Etl_jupyter_notebook

March 29, 2020

Extração, Transformação e Carregamento dos dados (ETL)
Extração no Excel

[1]:
```
# Importar pacote do Oracle e o Pandas
import cx_Oracle
import pandas as pd
```

[2]:
```
# Importar o dataset "Venda de carros"
dataset = pd.read_excel("Venda_de_carros.xlsx")
```

[5]:
```
# Mostrar o conjunto de dados
dataset.head()
```

[5]:

	Data	Fabricante	Estado	Valor_venda	Valor_custo	Desconto	Custo_entrega	Custo_mao_de_obra	Nome_cliente	Modelo	Cor
0	2012-10-04	Rolls Royce	São Paulo	95000	50000	500.0	750	750	Aldo Motors	Camargue	Vermelho
1	2012-01-01	Aston Martin	São Paulo	120000	75000	0.0	1500	550	Honest John	DBS	Azul
2	2012-02-02	Rolls Royce	São Paulo	88000	75000	750.0	1000	550	Bright Orange	Prata Ghost	Verde
3	2012-03-03	Rolls Royce	São Paulo	89000	88000	0.0	1000	550	Honest John	Prata Ghost	Azul
4	2012-04-04	Rolls Royce	São Paulo	92000	62000	0.0	1500	550	Wheels'R'Us	Camargue	Prata

Transformação com Python

[ ]:
```
# Mini pré-processamento dos dados
## Dividir o dataset em várias tabelas, cada uma com os seus atributos e os seus␣
 ↪próprios registos únicos
tabela_fabricantes = pd.read_excel("Venda_de_carros.xlsx",␣
 ↪sheet_name="Tb_fabricantes")

### Alterar dados peculiares
tabela_clientes = pd.read_excel("Venda_de_carros.xlsx", sheet_name="Tb_clientes")
for x, y in enumerate(tabela_clientes["nome_cliente"]):
    if (y == "Wheels'R'Us"):
        tabela_clientes["nome_cliente"][x] = "Wheels are us"
```

```
    elif (y == "Cut'n'Shut"):
        tabela_clientes["nome_cliente"][x] = "Cut and shut"
    else:
        pass

tabela_automoveis = pd.read_excel("Venda_de_carros.xlsx",␣
 ↪sheet_name="Tb_automoveis")
tabela_vendas = pd.read_excel("Venda_de_carros.xlsx", sheet_name="Tb_vendas")

### Criar uma nova coluna da data
nova_data = [x.date().strftime("%d/%m/%Y") for x in tabela_vendas["data"]]

### Substituir coluna velha da data pela nova
tabela_vendas["data"] = nova_data

## Preparar os dados para serem inseridos
tabela_fabricantes = [tuple(registo) for registo in␣
 ↪tabela_fabricantes[["fabricante_id", "nome_fabricante"]].values]
tabela_clientes = [tuple(registo) for registo in tabela_clientes[["cliente_id",␣
 ↪"nome_cliente", "estado"]].values]
tabela_automoveis = [tuple(registo) for registo in␣
 ↪tabela_automoveis[["automovel_id", "cor", "modelo", "valor_custo", \
                                                                  ␣
 ↪"custo_entrega", "custo_mao_de_obra", "desconto"]].values]
tabela_vendas = [tuple(registo) for registo in tabela_vendas[["fabricante_id",␣
 ↪"cliente_id", "automovel_id", "valor_venda", "data"]].values]
```

Carregamento no SGBD Oracle

[164]:
```
# Conectar à base de dados
conexao = cx_Oracle.connect("up201704685", "Simão", "oraalu.fe.up.pt:1521/ALU",␣
 ↪encoding = "UTF-8")
```

[165]:
```
# Criar um cursor para executar comandos SQL
cursor = conexao.cursor()
```

[22]:
```
# Criar as tabelas na base de dados
## Como forma de testar o SGBD Oracle e o sql developer, todas as tabelas foram␣
 ↪lá criadas
```

[166]:
```
# Criar os vários comandos para inserir registos nas várias tabelas (LMD)
comando_tabela_fabricantes = "insert into fabricantes (fabricante_id,␣
 ↪nome_fabricante) values"
comando_tabela_clientes = "insert into clientes(cliente_id, nome_cliente,␣
 ↪estado) values"
comando_tabela_automoveis = "insert into automoveis(automovel_id, cor, modelo,␣
 ↪valor_custo, custo_entrega, custo_mao_de_obra, desconto) values"
comando_tabela_vendas = "insert into vendas(fabricante_id, cliente_id,␣
 ↪automovel_id, valor_venda, data) values"
```

[167]:
```
# Funções para inserir registos
def insercao_de_dados_1(comando, tabela):
    for registo in tabela:
        comando_rasc = comando + str(registo)
        print(comando_rasc)
        cursor.execute(comando_rasc)
        conexao.commit()
        comando_rasc = ""

def insercao_de_dados_2(comando, tabela):
    for registo in tabela:
        comando_rasc = comando + "(" + str(registo[0]) + "," + " " +␣
 ↪str(registo[1]) + "," + " " + str(registo[2]) + "," + \
        " " + str(registo[3]) + "," + " " + "to_date(" + f"'{registo[4]}'" + ","␣
 ↪+ " " + "'dd/mm/yyyy'" + "))"
        print(comando_rasc)
        cursor.execute(comando_rasc)
        conexao.commit()
        comando_rasc = ""
```

[ ]:
```
# Inserir registos
insercao_de_dados_1(comando_tabela_fabricantes, tabela_fabricantes)
insercao_de_dados_1(comando_tabela_clientes, tabela_clientes)
insercao_de_dados_1(comando_tabela_automoveis, tabela_automoveis)
insercao_de_dados_2(comando_tabela_vendas, tabela_vendas)
```