| Topic | UNION operator | |
|---|---|---|
| **Class Description** | **Students will learn about UNION statements in SQL** | |
| **Class** | **C-233** | |
| **Class time** | **45 mins** | |
| **Goal** | • Understand about UNION SQL Statements<br>• Importance of Data Types in columns | |
| **Resources Required** | • Teacher Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Visual Studio Code<br><br>• Student Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Visual Studio Code | |
| **Class structure** | **Warm-Up**<br>**Teacher-led Activity 1**<br>**Student-led Activity 1**<br>**Wrap-Up** | **10 mins**<br>**20 mins**<br>**10 mins**<br>**5 mins** |

| WARM-UP SESSION - 10mins | |
|---|---|
| **Teacher Action** | **Student Action** |
| *Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?* | **ESR**: Hi, thanks, yes, I am excited about it! |

| | |
|---|---|
| In the last session, we learned about JOIN statements and saw how they can be used to join the columns of 2 tables together, by finding a relation between the tables. We also saw and learned about different types of JOIN statements.<br><br>Any doubts from the last session?<br><br>*The teacher clarifies doubts (if any)*<br><br>*So what do you think now we all know about SQL statements or still we need to learn more about this!*<br><br>*Yes,*<br><br>*Still, many things need to be learned so today, we are going to learn about UNION statements.*<br><br>*Let's get started* | **ESR**:<br>Varied! |
| **Q&A Session** | |
| **Question** | **Answer** |
| How is INNER JOIN different from OUTER JOIN?<br><br>A. Inner join joins only that data that satisfies the condition in both the tables while outer join merges all the data regardless of the condition<br>B. Outer join joins only that data that satisfies the condition in both the tables while inner join merges all the data regardless of the condition<br>C. Inner join takes all the data from the first table and only that data that satisfies the condition in the second table<br>D. Inner join takes all the data from the second table and only that data that satisfies the condition in the | **A** |

| | |
|---|---|
| first table | |
| How would you join 2 tables if you want to get all the data from the first table and only some data based on the condition from the second table?<br><br>A. first_table RIGHT JOIN second_table<br>B. first_table OUTER JOIN second_table<br>C. first_table LEFT JOIN second_table<br>D. first_table INNER JOIN second_table | **C** |
| **TEACHER-LED ACTIVITY - 20mins** | |
| **Teacher Initiates Screen Share** | |
| **ACTIVITY**<br><br>• **UNION statements**<br>• **Importance of column data types in UNION statements** | |
| **Teacher Action** | **Student Action** |
| In the last class, we learned about join statements in SQL. We saw the differences b/w different types of joins.<br><br>Today, we are going to learn about the Union statements.<br><br>Before we deep dive into what a union statement is, what do you think to join statements do?<br><br>How did they join it together?<br><br>We can say that it looks for the condition to join, and also for the relation b/w the two columns but what is ultimately | **ESR:**<br>They joined 2 tables together<br><br>**ESR:**<br>Varied |

| | |
|---|---|
| happening here, is that it is joining the columns of 2 tables, and displaying it as a third table.<br><br>That's how it works, right?<br><br>It simply takes a few columns from the first table, a few columns from the second table, and displays the newly constructed table.<br><br>Now, what if, instead of columns, you wanted to join the rows. What would we do then?<br><br>For these cases, we have a new type of SQL statement we will be learning about today, called the UNION statements.<br><br>By definition, the **UNION** operator is used to combine the result-set of two or more **SELECT** statements. It just takes all the rows of the 2 tables and displays them as one.<br><br>There are, however, a few conditions that the **UNION operators** require in order to work. They are as follows -<br><br>1. Every **SELECT** statement within **UNION** must have the same number of columns.<br>2. The columns must all have similar data types.<br>3. The column in every **SELECT** statement must also be in the same order.<br><br>We will understand these later when we learn how to use the union operator in this class.<br><br>To summarize the union operator, here are the differences between the union operator and join statements - | **ESR:**<br>Yes!<br><br>**ESR:**<br>Varied! |

| JOIN | UNION |
|---|---|
| JOIN combines data from many tables based on the condition | UNION combines the result of two or more SELECT statements |
| It combines data into new columns | It combines data into new rows |
| The number of columns selected from each table can be different | The number of columns selected from different tables should be the same |
| Datatypes of corresponding columns in different tables can be different | Datatypes of corresponding columns in different tables should be the the same |

Now, let's

Since it just joins the result of multiple select statements, its syntax looks like this -

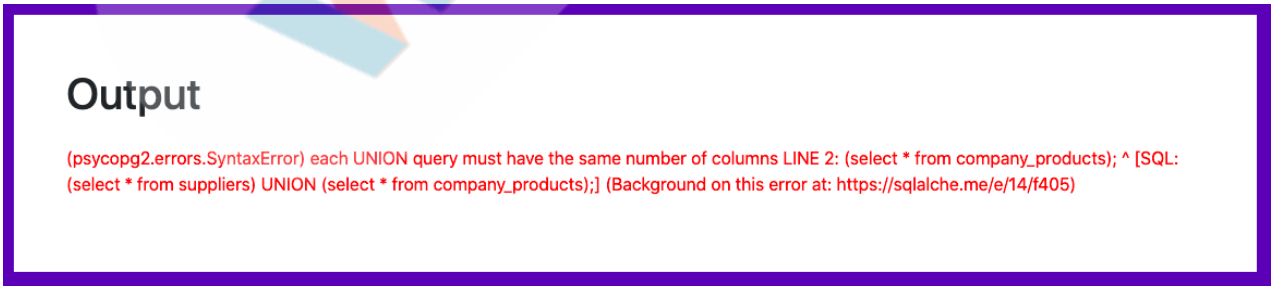
```
(select * from table_1)
UNION
(select * from table_2)
UNION
(select * from table_3);
```

From this, we can observe that the select statements are wrapped in parentheses () and a union operator is used to join them together between select statements.

| | |
|---|---|
| Let's give it a try!<br><br>For that, let's open our SQL Editor from Student Activity 1<br><br>*Teacher refers to Teacher Activity 1* | *Student refers to Student Activity 1* |
| Let's try to merge the data of table "suppliers" and table "company_products" -<br><br>*Teacher executes the query on the editor* | *Student observes* |

(SELECT * FROM suppliers) UNION (SELECT * FROM company_products);


**Output -**



| | |
|---|---|
| We can see that we have received an error. If we read it carefully, it says -<br><br>***"Each UNION query must have the same number of columns"***<br><br>The UNION operator needs 3 conditions satisfied between the select statements to work. They are as follows -<br><br>1. Every **SELECT** statement within **UNION** must have the same number of columns.<br>2. The columns must all have similar data types.<br>3. The column in every **SELECT** statement must also be in the same order.<br><br>Before we understand these statements, let's understand the requirement of a union operator with an example of an eCommerce company.<br><br>If this e-commerce company operates on a big scale, it will have different databases for different regions. It will have different databases for different countries, and different databases of different states and cities in it.<br><br>Suppose, you wanted to query the data of India's all metropolitan cities. These cities would be -<br><br>1. NCR (National Capital Region)<br>2. Mumbai<br>3. Kolkata<br>4. Bangalore<br><br>Now, since these cities would be having their own databases, with of course similar data, would it make more sense to merge the data in columns or in rows in this case?<br><br>Now to merge this data in rows, the UNION operator is used. | **ESR:**<br>In rows |

| | |
|---|---|
| Now, let's understand the 3 conditions -<br><br>***First -***<br>Since the union operator merges the rows of the 2 tables, it needs to make sure that the number of columns are the same. It cannot merge 2 columns of one table with 3 columns of the other table. Therefore, the union operator requires the same number of columns in all the select statements.<br><br>***Second -***<br>Suppose that there are 2 tables having the same number of columns, but the datatypes of these columns are different, then would it make sense to merge the data together through union?<br><br>Exactly! We wouldn't want to see strings in a column that should show numbers, or vice versa, since it will confuse us. Therefore, the data types of the columns should be the same.<br><br>***Third -***<br>The columns in the select statement should be in the same order. Union operator works in a way where it will merge the first column of the first table with the first column of the second table, so on and so forth. Therefore we should always keep in mind to keep our columns aligned in the select statements.<br><br>For example, if you wanted to get the *first_name* and *phone* from ***customers***, and *contact_name* and *phone* from ***suppliers*** - then the resulting query would be?<br><br>*Teacher discusses the query with the student and tries constructing it* | **ESR:**<br>No<br><br>**ESR:**<br>*Student helps the teacher in constructing the query* |
| (SELECT first_name, phone from customers) UNION (SELECT contact_name, phone from suppliers); | |

```
1 (SELECT first_name, phone from customers) UNION (SELECT contact_name, phone from suppliers);
```

Output -


Here, we can observe that since we asked for

1. 2 columns from both the tables
2. All columns of String data type
3. Incorrect order with the name first and contact info

second

We got the resulting output with 2 columns and the data merged together in the form of rows.

Sometimes, there are duplicates. For example, what if there was a customer, who was also a supplier?

For that, we have a ***UNION ALL*** operator. The key difference between ***UNION*** and ***UNION ALL*** is that ***UNION removes the duplicates*** whereas ***UNION ALL keeps the duplicates***.

For example, suppose we want to get the data for all the cities and countries from both customers and suppliers tables. What would be the SQL query for that?

*Teacher discusses the query with the student and tries constructing it*

**ESR:**
*Student helps the teacher in constructing the query*

(SELECT city, country FROM customers)
UNION ALL
(SELECT city, country FROM suppliers);


Output -


| | |
|---|---|
| Here, we have received the data as expected. We can see that we have around 120 entries as output. If we instead use UNION, we would see 94 entries. This means that we have some duplicates.<br><br>Let's say that you want to find out these duplicates, and not from this data, but some other data with thousands of rows. Would you be willing to find that duplicate?<br><br>To find such duplicates, there is another operator called ***INTERSECT***.<br><br>Let's see it in action and replace our ***UNION ALL*** operator in the above query with ***INTERSECT*** -<br><br>*Teacher executes the query* | **ESR:**<br>*No!*<br><br>*Student observes* |

(SELECT city, country FROM customers)
INTERSECT
(SELECT city, country FROM suppliers);

**Executing the above query on the Editor gives the following output -**


Here, we get a table of data that was duplicated in the tables.

Since ***UNION, UNION ALL,*** and ***INTERSECT*** operators are all used together most of the time, they are known as ***SET Operators***.

The 3 conditions for UNION operators also apply to the UNION ALL and INTERSECT operators.

**STUDENT-LED ACTIVITY - 10 mins**

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

**ACTIVITY**

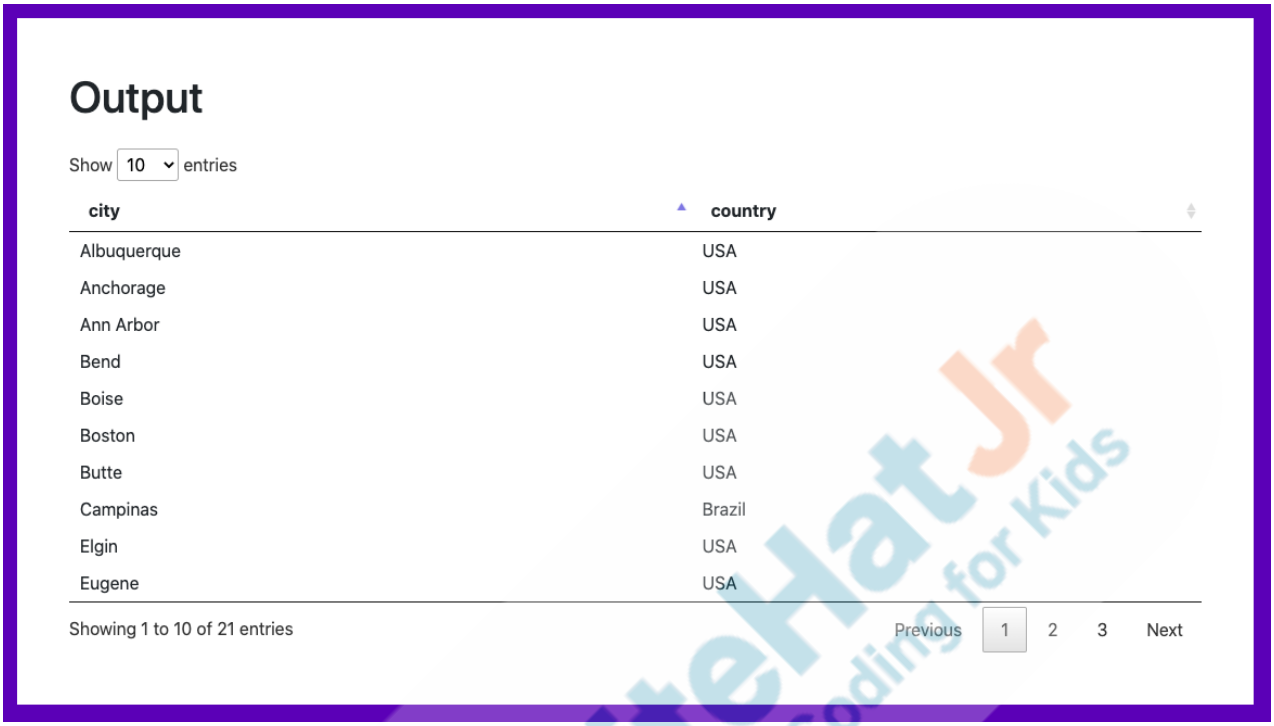
- **Practice SET Operators**
- **Tweaking set operator rules**

| Teacher Action | Student Action |
|---|---|
| We now understand what the set operators are. Let's use that knowledge to search for some data!<br><br>Let's say that you want to fetch the names of cities in the USA and Brazil from customers and supplies, how would you do that?<br><br>*Teacher guides the student in constructing the query* | **ESR:**<br>*Student opens the editor from Student Activity 1 and constructs the query* |

(SELECT city, country FROM customers WHERE country='USA' OR country='Brazil')
UNION
(SELECT city, country FROM suppliers WHERE country='USA' OR country='Brazil');

```
1 (SELECT city, country FROM customers WHERE country='USA' OR country='Brazil')
2 UNION
3 (SELECT city, country FROM suppliers WHERE country='USA' OR country='Brazil');
```

Output -




| | |
|---|---|
| Great! SET operators are simple, right? | **ESR:** Yes! |
| Now, let's say that you wanted to find out about the amount and the customer ID from the ***company_orders*** table, and just the customer ID from the ***customer's*** table. Can you do that? | **ESR:** We don't have the same number of columns to get from both the tables. |
| That's right, but there is still a way in which this data can be fetched. Give it a try!<br><br>*The teacher lets the student fiddle with the statement for a while.* | *The student tries to fetch the data* |

| | |
|---|---|
| The best way to do this, is to use column names as variables representing the data of the row.<br><br>Instead of using the variable, can we use a constant value?<br><br>Let's give it a try!<br><br>*Teacher guides the student to execute the query* | **ESR:**<br>Yes!<br><br>*Student executes the query* |

(SELECT customer_id, total_amount FROM company_orders) UNION (SELECT id, 0 from customers);

```
(SELECT customer_id, total_amount FROM company_orders)
UNION
(SELECT id, 0 from customers);
```

Gives the Output -

## Output

Show 10 entries

| customer_id | total_amount |
|---|---|
| 1 | 878 |
| 1 | 491.2 |
| 1 | 960 |
| 1 | 851 |
| 1 | 330 |
| 1 | 0 |
| 1 | 1086 |
| 2 | 514.4 |
| 2 | 479.75 |
| 2 | 0 |

Showing 1 to 10 of 920 entries Previous 1 2 3 4 5 ... 92 Next

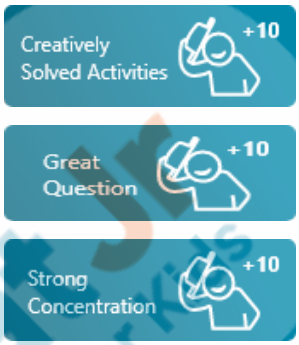
Here, we could see the user through their customer_id, and see which users have what number of spending on the website.

If you look at the query, we have, instead of using the name of the column in the second select statement, are using a constant value 0. This way, it just places 0 in place of that column wherever it has fetched data and filled the row with the customer's data.

With this technique, we can manipulate the rules and conditions of the set operators.

In the next class, we will see how we can combine the JOIN with UNION operators and perform a SQL injection on the website, attempting to fetch their user's sensitive data.

| Teacher Guides Student to Stop Screen Share | |
|---|---|
| **WRAP UP SESSION - 5 Mins** | |
| **Quiz time - Click on in-class quiz** | |
| **Question** | **Answer** |
| Which of these operators does not remove the duplicates?<br><br>A. UNION ALL<br>B. UNION<br>C. SET<br>D. INTERSECT | **A** |
| Which of these is a condition for set operators?<br><br>A. There should be same number of rows in the tables<br>B. There should be same number of columns in the tables<br>C. There should be different number of rows in the tables<br>D. There should be different number of columns in the tables | **B** |
| Which operator would you use to get all the duplicates in the 2 tables?<br><br>A. UNION ALL<br>B. UNION<br>C. INTERSECT<br>D. SET | **C** |
| **End the quiz panel** | |
| **FEEDBACK**<br>• **Appreciate the students for their efforts in the class.**<br>• **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.** | |

| Teacher Action | Student Action |
|---|---|
| You get Hats off for your excellent work!<br><br>In the next class we will learn about SQL Union | *Make sure you have given at least 2 Hats Off during the class for:*<br>  |
| **Project Discussion**<br><br>You were approached by a friend, who is trying to learn MySQL and is stuck on trying to find answers to simple questions like getting all the users who are from a particular state, or which neighborhood has the most number of users.<br><br>Your task is to help your friend in trying to find these data attributes. | |
| **Teacher Clicks** ✖ End Class | |
| **ADDITIONAL ACTIVITIES** | |
| **Additional Activities**<br>*Encourage the student to write reflection notes in their reflection journal using markdown.* | *The student uses the markdown editor to write* |

| | |
|---|---|
| Use these as guiding questions:<br>● What happened today?<br>○ Describe what happened.<br>○ The code I wrote.<br>● How did I feel after the class?<br>● What have I learned about programming and developing games?<br>● What aspects of the class helped me? What did I find difficult? | *her/his reflections in the reflection journal.* |

| ACTIVITY LINKS | | |
|---|---|---|
| **Activity Name** | **Description** | **Link** |
| Teacher Activity1 | SQL Editor | http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/editor |
| Student Activity 1 | SQL Editor | http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/editor |