Yaisah Granillo 03/27/2019

API Instructions

Dillard's Transaction Data API

Introduction

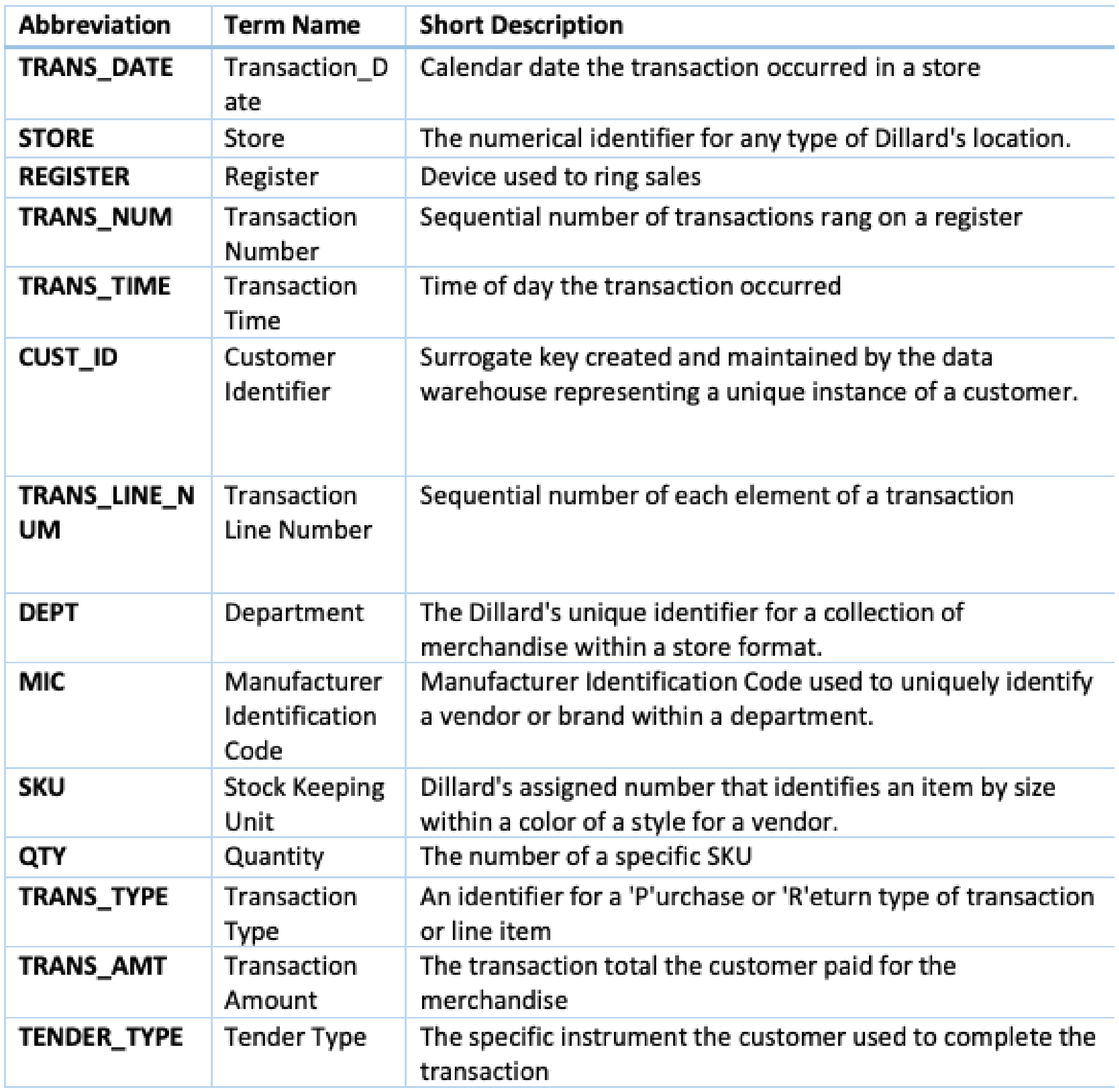
The Dillard's API provides access to JSON endpoints for Dillard's transactions that took place from 2014-2016. They are designed to meet the demands of enterprise business platforms to understand the customers better.

This API collects different portions of the Dillard's database. The Dillard's Department Store Database (UA_Dillards_2016) contains millions of records of information gathered from sales transactions from the Dillard's stores. For this API, the focus is on the Transaction table within the database. The API returns a JSON object that can be utilized for integration and development.

APIs available:

- http://0.0.0.0:5000/
- http://0.0.0.0:5000/tranamount
- http://0.0.0.0:5000/transaction/215663021

Dillard’s Transaction Table



Abbreviation	Term Name	Short Description
TRANS_DATE	Transaction_Date	Calendar date the transaction occurred in a store
STORE	Store	The numerical identifier for any type of Dillard's location.
REGISTER	Register	Device used to ring sales
TRANS_NUM	Transaction Number	Sequential number of transactions rang on a register
TRANS_TIME	Transaction Time	Time of day the transaction occurred
CUST_ID	Customer Identifier	Surrogate key created and maintained by the data warehouse representing a unique instance of a customer.
TRANS_LINE_NUM	Transaction Line Number	Sequential number of each element of a transaction
DEPT	Department	The Dillard's unique identifier for a collection of merchandise within a store format.
MIC	Manufacturer Identification Code	Manufacturer Identification Code used to uniquely identify a vendor or brand within a department.
SKU	Stock Keeping Unit	Dillard's assigned number that identifies an item by size within a color of a style for a vendor.
QTY	Quantity	The number of a specific SKU
TRANS_TYPE	Transaction Type	An identifier for a 'P'urchase or 'R'eturn type of transaction or line item
TRANS_AMT	Transaction Amount	The transaction total the customer paid for the merchandise
TENDER_TYPE	Tender Type	The specific instrument the customer used to complete the transaction

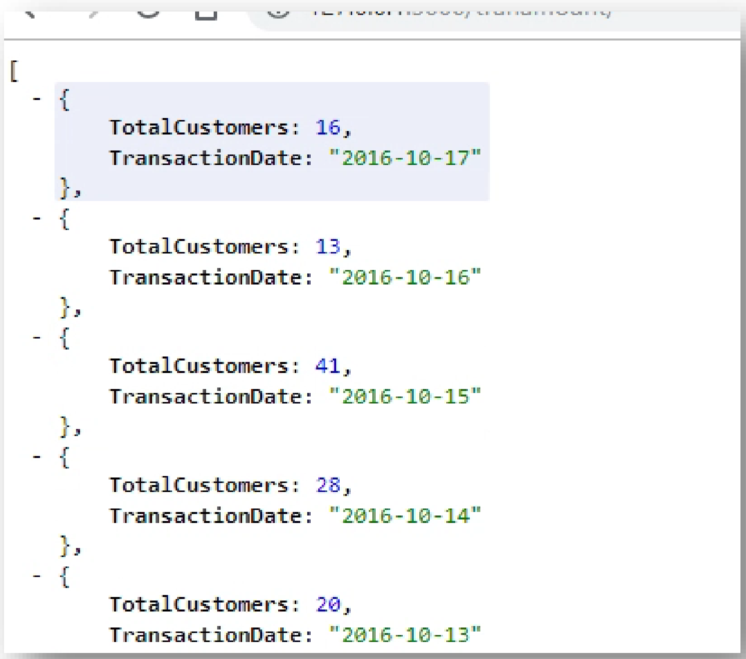
Endpoint Overview

Endpoint category	Description
/tranamount/	Endpoint that return data on the number of customers that spent over $500 by transaction date.
/transaction/	Endpoints that return data around transaction information such as date, time, transaction type, amount, etc.

List amount of customers that spent over $500 (Static API)

- For a static API with information on the number of customers that spent over $500 in 2014-2016 by transaction date.

Sample Output:



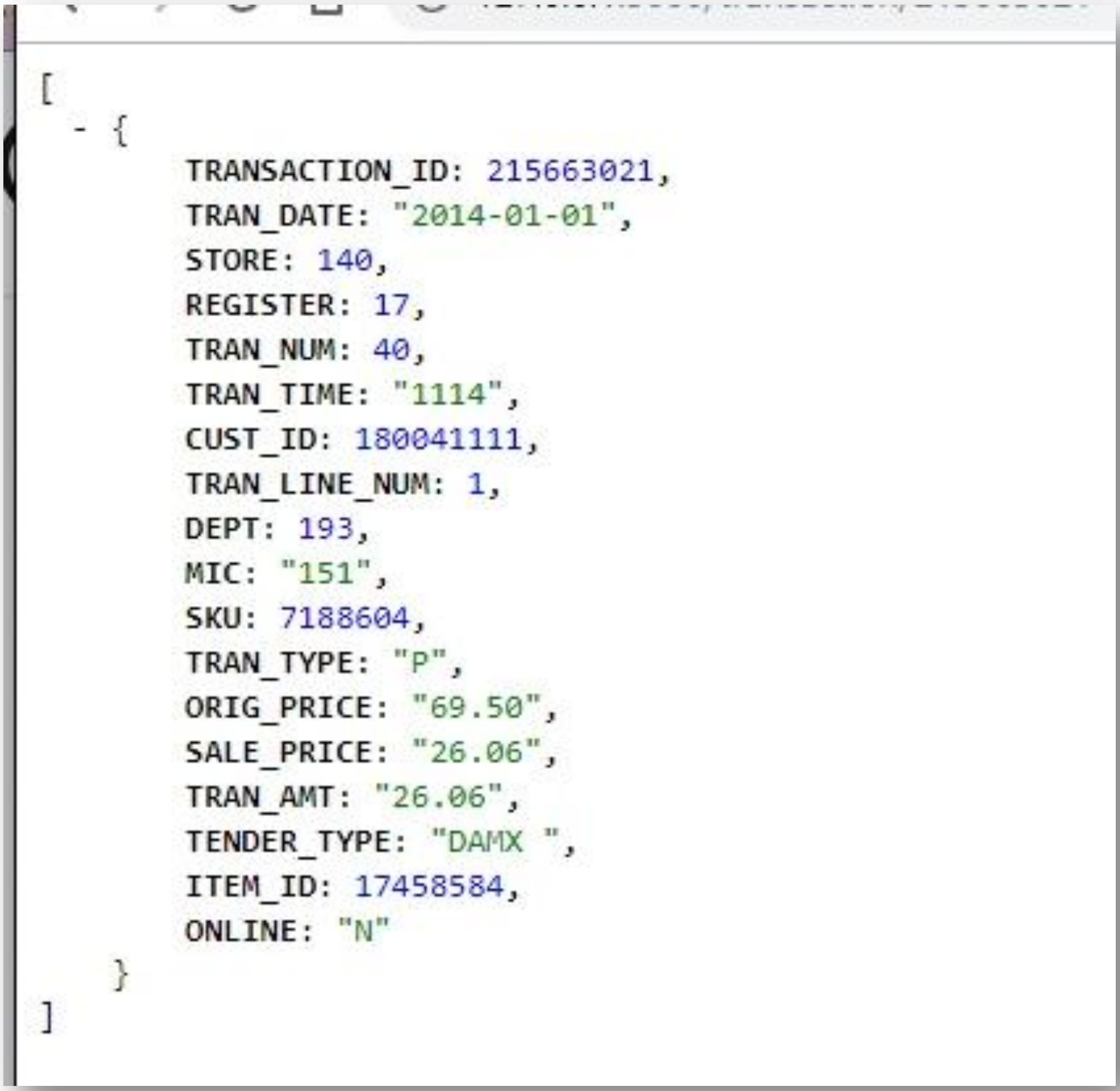
```
[
  - {
        TotalCustomers: 16,
        TransactionDate: "2016-10-17"
    },
  - {
        TotalCustomers: 13,
        TransactionDate: "2016-10-16"
    },
  - {
        TotalCustomers: 41,
        TransactionDate: "2016-10-15"
    },
  - {
        TotalCustomers: 28,
        TransactionDate: "2016-10-14"
    },
  - {
        TotalCustomers: 20,
        TransactionDate: "2016-10-13"
```

List of transaction data per transaction ID (Dynamic API)

API: http://0.0.0.0:5000/transaction/<id>

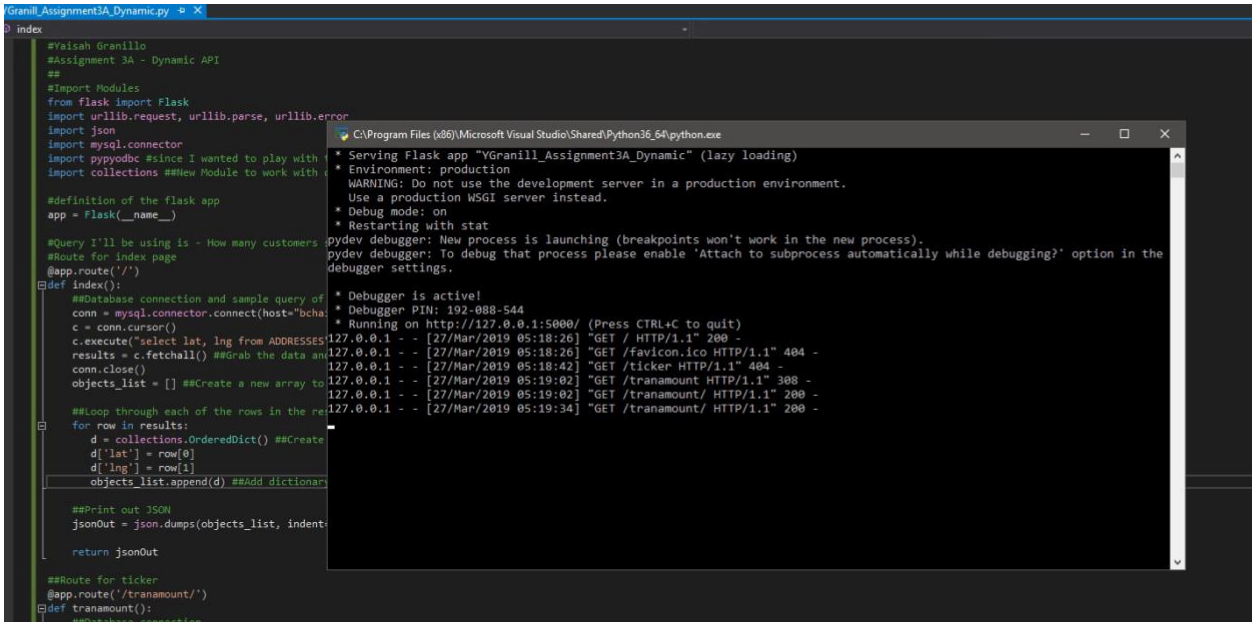
- A dynamic API with all Dillard's transaction information that took place from 2014-2016 for any specific Dillard's transactions ID requested. You may replace the transaction ID <id> with any you desire to view specific information.

Sample Output:
http://0.0.0.0:5000/transaction/12345678



```
[
  - {
        TRANSACTION_ID: 215663021,
        TRAN_DATE: "2014-01-01",
        STORE: 140,
        REGISTER: 17,
        TRAN_NUM: 40,
        TRAN_TIME: "1114",
        CUST_ID: 180041111,
        TRAN_LINE_NUM: 1,
        DEPT: 193,
        MIC: "151",
        SKU: 7188604,
        TRAN_TYPE: "P",
        ORIG_PRICE: "69.50",
        SALE_PRICE: "26.06",
        TRAN_AMT: "26.06",
        TENDER_TYPE: "DAMX ",
        ITEM_ID: 17458584,
        ONLINE: "N"
    }
]
```

Dynamic API running



```
#Yaisah Granillo
#Assignment 3A - Dynamic API
##
#Import Modules
from flask import Flask
import urllib.request, urllib.parse, urllib.error
import json
import mysql.connector
import pypyodbc #since I wanted to play with
import collections ##New Module to work with

#definition of the flask app
app = Flask(__name__)

#Query I'll be using is - How many customers
#Route for index page
@app.route('/')
def index():
    ##Database connection and sample query of
    conn = mysql.connector.connect(host="bcha
    c = conn.cursor()
    c.execute("select lat, lng from ADDRESSES
    results = c.fetchall() ##Grab the data and
    conn.close()
    objects_list = [] ##Create a new array to

    ##Loop through each of the rows in the re
    for row in results:
        d = collections.OrderedDict() ##Create
        d['lat'] = row[0]
        d['lng'] = row[1]
        objects_list.append(d) ##Add dictionary

    ##Print out JSON
    jsonOut = json.dumps(objects_list, indent

    return jsonOut

##Route for ticker
@app.route('/tranamount/')
def tranamount():
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
 * Serving Flask app "YGranill_Assignment3A_Dynamic" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
pydev debugger: New process is launching (breakpoints won't work in the new process).
pydev debugger: To debug that process please enable 'Attach to subprocess automatically while debugging?' option in the
debugger settings.

 * Debugger is active!
 * Debugger PIN: 192-088-544
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Mar/2019 05:18:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2019 05:18:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Mar/2019 05:18:42] "GET /ticker HTTP/1.1" 404 -
127.0.0.1 - - [27/Mar/2019 05:19:02] "GET /tranamount HTTP/1.1" 308 -
127.0.0.1 - - [27/Mar/2019 05:19:02] "GET /tranamount/ HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2019 05:19:34] "GET /tranamount/ HTTP/1.1" 200 -
```