QUÁ TRÌNH XỬ LÝ DATA NHÓM ADY

1. Đọc file json và chuyển đổi file thành DATAFRAME

```
df = pd.read_json("CHOTOT_motorcycles2.json")
```

* Lưu ý lệnh này chỉ có thể chuyển đổi data thành DATAFRAME chứ không thể thành time series (có tham số là hàng index và cột columns)

- Nếu muốn data của mình trở thành định dạng TIME SERIES thì chúng ta cần phải tạo hoặc sử dụng một cột datetime có sẵn nhằm để có thể
    + Sắp xếp các giá trị theo cột datetime
    + Tổng hợp theo khoảng thời gian

2. Chỉ lấy ra những Columns cần dùng trong DATAFRAME (sau khi lấy ra)

```
df_binary = df[["name", "price", "Location", "Year_of_manufacture",
"Kilometers_driven"]]

df_binary.columns = ["Name", "Price", "Location", "Year", "Km"]
```

- Ở đây dòng code df_binary nhằm mục đích là sẽ lấy các cột trùng với các giá trị có trong file JSON (phải match mới lấy ra được) để có thể tối ưu thay vì lấy ra hết tất cả các cột mà không dùng đến

- Tiếp theo, ở câu lệnh df_binary.columns thì ở đây việc dùng câu lệnh này nhằm gán lại tên cho các cột ở trên để dễ đọc và thao tác (luôn nhớ chỉ có columns và index)

3. Xử lý sạch DATA

```
df_binary["Price"] = pd.to_numeric(df_binary["Price"].str.replace(".",
"").str.replace(" đ", ""), errors="coerce")
df_binary["Year"] = pd.to_numeric(df_binary["Year"], errors="coerce")
df_binary["Km"] = pd.to_numeric(df_binary["Km"].str.replace(" km", ""),
errors="coerce")
df_binary["Location"] = df_binary["Location"].str.extract(r"(Quận
[^,]+)")
```

- Price: ở đây chúng ta sẽ nói chuyển đổi giá trị price từ một chuỗi thành giá trị số. Tiếp theo, dùng các lệnh str.replace (….) nhằm replace dấu . và đ (20.000.000 đ)

- Year: cũng chuyển qua thôi

- Km: chuyển qua, loại bỏ km

- Location: có đặc biệt chút, do chỉ lấy quận trong 1 chuỗi (địa chỉ, phường, quận, tp….) nên mình sẽ dùng str.extract để trích xuất regex ra. Dùng các ký tự đó là như này nè "phường…., Quận Hải Châu, tp ….) thì khi mà dùng các ký tự **[^,]** là lấy hết những ký tự trừ dấu phẩy còn dấu **+** là nhằm giúp cho hiểu là sẽ khớp với tất cả các ký tự nhưng phải tuân theo cái điều kiện trước đó (giống như add with condition)

*Lưu ý: errors = "coerce" là thay vì nó báo lỗi nếu định dạng hay bla bla data lỗi thì nó trả về NaN.

4. Câu lệnh đầu tiên nếu bạn muốn đụng vào MATPILOT :)

```
fig, axes = plt.subplots(2,2, figsize=(18, 10))
```

```
fg, (ax1, ax2) = plt.subplots(1, 2 , figsize= (15,10))
```

- Cơ bản thì là tạo 1 khung để mà có thể chứa vào các biểu đồ mà mình muốn trình bày theo kích cỡ là figsize=(.. inch rộng, … inch cao)
- Còn các số như 1,2 hoặc 2,2 biểu thị cho việc tạo khung với một ma trận với 1 hàng 2 cột hoặc 2 hàng 2 cột (cái này để tiện sắp xếp vị trí của biểu đồ theo vị trí trên khung)

5. Thao tác vẽ biểu đồ bằng SEABORN

```
sns.regplot(x="Km", y="Price", data=df_binary, ax=axes[0,0])
sns.regplot(x="Year", y="Price", data=df_binary, ax=axes[0,1])
sns.barplot(x="Location", y="Price", data=df_binary, ax=axes[1,0])
axes[1,1].axis("off")
```

- Cơ bản thì mình sẽ vẽ 3 chart, nhằm xác định các giá trị tương quan (features) để mà tìm ra điểm khác biệt khi mà mình muốn so sánh giá trị của nhiều sản phẩm có cùng features:

+ Km and Price: Đương nhiên thì việc mà xác định giá của một chiếc xe máy sẽ phụ thuộc vào số km của chiếc xe (cái mà ax=axes[0,0] là ở hàng 0 và cột 0 ma trận á)

+ Year and Price: Tương tự và tương tự

+ Location and Price: Riêng cái này khác chút dùng cái barplot còn lại không khác gì lắm

*Lưu ý cái axes[1,1].axis("off") ý, thử tắt đi là hiểu vì sao có nó 🙂

6. Set một chút điều kiện để chart mình nhìn đẹp và trực quan hơn (dù chưa đẹp)

```
y_max = 600000000
for ax in axes.flatten():
    ax.set_ylim(0, y_max)
    ax.yaxis.set_major_locator(MultipleLocator(50_000_000))
    ax.ticklabel_format(style="plain", axis="y")

axes[0,0].ticklabel_format(style="plain", axis="x")
axes[0,0].tick_params(axis="x", rotation=45)
axes[0,0].set_xlabel("Số Km")

axes[0,1].tick_params(axis="x", rotation=45)
axes[0,1].set_xlabel("Năm sản xuất")

axes[1,0].tick_params(axis="x", rotation=290)
axes[1,0].set_xlabel("Khu vực")
```

- Xử lý **Y AXIS** của all biểu đồ:

    + Vòng lặp for qua các axes.flatten() nhằm để mà set cái yaxis lại là nó chỉ có thể chạy max 600 triệu và chia mỗi cái đấy ra là 50 triệu (dùng hàm có sẵn import vào)

- Xử lý **X AXIS** của từng biểu đồ:

    + Áp lệnh là ra :)

*Lưu ý: do ở đây vì số lớn nên cái thư viện nó hay trả về là 1e…. nên là dùng cái này để mà chuyển hóa nó ra số chứ không nó ra công thức toán ấy.

```
ax.ticklabel_format(style="plain", axis="y")
```

7. Kết bài

```
plt.tight_layout()
plt.show()
```

- Tới đây là hết bài rồi, bình thường sẽ có thêm cái mà plt.legend() để mà tạo cái ô vuông note lên, nhưng mà bài này không cần nên thôi.