HStore is a key-value pair data type in PostgreSQL that allows you to store sets of key-value pairs within a single value. It's a useful tool for storing semi-structured data where the schema is not known in advance. Here's how you can create, store, and retrieve HStore data in PostgreSQL, using an example with an orders table:

**Enable HStore Extension**:

First, ensure that the HStore extension is enabled in your PostgreSQL database. You can do this by running the following SQL command as a superuser:

CREATE EXTENSION IF NOT EXISTS hstore;

**Create Orders Table**:

Let's create an orders table with an HStore column to store order details:

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

customer\_id INT,

order\_details HSTORE

);

Insert data

INSERT INTO orders (customer\_id, order\_details)

VALUES (1, 'color => "blue", size => "large", quantity => "2"'),

(2, 'color => "red", size => "medium", quantity => "3"');

Retrive hstore

**Retrieve HStore Data**: You can retrieve data from the HStore column using various functions provided by PostgreSQL. For example, you can use the **->** operator to access a specific key or the **#>** operator to retrieve multiple keys at once.

-- Retrieve all orders

SELECT \* FROM orders;

-- Retrieve specific order details by order\_id

SELECT order\_details FROM orders WHERE order\_id = 1;

-- Retrieve specific key from HStore column

SELECT order\_details -> 'color' AS color FROM orders WHERE order\_id = 1;

-- Retrieve multiple keys from HStore column

SELECT order\_details #> '{color, quantity}' AS details FROM orders WHERE order\_id = 1;

-- Create the Employee table

CREATE TABLE Employee (

EmpNo INT PRIMARY KEY,

EmpName VARCHAR(50),

Emp\_email VARCHAR(100),

Emp\_grade VARCHAR(50) -- Assuming job titles can be longer

);

-- Insert sample data into the Employee table

INSERT INTO Employee (EmpNo, EmpName, Emp\_email, Emp\_grade) VALUES

(1, 'John Doe', 'john@example.com', 'Manager'),

(2, 'Jane Smith', 'jane@example.com', 'Software Developer'),

(3, 'Michael Johnson', 'michael@example.com', 'Sales Associate'),

(4, 'Emily Brown', 'emily@example.com', 'HR Specialist'),

(5, 'Christopher Lee', 'chris@example.com', 'Marketing Coordinator'),

(6, 'Jessica Taylor', 'jessica@example.com', 'Project Manager'),

(7, 'David Martinez', 'david@example.com', 'Financial Analyst'),

(8, 'Ashley Anderson', 'ashley@example.com', 'Customer Service Representative'),

(9, 'Matthew Wilson', 'matthew@example.com', 'Data Scientist'),

(10, 'Jennifer Garcia', 'jennifer@example.com', 'Operations Manager');

-- Create the Grade table

CREATE TABLE Grade (

Emp\_grade VARCHAR(50) PRIMARY KEY,

Hrlyrate DECIMAL(10, 2) -- Assuming hourly rate is a decimal number

);

-- Insert sample data into the Grade table

INSERT INTO Grade (Emp\_grade, Hrlyrate) VALUES

('Manager', 50.00),

('Software Developer', 40.00),

('Sales Associate', 25.00),

('HR Specialist', 30.00),

('Marketing Coordinator', 28.00),

('Project Manager', 45.00),

('Financial Analyst', 35.00),

('Customer Service Representative', 22.00),

('Data Scientist', 55.00),

('Operations Manager', 48.00);

-- Create the Proj table

CREATE TABLE Proj (

ProjNo INT PRIMARY KEY,

ProjName VARCHAR(100)

);

-- Insert sample data into the Proj table

INSERT INTO Proj (ProjNo, ProjName) VALUES

(101, 'Website Redesign'),

(102, 'Product Launch: Mobile App'),

(103, 'Marketing Campaign: Spring Sale 2024'),

(104, 'Infrastructure Upgrade'),

(105, 'New Product Development: Widget X'),

(106, 'Expansion into European Market'),

(107, 'Customer Relationship Management (CRM) Implementation'),

(108, 'Research and Development: AI Integration'),

(109, 'Supply Chain Optimization Project'),

(110, 'Strategic Partnerships Development');

-- Create the Proj\_Emp table

CREATE TABLE Proj\_Emp (

EmpNo INT,

ProjNo INT,

Hrlyrate DECIMAL(10, 2),

FOREIGN KEY (EmpNo) REFERENCES Employee(EmpNo),

FOREIGN KEY (ProjNo) REFERENCES Proj(ProjNo),

PRIMARY KEY (EmpNo, ProjNo)

);

-- Insert sample data into the Proj\_Emp table

INSERT INTO Proj\_Emp (EmpNo, ProjNo, Hrlyrate)

VALUES

(1, 101, 50.00), -- John Doe working on Website Redesign

(2, 102, 40.00), -- Jane Smith working on Product Launch: Mobile App

(3, 103, 25.00), -- Michael Johnson working on Marketing Campaign: Spring Sale 2024

(4, 104, 30.00), -- Emily Brown working on Infrastructure Upgrade

(5, 105, 28.00), -- Christopher Lee working on New Product Development: Widget X

(6, 106, 45.00), -- Jessica Taylor working on Expansion into European Market

(7, 107, 35.00), -- David Martinez working on CRM Implementation

(8, 108, 22.00), -- Ashley Anderson working on AI Integration

(9, 109, 55.00), -- Matthew Wilson working on Supply Chain Optimization Project

(10, 110, 48.00); -- Jennifer Garcia working on Strategic Partnerships Development

WITH ProjectHourlyRates AS (

SELECT

p.ProjNo,

p.ProjName,

AVG(pe.Hrlyrate) AS AvgHourlyRate

FROM

Proj p

INNER JOIN

Proj\_Emp pe ON p.ProjNo = pe.ProjNo

GROUP BY

p.ProjNo, p.ProjName

),

RankedProjects AS (

SELECT

ProjNo,

ProjName,

AvgHourlyRate,

RANK() OVER (ORDER BY AvgHourlyRate DESC) AS Rank

FROM

ProjectHourlyRates

)

SELECT

rp.ProjNo,

rp.ProjName,

rp.AvgHourlyRate,

rp.Rank,

SUM(rp.AvgHourlyRate) OVER (ORDER BY rp.Rank) AS CumulativeHourlyRate

FROM

RankedProjects rp;

We first use a CTE called ProjectHourlyRates to calculate the average hourly rate per project.

Then, we use another CTE called RankedProjects to rank the projects based on their average hourly rates.

Finally, we select data from the RankedProjects CTE and calculate the cumulative hourly rate for each project by using the SUM window function over the ranked projects.

This query provides insights into the distribution of average hourly rates across projects and ranks them accordingly, while also showing the cumulative hourly rate as projects are ranked. Adjust the query and column names as needed to fit your specific database schema.

-- Create clients table

CREATE TABLE clients (

client\_id INT PRIMARY KEY,

assigned\_salesrep\_id INT,

client\_name VARCHAR(100),

signup\_datetime DATETIME,

churn\_datetime DATETIME,

email VARCHAR(100),

phone\_number VARCHAR(20)

);

-- Create trades table

CREATE TABLE trades (

trade\_id INT PRIMARY KEY,

client\_id INT,

trade\_datetime DATETIME,

facevalue DECIMAL(10, 2),

revenue DECIMAL(10, 2),

product\_id INT

);

-- Create salesreps table

CREATE TABLE salesreps (

salesrep\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

region VARCHAR(50)

);

-- Insert sample data into clients table

INSERT INTO clients (client\_id, assigned\_salesrep\_id, client\_name, signup\_datetime, churn\_datetime, email, phone\_number) VALUES

(1, 101, 'Client A', '2015-06-01', '2015-09-10', 'client\_a@example.com', '123-456-7890'),

(2, 102, 'Client B', '2015-06-15', '2015-08-20', 'client\_b@example.com', '456-789-0123'),

(3, 103, 'Client C', '2015-07-01', NULL, 'client\_c@example.com', '789-012-3456'),

(4, 104, 'Client D', '2015-07-05', '2015-09-25', 'client\_d@example.com', '987-654-3210'),

(5, 105, 'Client E', '2015-07-10', NULL, 'client\_e@example.com', '321-654-0987'),

(6, 106, 'Client F', '2015-08-01', '2015-09-15', 'client\_f@example.com', '654-321-9870'),

(7, 107, 'Client G', '2015-08-05', NULL, 'client\_g@example.com', '012-345-6789'),

(8, 108, 'Client H', '2015-08-10', '2015-09-20', 'client\_h@example.com', '234-567-8901'),

(9, 109, 'Client I', '2015-08-15', '2015-09-25', 'client\_i@example.com', '543-210-9876'),

(10, 110, 'Client J', '2015-08-20', NULL, 'client\_j@example.com', '678-901-2345');

-- Insert sample data into trades table

INSERT INTO trades (trade\_id, client\_id, trade\_datetime, facevalue, revenue, product\_id) VALUES

(1, 1, '2015-07-05 10:00:00', 5000.00, 250.00, 1),

(2, 2, '2015-07-10 11:30:00', 7000.00, 350.00, 2),

(3, 3, '2015-07-15 09:45:00', 3000.00, 150.00, 1),

(4, 4, '2015-08-05 14:20:00', 6000.00, 300.00, 2),

(5, 5, '2015-08-10 16:00:00', 4000.00, 200.00, 1),

(6, 6, '2015-08-15 12:10:00', 5500.00, 275.00, 2),

(7, 7, '2015-09-05 08:45:00', 8000.00, 400.00, 1),

(8, 8, '2015-09-10 10:30:00', 3500.00, 175.00, 2),

(9, 9, '2015-09-15 13:20:00', 4500.00, 225.00, 1),

(10, 10, '2015-09-20 15:00:00', 6500.00, 325.00, 2);

-- Insert sample data into salesreps table

INSERT INTO salesreps (salesrep\_id, first\_name, last\_name, region) VALUES

(101, 'John', 'Doe', 'North'),

(102, 'Jane', 'Smith', 'South'),

(103, 'Michael', 'Johnson', 'East'),

(104, 'Emily', 'Williams', 'West'),

(105, 'David', 'Brown', 'Central'),

(106, 'Sarah', 'Jones', 'North'),

(107, 'Christopher', 'Garcia', 'South'),

(108, 'Amanda', 'Martinez', 'East'),

(109, 'Daniel', 'Robinson', 'West'),

(110, 'Jennifer', 'Lee', 'Central');

SELECT

-- Selecting the sales representative ID, first name, last name, and region

s.salesrep\_id,

s.first\_name,

s.last\_name,

s.region,

-- Counting the number of distinct clients signed up by the sales representative

COUNT(DISTINCT CASE WHEN c.signup\_datetime IS NOT NULL THEN c.client\_id END) AS NumClientSign,

-- Counting the number of distinct clients churned by the sales representative

COUNT(DISTINCT CASE WHEN c.churn\_datetime IS NOT NULL THEN c.client\_id END) AS NumClientChurn,

-- Summing up the face value of trades associated with the clients

SUM(t.facevalue) AS TotalFaceValue,

-- Summing up the revenue generated from trades associated with the clients

SUM(t.revenue) AS TotalRevenue

FROM

-- Selecting from the sales representatives table

salesreps s

-- Performing a left join with the clients table based on the sales representative ID

LEFT JOIN

clients c ON s.salesrep\_id = c.assigned\_salesrep\_id

-- Performing a left join with the trades table based on the client ID

LEFT JOIN

trades t ON c.client\_id = t.client\_id

-- Filtering rows where the trade datetime falls within the specified range

WHERE

t.trade\_datetime BETWEEN '2015-07-01 00:00:00' AND '2015-09-30 23:59:59'

-- Grouping the results by sales representative ID, first name, last name, and region

GROUP BY

s.salesrep\_id,

s.first\_name,

s.last\_name,

s.region;

This query looks at the data to see how well sales representatives are performing. It tells us:

* How many new clients each sales representative has signed up.
* How many clients each sales representative has lost (churned).
* The total value of all the trades made by the clients associated with each sales representative.
* The total amount of money generated from those trades during a certain period.

JS

* Single-Threaded JavaScript Execution: When we say Node.js is “single-threaded”, it means that the JavaScript code you write runs in a single thread. In other words, your JavaScript code can only do one thing at a time. This is because JavaScript was designed to run in web browsers and handle user interactions, which are typically isolated and short-lived events. [So, if you’re running a complex calculation or a task that takes a while to complete, it will block the rest of your code from running1](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[2](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).
* Multi-Threaded libuv Library: While your JavaScript code runs in a single thread, Node.js itself is not single-threaded. [It uses a library called libuv that provides a multi-threaded environment3](https://kadirosmanust.medium.com/is-nodejs-single-threaded-1ab537430626)[4](https://www.digitalocean.com/community/tutorials/how-to-use-multithreading-in-node-js)[libuv is responsible for handling I/O operations (like reading from the network or accessing the file system), which can be slow and would block your JavaScript code if they were run in the same thread1](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[2](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool). [To prevent this, libuv runs these operations in separate threads1](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[2](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).
* Event-Driven Architecture: Node.js uses an event-driven architecture, which means it uses events to trigger functionality. When an I/O operation is started, Node.js registers an event handler (a callback function) and then continues running the rest of your code. [When the I/O operation is finished, it triggers the event, and Node.js runs the corresponding event handler1](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[2](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).

So, in summary, while your JavaScript code runs in a single thread, Node.js uses multiple threads in the background to handle I/O operations. [This design allows Node.js to handle many operations concurrently, without blocking your JavaScript code1](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[2](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).

I hope this explanation helps! If you have any more questions, feel free to ask.

Node.js is often referred to as single-threaded because the JavaScript code you write in Node.js is executed in a single thread. [This means that your JavaScript code can only do one task at a time1](https://kadirosmanust.medium.com/is-nodejs-single-threaded-1ab537430626)[2](https://www.digitalocean.com/community/tutorials/how-to-use-multithreading-in-node-js).

However, Node.js itself is not single-threaded. [It uses the libuv library, which provides a multi-threaded environment1](https://kadirosmanust.medium.com/is-nodejs-single-threaded-1ab537430626)[2](https://www.digitalocean.com/community/tutorials/how-to-use-multithreading-in-node-js)[libuv handles I/O operations like reading files from a disk or network requests1](https://kadirosmanust.medium.com/is-nodejs-single-threaded-1ab537430626)[2](https://www.digitalocean.com/community/tutorials/how-to-use-multithreading-in-node-js). [These operations are performed in separate threads, which are managed by libuv3](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[4](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).

The reason for this design is to provide a way to perform non-blocking I/O operations. [While the I/O operations are being handled by libuv in separate threads, the main thread (where your JavaScript code runs) is free to handle other tasks3](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[4](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).

[This architecture gives you much of the benefit of multi-threading (like performing I/O operations in parallel) without having to deal with the complexities of multi-threaded code in your JavaScript3](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[4](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool). [It’s important to note that CPU-intensive tasks can block the main thread, and for such tasks, other solutions like worker threads or child processes might be more appropriate3](https://stackoverflow.com/questions/17959663/why-is-node-js-single-threaded)[4](https://stackoverflow.com/questions/48241234/why-is-node-js-called-single-threaded-when-it-maintains-threads-in-thread-pool).

Node.js's architecture embodies a nuanced blend of single-threaded JavaScript execution and multi-threaded functionality facilitated by the libuv library. While JavaScript code executes in a single thread, Node.js capitalizes on libuv's multi-threaded capabilities to handle I/O operations concurrently. This dichotomy enables Node.js to remain responsive, even when tasked with lengthy operations, without compromising its single-threaded nature. By leveraging libuv, Node.js deftly manages resource-intensive tasks such as file I/O and network operations in separate threads, thereby preventing blocking and ensuring optimal performance.

The event-driven architecture further enhances Node.js's efficiency by decoupling I/O operations from the main thread. This allows Node.js to register event handlers and continue executing JavaScript code while awaiting the completion of asynchronous tasks. Consequently, Node.js can seamlessly juggle multiple operations concurrently, maximizing throughput and responsiveness. However, it's essential to acknowledge that while Node.js simplifies asynchronous programming, careful consideration must be given to CPU-intensive tasks to avoid potential bottlenecks. In such scenarios, alternative approaches like worker threads or child processes may be warranted to maintain optimal performance and scalability. Thus, Node.js's adept orchestration of single-threaded execution and multi-threaded processing via libuv epitomizes its prowess in facilitating high-performance, non-blocking I/O operations within JavaScript environments.

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>404 Page Not Found</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="glitch">

<span aria-hidden="true">404 Page Not Found</span>

404 Page Not Found

<span aria-hidden="true">404 Page Not Found</span>

</div>

</body>

</html>

body {

background: #222;

min-height: 100vh;

display: grid;

place-items: center;

color: white;

font-family: sans-serif;

}

.glitch {

font-size: 8rem;

font-weight: 700;

text-transform: uppercase;

position: relative;

text-shadow: 0.05em 0 0 rgba(255, 0, 0, 0.75),

-0.025em -0.05em 0 rgba(0, 255, 0, 0.75),

0.025em 0.05em 0 rgba(0, 0, 255, 0.75);

animation: glitch 500ms infinite;

}

.glitch span {

position: absolute;

top: 0;

left: 0;

}

.glitch span:first-child {

animation: glitch 650ms infinite;

clip-path: polygon(0 0, 100% 0, 100% 45%, 0 45%);

transform: translate(-0.025em, -0.0125em);

opacity: 0.8;

}

.glitch span:last-child {

animation: glitch 375ms infinite;

clip-path: polygon(0 80%, 100% 20%, 100% 100%, 0 100%);

transform: translate(0.0125em, 0.025em);

opacity: 0.8;

}

@keyframes glitch {

0%, 14% {

text-shadow:

0.05em 0 0 rgba(255, 0, 0, 0.75),

-0.05em -0.025em 0 rgba(0, 255, 0, 0.75),

-0.025em 0.05em 0 rgba(0, 0, 255, 0.75);

}

15%, 49% {

text-shadow:

-0.05em -0.025em 0 rgba(255, 0, 0, 0.75),

0.025em 0.025em 0 rgba(0, 255, 0, 0.75),

-0.05em -0.05em 0 rgba(0, 0, 255, 0.75);

}

50%, 99% {

text-shadow:

0.025em 0.05em 0 rgba(255, 0, 0, 0.75),

0.05em 0 0 rgba(0, 255, 0, 0.75),

0 -0.05em 0 rgba(0, 0, 255, 0.75);

}

100% {

text-shadow:

-0.025em 0 0 rgba(255, 0, 0, 0.75),

-0.025em -0.025em 0 rgba(0, 255, 0, 0.75),

-0.025em -0.05em 0 rgba(0, 0, 255, 0.75);

}

}

………………………

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Hover Cards with Modal</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background-color: #f0f0f0;

}

.container {

display: flex;

flex-wrap: wrap;

gap: 20px;

}

.card {

position: relative;

width: 200px;

height: 250px;

background-color: #fff;

border-radius: 10px;

overflow: hidden;

transition: transform 0.3s ease;

}

.card:hover {

transform: translateY(-5px);

}

.card img {

width: 100%;

height: auto;

border-top-left-radius: 10px;

border-top-right-radius: 10px;

}

.overlay {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(255, 255, 0, 0.5);

display: flex;

justify-content: center;

align-items: center;

opacity: 0;

transition: opacity 0.3s ease;

border-top-left-radius: 10px;

border-top-right-radius: 10px;

}

.card:hover .overlay {

opacity: 1;

}

.read-more {

color: #333;

font-weight: bold;

text-decoration: none;

padding: 5px 10px;

background-color: #fff;

border: 1px solid #333;

border-radius: 5px;

transition: background-color 0.3s ease;

}

.read-more:hover {

background-color: #333;

color: #fff;

}

/\* Modal Styling \*/

.modal {

display: none;

position: fixed;

z-index: 1;

left: 0;

top: 0;

width: 100%;

height: 100%;

overflow: auto;

background-color: rgba(255, 255, 0, 0.5);

}

.modal-content {

background-color: #fefefe;

margin: 15% auto;

padding: 20px;

border: 1px solid #888;

width: 80%;

border-radius: 10px;

}

.close {

color: #aaa;

float: right;

font-size: 28px;

font-weight: bold;

}

.close:hover,

.close:focus {

color: black;

text-decoration: none;

cursor: pointer;

}

</style>

</head>

<body>

<div class="container">

<div class="card">

<img src="https://via.placeholder.com/200x150" alt="Pic">

<div class="overlay">

<a href="#" class="read-more">Read More</a>

</div>

</div>

<div class="card">

<img src="https://via.placeholder.com/200x150" alt="Pic">

<div class="overlay">

<a href="#" class="read-more">Read More</a>

</div>

</div>

<div class="card">

<img src="https://via.placeholder.com/200x150" alt="Pic">

<div class="overlay">

<a href="#" class="read-more">Read More</a>

</div>

</div>

<div class="card">

<img src="https://via.placeholder.com/200x150" alt="Pic">

<div class="overlay">

<a href="#" class="read-more">Read More</a>

</div>

</div>

</div>

<!-- The Modal -->

<div id="myModal" class="modal">

<!-- Modal content -->

<div class="modal-content">

<span class="close">&times;</span>

<p>This is a full description text.</p>

</div>

</div>

<script>

// Get the modal

var modal = document.getElementById("myModal");

// Get the button that opens the modal

var btns = document.getElementsByClassName("read-more");

// Get the <span> element that closes the modal

var span = document.getElementsByClassName("close")[0];

// When the user clicks on the button, open the modal

for (var i = 0; i < btns.length; i++) {

btns[i].onclick = function() {

modal.style.display = "block";

}

}

// When the user clicks on <span> (x), close the modal

span.onclick = function() {

modal.style.display = "none";

}

// When the user clicks anywhere outside of the modal, close it

window.onclick = function(event) {

if (event.target == modal) {

modal.style.display = "none";

}

}

</script>

</body>

</html>