Cento Universitário UNA

Análise de Dados e Big Data

Graduação – TI e Engenharias

Práticas de Laboratório

Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel

2020/02

Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02

Banco de dados SQL

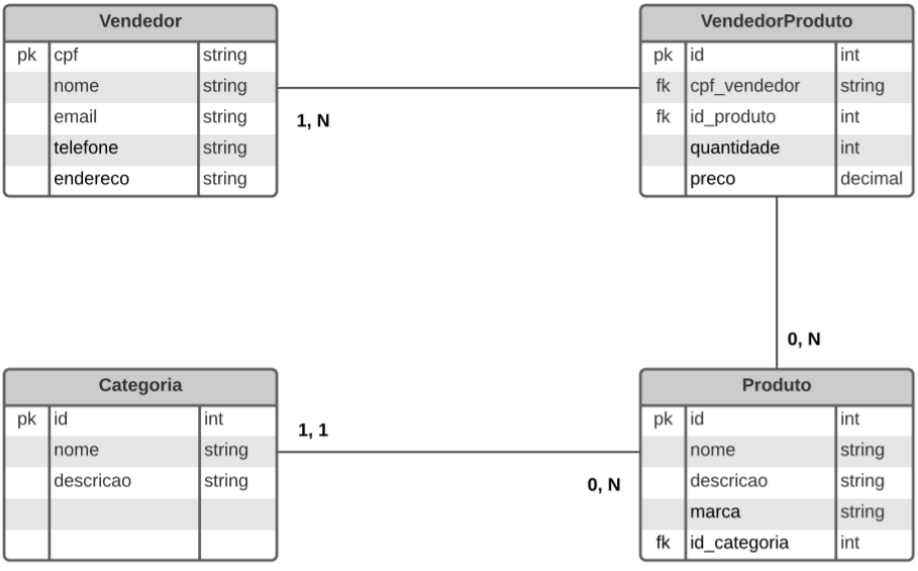
Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02

Prática 04

Especificação

Desenvolva o banco de dados relacional do "marketplace" de um importante "e-commerce" nacional. O banco de dados deve possuir as tabelas: Vendedor, Produto e Categoria. A tabela Vendedor possui os atributos: CPF, nome, e-mail, telefone e endereço. A tabela Produto possui os atributos: identificador, nome, descrição e marca. A tabela Categoria possui os atributos: identificador, nome e descrição. Nesse "marketplace" cada vendedor pode vender quantos produtos desejar, de quantas categorias desejar. Pode acontecer de um determinado produto ser vendido por mais de um vendedor. É necessário registrar a quantidade e o preço dos produtos de cada vendedor. Cada produto é classificado em apenas uma categoria. Mas cada categoria pode ser atribuída a vários produtos. O banco de dados deve ser criado usando o SQL Server Express e a linguagem de programação Python.

Modelo

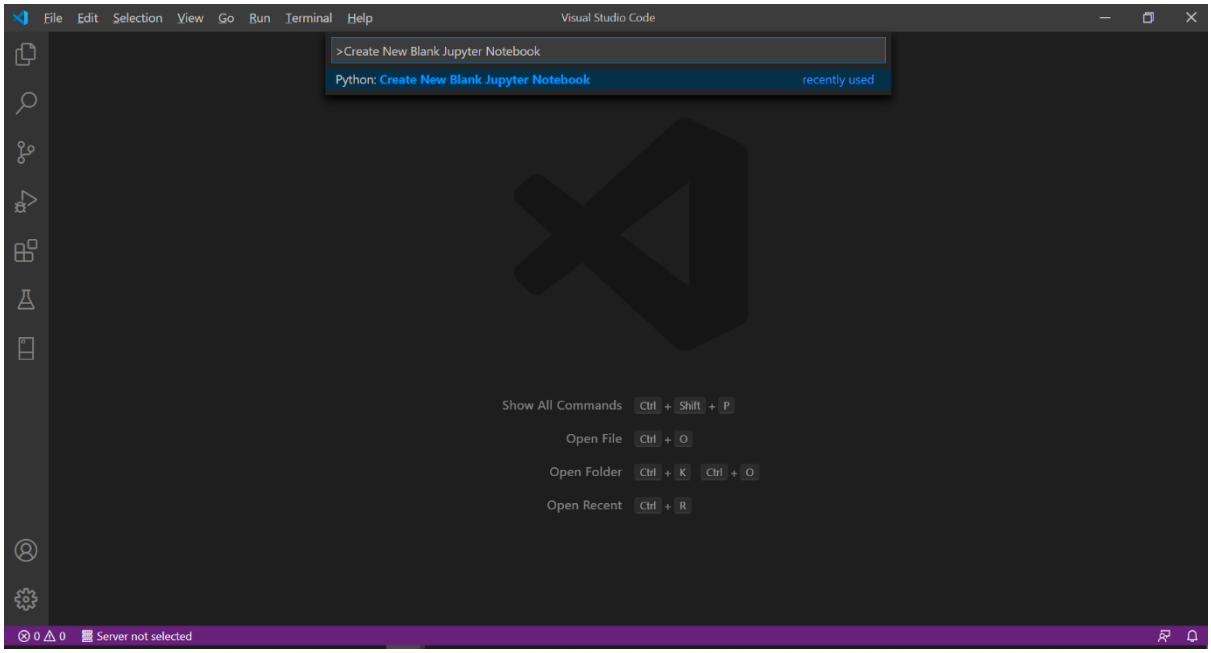


Categoria

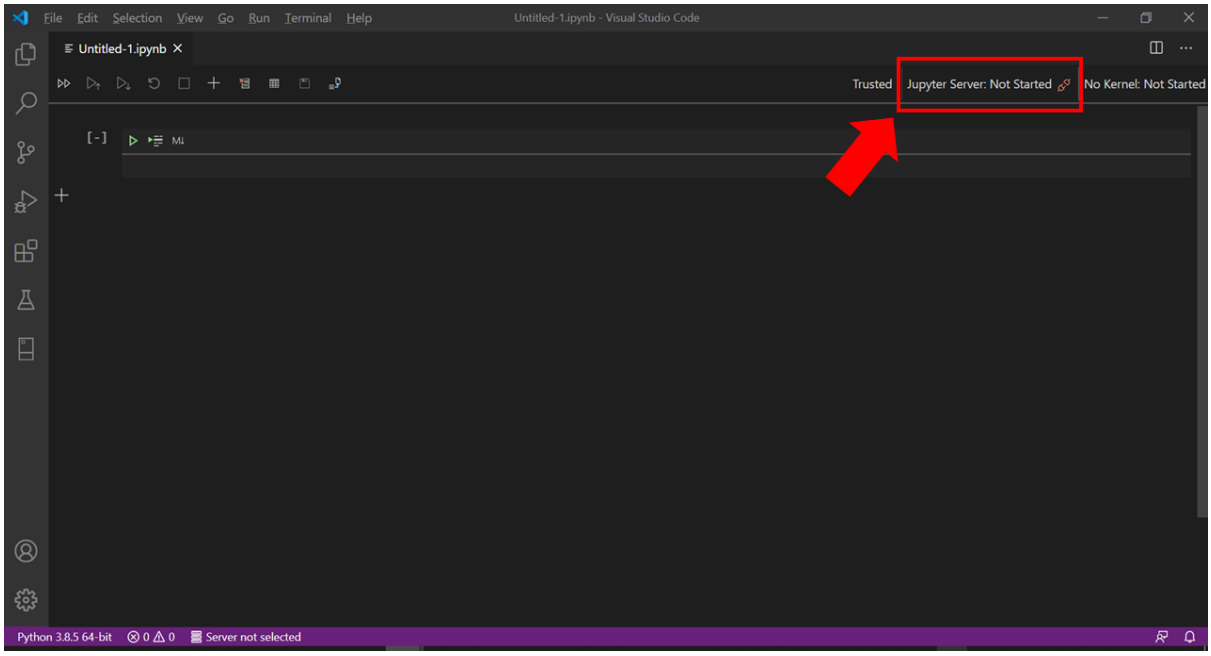
1) No Visual Studio Code, crie um novo notebook. Clique em View > Command Palette ou digite Ctrl + Shift + P. Na caixa de entrada, informe:

Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02

>Create New Blank Jupyter Notebook

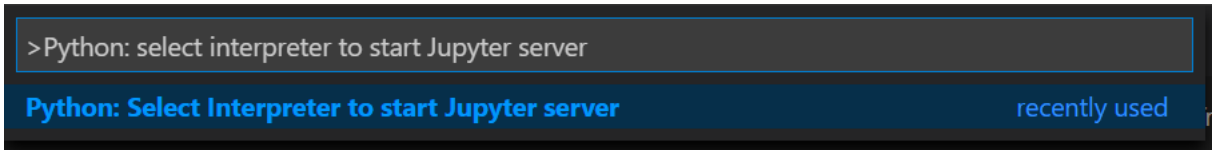


2) Observe se o servidor do Jupyter foi iniciado.

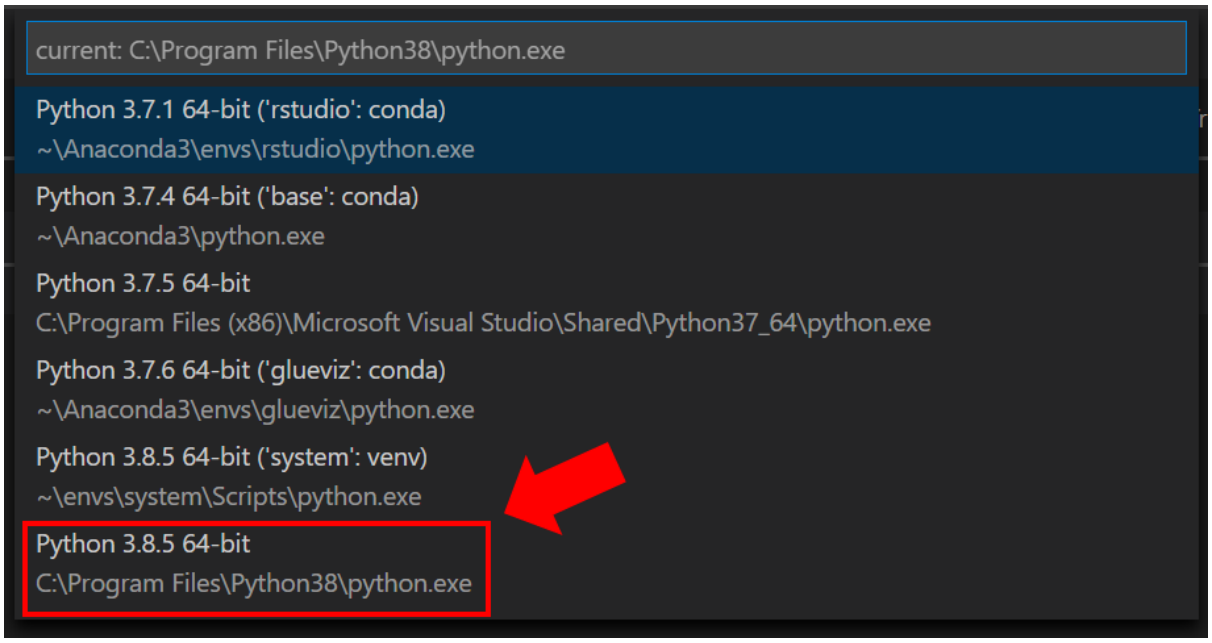


Caso o servidor do Jupyter não tenha sido iniciado, clique em View > Command Palette ou digite Ctrl + Shift + P e Informe:

>Python: select interpreter to start Jupyter server

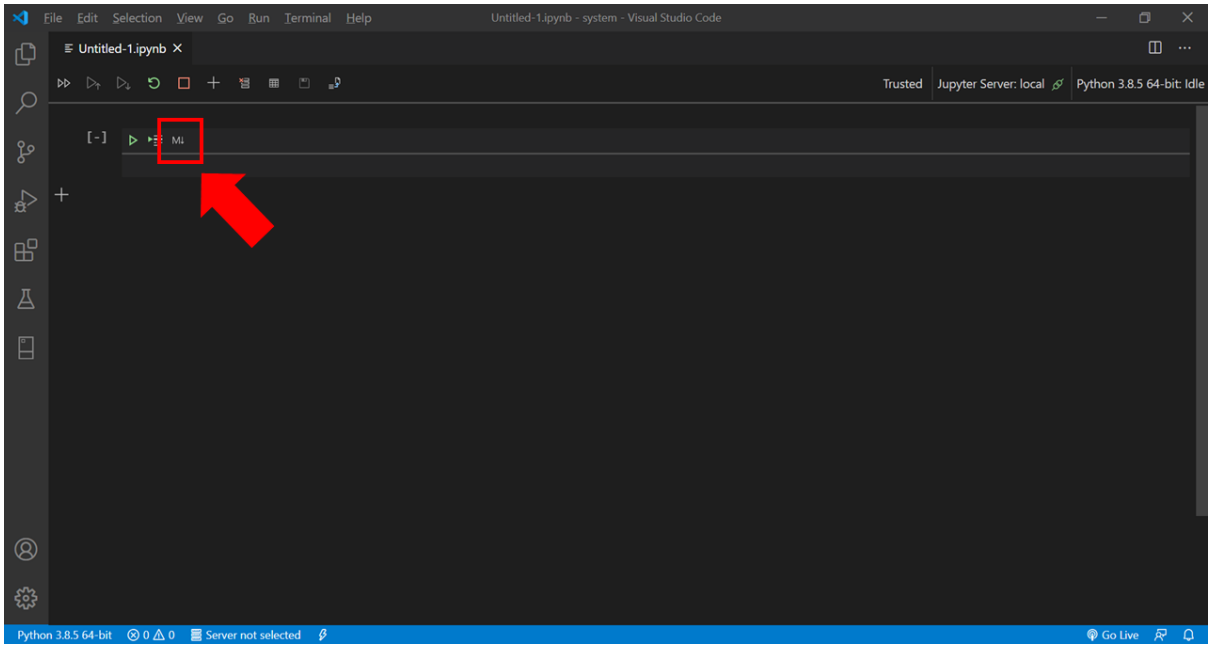


Depois, selecione o interpretador Python que você instalou em sua máquina.



3) Clique no ícone da linguagem Markdown para criar uma nova célula para texto.

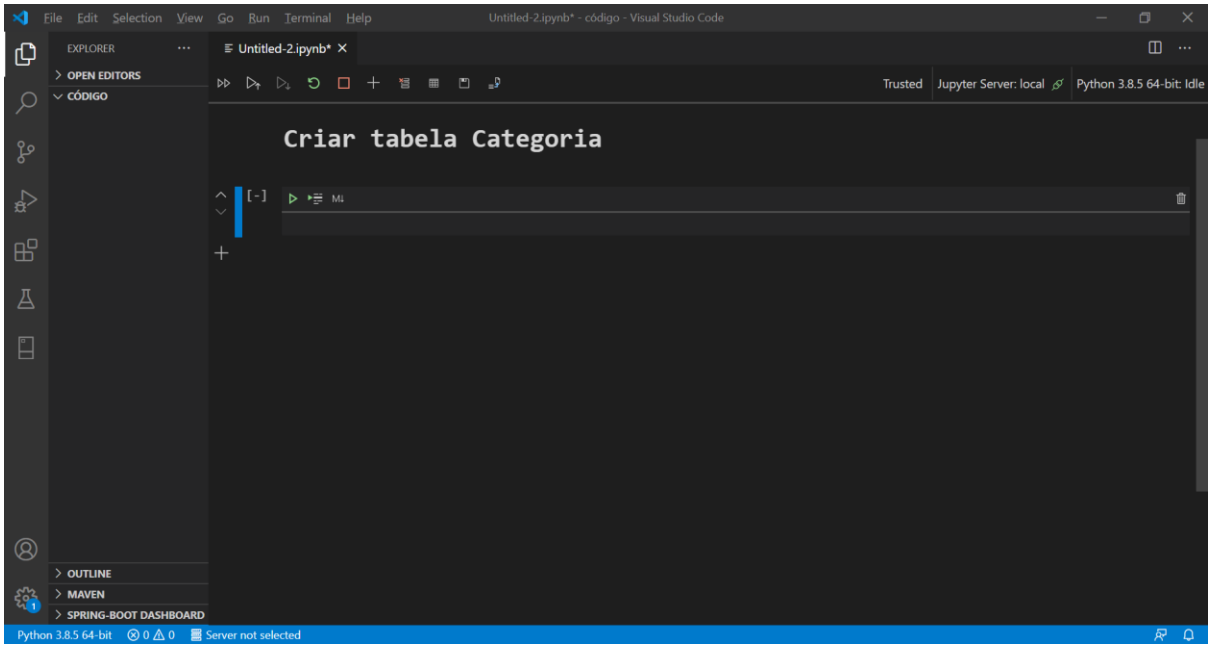
Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02



4) Na nova célula, crie o título:

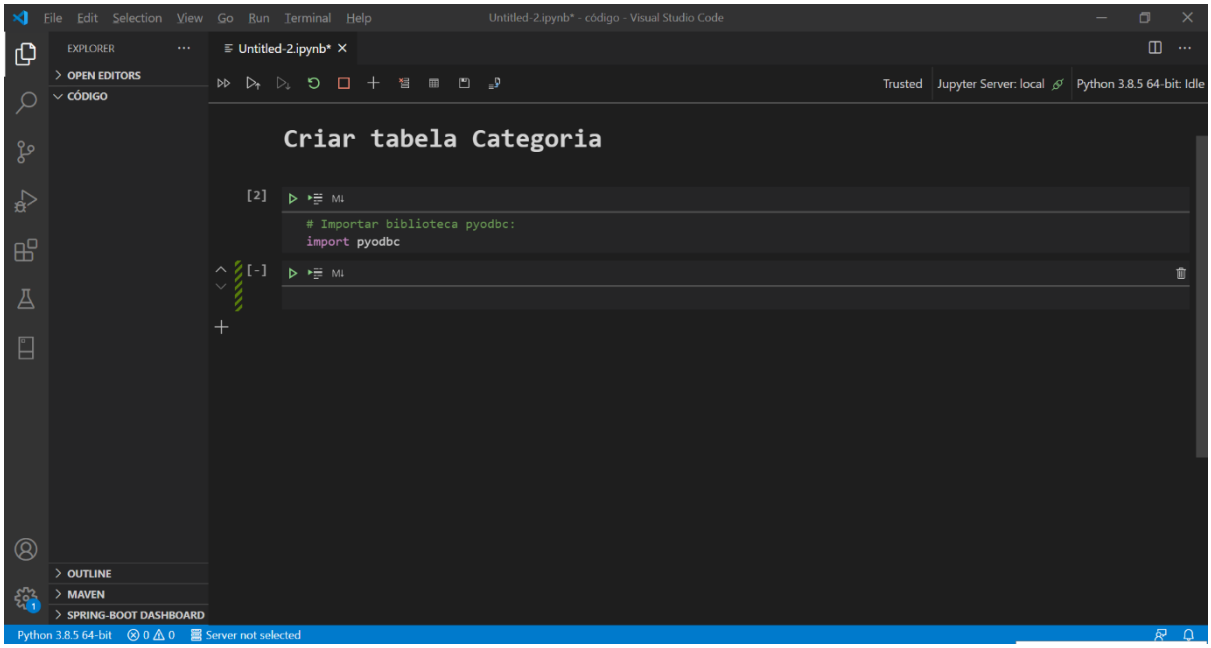
Criar tabela Categoria:

Em seguida, clique em Shift + Enter, para executar a código Markdown e já criar uma nova célula de código.



5) Na nova célula, informe o código abaixo. Em seguida, clique em Shift + Enter, para executar a importação da biblioteca “pyodbc” e já criar uma nova célula de código.

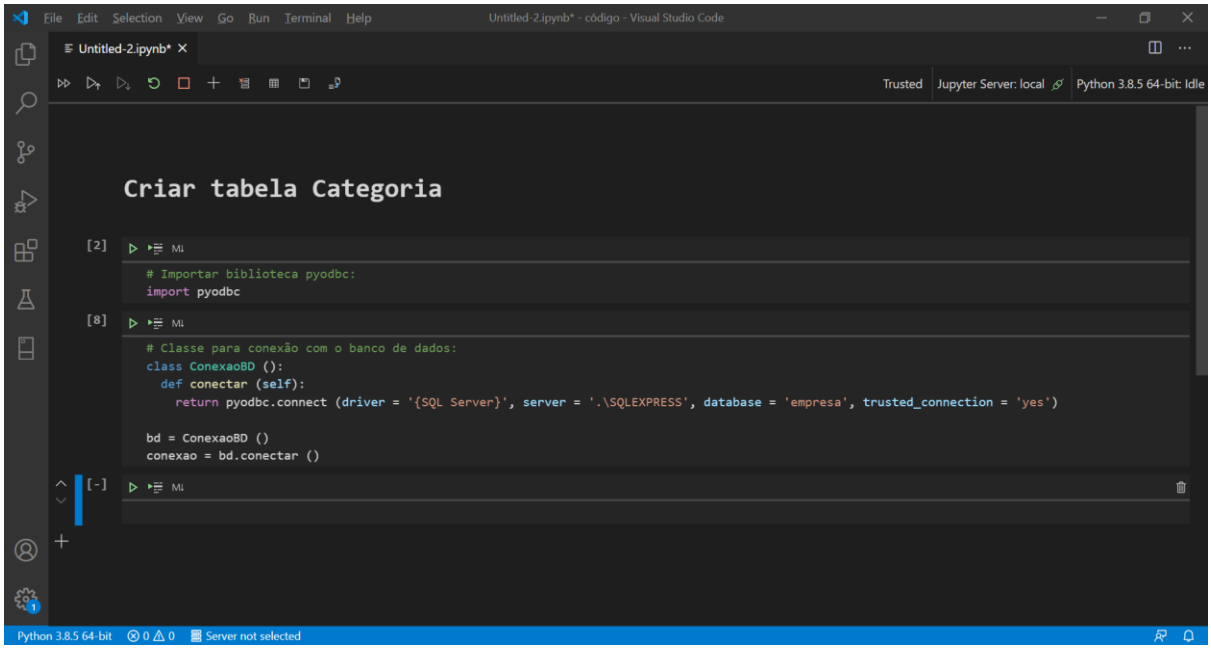
```
# Importar biblioteca pyodbc:

import pyodbc
```



6) Na nova célula, execute o código abaixo para definir a classe para conexão com o banco de dados, instanciar um objeto dessa classe e pegar uma conexão com o banco.

```
# Classe para conexão com o banco de dados:

class ConexaoBD ():

  def conectar (self):

    return pyodbc.connect (driver = '{SQL Server}', server = '.\SQLEXPRESS', database = 'empresa', trusted_connection = 'yes')


# Pegar uma conexão com o banco de dados:

bd = ConexaoBD ()

conexao = bd.conectar ()
```

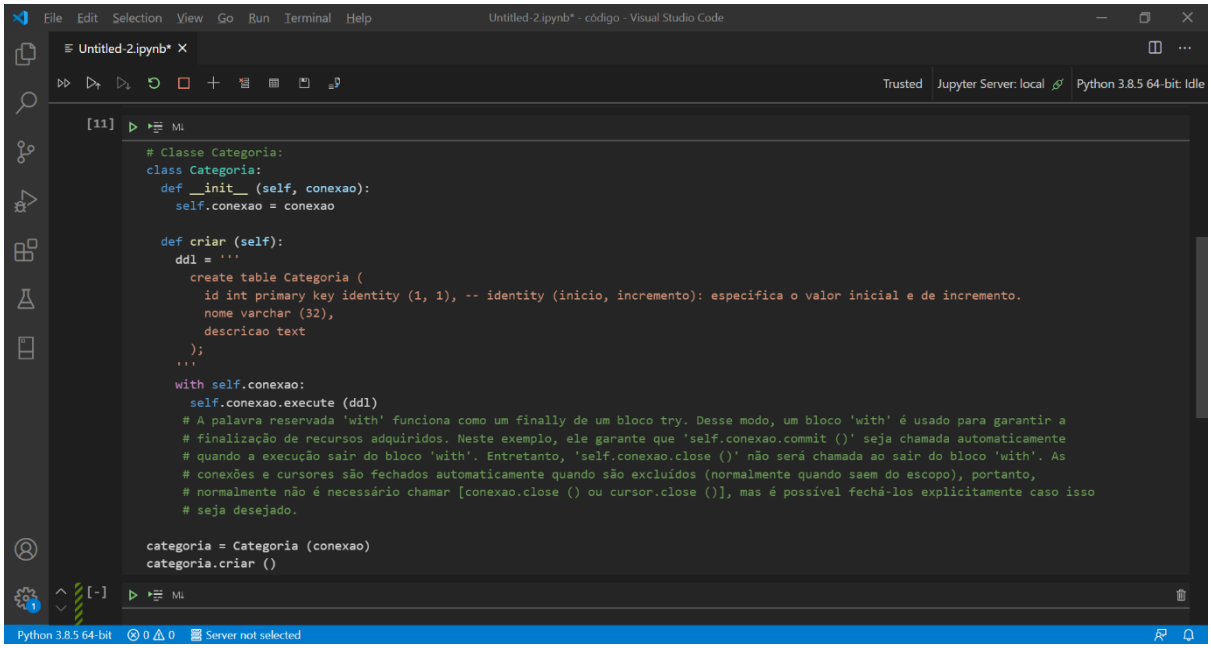
Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02



7) Na nova célula, execute o código abaixo para criar a classe Categoria, instanciar um objeto dessa classe e criar a tabela Categoria no banco de dados.

```
# Classe Categoria:

class Categoria:

 def __init__ (self, conexao):

  self.conexao = conexao


 def criar (self):

  ddl = '''

   create table Categoria (

    id int primary key identity (1, 1), -
- identity (inicio, incremento): especifica o valor inicial e de incremento.
```

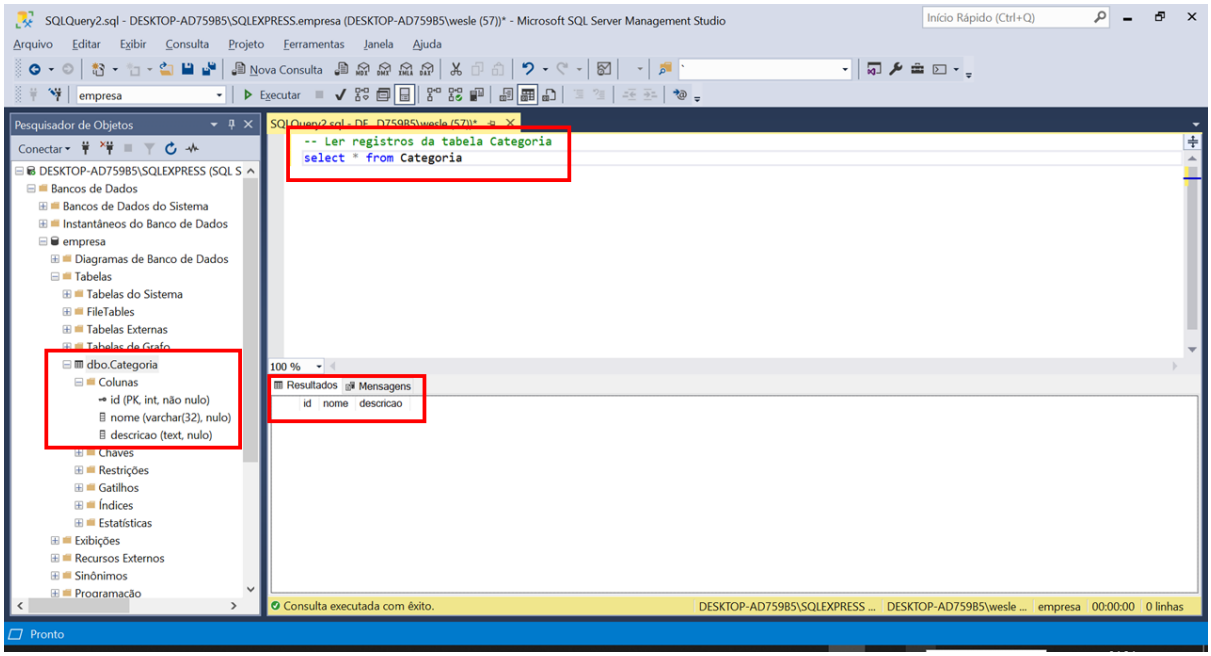
```
    nome varchar (32),
    descricao text
  );
'''
with self.conexao:
  self.conexao.execute (ddl)
```



OBS: a palavra reservada “with” funciona como uma instrução “finally” de um bloco “try”. Desse modo, ela é usada para garantir a finalização de recursos adquiridos. Neste exemplo, ela garante que “self.conexao.commit ()” seja chamado automaticamente quando a execução sair do bloco “with”. Entretanto, “self.conexao.close ()” não será chamado ao sair do bloco “with”. As conexões e os cursores são fechados automaticamente quando são excluídos (normalmente quando saem de escopo). Portanto, normalmente não é necessário chamar [conexao.close () ou cursor.close ()], mas é possível fechá-los explicitamente caso isso seja desejado.

8) No SSMS, observe que a tabela Categoria foi criada:

Centro Universitário UNA
Graduação – TI e Engenharias
Análise de Dados e Big Data
Prática de Laboratório
Diego Augusto de Faria Barros, Tadeu Moreira Perona, Thiérs Hofman do Bom Conselho, Wesley Dias Maciel
2020/02



OBS: caso necessário, clique no ícone de “refresh” do SSMS para visualizar a tabela criada.

Exercício

1) Em um notebook Jupyter, crie as classes correspondentes para criação das demais tabelas descritas na especificação do banco de dados. Os scripts para criação das tabelas são apresentados abaixo.

```
CREATE TABLE Categoria (
  id int primary key identity (1, 1), -- identity (inicio, incremento): especifica o
valor inicial e de incremento dos valores gerados para a primary key.
  nome varchar (32),
  descricao text
);

CREATE TABLE Vendedor (
  cpf varchar (14) primary key,
  nome varchar (128),
  email varchar (128),
  telefone varchar (20),
  endereco varchar (256)
);

CREATE TABLE Produto (
  id int primary key identity (1, 1), -- identity (inicio, incremento): especifica o
valor inicial e de incremento dos valores gerados para a primary key.
  nome varchar (128),
  descricao text,
```

```
  marca varchar (32),
  id_categoria int foreign key references Categoria (id)
);

CREATE TABLE VendedorProduto (
  id int primary key identity (1, 1), -- identity (inicio, incremento): especifica o valor inicial e de incremento dos valores gerados para a primary key.
  cpf_vendedor varchar (14) foreign key references Vendedor (cpf),
  id_produto int foreign key references Produto (id),
  quantidade int,
  preco decimal
);
```

2) Altere as classes criadas, incluindo, em cada uma delas, o método responsável por ler todos os registros da tabela correspondente. Exemplo:

```
-- Ler os registros da tabela Categoria:
select * from Categoria
```