains the top 10 purchased categories contained in the `ecommerce_behavior` table. The following query will return the top 10 purchased categories and store the output in a new table called `top_purchased_categories`. Replace [user_name] with your username and execute:

```
CREATE TABLE top_purchased_categories
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/[user_name]/top_purchased_categories/'
AS
SELECT  REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
        category_type,
        COUNT(*) total_items
FROM ecommerce_behavior
WHERE event_type = 'purchase'
GROUP BY REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
ORDER BY total_items DESC
LIMIT 10;
```

You can run the following command to confirm the table was created:

```
SHOW TABLES;
```

You should see the following output on your terminal:

```
+----------------------------+--+
|          tab_name          |
+----------------------------+--+
| ecommerce_behavior         |
| top_purchased_categories   |
+----------------------------+--+
```

Then, execute the following query to ensure the top 10 purchased categories are stored in the `top_purchased_categories` table:

```
SELECT * FROM top_purchased_categories;
```

You should see the following output:

```
+-----------------------------------------+-------------------
--------------------+--+
| top_purchased_categories.category_type  |
top_purchased_categories.total_items  |
+-----------------------------------------+-------------------
--------------------+--+
| construction                            | 2427067
| appliances                              | 765358
| electronics                             | 505484
| apparel                                 | 434182
|                                         | 392360
| sport                                   | 313709
| furniture                               | 179235
| computers                               | 145376
| kids                                    | 83251
| auto                                    | 34837
+-----------------------------------------+-------------------
--------------------+--+
```

2. Now that you know that construction is the top purchased category, we will query the `ecommerce_behavior` table to identify the top 10 construction items. The results from the query will be stored in a new table called `top_purchased_construction`. Execute the following query:

```
CREATE TABLE top_purchased_construction
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/[user_name]/top_purchased_construction/'
AS
SELECT REGEXP_EXTRACT(category_code, '(\\w+[^.])$')
category_subtype,
      COUNT(*) total_items
FROM ecommerce_behavior
WHERE event_type = 'purchase' AND
REGEXP_EXTRACT(category_code, '^(\\w+[^.])') = 'construction'
GROUP BY REGEXP_EXTRACT(category_code, '(\\w+[^.])$')
ORDER BY total_items DESC
LIMIT 10;
```

To confirm your table was successfully created, execute the following SELECT statement:

```
SELECT * FROM top_purchased_construction;
```

You should see the following output:

```
+-----------------------------------------------+--------------
---------------------------+--+
| top_purchased_construction.category_subtype  |
top_purchased_construction.total_items  |
+-----------------------------------------------+--------------
---------------------------+--+
| light                                         | 2312912
| faucet                                        | 45106
| welding                                       | 22590
| drill                                         | 16719
| generator                                     | 13807
| saw                                           | 9933
| pump                                          | 3199
| painting                                      | 2185
| screw                                         | 385
| soldering                                     | 153
+-----------------------------------------------+--------------
---------------------------+--+
```

3. Lastly, we want to know during what hour of the day were most construction lights purchased. As we did in the previous steps, we will create another table containing a count of lights purchased by hour of day. The results will be based on 24 hours. Execute the following:

```
CREATE TABLE category_purchased_hour
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/[user_name]/category_purchased_hour/'
AS
SELECT SUBSTRING(event_time, 12,2) purchased_hour, COUNT(*)
total_purchased
FROM ecommerce_behavior
WHERE REGEXP_EXTRACT(category_code, '^(\\w+[^.])') =
'construction' AND
     REGEXP_EXTRACT(category_code, '(\\w+[^.])$') = 'light'
AND
     event_type ='purchase'
GROUP BY SUBSTRING(event_time, 12,2);
```

By now, you should have four tables at this point. Run the following statement:

```
SHOW TABLES;
```

You should see the following tables:

```
+-----------------------------+--+
|          tab_name           |
+-----------------------------+--+
| category_purchased_hour     |
```

```
| ecommerce_behavior           |
| top_purchased_categories     |
| top_purchased_construction   |
+------------------------------+--+
```

4. Lastly, we want to determine what the top purchased category was for each month in the data set, which includes the months of October 2019 through April 2020. To perform this analysis, you will create a table from the results of a union. The union is comprised of multiple select statements, each summarizing the top purchased category for each month in the dataset. Run the following:

```
CREATE TABLE top_monthly_category
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/[user_name]/top_monthly_category/'
AS
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
              MONTH(event_time) = '10'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items DESC LIMIT 1) AS u1
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
              MONTH(event_time) = '11'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items
     DESC LIMIT 1) AS u2
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
              MONTH(event_time) = '12'
     GROUP BY MONTH(event_time),
```

```
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items
     DESC LIMIT 1) AS u3
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
           MONTH(event_time) = '1'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items
     DESC LIMIT 1) AS u4
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
             MONTH(event_time) = '2'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items
     DESC LIMIT 1) AS u5
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
             MONTH(event_time) = '3'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
     ORDER BY total_items DESC LIMIT 1) AS u6
UNION
SELECT *
FROM (SELECT MONTH(event_time) AS calendar_month,
               REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
category_type,
               COUNT(*) total_items
     FROM ecommerce_behavior
     WHERE event_type = 'purchase' AND
             MONTH(event_time) = '4'
     GROUP BY MONTH(event_time),
                REGEXP_EXTRACT(category_code, '^(\\w+[^.])')
```

```
        ORDER BY total_items
        DESC LIMIT 1) AS u7;
```

At this point, you should have five tables. Run the following statement:

```
SHOW TABLES;
```

You should see the following output on your terminal:

```
+----------------------------+--+
|      tab_name              |
+----------------------------+--+
| category_purchased_hour    |
| ecommerce_behavior         |
| top_monthly_category       |
| top_purchased_categories   |
| top_purchased_construction |
+----------------------------+--+
```

Step 5: Download Data to Desktop

In this step, you will use the `scp` command to download the files that were generated when the following tables were created: `top_purchased_categories`, `top_purchased_construction`, `category_purchased_hour`, and `top_monthly_category`.

When the four mentioned tables were created, a location clause was specified. The location points to a directory where the file we will download is located. You will need to move a copy of those files from HDFS to OBDCE before you can download the files to your desktop. NOTE: The commands in this section are one line each.

1. Open another terminal (or exit the beeline CLI) and run the following command to copy the `top_purchase_categories` file to OBDCE. The command is one line:

   ```
   hdfs dfs -get
   /user/[user_name]/top_purchased_categories/000000_0
   top_category.csv
   ```

   You can confirm if the file was copied to Oracle by executing the following command:

   ```
   ls .
   ```

   You should see the `top_category.csv` file.

2. Now, we will repeat the same process for the `top_purchased_construction` file. Run the following and then perform a `ls` command to confirm the file was successfully copied:

   ```
   hdfs dfs -get
   /user/[user_name]/top_purchased_construction/000000_0
   top_construction.csv
   ```

3. The `category_purchased_hour` directory may contain several files. You will merge all the files and store them in a different directory to facilitate the move of this file. Run the following to merge all files in the `category_purchased_hour` directory and store the merged file in a new directory called `merged_data`:

```
hdfs dfs -cat /user/[user_name]/category_purchased_hour/* |
hdfs dfs -put -
/user/[user_name]/merged_data/purchased_hour.csv
```

You can run the following command to confirm the file was successfully merged and stored in the merged_data directory:

```
hdfs dfs -ls merged_data/
```

You should see the purchased_hour.csv file. Then, run the following command to copy the file to Oracle:

```
hdfs dfs -get /user/[username]/merged_data/purchased_hour.csv
```

4. Then, we will move the file located in the `top_monthly_category` directory. Run the following and then perform a `ls` command to confirm the file was successfully copied:

```
hdfs dfs -get /user/[user_name]/top_monthly_category /000000_0
top_monthly_category.csv
```

If you run `ls`, you should now have four files:

```
-bash-4.1$ ls
purchased_hour.csv  top_category.csv  top_construction.csv
top_monthly_category.csv
```

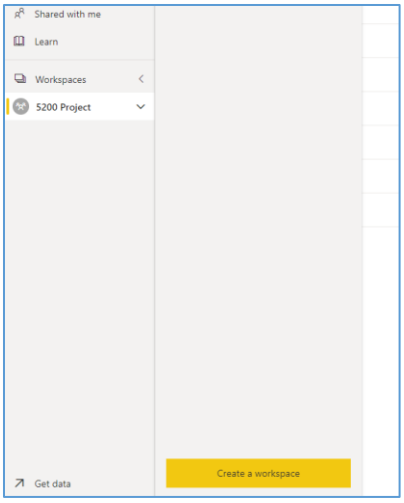
5. To download each file to your desktop, open a new terminal and run the following command. You will need to run the command four times, one for each file. Replace the file name in the command to the corresponding file name.

```
scp
[user_name]@129.150.69.91:/home/[user_name]/purchased_hour.csv
purchased_hour.csv
```

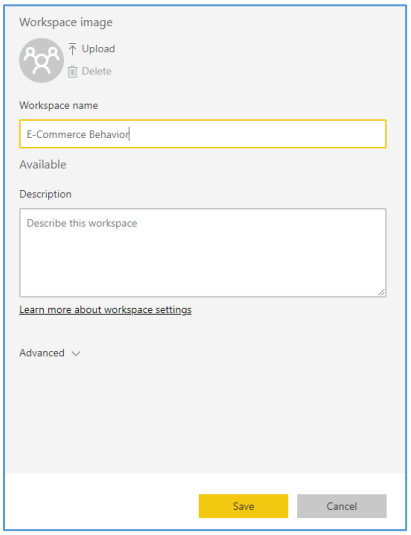
Step 6: Visualization

In this step, you will import the csv files you downloaded to your desktop in the previous stpe into Power BI. Using your Cal State LA account, login to https://app.powerbi.com.

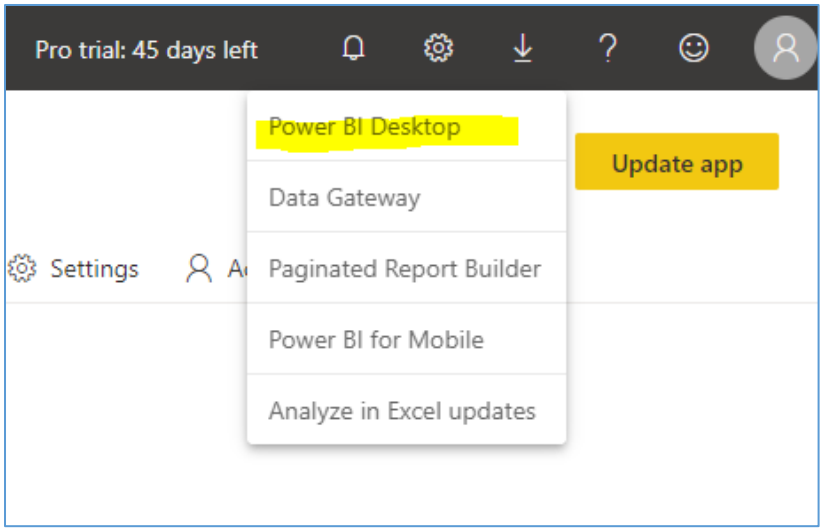
1. Once you are logged in, click on the Workspaces menu then Create a Workspace:



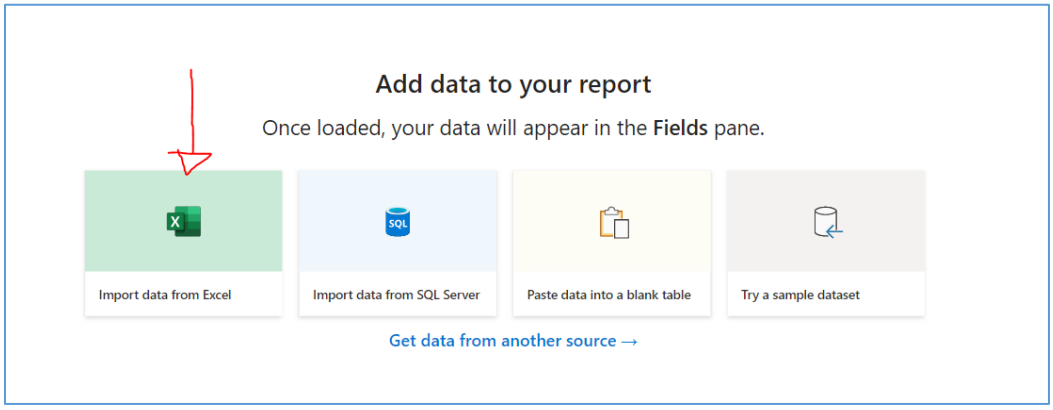
Give the workspace a descriptive name of your choice and click save:



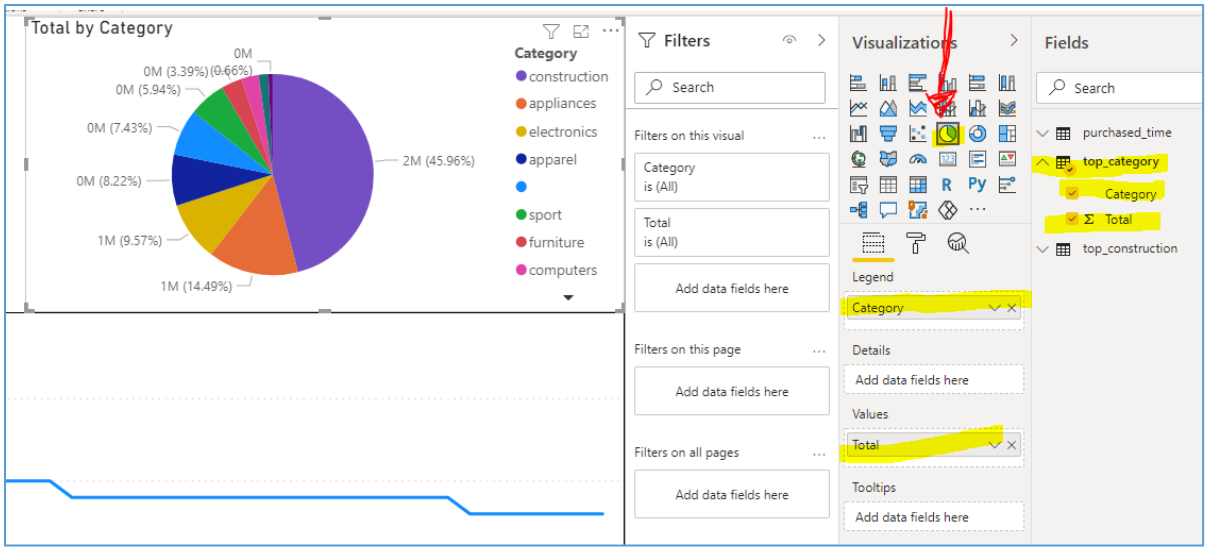
2. You will need to download the Power BI Desktop app to create a visual with all four files. On the top right, you will see a download button. Click on it to download the desktop app. After you install it, sign into the desktop app with your CSULA account.



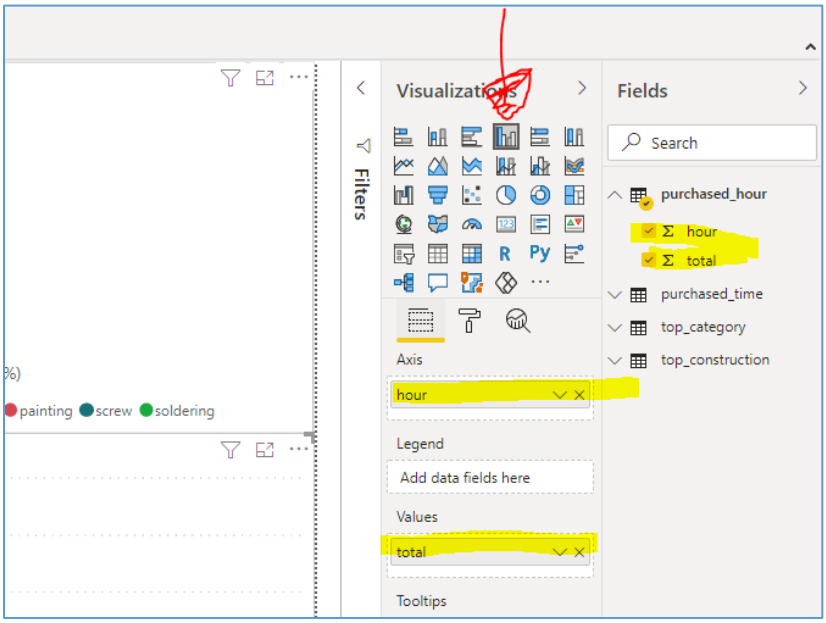
3. Then, select Excel as the data source and select the csv file. You will see the "Add data to your report wizard" when you first open the app. For additional files, you will need to click on the import option from the ribbon. You will need to upload the four csv files, i.e. `purchased_hour.csv`, `top_category.csv`, `top_construction.csv`, and `top_monthly_category.csv`.



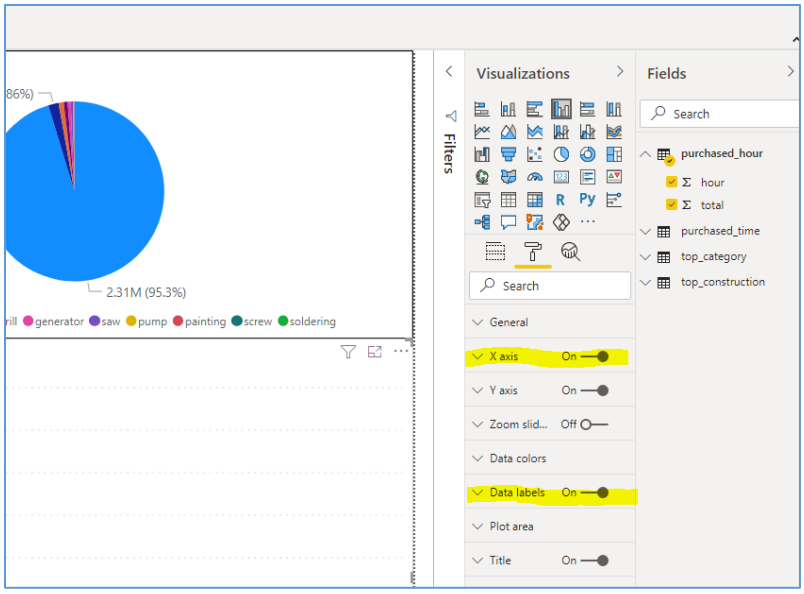
4. Then, under the visualizations pane located on the right side of the screen. Select the Pie Chart visual. Then, click on the two fields listed under the `top_category` table. You will find the fields under the Fields pane. Then, click and drag the Category (or Column1) field to the Legend field under the visualizations pane. Lastly, click and drag the Total (or Column2) field to the Values field. You will need to follow this step for the `top_construction` table too; for that table, Column1 can be renamed to Item and Column2 to Total.



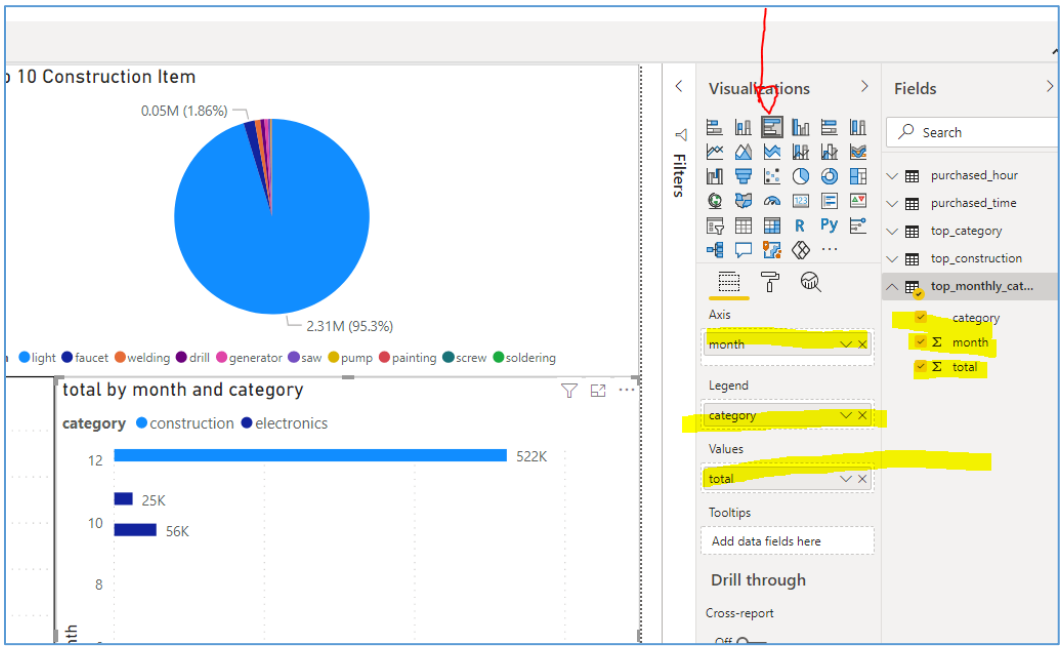
5. Click on the clustered column bar graph visualization. Then, add the fields listed under the `purchased_hours` table. You can click on the ellipses that appears to the right of the field name when you hover your cursor over it and select Rename form the pop-up menu. Column1 can be renamed to “hour” and Column2 can be renamed to “total. The Hour field goes in the Axis field, and the Total field goes in the Values field:



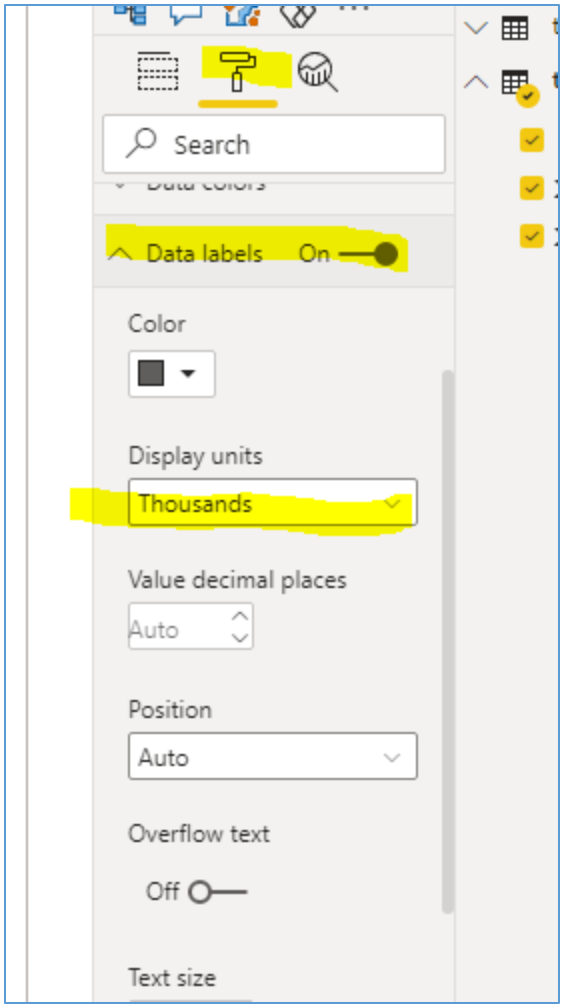
While the clustered column bar graph is selected, click on the formatting menu. Then, turn on the data labels. Expand the X-Axis section and switch the type to Categorical.



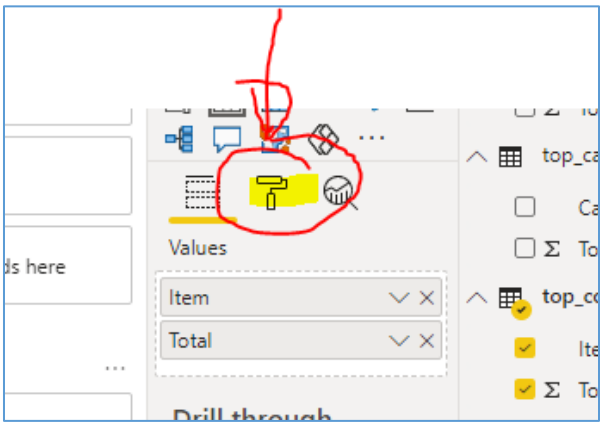
6. Next, on the fields pane, you will see a table title “top_monthly_category.” This table contains three columns. Rename column 1 to “month”, column 2 to “category”, and column 3 to “total.” Then, select the clustered bar chart visual. Click and drag the month field to the Axis section, the category field to the Legend section, and the total field to the Values section.



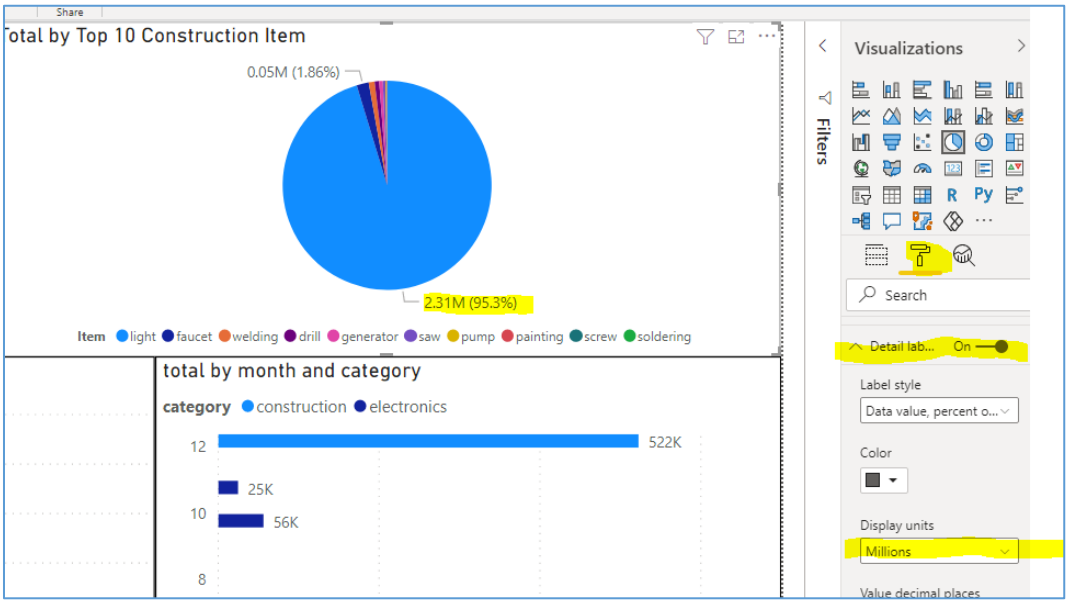
While the chart is selected, click on the formatting icon. Then, turn on Data Labels, and switch the Display units to Thousands:



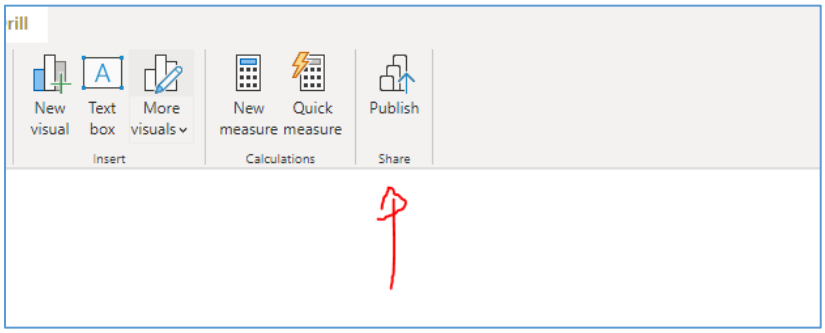
You should now see four visuals on your dashboard, one for each table listed under the fields pane. To format your visual, such as centering headings, adding borders and positioning the legend, click on the visual you want to format then on the formatting icon. You can alter any formatting from this pane:



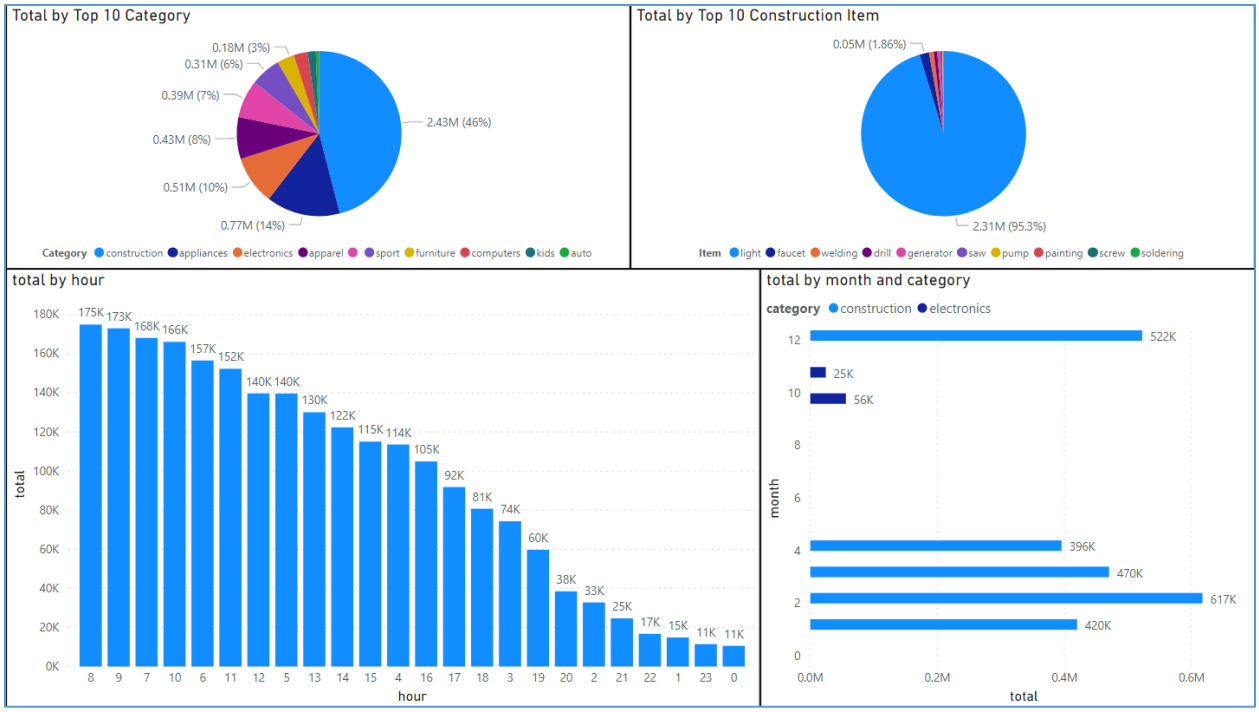
7. On the formatting section, you can change the numbering format. For example, you can click on the Detail Labels section while the pie chart is selected, and select Millions as the display unit with two decimal places:



8. Now that your report is complete, you will publish it to the workspace you created earlier in this step. Click on the Publish button located under the Home ribbon. You will be prompt to select a workspace. Select the one you created earlier.



Now that your report has been published to the Power BI service, you are ready to share with others! The following is a screenshot of the completed visualization.



References

7. URL of Data Source:
   https://drive.google.com/drive/folders/1Nan8X33H8xrXS5XhCKZmSpClFTCJsSpE
   https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store

8. URL of your Github:

   https://github.com/jramirez162/ecommerce_behavior.git