Estas melhores práticas incluem diretrizes para escrever consultas SQL, desenvolver documentação e exemplos que demonstram estas práticas. Este é um grande recurso para ter à mão quando você mesmo estiver usando SQL; você pode simplesmente ir diretamente à seção relevante para rever estas práticas. Pense nisso como um guia de campo SQL!

Capitalização e sensibilidade ao caso

Com SQL, a capitalização geralmente não importa. Você poderia escrever SELECT ou selecionar ou SeLeCT. Ambos iriam funcionar! Mas se você usar a capitalização como parte de um estilo consistente, suas consultas parecerão mais profissionais.

Para escrever consultas SQL como um profissional, é sempre uma boa ideia usar todas as tampas para iniciar as cláusulas (por exemplo, SELECT, FROM, WHERE, etc.). As funções também devem estar todas em letras maiúsculas (por exemplo, SUM()). Os nomes das colunas devem ser todos em letras minúsculas. (consulte a seção sobre o Snake_case, mais adiante neste guia). Os nomes das tabelas devem estar em CamelCase (consulte a seção sobre CamelCase mais adiante neste guia). Isto ajuda a manter suas consultas consistentes e mais fáceis de ler, sem afetar os dados que serão puxados quando você os executar. A única vez que escrever com letra maiúscula importa é quando ela está dentro de aspas (mais sobre aspas abaixo).

Os fornecedores de bancos de dados SQL podem usar variações ligeiramente diferentes de SQL. Estas variações são chamadas de **dialetos SQL**. Alguns dialetos SQL são sensíveis a maiúsculas e minúsculas. A BigQuery é uma delas. Vertica é outro. Mas a maioria, como MySQL, PostgreSQL e SQL Server, não são sensíveis a maiúsculas e minúsculas. Isto significa que se você procurou por country_code = 'us', ele retornará todas as entradas que tenham 'us', 'uS', 'Us' e 'US'. Este não é o caso da BigQuery. BigQuery é sensível a maiúsculas e minúsculas, de modo que ela busca só retornaria entradas onde o código do país é exatamente 'nós'. Se o código de país for 'US', BigQuery não retornaria essas entradas como parte de seu resultado.

Aspas simples ou duplas: "' ou " "

Na maioria das vezes, também não importa se você usa aspas simples '' ou aspas duplas "" quando se refere a strings. Por exemplo, o SELECT é uma cláusula inicial. Se você colocar o SELECT entre aspas como 'SELECT' ou "SELECT", então o SQL o tratará como uma string de texto. Sua consulta retornará um erro porque sua consulta necessita de uma cláusula SELECT.

Mas há duas situações em que importa que tipo de aspas que você usa:

1. Quando você quer que as strings sejam identificáveis em *qualquer* dialeto SQL
2. Quando sua string contém um apóstrofo ou aspas

Dentro de cada dialeto SQL existem regras para o que é aceito e o que não é. Mas uma regra geral em quase todos os dialetos SQL é usar aspas simples para as cordas. Isto ajuda a se livrar de muita confusão. Portanto, se quisermos fazer referência ao país US em uma cláusula WHERE (por exemplo, country_code = 'US'), então use aspas simples ao redor da string 'US'.

A segunda situação é quando sua string tem aspas dentro dela. Suponha que você tenha uma coluna de alimentos favoritos em uma tabela chamada FavoriteFoods e a outra coluna corresponda a cada amigo.

Amigo	**Favorite_food**
Rachel DeSantos	Shepherd’s pie
Sujin Lee	Tacos
Najil Okoro	Spanish paella

Você pode notar como a comida favorita de Rachel contém um apóstrofo. Se você usasse citações simples em uma cláusula WHERE para encontrar o amigo que tem esta comida favorita, seria parecido com isto:

```
SELECT
      Friend
FROM
      FavoriteFoods
WHERE
      Favorite_food = 'Shepherd's pie'
```

Não vai funcionar Se você executar esta consulta, será exibido um erro. Isto porque SQL reconhece uma cadeia de texto como algo que começa com uma citação 'e termina com outra aspas'. Portanto, na consulta ruim acima, o SQL pensa que o Favorite_food que você está procurando é 'Shepherd'. Apenas 'Shepherd' porque o apóstrofo em Shepherd termina a string.

Em geral, esta deveria ser a única vez que você usaria aspas duplas em vez de aspas simples. Assim, sua consulta ficaria assim:

```
SELECT
      Friend
FROM
      FavoriteFoods
WHERE
      Favorite_food = "Shepherd's pie"
```

SQL entende as string de texto como começando com uma aspas simples ' ou aspas dupla". Como esta string começa com aspas duplas, SQL esperará outra aspas duplas para sinalizar o fim da string. Isto mantém o apóstrofo a salvo, de modo que retorna "Shepherd's pie" e não "Shepherd".

Comentários como lembrete

À medida que você se sentir mais confortável com SQL, você será capaz de ler e entender as consultas num relance. Mas nunca custa ter comentários na consulta para se lembrar do que você está tentando fazer. E se você compartilhar sua consulta, isso também ajuda os outros a compreendê-la.

Por exemplo,

```
--This is an important query used later to join with the accounts table
SELECT
      rowkey, --key used to join with account_id
      Info.date, --date is in string format YYYY-MM-DD HH:MM:SS
      Info.code --e.g., 'pub-###'
FROM
      Publishers
```

Você pode usar # no lugar dos dois traços, --, na consulta acima, mas tenha em mente que # não é reconhecido em todos os dialetos SQL (o MySQL não reconhece #). Portanto, é melhor usar e ser consistente nisso. Quando você adiciona um comentário a uma consulta usando --, o mecanismo de consulta de banco de dados ignorará tudo na mesma linha depois de --.

Continuará a processar a consulta a partir da próxima linha.

Nomes "Snake_case" para colunas

É importante sempre garantir que o resultado de sua consulta tenha nomes fáceis de entender. Se você criar uma nova coluna (digamos, a partir de um cálculo ou da concatenação de novos campos), a nova coluna receberá um nome genérico padrão (por exemplo, f0). Por exemplo,

```
SELECT
       SUM(tickets),
       COUNT(tickets),
       SUM(tickets) AS total_tickets,
       COUNT(tickets) AS number_of_purchases
FROM
       purchases
```

A tabela a seguir apresenta os resultados desta consulta: f0: 8 f1: 4 total_tickets: 8 Number_of_purchases: 4
Resultados:

f0	f1	total_tickets	number_of_purchases
8	4	8	4

As duas primeiras colunas são denominadas f0 e f1 porque não foram nomeadas na consulta acima. O padrão SQL é f0, f1, f2, f3, e assim por diante. Nomeamos as duas últimas colunas total_de_tickets e número_de_compras para que estes nomes de colunas apareçam nos resultados da consulta. É por isso que é sempre bom dar nomes úteis a suas colunas, especialmente quando se utilizam funções.

Após executar sua consulta, você quer ser capaz de compreender rapidamente seus resultados, como as duas últimas colunas que descrevemos no exemplo.

Além disso, você pode notar como os nomes das colunas têm um sublinhado entre as palavras. Nunca deve haver espaços nos nomes. Se 'total_tickets' tivesse um espaço e parecesse 'total tickets' então SQL renomearia SUM(tickets) como apenas 'total'. Por causa do espaço, SQL usará 'total' como o nome e não entenderá o que você quer dizer com 'tickets'. Portanto, os espaços são ruins em nomes SQL. Jamais use espaços

A melhor prática é usar o Snake_case. Isto significa que 'total tickets', que tem um espaço entre

as duas palavras, deve ser escrito como 'total_tickets' com um sublinhado em vez de um espaço.

Nomes CamelCase para tabelas

Você também pode usar letras maiúsculas CamelCase ao nomear sua tabela. O uso de letras maiúsculas do CamelCase significa que o início de cada palavra será maiúscula, como um camelo de duas corcovas (bactriano). Portanto, a tabela TicketsByOccasion utiliza o padrão CamelCase. Observe que a letra maiúscula da primeira palavra em CamelCase é *opcional*; camelCase também é usado. Algumas pessoas diferenciam entre os dois estilos chamando CamelCase, PascalCase, e reservando CamelCase para quando a primeira palavra não é maiúscula, como um camelo de um só salto (Dromedary); por exemplo, ticketsByOccasion.

No final das contas, o CamelCase é uma escolha de estilo. Há outras maneiras de nomear suas tabelas, inclusive:

- Todas as letras minúsculas ou maiúsculas, como bilhetesbyoccasion ou TICKETSBYOCCASION
- Com snake_case, como tickets_by_occasion

Tenha em mente que a opção com todas as letras minúsculas ou maiúsculas pode dificultar a leitura do nome de sua tabela, por isso não é recomendada para uso profissional.

A segunda opção, Snake_case, é tecnicamente boa. Com palavras separadas por sublinhados, o nome de sua tabela é fácil de ler, mas pode ficar muito longo porque você está acrescentando os sublinhados. Também leva mais tempo para escrever. Se você usar muito esta tabela, ela pode se tornar uma tarefa difícil.

Em resumo, cabe a você usar o Snake_case ou CamelCase ao criar nomes de tabelas. Certifique-se apenas de que o nome de sua tabela seja fácil de ler e consistente. Certifique-se também de descobrir se sua empresa tem uma maneira preferida de nomear suas tabelas. Se o fizerem, sempre devem ir com sua convenção de nomeação por consistência.

Recuo (Indentação)

Como regra geral, você quer manter o comprimento de cada linha em uma consulta <=100 caracteres.
Isto facilita a leitura de suas consultas. Por exemplo, verifique esta consulta com uma linha com >100 caracteres:

```
SELECT
CASE WHEN genre = 'horror' THEN 'Will not watch' WHEN genre = 'documentary'
THEN 'Will watch alone' ELSE 'Watch with others' END AS
Watch_category, COUNT(movie_title) AS number_of_movies
  FROM
       MovieTheater
  GROUP BY
       1
```

SELECT CASE WHEN genre = 'horror' THEN 'Will not watch' WHEN genre = 'documentary'

THEN 'Will watch alone' ELSE 'Watch with others' END AS Watch_category,COUNT()

Esta consulta é difícil de ler e igualmente difícil de solucionar ou editar. Agora, aqui está uma pergunta onde nos mantemos fiéis à regra <= 100 caracteres:

```
SELECT
      CASE
            WHEN genre = 'horror' THEN 'Will not watch'
            WHEN genre = 'documentary' THEN 'Will watch alone'
            ELSE 'Watch with others'
            END AS watch_category, COUNT(movie_title) AS number_of_movies
FROM
      MovieTheater
GROUP BY
      1
```

Agora é muito mais fácil entender o que você está tentando fazer com a cláusula SELECT. Claro, ambas as consultas funcionarão sem problemas, pois a indentação não importa em SQL. Mas o recuo adequado ainda é importante para manter as linhas curtas. E será valorizado por qualquer pessoa que ler sua consulta, inclusive você mesmo!

Comentários multi-linha

Se você fizer comentários que ocupem várias linhas, você pode usar -- para cada linha. Ou, se você tiver mais de duas linhas de comentários, pode ser mais limpo e mais fácil é usar /* para iniciar o comentário e */ para fechar o comentário. Por exemplo, você pode usar o -- método como abaixo:

```
-- Date: September 15, 2020
-- Analyst: Jazmin Cisneros
-- Goal: Count the number of rows in the table
SELECT
      COUNT(*) number of rows -- the * stands for all so count all
FROM
      table
```

-- Data: 15 de setembro de 2020 -- Analista: Jazmin Cisneros -- Objetivo: Conta o número de linhas na tabela SELECT COUNT(*) número de linhas -- o * representa todos, portanto, conte todos da tabela SELECT COUNT(*)

Ou, você pode usar o método /* */ como no exemplo:

```
/*
Date: September 15, 2020
Analyst: Jazmin Cisneros
Goal: Count the number of rows in the table
*/
SELECT
      COUNT(*) AS number_of_rows -- the * stands for all so count all
FROM
      table
```

/* Data: 15 de setembro de 2020 Analista: Jazmin Cisneros Goal: Conta o número de linhas na tabela */ SELECT COUNT(*) AS number_of_rows -- o * representa todos, portanto conte todos da tabela */ SELECT COUNT(*)

Em SQL, não importa qual método você usa. A SQL ignora os comentários, independentemente do que você usa: #, --, ou /* e */. Portanto, depende de você e de sua preferência pessoal. O método /* e */ para comentários de várias linhas geralmente parece mais limpo e ajuda a separar os comentários da consulta. Mas não há um método certo ou errado.

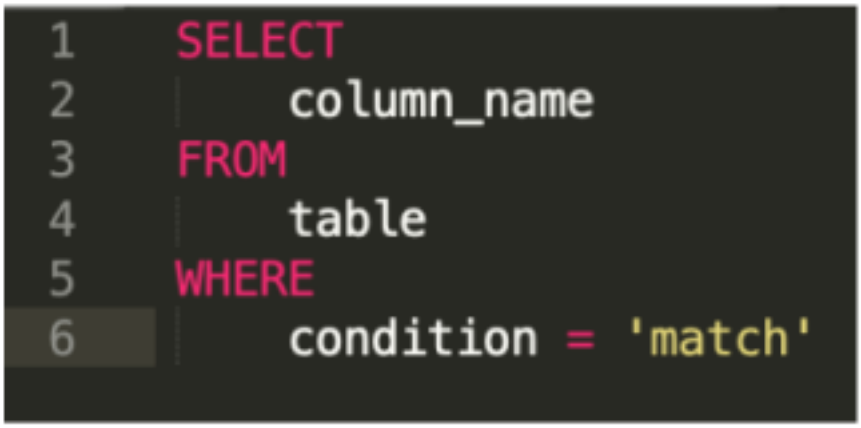
Editores de texto SQL

Ao ingressar em uma empresa, você pode esperar que cada empresa utilize sua própria plataforma SQL e dialeto SQL. A plataforma SQL que eles usam (por exemplo, BigQuery, MySQL ou SQL Server) é onde você escreverá e executará suas consultas SQL. Mas tenha em mente que nem todas as plataformas SQL fornecem editores de scripts nativos para escrever código SQL. Os editores de texto SQL fornecem uma interface onde você pode escrever suas consultas SQL de uma maneira mais fácil e codificada por cores. Na verdade, todo o código com o qual temos trabalhado até agora foi escrito com um editor de texto SQL.

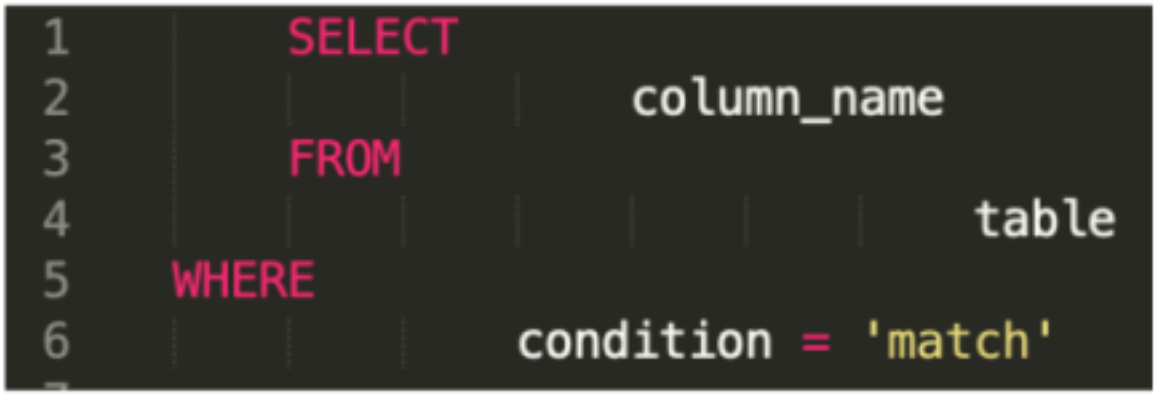
Exemplos com o Sublime Text

Se sua plataforma SQL não tiver codificação por cores, você pode pensar em usar um editor de texto como Sublime Text ou Atom. Esta seção mostra como o SQL é exibido no Sublime Text.

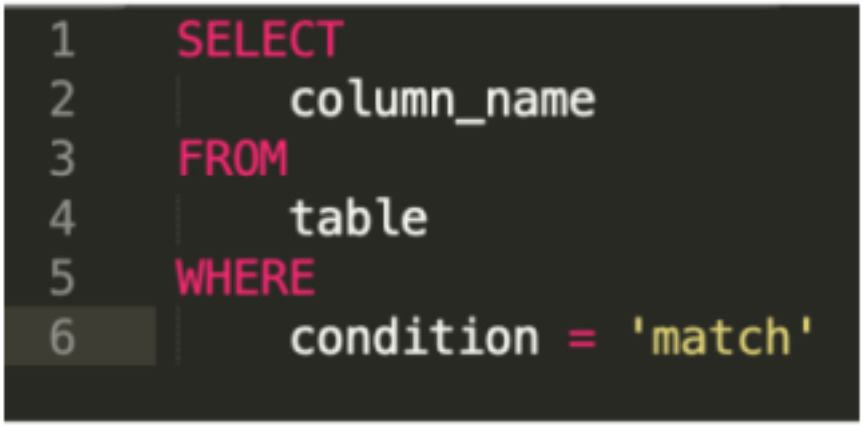
Aqui está uma consulta no Sublime Text.



Com o Sublime Text, você também pode fazer edições avançadas como a eliminação de travessões em várias linhas ao mesmo tempo. Por exemplo, suponha que sua consulta, de alguma forma, tivesse recuos nos lugares errados e tivesse este aspecto:



Isto é realmente difícil de ler, então você vai querer eliminar esses travessões e começar de novo. Em uma plataforma SQL regular, você teria que ir em cada linha e pressionar BACKSPACE para apagar cada travessão por linha. Mas no Sublime, você pode se livrar de todos os travessões ao mesmo tempo, selecionando todas as linhas e pressionando Command (ou CTRL em Windows) + [. Isto elimina os travessões de todas as linhas. Em seguida, você pode selecionar as linhas que deseja indentar (isto é, linhas 2, 4 e 6) pressionando a tecla Command (ou a tecla CTRL no Windows) e selecionando essas linhas. Depois, mantendo pressionada a tecla Command (ou a tecla CTRL no Windows), pressione] para indentar as linhas 2, 4, e 6 ao mesmo tempo. Isto limpará sua consulta e fará com que ela se pareça com isto:



O Sublime Text também suporta expressões regulares. **Expressões regulares** (ou **regex**) podem ser usadas para procurar e substituir padrões de string em consultas. Não falaremos aqui sobre as expressões regulares, mas talvez você queira aprender mais sobre elas por conta própria, pois são uma ferramenta muito poderosa.

Você pode começar com estes recursos:

- Pesquisar e substituir no Sublime Text
- Regex tutorial (se você não souber o que são expressões regulares)
- Folha de referências sobre Regex