

[Github URL](#)

[Github URL for Vercel](#)

[Vercel URL](#)

Video: W15-P1: Create tables category2_xx, shop2_xx, user2_xx, cart2_xx, and put 5 products into a cart for the user of your id

=> pgAdmin4, show SQL command to get the needed info

The screenshot shows the pgAdmin4 interface. On the left, the Object Explorer pane displays the schema structure under the 'public' schema, including tables like 'cart2_02', 'category2_02', 'shop2_02', and 'user2_02'. A red box highlights the 'Tables(4)' section. In the center, the SQL tab contains a query:

```
103 (2, 213410102, 10, 3, 0, '2025-12-24 18:50:50'),  
104 (3, 213410102, 15, 1, 0, '2025-12-24 18:50:50'),  
105 (4, 213410102, 23, 2, 0, '2025-12-24 18:50:50'),  
106 (5, 213410102, 30, 2, 0, '2025-12-24 18:50:50'),  
107 (6, 1, 1, 2, 0, '2025-12-24 18:50:50'),  
108 (7, 2, 10, 3, 0, '2025-12-24 18:50:50'),  
109 (8, 3, 15, 1, 0, '2025-12-24 18:50:50'),  
110 (9, 4, 23, 2, 0, '2025-12-24 18:50:50'),  
111 (10, 4, 30, 2, 0, '2025-12-24 18:50:50')  
112  
113 select C.user_id, U.uname as user_name, C.product_id, S.pname as product_name, C.quantity, S.price,  
114 from cart2_02 as C, user2_02 as U, shop2_02 as S  
115 where U.uid = '213410102' and C.user_id = U.uid and C.product_id = S.pid
```

A red box highlights the entire SQL query. Below it, the Data Output tab shows the results of the query:

	user_id	user_name	product_id	product_name	quantity	price	img_url
1	213410102	李國蘋	1	Brown Brim	2	25	/images/midterm/hats/brown-brim.png
2	213410102	李國蘋	10	Black Jean Shearling	3	125	/images/midterm/jackets/black-shearling.j
3	213410102	李國蘋	15	Adidas NMD	1	220	/images/midterm/sneakers/adidas-nmd.n
4	213410102	李國蘋	23	Blue Tanktop	2	25	/images/midterm/womens/blue-tank.pn
5	213410102	李國蘋	30	Camo Down Vest	2	325	/images/midterm/mens/camo-vest.png

Total rows: 5 Query complete 00:00:00.061

=> sql code

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists project files and folders. A red box highlights the 'data' folder, which contains several SQL files: 'midterm', 'blog_02.sql', 'cart2_02.sql', 'category2_02.sql', 'shop2_02.sql', and 'user2_02.sql'. The main editor area displays the contents of the 'cart2_02.sql' file. This file contains SQL code to create a table 'cart2_02' and insert 10 rows of data. The table structure includes columns for cid (int, primary key), user_id (int), product_id (int), quantity (int), total (int, default 0), and added_at (timestamp, default current_timestamp). The inserted data spans from row 1 to row 10, showing various combinations of user IDs, product IDs, and quantities.

```
PGSQL Disconnected
1 CREATE TABLE cart2_02 (
2     cid int NOT NULL PRIMARY KEY,
3     user_id int,
4     product_id int,
5     quantity int,
6     total int DEFAULT 0,
7     added_at timestamp DEFAULT CURRENT_TIMESTAMP
8 );
9
10 INSERT INTO cart2_02 (cid, user_id, product_id, quantity, total, added_at)
11 VALUES
12 (1, 213410102, 1, 2, 0, '2025-12-24 18:50:50'),
13 (2, 213410102, 10, 3, 0, '2025-12-24 18:50:50'),
14 (3, 213410102, 15, 1, 0, '2025-12-24 18:50:50'),
15 (4, 213410102, 23, 2, 0, '2025-12-24 18:50:50'),
16 (5, 213410102, 30, 2, 0, '2025-12-24 18:50:50'),
17 (6, 1, 1, 2, 0, '2025-12-24 18:50:50'),
18 (7, 2, 10, 3, 0, '2025-12-24 18:50:50'),
19 (8, 3, 15, 1, 0, '2025-12-24 18:50:50'),
20 (9, 4, 23, 2, 0, '2025-12-24 18:50:50'),
21 (10, 4, 30, 2, 0, '2025-12-24 18:50:50')
22
23 select C.user_id, U.uname as user_name, C.product_id, S.pname as product_name, C.quantity, S.price, S.img_url
24 from cart2_02 as C, user2_02 as U, shop2_02 as S
25 where U.uid = '213410102' and C.user_id = U.uid and C.product_id = S.pid
```

f43d4bd apple550678

2025-12-28 15:20:35 +0800

Video: W15-P1: Create tables category2_>

Video: W15-P2: Implement route /api/shop_xx/cart/:uid to get the info as the SQL in W15-P1

The screenshot shows a developer's environment with several windows open:

- Code Editor:** Shows `server_02.js` with a red box highlighting the route definition: `app.use('/api/shop2_02/cart/:uid', async (req, res, next) => {`. Below it, a query is shown: `const results = await db.query(`select C.user_id, U.username as user_name, C.product_id, S.product_name, C.quantity, S.price, S.img_url from cart2_02 as C, user2_02 as U, shop2_02 as S where U.uid = $1 and C.user_id = U.uid and C.product_id = S.pid`, [req.params.uid])`.
- Terminal:** Shows the command `npm run dev` being run, outputting: "Connecting local database wp1_demo_02 Server running on port 5001 uid 213410102".
- Browser:** Shows a request to `calhost:500 /api/shop2_02/cart/213410102` with a red box around the JSON response:

```
[{"user_id": 213410102, "user_name": "李國蘋", "product_id": 1, "product_name": "Brown Brim", "quantity": 2, "price": 25, "img_url": "/images/midterm/hats/brown-brim.png"}, {"user_id": 213410102, "user_name": "李國蘋", "product_id": 30, "product_name": "Camo Down Vest", "quantity": 2, "price": 325, "img_url": "/images/midterm/mens/camo-vest.png"}]
```
- Status Bar:** Shows the commit hash `3286bd9 apple550678`, the date `2025-12-28 15:48:13 +0800`, and the video title `Video: W15-P2: Implement route /api/shop_xx/cart/:uid to get the info as the SQL in W15-P1`.

3286bd9 apple550678

2025-12-28 15:48:13 +0800

Video: W15-P2: Implement route /api/shop_xx/cart/:uid to get the info as the SQL in W15-P1

Video: W15-P3: Use localStorage

=> Implement setLocalStorage

The screenshot shows a development environment with several windows open:

- Code Editor:** An IDE window displaying `page.jsx` with code related to `localStorage`. A red box highlights the line `localStorage.setItem('list', JSON.stringify(items))`.
- Browser:** A browser window showing a simple grocery list application titled "Grocery Bud". It lists "Apple" and "Eggs" with "Delete" buttons.
- Storage Viewer:** A sidebar titled "儲存空間" (Storage Space) showing the contents of `localStorage`. A red box highlights the entry for "list":

```
list [{"name": "Apple", "completed": false}, {"name": "Apple", "completed": false, "id": "4ZD4uhMODU7j-o2MUuj", "name": "Apple"}, {"name": "Eggs", "completed": false, "id": "Im_zqlf0e_7iDAN1bAlYY", "name": "Eggs"}]
```

=> Implement getLocalStorage

The image shows a developer's workspace with two main components:

- VS Code Editor:** On the left, the file `page.jsx` is open. Two specific sections of the code are highlighted with red boxes:
 - The first box highlights the `getLocalStorage` function, which reads from `window.localStorage.getItem('list')` and parses it as JSON if it exists.
 - The second box highlights the `GroceryPage_02` component definition, which uses `useState` to manage the list of items and calls `getLocalStorage` in its effect hook to initialize the state.
- Browser Screenshot:** On the right, a browser window displays a grocery list application titled "Grocery Bud". It shows two items: "Apple" and "Eggs", each with a checkbox and a "Delete" button. A red box highlights this list area.

Storage Inspector: A sidebar on the right shows the browser's storage state. It lists the key `list` with the value `[{"name": "Apple", ...}, {"name": "Eggs", ...}]`. This value corresponds to the list displayed in the browser.

```
const getLocalStorage = () => {
  if (typeof window !== 'undefined') {
    let list = localStorage.getItem('list')
    if (list) {
      list = JSON.parse(list)
    } else {
      list = []
    }
    return list
  }
  return [] // Return default for server-side
}

const setLocalStorage = (items) => {
  localStorage.setItem('list', JSON.stringify(
    items
  ))
}

const GroceryPage_02 = () => [
  const [items, setItems] = useState([])
  useEffect(() => {
    const storedList = getLocalStorage()
    if (storedList.length > 0) {
      setItems(storedList)
    }
  }, [])
]
```

59bba84 apple550678

2025-12-28 16:44:18 +0800

Video: W15-P3: Use localStorage