Aula # 2 - Analisando os dados

Ready ☑

▼ 1.0. A união entre tabelas do banco de dados

Usando a linguagem SQL é possível fazer a união de mais de uma tabela, para formar a tabela de origem dos dados, do qual serão selecionadas as colunas e as linhas.

Para fazer a união das tabelas é necessário identificar as colunas comuns entre as duas tabelas e usar os seguintes comandos:

```
SELECT
    A.coluna1,
    A.coluna2,
    B.coluna1,
    B.coluna2
FROM
    tabela A LEFT JOIN tabela B ON (A.coluna3 = B.coluna3)
WHERE
    A.coluna1 > 1000
```

▼ 1.1 A tabela A (tabela_transacao)

ID da Transação	Data da compra	Nome do Produto	Valor da transação	Moeda	Nome do Comprador	Nome do Vendedor
AZ102931	29/04/2022	Fone de Ouvido	650	Real	Meigarom	Amazon

▼ 1.2 A tabela B (tabela_produto)

Nome do Produto	Marca	Preço	Material	Peso	Dimensões	Custo de fabricação	Garan
Fone de Ouvido	Bozer	650	Alumínio	30g	40cm x 2mm	100	1 ano

▼ 1.3 União da tabela A com a B

```
SELECT
    tt.nome_produto,
    tt.data_compra,
    tt.valor_transacao,
    tp.marca,
    tp.peso,
    tp.material
FROM tabela_transacao tt LEFT JOIN tabela_produto tp
    ON(tt.nome_produto = tp.nome_produto)
WHERE
    tp.preco > 650
```

▼ 1.4 Coletando os dados para análise

Para executar o próximo passo do plano de solução do problema de negócio, precisamos coletar os dados do banco de dados.

```
# abrir a conexao com o banco de dados
import sqlite3
import pandas as pd
conn = sqlite3.connect( "database.db" )


# consulta dos dados no banco de dados
consulta_atividade = """
  SELECT
    *
  FROM
    flight_activity fa LEFT JOIN flight_loyalty_history flh
            ON(fa.loyalty_number = flh.loyalty_number)
"""

df_atividade = pd.read_sql_query( consulta_atividade, conn )

# comando para mostrar a tabela resultante da consulta
df_atividade.head()

# fecha a conexao com o banco de dados
conn.close()
```

2.0. O que são bibliotecas

As bibliotecas em Python são comandos pré-escrito que nos permitem realizar diversas tarefas sem precisar reinventar a roda.

Elas fornecem funcionalidades adicionais ao Python, permitindo-nos economizar tempo e esforço na implementação de determinadas tarefas.

As bibliotecas são criadas e mantidas pela comunidade Python e ao usar bibliotecas, podemos aproveitar o trabalho de outros desenvolvedores e acelerar o desenvolvimento de nossos próprios projetos.

2.1 A biblioteca Pandas

A biblioteca Pandas é uma das principais bibliotecas utilizadas no Python para análise e manipulação de dados. Ela oferece estruturas de dados eficientes e fáceis de usar, como o DataFrame, que permite trabalhar com dados tabulares de forma intuitiva.

Com o Pandas, é possível importar dados de diferentes fontes, como arquivos CSV e Excel, e realizar operações como filtragem, ordenação, agrupamento e transformação dos dados. Além disso, a biblioteca também fornece ferramentas para lidar com valores ausentes e realizar cálculos estatísticos.

A simplicidade e a versatilidade do Pandas tornam-no uma escolha popular para análise de dados em várias áreas, como ciência de dados, finanças, economia e pesquisa acadêmica.

2.1.1 Exemplos do uso da biblioteca Pandas

Para usar os comandos escritos em uma biblioteca, você precisa importar ela dentro do notebook.

```
# Sintaxe para importar bibliotecas no Python
IMPORT <nome da biblioteca> AS <apelino>
```

```
# Comando: Importar a biblioteca Pandas
import pandas as pd

# Comando: Import a biblioteca SQlite
import sqlite3 as sl3

# Usar o comando "connect" da biblioteca sqlite3 para
#     conectar ao banco de dados e guarde a conexão dentro da
#     variável "conn"

conn = sl3.connect( "database.db" )
```

3.0. Inspecionando os dados

3.1. Coletando os dados

```
# importando as bibliotecas
import sqlite3
import pandas as pd

# abrindo a conexão com o banco de dados
conn = sqlite3.connect( "database.db" )

# coletando os dados
consulta_atividade = """
    SELECT *
    FROM
        flight_activity fa LEFT JOIN flight_loyalty_history flh
            ON (fa.loyalty_number = flh.loyalty_number )
"""

# executando a consulta
df_atividade = pd.read_sql_query( consulta_atividade, conn )
```

3.2. Inspecionando a planilha de dados

```
# verificando a quantidade de linhas
numero_linhas = df_atividade.shape[0]
print( 'O numero de linhas eh:', numero_linhas )

# verificando a quantidade de colunas
numero_colunas = df_atividade.shape[1]
print( 'O numero de linhas eh:', numero_linhas )

# descobrindo as informacoes gerais sobre a planilha de dados
df_atividade.info()

# Somar a colunas "total_flights"
df_atividade.loc[:, 'total_flights'].sum()

# Somar a colunas "distance"
df_atividade.loc[:, 'distance'].mean()
```

```
# Valor mínimo de salário
df_atividade.loc[:, 'distance'].min()

# Valor máximo de salário
df_atividade.loc[:, 'distance'].max()

# checando o número de dados faltante nas colunas
df_atividade.isna().sum()
```

3.3. Seleção de linhas e colunas

```
# Comando do Pandas para selecionar linhas e colunas
#df = df1.iloc[linhas, colunas]

# selecionando colunas de uma planilha
colunas = ['year', 'month', 'flights_booked', 'flights_with_companions',
'total_flights', 'distance', 'points_accumulated', 'points_redeemed',
'dollar_cost_points_redeemed', 'salary', 'clv', 'enrollment_year',
'enrollment_month', 'loyalty_card']

df_dados_limpos = df_atividade.loc[:, colunas]
```

4.0. Preparando os dados para treinamento do algoritmo

4.1. Dados para treinamento

Os algoritmos de Machine Learning aceitam alguns tipos de dados e rejeitam outros tipos. Uma das etapas cruciais da preparação dos dados para treinamento é a limpeza e preparação dos dados.

Nessa etapa, devemos realizar as seguintes operações.

1. Remoção ou substituição linha que contém dados faltantes.
2. Remoção das colunas com dados sem variabilidade.
3. Colunas com valores altos de correlação.
4. Transformação dos dados categóricos em dados numéricos.

4.2. Limpando a base de dados

```
# Numero de dados faltantes
df_atividade.isna().sum()

# selecionando somente as colunas numéricas
colunas=["year", "month", "flights_booked", "flights_with_companions",
                "total_flights", "distance", "points_accumulated", "salary",
                "clv"]

df_colunas_numericas = df_atividade.loc[:, colunas]

# removendo linhas com alguma coluna vazia.
df_dados_limpos = df_colunas_numericas.dropna()

# verificando o numero de linhas vazias
df_dados_limpos.isna().sum()
```

▼ 5.0 Exercícios

1. Selecionar os números do cartão de fidelidade dos passeiros, a cidade e o gênero, dos passageiros que tem o cartão Star de fidelidade, mas nunca realizaram nenhuma viagem de avião (Dica: “loyalty_card” = “Star” e “distance” = 0)

   ▼ Resposta:

   ```
   SELECT
       fa.loyalty_number,
     flh.city,
     flh.gender,
     flh.loyalty_card
   FROM
     flight_activity fa LEFT JOIN flight_loyalty_history flh
       ON (fa.loyalty_number = flh.loyalty_number)
   WHERE
     fa.distance = 0 AND flh.loyalty_card = "Star"
   ```

2. Selecionar os números do cartão de fidelidade, o gênero e a cidade de todos os passageiros do sexo feminino que moram na cidade de Toronto, fizeram mais de 30 viagens no total e tem o cartão de fidelidade do tipo Aurora.

   ▼ Resposta:

   ```
   SELECT
       fa.loyalty_number,
     fa.total_flights,
     flh.city,
     flh.gender,
     flh.loyalty_card
   FROM
     flight_activity fa LEFT JOIN flight_loyalty_history flh
       ON (fa.loyalty_number = flh.loyalty_number)
   WHERE
     flh.gender = "Female"
     AND flh.city = "Toronto"
     AND fa.total_flights = 30
     AND flh.loyalty_card = "Aurora"
   ```

3. Selecionar os números do cartão de fidelidade, o tipo do cartão, o genero e os pontos acumulados, dos passageiros com salário acima de 13200, estado civil como casado e nível acadêmico como mestrado e número de voos agendados igual ao número total de voos.

   ▼ Resposta:

   ```
   SELECT
       fa.loyalty_number,
     flh.loyalty_card,
     flh.gender,
     fa.points_accumulated
   FROM
       flight_activity fa LEFT JOIN flight_loyalty_history flh
       ON(fa.loyalty_number = flh.loyalty_number)
   WHERE
       flh.salary > 132000
   ```

```
    AND flh.marital_status = "Married"
    AND flh.education = "Master"
    AND ( fa.total_flights = fa.flights_booked )
```

4. Selecionar os números do cartão de fidelidade, o tipo do cartão, o genero e os pontos acumulados, dos passageiros com salário acima de 13200, estado civil como casado e nível acadêmico como mestrado e número de voos agendados igual ao número total de voos.

   ▼ Resposta:

```
SELECT
    fa.loyalty_number,
  flh.loyalty_card,
  flh.gender,
  fa.points_accumulated
FROM
    flight_activity fa LEFT JOIN flight_loyalty_history flh
    ON ( fa.loyalty_number = flh.loyalty_number )
WHERE
    flh.salary > 132000
    AND flh.marital_status = "Married"
    AND flh.education = "Master"
    AND ( fa.total_flights = fa.flights_booked )
```

5. Qual o valor da soma total da distância percorrida pelos voos registrados na planilha de dados.?

   ▼ Resposta:

```
# Somar a colunas "distance"
df_atividade.loc[:, 'distance'].sum()
```

6. Qual o salário médio dos passageiros?

   ▼ Resposta:

```
# Somar a colunas "distance"
df_atividade.loc[:, 'salary'].mean()
```

7. Qual o valor total de pontos acumulados?

   ▼ Resposta:

```
# Somar a colunas "distance"
df_atividade.loc[:, 'points_accumulated'].sum()
```