Applying filters to SQL queries

Project description

In this project, I use SQL to manage and retrieve data from a database efficiently. By writing SQL queries, I can filter, sort, and extract specific information based on various conditions, helping to organize and analyze data effectively. For example, I retrieve records for employees in certain departments, exclude records based on conditions, and perform other data manipulations to support decision-making and reporting needs within an organization.

Through this project, I demonstrate how SQL can be used to interact with databases, streamline data management tasks, and generate meaningful insights.

Retrieve after hours failed login attempts

This query retrieves records of all failed login attempts that occurred after regular business hours, specifically after 6:00 PM.

- **SELECT ***: Selects all columns from the log_in_attempts table, providing full details for each record that matches the criteria.
- **FROM log_in_attempts**: Specifies the table to query, which is named log_in_attempts.
- **WHERE login_time > '18:00' AND success = 0**: Filters the records to include only those where login_time is later than 6:00 PM (18:00) and where the success field is equal to 0, indicating a failed login attempt.

This query helps identify potentially suspicious activity by isolating login failures that happen outside of regular hours, which could indicate unauthorized access attempts.

Retrieve login attempts on specific dates

This query retrieves all login attempts that occurred on **May 8, 2022**, or **May 9, 2022**, to help investigate a suspicious event on May 9.

- **SELECT ***: Selects all columns from the log_in_attempts table, so we can view the full details of each login attempt.

- **FROM log_in_attempts**: Specifies the table to query, which is named log_in_attempts.
- **WHERE login_date = '2022-05-09' OR login_date = '2022-05-08'**: Filters the results to include only records where the login_date matches either May 8 or May 9, 2022.

This query allows a focused review of login activity around the date of the suspicious event, which may reveal patterns or unauthorized attempts relevant to the investigation.

Retrieve login attempts outside of Mexico

This query retrieves all login attempts that occurred outside of Mexico by excluding entries where the country is either **MEX** or **MEXICO**.

- **SELECT ***: Selects all columns from the log_in_attempts table, allowing for a full view of each login attempt's details.
- **FROM log_in_attempts**: Specifies the table being queried, which is log_in_attempts.
- **WHERE country NOT LIKE 'MEX%'**: Uses the NOT LIKE condition with MEX% to exclude any rows where the country field starts with "MEX" (including both "MEX" and "MEXICO").

This query ensures that only login attempts from outside of Mexico are displayed, helping the team focus on activity that may be more relevant to the investigation.

Retrieve employees in Marketing

This query retrieves information on all employees in the **Marketing department** located in the **East building**.

- **SELECT ***: Selects all columns from the employees table, so that detailed information for each employee is shown.
- **FROM employees**: Specifies the table being queried, which is employees.
- **WHERE department LIKE '%Marketing%'**: Filters the results to include only employees in departments that contain "Marketing" in their names, allowing for any variations like "Digital Marketing" or "Content Marketing."
- **AND office LIKE 'East%'**: Further filters the results to include only those employees whose office location starts with "East," capturing rooms such as East-170, East-320, etc.

This query provides a list of all employees in the Marketing department whose offices are in the East building, allowing the team to identify machines that need security updates.

Retrieve employees in Finance or Sales

This query retrieves information on all employees in the **Sales** or **Finance** departments, regardless of their specific office location.

- **SELECT ***: Selects all columns from the employees table, providing full details of each employee.
- **FROM employees**: Specifies the table being queried, which is employees.
- **WHERE department LIKE '%Sales%' OR department LIKE '%Finance%'**: Filters the results to include only employees whose department contains "Sales" or "Finance." Using LIKE '%Sales%' and LIKE '%Finance%' ensures that any department variations, such as "Corporate Sales" or "Finance & Accounting," are captured.

This query provides the team with a list of employees in the Sales or Finance departments, allowing them to focus on machines that need this specific security update.

Retrieve all employees not in IT

This query retrieves all employees who are **not** in the Information Technology department, ensuring that only those who still need the update are included.

- **SELECT ***: Selects all columns from the employees table, showing complete details for each employee.
- **FROM employees**: Specifies the table being queried, which is employees.
- **WHERE department NOT LIKE '%Information Technology%'**: Filters the results to exclude any employees whose department contains "Information Technology." Using NOT LIKE '%Information Technology%' ensures that any variations, such as "Global Information Technology," are also excluded.

This query allows the team to identify employees in all departments other than IT, targeting only those employees who still require the security update.

Summary

In this project, I used SQL queries to filter employee data for specific security updates. I retrieved records based on department, office location, and login activity to help the team identify which employees needed updates. I used various SQL filtering techniques, including the LIKE and NOT LIKE operators, to match specific department names and office locations. These queries allowed the team to focus on relevant employee groups, such as those outside of IT or those in the Sales and Finance departments, ensuring accurate and efficient targeting for security tasks. This approach showcases SQL's power in data management and selective data retrieval for cybersecurity needs.