Ciência de Dados e Big Data

Objetivo: **Prepara os dados**

- Limpar/ Uniformizar os dados
- Derivar dados
- Integrar os Dados
- Formatar os Dados

Na primeira entrega a equipe fez uma pre formatação de dados.

RideAddressID	Lat	Lng	RideAddressTypeID	RideID	Coordenadas
2334277	-26.329.754.299.999.900	-48.840.427.999.599.900	1	1183200	-26.329754299999900, -48.840427999599900
2334278	-262.554.657	-486.434.197	2	1183200	-26.2554657, -48.6434197
2334279	-274.919.788	-48.528.287.999.599.900	1	1183201	-27.4919788, -48.528287999999900
2334280	-274.371.486	-4.839.824.309.959.990	2	1183201	-27.4371486, -48.39824309999990
2334281	-198.495.799	-44.019.915.999.599.900	1	1183202	-19.8495799, -44.019915999999900
2334282	-19.936.899	-439.401.603	2	1183202	-19.936899, -43.9401603
2334283	-239.624.233	-4.625.465.759.999.990	1	1183203	-23.9624233, -46.25465759999990
2334284	-238.373.074	-461.321.725	2	1183203	-23.8373074, -46.1321725
2334285	-109.198.019	-37.077.441.799.599.900	1	1183204	-10.9198019, -37.077441799999900

Agora precisamos unificar a coluna RideAddressTypeID em uma única coluna.

```
import pandas as pd

df = pd.read_csv("rideaddress_v1.csv", sep=';')

df['Coordenadas'] = df['Lat'].astype(str) + ', ' + df['Lng'].astype(str)

df_tipo1 = df[df['RideAddressTypeID'] == 1].copy()
df_tipo2 = df[df['RideAddressTypeID'] == 2][['RideID', 'Lat', 'Lng']].copy()

df_tipo2['Coordenadas_2'] = df_tipo2['Lat'].astype(str) + ', ' + df_tipo2['Lng'].astype(str)

resultado = df_tipo1.merge(df_tipo2[['RideID', 'Coordenadas_2']], on='RideID', how='left')

resultado.to_csv('coordenadas_unificadas.csv', index=False, sep=';')
```

Utilizando esse código em python, conseguimos subir o RideAddressTypeID = 2 e trazer a Lat e Lng para a linha 1. Fazendo assim o arquivo fica nesse formato.

RideAddressID	Lat	Lng	RideAddressTypeID	RideID	Coordenadas	Coordenadas_2
2334277	-26.329.754.299.999.900	-48.840.427.999.999.900	1	1183200	-26329754299999900, -48840427999999900	-262554657, -486434197
2334279	-274.919.788	-48.528.287.999.999.900	1	1183201	-274919788, -48528287999999900	-274371486, -4839824309999990
2334281	-198.495.799	-44.019.915.999.999.900	1	1183202	-198495799, -44019915999999900	-19936899, -439401603

Dessa forma conseguimos utilizar esse script.

```python
import pandas as pd
import sys

try:
    from WazeRouteCalculator import WazeRouteCalculator, WRCError
except ImportError:
    sys.exit("ERRO: É necessário instalar a biblioteca WazeRouteCalculator.\nUse: pip install WazeRouteCalculator")

def processar_rotas():
    df = pd.read_csv('Coordenadas_Limpas - Copia.csv', sep=';', low_memory=False)

    origem_col = 'Coordenadas Inicio'
    destino_col = 'Coordenadas Fim'

    df['Distancia (km)'] = None
    df['Tempo (min)'] = None
    df['Erro'] = None

    region = 'EU'
    batch_size = 5000
    batch_number = 1  # Controla a numeração dos arquivos

    print("Iniciando o cálculo das rotas...")

    for index, row in df.iterrows():
        origem_str = row[origem_col]
        destino_str = row[destino_col]

        try:
            waze_route = WazeRouteCalculator(origem_str, destino_str, region)
            tempo_min, distancia_km = waze_route.calc_route_info(real_time=False)

            df.at[index, 'Distancia (km)'] = distancia_km
            df.at[index, 'Tempo (min)'] = tempo_min

        except WRCError as e:
            df.at[index, 'Erro'] = str(e)

        # Salvar lote de 5000 registros
        if (index + 1) % batch_size == 0 or index == len(df) - 1:
            file_name = f'resultado_calculo_rotas_parte_{batch_number}.csv'
            df.to_csv(file_name, index=False, sep=';')
            print(f"Salvando progresso: {file_name}")
            batch_number += 1  # Incrementa o número do lote

    print("Processamento concluído.")

if __name__ == "__main__":
    processar_rotas()
```

Assim conseguimos uma planilha que informa a distância entre um ponto e outro.

RideAddressID	Lat	Lng	RideAddressTypeID	RideID	Coordenadas	Coordenadas_2	KM
2334277	-26.329.754.299.999.900	-48.840.427.999.999.900	1	1183200	-26329754299999900, -48840427999999900	-262554657, -486434197	12.3
2334279	-274.919.788	-48.528.287.999.999.900	1	1183201	-274919788, -48528287999999900	-274371486, -4839824309999990	20.76
2334281	-198.495.799	-44.019.915.999.999.900	1	1183202	-198495799, -44019915999999900	-19936899, -439401603	3.8
2334283	-239.624.233	-4.625.465.759.999.990	1	1183203	-239624233, -4625465759999990	-238373074, -461321725	15.08
2334285	-109.198.019	-37.077.441.799.999.900	1	1183204	-109198019, -37077441799999900	-109071288, -370877194	7.09

O próximo passo será tirar as colunas que não serão mais utilizadas.

```
import pandas as pd

df = pd.read_csv("coordenadas_unificadas.csv",  sep=';')

df = df[['RideAddressID', 'RideID', 'KM']]

df.to_csv("coordenadas_filtradas.csv", index=False, sep=';')
```

e descobrir o dia da semana que essa corrida foi solicitada.

```
import pandas as pd

df_base = pd.read_csv("coordenadas_filtradas.csv",  sep=';')

df_create = pd.read_csv("Ride_v2.csv",  sep=';')

df_merged = df_base.merge(df_create[['RideID', 'Schedule']], on='RideID', how='left')

df_merged.to_csv("planilha_com_create.csv", index=False,  sep=';')
```

Com esse script eu consigo trazer a coluna Schedule, que indica o momento da corrida, para a base que o time está criando e utilizar esse script.

Com esse Script será possível saber qual o dia da solicitação.

```
import pandas as pd

df = pd.read_csv("planilha_com_create.csv",  sep=';')

df['Create'] = pd.to_datetime(df['Schedule'])

df['DiaSemana'] = df['Create'].dt.weekday

df.to_csv("datas_com_numero_dia_semana.csv", index=False,  sep=';')
```

Esse foi o Resultado:

RideAddre	RideID	KM	Schedule	DiaSemana
2334277	1183200	12.3	2021-08-17 10:09:45.6281709	1
2334279	1183201	20.76	2021-08-17 10:09:51.9849278	1
2334281	1183202	3.8	2021-08-17 10:10:07.1660385	1
2334283	1183203	15.08	2021-08-17 10:10:13.8257923	1

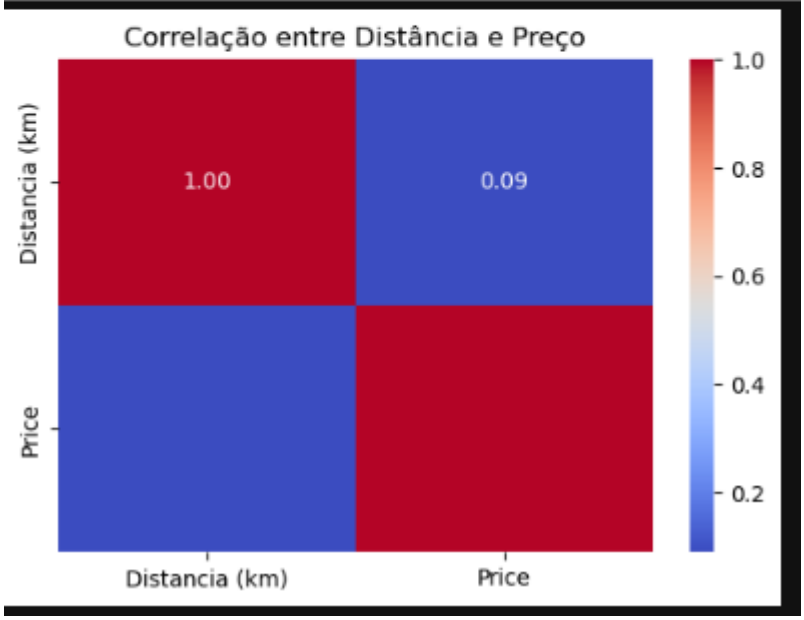
Agora irei trazer a coluna Price:

RideAddr	RideID	KM	Schedule	DiaSemana	Price
2334277	1183200	12.3	2021-08-17 10:09:45.6281709	1	36.00
2334279	1183201	20.76	2021-08-17 10:09:51.9849278	1	151.05
2334281	1183202	3.8	2021-08-17 10:10:07.1660385	1	17.50
2334283	1183203	15.08	2021-08-17 10:10:13.8257923	1	47.50

Dessa forma, iremos utilizar, a distância da corrida, o valor dela e o dia da semana para treinar o algoritmo da Machine Learning.

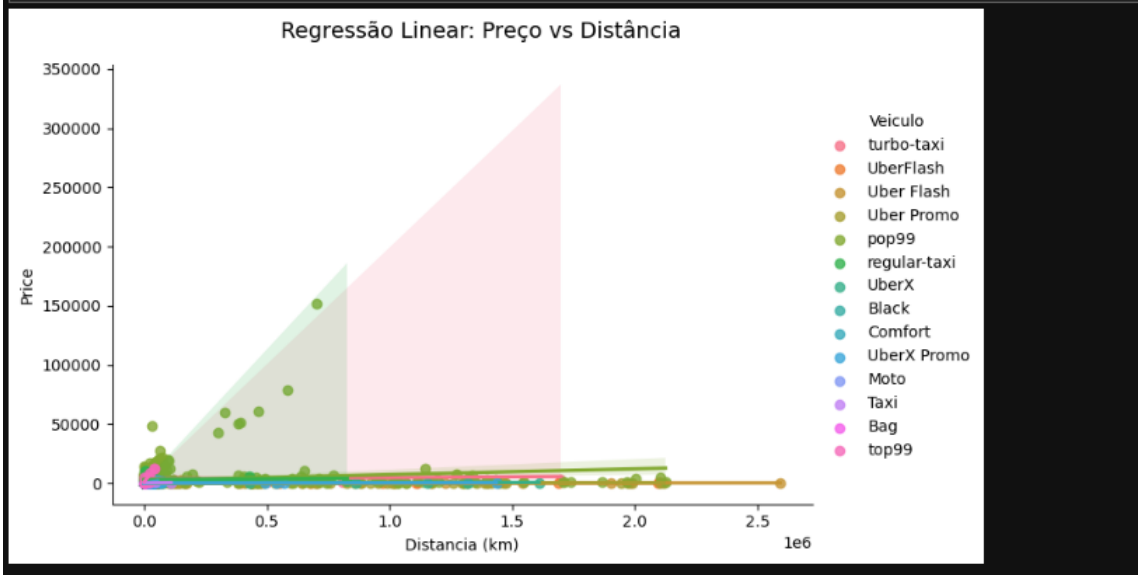
Analisando os dados:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Calcula a correlação
corr = df[['Distancia (km)', 'Price']].corr()

# Plot
plt.figure(figsize=(6, 4))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlação entre Distância e Preço')
plt.show()
```



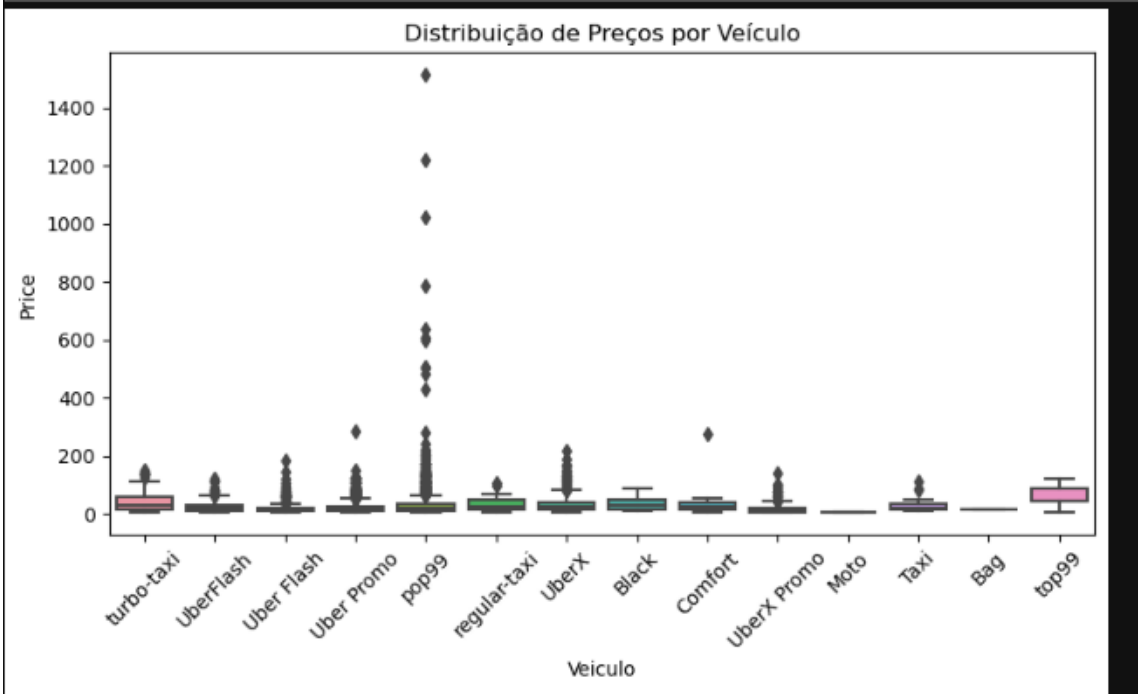
Conseguimos ver que nesse gráfico a correlação entre Distância (KM) e Price é: 0.09, o que quer dizer que é muito baixa. Não seria uma correlação indicada para usar em um modelo de Machine Learning.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Simulação do seu DataFrame, substitua por pd.read_csv se estiver lendo de um arquivo
data = pd.read_csv("resultado_com_dia_semana.csv", sep=";")

# Corrigir números com ponto como milhar e vírgula como decimal
df['Distancia (km)'] = df['Distancia (km)'].astype(str).str.replace('.', '', regex=False).str.replace(',', '.', regex=False).astype(float)
df['Price'] = df['Price'].astype(str).str.replace('.', '', regex=False).str.replace(',', '.', regex=False).astype(float)

# Agora sim, rodar o lmplot
g = sns.lmplot(data=df, x='Distancia (km)', y='Price', hue='Veiculo', aspect=1.5)
g.fig.suptitle('Regressão Linear: Preço vs Distância', fontsize=14)
g.fig.subplots_adjust(top=0.9)
plt.show()
```



- Existe uma tendência geral de aumento de preço com a distância, mas nem todos os veículos seguem isso de forma linear.
- Para alguns tipos, como Moto e Uber Flash, os pontos estão mais agrupados, o que pode indicar preços mais estáveis por distância.
- Outros veículos (ex: Taxi, regular-taxi) têm maior dispersão — mesmo para distâncias curtas, os preços variam muito.

```python
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='Veiculo', y='Price')
plt.title('Distribuição de Preços por Veiculo')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



- Alguns veículos como regular-taxi, pop99, turbo-taxi e UberFlash têm uma distribuição mais ampla de preços.
- Existem muitos outliers, indicando que em alguns casos o mesmo tipo de veículo pode ter preços muito mais altos (talvez por longa distância ou horário).
- Tipos como Uber Flash e Moto parecem ter valores menores e mais concentrados.