BRAZILIAN E-COMMERCE

FINAL REPORT FOR ETL PROJECT

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

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

   2 CSV files

   18 columns

   6 string

   6 Uuid

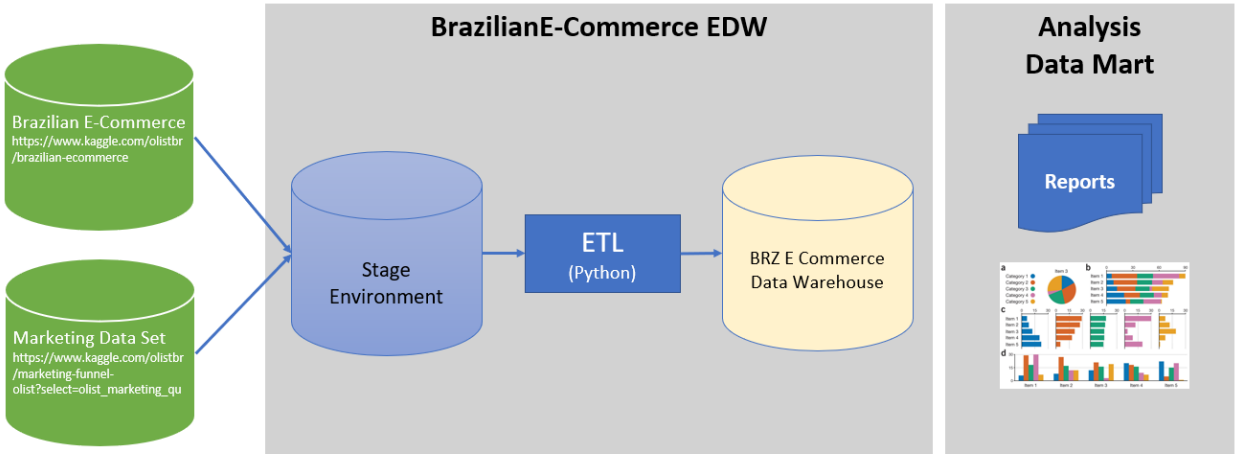
   2 DateTime

   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 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.

Text(Date/ Time: "2018-10-10 13:10:23"

Date: 2018-10-10

date_id: 20181010

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.

A	B	C	D	E	F	G	H
order_id	customer_	order_sta	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
e481f51cb	9ef432eb6	delivered	10/2/2017 10:56	10/2/2017 11:07	10/4/2017 19:55	10/10/2017 21:25	10/18/2017 0:00
53cdb2fc8	b0830fb47	delivered	7/24/2018 20:41	7/26/2018 3:24	7/26/2018 14:31	8/7/2018 15:27	8/13/2018 0:00
47770eb9	41ce2a54c	delivered	8/8/2018 8:38	8/8/2018 8:55	8/8/2018 13:50	8/17/2018 18:06	9/4/2018 0:00
949d5b44	f88197465	delivered	11/18/2017 19:28	11/18/2017 19:45	11/22/2017 13:39	12/2/2017 0:28	12/15/2017 0:00
ad21c59c0	8ab97904e	delivered	2/13/2018 21:18	2/13/2018 22:20	2/14/2018 19:46	2/16/2018 18:17	2/26/2018 0:00
a4591c265	503740e9c	delivered	7/9/2017 21:57	7/9/2017 22:10	7/11/2017 14:58	7/26/2017 10:57	8/1/2017 0:00
136cce7fa	ed0271e0	invoiced	4/11/2017 12:22	4/13/2017 13:25			5/9/2017 0:00
6514b8ad8	9bdf08b4	delivered	5/16/2017 13:10	5/16/2017 13:22	5/22/2017 10:07	5/26/2017 12:55	6/7/2017 0:00
76c6e8662	f54a9f0e6	delivered	1/23/2017 18:29	1/25/2017 2:50	1/26/2017 14:16	2/2/2017 14:08	3/6/2017 0:00
e69bfb5e	31ad1d1b	delivered	7/29/2017 11:55	7/29/2017 12:05	8/10/2017 19:45	8/16/2017 17:14	8/23/2017 0:00
e6ce16cb	494dded5	delivered	5/16/2017 19:41	5/16/2017 19:50	5/18/2017 11:40	5/29/2017 11:18	6/7/2017 0:00
34513ce0c	7711cf624	delivered	7/13/2017 19:58	7/13/2017 20:10	7/14/2017 18:43	7/19/2017 14:04	8/8/2017 0:00
82566a66	d3e3b74c	delivered	6/7/2018 10:06	6/9/2018 3:13	6/11/2018 13:29	6/19/2018 12:05	7/18/2018 0:00
5ff96c15d	19402a48f	delivered	7/25/2018 17:44	7/25/2018 17:55	7/26/2018 13:16	7/30/2018 15:52	8/8/2018 0:00
432aaf21d	3df704f53	delivered	3/1/2018 14:14	3/1/2018 15:10	3/2/2018 21:09	3/12/2018 23:36	3/21/2018 0:00
dcb36b51	3b6828a5	delivered	6/7/2018 19:03	6/12/2018 23:31	6/11/2018 14:54	6/21/2018 15:34	7/4/2018 0:00
403b9783	738b0868	delivered	1/2/2018 19:00	1/2/2018 19:09	1/3/2018 18:19	1/20/2018 1:38	2/6/2018 0:00
116f0b093	3187789b	delivered	12/26/2017 23:41	12/26/2017 23:50	12/28/2017 18:33	1/8/2018 22:36	1/29/2018 0:00
85ce859fd	059f7fc57	delivered	11/21/2017 0:03	11/21/2017 0:14	11/23/2017 21:32	11/27/2017 18:28	12/11/2017 0:00
83018ec11	7f8c8b9c2	delivered	10/26/2017 15:54	10/26/2017 16:08	10/26/2017 21:46	11/8/2017 22:22	11/23/2017 0:00
203096f03	d2b09157	delivered	9/18/2017 14:31	9/19/2017 4:04	10/6/2017 17:50	10/9/2017 22:23	9/28/2017 0:00
f848643ee	4fa1cd166	delivered	3/15/2018 8:52	3/15/2018 9:09	3/15/2018 19:52	3/19/2018 18:08	3/29/2018 0:00
2807d0e5	72ae2816	delivered	2/3/2018 20:37	2/3/2018 20:50	2/5/2018 22:37	2/8/2018 16:13	2/21/2018 0:00

Awesome code ahead:

Transfer text to date/time value using

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

```
order_reviews_df['review_answer_timestamp']=stg_order_reviews_df['review_answer_timestamp'].apply(lambda x: 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)

```
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('-',''))

```
order_reviews_df['sk_review_answer_dt']=stg_order_reviews_df['review_answer_timestamp'].apply(lambda x: int(x[:x.find(' ')].replace('-',
'')))
```

Handling Null values

```
orders_df['sk_order_approved_at_dt']=stg_orders_df['order_approved_at'].apply(lambda x: int(x[:x.find(' ')].replace('-','')) if
(x!=None) else None)
```

Create Integer value from Date:

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

```
order_reviews_df['sk_review_creation_dt'] = stg_order_reviews_df['review_creation_date'].apply(lambda x: int(x.strftime('%Y%m%d')))
```

```
#Pull data from stg_olist_orders_dataset table

stg_orders_df=pd.read_sql("select * from stg_olist_orders_dataset", connection)
#stg_orders_df.head()
orders_df=stg_orders_df[['order_id', 'customer_id', 'order_status']].copy()
orders_df['order_purchase_timestamp']=stg_orders_df['order_purchase_timestamp'].apply(lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S') if (x!=None) else None)
orders_df['sk_order_purchase_dt']=stg_orders_df['order_purchase_timestamp'].apply(lambda x: int(x[:x.find(' ')].replace('-','')) if (x!=None) else None)

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)
orders_df['sk_order_approved_at_dt']=stg_orders_df['order_approved_at'].apply(lambda x: int(x[:x.find(' ')].replace('-','')) if (x!=None) else None)

orders_df['order_delivered_carrier_date']=stg_orders_df['order_delivered_carrier_date'].apply(lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S') if (x!=None) else None)
orders_df['sk_order_delivered_carrier_dt']=stg_orders_df['order_delivered_carrier_date'].apply(lambda x: int(x[:x.find(' ')].replace('-','')) if (x!=None) else None)

orders_df['order_delivered_customer_date']=stg_orders_df['order_delivered_customer_date'].apply(lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S') if (x!=None) else None)
orders_df['sk_order_delivered_customer_dt']=stg_orders_df['order_delivered_customer_date'].apply(lambda x: int(x[:x.find(' ')].replace('-','')) if (x!=None) else None)

orders_df['order_estimated_delivery_date']=stg_orders_df['order_estimated_delivery_date']
orders_df['sk_order_estimated_delivery_dt']=stg_orders_df['order_estimated_delivery_date'].apply(lambda x: int(x.strftime('%Y%m%d')))

#orders_df.head(10)


orders_df.to_sql(name='orders', con=engine, if_exists='append', index=False)
```

```
stg_order_reviews_df=pd.read_sql("select * from stg_olist_order_reviews_dataset", connection)
stg_order_reviews_df.head()
```

	review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	None	None	2018-01-18	2018-01-18 21:46:59
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	None	None	2018-03-10	2018-03-11 03:05:13
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	None	None	2018-02-17	2018-02-18 14:36:24
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	None	Recebi bem antes do prazo estipulado.	2017-04-21	2017-04-21 22:02:06
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	None	Parabéns lojas lannister adorei comprar pela I...	2018-03-01	2018-03-02 10:26:53

```
order_reviews_df=stg_order_reviews_df[['review_id', 'order_id', 'review_score', 'review_comment_title', 'review_comment_message', 'review_creation_date']].copy()
order_reviews_df['sk_review_creation_dt'] = stg_order_reviews_df['review_creation_date'].apply(lambda x: int(x.strftime('%Y%m%d')))
order_reviews_df['review_answer_timestamp']=stg_order_reviews_df['review_answer_timestamp'].apply(lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
order_reviews_df['sk_review_answer_dt']=stg_order_reviews_df['review_answer_timestamp'].apply(lambda x: int(x[:x.find(' ')].replace('-','')))
order_reviews_df.head()
```

	review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation_date	sk_review_creation_dt	review_answer_timestamp	sk_review_answer_dt
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	None	None	2018-01-18	20180118	2018-01-18 21:46:59	20180118
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	None	None	2018-03-10	20180310	2018-03-11 03:05:13	20180311
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	None	None	2018-02-17	20180217	2018-02-18 14:36:24	20180218
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	None	Recebi bem antes do prazo estipulado.	2017-04-21	20170421	2017-04-21 22:02:06	20170421
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	None	Parabéns lojas lannister adorei comprar pela I...	2018-03-01	20180301	2018-03-02 10:26:53	20180302

```
order_reviews_df.to_sql(name='order_reviews', con=engine, if_exists='append', index=False)
```

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 (dat
e_dim_id);
```

```sql
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_da
y_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,
```

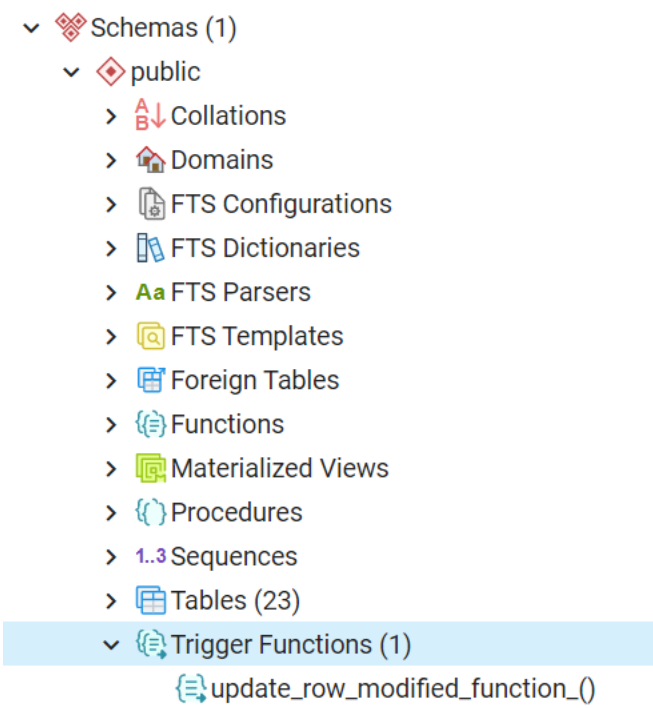
```
        TO_CHAR(datum, 'mmyyyy') AS mmyyyy,
        TO_CHAR(datum, 'mmddyyyy') AS mmddyyyy,
        CASE
            WHEN EXTRACT(ISODOW FROM datum) IN (6, 7) THEN TRUE
            ELSE FALSE
            END AS weekend_indr

        /*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!

This simple query looks a the top products:

Brazilian E-Commerce/postgres@PostgreSQL 13

Query Editor Query History

```
select
delivered_dt.year_actual,
delivered_dt.quarter_name,
p.product_category_name,

sum(cast(price as bigint)) as "Price_sum",
count(o.order_id) as "Order_cnt"

from public.orders o
join public.order_items oi on o.order_id = oi.order_id
join date_dim delivered_dt on delivered_dt.date_dim_id = o.sk_order_delivered_customer_dt

join public.products p on p.product_id = oi.product_id

where o.order_status ='delivered'
group by delivered_dt.year_actual,
delivered_dt.quarter_name,
p.product_category_name
```

Data Output Explain Messages Notifications

	year_actual integer	quarter_name character varying (9)	product_category_name character varying (50)	Price_sum numeric	Order_cnt bigint
1	2018	2Q-2018	relogios_presentes	331084	1724
2	2018	2Q-2018	beleza_saude	311755	2409
3	2018	3Q-2018	beleza_saude	249117	1853
4	2018	1Q-2018	informatica_acessorios	240980	2125
5	2018	2Q-2018	cama_mesa_banho	236077	2494
6	2018	1Q-2018	esporte_lazer	221986	1820
7	2017	4Q-2017	relogios_presentes	217177	1077

Want to try? Check it out at:

https://github.com/nt1983/Team-A-Kaggle_Brazilian_E_commerce.git

DATA WAREHOUSE

Figure 1: Schema

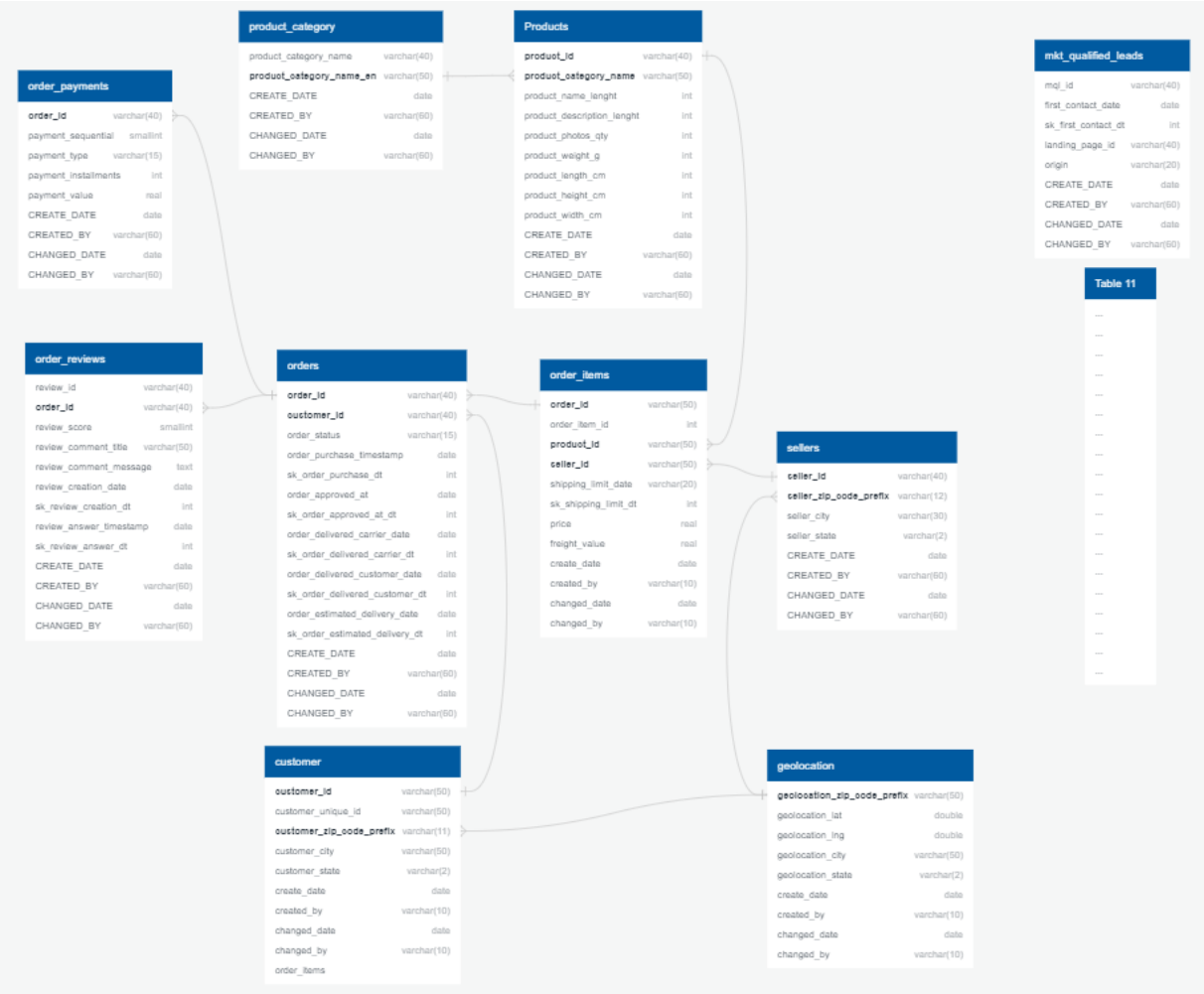
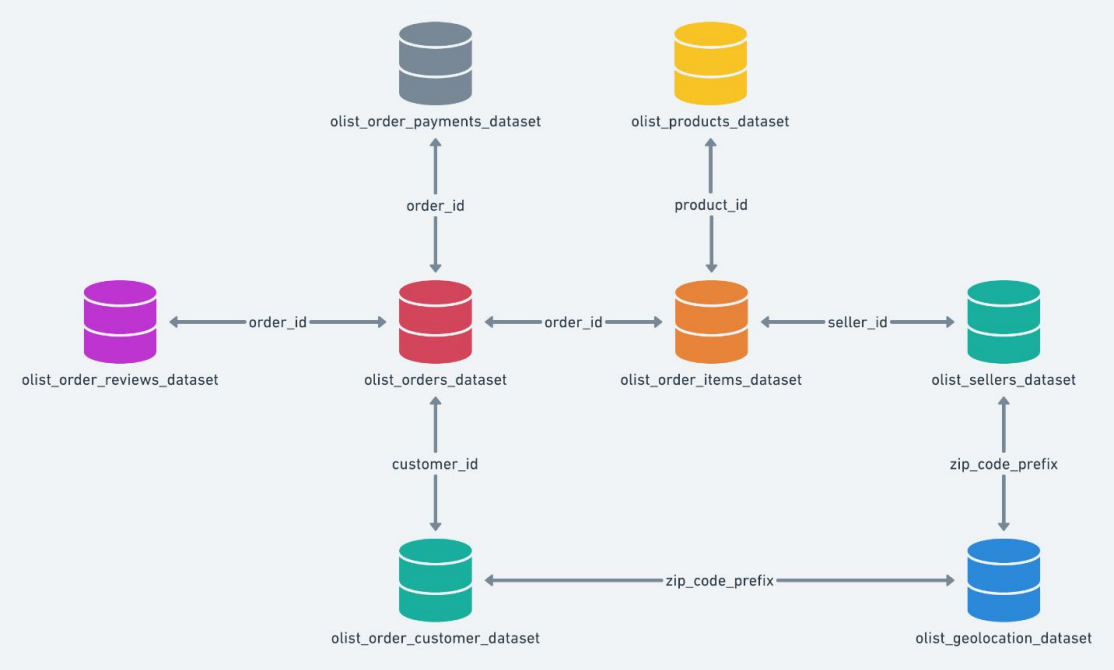
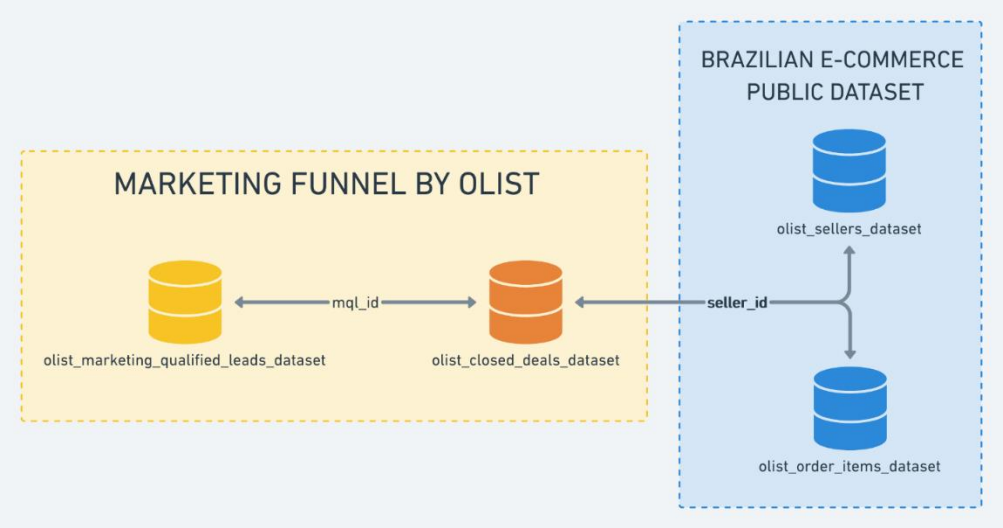


Figure 2: Data Schema for Brazilian E-Commerce Public Dataset



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.