Entrega de Teste referente a base de dados Online_Retails feita

por Bruno Dezorzi 06/11/2024

PYTHON

Primeiro arquivo se refere a importação dos dados usando python para inserir no postgres utilizando sqlalchemy

```python
import pandas as pd
from sqlalchemy import create_engine, Column, String, Integer
from sqlalchemy.orm import declarative_base, sessionmaker


#df = pd.read_excel("C:\Projetos\Data_Science\Projetos_Data_Science\Online_Retail\Online_Retail.xlsx", sheet_name='Online Retail',engine='openpyxl')
# df.to_csv("C:/Projetos/Data_Science/Projetos_Data_Science/Online_Retail/Online_Retail_at.csv", index=False, encoding='utf-8')


df = pd.read_csv("C:/Projetos/Data_Science/Projetos_Data_Science/Online_Retail/Online_Retail_at.csv", encoding='utf-8')
display(df.head())
df.info()
```

Python

```python
df.isnull().sum()
```

Python

```python
df[df["Description"].isnull()]
```

Python

```python
df.isna().sum()
```

Python

```python
df = df.astype(str)
```

Python

```python
DATABASE_URL = "postgresql://postgres:123456@localhost:5432/online_retail"
engine = create_engine(DATABASE_URL)
Base = declarative_base()
```

```python
class OnlineRetail(Base):
    __tablename__ = 'online_retail'
    __table_args__ = {'extend_existing': True}
    id = Column(Integer, primary_key=True)
    invoiceno = Column(String, nullable=True)
    stockcode = Column(String, nullable=True)
    description = Column(String, nullable=True)
    quantity = Column(String, nullable=True)
    invoicedate = Column(String, nullable=True)
    unitprice = Column(String, nullable=True)
    customerid = Column(String, nullable=True)
    country = Column(String, nullable=True)
```

Python

```python
Base.metadata.create_all(engine)

Session = sessionmaker(bind=engine)
session = Session()

session.query(OnlineRetail).delete()


values = [
    OnlineRetail(
        invoiceno=row['InvoiceNo'],
        stockcode=row['StockCode'],
        description=row['Description'],
        quantity=row['Quantity'],
        invoicedate=row['InvoiceDate'],
        unitprice=row['UnitPrice'],
        customerid=row['CustomerID'],
        country=row['Country']
    )
    for index, row in df.iterrows()
]


session.add_all(values)
rows_added = len(values)

try:
    session.commit()  # Confirma a transação
    print(f"{rows_added} linhas foram adicionadas.")
except Exception as e:
    print(f"Erro ao inserir dados: {e}")
    session.rollback()  # Desfaz a transação em caso de erro
finally:
    session.close()  # Fecha a sessão
```

Python

SQL

Aqui vai seguir os scripts em SQL para a criação a tabela e inserção em outra tabela com os dados em suas tipagens corretas

```sql
DROP TABLE IF EXISTS online_retail;

CREATE TABLE online_retail (
id serial PRIMARY KEY,
InvoiceNo VARCHAR(225),
StockCode   VARCHAR(225),
Description VARCHAR(225),
Quantity    VARCHAR(225),
InvoiceDate VARCHAR(225),
UnitPrice   VARCHAR(225),
CustomerID  VARCHAR(225),
Country VARCHAR(225)
);

--  script python -> IMPORTACAO

DROP TABLE IF EXISTS online_retail_refeito;

CREATE TABLE online_retail_refeito (
id serial PRIMARY KEY,
nm_fatura VARCHAR(225),
cd_estoque  VARCHAR(225),
descricao VARCHAR(225),
quantidade  INTEGER,
dt_fatura  TIMESTAMP,
vl_unitario FLOAT,
id_cliente  integer,
pais VARCHAR(225)
);
```

```sql
INSERT INTO online_retail_refeito (
    nm_fatura,
    cd_estoque,
    descricao,
    quantidade,
    dt_fatura,
    vl_unitario,
    id_cliente,
    pais
)
SELECT
    invoiceno,
    stockcode,
    description,
    CAST(quantity AS integer),
    CAST(invoicedate AS timestamp),
    CAST(unitprice AS float),
    CAST(
        CASE
            WHEN customerid = 'nan' THEN '0' -- Substitui "nan" por 0
            ELSE REPLACE(customerid, '.', '') -- Remove pontos
        END AS integer
    ),
    country
FROM
    public.online_retail
;

-- Ajustando casos de NaN em descrições que não possuem dados validos


UPDATE online_retail_refeito t1
SET descricao = t2.descricao
FROM (
    SELECT
        cd_estoque,
        descricao
    FROM online_retail_refeito
    WHERE descricao NOT LIKE 'nan'
) AS t2
WHERE t1.cd_estoque = t2.cd_estoque
  AND t1.descricao LIKE 'nan';
```

Durante a inserção na nova tabela(definitiva), eu fui realizando umas consultas para achar valores inconsistentes e faltantes, alguns eliminei e tem um caso que tratei com updates descrições que faltaram (caso UPDATE último script)

CONSULTAS

```sql
SELECT
    COUNT(*) FILTER (WHERE nm_fatura IS NULL) AS nm_fatura_null_count,
    COUNT(*) FILTER (WHERE cd_estoque IS NULL) AS cd_estoque_null_count,
    COUNT(*) FILTER (WHERE descricao IS NULL) AS descricao_null_count,
    COUNT(*) FILTER (WHERE quantidade IS NULL) AS quantidade_null_count,
    COUNT(*) FILTER (WHERE dt_fatura IS NULL) AS dt_fatura_null_count,
    COUNT(*) FILTER (WHERE vl_unitario IS NULL) AS vl_unitario_null_count,
    COUNT(*) FILTER (WHERE id_cliente IS NULL) AS id_cliente_null_count,
    COUNT(*) FILTER (WHERE pais IS NULL) AS pais_null_count
FROM
    online_retail_refeito;



-- Verificar se tem algum registro com Cliente Nulo
SELECT
    *
FROM online_retail_refeito or2
WHERE or2.id_cliente  IS NULL



SELECT
    count(*)
FROM online_retail_refeito or2
WHERE 1 = 1
AND id_cliente = 0
-- AND id_cliente <= 0 ambos o mesmo resultado, ID CLIENTE ZERO TEM 135080 REGISTROS



SELECT
    count(*)
FROM online_retail_refeito or2
WHERE 1 = 1
AND id_cliente != 0



-- Quantidade de registros com quantidade negativa
SELECT
    quantidade ,
    count(*)
FROM online_retail_refeito
WHERE quantidade <= 0
GROUP BY quantidade
ORDER BY count(*) DESC
```

```sql
SELECT
    quantidade ,
    count(*)
FROM online_retail_refeito
WHERE quantidade = 0
GROUP BY quantidade
ORDER BY count(*) DESC

--  Analiasando semelhanças entre cd_estoque e descricao
-- CONCLUSÃO: cada cd_estoque tem uma descricao, então, para descrições em branco, o ideal seria fazer um update caso tivesse casos com cd_estoque e sem descricao
-- É O CASO!
SELECT
    DISTINCT cd_estoque ,
    descricao
FROM online_retail_refeito orr
ORDER BY descricao desc


SELECT count(*) FROM online_retail_refeito orr WHERE descricao IS null




SELECT
    *,
    COUNT(*) FILTER (WHERE * IS NULL) AS null_count
FROM
    online_retail_refeito
GROUP BY
    *;

-- Provavel método duplicado
   SELECT
   *
   FROM online_retail_refeito orr
   WHERE nm_fatura like '580877'
   AND id_cliente = 172500
   AND cd_estoque LIKE '84347'




   SELECT
   *
   FROM online_retail_refeito WHERE descricao LIKE 'nan'

   SELECT * FROM online_retail_refeito WHERE cd_estoque LIKE '84251J'
```

Após isso eu criei a tabela dCalendario e fiz duas view básicas para o Power BI

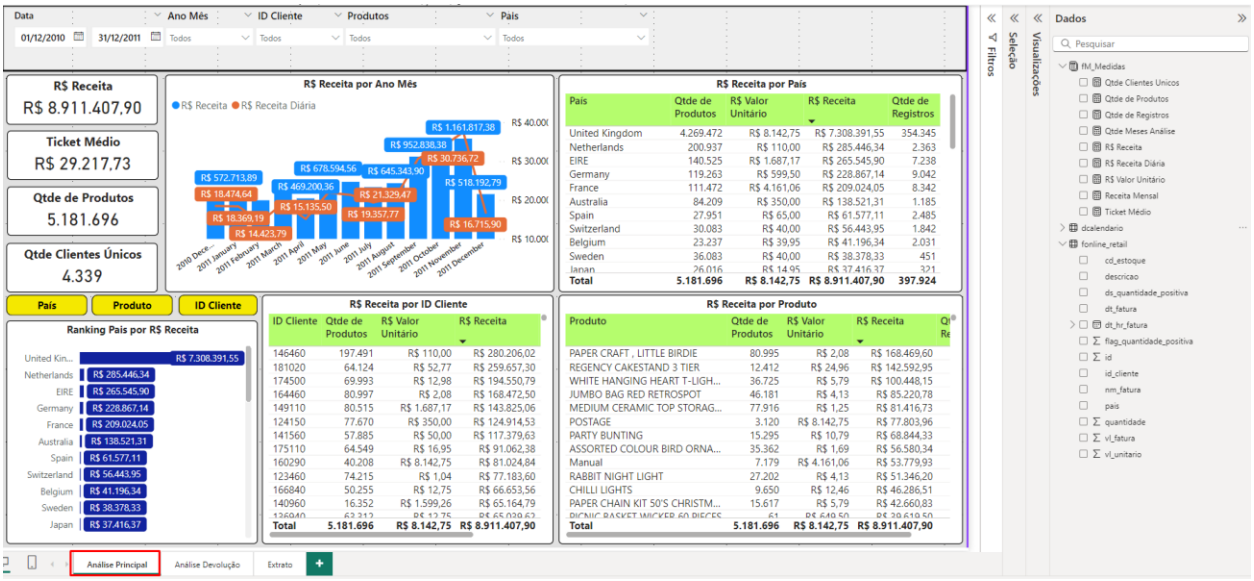
```sql
DROP VIEW IF EXISTS view_fonline_retail;
CREATE VIEW view_fonline_retail
AS

SELECT
    ID,
    NM_FATURA,
    CD_ESTOQUE,
    DESCRICAO,
    QUANTIDADE,
    DT_FATURA AS DT_HR_FATURA,
    DT_FATURA :: DATE AS DT_FATURA,
    VL_UNITARIO,
    (VL_UNITARIO * QUANTIDADE) VL_FATURA,
    ID_CLIENTE,
    PAIS,
    CASE
        WHEN QUANTIDADE > 0  THEN 1
        ELSE 0
    END AS FLAG_QUANTIDADE_POSITIVA,
    CASE
        WHEN QUANTIDADE > 0  THEN 'Receita'
        ELSE 'Prejuízo'
    END AS DS_QUANTIDADE_POSITIVA
FROM public.online_retail_refeito
WHERE ID_CLIENTE != 0
AND DESCRICAO NOT LIKE 'nan';
```

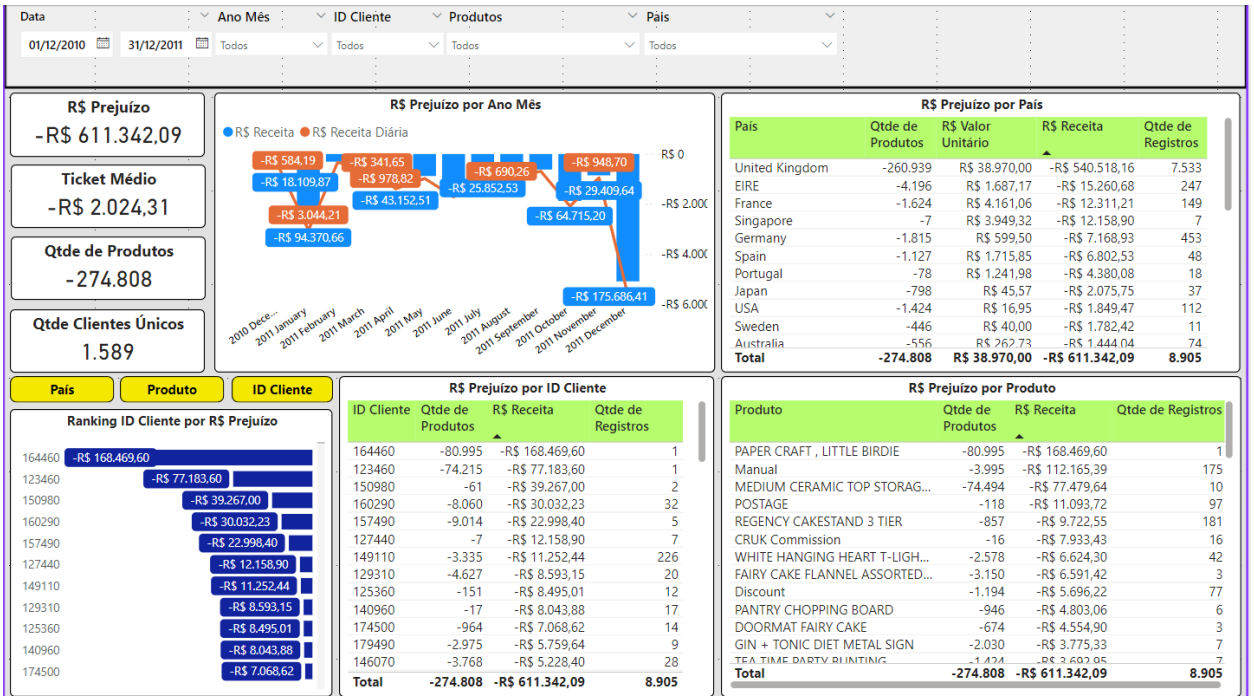
```sql
DROP VIEW IF EXISTS view_dcalendario;
CREATE VIEW view_dcalendario
AS

SELECT
    dcalendario.*,
    RIGHT(TO_CHAR(data_mes_final, 'YYYY-MM-DD'), 2) AS dias_mes
FROM dcalendario
```

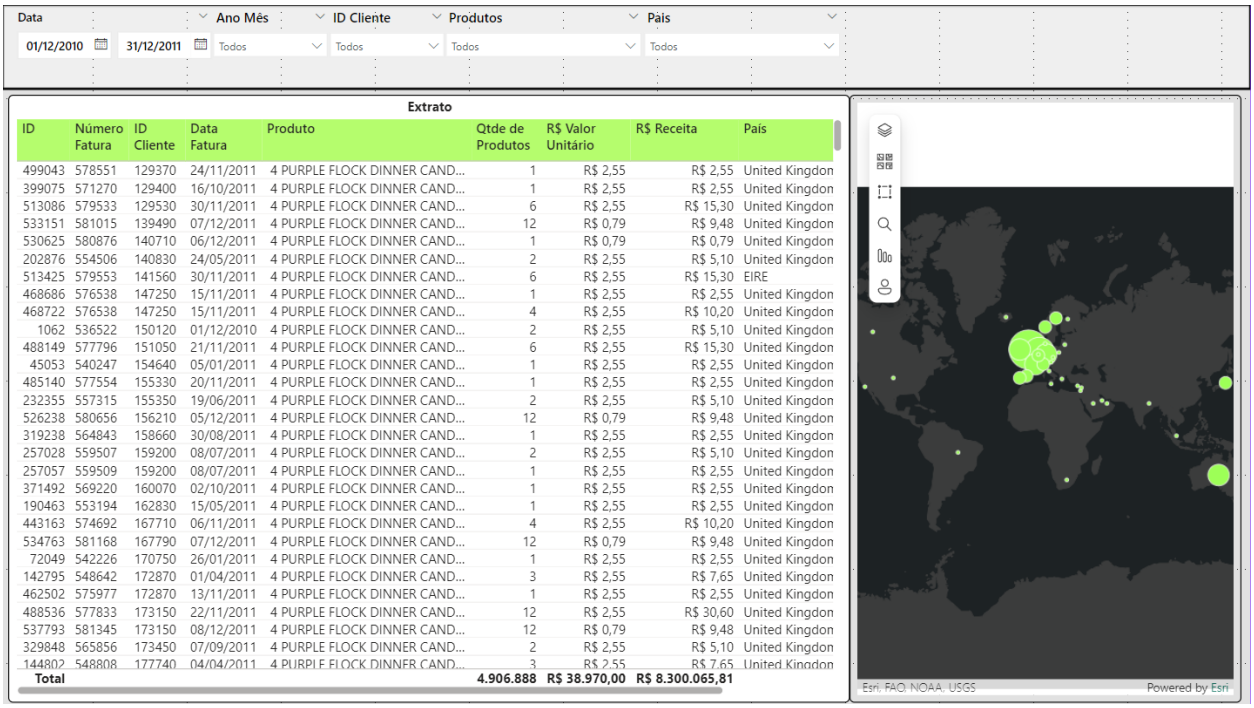
POWER BI



Essa primeira página é focada em análises com receita positiva, os botões em amarelo a esquerda são indicadores que mudam a tabela abaixo



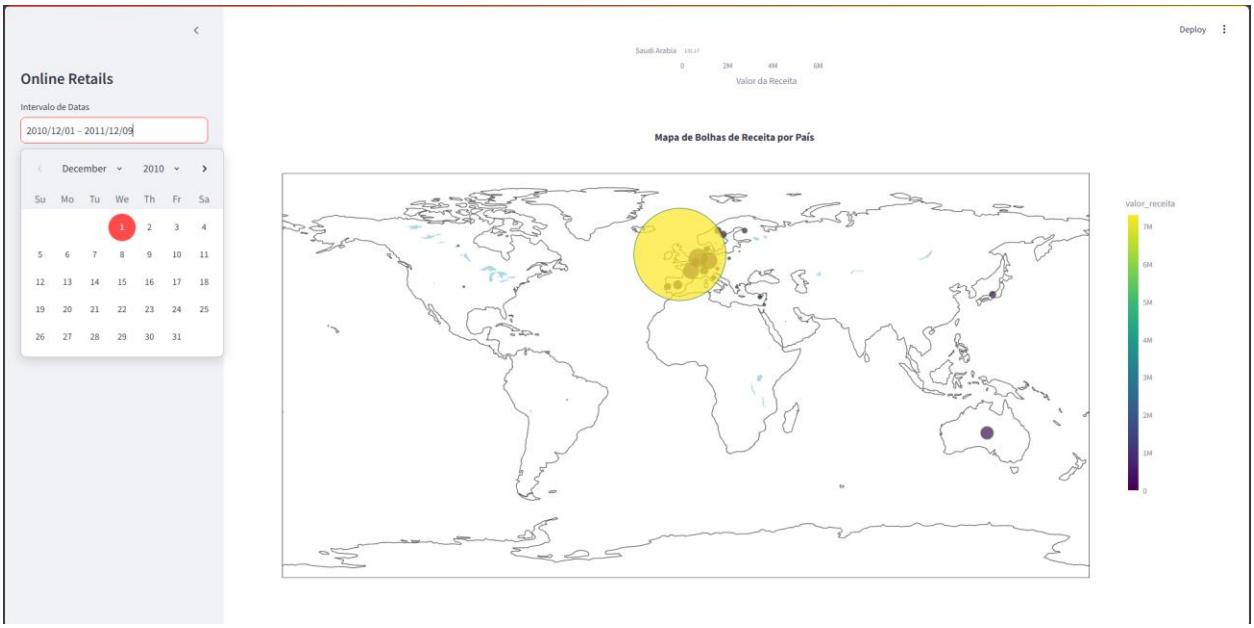
Voltada para a análise de casos de receita negativa

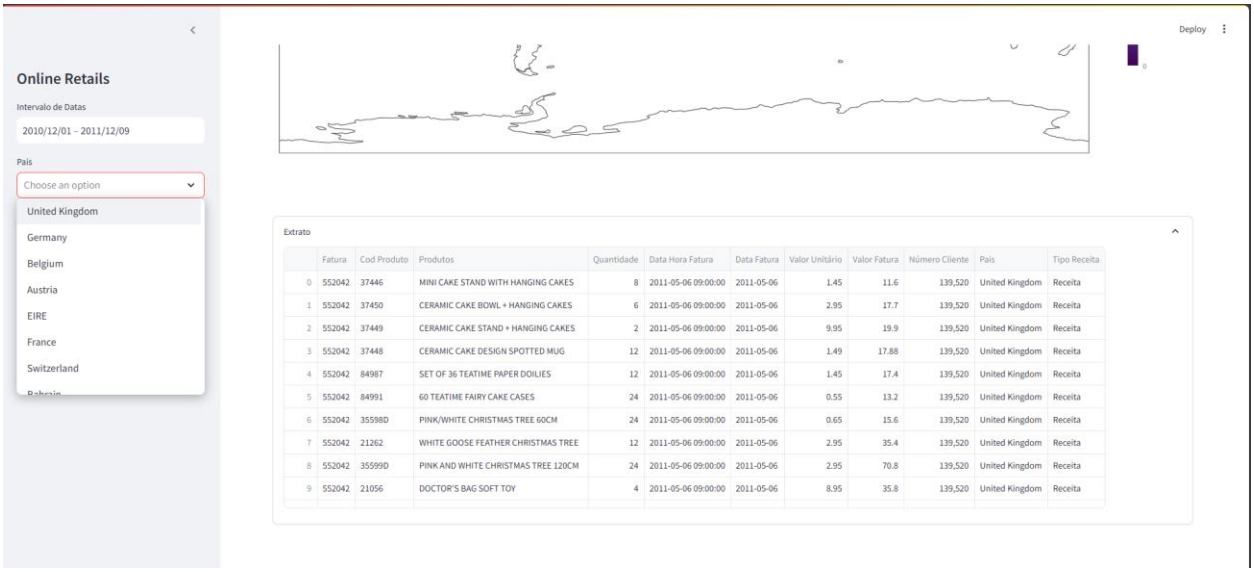


Um extrato e um mapa de bolha mostrando a concentração de R$ Receita

PYTHON

Por fim, a análise em streamlit, foi mais difícil pois não tenho usei muito, basicamente repliquei as ideias do Power BI, vou deixar o script anexado





	Fatura	Cod Produto	Produtos	Quantidade	Data Hora Fatura	Data Fatura	Valor Unitário	Valor Fatura	Número Cliente	Pais	Tipo Receita
0	552042	37446	MINI CAKE STAND WITH HANGING CAKES	8	2011-05-06 09:00:00	2011-05-06	1.45	11.6	139,520	United Kingdom	Receita
1	552042	37450	CERAMIC CAKE BOWL + HANGING CAKES	6	2011-05-06 09:00:00	2011-05-06	2.95	17.7	139,520	United Kingdom	Receita
2	552042	37449	CERAMIC CAKE STAND + HANGING CAKES	2	2011-05-06 09:00:00	2011-05-06	9.95	19.9	139,520	United Kingdom	Receita
3	552042	37448	CERAMIC CAKE DESIGN SPOTTED MUG	12	2011-05-06 09:00:00	2011-05-06	1.49	17.88	139,520	United Kingdom	Receita
4	552042	84987	SET OF 36 TEATIME PAPER DOILIES	12	2011-05-06 09:00:00	2011-05-06	1.45	17.4	139,520	United Kingdom	Receita
5	552042	84991	60 TEATIME FAIRY CAKE CASES	24	2011-05-06 09:00:00	2011-05-06	0.55	13.2	139,520	United Kingdom	Receita
6	552042	35598D	PINK/WHITE CHRISTMAS TREE 60CM	24	2011-05-06 09:00:00	2011-05-06	0.65	15.6	139,520	United Kingdom	Receita
7	552042	21262	WHITE GOOSE FEATHER CHRISTMAS TREE	12	2011-05-06 09:00:00	2011-05-06	2.95	35.4	139,520	United Kingdom	Receita
8	552042	35599D	PINK AND WHITE CHRISTMAS TREE 120CM	24	2011-05-06 09:00:00	2011-05-06	2.95	70.8	139,520	United Kingdom	Receita
9	552042	21056	DOCTOR'S BAG SOFT TOY	4	2011-05-06 09:00:00	2011-05-06	8.95	35.8	139,520	United Kingdom	Receita