|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  |  |  | |  |
| Brazilian E-Commerce  Final Report Project 2: ETL  UCSD Data Science and Visualization Bootcamp | | |
| Team EDW:  Amir Vazquez Narjes Taqvaei Annette Broeren | | February 24, 2020 |

# Introduction

The objective of this report is to describe the approach and process of transforming a dataset into a data warehouse for analytical purposes and business reporting. For this project, Brazilian E-Commerce and marketing funnel datasets released by Olist Stores were used.

The E-Commerce dataset contains data for approximately 100,000 transactions from 2016 to 2018 made at multiple marketplaces in Brazil.

The marketing funnel dataset contains data from sellers that filled-in requests for contact to sell their products on Olist Store. The dataset has information of approximately 8,000 Marketing Qualified Leads (MQLs) collected between June 1, 2017 and June 1, 2018. They were randomly sampled from the total MQLs.

The following tasks were explored:

***Extract*** original data sources;

***Transform*** data into clean repositories

***Load*** the final repositories into a data warehouse

# Sources

1. Brazilian E-Commerce Public Dataset by Olist (<https://www.kaggle.com/olistbr/brazilian-ecommerce>)

9 CSV files

52 columns

13 string

13 Integer

12 Uuid

14 Other

1. Marketing Funnel by Olist (<https://www.kaggle.com/olistbr/marketing-funnel-olist>)

2 CSV files

18 columns

6 string

6 Uuid

2 DateTime

4 Other

# Approach

The approach consists of a two-tier stage environment where the raw data is imported into the database, then prepared to populate final tables for the data warehouse.

* Download the source files to a Github repository with branches for all team members
* Create staging tables from the source files in PGAdmin
* Add 4 columns: Create-date, Created\_by, Changed\_date, Changed\_by
* Create a date dimension table to facilitate analysis based on dates and periods
* Use Python to
  + Inspect the tables
  + Convert date/time and date strings to date dimensions and integers
  + Add triggers to auto-populate date created plus username as well as date updated plus username
* Load reformatted tables into PGAdmin



# Execution

## Raw data to Staging

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Source file** | To | **Staging table** |
| 1 | olist\_customer\_dataset.csv |  | stg\_olist\_customer\_dataset |
| 2 | olist\_geolocation\_dataset.csv |  | stg\_olist\_geolocation\_dataset |
| 3 | olist\_order\_items\_dataset.csv |  | stg\_olist\_order\_items\_dataset |
| 4 | olist\_order\_payments\_dataset.csv |  | stg\_olist\_order\_payments\_dataset |
| 5 | olist\_order\_reviews\_dataset.csv |  | stg\_olist\_order\_reviews\_dataset |
| 6 | olist\_orders\_dataset.csv |  | stg\_olist\_orders\_dataset |
| 7 | olist\_products\_dataset.csv |  | stg\_olist\_products\_dataset |
| 8 | olist\_sellers\_dataset.csv |  | stg\_olist\_sellers\_dataset |
| 9 | product\_category\_name\_translation.csv |  | stg\_product\_category\_name\_translation |
| 10 | olist\_marketing\_qualified\_leads\_dataset.csv |  | stg\_olist\_marketing\_qualified\_leads\_dataset |
| 11 | olist\_closed\_deals\_dataset.csv |  | stg\_olist\_closed\_deals\_dataset |

## Create tables and Columns

Sample code:

-- Table: public.products

DROP TABLE IF EXISTS public.products;

CREATE TABLE public.products

(

    product\_id character varying(40) COLLATE pg\_catalog."default" NOT NULL,

    product\_category\_name character varying(50) COLLATE pg\_catalog."default",

    product\_name\_lenght integer,

    product\_description\_lenght integer,

    product\_photos\_qty integer,

    product\_weight\_g integer,

    product\_length\_cm integer,

    product\_height\_cm integer,

    product\_width\_cm integer,

    "CREATE\_DATE" date,

    "CREATED\_BY " character varying(60)[] COLLATE pg\_catalog."default",

    "CHANGED\_DATE" date,

    "CHANGED\_BY" character varying(60) COLLATE pg\_catalog."default",

    CONSTRAINT products\_pkey PRIMARY KEY (product\_id)

)

## INsert Data

/\*Insert data from olist\_customer\_dataset.csv\*/

COPY stg\_olist\_customer\_dataset (

    customer\_id,

    customer\_unique\_id,

    customer\_zip\_code\_prefix,

    customer\_city,

    customer\_state)

/\*Update your location of the files here\*/

FROM ‘(PATH)\Resources/olist\_customers\_dataset.csv' DELIMITER ',' CSV HEADER;

## staging to final

|  |  |  |  |
| --- | --- | --- | --- |
|  | Staging Table | To | Final Table |
| 1 | stg\_olist\_customer\_dataset |  | customer |
| 2 | stg\_olist\_geolocation\_dataset |  | geolocation |
| 3 | stg\_olist\_order\_items\_dataset |  | order\_items |
| 4 | stg\_olist\_order\_payments\_dataset |  | order\_payments |
| 5 | stg\_olist\_order\_reviews\_dataset |  | order\_reviews |
| 6 | stg\_olist\_orders\_dataset |  | orders |
| 7 | stg\_olist\_products\_dataset |  | products |
| 8 | stg\_olist\_sellers\_dataset |  | sellers |
| 9 | stg\_product\_category\_name\_translation |  | prod\_category |
| 10 | stg\_olist\_marketing\_qualified\_leads\_dataset |  | mkt\_leads |
| 11 | stg\_olist\_closed\_deals\_dataset |  | mkt\_deals |

## DAte Conversions

Source files contained columns with dates formatted as date stamps showing date and time. However, some columns showed a the time of 0:00 for every date in that column so these columns required different treatment than the other date columns.

Date stamps were inserted into the staging tables as text (date/time) or date if the timestamp was 0:00. Text-dates showed the date stamp inside quotes.

Any text values for dates had to be transformed to time stamp values for the final .SQL database.

All dates had to be transformed to integer values for use with a date dimension table. The date dimension table was created to serve as a key for date columns for period-analysis purposes.

Awesome code ahead:

### Transfer text to date/time value using

dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')



### Transfer text to date/time value for columns include empty or Null value using:

orders\_df['order\_approved\_at']=stg\_orders\_df['order\_approved\_at'].apply(lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S') if (x!=None) else None)



### Create integer value from text (date/time) value

int(x[:x.find(' ')].replace('-',''))



### Handling Null values



### Create Integer value from Date:

int(x.strftime('%Y%m%d'))



## Date Dimension Table

The date dimension table was created as a tool to perform period-analysis using sql.

Awesome code ahead:

/\* script to create the date dimension\*/

DROP TABLE if exists date\_dim;

CREATE TABLE date\_dim

(

  date\_dim\_id              INT NOT NULL,

  date\_actual              DATE NOT NULL,

  epoch                    BIGINT NOT NULL,

  day\_suffix               VARCHAR(4) NOT NULL,

  day\_name                 VARCHAR(9) NOT NULL,

  day\_of\_week              INT NOT NULL,

  day\_of\_month             INT NOT NULL,

  day\_of\_quarter           INT NOT NULL,

  day\_of\_year              INT NOT NULL,

  week\_of\_month            INT NOT NULL,

  week\_of\_year             INT NOT NULL,

  week\_of\_year\_iso         CHAR(10) NOT NULL,

  month\_actual             INT NOT NULL,

  month\_name               VARCHAR(9) NOT NULL,

  month\_name\_abbreviated   CHAR(3) NOT NULL,

  quarter\_actual           INT NOT NULL,

  quarter\_name             VARCHAR(9) NOT NULL,

  year\_actual              INT NOT NULL,

  first\_day\_of\_week        DATE NOT NULL,

  last\_day\_of\_week         DATE NOT NULL,

  first\_day\_of\_month       DATE NOT NULL,

  last\_day\_of\_month        DATE NOT NULL,

  first\_day\_of\_quarter     DATE NOT NULL,

  last\_day\_of\_quarter      DATE NOT NULL,

  first\_day\_of\_year        DATE NOT NULL,

  last\_day\_of\_year         DATE NOT NULL,

  mmyyyy                   CHAR(6) NOT NULL,

  mmddyyyy                 CHAR(10) NOT NULL,

  weekend\_indr             BOOLEAN NOT NULL

);

ALTER TABLE public.date\_dim ADD CONSTRAINT d\_date\_date\_dim\_id\_pk PRIMARY KEY (date\_dim\_id);

CREATE INDEX d\_date\_date\_actual\_idx

  ON date\_dim(date\_actual);

COMMIT;

INSERT INTO date\_dim

SELECT TO\_CHAR(datum, 'yyyymmdd')::INT AS date\_dim\_id,

       datum AS date\_actual,

       EXTRACT(EPOCH FROM datum) AS epoch,

       TO\_CHAR(datum, 'fmDDth') AS day\_suffix,

       TO\_CHAR(datum, 'Day') AS day\_name,

       EXTRACT(ISODOW FROM datum) AS day\_of\_week,

       EXTRACT(DAY FROM datum) AS day\_of\_month,

       datum - DATE\_TRUNC('quarter', datum)::DATE + 1 AS day\_of\_quarter,

       EXTRACT(DOY FROM datum) AS day\_of\_year,

       TO\_CHAR(datum, 'W')::INT AS week\_of\_month,

       EXTRACT(WEEK FROM datum) AS week\_of\_year,

       EXTRACT(ISOYEAR FROM datum) || TO\_CHAR(datum, '"-W"IW-') || EXTRACT(ISODOW FROM datum) AS week\_of\_year\_iso,

       EXTRACT(MONTH FROM datum) AS month\_actual,

       TO\_CHAR(datum, 'Month') AS month\_name,

       TO\_CHAR(datum, 'Mon') AS month\_name\_abbreviated,

       EXTRACT(QUARTER FROM datum) AS quarter\_actual,

       CASE

           WHEN EXTRACT(QUARTER FROM datum) = 1 THEN '1Q-'||EXTRACT(YEAR FROM datum)

           WHEN EXTRACT(QUARTER FROM datum) = 2 THEN '2Q-'||EXTRACT(YEAR FROM datum)

           WHEN EXTRACT(QUARTER FROM datum) = 3 THEN '3Q-'||EXTRACT(YEAR FROM datum)

           WHEN EXTRACT(QUARTER FROM datum) = 4 THEN '4Q-'||EXTRACT(YEAR FROM datum)

           END AS quarter\_name,

       EXTRACT(ISOYEAR FROM datum) AS year\_actual,

       datum + (1 - EXTRACT(ISODOW FROM datum))::INT AS first\_day\_of\_week,

       datum + (7 - EXTRACT(ISODOW FROM datum))::INT AS last\_day\_of\_week,

       datum + (1 - EXTRACT(DAY FROM datum))::INT AS first\_day\_of\_month,

       (DATE\_TRUNC('MONTH', datum) + INTERVAL '1 MONTH - 1 day')::DATE AS last\_day\_of\_month,

       DATE\_TRUNC('quarter', datum)::DATE AS first\_day\_of\_quarter,

       (DATE\_TRUNC('quarter', datum) + INTERVAL '3 MONTH - 1 day')::DATE AS last\_day\_of\_quarter,

       TO\_DATE(EXTRACT(YEAR FROM datum) || '-01-01', 'YYYY-MM-DD') AS first\_day\_of\_year,

       TO\_DATE(EXTRACT(YEAR FROM datum) || '-12-31', 'YYYY-MM-DD') AS last\_day\_of\_year,

       TO\_CHAR(datum, 'mmyyyy') AS mmyyyy,

       TO\_CHAR(datum, 'mmddyyyy') AS mmddyyyy,

       CASE

           WHEN EXTRACT(ISODOW FROM datum) IN (6, 7) THEN TRUE

           ELSE FALSE

           END AS weekend\_indr

       /\*2015-01-01 is the start day of the date dimension \*/

FROM (SELECT '2015-01-01'::DATE + SEQUENCE.DAY AS datum

    /\* 2000 numberis the number of days to generate.

    GENERATE\_SERIES(0, 2000) \*/

      FROM GENERATE\_SERIES(0, 2000) AS SEQUENCE (DAY)

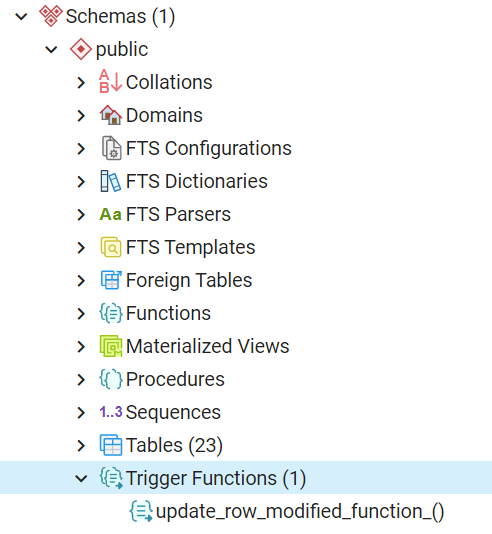
      GROUP BY SEQUENCE.DAY) DQ

ORDER BY 1;

COMMIT;

## Triggers

To track creation and update data for every table, a trigger script was created. Trigger scripts can be created in the Trigger function in Schemas, and then added to all applicable tables (right click on table name -> create ->trigger).



Awesome code ahead:

- FUNCTION: public.update\_row\_modified\_function\_()

-- DROP FUNCTION public.update\_row\_modified\_function\_();

CREATE FUNCTION public.update\_row\_modified\_function\_()

    RETURNS trigger

    LANGUAGE 'plpgsql'

    COST 100

    VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

IF TG\_OP = 'INSERT' THEN

NEW.create\_date = CURRENT\_TIMESTAMP(0);

NEW.created\_by = CURRENT\_USER;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

NEW.changed\_date = CURRENT\_TIMESTAMP(0);

NEW.changed\_by = CURRENT\_USER;

RETURN NEW;

END IF;

END;

$BODY$;

ALTER FUNCTION public.update\_row\_modified\_function\_()

    OWNER TO postgres;

## Time to ~~code~~ play!

Check it out at:

https://github.com/nt1983/Team-A-Kaggle\_Brazilian\_E\_commerce.git

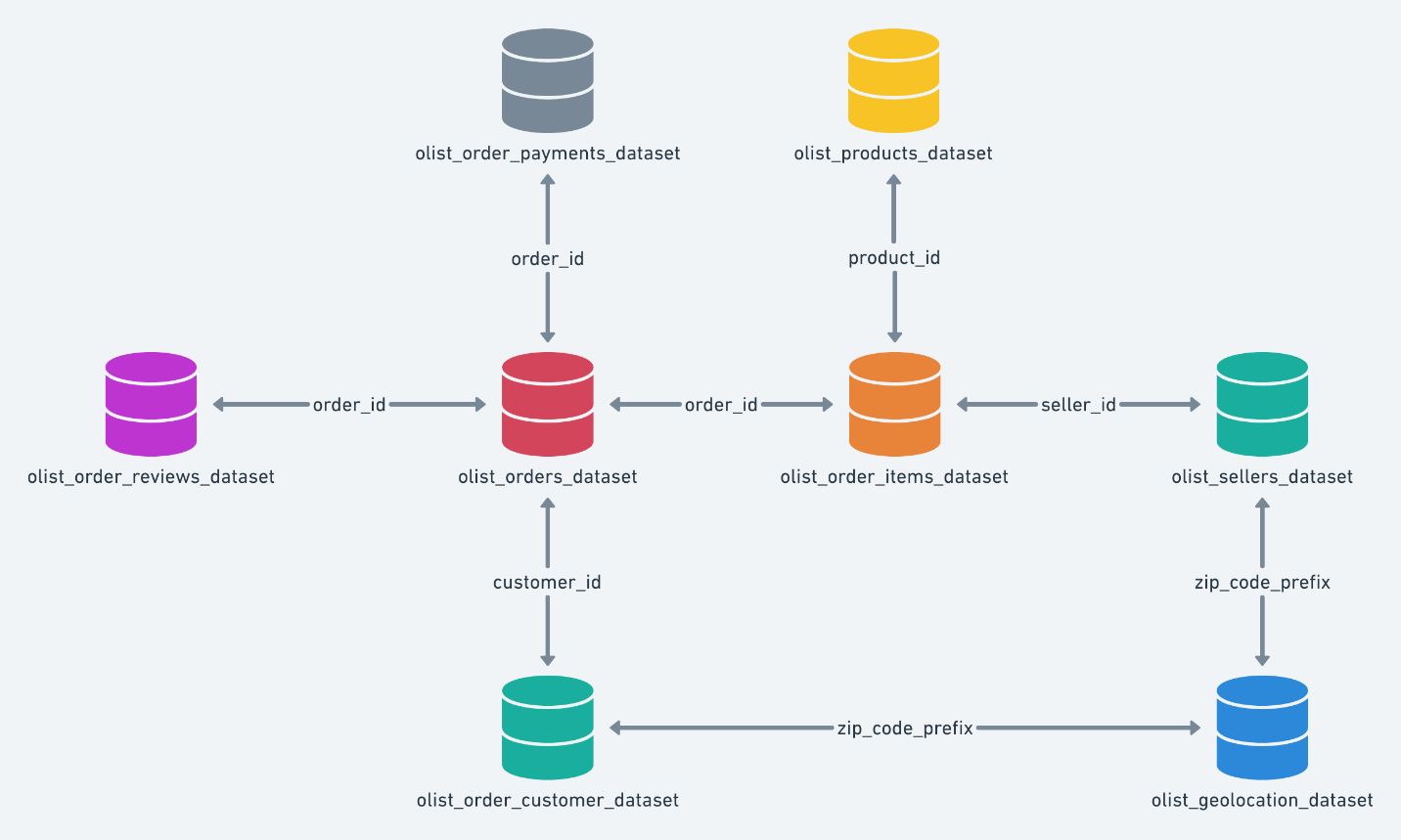
# Data Warehouse

Figure 1: Schema

Graphical user interface, text, application, chat or text message

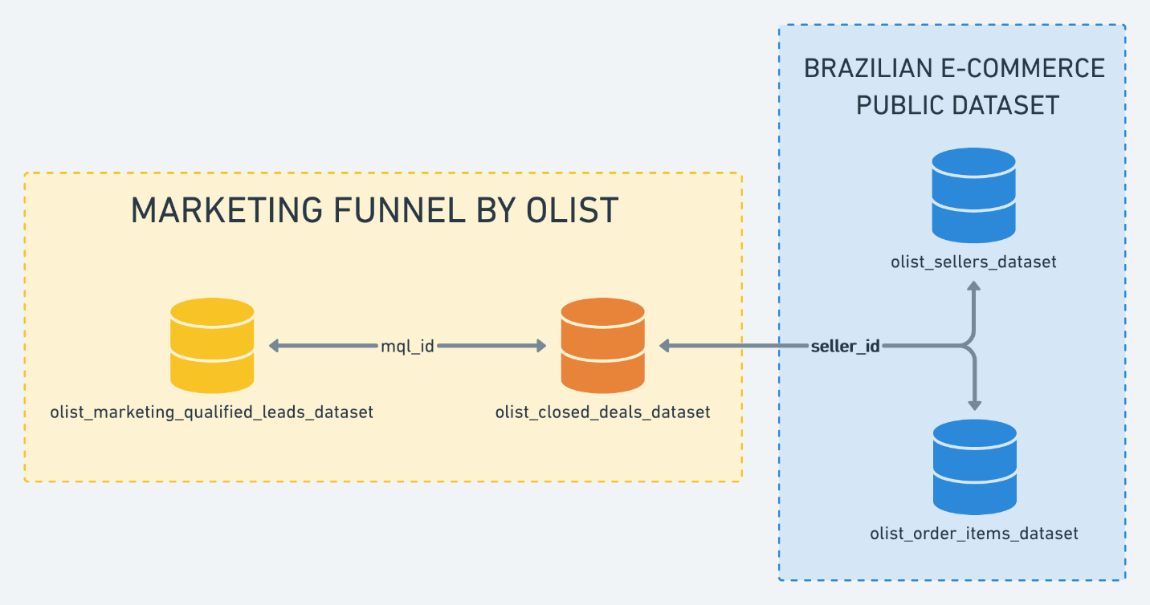
Description automatically generated

Figure 2: Data Schema for Brazilian E-Commerce Public Dataset



Source: <https://www.kaggle.com/olistbr/brazilian-ecommerce>

Figure 3: Data Schema for Marketing Funnel Dataset

Source: https://www.kaggle.com/olistbr/marketing-funnel-olist/home

# Issues Encountered

## Source Files

Source files were read only, so the SQL server could not read the data.

### Solution:

1. Right click on file name
2. Go to security tab
3. Add Everyone to usernames
4. Change permissions to allow for all

Note: Pushing and pulling from Github may reverse these property-settings.

## Github

An empty .git file caused fatal push and pull errors for a team member. This file was located in the root of the C-drive and had to be deleted in order for git to resume normal functionality for that team member.