Online marketplace data cleaning

Salah satu tahap dari teknik pengelola data dalam bidang *data scientists* adalah melakukan persiapan data dengan cara melakukan *data cleaning* dan *data wrangling*, *data cleaning* ada pemrosesan data oleh seorang *data scientists* dengan cara melakukan normalisasi *raw data* dari sumber data seperti *database*, data hasil *scraping* dan lain-lain sehingga menghindari terjadinya kesalahan analisis selanjutnya. Proses yang dilakukan saat dilakukan *data cleaning* antara lain memfilter data null atau kosong, menghapus *outlier*, mengelola data yang inkonsisten dan menghapus *data duplicate* dari data yang akan di analisis.

Untuk proses *data wrangling* sendiri adalah proses mengatur ulang data menjadi bentuk yang lebih terstruktur sehingga sesuai dengan permasalahan yang akan di eksplorasi oleh seorang data scientists, proses ini mencakup melakukan agregasi data, manipulasi data dan melakukan transformasi waktu.

Dataset:

1. Olist customers
    a. customer_id : nilai unik yang dipakai dalam dataset untuk menghubungkan ke dataset yang lain.
    b. Customer_uniq_id: nilai untuk mengidentifikasi pelanggan.
    c. Customer_zip_code_prefix: 5 digit pertama dari zip code customer
    d. Customer_city: nama kota kostumer
    e. Customer_state : nama negara bagian kostumer
2. Olist geolocation
    a. Geolocation_zip_code_prefix: 5 digit pertama zip code
    b. Geolocation_lat: nilai latitude
    c. Geolocation_lng: nilai longtitude
    d. Geolocation_city : nama kota
    e. Geolocation _state: negara bagian
3. Olist order items:
    a. Order_id; nomer unik pesanan
    b. Order_item_id: nomer yang berujuk pada item yang sama dalam satu pesanan
    c. Product_id: nomer unik produk
    d. Seller_id: nomer unik penjual
    e. Shipping_limit_data: tanggal pengiriman terakhir untuk penjual
    f. Price: harga barang
    g. Freight_value: biaya pengiriman barang
4. Olist order payments
    a. Order_id: nomer unik pesanan
    b. Payement_sequential: nilai yang menyatakan berapa kali pembayaran yang dilakukan oleh customer
    c. Payment_type: metode pembayaran yang dipilih oleh customer

    d. Payment_installments: banyaknya angsuran yang dilakuikan oleh customer
    e. Payment_value: nilai transaksi
5. Olist order review
    a. Review_id: nomer unik review
    b. Order_id: nomer unik pesanan
    c. Review_score: nilai yang diberikan oleh customer dalam rentan 1 sampai 5
    d. Review_comment_message: komentar yang diberikan oleh kostumer dalam bahasa portugis
    e. Review_creation_date: menunjukan tanggal dikirimnya survey pada customer
    f. Review_answer_timestamp: menunjukan tanggal setalah customer mengisi survey yang diberikan
6. Olist order
    a. Order_id: nomer unik pesanan
    b. customer_id : nilai unik yang dipakai dalam dataset untuk menghubungkan ke dataset yang lain.
    c. Order_status: acuan status pesanan (pengiriman, diantar dan lain-lain)
    d. Order_purchase_timestamp: menunjukan waktu pembelian
    e. Order_approved_at: menunjukan waktu pembayaran
    f. Order_deliver_carrier_date: menunjukan tanggal dikirimnya pesanan oleh kurir
    g. Order_deilver_customer_date: menunjukan tanggal dikirimnya pesanan ke kostumer
    h. Order_estimated_delivery_date: menunjukan estimasi lama waktu pengiriman ke pada kostomer
7. Olist product
    a. Product_id: nomer unik produk
    b. Product_category_name: ketogy produk dalam portugis
    c. Product_name_lenght: panjang huruf nama produk
    d. Product_description_lenght: panjang huruh deskripsi dari produk
    e. Product_photos_qty: banyaknya photo produk
    f. Product_weight _g: nilai berat produk dalam gram
    g. Product_lenght_cm: nilai panjang produk dalam cm
    h. Product_height_cm: nilai tinggi produk dalam cm
    i. Product_width_cm: nilai lebar produk dalam cm
8. Olist seller
    a. Seller_id: nilai unik penjual
    b. Seller_zip_code_prefix: 5 digit pertama dari zip code penjual
    c. Seller_city: kota penjual
    d. Seller_state: negara bagian penjual
9. Product category name transalation
    a. Product_category_name: nama kategori dalam portugis
    b. Product_category_name_english: nama kategori dalam bahasa inggris
    c.

Data proses

Dalam riset ini proses data *cleaning* dan *wrangling* data dilakukan menggunakan python dibantu dengan library yang tersedia untuk python diantaranya:

1. Library pandas
2. Library numpy
3. Library Sqlite3
4. Library IPython
5. Library seaborn
6. Library dataFrame_image

Tahap pertama dalam penelitian ini adalah membuka data set yang akan dilakukan, dengan menggunakan library sqlite3 maka Python dapat membaca format .db dengan menggunakan query

```
# create connection to database file
database = "olist.db"
connection = sql.connect(database)
cur = connection.cursor()
# print all table in database
table_list = [a for a in cur.execute("SELECT name FROM sqlite_master WHERE type = 'table'")] # fungsi ini akan membuat list table yang terdabat dalam file database yang ada
print(table_list)
```

setelah mendapat list table dari data set maka peneliti dapat menggunakan library pandas untuk membuat tabular data dari dataset tersebut:

```
# make geolocation table
df_geoloc = "SELECT * FROM olist_geolocation_dataset"
geoloc_table = pd.read_sql_query(df_geoloc, connection)
```

setelah mendapatkan table pada data tersebut maka kita bisa melanjutkan dalam cleaning data dan wrangling data menggunakan python. Tahap selanjutnya adalah cleaning dan wrangling menggunakan objek tujuan yang sudah ditentukan.

objek penelitian:

1. **Mencari sales summary per item**
   Objek penelitian pertama adalah mencari jumlah transaksi yang dilakukan per item dalam data yang ada

```
#make order_items table
df_order_items = "SELECT * FROM olist_order_items_dataset"
order_items_table = pd.read_sql_query(df_order_items, connection)
```

```
#make products table
df_products = "SELECT * FROM olist_products_dataset"
products_table = pd.read_sql_query(df_products, connection)

#make products eng name table
df_products_translate = "SELECT * FROM product_category_name_translation"
products_eng_table = pd.read_sql_query(df_products_translate, connection)

#Merge two table with product id as join
sales_products = order_items_table.merge(products_table, on = "product_id")
sales_products = sales_products.merge(products_eng_table, on =
"product_category_name")
```

tabel 1. table pergabungan table olist_order_items_dataset, olist_products_dataset dan product_category_name_translation

	index_x	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value	index_y	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm	index	product_category_name_english
0	0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.9	13.29	25865	cool_stuff	58.0	598.0	4.0	650.0	28.0	9.0	14.0	22	cool_stuff
1	8346	130898c0987d1801452a8ed92a670612	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-07-05 02:44:11	55.9	17.96	25865	cool_stuff	58.0	598.0	4.0	650.0	28.0	9.0	14.0	22	cool_stuff
2	36646	532ed5e14e24ae1f0d735b91524b98b9	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2018-05-23 10:56:25	64.9	18.33	25865	cool_stuff	58.0	598.0	4.0	650.0	28.0	9.0	14.0	22	cool_stuff
3	49106	6ffc31653edb8c83e1ef3940ff0ff750	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-08-07 18:55:08	58.9	16.17	25865	cool_stuff	58.0	598.0	4.0	650.0	28.0	9.0	14.0	22	cool_stuff
4	55021	7d19f4ef4d04461989632411b7e588b9	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-08-16 22:05:11	58.9	13.29	25865	cool_stuff	58.0	598.0	4.0	650.0	28.0	9.0	14.0	22	cool_stuff

Sumber: penulis

Kemudian dilakukan pencarian kolom yang terdapat nilai null dan duplicate:

```
# find missing value
sales_products.isna().sum()
# find duplicated data in dataset
sales_products.duplicated(keep=False).sum()
```

selanjutnya kita dapat melakukan grouping dan agregasi

```
# define coloum that needed for analysis
coll_1 = ["product_category_name_english", "price"]
category_grouping =
sales_products[coll_1].groupby("product_category_name_english").agg(['max', 'min',
'sum', 'mean'],inplace=True)
table_1 = category_grouping.sort_values(by =["product_category_name_english"],
ascending= True )
```

tabel 2. table hasil grouping dan agregasi

	price			
	max	min	sum	mean
product_category_name_english				
agro_industry_and_commerce	2990.00	12.99	72530.47	342.124858
air_conditioning	1599.00	10.90	55024.96	185.269226
art	6499.00	3.50	24202.64	115.802105
arts_and_craftmanship	289.49	9.80	1814.01	75.583750
audio	598.99	14.90	50688.50	139.254121

2. **Mencari Total penjualan per region**

Objektif penelitian ini mencari nilai total transaksi penjualan masing-masing region.

Tahapan pertama adalah mengecek nilai customer state apakah sudah benar dengan data geografis pada brazil, brazil mempunyai 27 negara bagian.

```
# finding how many state in the table
customer_table["customer_state"].unique()
```

hasil data fungsi unique adalah menampilkan data yang berbeda dari kolom customer_state terdapat 27 negara bagian sesuai dengan hasil geografis di brazil.

```
# nilai unique dari customer_state
array(['SP', 'SC', 'MG', 'PR', 'RJ', 'RS', 'PA', 'GO', 'ES', 'BA', 'MA', 'MS', 'CE', 'DF', 'RN', 'PE',
'MT', 'AM', 'AP', 'AL', 'RO', 'PB','TO', 'PI', 'AC', 'SE', 'RR'], dtype=object)
```

kemudian dilakukan data cleaning dan mencari nilai null yang sama seperti objektivitas yang pertama.

```
table_2 = sales_per_state.groupby(by="customer_state").sum().sort_values(by="price", ascending=False)
```

Tabel 3. tabel hasil grouping dan sort value dari wilayah yang mempunyai transaksi penjualan paling tinggi

customer_state	price
SP	5202955.05
RJ	1824092.67
MG	1585308.03
RS	750304.02
PR	683083.76

3. **mencari produk yang paling banyak pada region yang paling banyak dibeli**
   Analisa ini adalah mencari produk yang paling sering di beli pada setiap negara bagian yang ada, table yang dibutuhkan adalah table order_customer dan order_dataset.

Tabel 4. Tabel hasil dari merge tabel dan grouping menggunakan produk

	product_category_name_english	customer_state
0	cool_stuff	RJ
1	cool_stuff	GO
2	cool_stuff	MG
3	cool_stuff	PR
4	cool_stuff	MG

```
region_sales_condition = product_sales_region.groupby(["customer_state","product_category_name_english"]).agg({"product_category_name_english":"count"})
```

Tabel 5. Tabel hasil dari grouping negara bagian dengan category produk

		product_category_name_english
customer_state	product_category_name_english	
AC	auto	4
	baby	3
	bed_bath_table	8
	books_general_interest	2
	christmas_supplies	1
	computers	1
	computers_accessories	15
	consoles_games	1
	cool_stuff	1
	electronics	4
	fashion_bags_accessories	1
	fashion_shoes	2
	furniture_decor	44
	furniture_living_room	1
	garden_tools	2
	health_beauty	9
	housewares	6
	luggage_accessories	2
	market_place	1
	musical_instruments	1

4. **seberapa kemungkinan seorang akan melakukan transaksi bila dia berada dalam satu region(city,atau provinsi) yang sama?**

table yang dibutuhkan adalah table sellers_dataset, customer_dataset dan order_dataset

tabel 6. Tabel hasil dari merge data sellers_dataset, customer_dataset dan order_dataset

	customer_city	seller_city	order_delivered_customer_date
0	campinas	campinas	2018-08-09 20:55:48
1	campinas	campinas	2018-08-29 18:32:58
3	campinas	campinas	2018-07-03 18:28:41
4	campinas	campinas	2017-11-18 14:21:26
5	campinas	campinas	2018-07-13 14:51:05

Setelah mendapat data yang dibutuhkan maka tahap selanjutnya adalah memfilter data dimana order_delivered_customer_data terisi karena kita hanya mencari nilai dimana sudah dilakukan pengiriman produk dari penjual kepada pembeli. Dan juga mencari data dimana bila pembeli

dan penjual dalam satu kota maka kolom lokasi_transaksi akan memberikan nilai “satu wilayah” dan bila tidak maka akan menampilka “beda wilayah”.

```
location_condition["lokasi_transaksi"] =np.where(location_condition["customer_city"] == location_condition["seller_city"],"Satu wilayah","beda wilayah")
```

tabel 7. Hasil penyortiran lokasi pembeli dan penjual

	customer_city	seller_city	order_delivered_customer_date	lokasi_transaksi
0	campinas	campinas	2018-08-09 20:55:48	Satu wilayah
1	campinas	campinas	2018-08-29 18:32:58	Satu wilayah
3	campinas	campinas	2018-07-03 18:28:41	Satu wilayah
4	campinas	campinas	2017-11-18 14:21:26	Satu wilayah
5	campinas	campinas	2018-07-13 14:51:05	Satu wilayah

```
# calculate data in tabel
total_data = len(location_condition)
kondisi_1 = len(location_condition[location_condition["lokasi_transaksi"] == "Satu wilayah"])
kondisi_2 = len(location_condition[location_condition["lokasi_transaksi"] == "beda wilayah"])

# mencari conditional probability
prob_sama_wilayah = kondisi_1/total_data
prob_beda_wilayah = kondisi_2/total_data

# print
print(f"-probability customer yang berada dalam wilayah yang sama dengan seler adalah {prob_sama_wilayah*100:.2f}% dan yang berbeda wilayah adalah {prob_beda_wilayah*100:.2f}%")
```

dengan menggunaka conditional probability bisa mendapatkan probability customer yang berada dalam wilayah yang sama dengan seler adalah 96.04% dan yang berbeda wilayah adalah 3.96%.

5. **mencari rata-rata review bintang yang diterima seller**

table yang dibutuhkan adalah table order_item_dataset, order_reviews_dataset dan order_dataset

tabel 8. Hasil merge data order_item_dataset, order_reviews_dataset dan order_dataset

	seller_id	review_score
0	6d803cb79cc31c41c4c789a75933b3c7	4
1	6d803cb79cc31c41c4c789a75933b3c7	4
2	8e6d7754bc7e0f22c96d255ebda59eba	5
3	a1043bafd471dff536d0c462352beb48	5
4	a1043bafd471dff536d0c462352beb48	5

```
# grouping and aggregate data
customer_review["avg_review_score"] =
customer_review.groupby("seller_id").agg({"review_score" :
"mean"}).reset_index(drop=True)
# search for null in coloum
seller_statistic.isna().sum()
#filter null in data
review_score_seller = seller_statistic.dropna(subset= ["avg_review_score"])
```

tabel 9. Hasil agregasi review skor

	seller_id	avg_review_score
0	6d803cb79cc31c41c4c789a75933b3c7	3.666667
1	6d803cb79cc31c41c4c789a75933b3c7	3.902542
2	8e6d7754bc7e0f22c96d255ebda59eba	1.000000
3	a1043bafd471dff536d0c462352beb48	3.982143
4	a1043bafd471dff536d0c462352beb48	5.000000

Buat histogram dari data yang ada menggunakan library seaborn, dengan menggunaka nilai avg_review_score

```
# plotting a histogram
sns.distplot(review_score_seller["avg_review_score"], hist=True, kde=True,
        bins=int(5), color = 'darkblue',
        hist_kws={'edgecolor':'black'},
        kde_kws={'linewidth': 1})
```

gambar 1. Histogram review score