Primeiros passos em SQL

(Re)Start BootCamp

27 de agosto de 2024
19:30 - 20:40

Oi, eu sou **Ana Duarte**

Brasiliense apaixonada em transformar dados em informação e produto inteligente, formada em **estatística** pela UnB, estudante de **computação, instrutora** de Data Science na Alura, atualmente **cientista de dados** na Caju e líder da squad de **Vivências** no DataGirls, adora andar de bike, fazer trilhas e conhecer novas praias e cachoeiras.

Nossa **Agenda**

Nosso **objetivo** é ter contato com os **principais comando de SQL**, para criar, consultar e alterar dados, focado em conceito e prática.

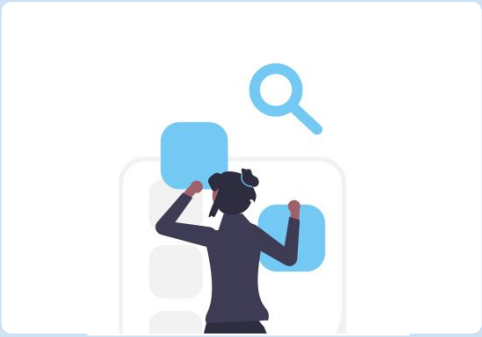
Nossa **dinâmica** será híbrida, teremos a (01) apresentação do **conceito**, (02) prática do **código** e posteriormente um (03) **checkpoint**.

Onde encontro o material?

Vamos passar os poucos aqui no chat e você também pode conferir no GitHub

Preciso fazer o código junto?

O ideal é fazer juntas, separe a tela do treinamento e de um navegador para praticarmos juntas.



Tenho dúvidas e agora?

Coloque no chat e/ou procure a Mari nosso ponto de apoio.

SQL

SQL

Structured **Q**uery **L**anguage

linguagem de consulta estruturada

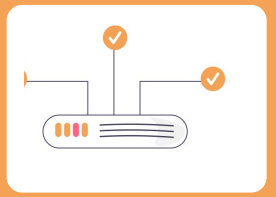
DDL

Data **Definition** Language

CREATE

ALTER

DROP



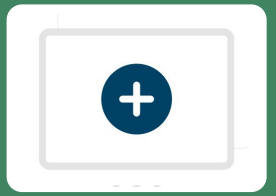
DML

Data **Manipulation** Language

INSERT

UPDATE

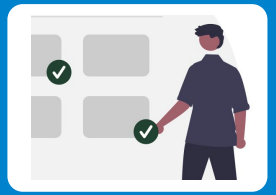
DELETE



DQL

Data **Query** Language

SELECT



Hora do **Checkpoint**

Alinhar o objetivo e possibilidades com SQL

menti.com

7740 7126

O que significa SQL?

System Query Language

Structured Query Language

Structured Query Logic

Quais afirmações estão corretas?

Os comandos create e insert são para definição de dados (DDL)

Podemos deletar linhas específicas utilizando o DROP.

Consultamos os dados apenas um comando de Query (DQL) o Select

Nosso **Case**

Destiny

A Destiny é uma **empresa de cosméticos** que oferece uma ampla gama de produtos, incluindo cremes, maquiagens e perfumes.

Atualmente a empresa armazena as informações sobre **produtos**, **clientes** e **vendas** em planilhas e até cadernos e gostaria de uma solução mais **segura e prática.**

Modelagem do banco de dados

Destiny

SGBD

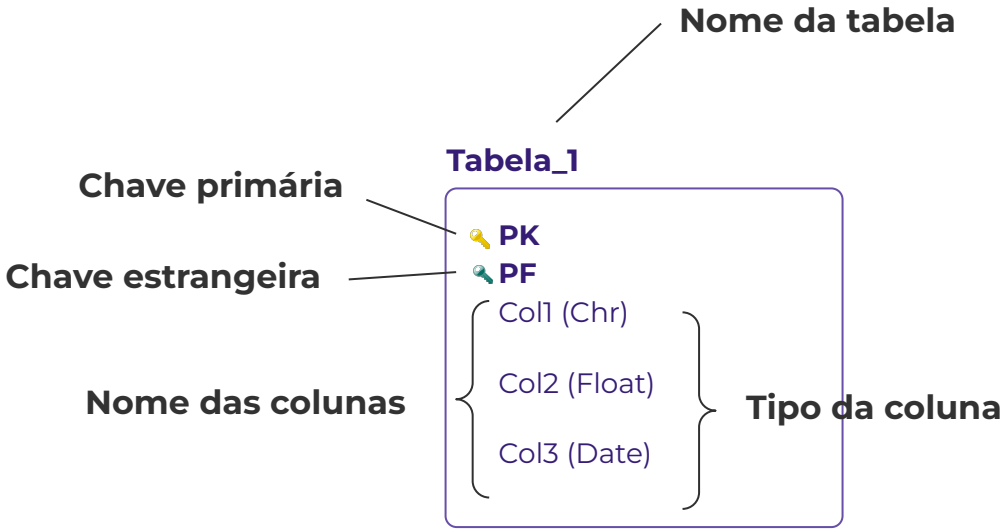
Sistemas de **G**erenciamento de **B**anco de **D**ados Relacionais

SGBD

Sistemas de **G**erenciamento de **B**anco de **D**ados Relacionais

Vamos **Praticar!**

Chegou a hora de **criar as tabelas** da empresa que irão receber as informações no futuro



Hora do **Checkpoint**

Modelagem, SGBD, Criação de tabelas e relacionamentos com SQL

menti.com

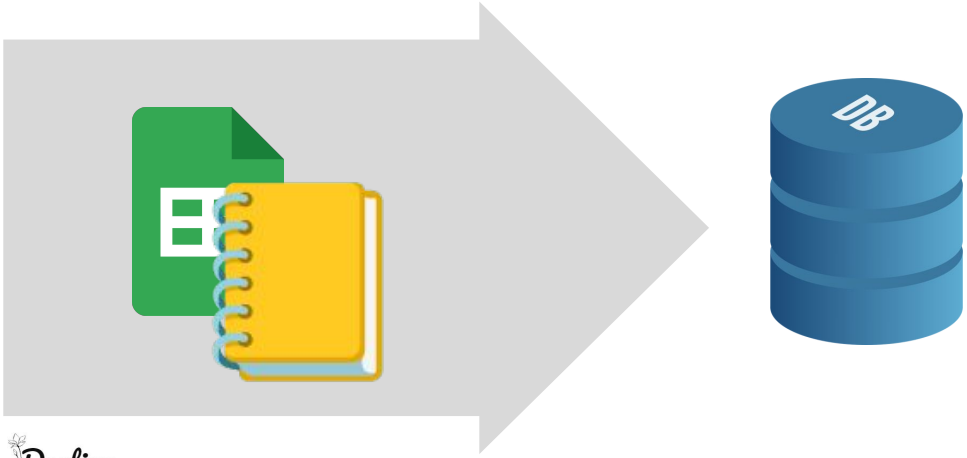
7740 7126

O que é modelagem de dados e qual a sua função

qual a diferenca entre SQL e SGBD

Vamos **Praticar!**

Vamos **inserir** os dados nas tabelas criadas no banco de dados.



Destiny

Hora do **Checkpoint**

Inserção de dados, seus tipos e sintaxe

menti.com

7740 7126

Podemos inserir de quais formas

Ufa!

Dados **estruturados** e **armazenados**!

Destiny

Vamos **Praticar!**

Realizar 3 consultas:

1. Conferência de **todos os dados** inseridos
2. Verificar os produtos **mais caros que** 100 reais
3. Quais categorias de cosméticos estão disponíveis para venda?
4. Quais são os cinco produtos mais baratos?

Vamos **Praticar!**

Precisamos ajudar uma cliente para o presente de um amigo-oculto

1. O presente tem que ser entre 20 e 50 reais
2. Será que ela prefere um tipo de produto específico?

Vamos **Praticar!**

Precisamos atualizar alguns dados

1. Uma venda foi registrada errada, precisamos excluir.
2. A cliente Ana trocou de e-mail.

Hora do **Checkpoint**

Selecionando todos os dados e dados específicos.

menti.com

7740 7126

Indo além

Dados **conferidos** e familiarizados!

Destiny

Vamos **Praticar!**

Precisamos de alguns números para serem repassados para a gestão

1. Quantos clientes temos? E quantos são de São Paulo?
2. Qual a média de preço por categoria de produto?
3. Quantos vendedores temos por loja?
4. Quantos produtos foram vendidos no total?

Quais desses clientes são da cidade de São paulo?

Ana Silva e Carlo Pereira

Camila Almeida e Fernanda Costa

Bruna Martins e Ricardo Oliveira

Quais desses clientes são da cidade de São paulo?

Ana Silva e Carlo Pereira

Camila Almeida e Fernanda Costa

Bruna Martins e Ricardo Oliveira

Hora do **Checkpoint**

Fazendo análises com os dados.

menti.com

7740 7126

Estamos indo bem...

Chegaram novas demandas!

Destiny

Vamos **Praticar!**

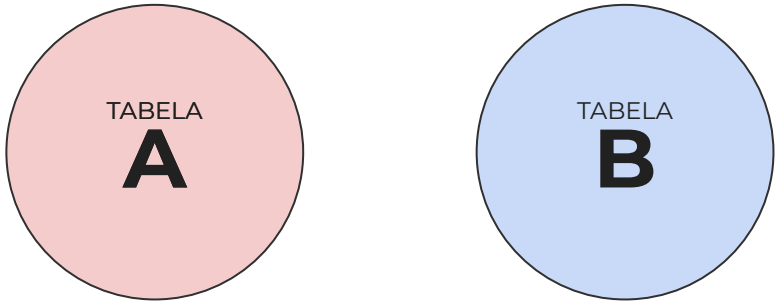
Solicitaram mais números!

1. Nome dos Produtos vendidos?
2. Nome dos Clientes que fizeram as compras.
3. Quais clientes não fizeram compras?
4. Qual o faturamento de agosto?

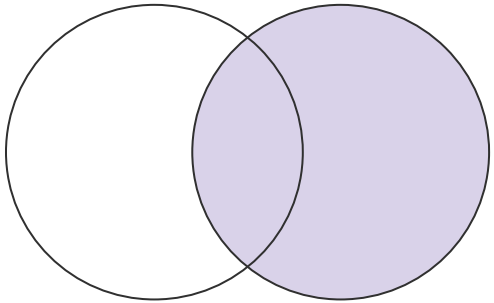
Modelagem do banco de dados

Destiny

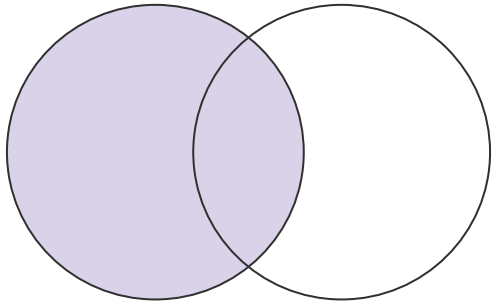
Juntando
tabelas



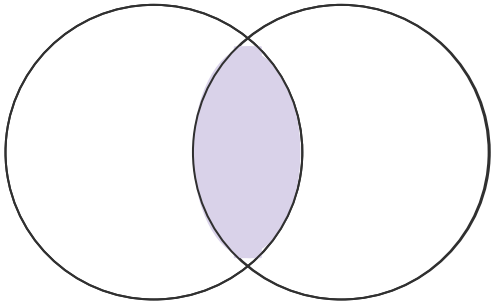
Juntando tabelas



RIGHT JOIN



LEFT JOIN



INNER JOIN

Hora do **Checkpoint**

Consultado dados em tabelas diferentes

menti.com

7740 7126

Quais clientes não fizeram compras?

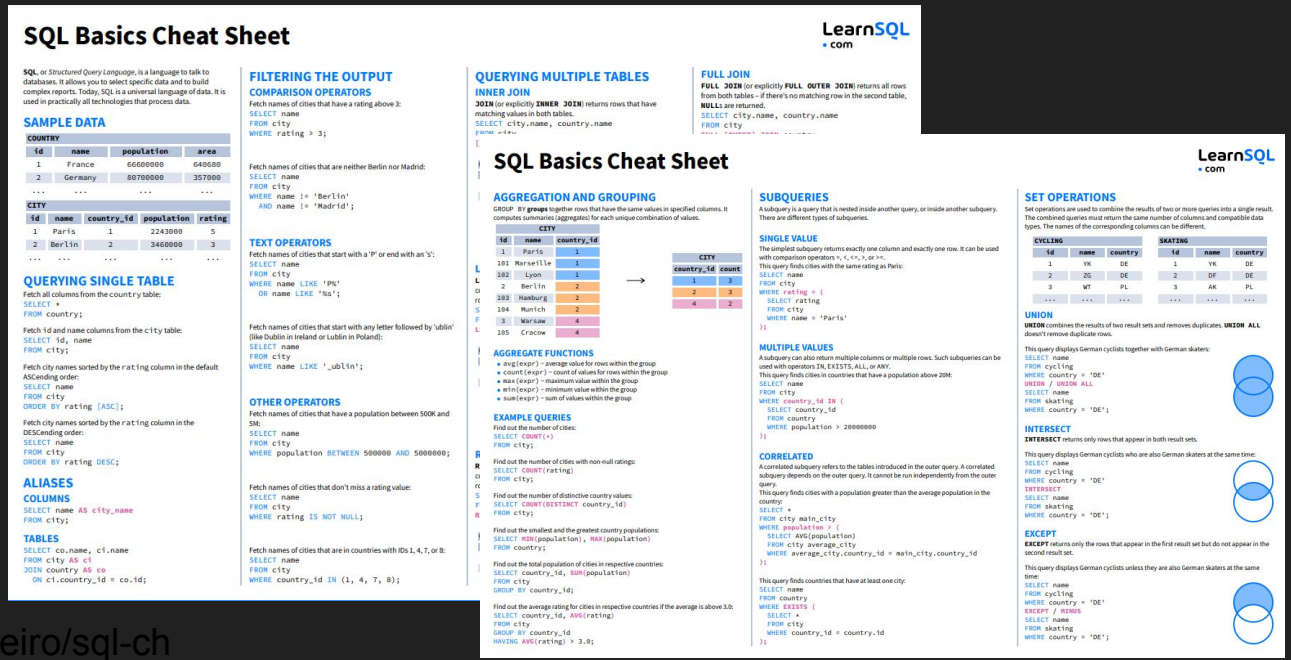
```
SELECT c.ID, c.Nome, c.Email, c.Telefone, c.Cidade

FROM Clientes c

LEFT JOIN Vendas v ON c.ID = v.ClienteID

WHERE v.ID IS NULL;
```

Materiais para **fixar**



SQL Basics Cheat Sheet

LearnSQL.com

SQL, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

COUNTRY

id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY

id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

QUERYING SINGLE TABLE

Fetch all columns from the country table:

```
SELECT *
FROM country;
```

Fetch id and name columns from the city table:

```
SELECT id, name
FROM city;
```

Fetch city names sorted by the rating column in the default ASCending order:

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

Fetch city names sorted by the rating column in the DESCending order:

```
SELECT name
FROM city
ORDER BY rating DESC;
```

ALIASES

COLUMNS

```
SELECT name AS city_name
FROM city;
```

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

FILTERING THE OUTPUT

COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
  AND name != 'Madrid';
```

TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%'
  OR name LIKE '%s';
```

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

Fetch names of cities that don't miss a rating value:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

Fetch names of cities that are in countries with IDs 1, 4, 7, or 8:

```
SELECT name
FROM city
WHERE country_id IN (1, 4, 7, 8);
```

QUERYING MULTIPLE TABLES

INNER JOIN

JOIN (or explicitly **INNER JOIN**) returns rows that have matching values in both tables.

```
SELECT city.name, country.name
FROM city
```

FULL JOIN

FULL JOIN (or explicitly **FULL OUTER JOIN**) returns all rows from both tables – if there's no matching row in the second table, **NULL**s are returned.

```
SELECT city.name, country.name
FROM city
```

SQL Basics Cheat Sheet

LearnSQL.com

AGGREGATION AND GROUPING

GROUP BY groups together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY

id	name	country_id
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	Munich	2
3	Warsaw	4
105	Cracow	4

→

CITY

country_id	count
1	3
2	3
4	2

AGGREGATE FUNCTIONS

- `avg(expr)` – average value for rows within the group
- `count(expr)` – count of values for rows within the group
- `max(expr)` – maximum value within the group
- `min(expr)` – minimum value within the group
- `sum(expr)` – sum of values within the group

EXAMPLE QUERIES

Find out the number of cities:

```
SELECT COUNT(*)
FROM city;
```

Find out the number of cities with non-null ratings:

```
SELECT COUNT(rating)
FROM city;
```

Find out the number of distinctive country values:

```
SELECT COUNT(DISTINCT country_id)
FROM city;
```

Find out the smallest and the greatest country populations:

```
SELECT MIN(population), MAX(population)
FROM country;
```

Find out the total population of cities in respective countries:

```
SELECT country_id, SUM(population)
FROM city
GROUP BY country_id;
```

Find out the average rating for cities in respective countries if the average is above 3.0:

```
SELECT country_id, AVG(rating)
FROM city
GROUP BY country_id
HAVING AVG(rating) > 3.0;
```

SUBQUERIES

A subquery is a query that is nested inside another query, or inside another subquery. There are different types of subqueries.

SINGLE VALUE

The simplest subquery returns exactly one column and exactly one row. It can be used with comparison operators `=`, `<`, `<=`, `>`, or `>=`.
This query finds cities with the same rating as Paris:

```
SELECT name
FROM city
WHERE rating = (
  SELECT rating
  FROM city
  WHERE name = 'Paris'
);
```

MULTIPLE VALUES

A subquery can also return multiple columns or multiple rows. Such subqueries can be used with operators `IN`, `EXISTS`, `ALL`, or `ANY`.
This query finds cities in countries that have a population above 20M:

```
SELECT name
FROM city
WHERE country_id IN (
  SELECT country_id
  FROM country
  WHERE population > 20000000
);
```

CORRELATED

A correlated subquery refers to the tables introduced in the outer query. A correlated subquery depends on the outer query. It cannot be run independently from the outer query.
This query finds cities with a population greater than the average population in the country:

```
SELECT *
FROM city main_city
WHERE population > (
  SELECT AVG(population)
  FROM city average_city
  WHERE average_city.country_id = main_city.country_id
);
```

This query finds countries that have at least one city:

```
SELECT name
FROM country
WHERE EXISTS (
  SELECT *
  FROM city
  WHERE country_id = country.id
);
```

SET OPERATIONS

Set operations are used to combine the results of two or more queries into a single result. The combined queries must return the same number of columns and compatible data types. The names of the corresponding columns can be different.

CYCLING

id	name	country
1	YK	DE
2	ZG	DE
3	WT	PL
...	...	...

SKATING

id	name	country
1	YK	DE
2	DF	DE
3	AK	PL
...	...	...

UNION

UNION combines the results of two result sets and removes duplicates. **UNION ALL** doesn't remove duplicate rows.

This query displays German cyclists together with German skaters:

```
SELECT name
FROM cycling
WHERE country = 'DE'
UNION / UNION ALL
SELECT name
FROM skating
WHERE country = 'DE';
```

INTERSECT

INTERSECT returns only rows that appear in both result sets.

This query displays German cyclists who are also German skaters at the same time:

```
SELECT name
FROM cycling
WHERE country = 'DE'
INTERSECT
SELECT name
FROM skating
WHERE country = 'DE';
```

EXCEPT

EXCEPT returns only the rows that appear in the first result set but do not appear in the second result set.

This query displays German cyclists unless they are also German skaters at the same time:

```
SELECT name
FROM cycling
WHERE country = 'DE'
EXCEPT / MINUS
SELECT name
FROM skating
WHERE country = 'DE';
```

https://github.com/manuseiro/sql-cheat-sheet

Obrigada!

Spoiler: Vou adicionar uma lista de perguntas no GitHub para você treinar! Fica de olho 👀