Data Wrangling using SQL

Data wrangling describes a series of processes designed to explore, transform, and validate raw datasets from their messy and complex forms into high-quality data with good integrity and consistency into produce valuable insights and guide business decisions in later analytics purposes.

For this Data Wrangling Project, I'm using electronic store sales dataset from Kaggle:
https://www.kaggle.com/datasets/saumaydhaundiyal/electronic-store-sales-data

For the steps I'm using for this project are:

1. Data Discovery
2. Data Cleaning
3. Data Transformation
4. Data Enriching
5. Data Validating
6. Data Publishing

Data Discovery

The main purposes in this step will be:

- Import data from our local machine
- Gather useful insight & information for future step.

Importing Data

In this step, instead using Table Import Wizard in MySQL Workbench, we will be using MySQL console. The reason is because using Table Import Wizard is inefficient to importing large dataset (the progress is too slow)

With this method we need to create our empty table to store the data later. We set our preferred datatype and match the column from our raw dataset.

```
mysql> use kaggle;
Database changed
mysql> CREATE TABLE `kaggle`.`merged_sales_data` (
    ->   `order_id` INT NULL,
    ->   `product` VARCHAR(50) NULL,
    ->   `quantity_ordered` INT NULL,
    ->   `price_each` FLOAT NULL,
    ->   `order_date` DATETIME NULL,
    ->   `purchase_address` VARCHAR(100) NULL);
Query OK, 0 rows affected (0.02 sec)
```

Then, since our data is in separated csv files. We need to merge it first (we can perform this using python or just CMD console)

```
C:\Users\renal>cd "C:/Users/renal/Documents/Renaldo's File/Data Analyst Portofolio -Renaldo
 Livando/Project2 Data Wrangling using SQL/Electronic Stores Sales Raw Data 2019"

C:\Users\renal\Documents\Renaldo's File\Data Analyst Portofolio -Renaldo Livando\Project2 D
ata Wrangling using SQL\Electronic Stores Sales Raw Data 2019>copy Sales*.csv merged_electr
onic_sales_data.csv
Sales_April_2019.csv
Sales_August_2019.csv
Sales_December_2019.csv
Sales_February_2019.csv
Sales_January_2019.csv
Sales_July_2019.csv
Sales_June_2019.csv
Sales_March_2019.csv
Sales_May_2019.csv
Sales_November_2019.csv
Sales_October_2019.csv
Sales_September_2019.csv
        1 file(s) copied.
```

Next, we need to make sure our local_infile variable is activate, then restart our console and login with local_infile used.

```
mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | OFF   |
+---------------+-------+
1 row in set (0.01 sec)

mysql> set global local_infile=true;
Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | ON    |
+---------------+-------+
1 row in set (0.00 sec)

mysql> quit
```

```
Command Prompt - mysql --

Microsoft Windows [Version 10.0.22631.2191]
(c) Microsoft Corporation. All rights reserved.

C:\Users\renal>mysql --local-infile=1 -u root -p
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.33 MySQL Community Server - GPL
```

And the last step in this step will load the data from raw dataset into our table that we created.

```
mysql> LOAD DATA LOCAL INFILE "C:/Users/renal/Documents/Renaldo's File/Data Analyst Portofolio -Renaldo Livando/Project2
 Data Wrangling using SQL/merged_electronic_sales_data.csv"
    -> INTO TABLE merged_sales_data
    -> FIELDS TERMINATED BY ","
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\r\n'
    -> IGNORE 1 ROWS
    -> (order_id,product ,quantity_ordered,price_each,@order_date ,purchase_address )
    -> set order_date = STR_TO_DATE(@order_date,'%m/%d/%y %H:%i');
Query OK, 186850 rows affected, 3055 warnings (3.58 sec)
Records: 186850  Deleted: 0  Skipped: 0  Warnings: 3055
```

Give a check on MySQL Workbench to make sure the data is imported correctly.

```
1 •  select * from merged_sales_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_id	product	quantity_ordered	price_each	order_date	purchase_address
176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001
0		0	0	0000-00-00 00:00:00	
176559	Bose SoundSport Headpho...	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215
176560	Google Phone	1	600	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001
176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001
176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001
176562	USB-C Charging Cable	1	11.95	2019-04-29 13:03:00	381 Wilson St, San Francisco, CA 94016
176563	Bose SoundSport Headpho...	1	99.99	2019-04-02 07:46:00	668 Center St, Seattle, WA 98101
176564	USB-C Charging Cable	1	11.95	2019-04-12 10:58:00	790 Ridge St, Atlanta, GA 30301
176565	Macbook Pro Laptop	1	1700	2019-04-24 10:38:00	915 Willow St, San Francisco, CA 94016

Check and review our dataset

```
1      -- Select our dataset head
2 •    select * from merged_sales_data
3      LIMIT 5;
4
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	product	quantity_ordered	price_each	order_date	purchase_address
176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001
0		0	0	0000-00-00 00:00:00	
176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215
176560	Google Phone	1	600	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001
176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001

Insight from dataset review:

- All the column in the dataset is necessary for our later analysis, there's no need to commit column filtering
- That seems we need to extract the data from column "Order Date" and "Purchase Address" to make new column like "City", "Postal Code", "Month", etc.
- We can add new calculated column from column "Quantity Ordered" and "Price Each"

Check dataset total rows

```
-- Check total rows in our dataset
select count(*) from merged_sales_data;
```

Result Grid | Filter Rows: | Export: | Wrap

count(*)
186850

Check empty values in our dataset. We will use 2 columns with different data type to check this.

```
-- Check empty values in our dataset by INT column
select count(*) from merged_sales_data WHERE order_id IS NULL OR order_id=0;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(*)
900

```
-- Check empty values in our dataset by string column
select count(*) from merged_sales_data WHERE product IS NULL OR product="";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(*)
545

From result we got in here. We can conclude that the number of missing values quite high. And because the order_id column occur more missing values than the product column, we will be using order_id column to remove the missing values later instead of using string/text column.

Check if our dataset ID is unique key or not

```
-- Check order_id is unique key or not
select count(DISTINCT order_id), count(order_id) from merged_sales_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(DISTINCT order_id)	count(order_id)
178438	186850

The number of rows is different. Means ID column in our dataset not unique.

Data Cleaning

The main purposes in this step will be remove errors that might distort or damage the accuracy of your analysis. This includes tasks like standardizing inputs, handling empty cells or missing values, handling duplicate values and fixing string data value by perform trimming.

Remove missing values

With insight from our Data Discovery, we already decide way to handle our data missing values is by removing it.

```
-- Remove missing values
delete from merged_sales_data
WHERE order_id IS NULL OR order_id=0;
```

on Output

	Time	Action	Message
3	17:38:54	SHOW COLUMNS FROM `kaggle`.`merged_sales_data`	OK
9	17:39:44	delete from merged_sales_data WHERE order_id IS NULL OR order_id=0	900 row(s) affected

Handle duplicate values

We need to perform checking for decide action to handling the duplicate values.

Checking duplicate values

Check count of duplicate values occurred. We will using row_number method.

```
-- Check duplicate values accross all columns
select count(*) from (
    SELECT
        *,
        ROW_NUMBER() OVER (
            PARTITION BY order_id, product, quantity_ordered,
                         price_each, order_date, purchase_address
            ORDER BY order_id) AS row_num
    FROM
        merged_sales_data
    ) as b
        where row_num>1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(*)
264

With this check, we found there's occur 264 duplicate values. Before we decide a way to handle these duplicate values, we need to understand and interpret the dataset.

- The data is recording any purchase in the store into a row.
- One Order ID should have One Order Date.
- One Order ID should have One Purchase Address.
- One Order ID could have several different products.

With these 4 points, we can conclude that if Order ID have same Product recorded different rows doesn't make sense. So, we decide to remove duplicate rows that duplicated across all column.

Remove duplicate value

```sql
-- Remove duplicate values
DELIMITER $$
CREATE TABLE `merged_sales_data_copy` (

INSERT INTO merged_sales_data_copy
select * from merged_sales_data
group by order_id, product, quantity_ordered, price_each, order_date, purchase_address;

DROP TABLE merged_sales_data;
ALTER TABLE merged_sales_data_copy RENAME TO merged_sales_data;
SELECT COUNT(*) FROM merged_sales_data;
$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
185686

Trim string column

We need perform trimming into our string column in case we got unnecessary space inside our value

```sql
-- Trim and remove double space from string column
update merged_sales_data
set product = trim(REPLACE(
                        REPLACE(
                            REPLACE(product,
                            ' ','<>'),
                        '><',''),
                    '<>',' '));

update merged_sales_data
set product = trim(REPLACE(
                        REPLACE(
                            REPLACE(purchase_address,
                            ' ','<>'),
                        '><',''),
```

Data Transformation

The main purposes in this step will be:

- Attribute Construction: in which new attributes are added or created from existing attributes
- Generalization: where low-level data attributes are converted into high-level data attributes (in this project is "Purchase Address" column).
- Re-Arrange Column

Attribute Construction

Create month column

```sql
-- Create new column Month that contain name of month from order_date
alter table merged_sales_data add month varchar(10);
UPDATE merged_sales_data set month=MONTHNAME(order_date);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_id	product	quantity_ordered	price_each	order_date	purchase_address	month
141234	944 Walnut St, Boston, MA ...	1	700	2019-01-22 21:25:00	944 Walnut St, Boston, MA 02215	January
141235	185 Maple St, Portland, OR ...	1	14.95	2019-01-28 14:15:00	185 Maple St, Portland, OR 97035	January
141236	538 Adams St, San Francisc...	2	11.99	2019-01-17 13:33:00	538 Adams St, San Francisco, CA 94016	January

Create calculation column total_cost

```sql
-- Create new column total_cost that contain calculaton from existing column
alter table merged_sales_data add total_cost float;
UPDATE merged_sales_data set total_cost=quantity_ordered * price_each;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_id	product	quantity_ordered	price_each	order_date	purchase_address	month	total_cost
141234	944 Walnut St, Boston, MA ...	1	700	2019-01-22 21:25:00	944 Walnut St, Boston, MA 02215	January	700
141235	185 Maple St, Portland, OR ...	1	14.95	2019-01-28 14:15:00	185 Maple St, Portland, OR 97035	January	14.95
141236	538 Adams St, San Francisc...	2	11.99	2019-01-17 13:33:00	538 Adams St, San Francisco, CA 94016	January	23.98
141237	738 10th St, Los Angeles, C...	1	149.99	2019-01-05 20:33:00	738 10th St, Los Angeles, CA 90001	January	149.99

Generalization

We perform generalization on purchase_address column

```
-- Perform generalization on purchase_address column
DELIMITER $$
alter table merged_sales_data add street VARCHAR(30);
update merged_sales_data set street= SUBSTRING_INDEX(purchase_address, ',', 1);
alter table merged_sales_data add city VARCHAR(30);
update merged_sales_data set city= SUBSTRING_INDEX(SUBSTRING_INDEX(purchase_address, ',', 2),',',-1);
alter table merged_sales_data add state VARCHAR(2);
update merged_sales_data set state= LEFT(TRIM(SUBSTRING_INDEX(purchase_address, ',', -1)),2);
alter table merged_sales_data add postal_code VARCHAR(5);
update merged_sales_data set postal_code=RIGHT(TRIM(SUBSTRING_INDEX(purchase_address, ',', -1)),5);
select * from merged_sales_data;
$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	quantity_ordered	price_each	order_date	purchase_address	month	total_cost	street	city	state	postal_code
Boston, MA ...	1	700	2019-01-22 21:25:00	944 Walnut St, Boston, MA 02215	January	700	944 Walnut St	Boston	MA	02215
ortland, OR ...	1	14.95	2019-01-28 14:15:00	185 Maple St, Portland, OR 97035	January	14.95	185 Maple St	Portland	OR	97035
San Francisc...	2	11.99	2019-01-17 13:33:00	538 Adams St, San Francisco, CA 94016	January	23.98	538 Adams St	San Francisco	CA	94016
s Angeles, C...	1	149.99	2019-01-05 20:33:00	738 10th St, Los Angeles, CA 90001	January	149.99	738 10th St	Los Angeles	CA	90001

Re-Arrange

I'm using MySQL Workbench feature to easily re-arrange this by drag it into my preferred order.

project2_dataTransformation_S... | merged_sales_data - Table

Table Name: merged_sales_data Schema: **kaggle**

Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
order_id	INT									NULL
order_date	DATETIME									NULL
month	VARCHAR(10)									NULL
product	VARCHAR(50)									NULL
quantity_ordered	INT									NULL
price_each	FLOAT									NULL
total_cost	FLOAT									NULL
purchase_address	VARCHAR(100)									NULL

This setting will execute syntax below:

```
-- Re-arrange table
ALTER TABLE `kaggle`.`merged_sales_data`
CHANGE COLUMN `order_date` `order_date` DATETIME NULL DEFAULT NULL AFTER `order_id`,
CHANGE COLUMN `month` `month` VARCHAR(10) NULL DEFAULT NULL AFTER `order_date`,
CHANGE COLUMN `total_cost` `total_cost` FLOAT NULL DEFAULT NULL AFTER `price_each`;
```

Data Enriching

The main purpose in this step will be: enrich the dataset by adding values from other datasets.

```sql
-- Enrich dataset with season column
alter table merged_sales_data add season VARCHAR(10);
update merged_sales_data set season= CASE   WHEN month='January' or month='February' or month='December' THEN 'Winter'
                                            WHEN month='March' or month='April' or month='May' THEN 'Spring'
                                            WHEN month='June' or month='July' or month='August' THEN 'Summer'
                                            ELSE 'Fall'
                                        END;
```

```sql
select order_id,month,season from merged_sales_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

order_id	month	season
141234	January	Winter
141235	January	Winter
141236	January	Winter
141237	January	Winter
141238	January	Winter
141239	January	Winter
141240	January	Winter
141241	January	Winter

Data Publishing

In MySQL, exported data by default won't export our table header. We need to add our header in first rows using UNION.

```sql
-- List our column name
SELECT group_concat(CONCAT("'",COLUMN_NAME,"'") ORDER BY ORDINAL_POSITION SEPARATOR ',')
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'merged_sales_data'
AND TABLE_SCHEMA = 'kaggle';

-- Union and export our table to csv file
SELECT 'order_id','order_date','month','product','quantity_ordered','price_each','total_cost','purchase_address','street','city','state','postal_code
UNION ALL
SELECT * FROM merged_sales_data
INTO OUTFILE "C:/Users/renal/Documents/Renaldo's File/Data Analyst Portofolio -Renaldo Livando/Project2 Data Wrangling using SQL/out/cleaned_electron
FIELDS ENCLOSED BY '"'
TERMINATED BY ','
LINES TERMINATED BY '\r\n';
```