**Análise Exploratória dos Dados**

O código SQL está criando várias tabelas em um banco de dados chamado "csgo" para armazenar dados relacionados a partidas de um jogo de tiro chamado "CS:GO" (Counter-Strike: Global Offensive).

As tabelas criadas são as seguintes:

* Tabela "players": Armazena informações dos jogadores, como a data, nome do jogador, equipe, oponente, país, ID do jogador, ID da partida, ID do evento, nome do evento, melhor de quantas partidas, informações sobre os mapas jogados e estatísticas de desempenho dos jogadores.
* Tabela "results": Armazena informações sobre os resultados das partidas, como a data, as duas equipes envolvidas, o mapa jogado, os placares de cada equipe, o vencedor do mapa, o lado inicial escolhido (CT ou T), estatísticas de pontuação por lado e informações sobre o evento e a partida.
* Tabela "economy": Armazena informações sobre a economia do jogo, incluindo a data, o ID da partida, o ID do evento, as duas equipes envolvidas, o formato da partida (melhor de quantas), o mapa jogado, as escolhas iniciais de cada equipe, informações sobre a economia de cada equipe em cada rodada e o vencedor de cada rodada.
* Tabela "picks": Armazena informações sobre as escolhas de mapas feitas pelas equipes, incluindo a data, as duas equipes envolvidas, um indicador para verificar se as equipes foram invertidas, o ID da partida, o ID do evento, o formato da partida, informações sobre o sistema de escolha de mapas, as remoções de cada equipe e as seleções finais de mapas.

Essas tabelas foram criadas para armazenar dados relacionados a partidas de CS:GO e permitir consultas e análises futuras desses dados. Cada tabela tem suas colunas específicas que correspondem aos diferentes aspectos das partidas e estatísticas dos jogadores.

**Utilizando o banco de dados no Jupyter Notebook**

Texto

Descrição gerada automaticamente

O código apresentado realiza as seguintes ações:

Importa os módulos necessários, como mysql.connector, plotly.graph\_objects, plotly.express, os, pandas e numpy.

Define uma função chamada connect que recebe parâmetros de conexão do banco de dados (host, banco de dados, usuário e senha) e tenta estabelecer uma conexão com o banco de dados MySQL usando a biblioteca mysql.connector. Em caso de sucesso, retorna o objeto de conexão.

Define uma função chamada execute\_query que recebe a conexão do banco de dados, uma consulta SQL e, opcionalmente, dados para a consulta. Essa função executa a consulta no banco de dados, busca os resultados e, se houver resultados, os retorna. Em caso de erro, retorna uma mensagem de erro.

Define uma função chamada loop\_result que recebe os resultados de uma consulta SQL e percorre cada resultado, adicionando-o a uma lista. Retorna a lista resultante.

Realiza a conexão com o banco de dados MySQL usando a função connect e armazena o objeto de conexão na variável connection.

Define uma consulta SQL na variável query\_results para selecionar todos os registros da tabela "results".

Define uma lista de colunas na variável columns\_results correspondente às colunas da tabela "results".

Executa a consulta SQL usando a função execute\_query, passando a conexão e a consulta como parâmetros, e armazena os resultados na variável results\_results.

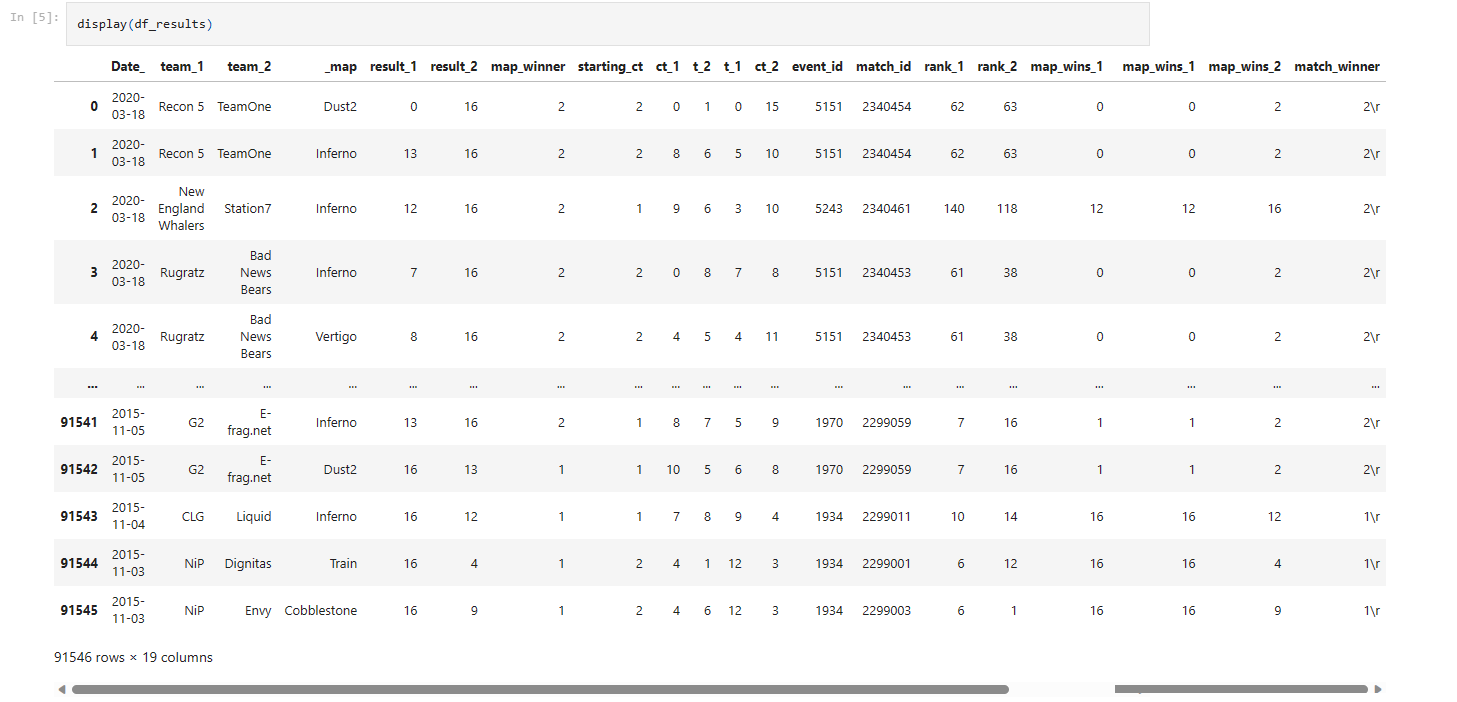
Chama a função loop\_result passando os resultados da consulta para converter os resultados em uma lista.

Cria um DataFrame do pandas chamado df\_results a partir da lista de resultados, usando as colunas definidas anteriormente.

Exibe o DataFrame df\_results utilizando a função display (provavelmente importada de um ambiente interativo como Jupyter Notebook ou IPython).

Em resumo, o código estabelece uma conexão com um banco de dados MySQL, executa uma consulta SQL para obter os resultados da tabela "results" e exibe os resultados em um DataFrame do pandas.

* Consultando DataFrame:



A função display(df\_results) é usada para exibir o DataFrame df\_results de forma mais amigável e interativa em um ambiente como Jupyter Notebook ou Python.

Ao chamar a função display, ela renderiza o DataFrame como uma tabela formatada, facilitando a visualização e análise dos dados. No exemplo fornecido, o DataFrame possui várias colunas (Date\_, team\_1, team\_2, \_map, result\_1, result\_2, map\_winner, starting\_ct, ct\_1, t\_2, t\_1, ct\_2, event\_id, match\_id, rank\_1, rank\_2, map\_wins\_1, map\_wins\_2, match\_winner) e 91546 linhas de dados.

A exibição do DataFrame permite visualizar os dados tabulares de forma organizada, com os nomes das colunas na parte superior e os valores correspondentes em cada linha. Isso facilita a análise, filtragem, ordenação e visualização dos dados contidos no DataFrame, proporcionando uma representação visual dos dados de maneira eficiente.

* Informações da tabela:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

A função df\_results.info() é usada para exibir informações sobre um DataFrame no pandas. Aqui está uma descrição dos resultados fornecidos:

A linha class 'pandas.core.frame.DataFrame' indica que estamos trabalhando com um objeto DataFrame.

A linha RangeIndex: 91546 entries, 0 to 91545 informa que o DataFrame possui 91546 linhas no intervalo de índices de 0 a 91545.

A seção Data columns lista todas as colunas do DataFrame e fornece informações adicionais sobre cada uma delas:

Column: Nome da coluna.

Non-Null Count: Número de valores não nulos na coluna, o que indica a quantidade de dados disponíveis.

Dtype: Tipo de dados da coluna.

A linha memory usage: 13.3+ MB mostra a quantidade de memória usada pelo DataFrame.

No exemplo específico fornecido, o DataFrame df\_results possui 19 colunas com uma variedade de tipos de dados, incluindo inteiros (int64) e objetos (object). Existem 91546 entradas no DataFrame, o que indica que cada coluna possui 91546 valores não nulos. A função df\_results.info() é útil para obter uma visão geral dos dados presentes no DataFrame, incluindo o número de linhas, tipos de dados das colunas e a presença de valores nulos.

* Verificando a presença de valores ausentes:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

A função display(df\_results.isna().sum()) é usada para verificar a presença de valores ausentes (ou nulos) em um DataFrame do pandas. Os resultados exibidos indicam o número de valores ausentes em cada coluna do DataFrame df\_results.

Aqui está uma descrição dos resultados fornecidos:

Cada linha representa uma coluna do DataFrame df\_results.

O nome da coluna é exibido à esquerda, seguido pelo número de valores ausentes naquela coluna.

A última linha, dtype: int64, indica o tipo de dados dos valores contados (neste caso, são inteiros).

No exemplo fornecido, os resultados mostram que não há valores ausentes em nenhuma das colunas do DataFrame df\_results. Todos os campos possuem contagem de valores nulos igual a zero, indicando que não há dados faltantes no conjunto de dados. Isso é útil para garantir que os dados estejam completos antes de realizar análises ou manipulações adicionais.

* Agrupando dados:

Tabela

Descrição gerada automaticamente

A expressão df\_results.groupby('\_map').Date\_.count() está sendo usada para agrupar os dados do DataFrame df\_results pela coluna '\_map' e, em seguida, contar o número de ocorrências da coluna Date\_ em cada grupo. O resultado é uma série que exibe o número de ocorrências de cada valor único na coluna '\_map'.

Aqui está uma descrição dos resultados fornecidos:

Cada linha representa um valor único da coluna '\_map'.

Os valores únicos da coluna '\_map' são exibidos na coluna do índice.

A coluna à direita, intitulada Name: Date\_, dtype: int64, indica que estamos contando o número de ocorrências da coluna Date\_.

Os valores numéricos representam o número de ocorrências de cada valor único na coluna '\_map'.

No exemplo fornecido, os resultados mostram a contagem de ocorrências para cada valor único na coluna '\_map' do DataFrame df\_results. Por exemplo, temos 9226 ocorrências do valor 'Cache', 7026 ocorrências do valor 'Cobblestone', 42 ocorrências do valor 'Default' e assim por diante para os demais valores únicos. Essa informação é útil para analisar a distribuição dos dados em relação aos diferentes valores da coluna '\_map'.

* Mapa seja mais favorável para CT

Para analisar qual dos mapas é mais favoráveis para CT(Counter Terrorist), determinei essa característica calculando as pontuações médias obtidas em cada lado do mapas e, em seguida, comparando ambos os lados.

Texto

Descrição gerada automaticamente

Existem longos períodos sem dados para um mapa no gráfico. Isso ocorre porque os mapas são adicionados e removidos constantemente pelos administradores do jogo.

Observando os gráficos o Nuke e Train oscilam como sendo os mapas mais favoráveis ao lado CT, tendo uma aproximação de 57% das rodadas vencidas pelo lado CT, enquanto Dust2 e Cache são historicamente os mapas mais favoráveis ao lado T.

Gráfico

Descrição gerada automaticamente

interessante em 2019 no mês abril o mapa Vertigo entrou no jogo tendo somente 4 meses, após o Cache foi removido do jogo.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

É interessante observar que o Inferno era conhecido por ser um mapa fortemente favorável ao lado CT antes de 2016, o que foi uma das razões para sua atualização. Desde sua atualização, Inferno tem sido na verdade o mapa mais equilibrado nesse aspecto.

Interface gráfica do usuário

Descrição gerada automaticamente

* Mapas mais removidos das partidas:

Tela de computador com jogo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Gráfico, Gráfico de barras

Descrição gerada automaticamente

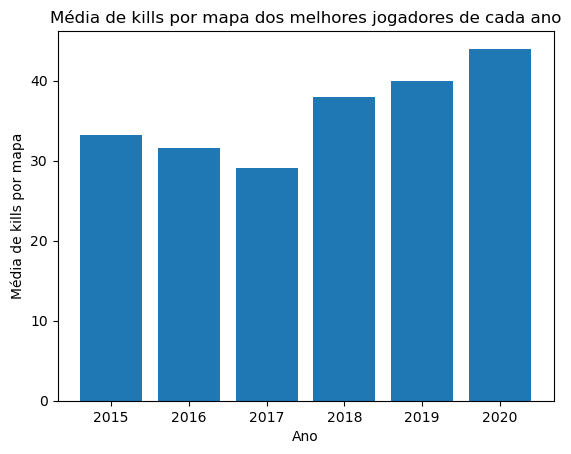
Neste gráfico, vemos os mapas mais removidos nas escolhas de mapas de cada partida, entre os anos de 2015-2020. Porém, vale lembrar, segundo o gráfico de comparações dos mapas mais escolhidos, que é possível notar que o mapa Vertigo, começou a ser jogado em -, e o Cobblestone foi retirado das escolhas para sua

* Desempenho dos melhores jogadores de cada ano:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente Texto

Descrição gerada automaticamente



Temos alguns pontos a se observar:

Um mapa tem 30 rounds.

Apenas Coldzera em 2017 fez uma media menor do que 30 kills por partida no ano em que foi o melhor jogador. O restante dos jogadores fizeram uma media de no minimo 1 kill por round do mapa, que é um dado muito alto e dificil de se alcançar.

Com este gráfico podemos ver como houve um aumento entre os anos de 2018 e 2020 na média dos kills por mapa entre os melhores daquele respectivo ano. Aumentando ainda mais as espectativas para os próximos anos.

* Maiores equipes vencedoras:

Texto

Descrição gerada automaticamente Tela de computador com fundo preto

Descrição gerada automaticamente Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente Gráfico, Gráfico de barras

Descrição gerada automaticamente

O código apresentado utiliza a biblioteca matplotlib.pyplot para criar um gráfico de barras horizontal que representa os top 5 times com mais vitórias. Vamos analisar cada linha do código:

fig, ax = plt.subplots(figsize=(8,6)): Cria uma figura (fig) e um conjunto de eixos (ax) para o gráfico. Define também o tamanho da figura como 8 polegadas de largura por 6 polegadas de altura.

bar\_container = ax.bar(sorted\_count\_df['team'].head(5), sorted\_count\_df['count'].head(5), color=['#FFA500', '#4169E1', '#FFD700', '#228B22', '#DC143C'], width=0.6): Cria um gráfico de barras utilizando os dados do DataFrame sorted\_count\_df. O eixo x é preenchido com os nomes dos times (sorted\_count\_df['team'].head(5)) e o eixo y é preenchido com o número de vitórias correspondente a cada time (sorted\_count\_df['count'].head(5)). O parâmetro color define as cores das barras e o parâmetro width define a largura das barras.

ax.set\_title('Top 5 times com mais vitórias', fontsize=16): Define o título do gráfico como "Top 5 times com mais vitórias" e define o tamanho da fonte do título como 16.

ax.set\_xlabel('Time', fontsize=14): Define o rótulo do eixo x como "Time" e define o tamanho da fonte do rótulo como 14.

ax.set\_ylabel('Número de vitórias', fontsize=14): Define o rótulo do eixo y como "Número de vitórias" e define o tamanho da fonte do rótulo como 14.

ax.tick\_params(axis='both', labelsize=12): Define os parâmetros dos ticks (marcas) nos eixos. Neste caso, define o tamanho da fonte dos ticks como 12.

for i, val in enumerate(sorted\_count\_df['count'].head(5)):: Cria um loop que percorre os valores das vitórias dos top 5 times (sorted\_count\_df['count'].head(5)).

ax.text(i, val-15, str(val), ha='center', fontsize=12): Adiciona um texto no gráfico próximo a cada barra representando o valor das vitórias correspondente. O parâmetro i representa a posição da barra, val-15 define a posição vertical do texto e str(val) converte o valor para string. O parâmetro ha='center' centraliza o texto horizontalmente e fontsize=12 define o tamanho da fonte do texto.

plt.show(): Exibe o gráfico de barras.

Em resumo, esse código cria um gráfico de barras horizontal que representa os top 5 times com mais vitórias. Cada barra representa um time e seu tamanho é proporcional ao número de vitórias. O gráfico é acompanhado por um título, rótulos nos eixos e valores das vitórias próximos às barras.

Alem de Esse código ser uma coleção de funções relacionadas à conexão e consulta a um banco de dados MySQL.

A função connect recebe os parâmetros host, database, user e password para estabelecer a conexão com o banco de dados. Ela utiliza a biblioteca mysql.connector para conectar ao servidor MySQL. Se a conexão for bem-sucedida, retorna o objeto de conexão. Caso ocorra algum erro, trata as exceções e exibe mensagens adequadas para os erros ER\_BAD\_DB\_ERROR (banco de dados inexistente) e ER\_ACCESS\_DENIED\_ERROR (usuário ou senha incorretos).

A função execute\_query recebe a conexão ao banco de dados, uma consulta SQL em formato de string (query) e opcionalmente dados para serem passados como parâmetros para a consulta (data). Essa função executa a consulta utilizando um cursor e retorna o resultado da consulta. Se data for fornecido, a consulta é executada com os parâmetros. O resultado da consulta é retornado e a conexão é confirmada (commit). Se ocorrer algum erro na execução da consulta, a função retorna uma mensagem de erro formatada. A conexão é fechada no bloco finally para garantir que a conexão seja encerrada independentemente de ter ocorrido algum erro ou não.

A função loop\_result recebe uma lista de resultados e itera sobre ela, convertendo cada resultado em uma lista e adicionando-a a uma lista from\_db. Em seguida, verifica se a lista from\_db não é vazia e a retorna. Caso contrário, retorna None.

Essas funções são úteis para estabelecer uma conexão com um banco de dados MySQL, executar consultas e processar os resultados. Elas fornecem uma abstração conveniente para realizar operações comuns em um banco de dados MySQL, como conexão, consulta e processamento dos resultados.