Ecommerce transaction management application
Information Management Project – Group 4

By:
Nittala Venkata Sai Aditya (vn5227), Aishwarya Sarkar(as99646), Kai Zhang (kz3967), Rajshree Mishra (rm62528), Prathmesh Savale (ps33296), Yvonne Wang (yw22547)

OBJECTIVE
The objective is to review the current data management design and come up with a new and scalable design for a Brazilian E-commerce site. The driving factor is the ability to handle transaction management systems and handle real time updates flowing into the database. We used Google Cloud Platform for doing our ETL processes and Elastic Search for NoSql infrastructure.

1. Team members and the task distribution among the team

Nittala Venkata Sai Aditya : Data Warehouse and ETL
Aishwarya Sarkar : DDL Statements, Transaction Management System Design, Data Analysis
Kai Zhang: Data Strategy, data transformations and creating presentation
Rajshree Mishra: Data Analysis, Transaction Management System Design
Prathmesh Savale: Data Lake setup, Modeling and data analysis
Yvonne Wang : Transaction Management System Designs

2. What was the discussion among the team before the actual topic/company was selected? Do touch upon what background study was done.

Since most of us had taken marketing and supply chain electives for our course and majority of the course material from these electives focused on case studies from retail, we decided to use a retail dataset. This helped us generate better insights and also simplified the task of creating the components of the transaction management system used in retail/ecommerce**.**

Furthermore, e-commerce dataset was publicly available on Kaggle - https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce so we decided to use it to generate insights on real world data.

3. Transaction model: What is the source of this data? What type of questions do you wish to address from this raw data?

This was a public dataset found on Kaggle.

Our aim was to build 3 efficient transactional management applications on top of this data. We also wanted to generate data driven insights about payment information, customer demographics, top and low selling product lines, delivery times and delays and customer reviews.

4. Is this data being transformed into some form of a warehouse? If not, do indicate how this transaction data will be made available for answering some of the questions identified in the above point.

Initially we divided the entire data system into 3 transaction management applications - order processing system, payment processing system and the customer and seller information management system.

In our case, we performed an ETL process on the data from these three systems to combine them, perform the necessary transformations and then load it into the data warehouse. We extracted the data from the three systems, combined all the relevant order information into one table and the buyer and seller information into another. Storing all the order related information in one table is optimal since this allows us to run analysis with the minimum number of joins. This also allows us to store the aggregated data from the transactions. The data in the warehouse is now available for running complex queries and extracting insights.

5. What subset of the data will be siphoned off for analysis at something like a data lake level? How will this serve as a MVOT from the SSOT of the above point?

Since data lake is agnostic and can store any form of data (structured or unstructured) we can potentially siphon any part of our ecommerce data. The data that is available to us is in the form of relational data and comes in flat csv files that were then loaded into the data warehouse as tables. We decided to denormalize these tables and pipe only the information related to product and payments into the datalake. This replica piped into the datalake serves as an alternative way to generate insights on the payments and product data.

Consider too the way that our ecommerce company's marketing and accounts departments might produce reports on marketing spend based on the payments and product information. The marketing department, focusing on the effectiveness of its campaigns, reports spending figures collected just after ads have aired. The accounts department, focusing instead on the organization's cash flow, reports spending figures as invoices are paid. The figures will be different, but neither is incorrect. They're just altered to fit the requirements of the separate departments. In short, SSoT works best at the data level, where the pure, raw data should be clinically managed and harvested. But once an area of an organization needs to set that data in context – to transform it into information – MVoT should come into play to create a more practical version of the truth.

6.How are the answers to the question in #3 above being obtained from the analysis ?
The answer of question in #3 are being obtained by querying the data in oracle and GCP and the query result was further visualized in google looker studio as well as excel. These are all covered in Q8.

7. What did you learn from the final project? What was most valuable? How can you use this learning going forward?

This final project provided immense learning opportunities for the team as not all team members had hands-on experience with data warehousing, ETL and querying. We also learned to use LucidChart for our ER diagrams. We also learnt how to build a reporting layer on top of the data warehouse layer. This project was an excellent example to showcase how large organizations maintain high data integrity.
Moreover, we got to work on Oracle Cloud, Elastic Search, Kibana and Docker. This experience added the most value to our experience as we got to explore the concepts of data lake and learn how to store and analyze such data.
The learnings from this project would be of great help in the industry as the real world use case gave us a glimpse of the kind of insights we can derive from data and use them to make business transforming decisions. The technical aspects of the project have also added to our arsenal of data management skills.

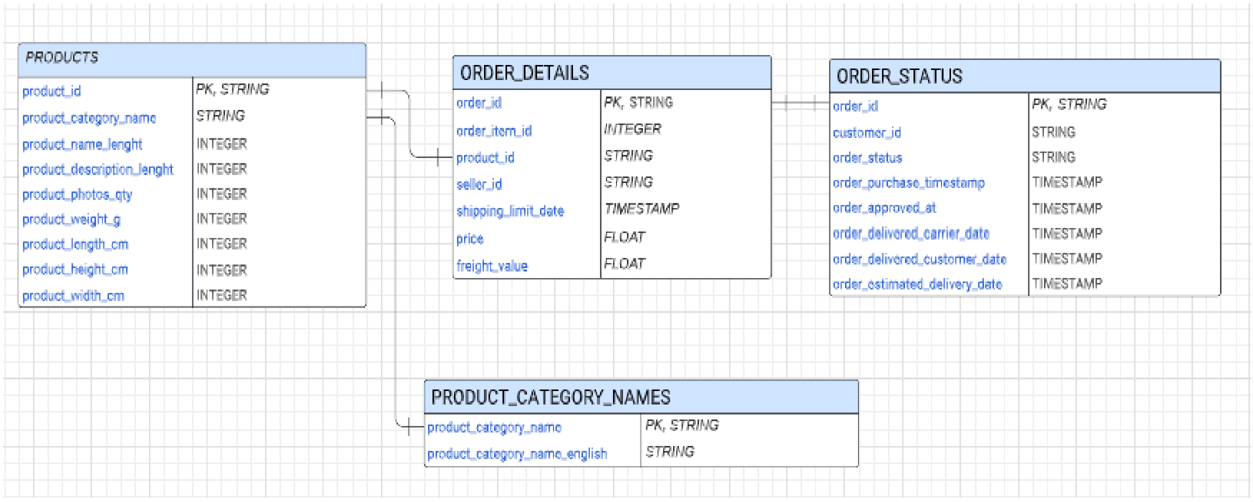
8. Give screenshots of every step showing parts of the scripts/programs as well as the results showcasing the work you did with clear explanations of what each screenshot is displaying

The whole project was divided into the below parts:

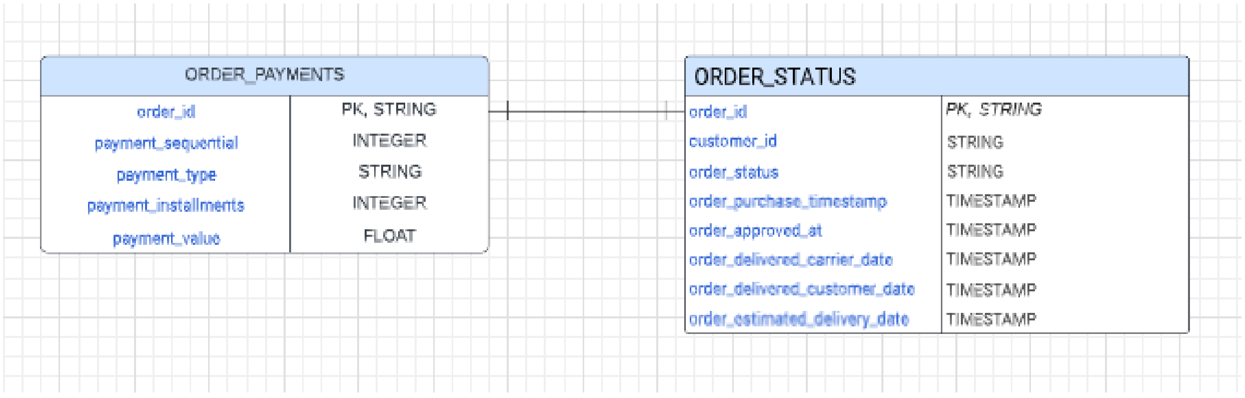
a. Data Models

The database management system is divided into 4 data models with one of them being the design of the data warehouse. Present below are the ER diagrams of the 3 transactional management applications and the data warehouse.

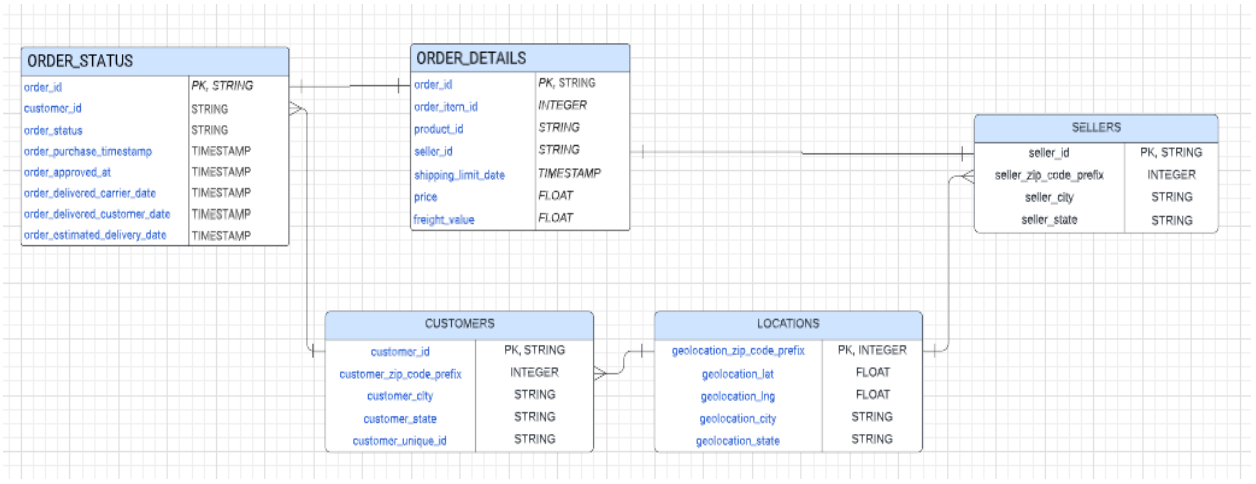
- **<u>Order Processing ER Diagram</u>**


PRODUCTS
product_id PK, STRING
product_category_name STRING
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
ORDER_DETAILS
order_id PK, STRING
order_item_id INTEGER
product_id STRING
seller_id STRING
shipping_limit_date TIMESTAMP
price FLOAT
freight_value FLOAT
ORDER_STATUS
order_id PK, STRING
customer_id STRING
order_status STRING
order_purchase_timestamp TIMESTAMP
order_approved_at TIMESTAMP
order_delivered_carrier_date TIMESTAMP
order_delivered_customer_date TIMESTAMP
order_estimated_delivery_date TIMESTAMP
PRODUCT_CATEGORY_NAMES
product_category_name PK, STRING
product_category_name_english STRING

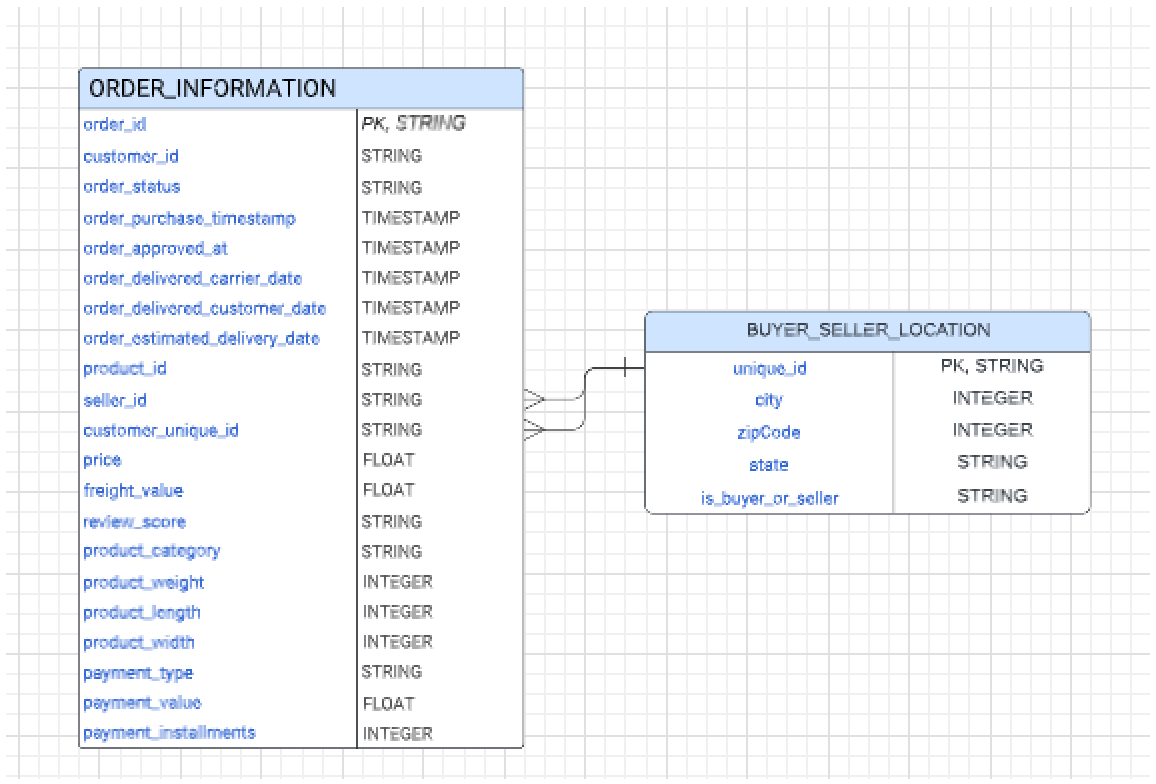
- **Payment Processing ER Diagram**


ORDER_PAYMENTS
order_id PK, STRING
payment_sequential INTEGER
payment_type STRING
payment_installments INTEGER
payment_value FLOAT
ORDER_STATUS
order_id PK, STRING
customer_id STRING
order_status STRING
order_purchase_timestamp TIMESTAMP
order_approved_at TIMESTAMP
order_delivered_carrier_date TIMESTAMP
order_delivered_customer_date TIMESTAMP
order_estimated_delivery_date TIMESTAMP

- **Buyer and Seller Processing ER Diagram**



- **Data Warehouse ER diagram**



ORDER_INFORMATION	
order_id	PK, STRING
customer_id	STRING
order_status	STRING
order_purchase_timestamp	TIMESTAMP
order_approved_at	TIMESTAMP
order_delivered_carrier_date	TIMESTAMP
order_delivered_customer_date	TIMESTAMP
order_estimated_delivery_date	TIMESTAMP
product_id	STRING
seller_id	STRING
customer_unique_id	STRING
price	FLOAT
freight_value	FLOAT
review_score	STRING
product_category	STRING
product_weight	INTEGER
product_length	INTEGER
product_width	INTEGER
payment_type	STRING
payment_value	FLOAT
payment_installments	INTEGER

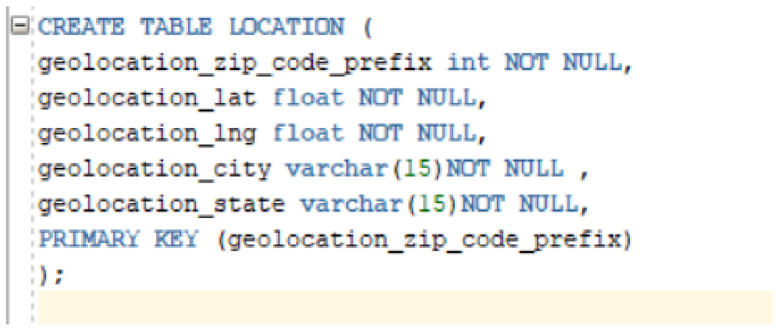
BUYER_SELLER_LOCATION	
unique_id	PK, STRING
city	INTEGER
zipCode	INTEGER
state	STRING
is_buyer_or_seller	STRING

b. **DDL for Data Models**:

- **<u>BUYER AND SELLER PROCESSING SYSTEM</u>**

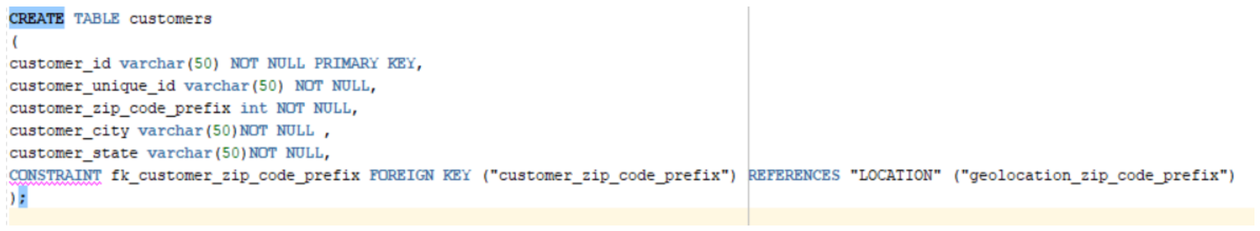
<u>LOCATION:</u>

This table stores the location data corresponding to a particular zip code.


```
CREATE TABLE LOCATION (
geolocation_zip_code_prefix int NOT NULL,
geolocation_lat float NOT NULL,
geolocation_lng float NOT NULL,
geolocation_city varchar(15)NOT NULL ,
geolocation_state varchar(15)NOT NULL,
PRIMARY KEY (geolocation_zip_code_prefix)
);
```

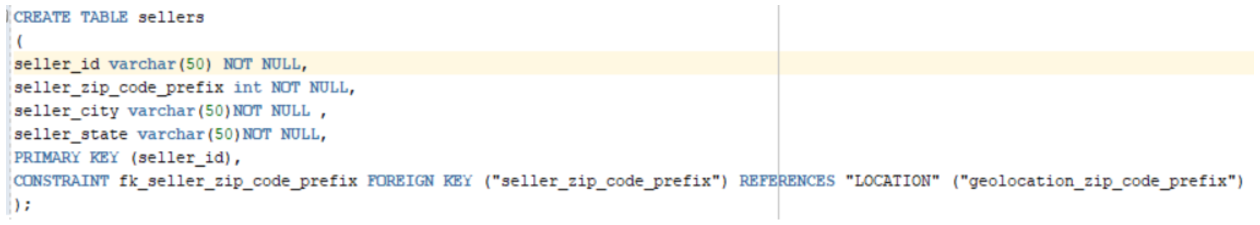
CUSTOMERS:

It stores all the customer related data


```
CREATE TABLE customers
(
customer_id varchar(50) NOT NULL PRIMARY KEY,
customer_unique_id varchar(50) NOT NULL,
customer_zip_code_prefix int NOT NULL,
customer_city varchar(50)NOT NULL ,
customer_state varchar(50)NOT NULL,
CONSTRAINT fk_customer_zip_code_prefix FOREIGN KEY ("customer_zip_code_prefix") REFERENCES "LOCATION" ("geolocation_zip_code_prefix")
);
```

SELLERS:

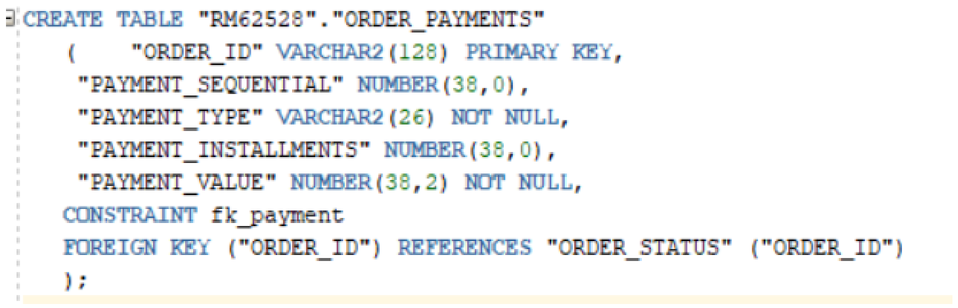
This has all the seller related data.


```
CREATE TABLE sellers
(
seller_id varchar(50) NOT NULL,
seller_zip_code_prefix int NOT NULL,
seller_city varchar(50)NOT NULL ,
seller_state varchar(50)NOT NULL,
PRIMARY KEY (seller_id),
CONSTRAINT fk_seller_zip_code_prefix FOREIGN KEY ("seller_zip_code_prefix") REFERENCES "LOCATION" ("geolocation_zip_code_prefix")
);
```

- **PAYMENT PROCESSING SYSTEM**

PAYMENTS:

Stores payment related data about the mode of payment as well as number of installments in which the payment is being made.

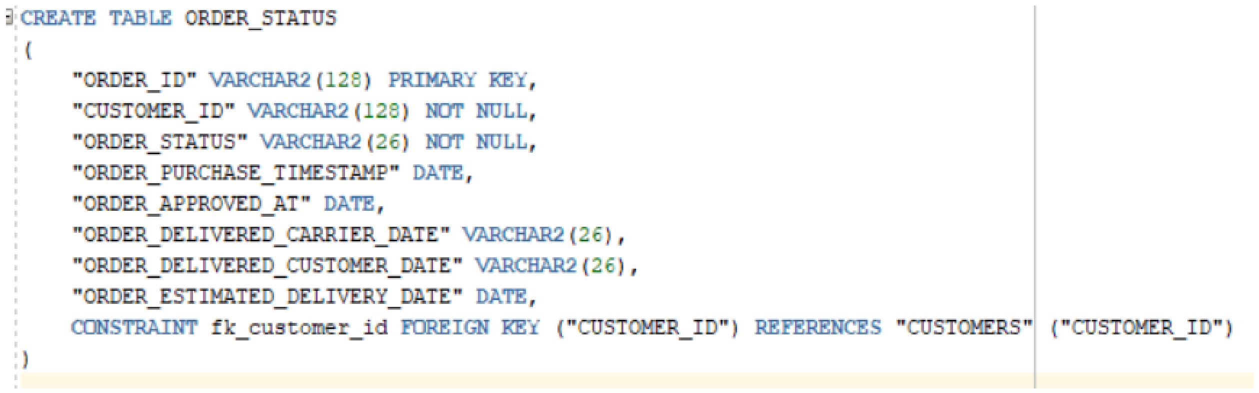


```
CREATE TABLE "RM62528"."ORDER_PAYMENTS"
   (    "ORDER_ID" VARCHAR2(128) PRIMARY KEY,
    "PAYMENT_SEQUENTIAL" NUMBER(38,0),
    "PAYMENT_TYPE" VARCHAR2(26) NOT NULL,
    "PAYMENT_INSTALLMENTS" NUMBER(38,0),
    "PAYMENT_VALUE" NUMBER(38,2) NOT NULL,
   CONSTRAINT fk_payment
   FOREIGN KEY ("ORDER_ID") REFERENCES "ORDER_STATUS" ("ORDER_ID")
   );
```

- **<u>ORDER PROCESSING SYSTEM</u>**

<u>ORDER_STATUS:</u>

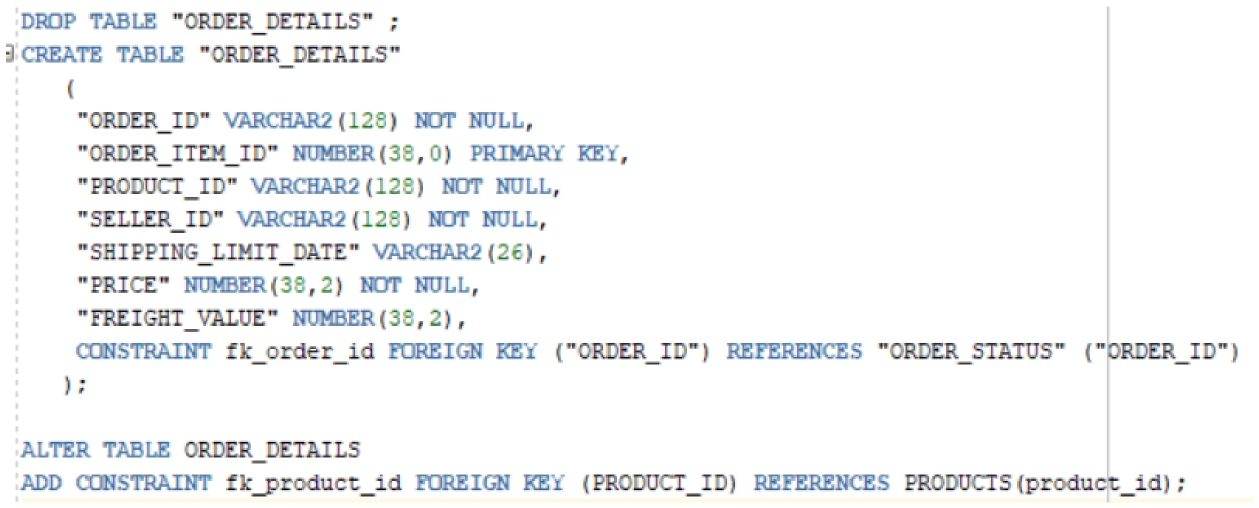
Tells us about the current status of the order.



```
CREATE TABLE ORDER_STATUS
(
    "ORDER_ID" VARCHAR2(128) PRIMARY KEY,
    "CUSTOMER_ID" VARCHAR2(128) NOT NULL,
    "ORDER_STATUS" VARCHAR2(26) NOT NULL,
    "ORDER_PURCHASE_TIMESTAMP" DATE,
    "ORDER_APPROVED_AT" DATE,
    "ORDER_DELIVERED_CARRIER_DATE" VARCHAR2(26),
    "ORDER_DELIVERED_CUSTOMER_DATE" VARCHAR2(26),
    "ORDER_ESTIMATED_DELIVERY_DATE" DATE,
    CONSTRAINT fk_customer_id FOREIGN KEY ("CUSTOMER_ID") REFERENCES "CUSTOMERS" ("CUSTOMER_ID")
)
```

<u>ORDER_DETAILS:</u>

All the order details are present in this table

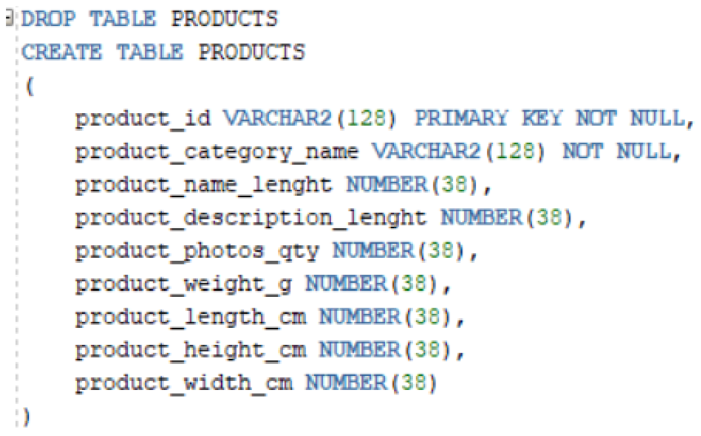


```
DROP TABLE "ORDER_DETAILS" ;
CREATE TABLE "ORDER_DETAILS"
   (
    "ORDER_ID" VARCHAR2(128) NOT NULL,
    "ORDER_ITEM_ID" NUMBER(38,0) PRIMARY KEY,
    "PRODUCT_ID" VARCHAR2(128) NOT NULL,
    "SELLER_ID" VARCHAR2(128) NOT NULL,
    "SHIPPING_LIMIT_DATE" VARCHAR2(26),
    "PRICE" NUMBER(38,2) NOT NULL,
    "FREIGHT_VALUE" NUMBER(38,2),
    CONSTRAINT fk_order_id FOREIGN KEY ("ORDER_ID") REFERENCES "ORDER_STATUS" ("ORDER_ID")
   );

ALTER TABLE ORDER_DETAILS
ADD CONSTRAINT fk_product_id FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCTS(product_id);
```

PRODUCTS:

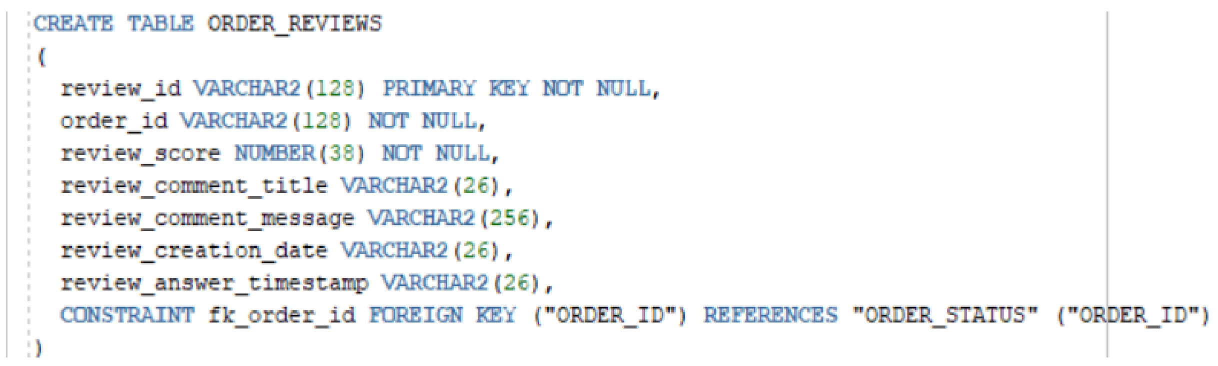
Stores information related to different product categories.



```
DROP TABLE PRODUCTS
CREATE TABLE PRODUCTS
(
    product_id VARCHAR2(128) PRIMARY KEY NOT NULL,
    product_category_name VARCHAR2(128) NOT NULL,
    product_name_lenght NUMBER(38),
    product_description_lenght NUMBER(38),
    product_photos_qty NUMBER(38),
    product_weight_g NUMBER(38),
    product_length_cm NUMBER(38),
    product_height_cm NUMBER(38),
    product_width_cm NUMBER(38)
)
```

ORDER_REVIEWS:

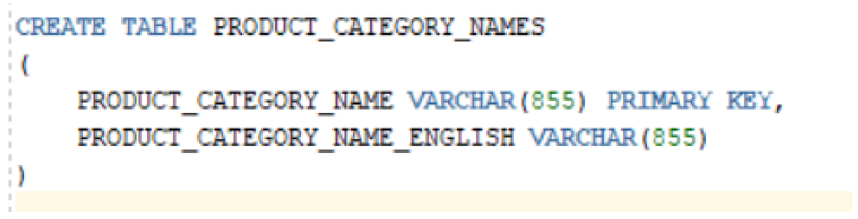
This table stores data related to customer ratings and reviews.



```
CREATE TABLE ORDER_REVIEWS
(
  review_id VARCHAR2(128) PRIMARY KEY NOT NULL,
  order_id VARCHAR2(128) NOT NULL,
  review_score NUMBER(38) NOT NULL,
  review_comment_title VARCHAR2(26),
  review_comment_message VARCHAR2(256),
  review_creation_date VARCHAR2(26),
  review_answer_timestamp VARCHAR2(26),
  CONSTRAINT fk_order_id FOREIGN KEY ("ORDER_ID") REFERENCES "ORDER_STATUS" ("ORDER_ID")
)
```

PRODUCT_CATEGORY_NAMES:

The product category names were in a different language, this tables provides a translation to English



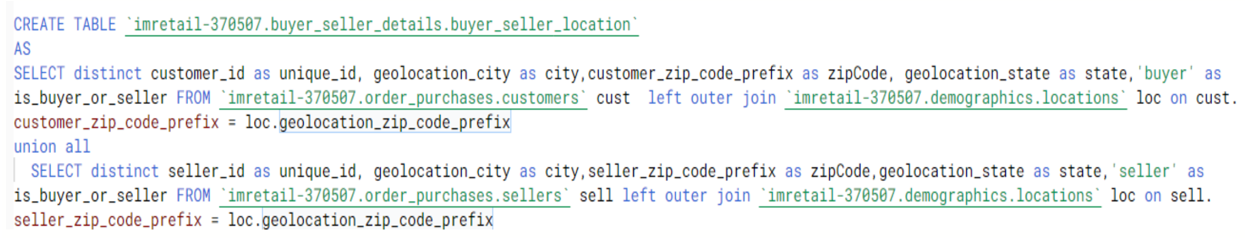
```
CREATE TABLE PRODUCT_CATEGORY_NAMES
(
    PRODUCT_CATEGORY_NAME VARCHAR(855) PRIMARY KEY,
    PRODUCT_CATEGORY_NAME_ENGLISH VARCHAR(855)
)
```

c. ETL code to populate the data models:

The data was extracted from the databases and transformed to create tables for the data warehouse and then loaded to Big Query.

BUYER_SELLER_LOCATIONS

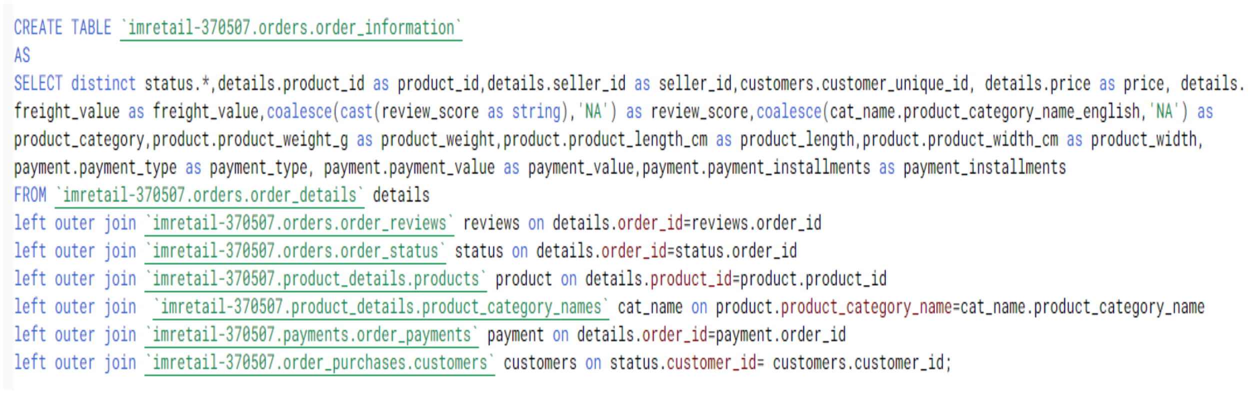
The first table consists of buyer and seller locations and if the id is that of a buyer or seller.



```
CREATE TABLE `imretail-370507.buyer_seller_details.buyer_seller_location`
AS
SELECT distinct customer_id as unique_id, geolocation_city as city,customer_zip_code_prefix as zipCode, geolocation_state as state,'buyer' as
is_buyer_or_seller FROM `imretail-370507.order_purchases.customers` cust  left outer join `imretail-370507.demographics.locations` loc on cust.
customer_zip_code_prefix = loc.geolocation_zip_code_prefix
union all
  SELECT distinct seller_id as unique_id, geolocation_city as city,seller_zip_code_prefix as zipCode,geolocation_state as state,'seller' as
is_buyer_or_seller FROM `imretail-370507.order_purchases.sellers` sell left outer join `imretail-370507.demographics.locations` loc on sell.
seller_zip_code_prefix = loc.geolocation_zip_code_prefix
```

ORDER_INFORMATION

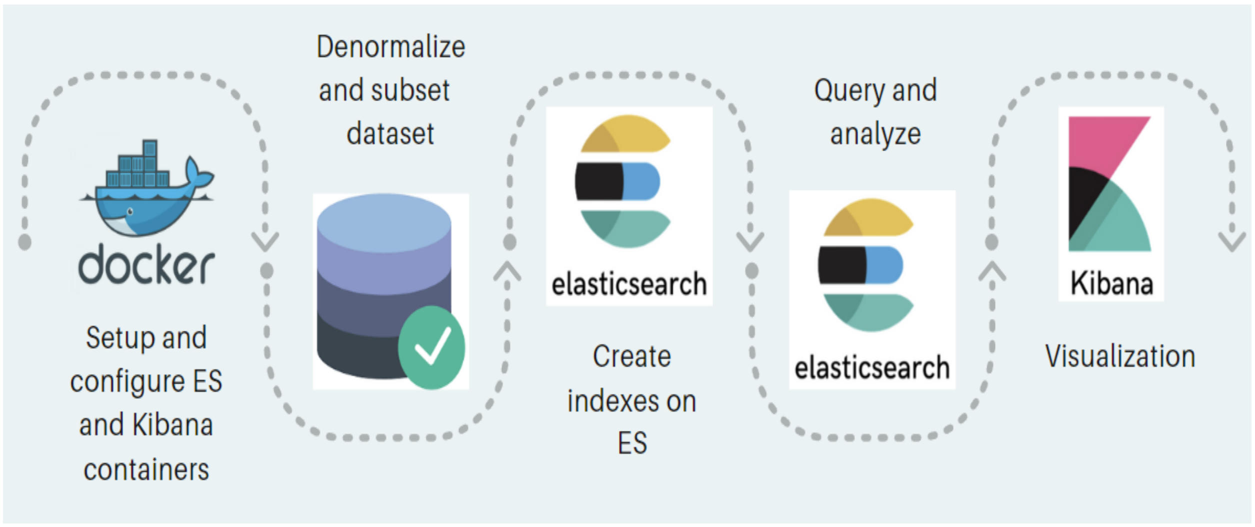
The second table consists of an aggregated view of the orders and the status and it's payment type. This helps in getting a holistic picture of the whole orders data.



```
CREATE TABLE `imretail-370507.orders.order_information`
AS
SELECT distinct status.*,details.product_id as product_id,details.seller_id as seller_id,customers.customer_unique_id, details.price as price, details.
freight_value as freight_value,coalesce(cast(review_score as string),'NA') as review_score,coalesce(cat_name.product_category_name_english,'NA') as
product_category,product.product_weight_g as product_weight,product.product_length_cm as product_length,product.product_width_cm as product_width,
payment.payment_type as payment_type, payment.payment_value as payment_value,payment.payment_installments as payment_installments
FROM `imretail-370507.orders.order_details` details
left outer join `imretail-370507.orders.order_reviews` reviews on details.order_id=reviews.order_id
left outer join `imretail-370507.orders.order_status` status on details.order_id=status.order_id
left outer join `imretail-370507.product_details.products` product on details.product_id=product.product_id
left outer join  `imretail-370507.product_details.product_category_names` cat_name on product.product_category_name=cat_name.product_category_name
left outer join `imretail-370507.payments.order_payments` payment on details.order_id=payment.order_id
left outer join `imretail-370507.order_purchases.customers` customers on status.customer_id= customers.customer_id;
```

d. The model for the data lake:

We implemented a NoSql Data Lake model for our project. We have used Elastic Search as our No SQL database and we deployed it on our local We configure a single cluster to perform all operations and we use Kibana as the visualization tool which can query Elastic Search to create visualizations for insight generation.



The database setup process follows 5 steps:

1. Setup ES and Kibana docker containers and map ports for the application to host ports
2. Denormalize the transactional data so that we have documents/records for each order
3. Create required indexes in ES and upload the denormalized data
4. Query required dimensions and metrics for analysis
5. Create visuals on Kibana

Model for datalake:

List of Indexes in datalake:
> GET /_aliases?pretty=true

```
{
 ".kibana_1" : {
   "aliases" : {
     ".kibana" : { }
   }
 },
 ".kibana_task_manager" : {
   "aliases" : { }
 },
 "geolocation_data" : {
   "aliases" : { }
 },
 "olist_data_updated_ts" : {
   "aliases" : { }
 },
 "olist_orders" : {
   "aliases" : { }
 },
 "olist_orders_dt_updated" : {
   "aliases" : { }
 },
 "olist_orders_info" : {
   "aliases" : { }
 },
 "olist_orders_info_es" : {
   "aliases" : { }
 }
}
```

For *olist_orders_dt_updated* Index:
> GET /olist_orders_dt_updated/_mapping

```
{
 "olist_orders_dt_updated" : {
   "mappings" : {
     "_doc" : {
       "_meta" : {
         "created_by" : "ml-file-data-visualizer"
       },
       "properties" : {
         "@timestamp" : {
           "type" : "date"
         },
         "column1" : {
           "type" : "long"
         },
         "customer_city" : {
           "type" : "keyword"
         },
         "customer_id" : {
           "type" : "keyword"
         },
         "customer_state" : {
           "type" : "keyword"
         },
         "customer_unique_id" : {
           "type" : "keyword"
         },
         "customer_zip_code_prefix" : {
           "type" : "long"
         },
         "freight_value" : {
           "type" : "double"
         },
         "order_approved_at" : {
           "type" : "date",
           "format" : "8yyyy-MM-dd HH:mm:ss"
         },
         "order_delivered_carrier_date" : {
           "type" : "date",
           "format" : "8yyyy-MM-dd HH:mm:ss"
```

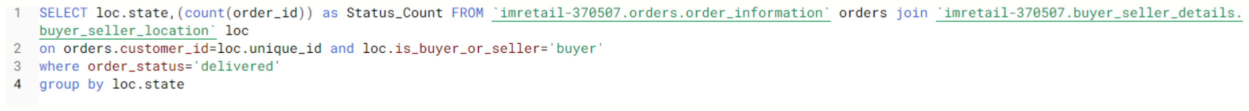
```
        },
        "order_delivered_customer_date" : {
          "type" : "date",
          "format" : "8yyyy-MM-dd HH:mm:ss"
        },
        "order_estimated_delivery_date" : {
          "type" : "keyword"
        },
        "order_id" : {
          "type" : "keyword"
        },
        "order_item_id" : {
          "type" : "double"
        },
        "order_purchase_timestamp" : {
          "type" : "date",
          "format" : "8yyyy-MM-dd HH:mm:ss"
        },
        "order_status" : {
          "type" : "keyword"
        },
        "payment_installments" : {
          "type" : "double"
        },
        "payment_sequential" : {
          "type" : "double"
        },
        "payment_type" : {
          "type" : "keyword"
        },
        "payment_value" : {
          "type" : "double"
        },
        "price" : {
          "type" : "double"
        },
        "product_category_name" : {
          "type" : "keyword"
        },
        "product_category_name_english" : {
          "type" : "keyword"
        },
        "product_description_lenght" : {
          "type" : "double"
        },
        "product_height_cm" : {
          "type" : "double"
        },
        "product_id" : {
          "type" : "keyword"
        },
        "product_length_cm" : {
          "type" : "double"
        },
        "product_name_lenght" : {
          "type" : "double"
        },
        "product_photos_qty" : {
          "type" : "double"
        },
        "product_weight_g" : {
          "type" : "double"
        },
        "product_width_cm" : {
          "type" : "double"
        },
        "review_answer_timestamp" : {
          "type" : "date",
          "format" : "8yyyy-MM-dd HH:mm:ss"
        },
        "review_comment_message" : {
          "type" : "text"
        },
        "review_comment_title" : {
          "type" : "keyword"
        },
        "review_creation_date" : {
          "type" : "keyword"
        },
        "review_id" : {
          "type" : "keyword"
        },
        "review_score" : {
          "type" : "double"
        },
        "seller_city" : {
          "type" : "keyword"
        },
        "seller_id" : {
          "type" : "keyword"
        },
        "seller_state" : {
          "type" : "keyword"
        },
        "seller_zip_code_prefix" : {
          "type" : "double"
        },
        "shipping_limit_date" : {
          "type" : "date",
          "format" : "8yyyy-MM-dd HH:mm:ss"
        }
      }
    }
  }
 }
}
```

Example query - Popular products across price bands (visual on dashboard) - data request:

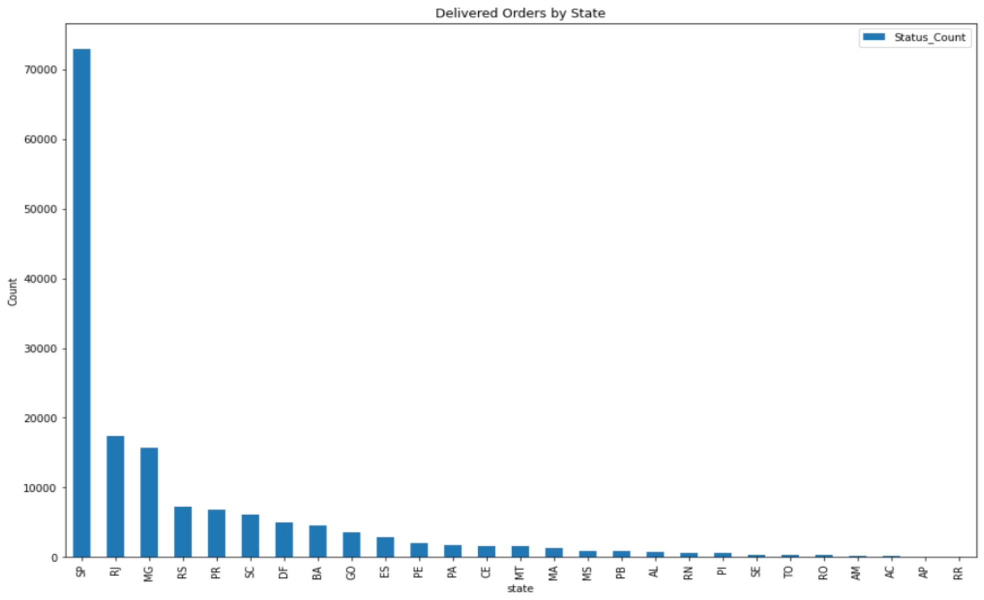
```
GET
{
 "aggs": {
  "2": {
   "terms": {
    "field": "product_category_name_english",
      "size": 10,
      "order": {
        "1": "desc"
      }
    },
    "aggs": {
      "1": {
        "cardinality": {
          "field": "order_id"
        }
      },
      "3": {
        "range": {
          "field": "payment_value",
          "ranges": [
            {
              "from": 0,
              "to": 50
            },
            {
              "from": 51,
              "to": 100
            },
            {
              "from": 101,
              "to": 150
            },
            {
              "from": 151,
              "to": 200
            },
            {
              "from": 200
            }
          ],
          "keyed": true
        },
        "aggs": {
          "1": {
            "cardinality": {
              "field": "order_id"
            }
          }
        }
      }
    }
  }
},
"size": 0,
"_source": {
  "excludes": []
},
"stored_fields": [
  "*"
],
"script_fields": {
  "order_delivered_carrier_date_dt": {
    "script": {
      "source": "new SimpleDateFormat('MM/DD/YYYY
HH:MM').parse(doc['order_delivered_carrier_date'].value);",
      "lang": "painless"
    }
  }
},
"docvalue_fields": [],
"query": {
  "bool": {
    "must": [],
    "filter": [
      {
        "match_all": {}
      },
      {
        "match_all": {}
      }
    ],
    "should": [],
    "must_not": []
  }
 }
}
```

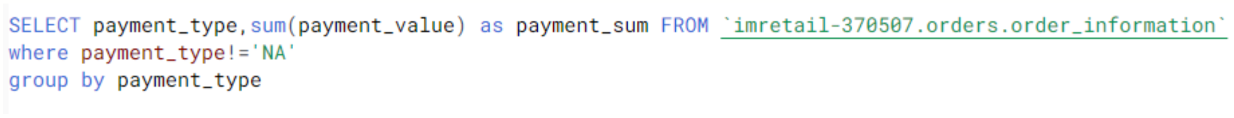
e. **Data Analysis**:

The title of the graphs provide information about what the graph shows:

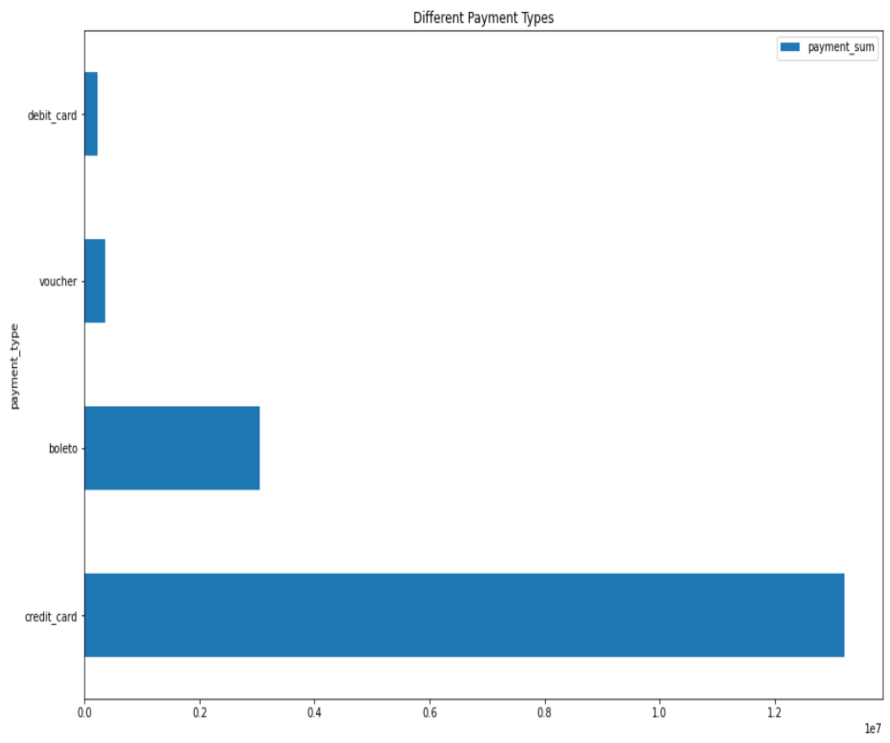


```sql
SELECT loc.state,(count(order_id)) as Status_Count FROM `imretail-370507.orders.order_information` orders join `imretail-370507.buyer_seller_details.buyer_seller_location` loc
on orders.customer_id=loc.unique_id and loc.is_buyer_or_seller='buyer'
where order_status='delivered'
group by loc.state
```

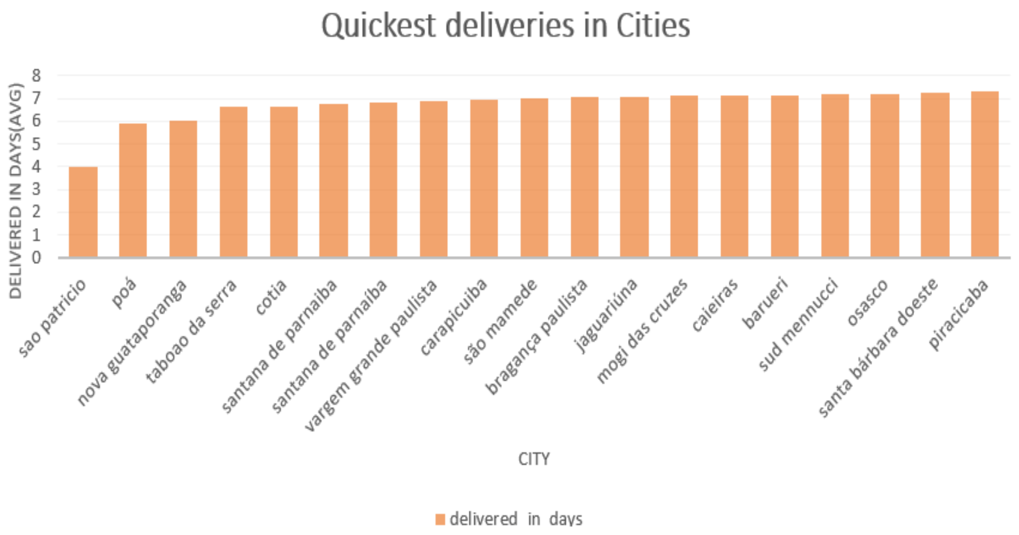


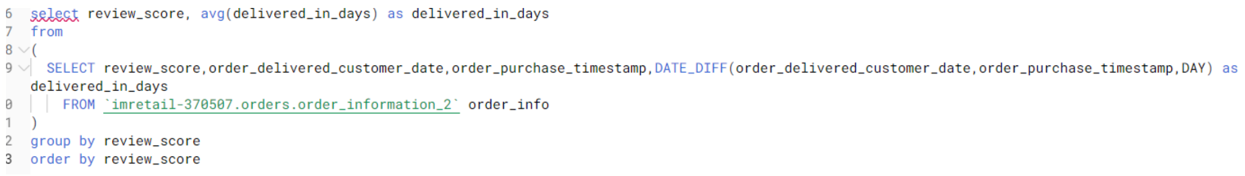


```sql
SELECT payment_type,sum(payment_value) as payment_sum FROM `imretail-370507.orders.order_information`
where payment_type!='NA'
group by payment_type
```

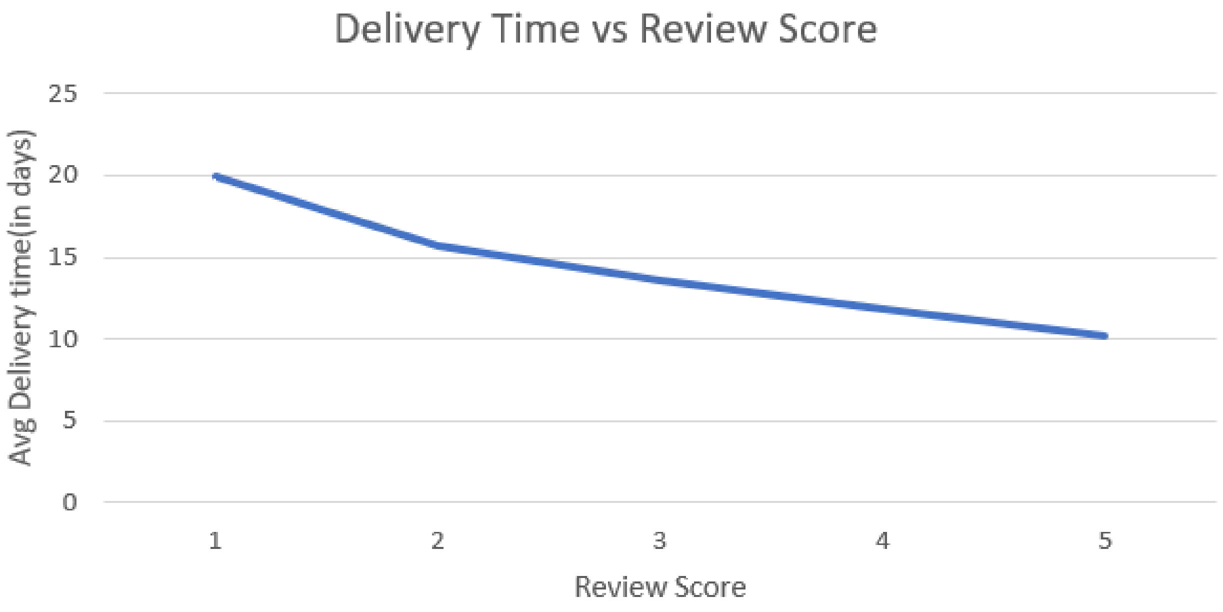


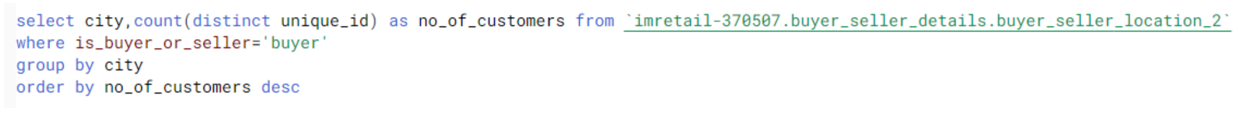
```
select city, avg(delivered_in_days) as delivered_in_days
from
(
  SELECT city,order_delivered_customer_date,order_purchase_timestamp,DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) as
delivered_in_days
  FROM `imretail-370507.orders.order_information_2` order_info
  LEFT JOIN `imretail-370507.buyer_seller_details.buyer_seller_location_2` bseller on order_info.customer_unique_id= bseller.unique_id
  where
  order_delivered_customer_date is not null and
  bseller.city is not null
)
group by city
```



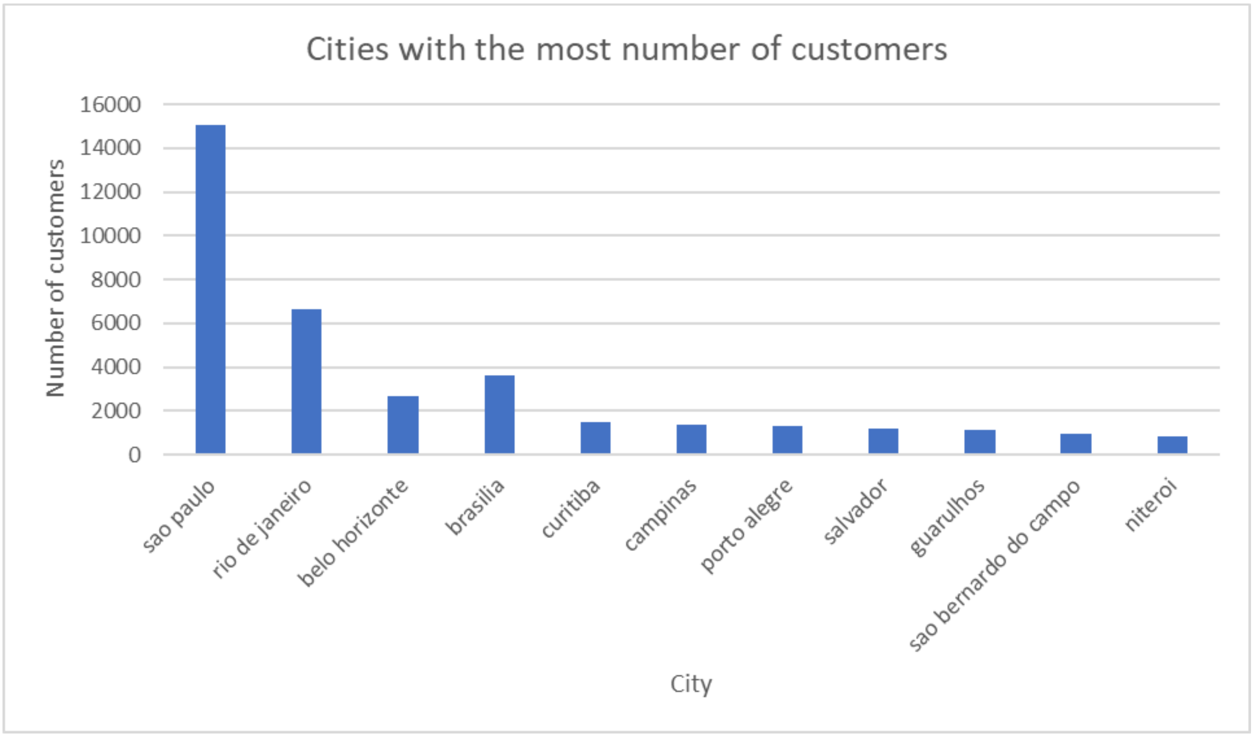


```sql
select review_score, avg(delivered_in_days) as delivered_in_days
from
(
  SELECT review_score,order_delivered_customer_date,order_purchase_timestamp,DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) as delivered_in_days
    FROM `imretail-370507.orders.order_information_2` order_info
)
group by review_score
order by review_score
```



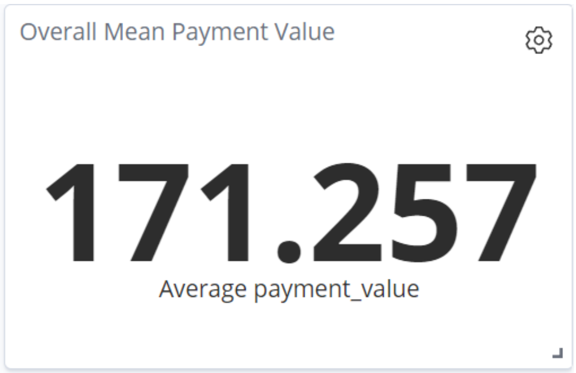


```sql
select city,count(distinct unique_id) as no_of_customers from `imretail-370507.buyer_seller_details.buyer_seller_location_2`
where is_buyer_or_seller='buyer'
group by city
order by no_of_customers desc
```



Code for Elastic Search Queries can be access on this gist -
https://gist.github.com/praths007/0eae9d75bde26131025d55a55082cae2

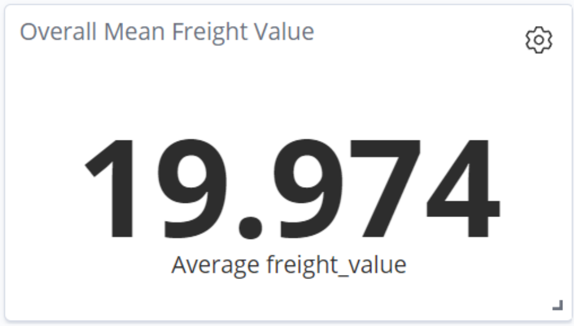
Insights generated from Data Lake:



Overall Mean Payment Value

171.257

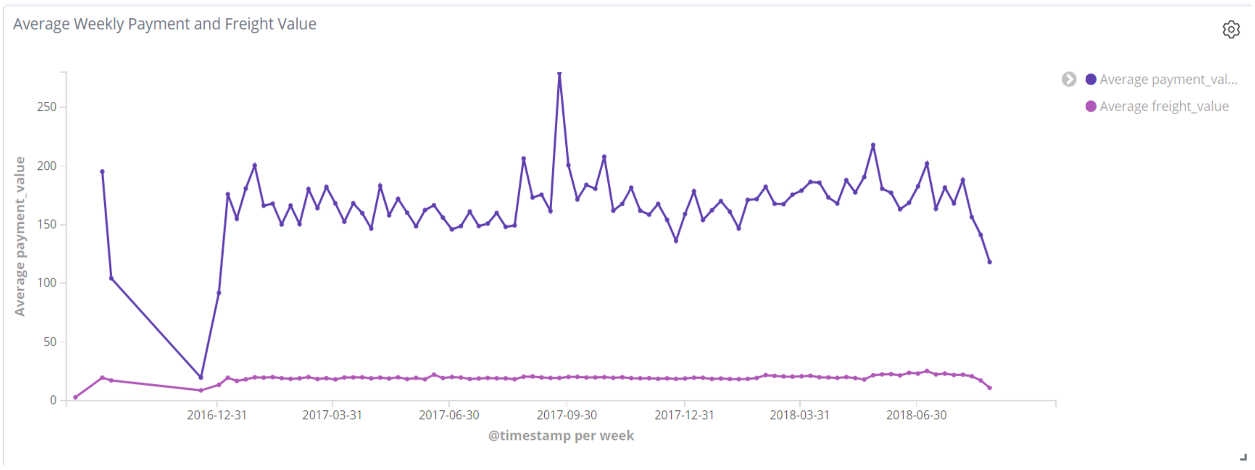
Average payment_value



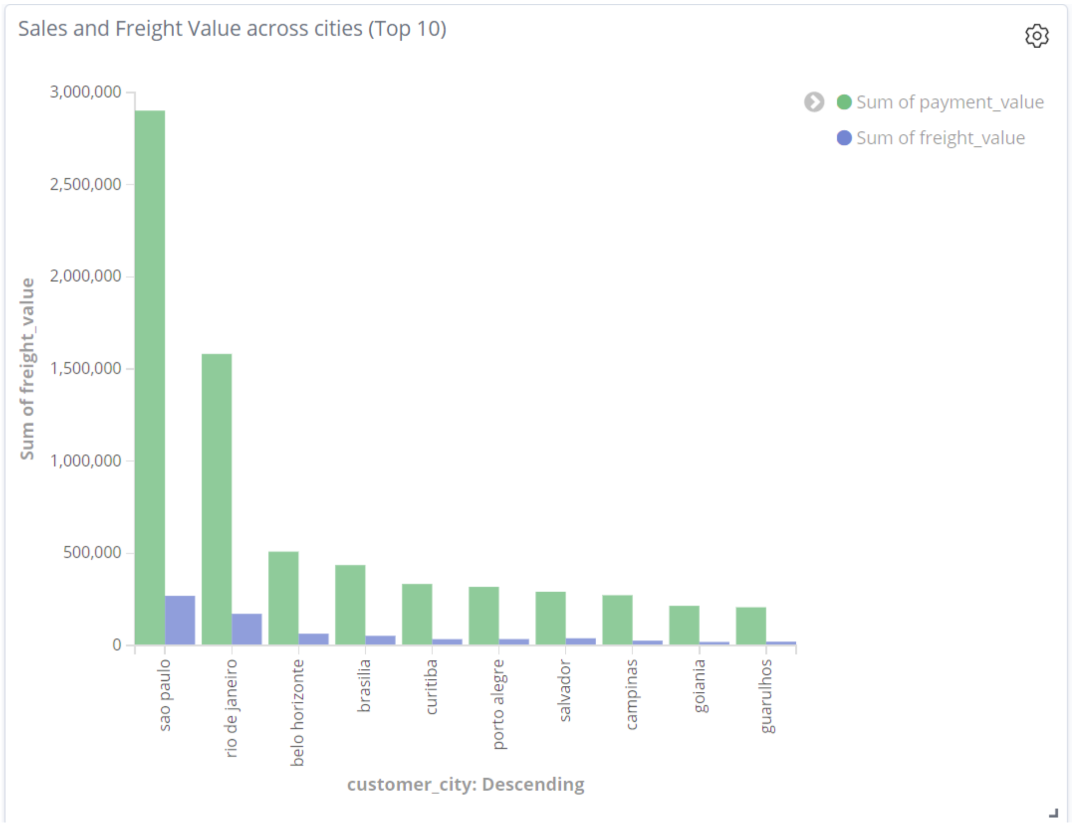
Overall Mean Freight Value

19.974

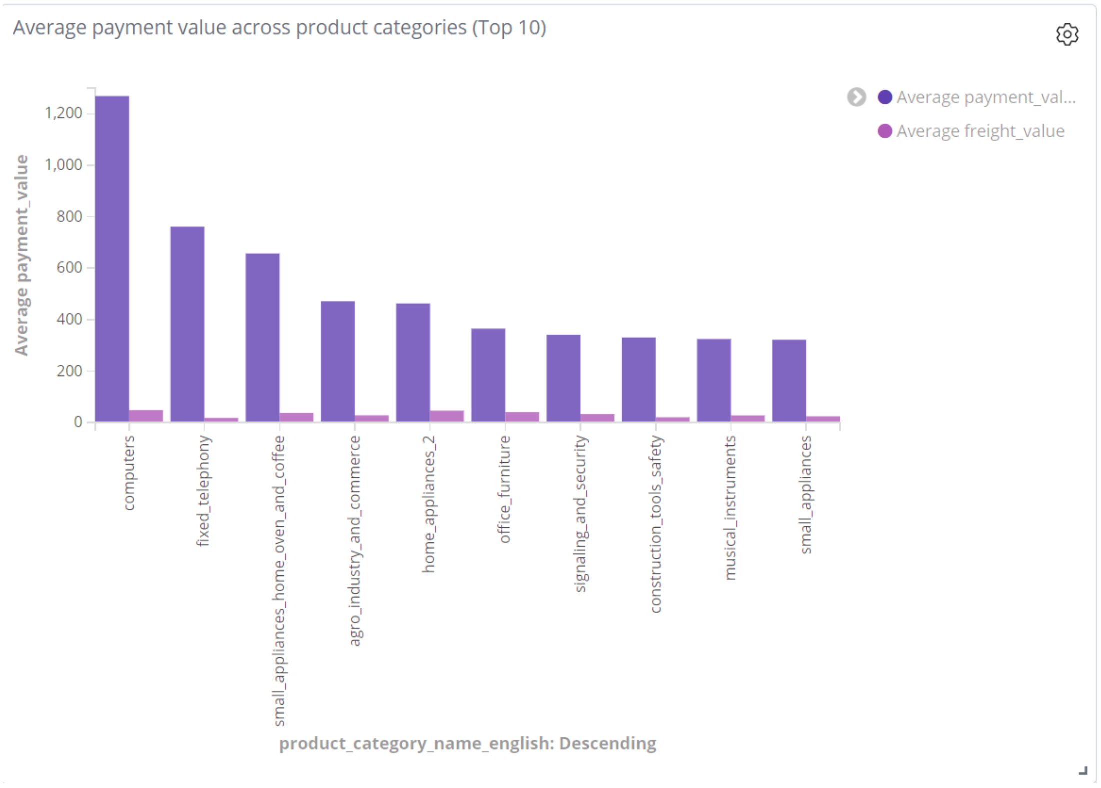
Average freight_value

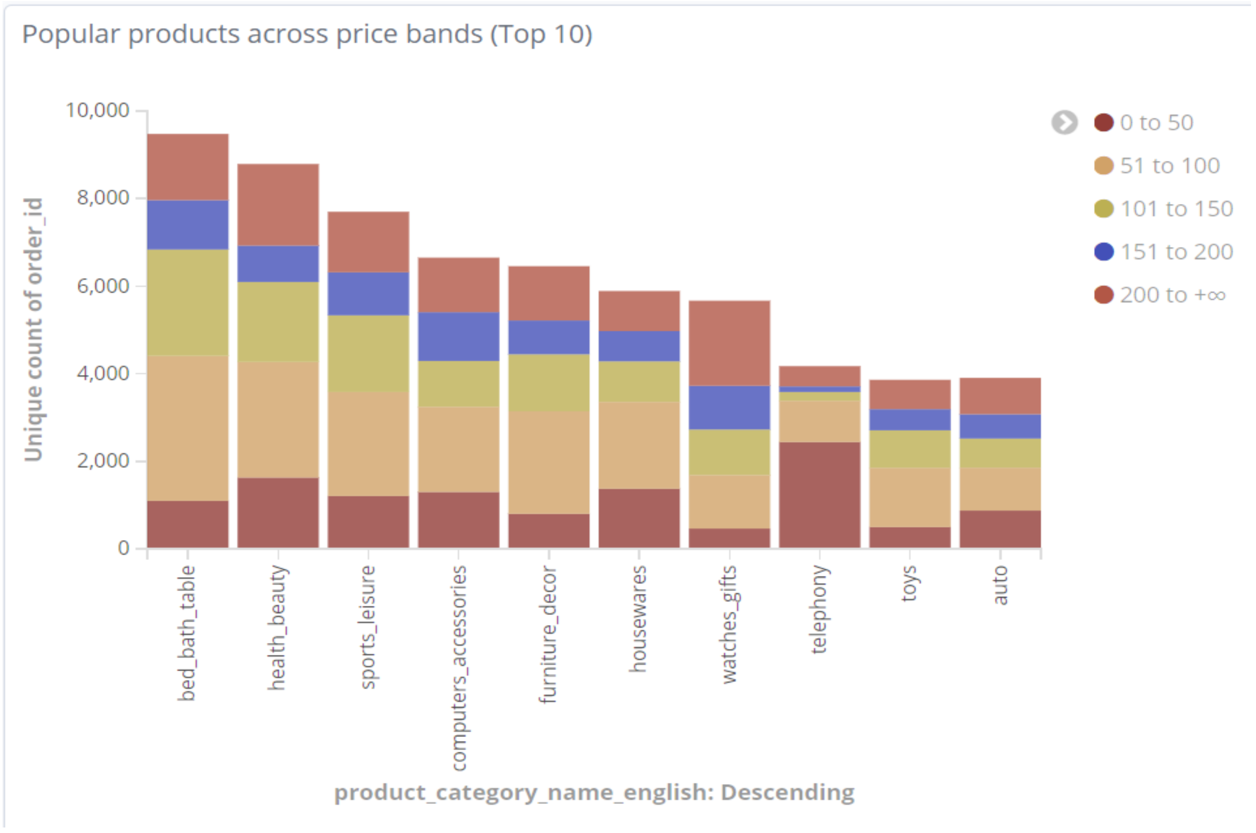


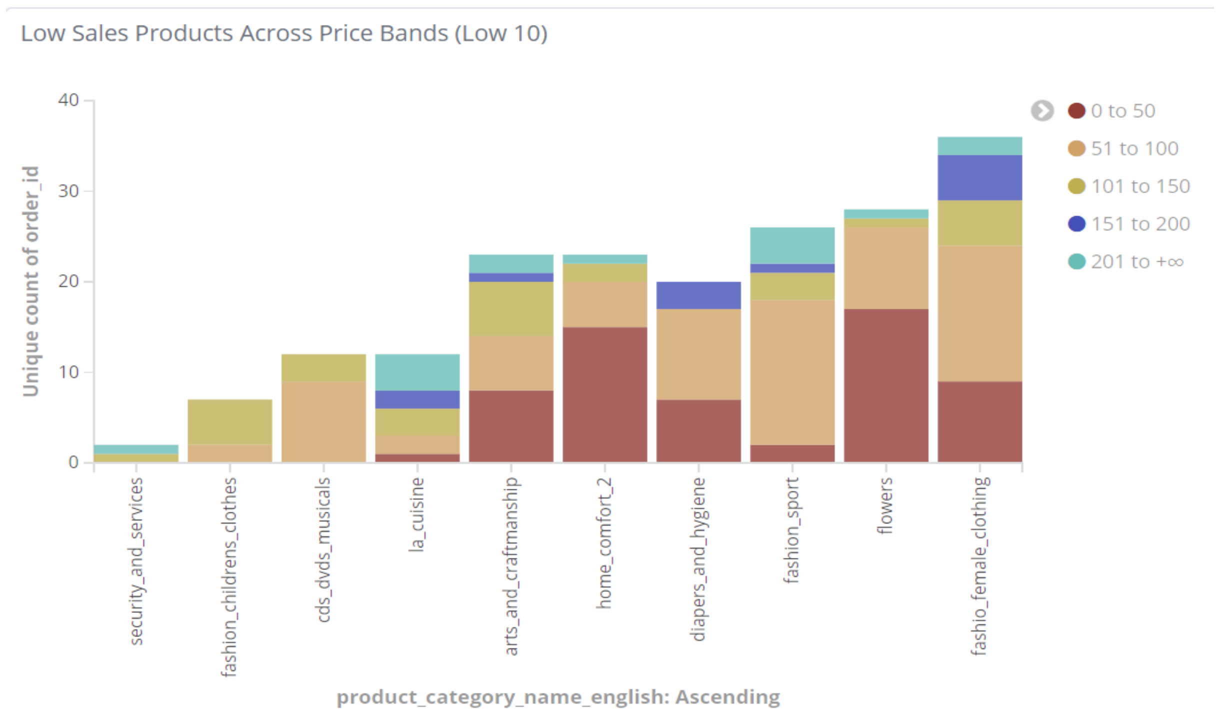
Average Weekly Payment and Freight Value
Average payment_value
250
200
150
100
50
0
2016-12-31
2017-03-31
2017-06-30
2017-09-30
2017-12-31
2018-03-31
2018-06-30
@timestamp per week
Average payment_val...
Average freight_value



Sales and Freight Value across cities (Top 10)
Sum of freight_value
3,000,000
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
sao paulo
rio de janeiro
belo horizonte
brasilia
curitiba
porto alegre
salvador
campinas
goiania
guarulhos
customer_city: Descending
Sum of payment_value
Sum of freight_value


Average payment value across product categories (Top 10)
Average payment_val...
Average freight_value
Average payment_value
1,200
1,000
800
600
400
200
0
computers
fixed_telephony
small_appliances_home_oven_and_coffee
agro_industry_and_commerce
home_appliances_2
office_furniture
signaling_and_security
construction_tools_safety
musical_instruments
small_appliances
product_category_name_english: Descending


Popular products across price bands (Top 10)
0 to 50
51 to 100
101 to 150
151 to 200
200 to +∞
Unique count of order_id
10,000
8,000
6,000
4,000
2,000
0
bed_bath_table
health_beauty
sports_leisure
computers_accessories
furniture_decor
housewares
watches_gifts
telephony
toys
auto
product_category_name_english: Descending



<u>Summary of Insights generated from Data Warehouse as well as data lake</u>:

- The average payment value was 171.26 real, the mean freight value was about 20 real
- The most popular payment method is credit card
- The payment and freight value trends are stable
- São Paulo and Rio De Janeiro are the business hotspots
- Freight value is proportional to the item type – bigger items (in terms of volume) have higher freight value
- Computers are the most expensive commodity sold on the website and its' payment has the highest average payment installment as well
- Bed bath table, health and beauty are the most popular categories
- Watches and gifts have the higher price band products
- Children and women's clothes show lower sales volume with less price bands
- Our top-rated product category is children clothing, whereas the lowest rated was security equipment
- Computers and small home appliances are the costliest categories