Tricia Harris, Sara Kayhan, Catherine Poirier, Aishwarya Rao, Jason Sheridan
DU Data Analytics Bootcamp
ETL Project | Submitted 3/13/2021

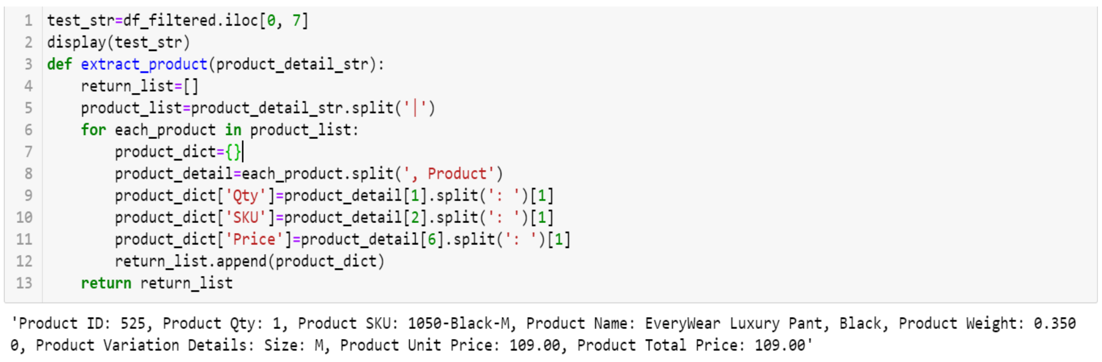
Project Report

Introduction: ETL (extract, transform, load) is an integral step to any data analytics project. The motivation behind performing ETL for Body Bark came from its founder, Catherine Poirier. Body Bark is a sustainable clothing brand whose mission is to create apparel that empowers the wearer. Catherine was interested in the numerous possibilities her company's data could portray. Furthermore, Catherine wanted to clean the data and standardize the formatting of the various data types. After creating an ERD and analyzing Body Bark's data for one week, below is the documentation for performing ETL on Body Bark's database.

- **E**xtract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).
  - Body Bark's CSV files
    - Internet user's order information was captured using BigCommerce
    - Boutique information was captured using QuickBooks
  - Created a schema using QuickERD tool
    - Helped us see how to declare primary and foreign keys

- **T**ransform: what data cleaning or transformation was required? This step was the most time consuming as the team had to work with schema constraints. Schema constraints must be acknowledged when working with a SQL database or else the tables will not properly load into the database. The transformation of all csv files was done using the Pandas library available to Python in Jupyter Notebook.
  - "Internet_Customers" and "Boutiques" tables: formatting of zip code
    - Some zip codes in the "zip" column were missing a leading 0 due to the fact that they were in a csv, which strips leading 0s, so we added the leading 0 back in using Pandas module within Python (new_df = df_name.rename(columns={"1. open": "open", "2. high": "high" }).
    - Some zip codes in the "zip" column had the full postal zip code (i.e. 12345-4444) so we removed all the digits following the hyphen using a lambda function in Python (df['Col1']=df['Col1'].apply(lambda x: '{0:0>5}'.format(x)).
  - "Boutiques" table: created a "boutique_id" using the index available in Pandas

- "Internet_Order" table: we dropped all extraneous columns that we did not need for analysis
    - "Product Details" column was partitioned into sku, total_each_sku and product_unit_price. Each of these newly created columns were generated into a new dataframe. We created a function to achieve the splitting along with a for loop. A dictionary was also created to store the string values. The code is as follows:



```
test_str=df_filtered.iloc[0, 7]
display(test_str)
def extract_product(product_detail_str):
    return_list=[]
    product_list=product_detail_str.split('|')
    for each_product in product_list:
        product_dict={}
        product_detail=each_product.split(', Product')
        product_dict['Qty']=product_detail[1].split(': ')[1]
        product_dict['SKU']=product_detail[2].split(': ')[1]
        product_dict['Price']=product_detail[6].split(': ')[1]
        return_list.append(product_dict)
    return return_list
```

```
'Product ID: 525, Product Qty: 1, Product SKU: 1050-Black-M, Product Name: EveryWear Luxury Pant, Black, Product Weight: 0.350
0, Product Variation Details: Size: M, Product Unit Price: 109.00, Product Total Price: 109.00'
```

    - We dropped many rows from the "order_status" column for orders that did not ship, the value counts and .unique functions were used in Python to see the status of shipments and to verify if the outlier shipments were dropped.
        - Got rid of "Canceled," "Fraud," "Declined," and "Awaiting Payment" shipment statuses to help drop outliers.
    - Converted the "coupon"column to boolean values of true or false
    - "Month" column was created from (mm-dd-yy) format to solely (mm) format
    - Final step in "Internet_Orders" was to reorder the column names to align with our schema. This step is crucial to help ensure that tables load properly into SQL.
- Normalization of data
- For future reference, we should have addressed the dropping of "Nan" aka null values in Pandas to help expedite the loading of tables into SQL. Since we did not address null values in Python, we received error messages when trying to

import tables in SQL since we gave the constraint of not null in the create table statements.

- **L**oad: the final database, tables/collections, and why this was chosen. Justification for choosing a SQL data structure:
    - Loaded newly cleaned data (generated by to_csv method in Python) into Postgres. Ensured to load the tables in the proper order so the database would function properly. This means that if you have a table with a foreign key, it cannot be loaded into Postgres prior to the loading of that table's primary key. The rigidity of SQL's schemas is one disadvantage of choosing to work with a SQL database as opposed to a NoSQL database.
        - Working with a relational database lends itself well to SQL, especially when the data goes from first normal form (1NF) to second and third normal form (2NF, 3NF.) BodyBark's original dataset (various csv files generated from BigCommerce and QuickBooks) was in 1NF and we were able to get the data into 2NF.
        - Structured schemas for database, which is another reason we used an SQL database.
        - Relatively small amounts of data → SQL
- Conclusion: Why would this be useful?
    - Use for future insights into Body Bark's marketing
        - Use cases for what someone could do with your database:
            - Determine where zip codes match for boutiques and internet orders
            - Determine which store locations may be driving internet orders
            - Determine the percentage of orders that use coupons
            - Determine how many internet customers order more than once, twice, three times etc.
            - Determine how month affects total sales and sales of particular products and color.
            - Determine if zip code affects products ordered.
            - Determine customer traits based on zip codes.

- For future study it would be interesting to look at customer zip code demographics from other sources and find relationships.