PostgreSQL

DDL

Data Definition Language

-- Postgres makes columns and tables names small by default, use " " to force case sensitivity.(Not Recommended)

أسماء الجداول والأعمدة تتحط بين DOUBLE QUOTES لو عايز تلتزم بحالة الأحرف

بينما القيم النصية VALUES لابد أن توضع بين SINGLE QUOTES

CREATE

Data Types:

INTEGER: عدد صحيح (4 BYTES)

BIGINT: عدد صحيح ضخم جدًا أكثر من 2 مليار

NUMERIC: عدد عشري دقيق جدًا للمال (الخانة العشرية ,الدقة)

REAL/DOUBLE PRECISION: عدد عشري تقريبي

VARCHAR(n): نص متغير بحد أقصى n

TEXT: نص مفتوح الطول

CHAR(n): نص ذو طول ثابت يكمل بمسافات

DATE: تاريخ بدون وقت

TIMESTAMP: تاريخ ووقت

BOOLEAN: TRUE or FALSE

SERIAL: INTEGER increases automatic

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) DEFAULT 'No Name' NOT NULL,
    last_name VARCHAR(50),
    hire_date DATE DEFAULT CURRENT_DATE NOT NULL,
    date_time TIMESTAMP DEFAULT NOW(),
    email VARCHAR(20)
);
```

CURRENT_DATE: دالة ترجع تاريخ اليوم

NOW(): تاريخ اليوم والوقت

```
CREATE TABLE students (
    name VARCHAR(50) NOT NULL,
    id INTEGER DEFAULT 1234567,
    phone_number CHAR(11),
    grade VARCHAR(10) DEFAULT 'Good' NOT NULL,
    gpa NUMERIC(3, 2) NOT NULL
);
```

PRIMARY KEY: UNIQUE + NOT NULL

وبيعمل فهرس تلقائي ع العمود عشان يكون سريع في البحث (القراءة) ولكن بطيء في الكتابة.

```
SELECT * FROM actor;
```

```
SELECT * FROM actor WHERE 1=1;
SELECT * FROM employees WHERE 1 = 2;
```

نسخ الجدول والمحتوى بدون القيود:

```
CREATE TABLE actor_copy AS SELECT * FROM actor;
SELECT * FROM actor_copy;
-----------------------------------------
CREATE TABLE employees_copy AS
SELECT * FROM employees WHERE 1 = 2;
SELECT * FROM employees_copy;
-----------------------------------------
CREATE TABLE family_films AS
SELECT * FROM film
WHERE rating = 'G';
SELECT * FROM family_films;
-----------------------------------------
CREATE TABLE film_copy AS
SELECT title, release_year, length FROM film;
SELECT * FROM film_copy;
-----------------------------------------
-- (Aliasing)
```

```
CREATE TABLE customer_contacts AS
SELECT
    first_name f_name,
    last_name l_name,
    email contact_info
FROM customer;
SELECT * FROM customer_contacts;
-----------------------------------
-- (Priority Rule)
CREATE TABLE actor_naming_test(fname, lname) AS
SELECT
    first_name,          --fname
    last_name surname    --surname هيتم تجاهله تمامًا
FROM actor;
SELECT * FROM actor_naming_test;
-----------------------------------
CREATE TABLE film_pricing (movie_title, release_year,
weekly_cost) AS
SELECT
```

```
    title,
    release_year,
    rental_rate * 7 -- suppose rentel_rate = daily
FROM film;
SELECT * FROM film_pricing;
```

```
CREATE VIEW actor_view AS
SELECT * FROM actor;
-- VIEW ZERO space - مجرد نافذة للاطلاع ع الجدول
```

ALTER

```
CREATE TABLE actor_copy AS
SELECT * FROM actor;
ALTER TABLE actor_copy
ADD COLUMN role VARCHAR(50) DEFAULT
'Protagonist' NOT NULL;
SELECT * FROM actor_copy;
```

```
ALTER TABLE actor_copy
ADD COLUMN fax_number VARCHAR(11),
ADD COLUMN birth_date DATE,
```

```
ADD COLUMN password VARCHAR(10) DEFAULT 'abc1234';

SELECT * FROM actor_copy;
```

كلمة عمود اختيارية والأفضل كتابتها (ADD (COLUMN

Virtual Primary Key:

اتفاق بينك وبين DBeaver إن العمود id مثلاً دهunique

فبيسمحلك تعدل من ال GUI بناءً على ذلك ولكن نظام postgres

مش بيعرف حاجة عنه وممكن يكرر ال id عادي وترجع تعدل من

Dbeaver يلاقي ال id مكرر يضرب في وشك.

Physical Primary Key: قيد حقيقي يعرفه الجميع

```
ALTER TABLE actor_copy

ALTER COLUMN password TYPE VARCHAR(50);
```

```
ALTER TABLE actor_copy
    ALTER COLUMN fax_number TYPE VARCHAR(17),
    ALTER COLUMN fax_number SET DEFAULT '-',
    ALTER COLUMN password TYPE VARCHAR(20),
    ALTER COLUMN password SET NOT NULL;
```

```
ALTER TABLE actor_copy DROP COLUMN fax_number;

ALTER TABLE employees_copy

DROP COLUMN IF EXISTS fax_number;
```

--IF EXISTS لو مش موجود لا يعطي خطأ
--DROP (COLUMN)

```
ALTER TABLE actor_copy

DROP COLUMN last_update,

DROP COLUMN password;

SELECT * FROM actor_copy;
```

```
REVOKE INSERT, UPDATE, DELETE, TRUNCATE

ON actor_copy FROM public;
```

-- كلمة public تعني كل المستخدمين العاديين
-- أنت ك super user تقدر تعدل عادي
-- REVOKE إلغاء أو سحب

```
ALTER TABLE actor_copy

ADD COLUMN gender CHAR(1),

ADD CONSTRAINT gender_check

CHECK (gender IN ('M', 'F'));
```

In DBeaver GUI:

لما بتسيب الخلية النصية زي ما هي بتكون قيمتها NULL غير معرفة
لأنك لم تدخل أي قيمة إنما لو ضغطت عليها وحاولت التعديل ثم
تركتها فارغة أصبحت قيمتها empty string وعشان ترجعها NULL
كليك يمين ثم EDIT SELL ثم SET TO NULL

```
tablename_columnname_type
-- Naming convention for constraint
-----------------------------------------
ALTER TABLE actor_copy
DROP COLUMN gender;
GRANT INSERT, UPDATE, DELETE, TRUNCATE
ON actor_copy TO public;
-- Grant منح صلاحية
-----------------------------------------
ALTER TABLE actor_copy
RENAME COLUMN birth_date TO birthday;
-----------------------------------------
ALTER TABLE actor_copy
RENAME TO actor_backup;
```

DROP & TRUNCATE

```
DROP TABLE actor_backup;
```

بيحذف الجدول نهائيًا الهيكل والبيانات وتحرير المساحة

```
DELETE FROM actor_copy;
```

بيحذف البيانات ولكن يبقى على الهيكل ولكنها عملية بطيئة Row by row processing بيحذف صف صف ويسجل فى ال log والعداد زى id مش بيصفر بيكمل من آخر رقم.

```
TRUNCATE TABLE actor_copy;
```

بيمسح البيانات كلها دفعة واحدة وبيصفر العداد لأنه بيعتمد على فكرة Page Deallocation يعني: بيقطع الورقة كلها ويحرر المساحة.

DELETE مش بتحرر المساحة لأنها لا تمسح البيانات فيزيائيًا هي بتشخبط عليها (بتعلم على الصفوف dead).

```
DELETE FROM jobs WHERE job_id = 'IT_PROG';

-- لحذف صفوف معينة

SELECT COUNT(*) FROM actor;

-- عدد الصفوف

SELECT COUNT(last_name) FROM actor;

-- عدد الصفوف ولكن مش بيحسب NULL
```

DML

INSERT

```
INSERT INTO jobs (job_id, job_title, min_salary, max_salary)

VALUES

    ('IT_PROG', 'Programmer', 4000, 10000),

    ('AD_PRES', 'President', 20000, 40000),

    ('FI_MGR', 'Finance Manager', 12000, 24000);

-- يمكن تغير ترتيب الأعمدة مع تغيير ترتيب القيم المناظرة لها

INSERT INTO jobs
```

```
VALUES ('DATA_ENG', 'Data Engineer', 8000, 21000);
-- You must stick to the order of columns
-----------------------------------------

INSERT INTO jobs (job_id, job_title, min_salary)
VALUES ('DATA_ARCH', 'Data Architecture', 8000);
-- only 3 columns / column4 value will be NULL
-----------------------------------------

ALTER TABLE jobs
ALTER COLUMN max_salary SET DEFAULT 10000;

-----------------------------------------

INSERT INTO jobs (job_id, min_salary)
VALUES ('DATA_ARCH2', 8000);
-- Alert job_title NOT NULL

-----------------------------------------

INSERT INTO jobs
VALUES ('SOFT_TEST', 'Software Tester');

-----------------------------------------

INSERT INTO jobs
VALUES ('DATA_ARCH3', 'Data Architecture3', 8000, NULL);
-- Put NULL instead of default value
INSERT INTO jobs
VALUES ('DATA_ARCH4', 'Data Architecture4', 8000, DEFAULT);
-- Put default value explicitly
-----------------------------------------
```

```
CREATE TABLE jobs_copy AS SELECT * FROM jobs
WHERE 1 = 0;
INSERT INTO jobs_copy SELECT * FROM jobs;
-- The same structure but only data you need
-- Constrains are not copied
```

```
TRUNCATE TABLE jobs_copy;
INSERT INTO jobs_copy SELECT * FROM jobs WHERE
job_id = 'IT_PROG';
```

```
TRUNCATE TABLE jobs_copy;
INSERT INTO jobs_copy (job_id, job_title)
SELECT job_id, job_title  FROM jobs WHERE
job_id = 'IT_PROG';
```

-- بمجرد أن تفتح القوس وتكتب قائمة أعمدة (col1, col2)، أنت توقع عقدًا ملزمًا مع قاعدة البيانات بأنك ستوفر قيمًا مساوية لها تمامًا في العدد والترتيب (val1, val2).

UPDATE

```
CREATE TABLE film_copy AS SELECT * FROM film;
UPDATE film_copy SET length = 90;
```

```
UPDATE film_copy
SET
```

```
    release_year = 2005,
    description = 'Parent Guidance'
WHERE rating = 'PG';
SELECT * FROM film_copy WHERE rating = 'PG';
```

```
UPDATE film_copy
SET length = length + 10 WHERE rating = 'PG';
SELECT * FROM film_copy WHERE rating = 'PG';
```

SELECT

Alias || ColumnOperations

```
SELECT first_name, last_name, email FROM
customer;
```

```
SELECT first_name f_name, last_name l_name,
email e_mail FROM customer;
```

```
SELECT first_name AS f_name, last_name AS
l_name, email AS e_mail FROM customer;
```

```
SELECT first_name AS "My Name", email "E-mail"
FROM customer;
```

```
-- "It enforces case sensitivity"
-----------------------------------------

SELECT 'My Name is Alex' FROM customer;
-- Echo string as many as rows of customer

-----------------------------------------

SELECT 'My Name is ' || first_name FROM
customer;
-- || Concatenation
-----------------------------------------

CREATE TABLE payments AS SELECT * FROM payment;
UPDATE payments SET amount = 5.5;
SELECT 'payment: ' || amount AS total FROM
payments;
-----------------------------------------

SELECT
    first_name || ' ' || last_name AS "Full Name"
FROM
    customer;
-----------------------------------------

SELECT
    address || ', ' || district || ', ' ||
postal_code
AS "full address"
```

```
FROM
    address;
```

```
SELECT CONCAT(address, ', ', district, ', ', postal_code)
AS "full address" FROM address;
```

لو عملت دمج ب || لنص مع NULL القيمة الكلية بتكون غير معرفة
ولو نص مع رقم بيعطي ERROR والحل استخدام دالة CONCAT NULL

```
SELECT
    title,
    rental_rate,
    (rental_rate + 3) * 7 AS weekly_revenue
FROM
    film;
```

```
-- (الوقت والتاريخ)
SELECT NOW();
-- أو للتاريخ فقط
SELECT CURRENT_DATE;
-- طباعة نص ثابت
SELECT 'I am Saif';
```

```
SELECT CURRENT_DATE + 4;
```

```
SELECT title, release_year, 2025 - release_year
AS years FROM film;
```

RelationalOperators BETWEEN...AND... IN

```
SELECT * FROM film WHERE replacement_cost > 20;
-- < >= <= =
SELECT * FROM film WHERE replacement_cost != 19.99;
```

```
SELECT title, replacement_cost FROM film
WHERE replacement_cost BETWEEN 20 AND 25;
-- [20, 25]
-- WHERE replacement_cost >= 20 AND replacement_cost <= 25;
```

```
SELECT * FROM payment
WHERE payment_date BETWEEN '2022-02-01' AND
'2022-02-28';
```

```
SELECT * FROM payment
WHERE payment_date >= '2007-02-01' AND
payment_date < '2007-03-01'
```

زودنا يوم واحد وخلينا (أقل من) عشان يجيب كل معاملات اليوم الأخير لأن ال Timestamp بيحتوي على وقت وبالتالي بيقف عند الساعة 00 بداية اليوم 28-02-2007 00:00:00 فأي معاملة في اليوم بعد 12 لن تحسب.

```
-----------------------------------------

SELECT * FROM actor
WHERE actor_id IN (1, 10, 25, 30, 45);

-----------------------------------------

SELECT * FROM actor
WHERE first_name IN ('NICK', 'TOM', 'JOE');

-----------------------------------------

SELECT * FROM payment
WHERE DATE(payment_date) IN ('2022-02-14', '2022-02-15');
-- DATE() Function converts timestamp to date
SELECT * FROM payment
WHERE payment_date::DATE IN ('2022-02-14', '2022-02-15');
-- Convert data type to DATE
```

_ تحل محل حرف واحد

% تحل محل أي عدد من الحروف (Zero or more)

LIKE used to match patterns

```
SELECT * FROM actor WHERE first_name LIKE '___';
```

الممثلين اللي أسماؤهم مكونة من ثلاث حروف فقط

```
-----------------------------------------

SELECT * FROM customer WHERE first_name LIKE 'J_AN';
```

أربع حروف مماثلة والحرف الثاني أي حاجة

```
-----------------------------------------

SELECT * FROM customer WHERE first_name  LIKE 'AM%';
```

```
-- Starts with AM...
-----------------------------------------
SELECT * FROM customer WHERE first_name LIKE 'A%';
-- Starts with A...
-----------------------------------------
SELECT * FROM customer WHERE first_name LIKE '%A';
-- Ends with ...A
-----------------------------------------
SELECT * FROM address WHERE address LIKE '%e';
-- Ends with small e
-----------------------------------------
SELECT * FROM address WHERE address LIKE '%a%';
-- contains a anywhere
-----------------------------------------
SELECT * FROM actor WHERE first_name LIKE 'JULIA%';
-- JULIA - JULIANNE (أمثلة)
-----------------------------------------
SELECT * FROM actor WHERE first_name LIKE '_R%';
-- Second char is r
-----------------------------------------
SELECT * FROM actor WHERE first_name LIKE '%_R';
                                -- ينتهي بحرفين آخرهم R
-----------------------------------------
```

```
SELECT * FROM actor WHERE first_name LIKE
'%_R%';
```

-- جميع الكلمات التي تحتوي على R بشرط ألا تبدأ بها

```
SELECT * FROM actor WHERE first_name LIKE
'%_R_%';
-- R can't be first or last
-----------------------------------------
SELECT * FROM address WHERE address2 = NULL;
-- NULL is not a value (No Return) NULL means
Unknown
-----------------------------------------
SELECT * FROM address WHERE address2 IS NULL;
-----------------------------------------
SELECT * FROM address WHERE address2 IS NOT
NULL;
-- It has a value
-----------------------------------------
```

المنطق ثلاثي القيم – Three valued logic

كل خلية في الجدول عبارة عن صندوق إما:

- فارغ empty string
- أو NULL قيمة غير معرفة
- أو قيمة سواء نصية أو عددية

ولذلك لما بتترك الخلية بدون قيمة بيعتبرها NULL قيمة مجهولة غير معرفة.

AND OR NOT

```
SELECT * FROM jobs WHERE job_id = 'DATA_ENG' OR
min_salary > 10000;
SELECT * FROM film WHERE replacement_cost > 25
AND rental_duration IN (5, 7);
```

```
SELECT * FROM film WHERE replacement_cost > 25
AND rental_duration NOT IN (5, 7);
```

```
SELECT first_name, last_name, address_id, customer_id
FROM customer
WHERE (address_id = 480 OR address_id = 490)
AND customer_id  > 100;
-- () > NOT > AND > OR
```

```
SELECT * FROM jobs
WHERE job_id = 'IT_PROG'
OR (job_id = 'ST_CLERK' and salary > 5000);
```

```
SELECT first_name, last_name, job_id, salary
FROM employees
WHERE job_id = 'IT_PROG' OR job_id = 'ST_CLERK'
AND salary > 5000;
```

```
SELECT * FROM employees
WHERE salary > 10000 AND department_id = 20
OR department_id = 30;
SELECT * FROM employees
WHERE (job_id = 'IT_PROG' OR job_id = 'ST_CLERK')
-- نجمع الوظائف الأول
AND salary > 5000;
-- نطبق شرط الراتب على القوس كله
-----------------------------------
SELECT * FROM film WHERE NOT rating = 'G';
SELECT title, rating FROM film WHERE rating NOT
IN ('G', 'PG');
SELECT title, description FROM film WHERE
description NOT LIKE '%Drama%';
SELECT title, length FROM film WHERE length NOT
BETWEEN 60 AND 90;
SELECT title, length, release_year FROM film
WHERE NOT (length = 90 AND release_year = 2018);
```

ORDER BY

```
SELECT * FROM actor ORDER BY first_name ASC;
-- From A to Z (ASCE)
SELECT * FROM actor ORDER BY first_name DESC;
-- From Z to A
```

Oracle: ASCII sort (Binary), placing Uppercase before Lowercase ('Z' comes before 'a').
PostgreSQL: Collation sort following natural human order ('a' comes before 'Z') case insensitive.

```
SELECT * FROM actor ORDER BY actor_id DESC;
```

```
SELECT payment_id , (amount*10) / 2 price FROM payment ORDER BY price;
```

ترتيب حسب قيم عمود تم إنشاؤه Sorting by Alias

```
SELECT first_name, last_name FROM actor ORDER BY 2;
-- Position order in SELECT query instead of column name
```

```
SELECT * FROM payment ORDER BY 1;
-- Column 1 in the original table
SELECT * FROM actor ORDER BY first_name, last_name;
```

```
SELECT * FROM actor ORDER BY first_name, last_name, last_update ;
```

```
SELECT * FROM actor ORDER BY first_name DESC, last_name ASC, last_update DESC;
```

```
SELECT * FROM actor ORDER BY 1 DESC, 2 ASC;
-- بيرتب حسب العمود الأول والعمود التاني مهمل إلا أن يحدث تعادل في قيمتين من العمود الأول ولو ثلاثة أعمدة فلن ينظر للثالث إلا إذا تعادل الأول والثاني.
-- ولو هناك صفان بنفس القيم بالظبط بيظلوا كما هم اللي اتعمله insert الأول هو السابق.

-----------------------------------------

SELECT * FROM country_copy ORDER BY country;
SELECT * FROM country_copy ORDER BY country DESC;
-- ASC put NULLS last while DESC put NULLS first;

-----------------------------------------

SELECT * FROM country_copy ORDER BY country
NULLS FIRST;
SELECT * FROM country_copy ORDER BY country DESC
NULLS LAST;
-- NULLS FIRST | NULLS LAST
SELECT * FROM actor LIMIT 5;
-- يعني أول خمس صفوف بس

-----------------------------------------

SELECT title, release_year, length FROM film
WHERE release_year = 2017
ORDER BY length DESC
LIMIT 10;
                                    -- أطول 10 أفلام

-----------------------------------------
```

```
SELECT title, release_year, length FROM film
WHERE release_year = 2017
ORDER BY length DESC
OFFSET 10 LIMIT 10;
```

-- offset 10 يعني سيب أول 10

UPPER(column): كل الحروف كابيتال

LOWER(column): كل الحروف سمول

INITCAP(column): Initial Capital كابيتال ابتدائي

```
SELECT UPPER(city) FROM city;
```

```
SELECT LOWER(city) FROM city;
```

```
SELECT UPPER(job_id),
INITCAP(job_title) FROM jobs;
```

```
SELECT LOWER(first_name), INITCAP('ahmed saif')
FROM actor;
```

```
SELECT * FROM actor WHERE first_name = 'TOM';
-- string in single quotes is case sensitive
```

```
SELECT * FROM actor WHERE LOWER(last_name) =
'dee';
-- retrieve anyone with the name ignoring case
```

```
-- يبحث متجاهلاً الحالة
-- UPPER(last_name) = 'DEE'
-- INITCAP(last_name) = 'Dee'
```

STRING CONCAT

```
SELECT first_name, SUBSTR(first_name, 2, 5),
last_name, LENGTH(last_name)
FROM actor;
-- start from second char and count 5 chars
-- SUBSTR(first_name, 2): start from 2 and
continue to the last
-----------------------------------------
SELECT CONCAT(first_name, ' ', last_name) AS
full_name FROM actor;
SELECT CONCAT_WS(' ', first_name, middle_name,
last_name) AS full_name FROM actor;
-- concat with separator
-- بيوزع الفاصل بين القيم بذكاء
-- يعني بتتجاهل ال NULL لا تضع مسافة قبلها أو بعدها
-- CONCAT() VS ||
    -- لو حاولت تدمج نص مع NULL النتيجة بتكون NULL في حالة || السيئة تعم.
-----------------------------------------
SELECT POSITION('@' IN 'user@gmail.com');
-- RESULT: 5
SELECT STRPOS('user@gmail.com', '@');
```

```
-- RESULT: 5
-----------------------------------------
SELECT version();
                    إصدار POSTGRES
-----------------------------------------
      -- ابدأ البحث من الحرف الرابع، وهات لي التكرار الثاني لعلامة @
SELECT REGEXP_INSTR('user@test@com', '@', 4, 2);
-- RESULT: 10
-- The only one that takes 4 parameters
-----------------------------------------
SELECT REGEXP_INSTR('I am learning oracle', 'o',
1, 1);
-- لازم موضع البداية يكون موجب مينفعش سالب أو صفر
                -- اتجاه البحث قد يتغير ولكن index ثابت لا يتغير.
-- POSTGRES doesn't have reverse search function
like INSTR Oracle.
                -- لما يبحث عن كلمة بيجلب موضع أول حرف فيها
-----------------------------------------
SELECT first_name, STRPOS(first_name, 'A') FROM
actor;
```

TRIM()

```
SELECT TRIM('    My name is Ahmed Saif    ');
-- تزيل المسافات من الأطراف
```

```
-- TRIM() تقص من الأطراف فقط
-----------------------------------------
SELECT TRIM( 'My' FROM 'My name is Ahmed Saif');
SELECT TRIM( 'Saif' FROM 'My name is Ahmed Saif');
-- trim only characters from both sides
-----------------------------------------
SELECT TRIM( 'My naif' FROM 'My name is Ahmed Saif');
-- me is Ahmed S
-- تبدأ من الطرف الأول إذا وجدت الحرف الأول في القائمة تقصه
وهكذا حتى تصطدم بحرف غير موجود فتنتقل للطرف الآخر وتعمل
بنفس الطريقة حتى تصطدم بحرف غير موجود.
-----------------------------------------
SELECT TRIM(BOTH ' ' FROM '  I am Ahmed  ');
SELECT TRIM(LEADING ' ' FROM '  I am Ahmed  ');
SELECT TRIM(TRAILING ' ' FROM '  I am Ahmed  ');
-- كلاهما - المقدمة - الذيل
-- both - left - right
-----------------------------------------
SELECT TRIM(LEADING 'My' FROM 'My Car');
-----------------------------------------
SELECT LTRIM('My name is Saif', 'My ');
-- (text, characters)
-- LEFTTRIM-RIGHTTRIM-BOTHTRIM
SELECT RTRIM('My name is Saif', 'fai');
SELECT BTRIM('me and you', 'muo');
```

```
-----------------------------------------
SELECT LTRIM('      My name is Saif');
SELECT RTRIM('My name is Adammmm', 'm');
-----------------------------------------
SELECT REPLACE('Ahmed', 'A', 'M');
-- REPLACE(source, oldText, newText)
-- Mhmed
-- 3 parameters are mandatory
-----------------------------------------
SELECT first_name, REPLACE(first_name, 'E', 'O')
FROM actor;
-----------------------------------------
SELECT first_name, REPLACE(first_name, 'EN',
'-') FROM actor;
```

Padding

حشو الفراغات برمز معين من اليسار أو اليمين حتى يكتمل عدد الحروف.

```
SELECT LPAD('123', 6, '0');
--000123
SELECT RPAD('Ali', 5, '*');
--Ali**
-----------------------------------------
```

```
SELECT first_name, LPAD(first_name, 10, '*')
left_padding FROM actor;
-----------------------------------------
SELECT first_name, RPAD(first_name, 7, '-') FROM
actor;
-----------------------------------------
SELECT first_name, LPAD('My name is ' || first_name,
20, '-') FROM actor;
```

لو الجملة زادت عن عشرين حرف مثلاً 22 هتعمل truncate للزيادة (حرفين) من جهة اليمين (النهاية) دائمًا.

DATE TO_CHAR() TIMESTAMP

DATE: YYYY-MM-DD

TIMESTAMP: YYYY-MM-DD HH:MI:SS.nnnnnn

```
-----------------------------------------
SELECT last_update FROM actor;
-- TO_CHAR(value, 'format')
SELECT last_update, TO_CHAR(last_update, 'YYYY')
FROM actor;
-- 2022
SELECT last_update, TO_CHAR(last_update, 'YYY')
FROM actor;
-- 022
```

```
SELECT last_update, TO_CHAR(last_update, 'YY')
FROM actor;
-- 22
SELECT last_update, TO_CHAR(last_update, 'MM')
FROM actor;
-- 02
SELECT last_update, TO_CHAR(last_update,
'month') FROM actor;
-- february
SELECT last_update, TO_CHAR(last_update,
'MONTH') FROM actor;
-- FEBRUARY
SELECT last_update, TO_CHAR(last_update,
'Month') FROM actor;
-- February
SELECT last_update, TO_CHAR(last_update, 'Mon')
FROM actor;
-- Feb
SELECT last_update, TO_CHAR(last_update, 'DD')
FROM actor;
-- 15 (رقم اليوم في الشهر)
SELECT last_update, TO_CHAR(last_update, 'DAY')
FROM actor;
-- TUESDAY
```

```
SELECT last_update, TO_CHAR(last_update, 'D')
FROM actor;
-- 3 (رقم اليوم في الأسبوع)
SELECT last_update, TO_CHAR(last_update, 'DDD')
FROM actor;
-- 046 (رقم اليوم في السنة)
SELECT last_update, TO_CHAR(last_update, 'WW')
FROM actor;
-- 07 (رقم الأسبوع في السنة)
SELECT payment_date, TO_CHAR(payment_date, 'HH')
FROM payments;
-- 12-hour system
SELECT payment_date, TO_CHAR(payment_date,
'HH24') FROM payments;
-- 24-hour system
SELECT last_update, TO_CHAR(last_update,
'MI:SS:MS PM') FROM actor;
-- 34:33:000
SELECT payment_date, TO_CHAR(payment_date,
'FMMonth DD, YYYY') FROM payment;
-- January 31, 2022
-- No extra spaces
-- FM = Fill Mode
-- كل شهر ياخد مساحته بالظبط مش تحجز سبتمبر للكل
```

```
SELECT payment_date, TO_CHAR(payment_date, 'DY')
FROM payment;
-- SAT
SELECT payment_date, TO_CHAR(payment_date,
'DD/MM/YYYY') FROM payment;
-- 25/01/2022
-----------------------------------
SELECT TO_CHAR(1250.5, '9,999.99');
-- 1,250.50
-- ساب فراغ للسالب ووضع كل رقم في الخانة المناظرة له
-- لو رقم مش موجود سيضع فراغ في مكانه
-- الفاصلة بين كل ثلاث أرقام
-- لو رقم ناقص في الدقة العشرية سيضع 0 مكانه
SELECT TO_CHAR(31250, '9,999.99');
--  لازم عدد الـ 9 في النمط يكون مساويًا أو أكبر من عدد خانات
الرقم
-- كي لا يقص الرقم الأصلي ويسبب خسائر
SELECT TO_CHAR(31250, '999,999.99');
--  31,250.00 (فراغين في البداية)
-----------------------------------
SELECT TO_CHAR(0.75, '9.99');
--  .75
--  الرقم 9 لما بيتعامل مع 0 في اليسار مش بيعترف بيه وبيسيب
مكانه فارغ
SELECT TO_CHAR(0.75, '0.99');
```

```
-- 0.75
SELECT TO_CHAR(5777, '00,000.0');
-- 05,777.0
-- الرقم صفر بيضع 0 مكان العدد الزيادة بدل ما يسيب مكانه فاضي
مثل 9
```

CASE

```
SELECT title, length,
    CASE
        WHEN length < 60 THEN 'Short Movie'
        WHEN length BETWEEN 60 AND 100 THEN
'Medium Movie'
        ELSE 'Long Movie'
    END AS "Film Duration"
FROM film;
```

```
SELECT title, rating,
    CASE rating
        WHEN 'G' THEN 'General'
        WHEN 'PG' THEN 'Parental Guidance'
        WHEN 'R' THEN 'Restricted'
        ELSE 'Other'
    END Genre
FROM film;
```

```
SELECT job_title, min_salary,
    CASE
        WHEN min_salary <= 7000 THEN min_salary * 1.5
        WHEN min_salary <= 8000 AND min_salary > 7000
THEN min_salary + 1000
        ELSE min_salary
    END "New_Salary"
FROM jobs;
```

```
SELECT job_title, min_salary,
    CASE
        WHEN min_salary <= 7000 THEN min_salary
* 1.5
        WHEN min_salary <= 8000 AND min_salary >
7000 THEN min_salary + 1000
        WHEN job_title = 'Data Engineer' THEN
min_salary * 2
        ELSE min_salary
    END "New_Salary"
FROM jobs;
```

-- بيمشي صف صف وبيقف عند الصف ويمشي case case لو case تحققت مش بيبص ع اللي بعدها.

```
SELECT * FROM jobs WHERE
    (CASE
        WHEN min_salary <= 7000 THEN 1
```

```
        WHEN job_title = 'Data Engineer' THEN 1
        ELSE 0
    END) = 1;
```

Aggregate Functions

دوال التجميع

```
SELECT AVG(amount) FROM payment;
SELECT ROUND(AVG(amount), 2) FROM payment;
-- 4.20 (عددين عشريين)
SELECT ROUND(AVG(amount), 2) FROM payment WHERE
payment_date::DATE = '2022-01-25';
-- '' التواريخ تعتبر نصوص توضع بين
-----------------------------------------
SELECT COUNT(amount) FROM payments WHERE
payment_date::DATE = '2022-01-25';
-- COUNT only NOT NULL VALUES
SELECT COUNT(*) FROM payments WHERE
payment_date::DATE = '2022-01-25';
-- COUNT ALL VALUES
-----------------------------------------
SELECT SUM(amount) FROM payments WHERE
payment_date::DATE = '2022-01-25';
-- المجموع لكل القيم
```

```
SELECT COUNT(*) FROM payment;
-- Number of all rows
-----------------------------------------
SELECT COALESCE(amount, 0) FROM payments;
-- Make NULL values in amount = 0 (retrieve not
update)
SELECT 5 + COALESCE(NULL, 0);
-- INSTEAD OF 5 + NULL = NULL | 5 + 0 = 5
SELECT COALESCE(address, address2, district, 'No
Info') AS address_info
FROM address;
-- Row by row if address has value retrieve and
if NULL check the second so on
-----------------------------------------
SELECT AVG(amount) FROM payments;
SELECT AVG(COALESCE(amount, 0)) FROM payment;
-- NULLS عشان يحسب العدد بتاع ال
SELECT ROUND(AVG(COALESCE(amount, 0)), 2) FROM
payment;
-----------------------------------------
SELECT MIN(first_name) FROM actor;
SELECT MAX(first_name) FROM actor;
-----------------------------------------
SELECT SUM(amount), SUM(ALL amount),
```

```
SUM(DISTINCT amount) FROM payment;
-- ALL is the default mode - DISTINCT removes duplication
-----------------------------------------
SELECT
    COUNT(*) AS total_transactions, -- عدد العمليات
    SUM(amount) AS total_revenue, -- إجمالي الدخل
    MAX(amount) AS max_payment, -- أكبر مبلغ
    MIN(amount) AS min_payment, -- أصغر مبلغ
    ROUND(AVG(amount), 2) AS avg_payment FROM payment; -- متوسط الدفع
-----------------------------------------
SELECT amount, SUM(amount) FROM payment; -- Error
-- الكويري الأولى صفوف والتانية صف واحد لا يمكن جمعهم في جدول
```

GROUP BY

```
SELECT rating FROM film ORDER BY rating;
SELECT DISTINCT rating FROM film;
-- Only filter
SELECT rating FROM film GROUP BY rating;
-- Remove Duplication by grouping equal values in the same group
-- تجهيز لإجراء عمليات
SELECT rating, COUNT(rating), SUM(length) FROM film GROUP BY rating;
```

```
SELECT rating, COUNT(*), SUM(length) FROM film
GROUP BY rating;
-- COUNT(*)
-- عشان لو فيه جروب أفلام rating NULLS تعدهم
-----------------------------------------
SELECT rating, rental_rate, COUNT(*),
SUM(length) FROM film
GROUP BY rating, rental_rate;
-----------------------------------------
SELECT rating, rental_rate, COUNT(*),
SUM(length) FROM film
GROUP BY rating, rental_rate
ORDER BY rating, rental_rate, count(*) DESC;
-----------------------------------------
SELECT rating, rental_rate FROM film
GROUP BY rating, rental_rate;
-----------------------------------------
SELECT rental_rate FROM film
GROUP BY rating, rental_rate;
-- الأعمدة الموجودة في SELECT لازم تتواجد في GROUP BY
-- والعكس غير صحيح عشان عدد الصفوف يكون متكافئ
-- الكلام خاص بالأعمدة الأصلية في الجدول
-- خاصة بالعرض SELECT
-----------------------------------------
SELECT rating, avg(rental_rate) FROM film
```

```
WHERE avg(rental_rate) > 2.9 GROUP BY rating;
-- WHERE sees only original columns not runtime columns
-- so it can't be used with aggregate function
```

-- WHERE بتشيك على عملية لم تحسب بعد وبالتالي عمود غير موجود أصلاً فمش هتنفع هنا ولا مع alias columns

---

HAVING

فلترة المجموعات

```
SELECT rating, avg(rental_rate) FROM film
GROUP BY rating HAVING avg(rental_rate) > 3;
-- HAVING works on calculated runtime columns
```

Order of query syntax:

```
SELECT => FROM => WHERE => GROUP BY => HAVING => ORDER BY
```

```
FROM/JOIN -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY -> LIMIT/OFFSET
```

```
SELECT SUM(amount) FROM payment
HAVING SUM(amount) > 1000000;
-- HAVING here considers the full table as one group
-- HAVING in most cases comes after GROUP BY as it works only on groups
```