Princípios básicos de SQL

**Sobre este codelab**

account\_circleEscrito por Equipe de treinamento do Google Developers

[1. Antes de começar](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#0)

Nos codelabs anteriores, você aprendeu a incorporar dados da rede ao app e a usar corrotinas para processar tarefas simultâneas. Neste módulo, você aprenderá outra habilidade fundamental para o desenvolvimento em Android, que permitirá criar apps de qualidade: a *persistência*. Mesmo que você nunca tenha ouvido esse termo antes, é provável que já tenha se deparado com a persistência ao usar um app. Os apps usam a persistência para proporcionar uma boa experiência ao usuário, seja para criar uma lista de compras, rolar imagens de vários anos atrás em um app de fotos ou mesmo pausar e retomar um jogo. Embora os usuários possam achar que esses são recursos comuns, a persistência de dados é uma habilidade essencial para um desenvolvedor criar apps de alta qualidade.

Mais adiante nesta unidade, você aprenderá mais sobre a persistência no Android e aprenderá sobre uma biblioteca chamada Room, que permite que seus apps leiam e gravem informações em um banco de dados. No entanto, antes de se aprofundar no trabalho com a persistência no Android, é importante conhecer os conceitos básicos dos bancos de dados relacionais e saber como ler e manipular dados com algo chamado SQL (linguagem de consulta estruturada, na sigla em inglês). Se você já estiver familiarizado com esses conceitos, considere este guia como uma revisão para garantir que eles estejam bem memorizados quando você aprender sobre o Room. Se você ainda não conhecê-los, não tem problema: Não esperamos que você saiba nada sobre bancos de dados ainda. Ao final deste codelab, você conhecerá todos os fundamentos básicos necessários para começar a aprender a trabalhar com bancos de dados em um app Android.

Pré-requisitos

* Saber navegar em um projeto no Android Studio.

O que você aprenderá

* A estrutura de um banco de dados relacional: tabelas, colunas e linhas.
* Instruções SELECT, incluindo WHERE, ORDER BY, GROUP BY e LIMIT.
* Como inserir, atualizar e excluir linhas usando SQL.

Pré-requisitos

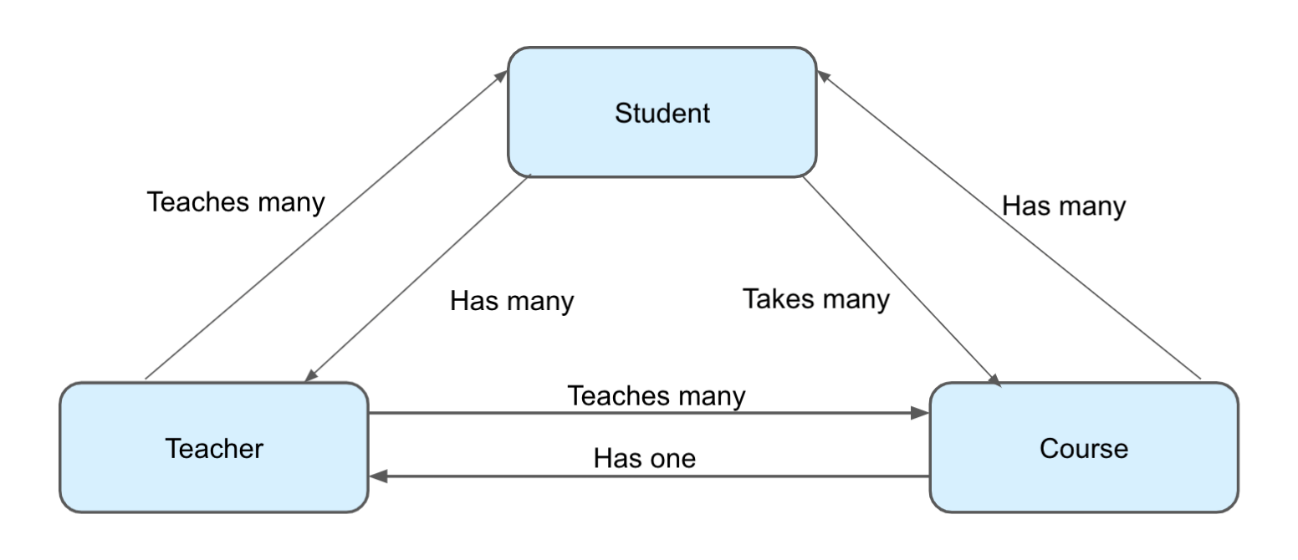
* Um computador com o Android Studio instalado.

## [2. Visão geral de um banco de dados relacional](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#1)

## O que é um banco de dados relacional?

Em computação, um banco de dados é simplesmente uma coleção estruturada de dados que podem ser acessados e gravados em meios eletrônicos. Os bancos de dados podem armazenar as mesmas informações que podem ser representadas em um app usando o Kotlin. Em dispositivos móveis, os bancos de dados geralmente são usados para salvar dados de um app em execução para que eles possam ser acessados na próxima vez em que o app for aberto, sem que seja necessário recuperar os dados de outra fonte, como a Internet. Isso é conhecido como persistência de dados.

Quando se trata de persistência de dados, você ouvirá bastante o termo "banco de dados relacional". Esse é um tipo de banco de dados comum que organiza informações em tabelas, colunas e linhas. Ao programar código em Kotlin, você cria classes que representam objetos. Uma tabela em um banco de dados relacional funciona da mesma forma. Além de representar dados, as tabelas também podem referenciar outras tabelas, para que seja possível estabelecer relações entre elas. Um exemplo clássico seria uma tabela de "alunos", "professores" e "cursos". Um curso pode ter somente um professor, mas um aluno pode fazer vários cursos. Um banco de dados pode representar as relações entre essas tabelas, e é por isso que você ouvirá bastante o termo banco de dados relacional.



As relações do mundo real podem ser representadas por relações entre tabelas.

## Tabelas, colunas e linhas

A definição das tabelas ou dos dados que serão representados é apenas a primeira etapa na criação de um banco de dados relacional. Também é necessário pensar em quais informações específicas serão armazenadas em cada tabela. As propriedades específicas são representadas por colunas. Uma coluna é composta por um nome e um tipo de dado. Você já está familiarizado com propriedades por trabalhar com classes em Kotlin. Pense nas tabelas SQL da mesma forma. Uma tabela é como uma definição de classe, que descreve o tipo de "coisa" que será representada. Uma coluna é uma propriedade específica da "coisa", criada com cada entrada da tabela.

|  |  |
| --- | --- |
| PlantidINTEGERspeciesTEXT (string)nameTEXT (string)colorTEXT (string) | GardenidINTEGERnameTEXT (string)lengthINTEGERwidthINTEGER |
| Planta | |  |
| id | INTEIRO |  |
| espécie | TEXTO (string) |  |
| nome | TEXTO (string) |  |
| cor | TEXTO (string) |  |
| Jardim | |  |
| id | INTEIRO |  |
| nome | TEXTO (string) |  |
| comprimento | INTEIRO |  |
| largura | INTEIRO |  |

As entradas individuais da tabela são chamadas de linhas. Elas são como uma instância de uma classe em Kotlin. Cada uma contém dados que correspondem a cada coluna. A tabela fornece o modelo, mas as linhas definem os dados reais que serão armazenados.

|  |  |  |  |
| --- | --- | --- | --- |
| id | espécie | nome | cor |
| 1 | Camellia Sinensis | Chá da Índia | verde |
| 2 | Echinacea Purpurea | Equinácea | roxo |
| 3 | Ferula Foetida | Assa-fétida | verde |

## Chave primária

No exemplo acima, observe que há uma coluna para uma propriedade id. Ainda que seja provável que as espécies de plantas na natureza, ou qualquer outra coisa representada no banco de dados, não tenham um ID numerado, é importante que as linhas em uma tabela de dados tenham algum tipo de identificador exclusivo. Ele é normalmente conhecido como chave primária. Essa chave é exclusiva para cada linha da tabela. Ela é útil para referenciar linhas de uma tabela de dados em outra tabela. Por exemplo, digamos que haja outra tabela chamada "jardim", em que você quer associar um jardim com todas as espécies de plantas que ele contém. Você poderia usar a chave primária da tabela de plantas para referenciar uma planta que está em uma entrada da tabela "jardim" ou de qualquer outra tabela do banco de dados.

As chaves primárias possibilitam estabelecer relações dentro de um banco de dados relacional. Você não usará bancos de dados com mais de uma tabela neste curso. Ainda assim, ter um ID exclusivo ajudará a consultar, atualizar e excluir itens existentes em uma tabela.

## Tipos de dados

Assim como ocorre ao definir propriedades de classes do Kotlin, as colunas em um banco de dados podem corresponder a vários tipos de dados possíveis. Uma coluna pode representar um caractere, uma string, um número (com ou sem decimal) ou dados binários. Outros dados, como datas e horários, podem ser representados numericamente ou como uma string, dependendo do caso de uso. Ao trabalhar com o Room, você lidará principalmente com os tipos do Kotlin, que são mapeados para os tipos SQL em segundo plano.

**Observação**: assim como no Kotlin, as colunas em um banco de dados podem ter diferentes tipos de dados. Contudo, eles não são exatamente os mesmos tipos do Kotlin. Se você quiser saber mais, consulte [este recurso](https://www.w3schools.com/sql/sql_datatypes.asp) (link em inglês) para ter uma visão geral dos tipos de dados básicos em SQL. No entanto, ao trabalhar com um banco de dados no Room, você não lidará diretamente com esses tipos de dados. Em vez disso, você definirá as tabelas de dados no código usando tipos de dados do Kotlin equivalentes.

## SQL

Para acessar um banco de dados relacional, seja por conta própria ou usando uma biblioteca como o Room, você precisará de algo chamado SQL.

O que é SQL? SQL significa linguagem de cultura estruturada e permite ler e manipular dados em um banco de dados relacional. Não se preocupe: você não precisará aprender uma linguagem de programação totalmente nova só para implementar a persistência no seu app. Ao contrário de uma linguagem de programação, como o Kotlin, a SQL consiste em apenas alguns tipos de instruções para ler e gravar informações em um banco de dados. Após aprender o formato básico de cada uma, basta preencher as lacunas referentes às informações específicas que estão sendo lidas ou gravadas no banco de dados.

Veja abaixo as instruções SQL mais comuns, que você usará no seu trabalho.

|  |  |
| --- | --- |
| SELECT | Recebe informações específicas de uma tabela de dados. Os resultados podem ser filtrados e classificados de várias maneiras. |
| INSERT | Adiciona uma nova linha a uma tabela. |
| UPDATE | Atualiza uma ou mais linhas em uma tabela. |
| DELETE | Remove uma ou mais linhas de uma tabela. |

Antes de fazer qualquer coisa no SQL, você precisará de um banco de dados. Na próxima tela, você receberá um projeto de amostra que inclui um banco de dados para praticar consultas SQL.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

[3. Código inicial: banco de dados de parques](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#2)

Você fará o download de um código inicial um pouco diferente dos usados nos codelabs anteriores. Em vez de criar com base em um projeto existente, forneceremos um projeto simples do Android Studio que cria um banco de dados a ser usado para praticar consultas SQL. Depois de executar o app uma vez, você poderá acessar o banco de dados usando uma ferramenta do Android Studio chamada Database Inspector.

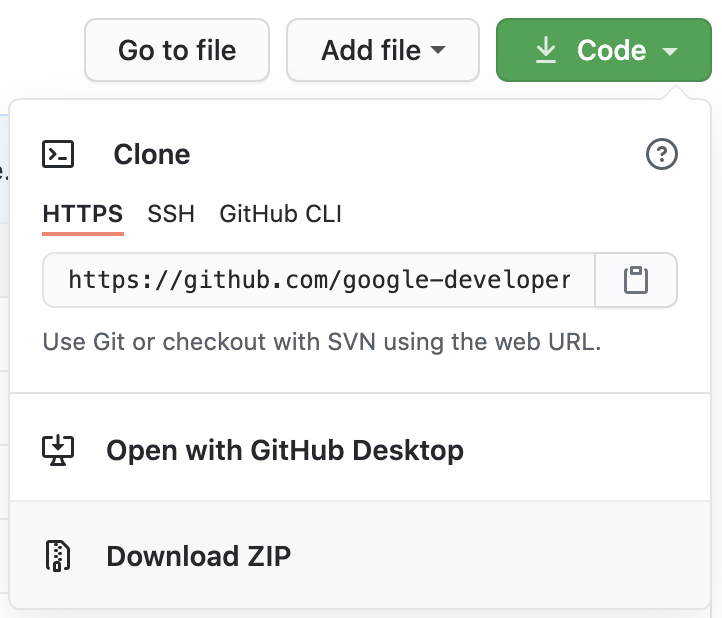
**URL do código do inicial:**[https://github.com/google-developer-training/android-fundamentals-kotlin-sql-basic-app](https://github.com/google-developer-training/android-basics-kotlin-sql-basics-app) (link em inglês)

**Ramificação: main**

Para encontrar o código deste codelab e abri-lo no Android Studio, faça o seguinte.

Buscar o código

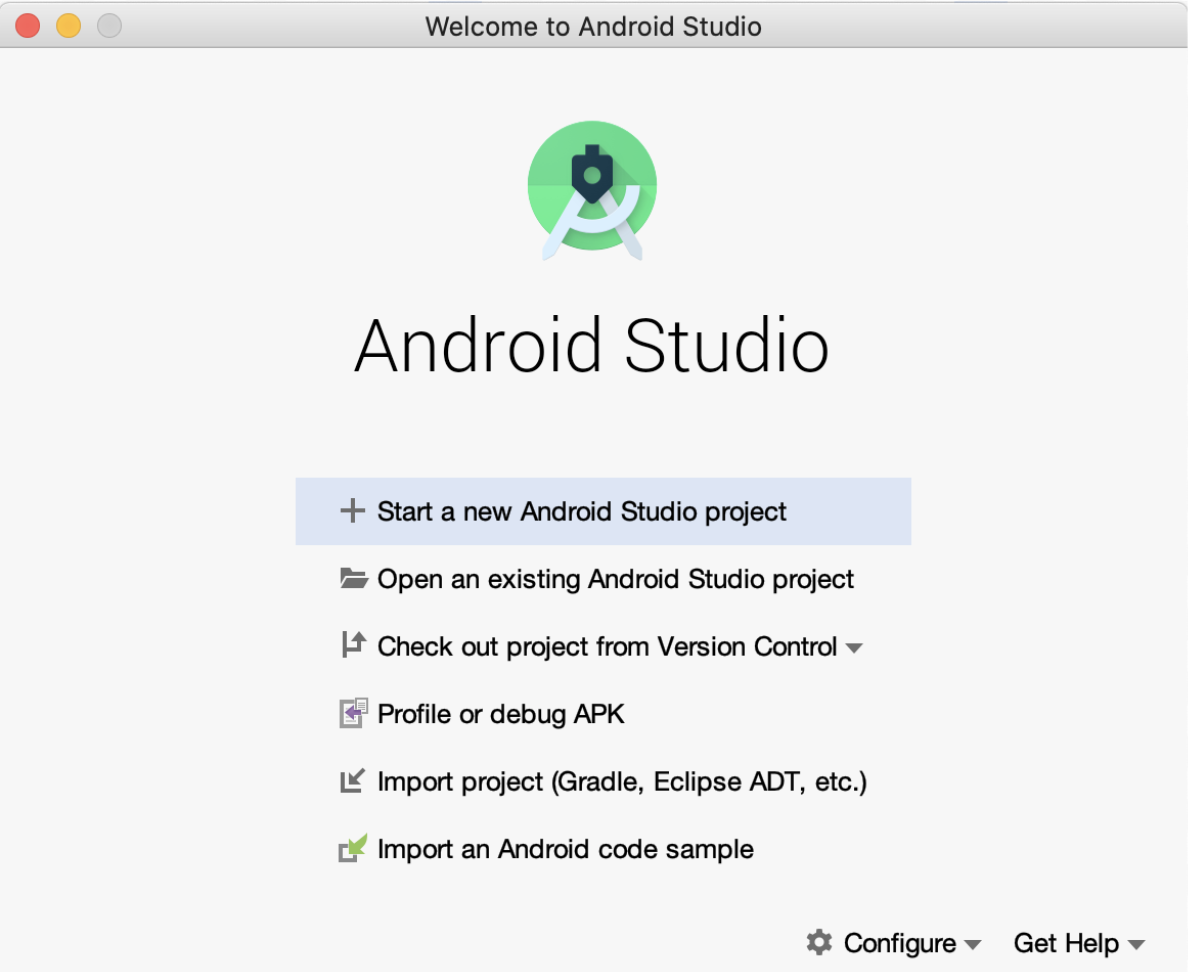
1. Clique no URL fornecido. Isso abrirá a página do GitHub referente ao projeto em um navegador.
2. Na página do GitHub do projeto, clique no botão **Code**, que exibirá uma caixa de diálogo.



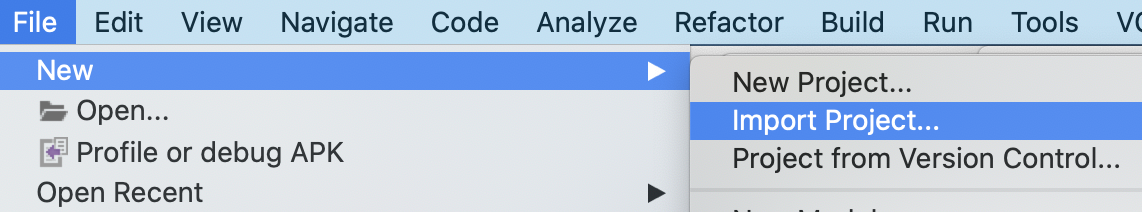
1. Na caixa de diálogo, clique no botão **Download ZIP** para salvar o projeto no seu computador. Aguarde a conclusão do download.
2. Localize o arquivo no computador, que provavelmente está na pasta **Downloads**.
3. Clique duas vezes no arquivo ZIP para descompactá-lo. Isso criará uma nova pasta com os arquivos do projeto.

Abrir o projeto no Android Studio

1. Inicie o Android Studio.
2. Na janela **Welcome to Android Studio**, clique em **Open an existing Android Studio project**.



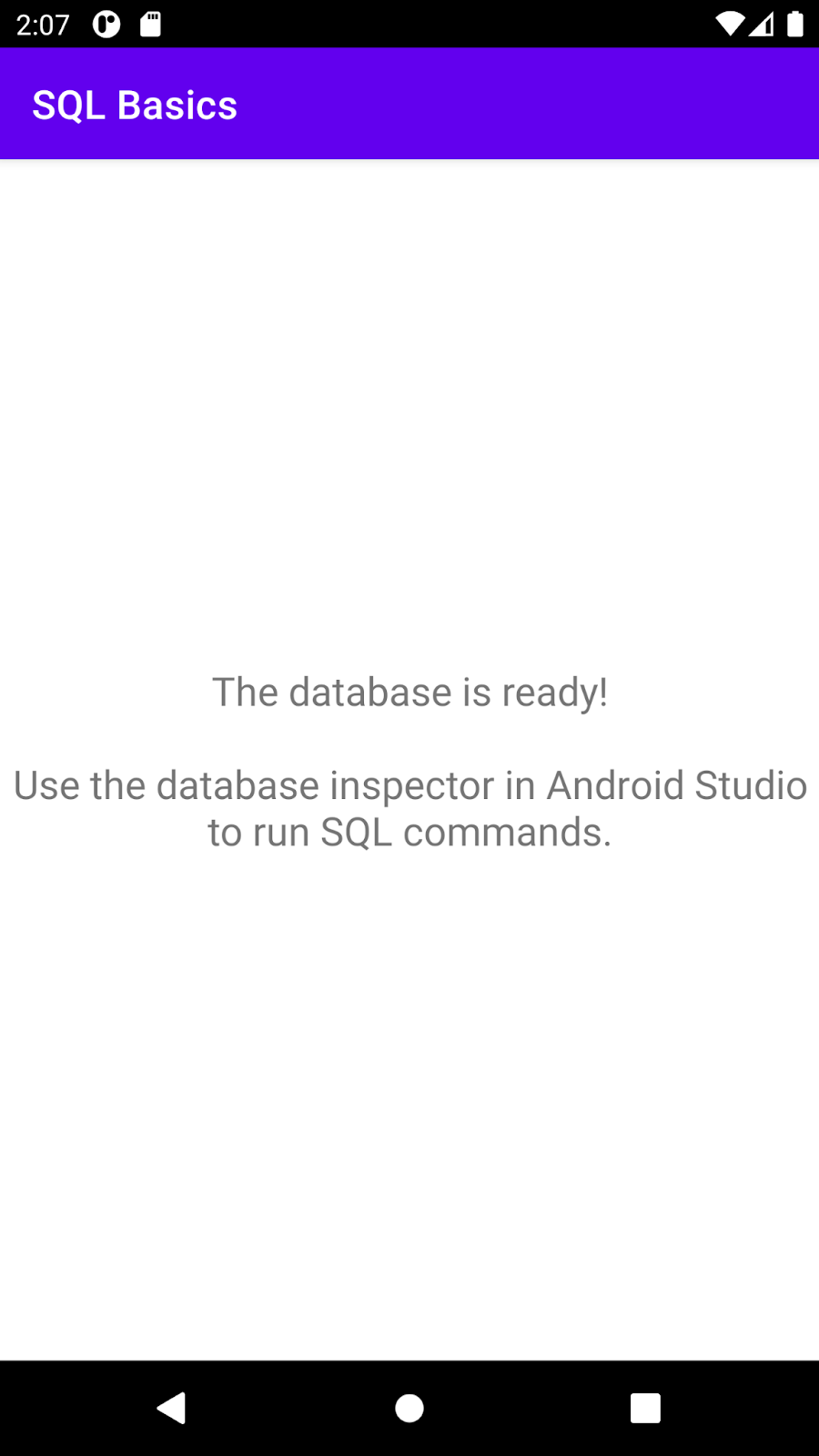
Observação: caso o Android Studio já esteja aberto, selecione a opção **File** > **New** > **Import Project**.



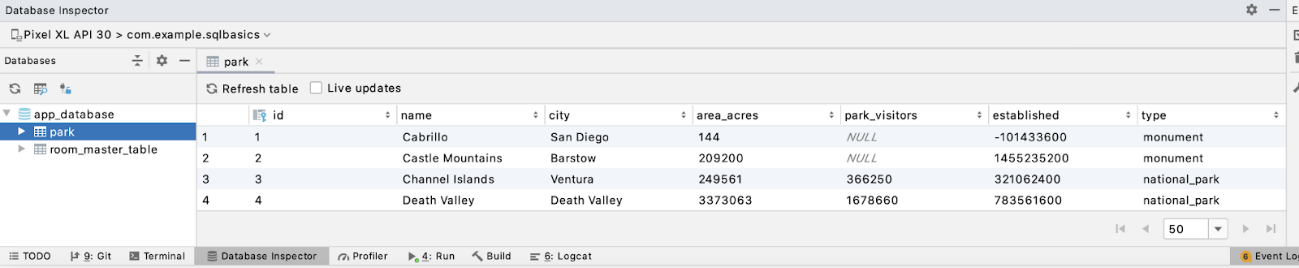
1. Na caixa de diálogo **Import Project**, vá até a pasta do projeto descompactada, que provavelmente está na pasta **Downloads**.
2. Clique duas vezes nessa pasta do projeto.
3. Aguarde o Android Studio abrir o projeto.
4. Clique no botão **Run**  para criar e executar o app. Confira se ele funciona da forma esperada.
5. Procure os arquivos do projeto na janela de ferramentas **Project** para ver como o app foi implementado.

Antes de avançar para a próxima seção, siga as etapas abaixo para garantir que o projeto inicial está pronto para uso.

1. Execute o app. Ele exibirá uma única tela parecida com o exemplo a seguir.



1. No Android Studio, abra o Database Inspector em **View** > **Tool Windows** > **Database Inspector**.
2. Você verá uma nova guia na parte inferior chamada "Database Inspector". Pode demorar alguns segundos para ela carregar, mas você verá uma lista à esquerda com as tabelas de dados, que podem ser selecionadas para fazer consultas.

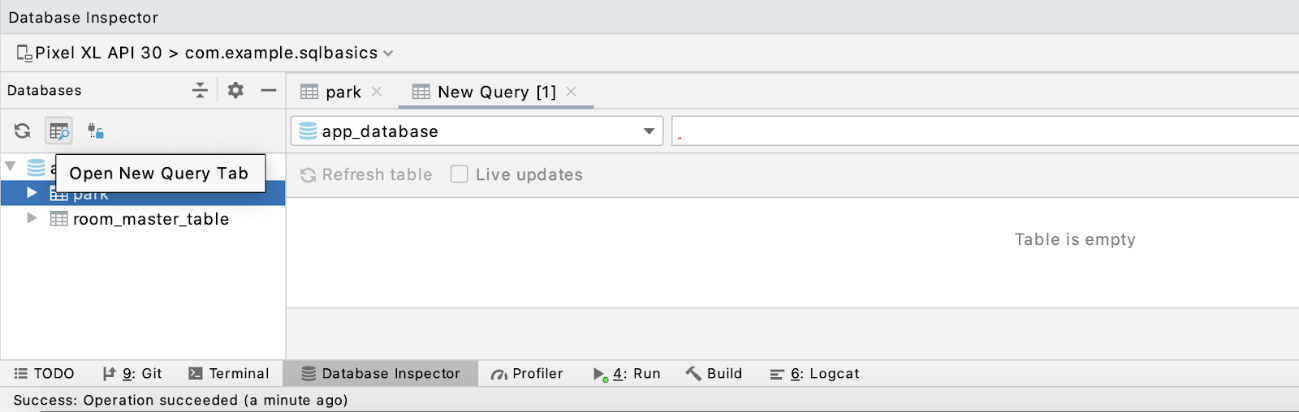


Se você usa o Android Studio 2020.3.1 Arctic Fox, o Database Inspector pode ser acessado em View > Tool > App Inspection. A guia na parte inferior ainda será chamada Database Inspector.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

## [4. Instruções SELECT básicas](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#3)

Nos exercícios a seguir, você fará consultas no Database Inspector. Verifique se você selecionou a tabela correta no painel esquerdo (park) dos parques e clique no botão **Open New Query Tab**. Uma caixa de texto será exibida, em que você pode digitar os comandos SQL.



Uma instrução SQL é um comando, parecido com uma linha de código, que acessa um banco de dados, seja para leitura ou gravação. A função SQL mais básica possível é simplesmente coletar todos os dados de uma tabela. Para fazer isso, comece com a palavra SELECT, o que significa que você quer ler dados. Em seguida, adicione um asterisco (\*). É nesse ponto que você especificará as colunas que quer selecionar. O asterisco é um atalho para selecionar todas as colunas. Depois disso, use a palavra-chave FROM, seguida pelo nome da tabela de dados, park. Execute o seguinte comando no Database Inspector e observe a tabela inteira, com todas as linhas e colunas.

SELECT \* FROM park

Caso você queira selecionar somente uma determinada coluna, e não todas as colunas da tabela de dados, especifique o nome da coluna.

SELECT city FROM park

Também é possível selecionar várias colunas específicas, separando cada uma delas com uma vírgula.

SELECT name, established, city FROM park

Às vezes, não é necessário selecionar todas as linhas de um banco de dados. É possível adicionar cláusulas (parte de uma instrução SQL) para restringir ainda mais os resultados.

Uma das cláusulas é LIMIT, que permite definir um limite de número de linhas retornadas. Portanto, em vez de retornar todos os 23 resultados, a consulta a seguir retornará apenas os cinco primeiros.

SELECT name FROM park  
LIMIT 5

Uma das cláusulas mais comuns e úteis é a WHERE. Uma cláusula WHERE permite filtrar os resultados com base em uma ou mais colunas.

SELECT name FROM park  
WHERE type = "national\_park"

**Observação**: diferentemente do Kotlin, em que o operador = é usado para a atribuição e o == é usado para comparação, em SQL, apenas um sinal de igual é usado para comparar dois valores.

Existe também um operador "diferente de" (!=). A consulta a seguir lista todos os parques com mais de 400 quilômetros quadrados que não sejam uma recreation\_area. Com cláusulas WHERE, também é possível usar operadores booleanos, como AND ou OR, para adicionar mais de uma condição.

SELECT name FROM park  
WHERE type != "recreation\_area"  
AND area\_acres > 100000

## Praticar

As consultas SQL podem ser úteis para responder a várias perguntas sobre seus dados. A melhor forma de praticar é programando suas próprias consultas. Nas próximas etapas, você terá a oportunidade de programar uma consulta para responder a uma pergunta específica. Execute um teste no Database Inspector antes de continuar.

Todos os exercícios consideram o conhecimento cumulativo de todas as seções anteriores, e haverá tutoriais no fim deste codelab para que você possa verificar suas respostas.

**Problema 1:**

Programe uma consulta SQL para buscar os nomes de todos os parques com menos de 1 milhão de visitantes.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

## [5. Funções SQL comuns](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#4)

A primeira consulta que você programou simplesmente retorna todas as linhas do banco de dados.

SELECT \* FROM park

Mas talvez você não queira receber uma longa lista de resultados. A SQL também oferece funções agregadas que podem ajudar a reduzir os dados em um único valor relevante. Por exemplo, digamos que você queira saber qual é o número de linhas da tabela park. Em vez de usar SELECT \* ..., use a função COUNT() e transmita \*, (para todas as linhas) ou o nome de uma linha. A consulta retornará uma contagem de todas as linhas.

SELECT COUNT(\*) FROM park

Outra função agregada útil é a SUM() para somar os valores de uma coluna. Essa consulta filtra somente os parques nacionais, porque são as únicas entradas com uma coluna park\_visitors que não é nula, e soma o número total de visitantes de cada parque.

SELECT SUM(park\_visitors) FROM park  
WHERE type = "national\_park"

Vale ressaltar que é possível usar SUM() em um valor nulo, mas o valor será simplesmente tratado como zero. A consulta a seguir retornará o mesmo resultado da consulta acima. No entanto, ainda é interessante ser o mais específico possível para evitar bugs quando você começar a usar a SQL em apps.

SELECT SUM(park\_visitors) FROM park

Além de agregar valores, existem também outras funções úteis, como MAX() e MIN(), que buscam o maior ou menor valor, respectivamente.

SELECT MAX(area\_acres) FROM park  
WHERE type = 'national\_park'

## Como receber valores DISTINCT

Em algumas linhas, o valor da coluna é igual ao de outras linhas. Por exemplo, a coluna "type" tem apenas um número finito de valores possíveis. É possível eliminar valores duplicados dos resultados da consulta usando a palavra-chave DISTINCT. Por exemplo, para receber todos os valores exclusivos da coluna "type", use a consulta a seguir.

SELECT DISTINCT type FROM park

Também é possível usar DISTINCT em uma função agregada. Assim, em vez de listar os types exclusivos e contá-los, basta retornar a contagem.

SELECT COUNT(DISTINCT type) FROM park

## Praticar

Coloque o que você aprendeu em prática e tente programar as consultas a seguir. Use o Database Inspector para verificar se o código funciona.

**Problema 2**:

Programe uma consulta SQL para descobrir o número de cidades diferentes na tabela **park**.

**Problema 3**:

Programe uma consulta SQL para buscar o número total de visitantes de parques em São Francisco.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

## [6. Como ordenar e agrupar resultados da consulta](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#5)

Nos exemplos anteriores, pode ter sido difícil encontrar uma entrada específica. Felizmente, também é possível classificar os resultados de uma instrução SELECT usando uma cláusula ORDER BY. Adicione uma cláusula ORDER BY no final da consulta, depois da cláusula WHERE, se houver, e especifique o nome da coluna que você quer classificar. O exemplo a seguir retorna o nome de cada parque no banco de dados, classificando os resultados em ordem alfabética.

SELECT name FROM park  
ORDER BY name

Por padrão, os resultados são classificados em ordem crescente, mas você pode adicionar a palavra-chave ASC ou DESC à ordem para classificar em ordem crescente ou decrescente. Não é necessário especificar ASC, porque a primeira consulta já lista os resultados em ordem crescente. Contudo, caso você queira receber os resultados em ordem decrescente, adicione a palavra-chave DESC ao final da cláusula ORDER BY.

SELECT name FROM park  
ORDER BY name DESC

Para facilitar a leitura dos resultados, também existe a opção de agrupá-los por coluna. Antes da cláusula ORDER BY, se houver, é possível especificar uma cláusula GROUP BY e uma coluna. Os resultados ficarão separados em um subconjunto específico da coluna em GROUP BY. Para cada coluna, os resultados serão filtrados e ordenados de acordo com o restante da consulta.

SELECT type, name FROM park  
GROUP BY type  
ORDER BY name

Fica mais fácil compreender isso com um exemplo. Em vez de contar todos os parques do banco de dados, é possível ver quantos parques de cada tipo estão presentes e receber uma contagem separada para cada um.

SELECT type, COUNT(\*) FROM park  
GROUP BY type  
ORDER BY type

## Praticar

Coloque o que você aprendeu em prática e tente programar a consulta a seguir. Use o Database Inspector para verificar se o código funciona.

**Problema 4**: programe uma consulta SQL para descobrir os cinco parques que tiveram mais visitantes, incluindo a contagem de visitantes, em ordem decrescente.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

## [7. Como inserir e excluir linhas](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#6)

Você precisará programar dados para aproveitar ao máximo a persistência de dados no Android com o Room. Além de consultar um banco de dados, também existem instruções SQL para inserir, atualizar e excluir linhas. Você precisará desses conhecimentos básicos para aprender a gravar dados com o Room mais tarde, no módulo 2.

## Instrução INSERT

Para adicionar uma nova linha, use a instrução INSERT. A instrução INSERT é seguida pela palavra-chave INTO e o nome da tabela em que você quer adicionar uma linha. Após a palavra-chave VALUES, indique o valor para cada coluna entre parênteses, na ordem, separando cada um deles por uma vírgula. Uma instrução INSERT tem o formato apresentado a seguir.

INSERT INTO table\_name  
VALUES (column1, column2, ...)

Para adicionar uma linha à tabela park, a instrução INSERT ficaria parecida com o exemplo a seguir. Os valores correspondem à ordem em que as colunas são definidas na tabela park. Observe que alguns dados não estão especificados. Por enquanto isso não é um problema, já que é possível atualizar uma linha depois que ela é inserida.

INSERT INTO park  
VALUES (null, 'Googleplex', 'Mountain View', 12, null, 0, '')

Observe também que null foi transmitido para o ID. Embora seja possível fornecer um número específico, isso não é muito conveniente, porque o app precisaria monitorar o ID mais recente para garantir que não haja duplicidade. No entanto, é possível configurar o banco de dados para que a chave primária seja incrementada automaticamente, como nesse caso. Dessa forma, é possível transmitir null, e o próximo ID será escolhido automaticamente.

Verifique se a entrada foi criada usando uma cláusula WHERE para especificar o parque chamado "Googleplex".

SELECT \* FROM park  
WHERE name = 'Googleplex'

## Instrução UPDATE

Depois que a linha é criada, é possível mudar o conteúdo dela a qualquer momento. Para isso, use a instrução UPDATE. Como todas as outras instruções SQL abordadas, primeiro é necessário especificar o nome da tabela. Na cláusula SET, basta definir cada coluna que será atualizada com o novo valor.

UPDATE table\_name  
SET column1 = ...,  
column2 = ...,  
...  
WHERE column\_name = ...  
...

Para a entrada "Googleplex", uma propriedade existente será atualizada e alguns outros campos serão preenchidos. Esses campos já tinham um valor, que era uma string vazia: "". Com a instrução UPDATE, é possível atualizar vários ou todos os campos de uma só vez.

UPDATE park  
SET area\_acres = 46,  
established = 1088640000,  
type = 'office'  
WHERE name = 'Googleplex'

Veja as atualizações refletidas nos resultados da consulta.

SELECT \* FROM park  
WHERE name = 'Googleplex'

## Instrução DELETE

Por fim, também é possível usar um comando SQL para excluir linhas do banco de dados. Especifique o nome da tabela mais uma vez. Assim como foi feito com as instruções SELECT, use uma cláusula WHERE para informar os critérios para as linhas que serão excluídas. Como uma cláusula WHERE pode corresponder a várias linhas, é possível excluir várias linhas com um único comando.

DELETE FROM table\_name  
WHERE <column\_name> = ...

Como o Googleplex não é um parque nacional, tente usar uma instrução DELETE para remover essa entrada do banco de dados.

DELETE FROM park  
WHERE name = 'Googleplex'

Verifique se a linha foi excluída usando uma instrução SELECT. A consulta não retornará resultados, o que significa que todas as linhas com o nome "Googleplex" foram excluídas.

SELECT \* FROM park  
WHERE name = 'Googleplex'

Já abordamos tudo sobre inserir, atualizar e excluir dados. Você só precisa saber o formato do comando SQL que será executado e especificar os valores que correspondam às colunas do banco de dados. Quando apresentarmos o Room no próximo codelab, o foco será ler um banco de dados. A inserção, atualização e exclusão de dados serão abordadas em detalhes no Módulo 2.

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

## [8. Resoluções dos exercícios](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#7)

Esperamos que os exercícios práticos tenham ajudado a consolidar sua compreensão dos conceitos de SQL. Caso você não tenha conseguido solucionar algum deles ou queira verificar suas respostas, consulte as respostas abaixo.

**Problema 1**: programe uma consulta SQL para buscar os nomes de todos os parques com menos de 1 milhão de visitantes.

Esse problema solicita os nomes dos parques (uma única coluna) que tenham menos de 1 milhão de visitantes, o que pode ser especificado na cláusula WHERE.

SELECT name FROM park  
WHERE park\_visitors < 1000000

**Problema 2**: programe uma consulta SQL para descobrir o número de cidades diferentes na tabela park.

A contagem total de uma coluna pode ser calculada usando a função COUNT(). No entanto, como você quer receber apenas cidades diferentes, já que algumas cidades têm vários parques, use a palavra-chave DISTINCT antes do nome da coluna na função COUNT().

SELECT COUNT(DISTINCT city) FROM park

**Problema 3**: programe uma consulta SQL para buscar o número total de visitantes de parques em São Francisco.

O número total de visitantes pode ser calculado usando a função SUM(). Além disso, também é necessário usar uma cláusula WHERE para especificar apenas parques em São Francisco.

SELECT SUM(park\_visitors) FROM park  
WHERE city = "San Francisco"

**Problema 4**: programe uma consulta SQL para descobrir os cinco parques (somente nomes) que tiveram mais visitantes, incluindo a contagem de visitantes, em ordem decrescente.

A consulta precisa abranger as colunas "name" e "park\_visitors". Os resultados serão classificados na coluna park\_visitors, em ordem decrescente, usando a cláusula ORDER BY. Como você não quer agrupar os resultados em outra coluna e classificá-los dentro desses grupos, não é necessário usar uma cláusula GROUP BY.

SELECT name, park\_visitors FROM park  
ORDER BY park\_visitors DESC  
LIMIT 5

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)

[9. Parabéns](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics#8)

Resumindo:

* Os bancos de dados relacionais permitem armazenar dados organizados em tabelas, colunas e linhas.
* É possível recuperar dados de um banco de dados usando a instrução SQL SELECT.
* Você pode usar uma variedade de cláusulas em uma instrução SELECT, incluindo WHERE, GROUP BY, ORDER BY e LIMIT, para tornar as consultas mais específicas.
* Use funções agregadas para combinar dados de várias linhas em uma única coluna.
* É possível adicionar, atualizar e excluir linhas de um banco de dados usando as instruções SQL INSERT, UPDATE e DELETE, respectivamente.

Saiba mais

* [Tipos de dados SQL](https://www.w3schools.com/sql/sql_datatypes.asp) (link em inglês)
* [Funções agregadas](https://www.codecademy.com/learn/learn-sql/modules/learn-sql-aggregate-functions/cheatsheet) (link em inglês)
* [Mesclagens](https://www.w3schools.com/sql/sql_join.asp) (link em inglês)

[**Voltar**](https://developer.android.com/codelabs/basic-android-kotlin-training-sql-basics?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-unit-5-pathway-1%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-sql-basics)