# AULA 5 E 6

CREATE DATABASE

CREATE TABLE simples,

CREATE TABLE com SELECT (<https://www.postgresql.org/docs/current/sql-createtableas.html> )

CREATE INDEX (<https://www.postgresql.org/docs/15/sql-createindex.html> )

<https://medium.com/analytics-vidhya/materialized-view-vs-view-in-postgresql-1ddb4fe86cb7>

CREATE VIEW (<https://www.postgresql.org/docs/15/sql-createview.html> )

CREATE MATERIALIZED VIEW (<https://www.postgresql.org/docs/15/sql-creatematerializedview.html> )

CONSTRAINTS (<https://www.postgresql.org/docs/15/ddl-constraints.html> ),

PRIMARY KEY e

FOREIGN KEY simples e compostas.

ALTER TABLE (<https://www.postgresql.org/docs/current/sql-altertable.html> )

DROP TABLE (<https://www.postgresql.org/docs/15/sql-droptable.html> )

DROP VIEW (<https://www.postgresql.org/docs/15/sql-dropview.html> )

REFRESH MATERIALIZED VIEW (<https://www.postgresql.org/docs/15/sql-refreshmaterializedview.html> ).

Implicações da integridade referencial na alteração de tabelas. Uso das cláusulas ON DELETE e ON UPDATE CASCADE.

SELECT, (<https://www.postgresql.org/docs/15/sql-select.html> )

LIMIT,

DISTINCT,

ORDER BY

CASE (<https://www.postgresql.org/docs/15/functions-conditional.html#FUNCTIONS-CASE> ).

SELECT \*, CASE WHEN nota>=7 THEN 'APROVADO' ELSE 'REPROVADO' END AS STATUS FROM nota;

SELECT aluno\_id, disciplina\_id, CASE WHEN nota>=7 THEN 'APROVADO' ELSE 'REPROVADO' END AS STATUS FROM nota;

SELECT \*, CASE WHEN nota>=7 THEN 'APROVADO' ELSE 'REPROVADO' END AS STATUS FROM nota WHERE disciplina\_id=1;

AND

SELECT \* FROM aluno WHERE data\_nascimento >= '1995-01-01' AND email LIKE '%@%.%';

OR

SELECT \* FROM aluno WHERE data\_nascimento>='1996-01-01' OR data\_nascimento<='1990-01-01'

IN

IS NULL

IS NOT NULL

LIKE

SELECT \* FROM aluno WHERE data\_nascimento >= '1995-01-01' AND nome\_completo\_aluno ILIKE '%santos%';

BETWEEN

SELECT \* FROM aluno WHERE data\_nascimento BETWEEN '1995-01-01' AND '1996-12-31';

SELECT \* FROM aluno WHERE data\_nascimento NOT BETWEEN '1995-01-01' AND '1996-12-31';

SELECT \* FROM aluno WHERE NOT data\_nascimento BETWEEN '1995-01-01' AND '1996-12-31';

Uso de alias (AS) para renomear colunas e tabelas.

Sub-consultas utilizando a cláusula WITH.

Utilizando funções prontas:

CURRENT\_DATE, CURRENT\_TIME e NOW.

Utilizando funções de formatação de dados:

<https://www.postgresql.org/docs/15/functions-math.html>

ROUND

<https://www.postgresql.org/docs/15/functions-datetime.html#FUNCTIONS-DATETIME-EXTRACT>

DATE\_PART,

SELECT \* FROM aluno WHERE EXTRACT(MONTH FROM CAST(data\_nascimento AS TIMESTAMP))=11

SELECT \* FROM ALUNO WHERE date\_part('month', cast(data\_nascimento as timestamp)) = 11

DATE\_TRUNC

<https://www.postgresql.org/docs/15/functions-string.html>

CONCAT

TRIM

LOWER,

SELECT lower(nome\_completo\_aluno), \* FROM aluno WHERE data\_nascimento >= '1995-01-01' AND UPPER(nome\_completo\_aluno) LIKE '%SANTOS%';

UPPER,

SELECT \* FROM aluno WHERE data\_nascimento >= '1995-01-01' AND UPPER(nome\_completo\_aluno) LIKE '%SANTOS%';

SUBSTRING

POSITION

REPLACE

REVIEW

SELECT \*

FROM aluno

WHERE data\_nascimento BETWEEN '1995-01-01' AND '1996-12-31';

SELECT \*

FROM aluno

WHERE EXTRACT(YEAR FROM CAST(data\_nascimento AS TIMESTAMP)) BETWEEN 1995 AND 1996;

select \* from aluno where substring(data\_nascimento, 0, 5) in ('1995', '1996');

# AULA 7

Agrupamento de dados utilizando GROUP BY e funções de agregadoras: MAX, AVG, SUM, COUNT, MIN e MAX.

<https://www.postgresql.org/docs/current/tutorial-agg.html>

<https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-group-by/>

<https://www.w3resource.com/PostgreSQL/postgresql-group-by.php>

Diferenças entre as cláusulas WHERE e HAVING.

Junção de tabelas e o produto cartesiano das tuplas. CROSS JOIN, INNER JOIN, RIGHT JOIN, LEFT JOIN, FULL JOIN e SELF JOIN.

<https://www.postgresql.org/docs/current/tutorial-join.html>

<https://www.devmedia.com.br/inner-cross-left-rigth-e-full-joins/21016>

<https://terminalroot.com.br/2019/10/inner-join-left-join-right-join-mysql.html>

Junção de tabelas e a união dos dados. UNION e UNION ALL.

Links Extras

<https://podprogramar.com.br/introducao-a-sql/>

<https://uibakery.io/sql-playground>