GYM WEBSITE

Github Repo And Members

Repo

https://github.com/Hir0Yat0/database-project-2024

Members

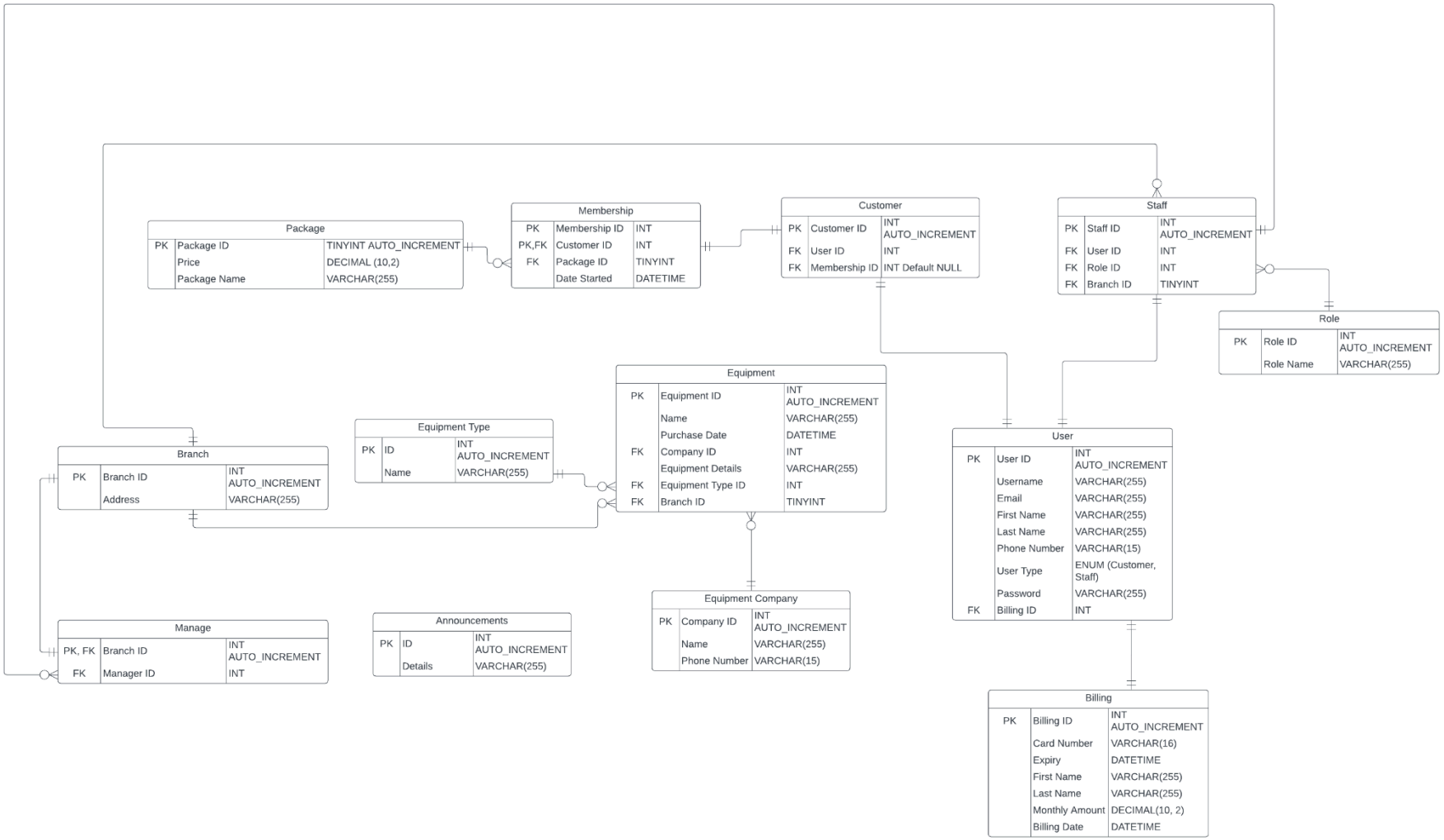
anishkhanijaur: Anish Khanijaur 6480492
Hir0Yat0: Ratchaphong Nittaya 6481285

.
.
.
.

ER Diagram

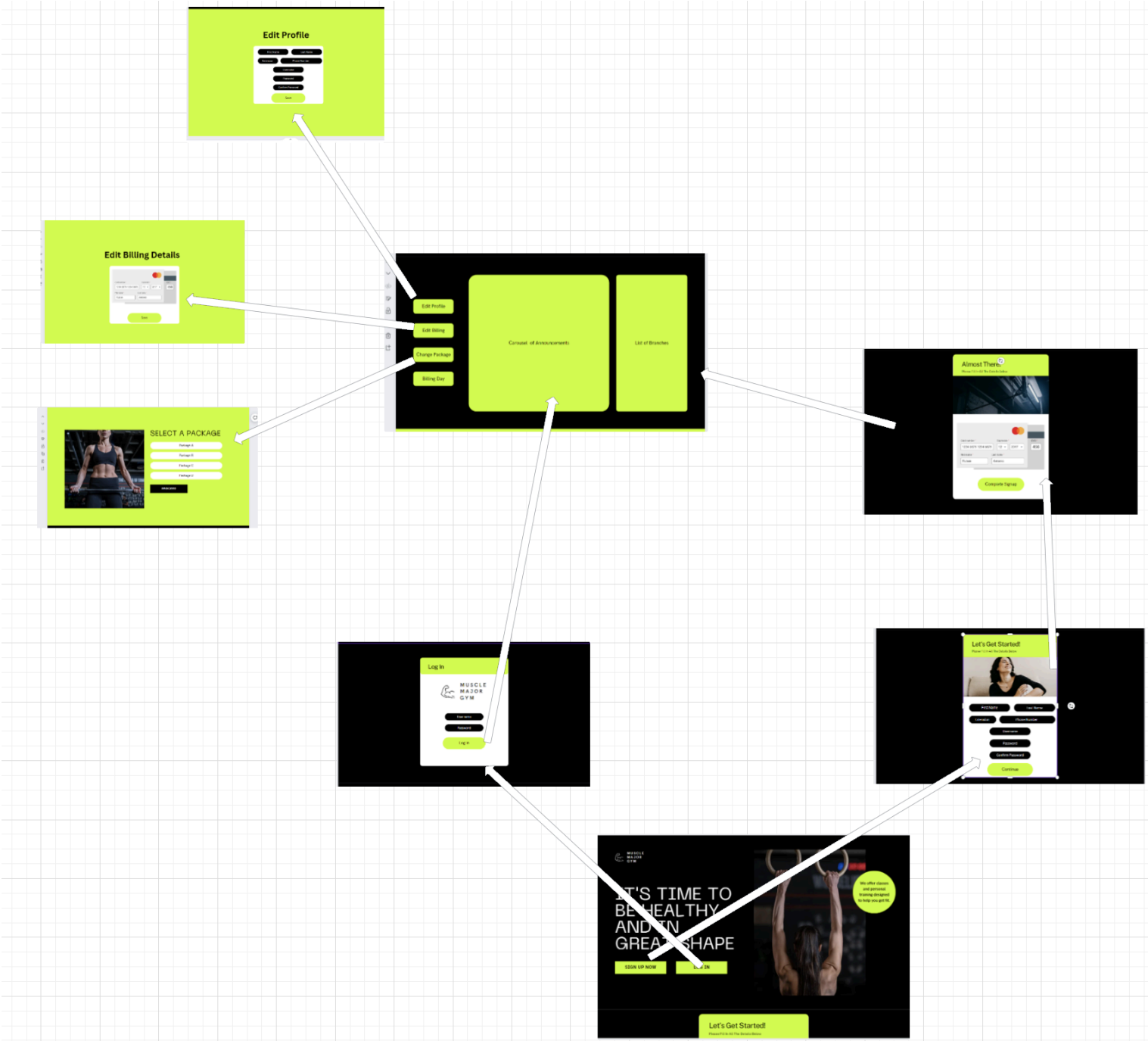
(Please zoom out)

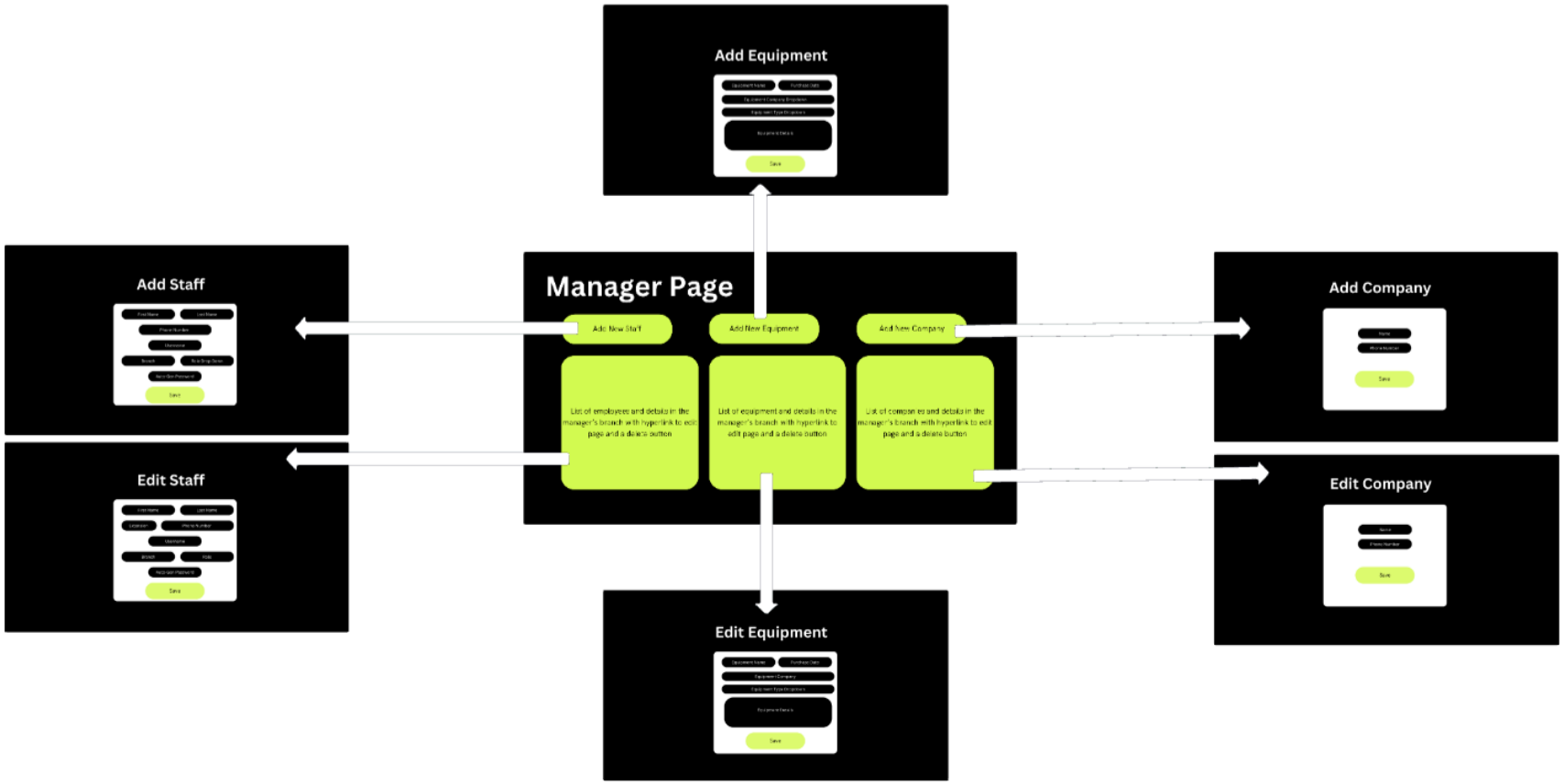
https://lucid.app/lucidchart/fe4850d4-4ee7-49d0-a70b-a031ce96a702/edit?viewport_loc=-2279%2C11559%2C2368%2C1330%2C0_0&invitationId=inv_5df0cd50-8dac-4a0b-a151-183547adacbb

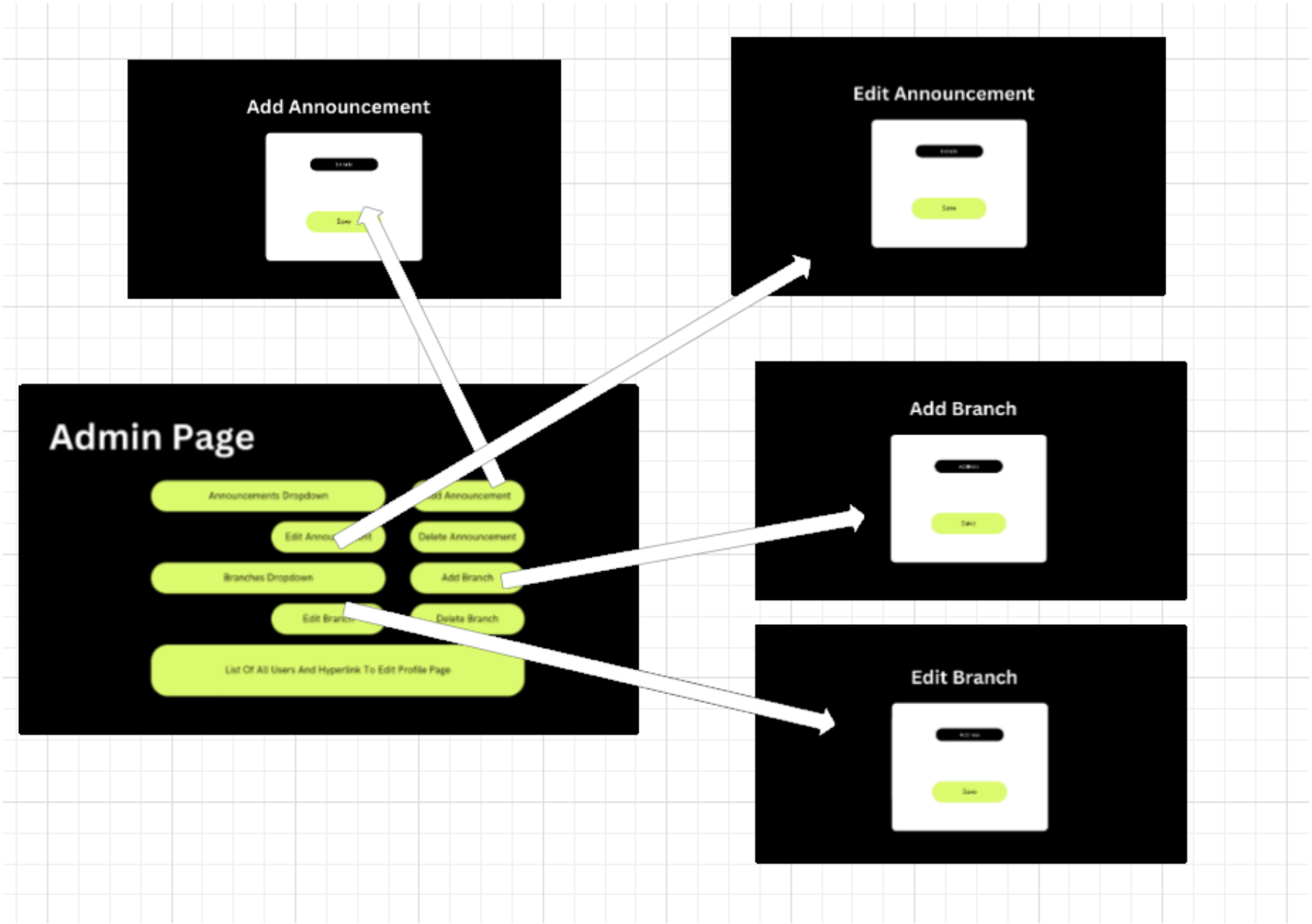


Sitemap

Sitemap:
https://lucid.app/lucidchart/324392d1-338e-46fa-ba12-3c5c9b1de77f/edit?viewport_loc=-5169%2C-4946%2C1707%2C800%2C0_0&invitationId=inv_b564fe77-956e-4972-bec2-f1bdea065d73
Full Website Design:
https://www.canva.com/design/DAGAnsfzVFU/vxi-XDDk9DiId0Pio4XN6w/edit?utm_content=DAGAnsfzVFU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton


Edit Profile
Edit Billing Details
Edit Profile
Edit Billing
Change Package
Billing Day
Carousel of Announcements
List of Branches
SELECT A PACKAGE
Almost There
Complete Signup
Log In
MUSCLE MAJOR GYM
Let's Get Started!
Continue
IT'S TIME TO BE HEALTHY AND IN GREAT SHAPE
SIGN UP NOW
Let's Get Started!

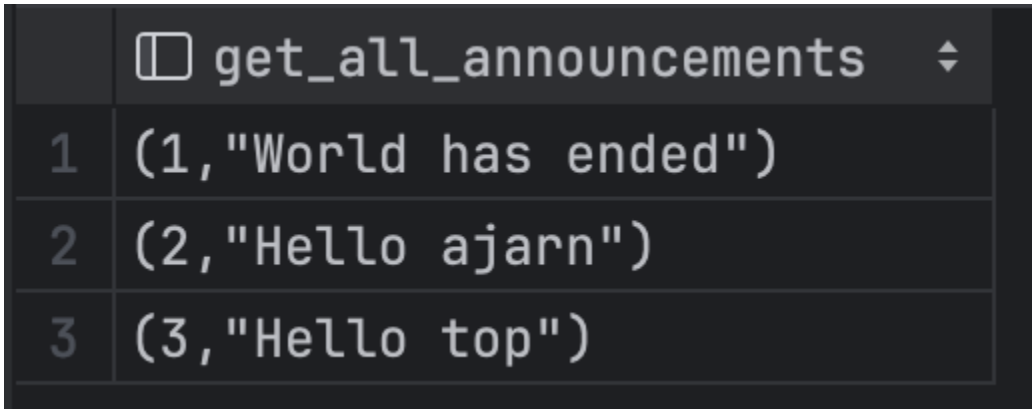

Add Equipment
Add Staff
Manager Page
Add Company
Add New Staff
Add New Equipment
Add New Company
List of employees and details in the manager's branch with hyperlink to edit page and a delete button
List of equipment and details in the manager's branch with hyperlink to edit page and a delete button
List of companies and details in the manager's branch with hyperlink to edit page and a delete button
Edit Staff
Edit Company
Edit Equipment


Add Announcement
Edit Announcement
Admin Page
Announcements Dropdown
Delete Announcement
Branches Dropdown
Add Branch
Delete Branch
List Of All Users And Hyperlink To Edit Profile Page
Add Branch
Edit Branch

Mock UI

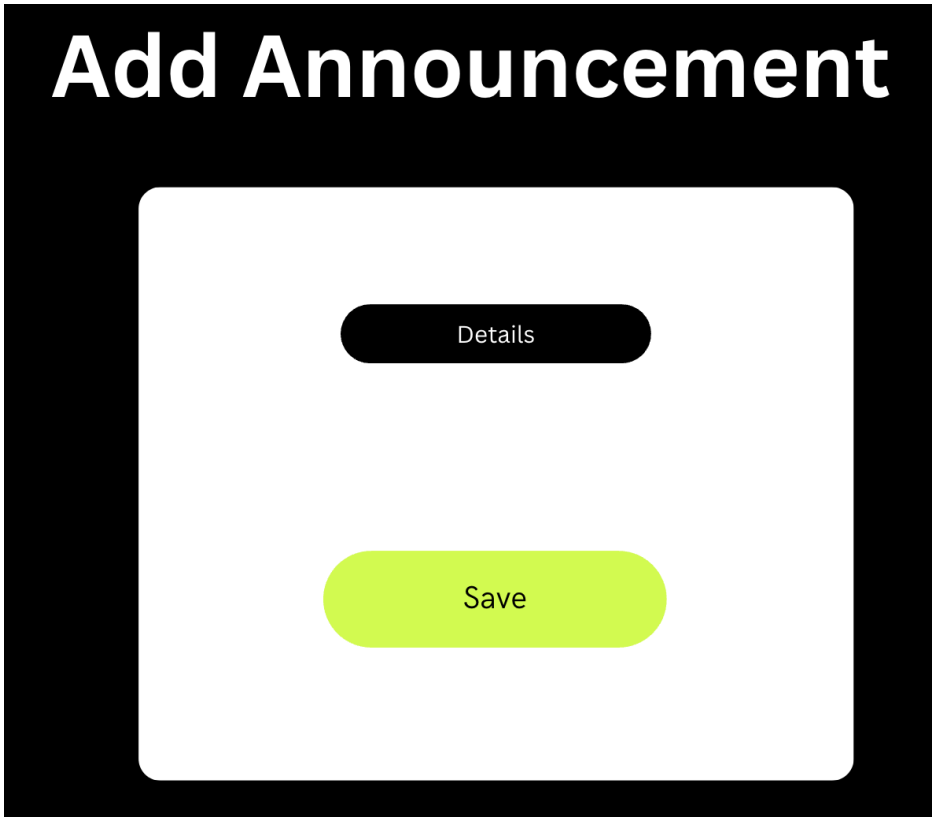

Admin Page
Announcements Dropdown
Add Announcement
Edit Announcement
Delete Announcement
Branches Dropdown
Add Branch
Edit Branch
Delete Branch
List Of All Users And Hyperlink To Edit Profile Page

1. Get all announcements dropdown:



	get_all_announcements
1	(1,"World has ended")
2	(2,"Hello ajarn")
3	(3,"Hello top")

2. Add announcement:



```sql
CREATE OR REPLACE PROCEDURE add_announcement(IN details VARCHAR(255), INOUT status_code INT) AS
$$
BEGIN
    INSERT INTO public."Announcements"("Details") VALUES (details);
    COMMIT;
    status_code := 0; -- Assigning 0 to status_code
END;
$$ LANGUAGE plpgsql;

CALL add_announcement( details: 'New announcement', status_code: 0);
```

Before (Announcement table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Hello top

After (Announcement table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Hello top
4	5	New announcement

3. Delete Announcement:

```sql
CREATE OR REPLACE PROCEDURE delete_announcement(
    IN announcement_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    DELETE FROM public."Announcements"
    WHERE "ID" = announcement_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL delete_announcement( announcement_id: 5,  status_code: 0);
```

Before (Announcement Table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Hello top
4	5	New announcement

After (Announcement Table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Hello top

4. Edit Announcement:

```sql
CREATE OR REPLACE PROCEDURE edit_announcement(
    IN announcement_id INT,
    IN details VARCHAR(255),
    INOUT status_code INT
) AS
$$
BEGIN
    UPDATE public."Announcements"
    SET "Details" = details
    WHERE "ID" = announcement_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL edit_announcement( announcement_id: 3, details: 'Updated announcement', status_code: 0);
```

Before (Announcement Table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Hello top

After (Announcement Table):

	ID	Details
1	1	World has ended
2	2	Hello ajarn
3	3	Updated announceme…

5. Branches dropdown to get all branches:

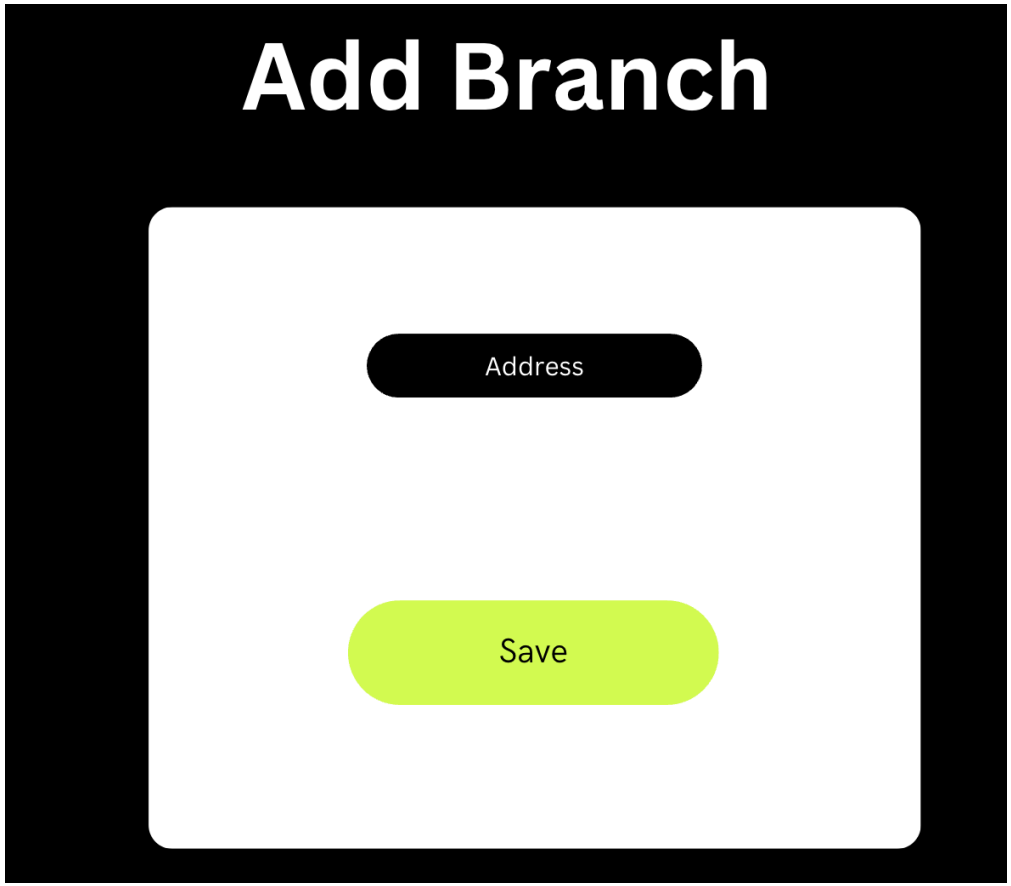
```
CREATE OR REPLACE FUNCTION getAllBranches()
RETURNS SETOF "Branch" AS
$$
    SELECT * FROM "Branch";
$$ LANGUAGE SQL;

SELECT getAllBranches();
```

Output:

Returns ID for future changes such as editing or deleting and other details

	getallbranches
1	(2,"456 Elm St")
2	(1,"1234 Main St")

6. Add branch:



```sql
CREATE OR REPLACE PROCEDURE add_branch(
    IN address VARCHAR(255),
    INOUT status_code INT
) AS
$$
BEGIN
    INSERT INTO public."Branch"("Address") VALUES (address);
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL add_branch( address: 'Test Address', status_code: 0);
```

Before (Branch table):

	Branch ID	Address
1	2	456 Elm St
2	1	1234 Main St

After (Branch table):

	Branch ID	Address
1	2	456 Elm St
2	1	1234 Main St
3	4	Test Address

7. Delete Branch:

```sql
CREATE OR REPLACE PROCEDURE delete_branch(
    IN branch_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    -- Delete from Equipment Type table
    DELETE FROM public."Branch"
    WHERE "Branch ID" = branch_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL delete_branch( branch_id: 4,  status_code: 0);
```

Before (Branch Table):

	Branch ID	Address
1	2	456 Elm St
2	1	1234 Main St
3	4	Test Address

After (Branch Table):

	Branch ID	Address
1	2	456 Elm St
2	1	1234 Main St

8. Edit Branch:

```sql
CREATE OR REPLACE PROCEDURE edit_branch(
    IN branch_id INT,
    IN address VARCHAR(255),
    INOUT status_code INT
) AS
$$
BEGIN
    -- Update Branch table
    UPDATE public."Branch"
    SET "Address" = address
    WHERE "Branch ID" = branch_id;
    COMMIT;

    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL edit_branch( branch_id: 1,  address: '123 Main St',  status_code: 0);
```

Before (Branch Table):

Branch ID	Address
2	456 Elm St
1	1234 Main St

After (Branch Table):

Branch ID	Address
2	456 Elm St
1	123 Main St

1. Get all employees dropdown:

```sql
CREATE OR REPLACE FUNCTION getAllEmployeesForBranch("branch id" INT)
RETURNS TABLE("Staff ID" int, "User ID" int, "Role ID" int,
              "Branch ID" int, "Username" varchar, "Email" varchar,
              "First Name" varchar, "Last Name" varchar, "Phone Number" varchar,
              "User Type" varchar, Password varchar, "Billing ID" int)
AS
$$
BEGIN
    RETURN QUERY
    SELECT a."Staff ID", b."User ID", a."Role ID", a."Branch ID",
           b."Username", b."Email", b."First Name", b."Last Name",
           b."Phone Number", b."User Type", b."Password", b."Billing ID"
    FROM public."Staff" a
    INNER JOIN public."User" b ON a."User ID" = b."User ID"
    WHERE a."Branch ID" = "branch id";
END;
$$ LANGUAGE plpgsql;


SELECT getAllEmployeesForBranch( branch id: 1);
```

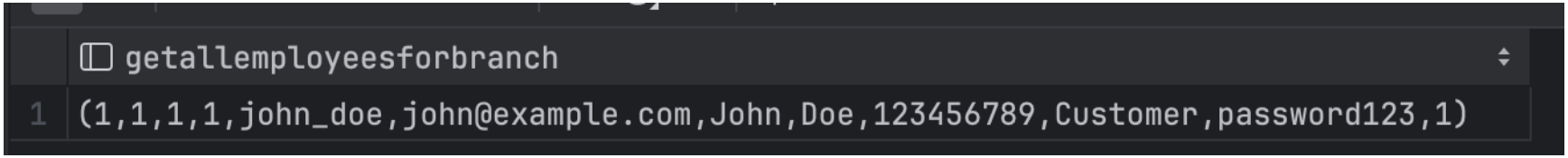
Staff Table:

	Staff ID	User ID	Role ID	Branch ID
1	1	1	1	1
2	2	2	2	2

User Table:

	User ID	Username	Email	First Name	Last Name	Phone Number	User Type	Password	Billing ID
1	1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2

Output:



2. Get all equipment dropdown:

```
CREATE OR REPLACE FUNCTION getAllEquipmentForBranch("branch id" INT)
RETURNS TABLE("Equipment ID" int, "Equipment Name" varchar, "Purchase Date" timestamp,
              "Equipment Details" varchar, "Equipment Type Name" varchar, "Company Name" varchar,
              "Company Phone Number" varchar)
AS
$$
BEGIN
    RETURN QUERY
    SELECT c."ID", c."Equipment Name", c."Purchase Date", c."Equipment Details", c."Equipment Type Name",
           d."Name" as "Company Name", "Phone Number"
    FROM (
        SELECT *, b."Name" as "Equipment Type Name", a."Name" as "Equipment Name"
        FROM public."Equipment" a
        INNER JOIN public."Equipment Type" b ON a."Equipment Type ID" = b."ID"
         ) c
    INNER JOIN "Equipment Company" d ON c."Company ID" = d."Company ID"
    WHERE c."Branch ID" = "branch id";
END;
$$ LANGUAGE plpgsql;

SELECT getAllEquipmentForBranch( branch id: 1);
```

Equipment Table:

Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2

Equipment Company Table:

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210

Output:

	getallequipmentforbranch
1	(1,"Treadmill 1","2023-01-01 00:00:00","High-quality treadmill",Treadmill,"Company A",123-456-7890)
2	(2,"Dumbbell Set 1","2023-02-01 00:00:00","Set of dumbbells",Dumbbell,"Company B",987-654-3210)

3. Get all companies dropdown:

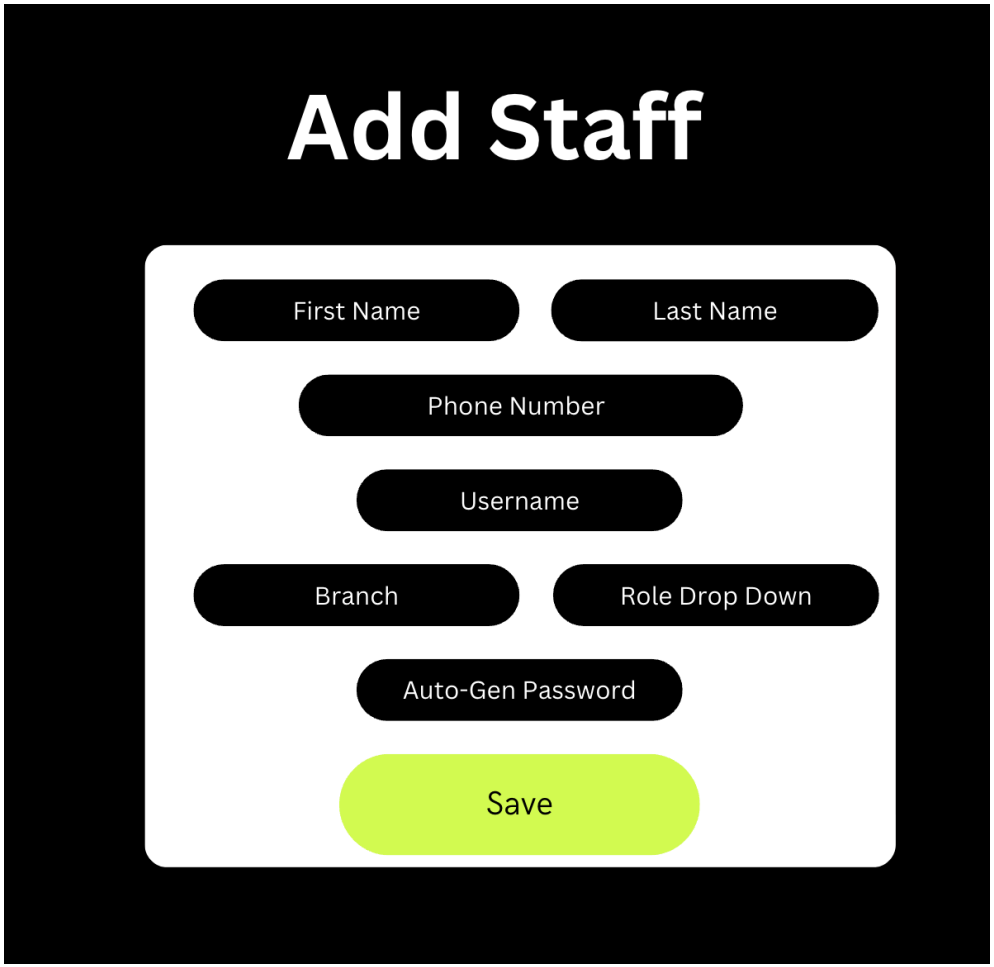
```sql
CREATE OR REPLACE FUNCTION getAllCompaniesForBranch("branch id" INT)
RETURNS TABLE("Company Name" varchar, "Phone Number" varchar)
AS
$$
BEGIN
    RETURN QUERY
    SELECT d."Name" as "Company Name", d."Phone Number"
    FROM (
        SELECT *, b."Name" as "Equipment Type Name", a."Name" as "Equipment Name"
        FROM public."Equipment" a
        INNER JOIN public."Equipment Type" b ON a."Equipment Type ID" = b."ID"
         ) c
    INNER JOIN "Equipment Company" d ON c."Company ID" = d."Company ID"
    WHERE c."Branch ID" = "branch id";
END;
$$ LANGUAGE plpgsql;

SELECT getAllCompaniesForBranch( branch id: 1);
```

Equipment Company Table:

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210

Output:

	getallcompaniesforbranch
1	("Company A",123-456-7890)
2	("Company B",987-654-3210)



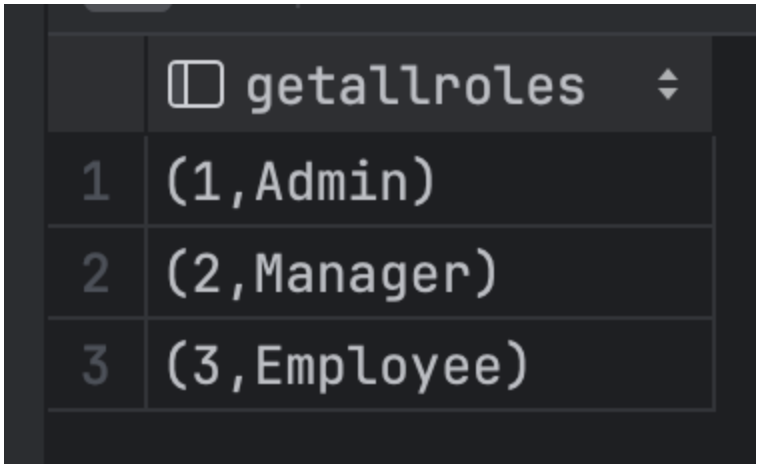
1. Role Drop Down:

```sql
CREATE OR REPLACE FUNCTION getAllRoles()
RETURNS TABLE("Role ID" int, "Role Name" varchar)
AS
$$
BEGIN
    RETURN QUERY
    SELECT * FROM public."Role";
END;
$$ LANGUAGE plpgsql;

SELECT getAllRoles();
```

Role Table:

	Role ID	Role Name
1	1	Admin
2	2	Manager
3	3	Employee

Output:



	getallroles
1	(1,Admin)
2	(2,Manager)
3	(3,Employee)

2. Adding Staff:

```sql
CREATE OR REPLACE PROCEDURE add_staff(IN username VARCHAR(255), IN email VARCHAR(255),
                                      IN first_name VARCHAR(255), IN last_name VARCHAR(255), IN phone_number VARCHAR(15),
                                      password IN VARCHAR(255), IN role_id INT, IN branch_id INT, INOUT status_code INT) AS
$$
DECLARE
    temp_user_id INT; -- Temporary variable to hold the new user id
BEGIN
    INSERT INTO public."User"("Username", "Email", "First Name", "Last Name", "Phone Number", "User Type",
                              "Password", "Billing ID") VALUES (username, email, first_name, last_name, phone_number,
                                                                'Staff', password, NULL)
    RETURNING "User ID" INTO temp_user_id;
    INSERT INTO public."Staff"("User ID", "Role ID", "Branch ID") VALUES (temp_user_id, role_id, branch_id);
    COMMIT;

    status_code := 0; -- Assigning 0 to status_code
END;
$$ LANGUAGE plpgsql;

CALL add_staff( username: 'anish', email: 'anish@anish.anish', first_name: 'anish', last_name: 'anish',
                phone_number: '000-000-0000', password: 'hardpassword', role_id: 1, branch_id: 1, status_code: 0);
```

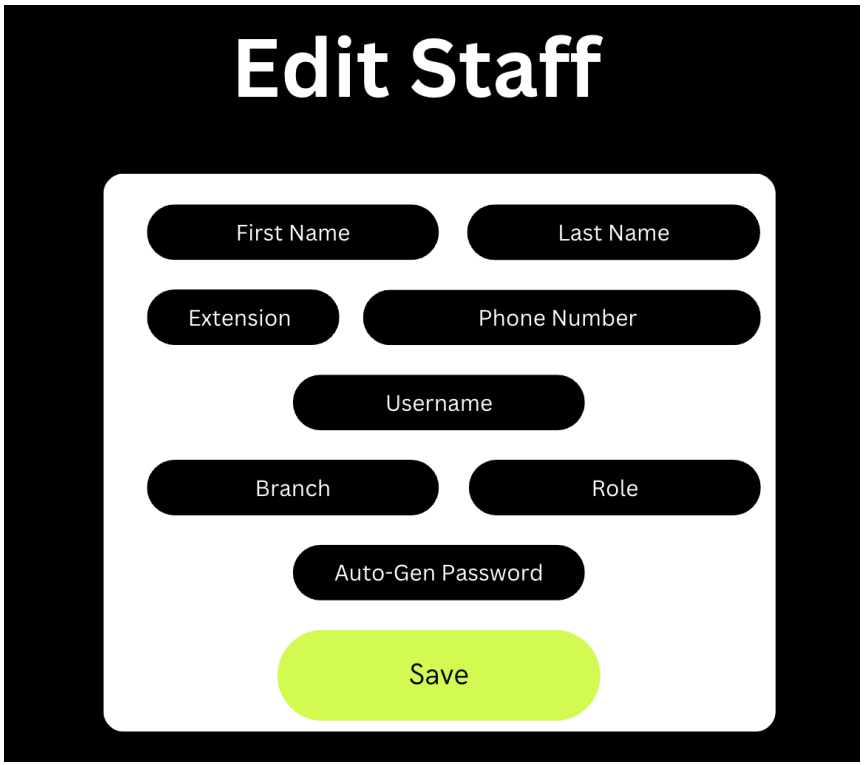
Before (Staff Table):

	Staff ID	User ID	Role ID	Branch ID
1	1	1	1	1
2	2	2	2	2

After (Staff Table):

	Staff ID	User ID	Role ID	Branch ID
1	1	1	1	1
2	2	2	2	2
3	7	5	1	1

3. Editing Staff



Using code block because the procedure is too long to fit in a page clearly:

Unset

```
CREATE OR REPLACE PROCEDURE edit_staff(
   IN user_id INT,
   IN staff_id INT,
   IN username VARCHAR(255),
   IN email VARCHAR(255),
   IN first_name VARCHAR(255),
   IN last_name VARCHAR(255),
   IN phone_number VARCHAR(15),
   IN password VARCHAR(255),
   IN role_id INT,
   IN branch_id INT,
   INOUT status_code INT
) AS
$$
BEGIN
   -- Update user details
   UPDATE public."User"
   SET
       "Username" = username,
       "Email" = email,
       "First Name" = first_name,
       "Last Name" = last_name,
       "Phone Number" = phone_number,
       "Password" = password
   WHERE
       "User ID" = user_id;
```

```
    -- Update staff details
    UPDATE public."Staff"
    SET
        "Role ID" = role_id,
        "Branch ID" = branch_id
    WHERE
        "Staff ID" = staff_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL edit_staff(7, 5, 'Anish again', 'anish@anish.anish', 'anish', 'anish', '000-000-0000',
'hardpassword', 1, 1, 0);
```

Before (Staff Table):

	Staff ID	User ID	Role ID	Branch ID
1	1	1	1	1
2	2	2	2	2
3	7	5	1	1

Before (User Table):

	User ID	Username	Email	First Name	Last Name	Phone Number	User Type	Password	Billing ID
1	1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2
3	4	user1	user1@gmail.com	user1firstname	user1lastname	0123456789	Customer	$pass1:hashed:salted^$	1
4	5	anish	anish@anish.ani…	anish	anish	000-000-0000	Staff	hardpassword	<null>

After (Staff Table):

	Staff ID	User ID	Role ID	Branch ID
1	1	1	1	1
2	2	2	2	2
3	7	5	1	1

After (User Table):

	User ID	Username	Email	First Name	Last Name	Phone Number	User Type	Password	Billing ID
1	1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2
3	4	user1	user1@gmail.com	user1firstname	user1lastname	0123456789	Customer	$pass1:hashed:salted^$	1
4	5	Anish again	anish@anish.ani…	anish	anish	000-000-0000	Staff	hardpassword	<null>

4. Deleting Staff: (We did add cascade rule later on to make things more smooth)

```
CREATE OR REPLACE PROCEDURE delete_staff(
    IN user_id INT,
    IN staff_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    -- Delete from Staff table
    DELETE FROM public."Staff"
    WHERE "Staff ID" = staff_id;

    DELETE FROM public."User"
    WHERE "User ID" = user_id;

    -- Set status_code to 0 indicating success
    status_code := 0;
    COMMIT;
END;
$$ LANGUAGE plpgsql;

CALL delete_staff( user_id: 5,  staff_id: 7,  status_code: 0)
```

Before (Staff Table):

Staff ID	User ID	Role ID	Branch ID
1	1	1	1
2	2	2	2
7	5	1	1

Before (User Table):

User ID	Username	Email	First Name	Last Name	Phone Number	User Type	Password	Billing ID
1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2
4	user1	user1@gmail.com	user1firstname	user1lastname	0123456789	Customer	$pass1:hashed:salted^$	1
5	Anish again	anish@anish.ani…	anish	anish	000-000-0000	Staff	hardpassword	<null>

After (Staff Table):

Staff ID	User ID	Role ID	Branch ID
1	1	1	1
2	2	2	2

After (User Table):

User ID	Username	Email	First Name	Last Name	Phone Number	User Type	Password	Billing ID
1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2
4	user1	user1@gmail.com	user1firstname	user1lastname	0123456789	Customer	$pass1:hashed:salted^$	1

Add Equipment

Equipment Name

Purchase Date

Equipment Company Dropdown

Equipment Type Dropdown

Equipment Details

Save

1. Equipment Company Dropdown:

```
CREATE OR REPLACE FUNCTION get_all_companies()
RETURNS SETOF "Equipment Company" AS
$$
    SELECT * FROM "Equipment Company";
$$ LANGUAGE SQL;

SELECT get_all_companies();
```

Equipment Company table:

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210

Output:

	get_all_companies
1	(1,"Company A",123-456-7890)
2	(2,"Company B",987-654-3210)

2. Equipment Type Dropdown

```
CREATE OR REPLACE FUNCTION get_all_equipment_types()
RETURNS SETOF "Equipment Type" AS
$$
    SELECT * FROM "Equipment Type";
$$ LANGUAGE SQL;

SELECT get_all_equipment_types();
```

Equipment Type Table:

ID	Name
1	Treadmill
2	Dumbbell
3	Stationary Bike

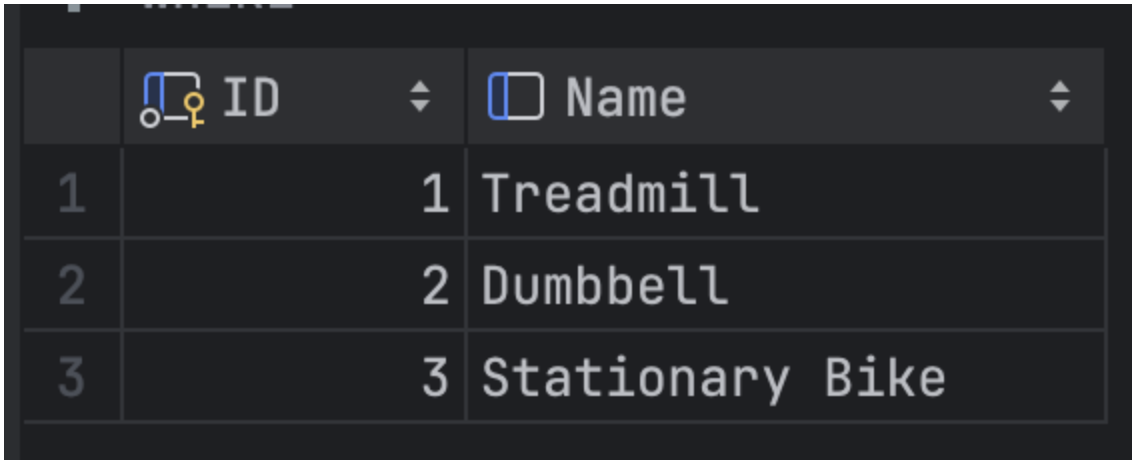
Output:

	get_all_equipment_types
1	(1,Treadmill)
2	(2,Dumbbell)
3	(3,"Stationary Bike")

3. Add equipment type:

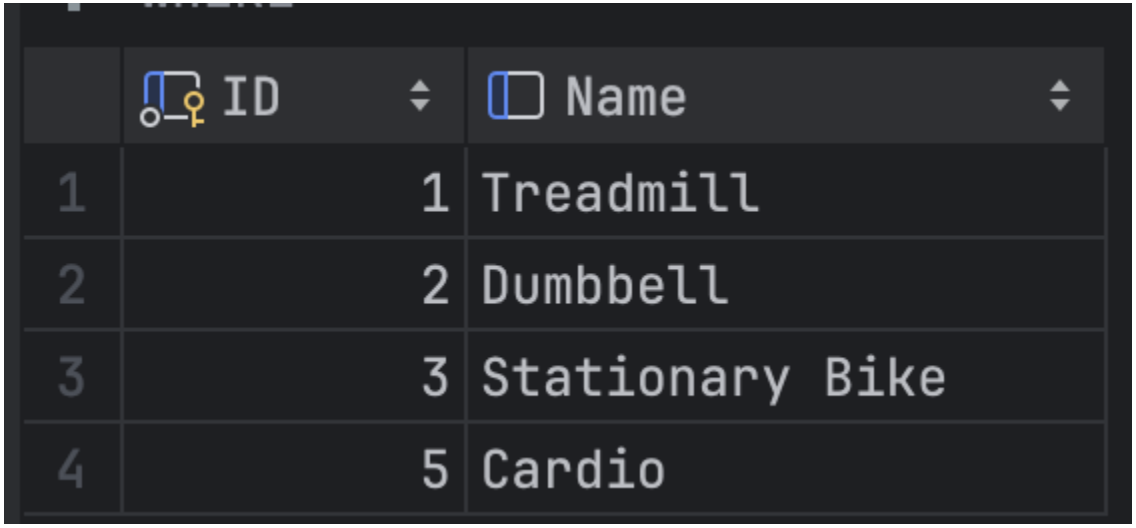
```sql
CREATE OR REPLACE PROCEDURE add_equipment_type(
    IN name VARCHAR(255),
    INOUT status_code INT
) AS
$$
BEGIN
    INSERT INTO public."Equipment Type"("Name") VALUES (name);
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL add_equipment_type( name: 'Cardio',  status_code: 0);
```

Before (Equipment Type Table):



	ID	Name
1	1	Treadmill
2	2	Dumbbell
3	3	Stationary Bike

After (Equipment Type Table):



	ID	Name
1	1	Treadmill
2	2	Dumbbell
3	3	Stationary Bike
4	5	Cardio

4. Deleting equipment type:

```sql
CREATE OR REPLACE PROCEDURE delete_equipment_type(
    IN type_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    -- Delete from Equipment Type table
    DELETE FROM public."Equipment Type"
    WHERE "ID" = type_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL delete_equipment_type( type_id: 5,  status_code: 0);
```

Before (Equipment Type Table):

	ID	Name
1	1	Treadmill
2	2	Dumbbell
3	3	Stationary Bike
4	5	Cardio

After (Equipment Type Table):

	ID	Name
1	1	Treadmill
2	2	Dumbbell
3	3	Stationary Bike

5. Adding the equipment:

```sql
CREATE OR REPLACE PROCEDURE add_equipment(
    IN equipment_name VARCHAR(255),
    IN purchase_date TIMESTAMP,
    IN company_id INT,
    IN equipment_details VARCHAR(255),
    IN equipment_type_id INT,
    IN branch_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    INSERT INTO public."Equipment"("Name", "Purchase Date", "Company ID", "Equipment Details",
                                   "Equipment Type ID", "Branch ID") VALUES (equipment_name, purchase_date, company_id,
                                                                             equipment_details, equipment_type_id,
                                                                             branch_id);
    -- Set status_code to 0 indicating success
    COMMIT;
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL add_equipment( equipment_name: 'Treadmill 2', purchase_date: '2023-01-01 00:00:00.000000', company_id: 1,
                    equipment_details: 'High-quality treadmill',
                    equipment_type_id: 1, branch_id: 1,  status_code: 0);
```

Before (Equipment Table):

Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2

After (Equipment Table):

Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2
5	Treadmill 2	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1

6. Editing the equipment:

```
CREATE OR REPLACE PROCEDURE edit_equipment(
    IN equipment_id INT,
    IN equipment_name VARCHAR(255),
    IN purchase_date TIMESTAMP,
    IN company_id INT,
    IN equipment_details VARCHAR(255),
    IN equipment_type_id INT,
    IN branch_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    UPDATE public."Equipment"
    SET "Name" = equipment_name,
        "Purchase Date" = purchase_date,
        "Company ID" = company_id,
        "Equipment Details" = equipment_details,
        "Equipment Type ID" = equipment_type_id,
        "Branch ID" = branch_id
    WHERE "Equipment ID" = equipment_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL edit_equipment( equipment_id: 5, equipment_name: 'Treadmill 2', purchase_date: '2023-01-01 00:00:00.000000', company_id: 1,
                     equipment_details: 'Highest-quality treadmill',
                     equipment_type_id: 1, branch_id: 1, status_code: 0);
```

Before (Equipment Table):

	Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2
4	5	Treadmill 2	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1

After (Equipment Table):

	Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2
4	5	Treadmill 2	2023-01-01 00:00:00.000000	1	Highest-quality treadmi…	1	1

7. Deleting the equipment:

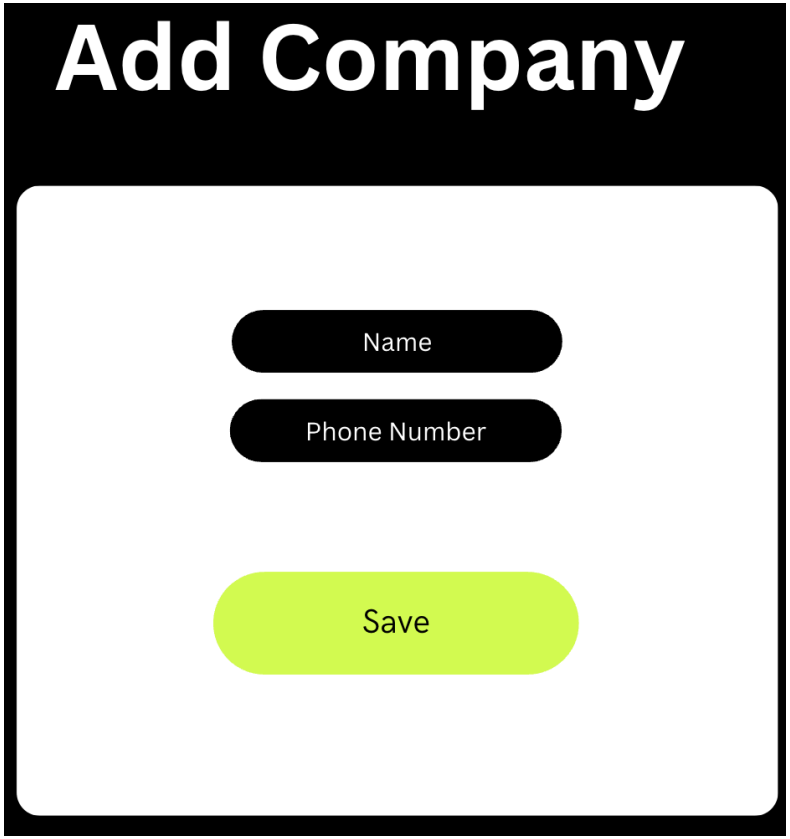
```
CREATE OR REPLACE PROCEDURE delete_equipment(
    IN equipment_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    DELETE FROM public."Equipment"
    WHERE "Equipment ID" = equipment_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL delete_equipment( equipment_id: 5,  status_code: 0);
```

Before (Equipment Table):

Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2
5	Treadmill 2	2023-01-01 00:00:00.000000	1	Highest-quality treadmi…	1	1

After (Equipment Table):

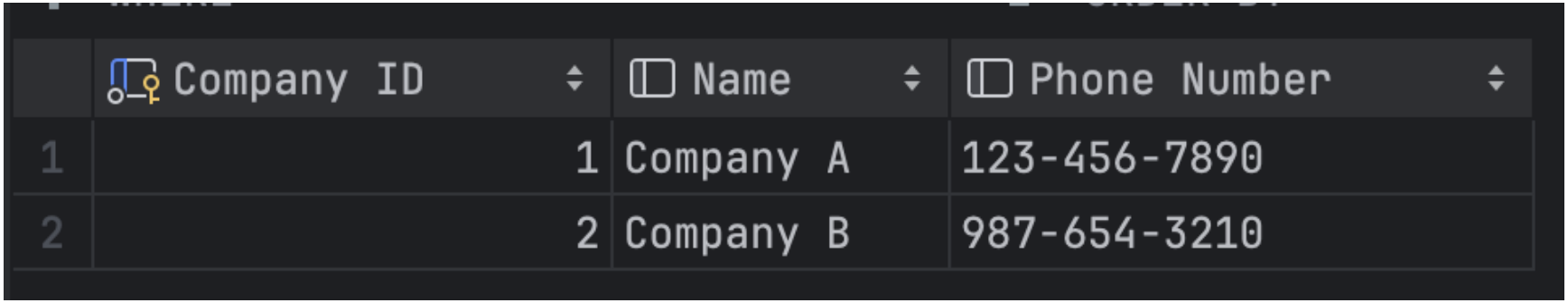
Equipment ID	Name	Purchase Date	Company ID	Equipment Details	Equipment Type ID	Branch ID
1	Treadmill 1	2023-01-01 00:00:00.000000	1	High-quality treadmill	1	1
2	Dumbbell Set 1	2023-02-01 00:00:00.000000	2	Set of dumbbells	2	1
3	Bike 1	2023-03-01 00:00:00.000000	1	Stationary bike	3	2



1. Add Company:

```
CREATE OR REPLACE PROCEDURE add_company(IN name VARCHAR(255), IN phone_number VARCHAR(15), INOUT status_code INT) AS
$$
BEGIN
    INSERT INTO public."Equipment Company"("Name", "Phone Number") VALUES (name, phone_number);
    COMMIT;

    status_code := 0; -- Assigning 0 to status_code
END;
$$ LANGUAGE plpgsql;

CALL add_company( name: 'Cool Company', phone_number: '555-555-6666', status_code: 0);
```

Before (Equipment Company Table):



	Company ID	Name	Phone Number
1	1	Company A	123-456-7890
2	2	Company B	987-654-3210

After (Equipment Company Table):

	Company ID	Name	Phone Number
1	1	Company A	123-456-7890
2	2	Company B	987-654-3210
3	4	Cool Company	555-555-6666

2. Edit Company:

```
CREATE OR REPLACE PROCEDURE edit_company(IN company_id INT, IN name VARCHAR(255), IN phone_number VARCHAR(15), INOUT status_code INT) AS
$$
BEGIN
    UPDATE public."Equipment Company"
        SET "Name" = name,
            "Phone Number" = phone_number
        WHERE "Company ID" = company_id;
    COMMIT;
    status_code := 0; -- Assigning 0 to status_code
END;
$$ LANGUAGE plpgsql;

CALL edit_company( company_id: 4,  name: 'Cooler Company',  phone_number: '555-555-6666',  status_code: 0);
```

Before (Equipment Company Table):

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210
4	Cool Company	555-555-6666

After (Equipment Company Table):

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210
4	Cooler Comp…	555-555-6666

3. Delete Company:

```
CREATE OR REPLACE PROCEDURE delete_company(
    IN company_id INT,
    INOUT status_code INT
) AS
$$
BEGIN
    -- Delete from Equipment Company table
    DELETE FROM public."Equipment Company"
    WHERE "Company ID" = company_id;
    COMMIT;
    -- Set status_code to 0 indicating success
    status_code := 0;
END;
$$ LANGUAGE plpgsql;

CALL delete_company( company_id: 4,  status_code: 0);
```

Before (Equipment Company Table):

Company ID	Name	Phone Number
1	Company A	123-456-7890
2	Company B	987-654-3210
4	Cooler Comp…	555-555-6666

After (Equipment Company Table):

	Company ID	Name	Phone Number
1	1	Company A	123-456-7890
2	2	Company B	987-654-3210

MUSCLE
MAJOR
GYM

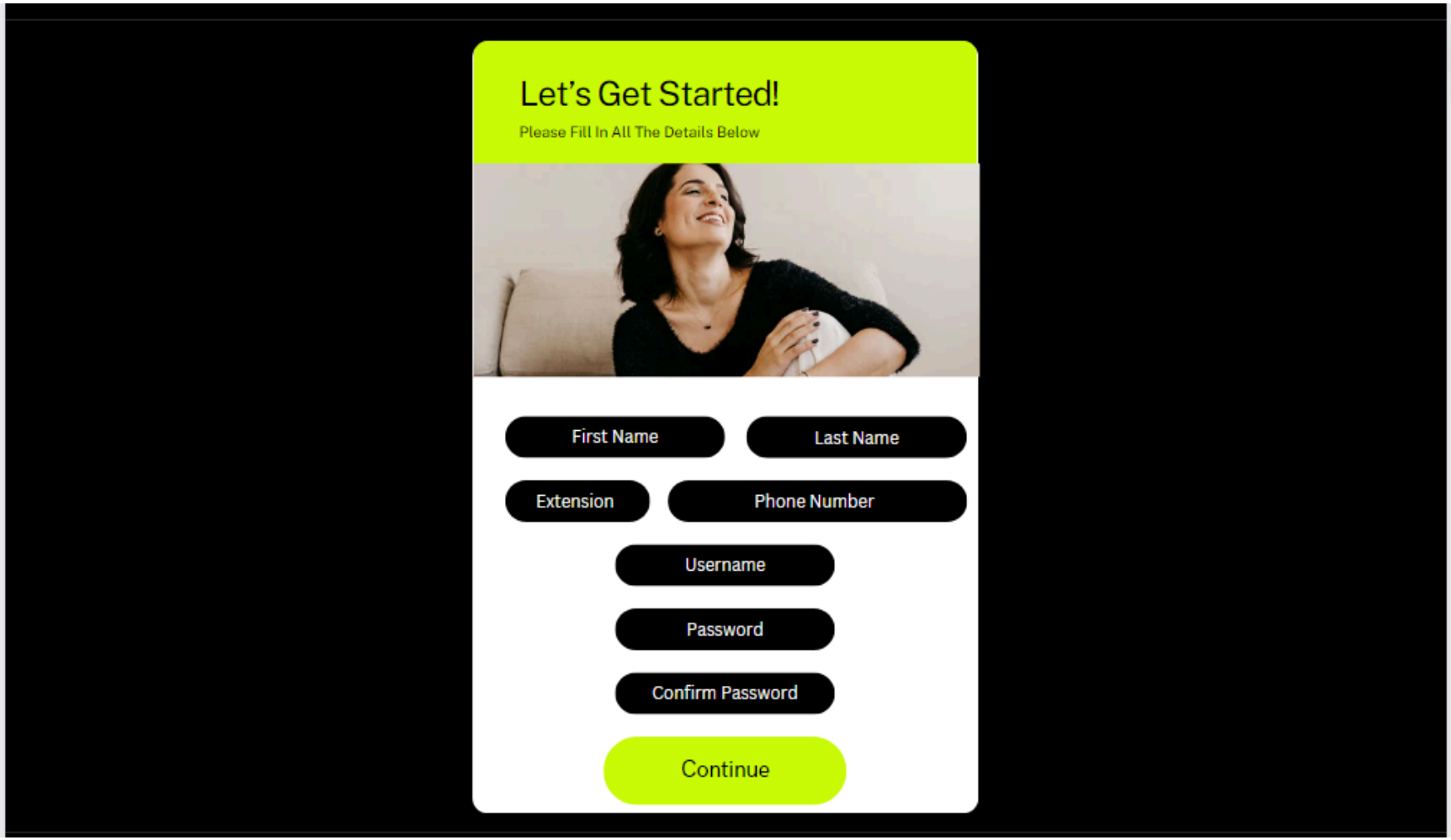
IT'S TIME TO BE HEALTHY AND IN GREAT SHAPE

SIGN UP NOW

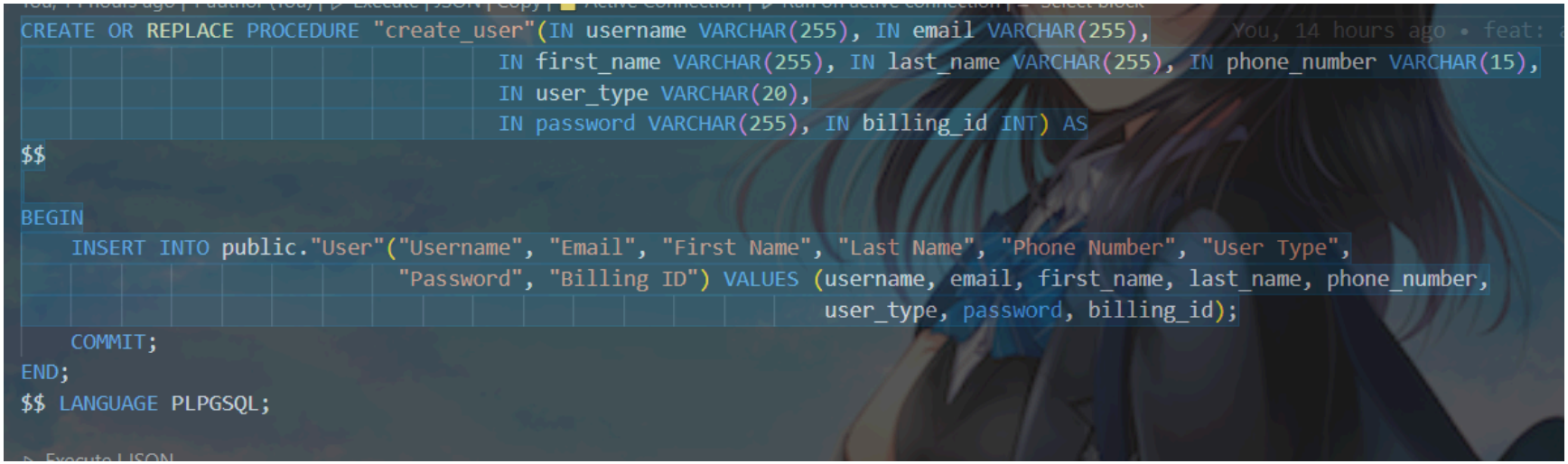
LOG IN

We offer classes and personal training designed to help you get fit.

Sign Up Page

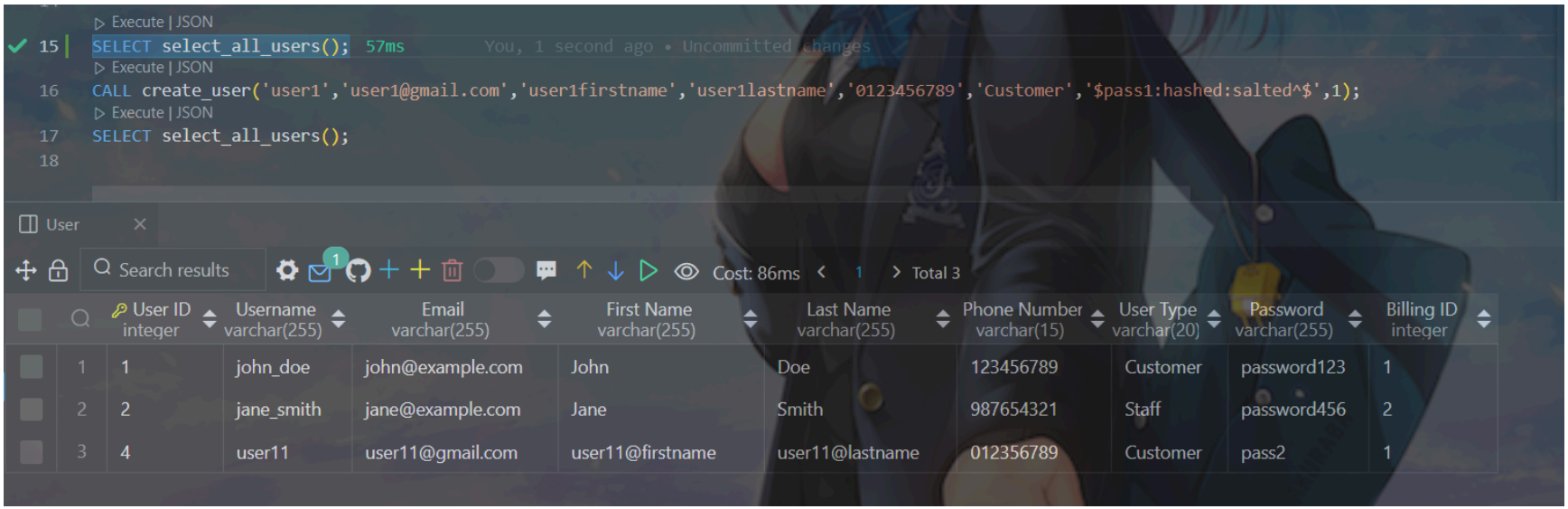

Let's Get Started!
Please Fill In All The Details Below
First Name
Last Name
Extension
Phone Number
Username
Password
Confirm Password
Continue

Code:

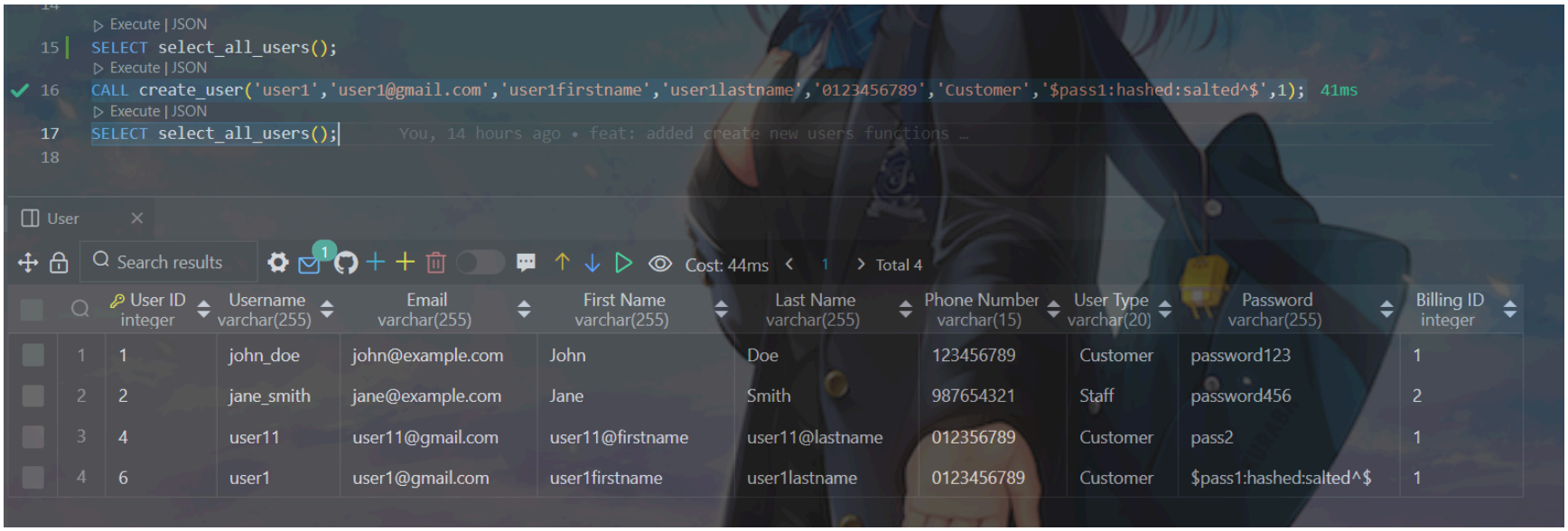


```
CREATE OR REPLACE PROCEDURE "create_user"(IN username VARCHAR(255), IN email VARCHAR(255),
                                          IN first_name VARCHAR(255), IN last_name VARCHAR(255), IN phone_number VARCHAR(15),
                                          IN user_type VARCHAR(20),
                                          IN password VARCHAR(255), IN billing_id INT) AS
$$

BEGIN
    INSERT INTO public."User"("Username", "Email", "First Name", "Last Name", "Phone Number", "User Type",
                              "Password", "Billing ID") VALUES (username, email, first_name, last_name, phone_number,
                                                                user_type, password, billing_id);
    COMMIT;
END;
$$ LANGUAGE PLPGSQL;
```

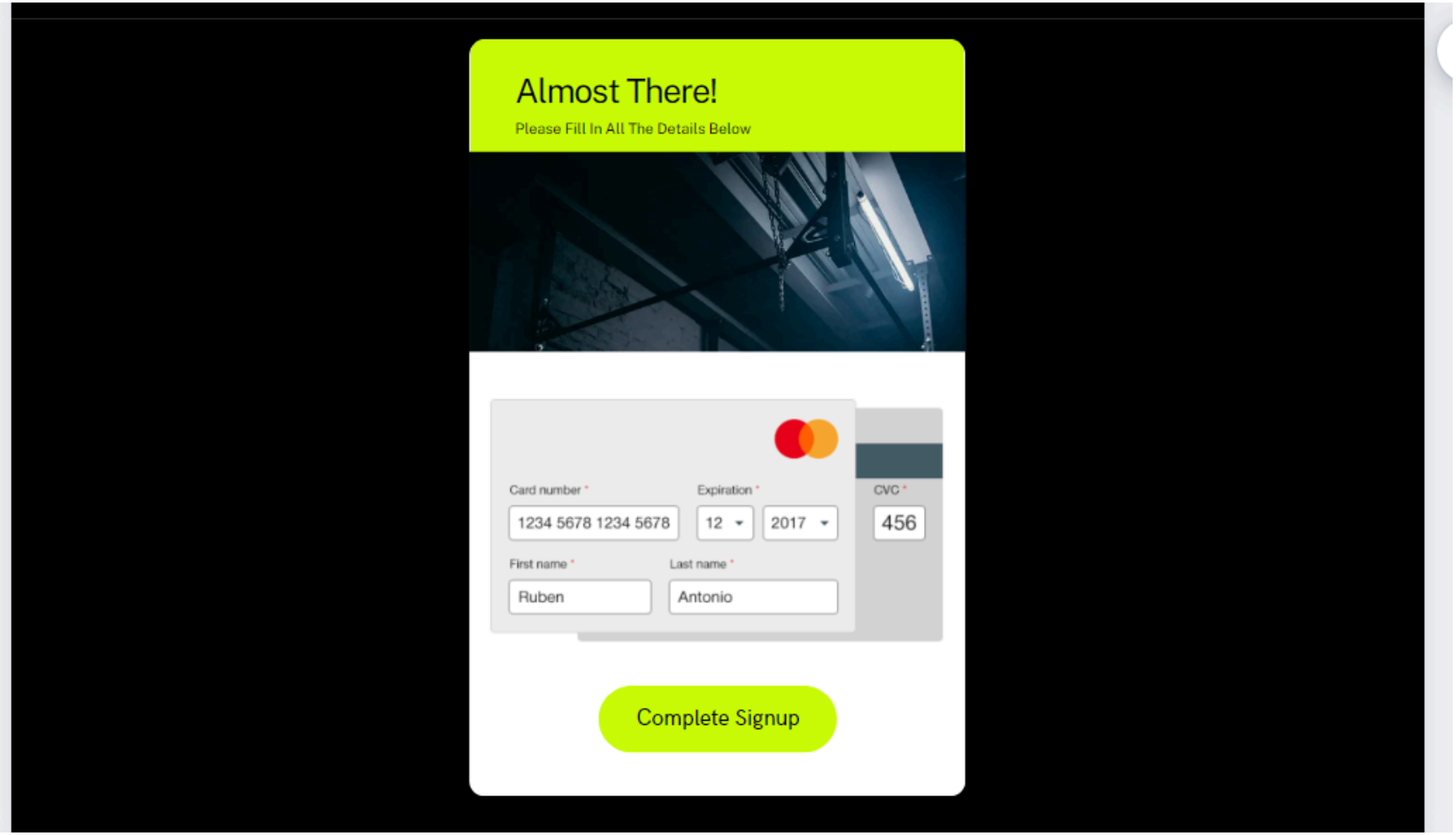
Before:


Execute | JSON
15 SELECT select_all_users(); 57ms You, 1 second ago • Uncommitted changes
Execute | JSON
16 CALL create_user('user1','user1@gmail.com','user1firstname','user1lastname','0123456789','Customer','$pass1:hashed:salted^$',1);
Execute | JSON
17 SELECT select_all_users();
18
User
Search results
Cost: 86ms 1 Total 3
User ID integer
Username varchar(255)
Email varchar(255)
First Name varchar(255)
Last Name varchar(255)
Phone Number varchar(15)
User Type varchar(20)
Password varchar(255)
Billing ID integer
1 1 john_doe john@example.com John Doe 123456789 Customer password123 1
2 2 jane_smith jane@example.com Jane Smith 987654321 Staff password456 2
3 4 user11 user11@gmail.com user11@firstname user11@lastname 012356789 Customer pass2 1

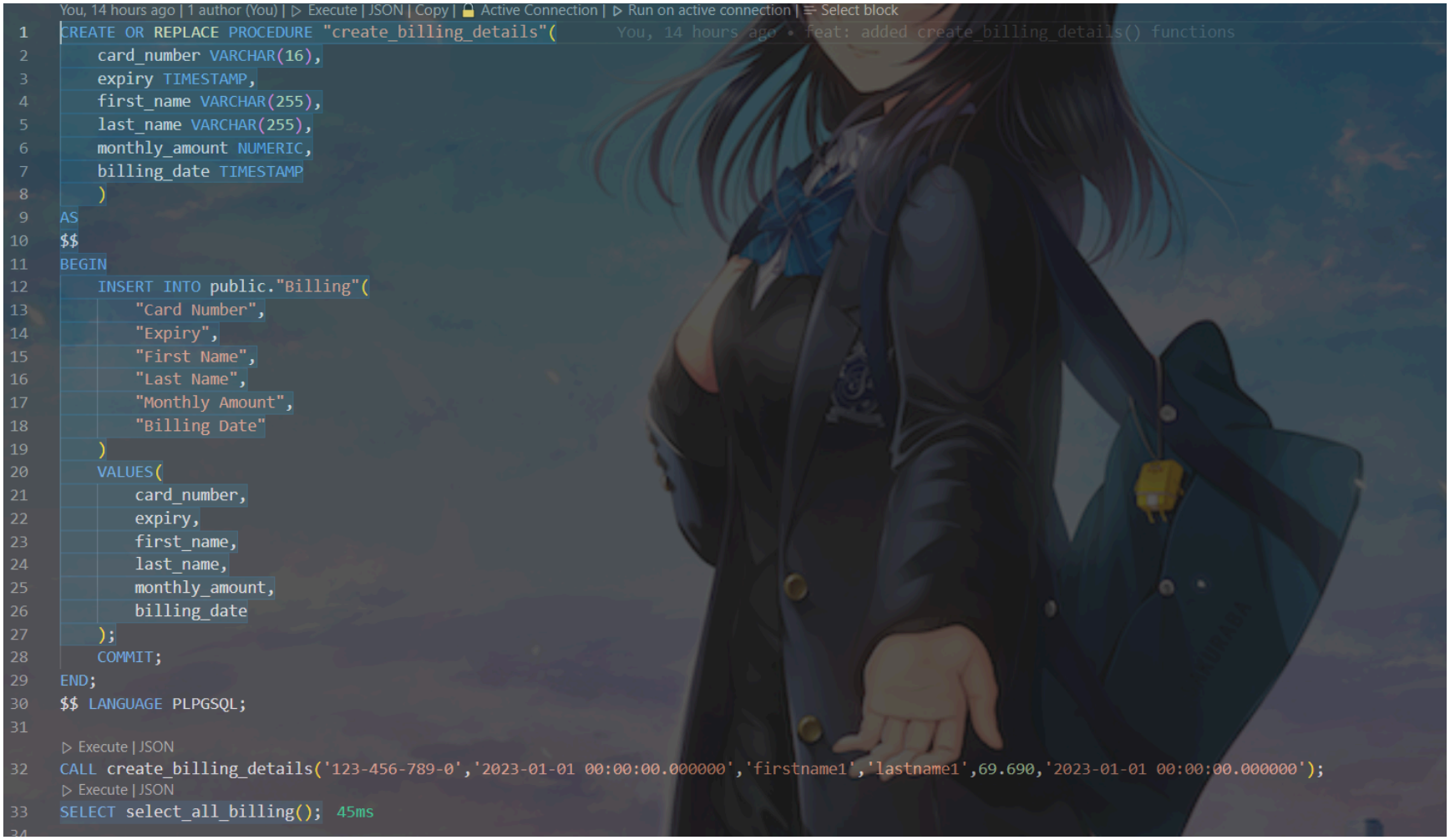
After:


Execute | JSON
15 SELECT select_all_users();
Execute | JSON
16 CALL create_user('user1','user1@gmail.com','user1firstname','user1lastname','0123456789','Customer','$pass1:hashed:salted^$',1); 41ms
Execute | JSON
17 SELECT select_all_users(); You, 14 hours ago • feat: added create new users functions …
18
User
Search results
Cost: 44ms 1 Total 4
User ID integer
Username varchar(255)
Email varchar(255)
First Name varchar(255)
Last Name varchar(255)
Phone Number varchar(15)
User Type varchar(20)
Password varchar(255)
Billing ID integer
1 1 john_doe john@example.com John Doe 123456789 Customer password123 1
2 2 jane_smith jane@example.com Jane Smith 987654321 Staff password456 2
3 4 user11 user11@gmail.com user11@firstname user11@lastname 012356789 Customer pass2 1
4 6 user1 user1@gmail.com user1firstname user1lastname 0123456789 Customer $pass1:hashed:salted^$ 1

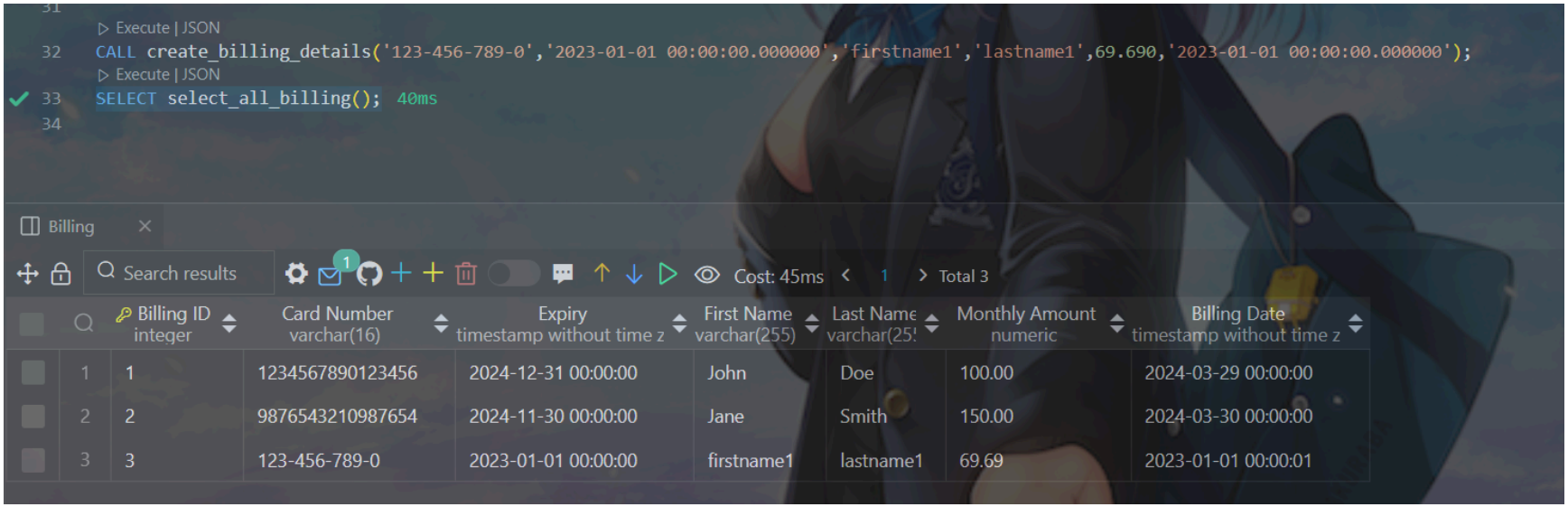
Create Billing Details



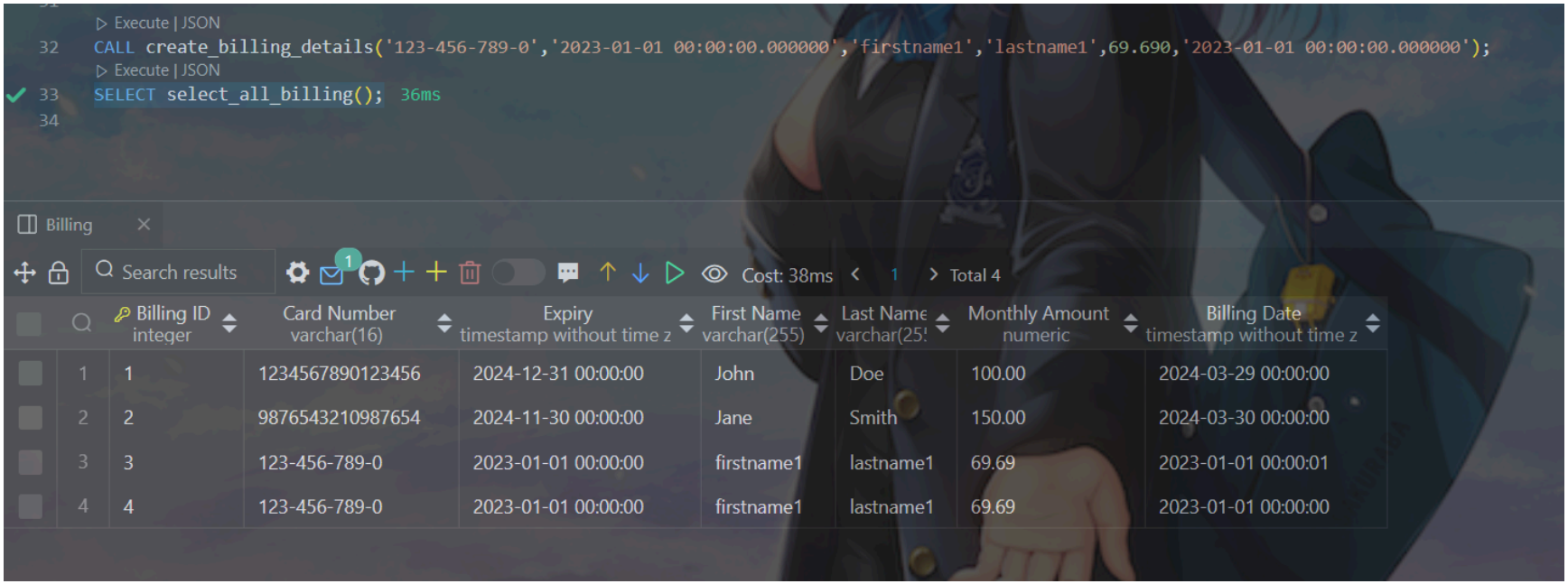
Code:


You, 14 hours ago | 1 author (You) | ▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
CREATE OR REPLACE PROCEDURE "create_billing_details"(
You, 14 hours ago • feat: added create_billing_details() functions
card_number VARCHAR(16),
expiry TIMESTAMP,
first_name VARCHAR(255),
last_name VARCHAR(255),
monthly_amount NUMERIC,
billing_date TIMESTAMP
)
AS
$$
BEGIN
INSERT INTO public."Billing"(
"Card Number",
"Expiry",
"First Name",
"Last Name",
"Monthly Amount",
"Billing Date"
)
VALUES(
card_number,
expiry,
first_name,
last_name,
monthly_amount,
billing_date
);
COMMIT;
END;
$$ LANGUAGE PLPGSQL;
▷ Execute | JSON
CALL create_billing_details('123-456-789-0','2023-01-01 00:00:00.000000','firstname1','lastname1',69.690,'2023-01-01 00:00:00.000000');
▷ Execute | JSON
SELECT select_all_billing(); 45ms

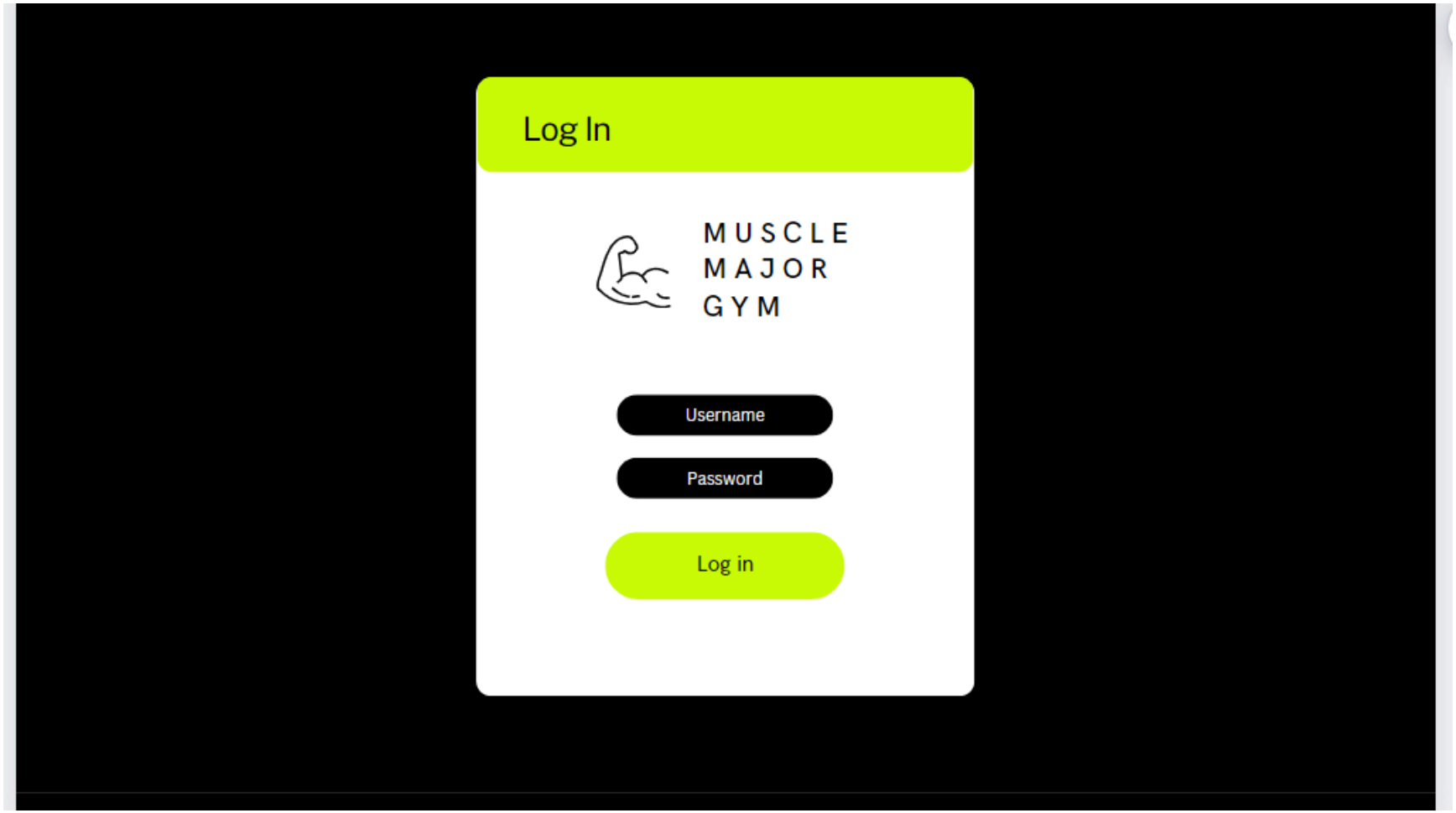
Before:


Execute | JSON
32 CALL create_billing_details('123-456-789-0','2023-01-01 00:00:00.000000','firstname1','lastname1',69.690,'2023-01-01 00:00:00.000000');
Execute | JSON
33 SELECT select_all_billing(); 40ms
34
Billing
Search results
Cost: 45ms
1
Total 3
Billing ID integer
Card Number varchar(16)
Expiry timestamp without time z
First Name varchar(255)
Last Name varchar(25
Monthly Amount numeric
Billing Date timestamp without time z
1 1 1234567890123456 2024-12-31 00:00:00 John Doe 100.00 2024-03-29 00:00:00
2 2 9876543210987654 2024-11-30 00:00:00 Jane Smith 150.00 2024-03-30 00:00:00
3 3 123-456-789-0 2023-01-01 00:00:00 firstname1 lastname1 69.69 2023-01-01 00:00:01

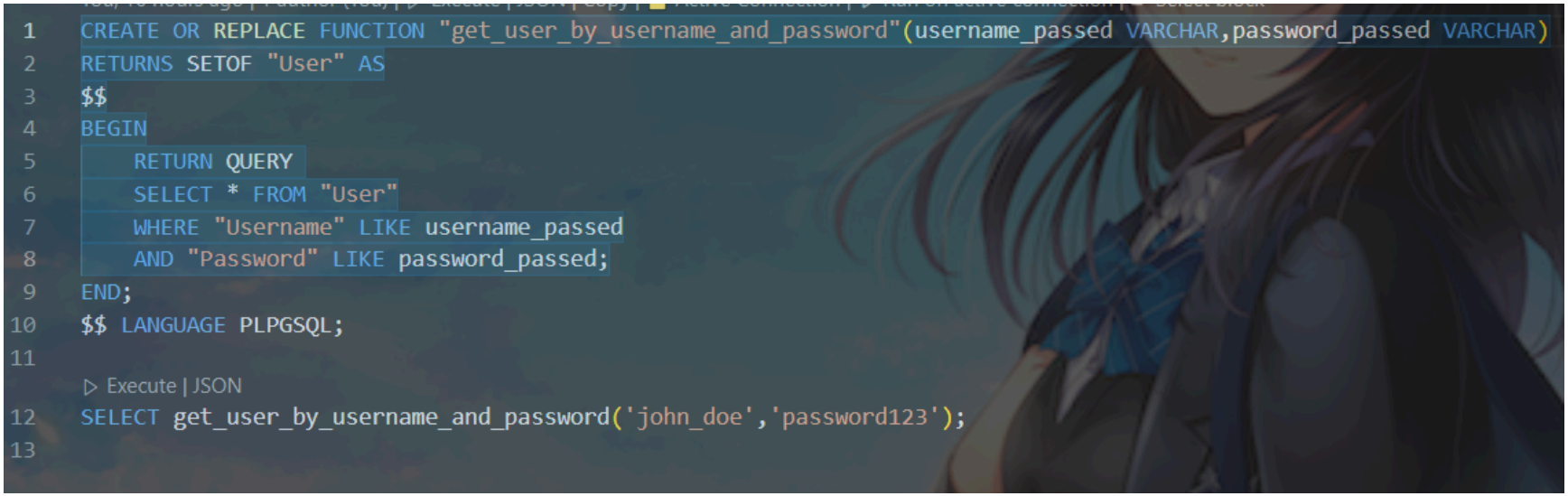
After:


Execute | JSON
32 CALL create_billing_details('123-456-789-0','2023-01-01 00:00:00.000000','firstname1','lastname1',69.690,'2023-01-01 00:00:00.000000');
Execute | JSON
33 SELECT select_all_billing(); 36ms
34
Billing
Search results
Cost: 38ms
1
Total 4
Billing ID integer
Card Number varchar(16)
Expiry timestamp without time z
First Name varchar(255)
Last Name varchar(25
Monthly Amount numeric
Billing Date timestamp without time z
1 1 1234567890123456 2024-12-31 00:00:00 John Doe 100.00 2024-03-29 00:00:00
2 2 9876543210987654 2024-11-30 00:00:00 Jane Smith 150.00 2024-03-30 00:00:00
3 3 123-456-789-0 2023-01-01 00:00:00 firstname1 lastname1 69.69 2023-01-01 00:00:01
4 4 123-456-789-0 2023-01-01 00:00:00 firstname1 lastname1 69.69 2023-01-01 00:00:00

Login

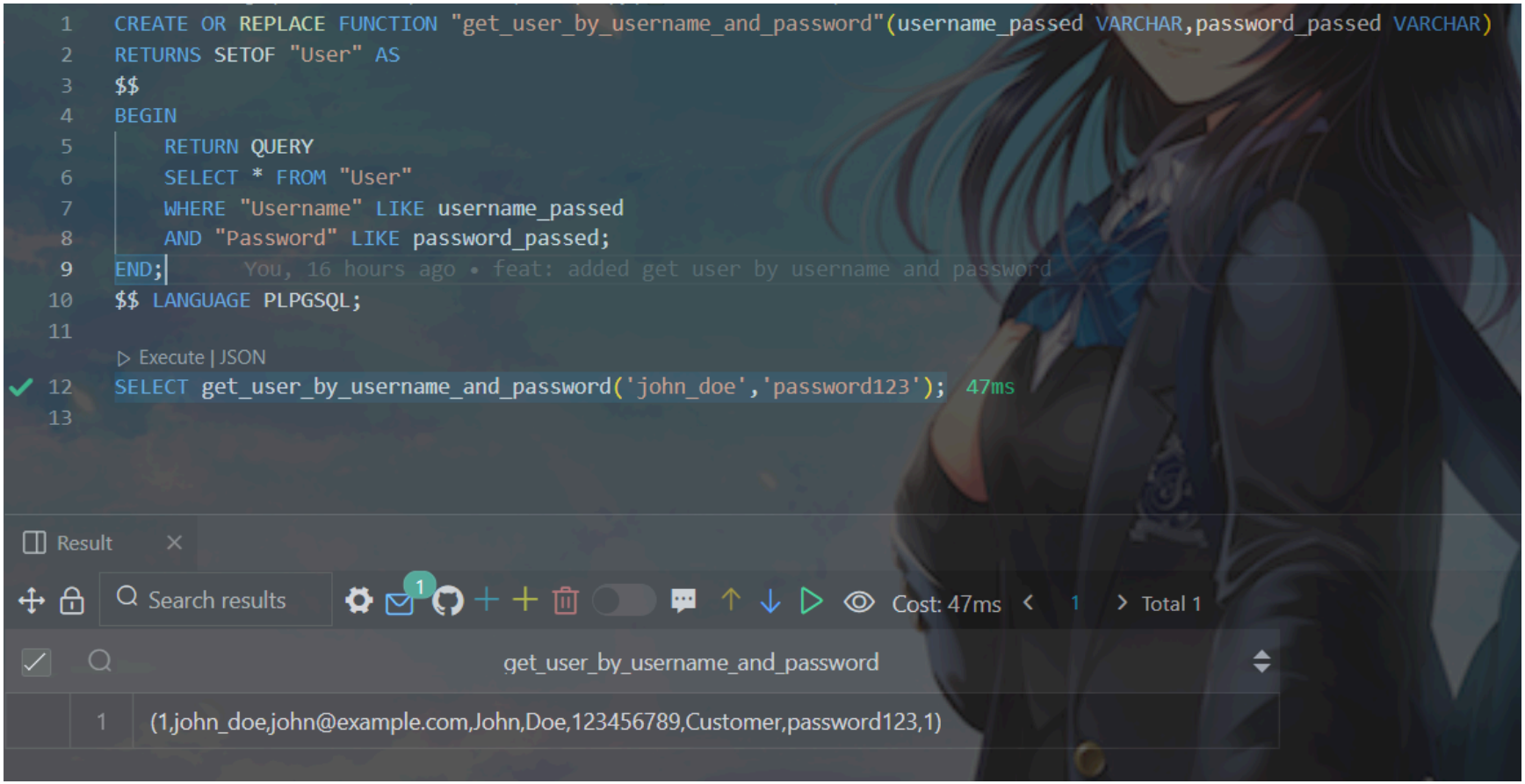


Code

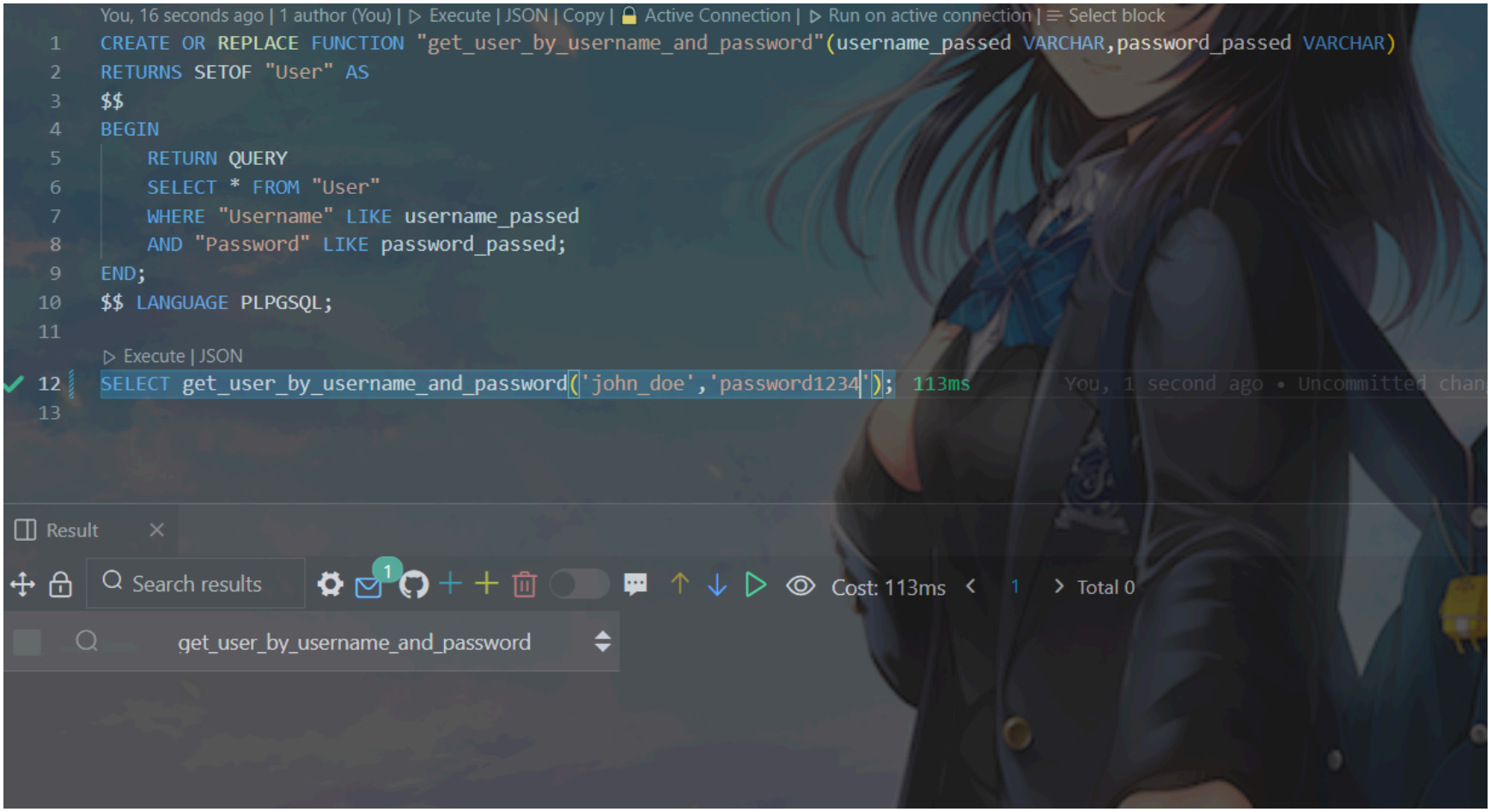


```sql
CREATE OR REPLACE FUNCTION "get_user_by_username_and_password"(username_passed VARCHAR,password_passed VARCHAR)
RETURNS SETOF "User" AS
$$
BEGIN
    RETURN QUERY
    SELECT * FROM "User"
    WHERE "Username" LIKE username_passed
    AND "Password" LIKE password_passed;
END;
$$ LANGUAGE PLPGSQL;

▷ Execute | JSON
SELECT get_user_by_username_and_password('john_doe','password123');
```

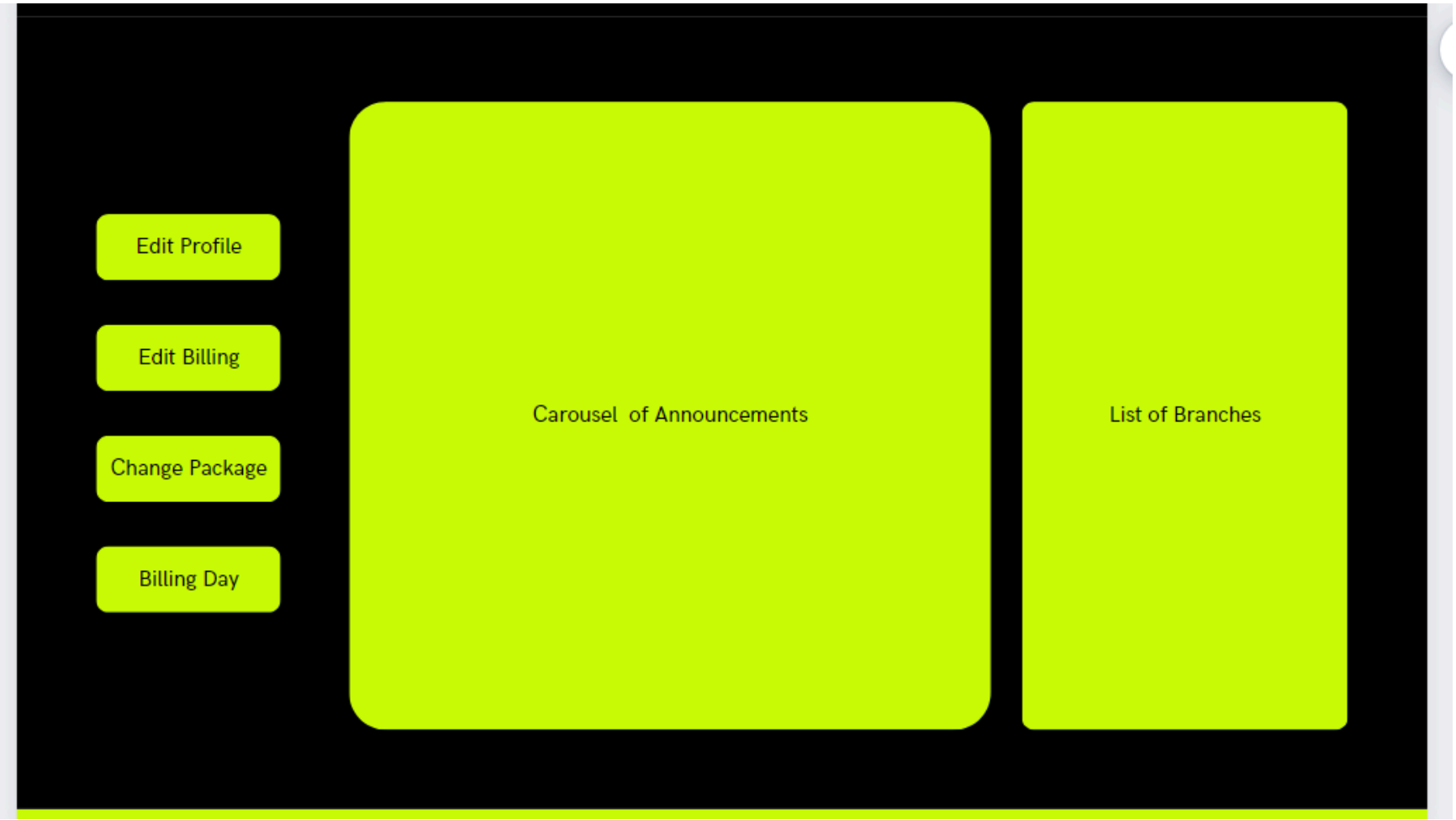
Correct Password



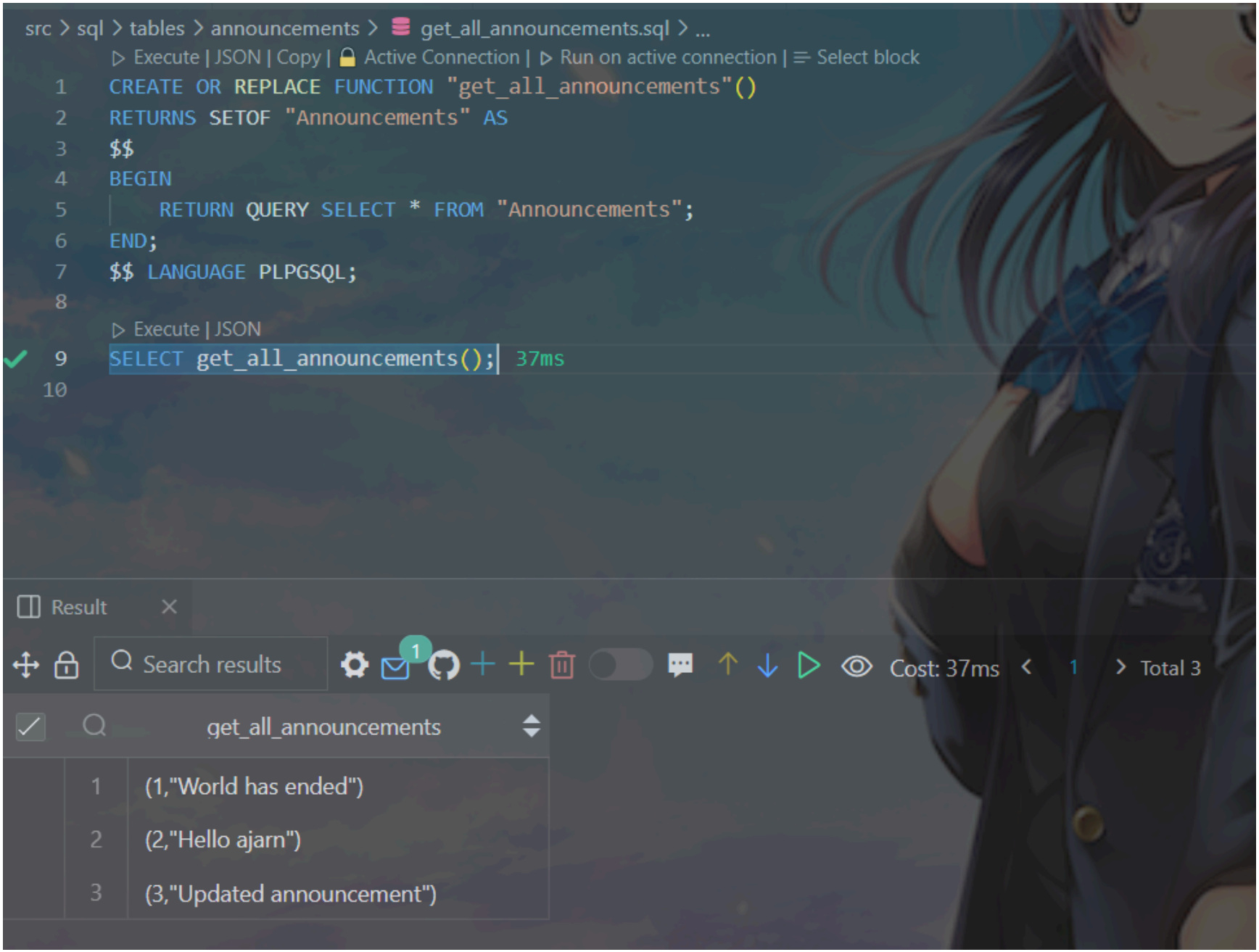
Wrong Password



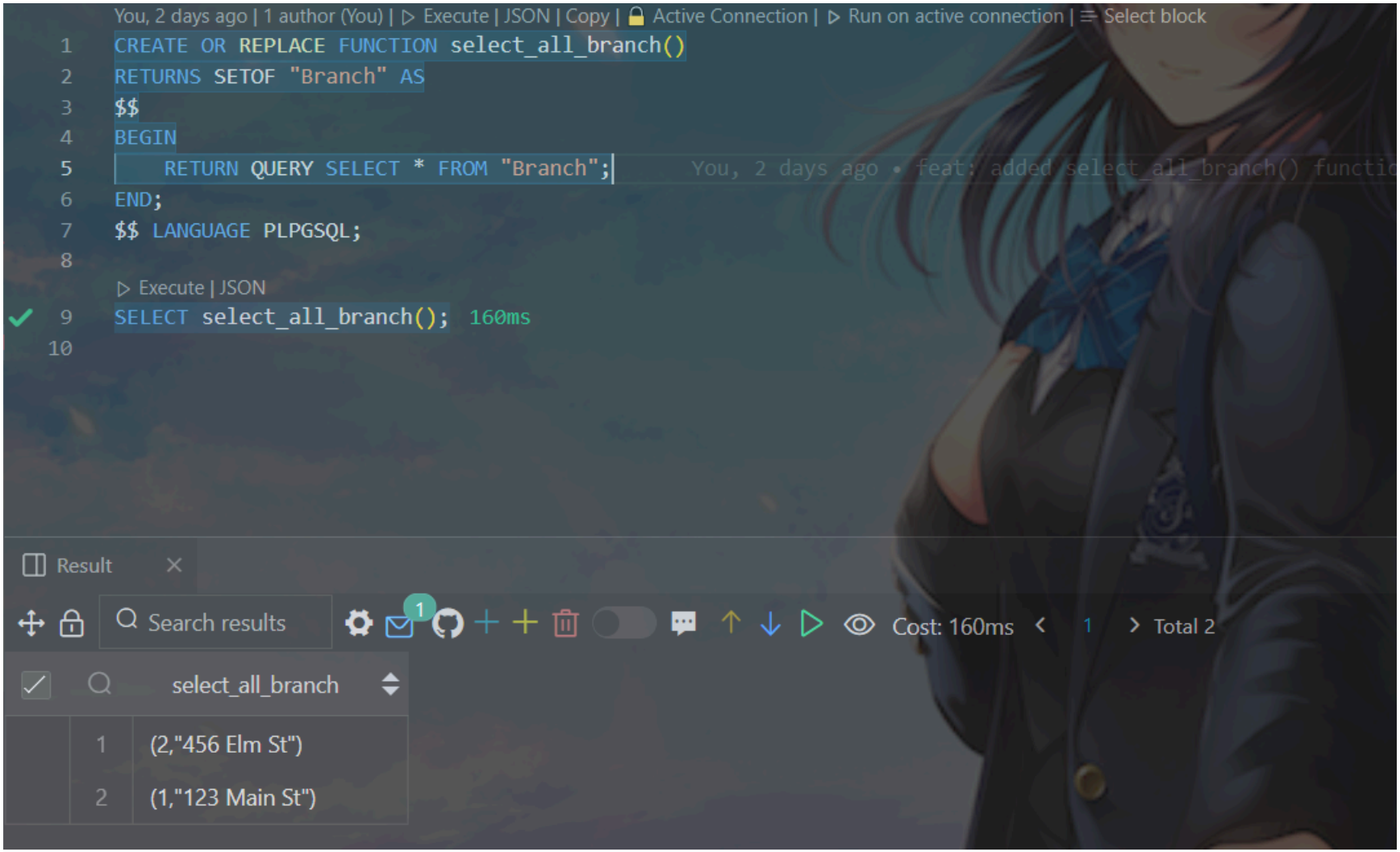
User Home Page



Announcements:
Code And Results:


src > sql > tables > announcements > get_all_announcements.sql > ...
▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
1 CREATE OR REPLACE FUNCTION "get_all_announcements"()
2 RETURNS SETOF "Announcements" AS
3 $$
4 BEGIN
5 RETURN QUERY SELECT * FROM "Announcements";
6 END;
7 $$ LANGUAGE PLPGSQL;
8
▷ Execute | JSON
9 SELECT get_all_announcements(); 37ms
10
Result
Search results
Cost: 37ms
1
Total 3
get_all_announcements
1 (1,"World has ended")
2 (2,"Hello ajarn")
3 (3,"Updated announcement")

List of Branches:
Code And Results:



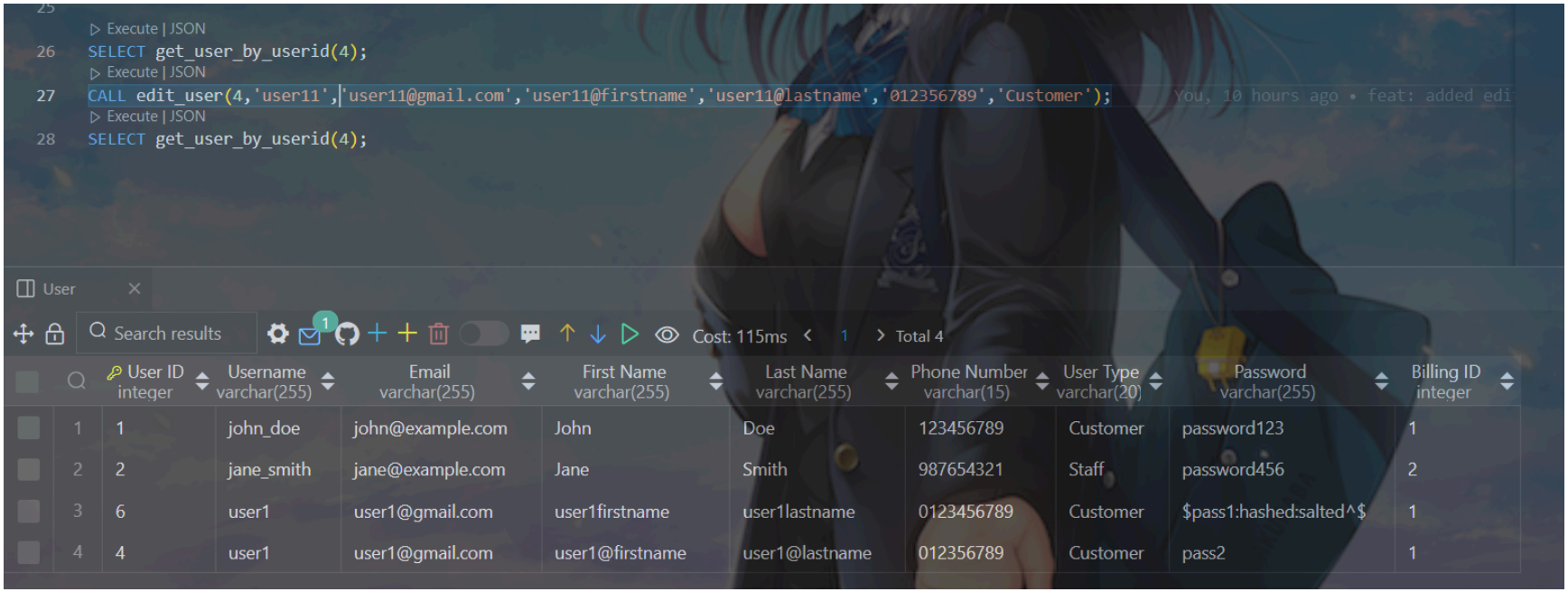
Edit Profiles


Edit Profile
First Name
Last Name
Extension
Phone Number
Username
Password
Confirm Password
Save

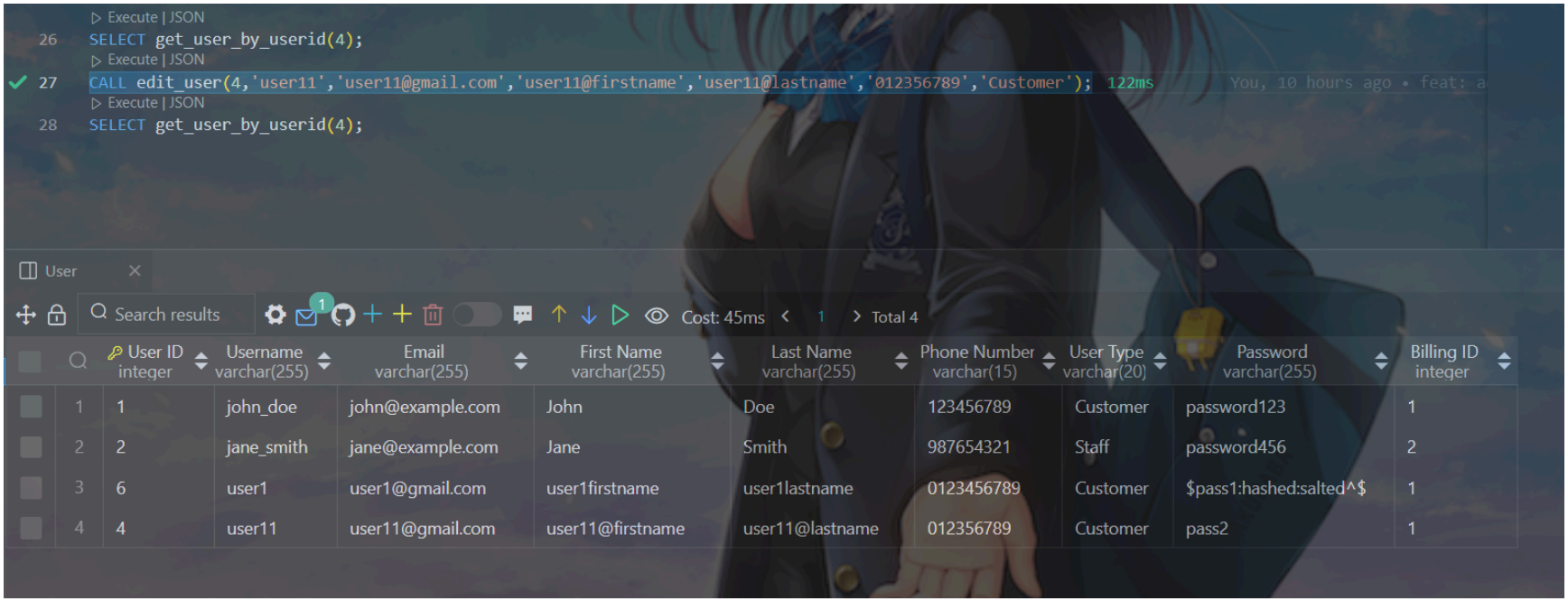
Code:

```
▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
CREATE OR REPLACE PROCEDURE "edit_user"(
    userid INT,
    new_username VARCHAR(255),
    new_email VARCHAR(255),
    new_first_name VARCHAR(255),
    new_last_name VARCHAR(255),
    new_phone_number VARCHAR(15),
    new_user_type VARCHAR(20)
)
AS
$$
BEGIN
    UPDATE public."User"
    SET
        "Username" = new_username,
        "Email" = new_email,
        "First Name" = new_first_name,
        "Last Name" = new_last_name,
        "Phone Number" = new_phone_number,
        "User Type" = new_user_type
    WHERE "User ID" = userid;
    COMMIT;
END;
$$ LANGUAGE PLPGSQL;

▷ Execute | JSON
SELECT get_user_by_userid(4);
▷ Execute | JSON
CALL edit_user(4,'user11','user11@gmail.com','user11@firstname','user11@lastname','012356789','Customer');
▷ Execute | JSON
SELECT get_user_by_userid(4);  35ms
```

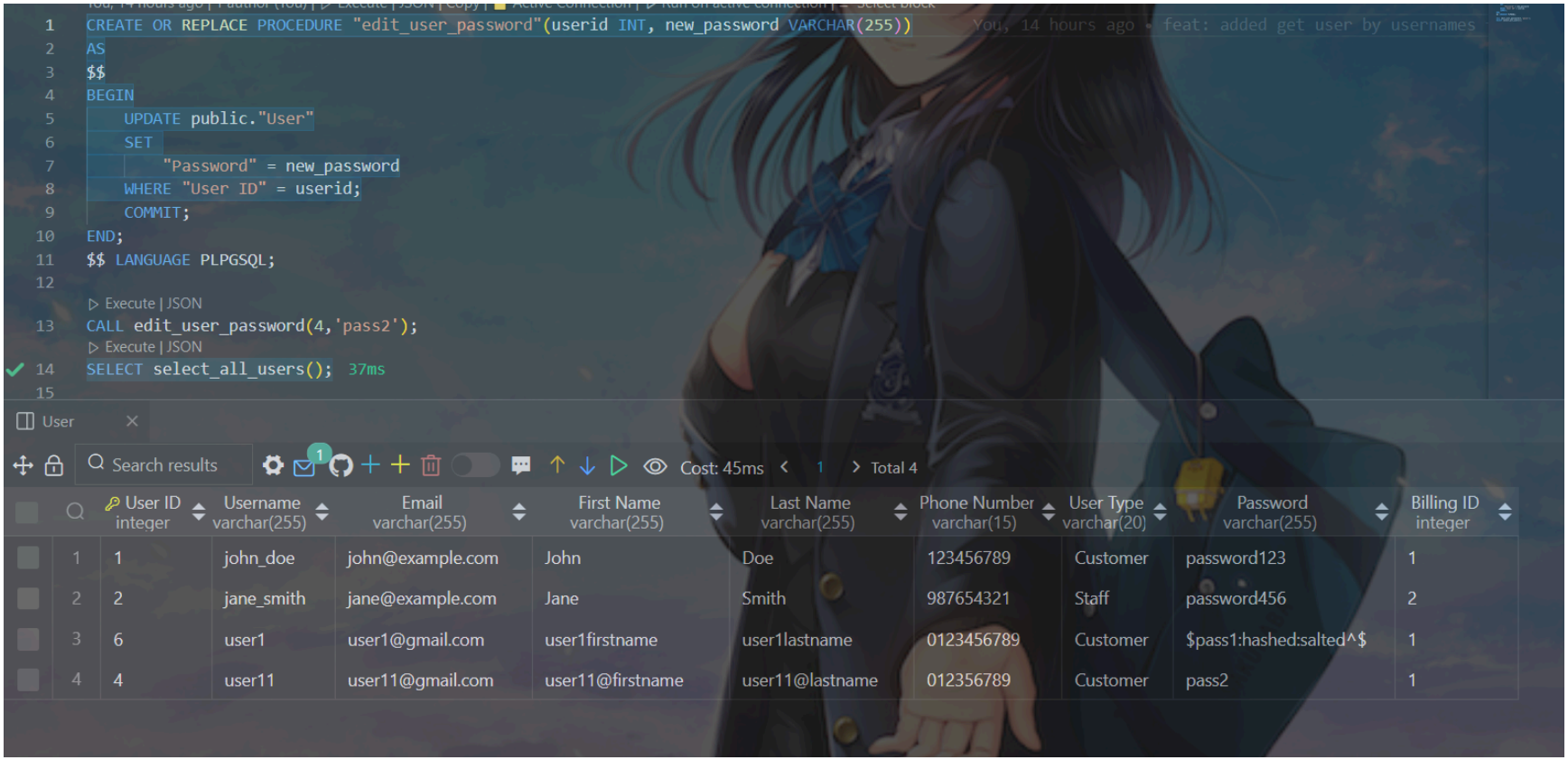
Before:


Execute | JSON
SELECT get_user_by_userid(4);
Execute | JSON
CALL edit_user(4,'user11','user11@gmail.com','user11@firstname','user11@lastname','012356789','Customer');
You, 10 hours ago • feat: added edi
Execute | JSON
SELECT get_user_by_userid(4);
User
Search results
Cost: 115ms
1
Total 4
User ID integer
Username varchar(255)
Email varchar(255)
First Name varchar(255)
Last Name varchar(255)
Phone Number varchar(15)
User Type varchar(20)
Password varchar(255)
Billing ID integer
1 1 john_doe john@example.com John Doe 123456789 Customer password123 1
2 2 jane_smith jane@example.com Jane Smith 987654321 Staff password456 2
3 6 user1 user1@gmail.com user1firstname user1lastname 0123456789 Customer $pass1:hashed:salted^$ 1
4 4 user1 user1@gmail.com user1@firstname user1@lastname 012356789 Customer pass2 1

After:



Change User Password
Code


CREATE OR REPLACE PROCEDURE "edit_user_password"(userid INT, new_password VARCHAR(255))
You, 14 hours ago • feat: added get user by usernames
AS
$$
BEGIN
UPDATE public."User"
SET
"Password" = new_password
WHERE "User ID" = userid;
COMMIT;
END;
$$ LANGUAGE PLPGSQL;
Execute | JSON
CALL edit_user_password(4,'pass2');
Execute | JSON
SELECT select_all_users(); 37ms
User
Search results
Cost: 45ms
1
Total 4
User ID integer
Username varchar(255)
Email varchar(255)
First Name varchar(255)
Last Name varchar(255)
Phone Number varchar(15)
User Type varchar(20)
Password varchar(255)
Billing ID integer
1 1 john_doe john@example.com John Doe 123456789 Customer password123 1
2 2 jane_smith jane@example.com Jane Smith 987654321 Staff password456 2
3 6 user1 user1@gmail.com user1firstname user1lastname 0123456789 Customer $pass1:hashed:salted^$ 1
4 4 user11 user11@gmail.com user11@firstname user11@lastname 012356789 Customer pass2 1

Before:

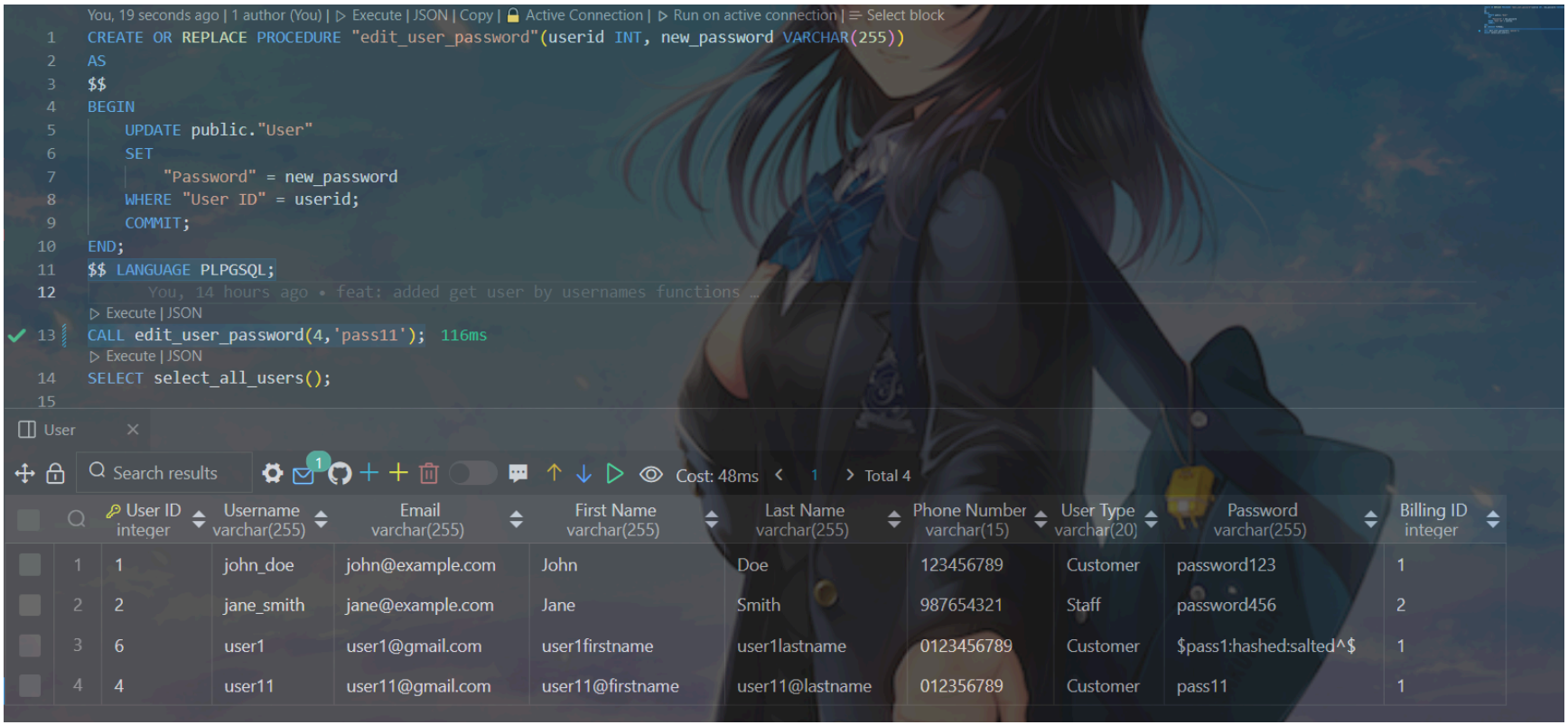
```sql
CREATE OR REPLACE PROCEDURE "edit_user_password"(userid INT, new_password VARCHAR(255))
AS
$$
BEGIN
    UPDATE public."User"
    SET
        "Password" = new_password
    WHERE "User ID" = userid;
    COMMIT;
END;
$$ LANGUAGE PLPGSQL;

▷ Execute | JSON
CALL edit_user_password(4,'pass11');      You, 1 second ago • Uncommitted changes
▷ Execute | JSON
SELECT select_all_users();
```

User

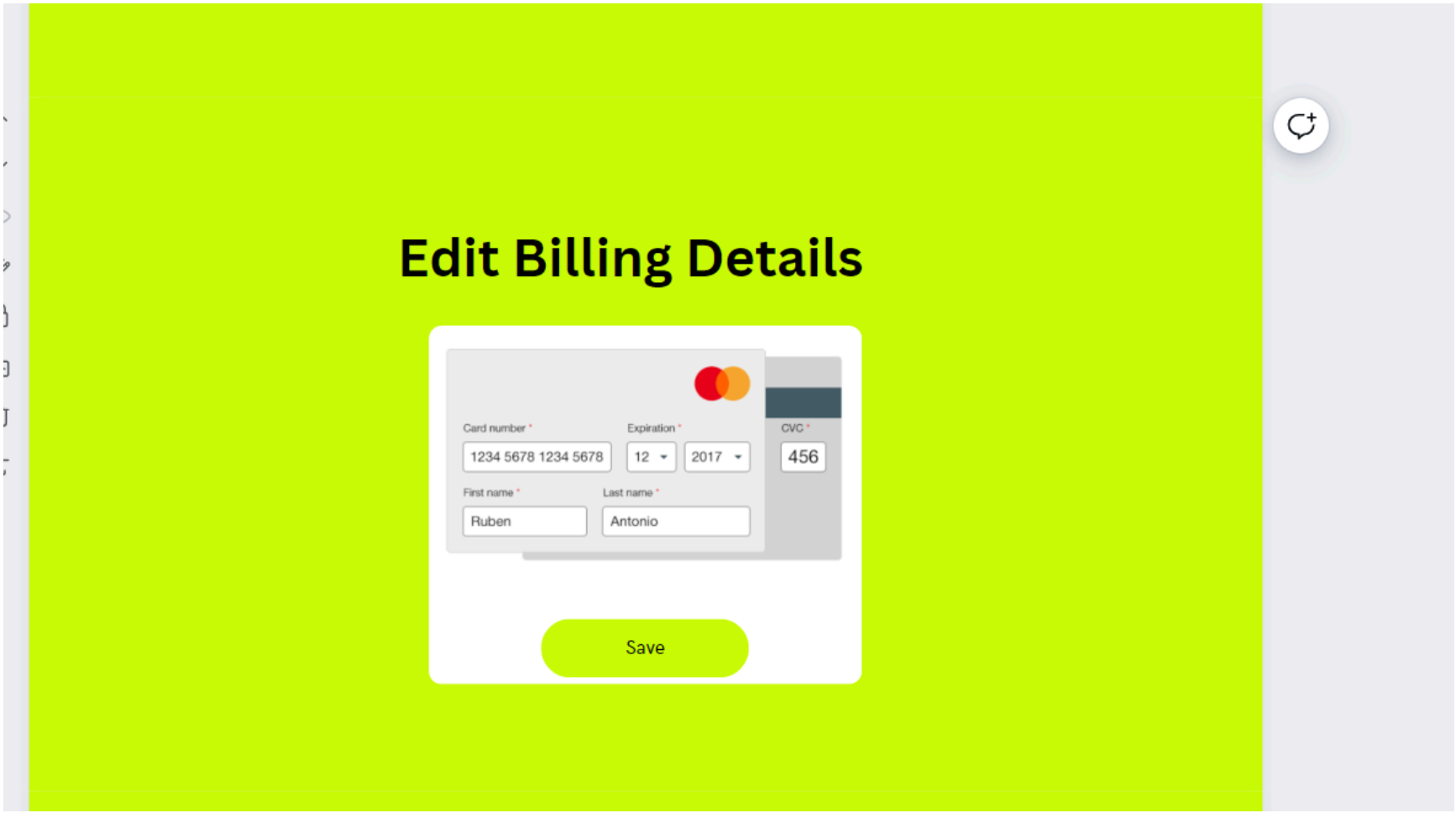
Search results Cost: 45ms 1 Total 4

	User ID integer	Username varchar(255)	Email varchar(255)	First Name varchar(255)	Last Name varchar(255)	Phone Number varchar(15)	User Type varchar(20)	Password varchar(255)	Billing ID integer
1	1	john_doe	john@example.com	John	Doe	123456789	Customer	password123	1
2	2	jane_smith	jane@example.com	Jane	Smith	987654321	Staff	password456	2
3	6	user1	user1@gmail.com	user1firstname	user1lastname	0123456789	Customer	$pass1:hashed:salted^$	1
4	4	user11	user11@gmail.com	user11@firstname	user11@lastname	012356789	Customer	pass2	1

After:



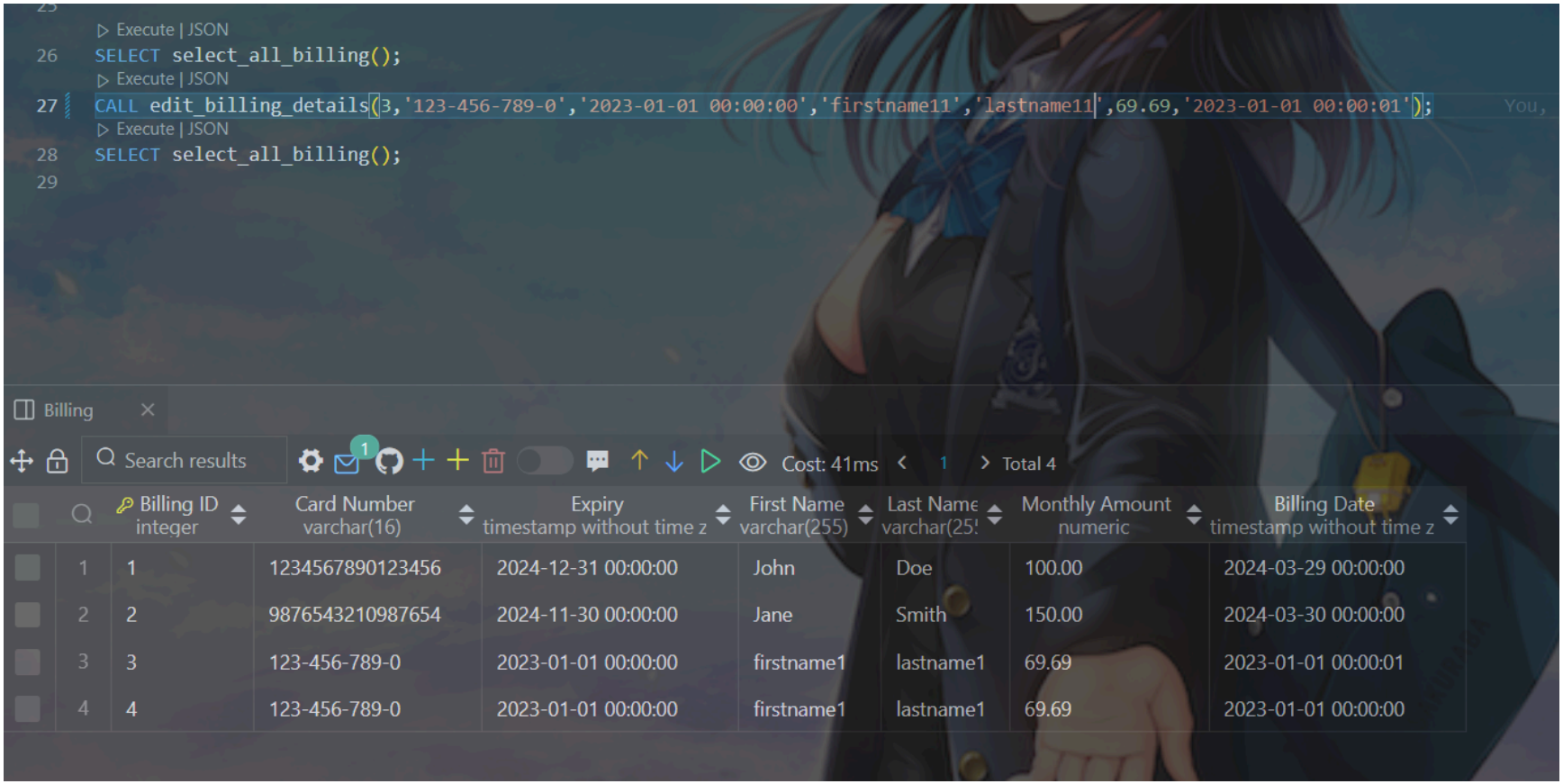
Edit Billing Details:


Edit Billing Details
Card number *
1234 5678 1234 5678
Expiration *
12
2017
CVC *
456
First name *
Ruben
Last name *
Antonio
Save

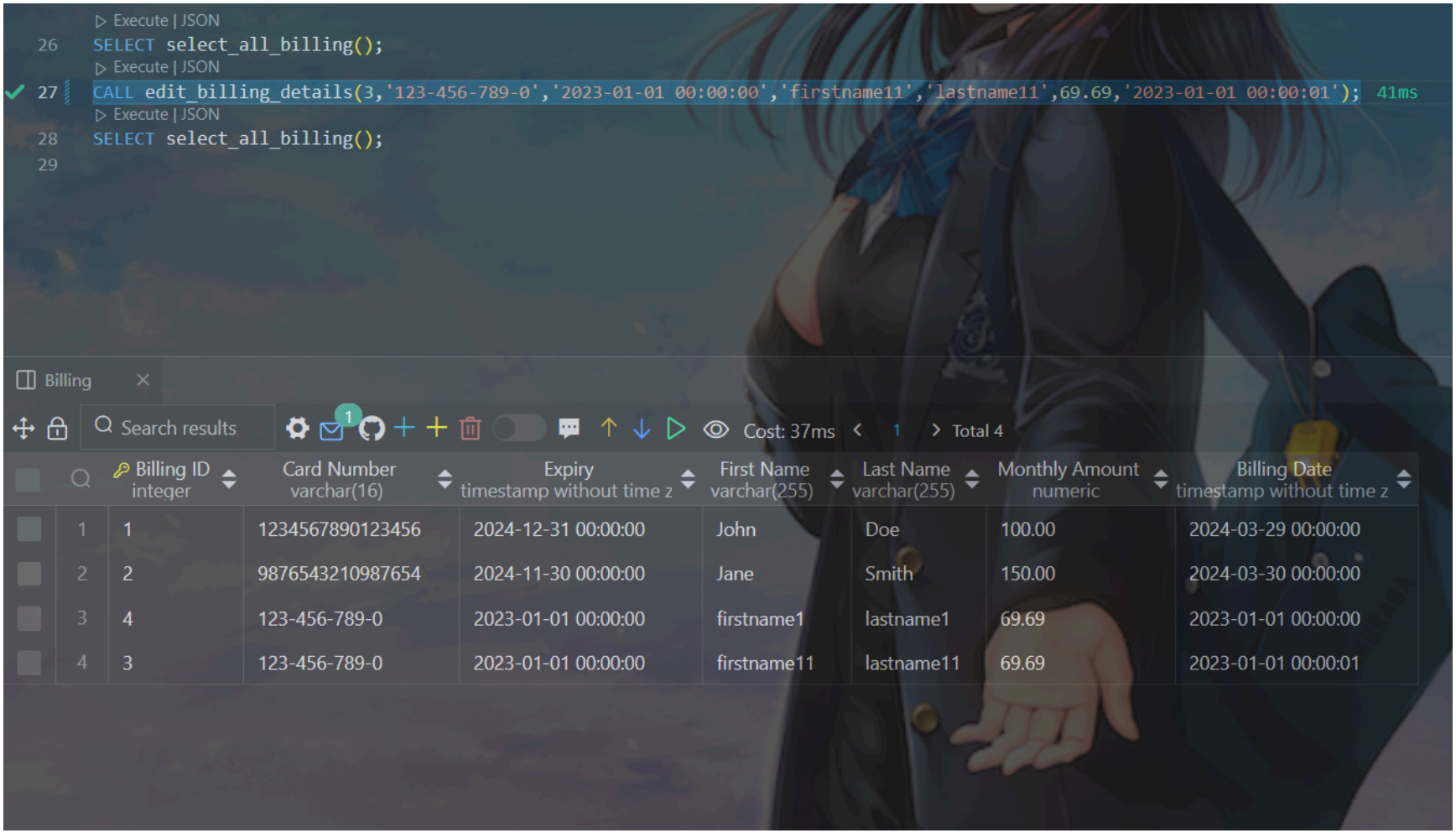
Codes:


You, 10 hours ago | 1 author (You) | ▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
CREATE OR REPLACE PROCEDURE "edit_billing_details"(
billing_id INT,
new_card_number VARCHAR(16),
new_expiry TIMESTAMP,
new_first_name VARCHAR(255),
new_last_name VARCHAR(255),
new_monthly_amount NUMERIC,
new_billing_date TIMESTAMP
)
AS
$$
BEGIN
UPDATE "Billing"
SET
"Card Number" = new_card_number,
"Expiry" = new_expiry,
"First Name" = new_first_name,
"Last Name" = new_last_name,
"Monthly Amount" = new_monthly_amount,
"Billing Date" = new_billing_date
WHERE "Billing ID" = billing_id;
COMMIT;
END;
$$ LANGUAGE PLPGSQL;
▷ Execute | JSON
SELECT select_all_billing();
▷ Execute | JSON
CALL edit_billing_details(3,'123-456-789-0','2023-01-01 00:00:00','firstname1','lastname1',69.69,'2023-01-01 00:00:01');
▷ Execute | JSON
SELECT select_all_billing(); 42ms
You, 10 hours ago • feat: added update billings details ...

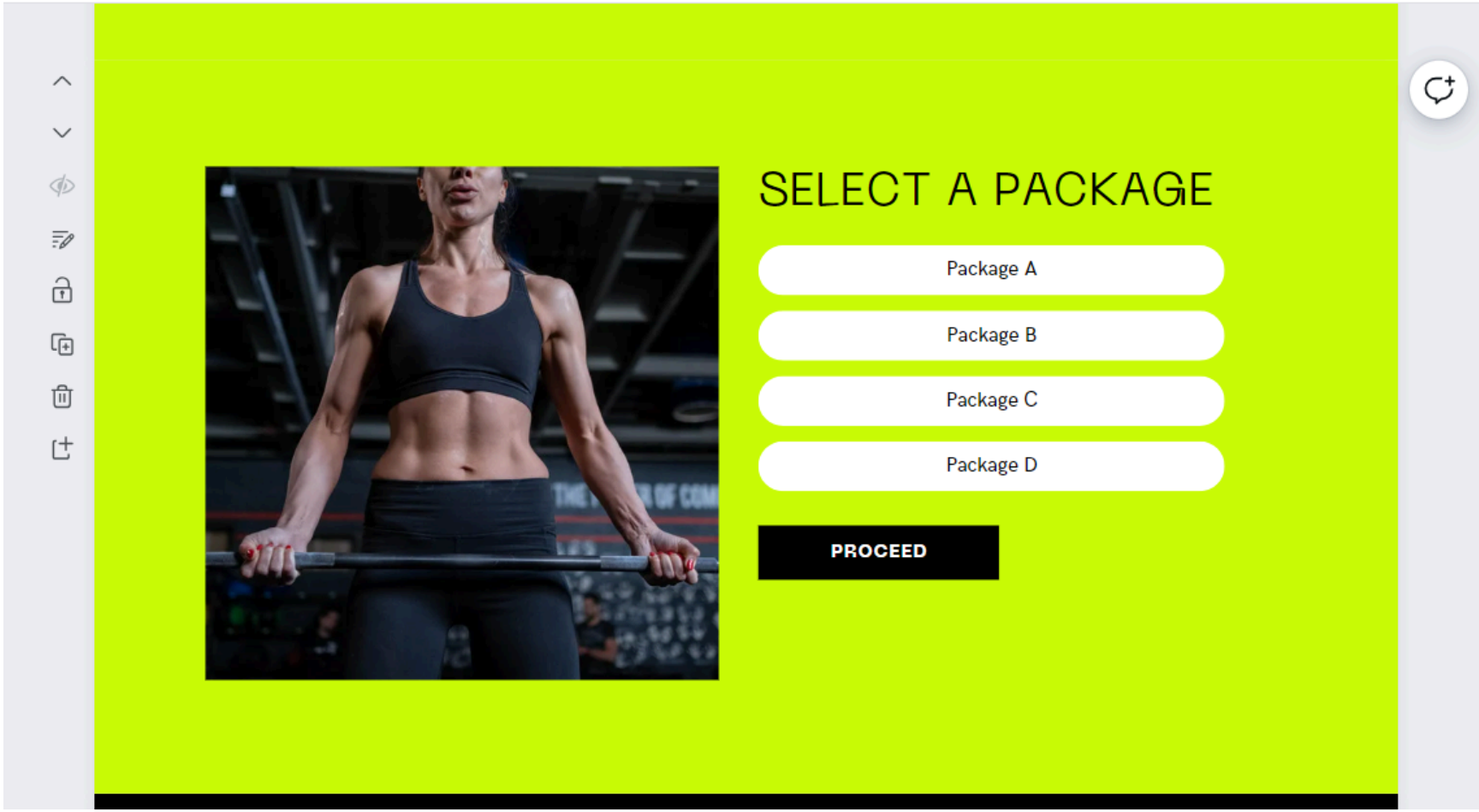
Before:


Execute | JSON
SELECT select_all_billing();
Execute | JSON
CALL edit_billing_details(3,'123-456-789-0','2023-01-01 00:00:00','firstname11','lastname11',69.69,'2023-01-01 00:00:01');
Execute | JSON
SELECT select_all_billing();
Billing
Search results
Cost: 41ms
1
Total 4
Billing ID integer
Card Number varchar(16)
Expiry timestamp without time z
First Name varchar(255)
Last Name varchar(25
Monthly Amount numeric
Billing Date timestamp without time z
1 1 1234567890123456 2024-12-31 00:00:00 John Doe 100.00 2024-03-29 00:00:00
2 2 9876543210987654 2024-11-30 00:00:00 Jane Smith 150.00 2024-03-30 00:00:00
3 3 123-456-789-0 2023-01-01 00:00:00 firstname1 lastname1 69.69 2023-01-01 00:00:01
4 4 123-456-789-0 2023-01-01 00:00:00 firstname1 lastname1 69.69 2023-01-01 00:00:00

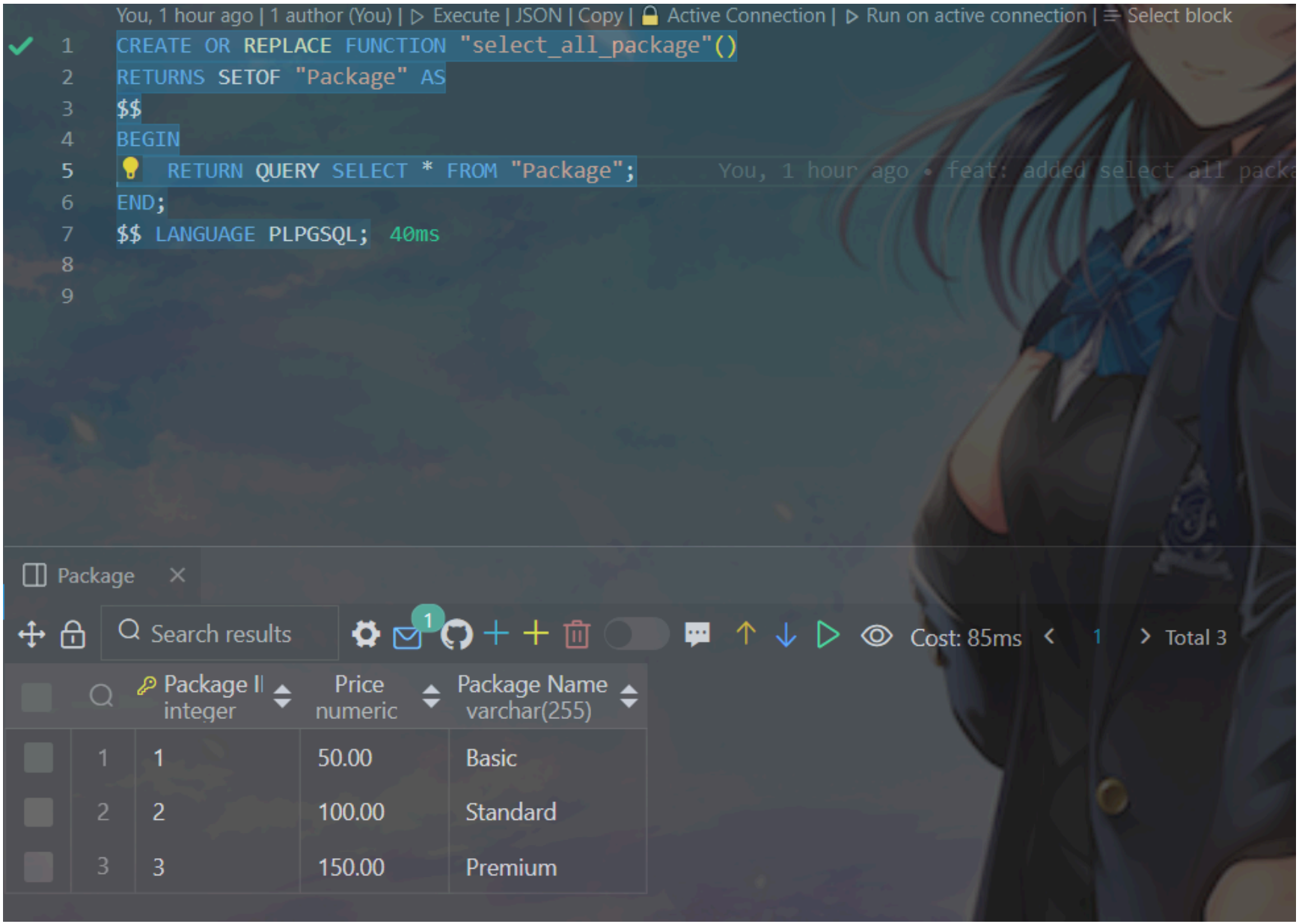
After:



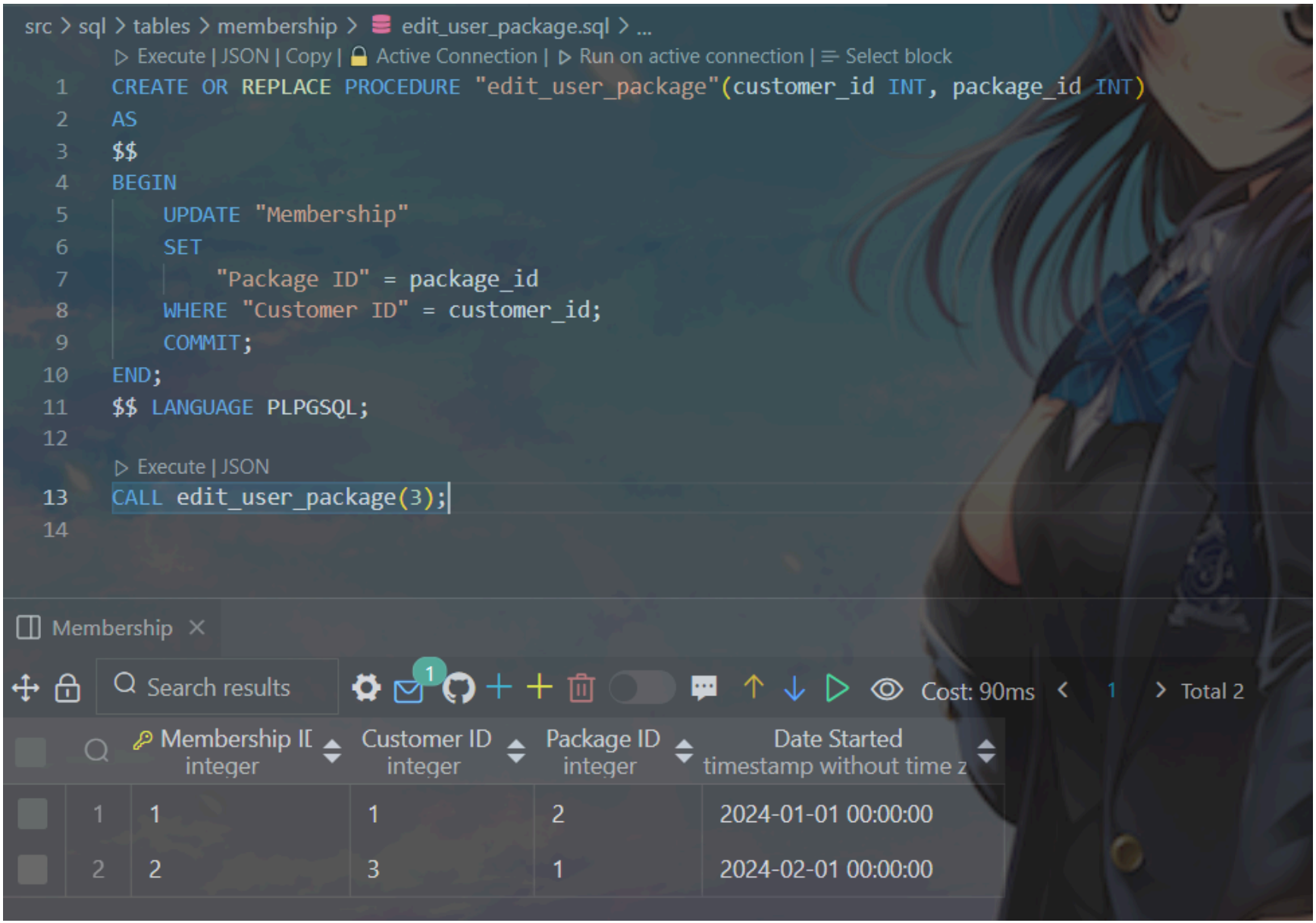
Select Package


SELECT A PACKAGE
Package A
Package B
Package C
Package D
PROCEED

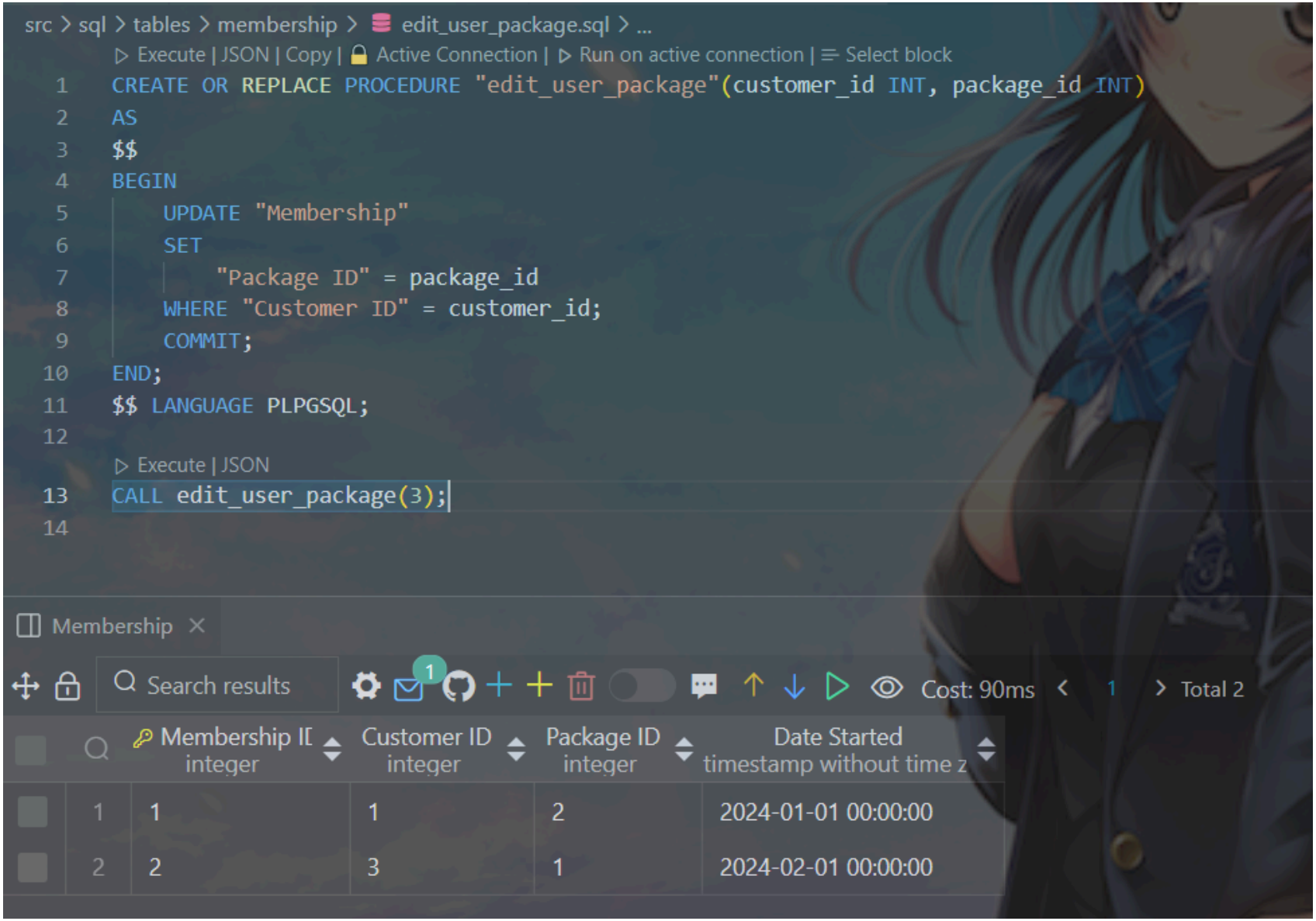
Code:


You, 1 hour ago | 1 author (You) | Execute | JSON | Copy | Active Connection | Run on active connection | Select block
CREATE OR REPLACE FUNCTION "select_all_package"()
RETURNS SETOF "Package" AS
$$
BEGIN
RETURN QUERY SELECT * FROM "Package";
END;
$$ LANGUAGE PLPGSQL; 40ms
You, 1 hour ago • feat: added select all packa
Package
Search results
Cost: 85ms
1
Total 3
Package Il integer
Price numeric
Package Name varchar(255)
1 1 50.00 Basic
2 2 100.00 Standard
3 3 150.00 Premium

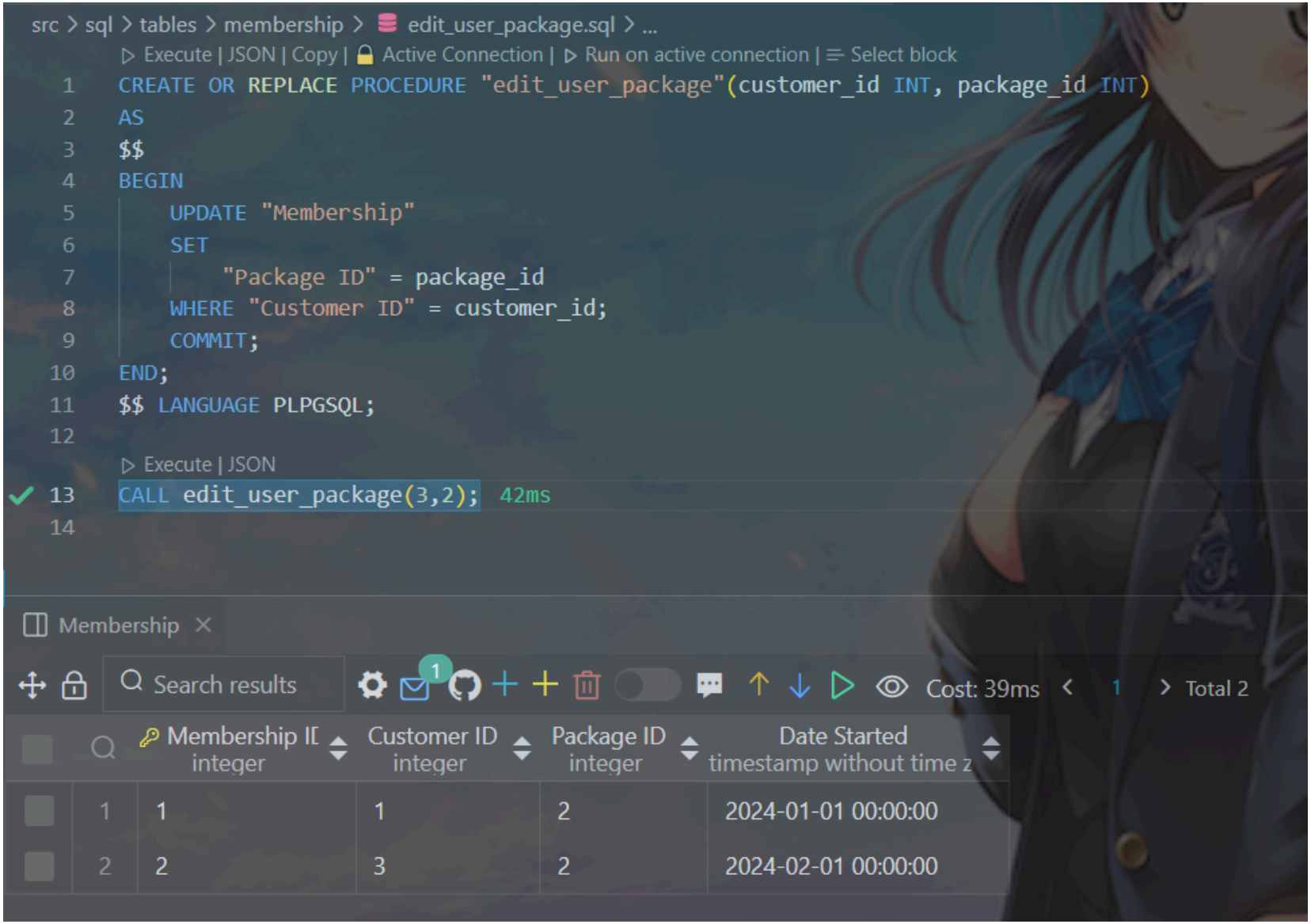
Code


src > sql > tables > membership > edit_user_package.sql > ...
▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
1 CREATE OR REPLACE PROCEDURE "edit_user_package"(customer_id INT, package_id INT)
2 AS
3 $$
4 BEGIN
5 UPDATE "Membership"
6 SET
7 "Package ID" = package_id
8 WHERE "Customer ID" = customer_id;
9 COMMIT;
10 END;
11 $$ LANGUAGE PLPGSQL;
12
▷ Execute | JSON
13 CALL edit_user_package(3);
14
Membership
Search results
Cost: 90ms
1
Total 2
Membership ID integer
Customer ID integer
Package ID integer
Date Started timestamp without time z
1 1 1 2 2024-01-01 00:00:00
2 2 3 1 2024-02-01 00:00:00

Before:

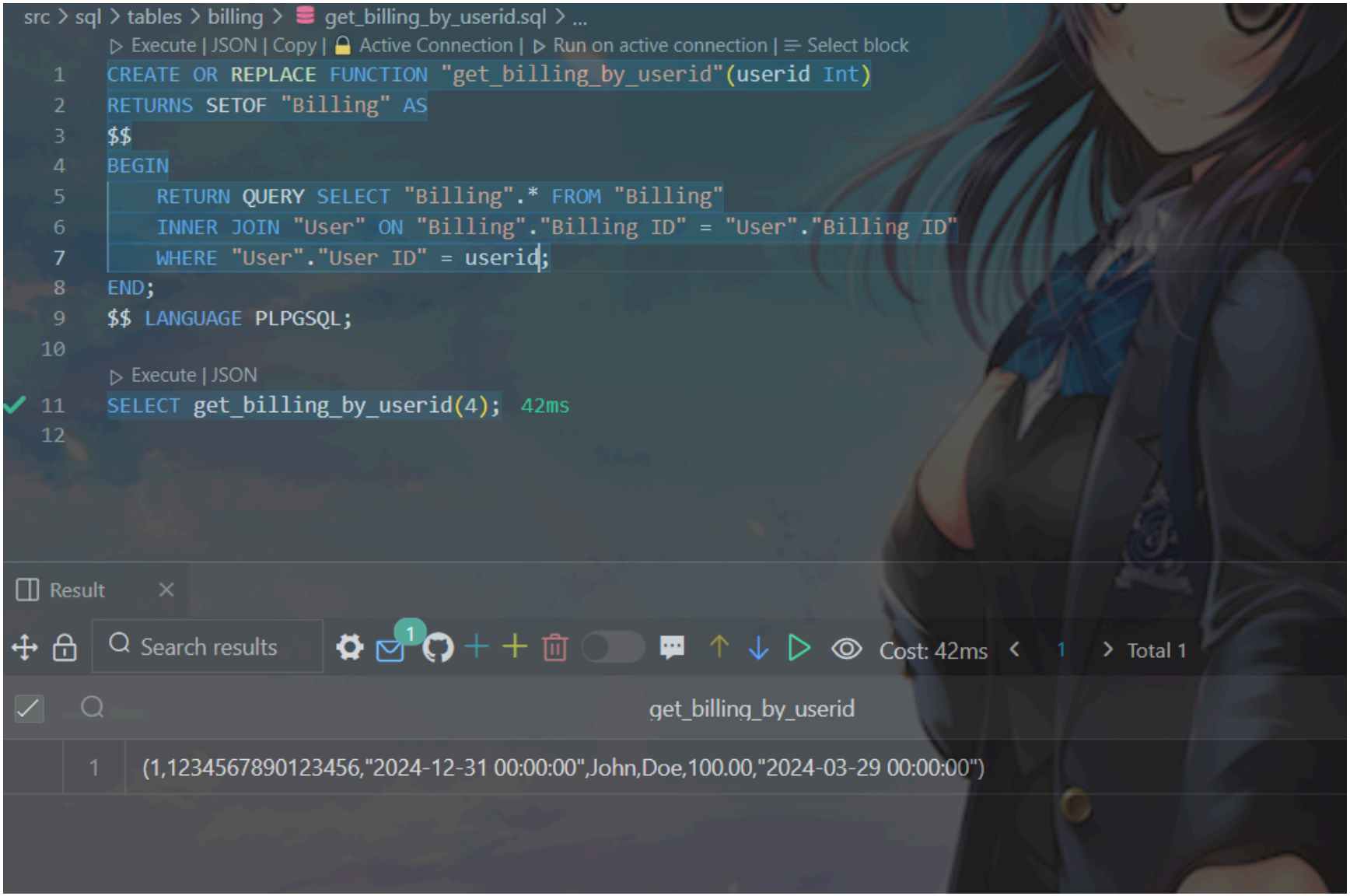

src > sql > tables > membership > edit_user_package.sql > ...
▷ Execute | JSON | Copy | Active Connection | ▷ Run on active connection | Select block
1 CREATE OR REPLACE PROCEDURE "edit_user_package"(customer_id INT, package_id INT)
2 AS
3 $$
4 BEGIN
5 UPDATE "Membership"
6 SET
7 "Package ID" = package_id
8 WHERE "Customer ID" = customer_id;
9 COMMIT;
10 END;
11 $$ LANGUAGE PLPGSQL;
12
▷ Execute | JSON
13 CALL edit_user_package(3);
14
Membership
Search results
Cost: 90ms
1
Total 2
Membership ID integer
Customer ID integer
Package ID integer
Date Started timestamp without time z
1 1 1 2 2024-01-01 00:00:00
2 2 3 1 2024-02-01 00:00:00

After:



View User's Billing Details:

Codes And Results:



Summary

In this project, we designed a database to support a fitness center with multiple branches. The database includes tables for managing users, roles, memberships, packages, billing information, equipment, branches, staff, equipment types, and equipment companies. We also provided SQL scripts to create these tables and dummy data to populate them.

The database design allows for the management of customers, staff, equipment, and branches across multiple locations. Each branch can have its own staff members, equipment inventory, and customer base. Users can be assigned different roles, such as admin, manager, or employee, to manage various aspects of the fitness center.

To improve the design, we could consider adding more constraints and validations to ensure data integrity. Additionally, implementing more sophisticated authorization and authentication mechanisms would enhance security. Future improvements could also involve optimizing queries for better performance and scalability. The UI was not designed beautifully. There are definitely things that can be improved in that department.

Overall, the project demonstrates the process of designing a database for a multi-branch fitness center and provides a foundation for building a corresponding application. Collaboration through GitHub allowed team members to contribute effectively to the project.