Apply filters to SQL queries

Project description

Este proyecto se centra en la aplicación de consultas SQL para la **investigación y detección de posibles problemas de seguridad** dentro de una organización. A través del análisis de las bases de datos **employees** y **log_in_attempts**, se desarrollaron consultas para identificar **intentos de inicio de sesión sospechosos**, **patrones de acceso inusuales** y **empleados que requieren actualizaciones de seguridad en sus dispositivos**.

Las consultas SQL implementadas permiten:
✅ **Detectar intentos de acceso fallidos fuera del horario laboral.**
✅ **Identificar empleados que han iniciado sesión desde ubicaciones inusuales.**
✅ **Analizar intentos de acceso desde múltiples direcciones IP en un corto período de tiempo.**
✅ **Listar empleados de departamentos específicos para aplicar medidas de seguridad.**
✅ **Filtrar empleados que aún requieren actualizaciones en sus dispositivos.**

Este trabajo demuestra **habilidades clave en seguridad informática, análisis de datos y optimización de consultas SQL**, proporcionando información crítica para la protección de la infraestructura de TI de una empresa. Además, se incluyen recomendaciones de seguridad para mitigar posibles amenazas, asegurando un enfoque proactivo en la gestión de accesos y dispositivos corporativos. 🚀

Retrieve after hours failed login attempts

Consulta SQL para identificar intentos de inicio de sesión fallidos después del horario laboral:

```
SELECT event_id, username, login_date, login_time, country,
ip_address, success
FROM log_in_attempts
WHERE login_time > '18:00:00'
AND success = 0;
```

Explicación de la consulta:

1. **`SELECT event_id, username, login_date, login_time, country, ip_address, success`**
    - Extraemos las columnas relevantes de la tabla `log_in_attempts` para analizar los intentos fallidos.
2. **`FROM log_in_attempts`**
    - Especificamos la tabla que contiene los registros de intentos de inicio de sesión.
3. **`WHERE login_time > '18:00:00'`**
    - Filtramos los registros donde la hora del intento de inicio de sesión (`login_time`) es después de las 18:00 (6:00 PM).
4. **`AND success = 0`**
    - La columna `success` es de tipo `TINYINT(1)`, lo que significa que puede almacenar valores `0` (fallido) o `1` (exitoso).
    - Filtramos solo los intentos de inicio de sesión que fallaron (`success = 0`).

Posibles Mejoras y Análisis Adicional:

Ordenar los resultados para priorizar la investigación:

```
ORDER BY login_date DESC, login_time DESC;
```

- Esto permitirá ver primero los intentos más recientes.

Investigar intentos desde países inusuales:

```
WHERE login_time > '18:00:00'
AND success = 0
AND country NOT IN ('USA', 'Mexico', 'Canada');
```

- Esto ayuda a detectar intentos desde ubicaciones sospechosas.

Cruzar datos con la tabla `employees` para ver el departamento del usuario:

```
SELECT l.event_id, l.username, e.department, l.login_date,
l.login_time, l.country, l.ip_address
FROM log_in_attempts l
JOIN employees e ON l.username = e.username
WHERE l.login_time > '18:00:00'
```

```
AND l.success = 0;
```

- Esto permitirá identificar si ciertos departamentos están siendo atacados con más frecuencia.

Conclusión

Esta consulta ayuda a detectar actividad sospechosa fuera del horario laboral, lo que podría indicar intentos de acceso no autorizados. Si se identifican múltiples intentos fallidos para el mismo usuario o desde una ubicación desconocida, sería recomendable investigar más a fondo y posiblemente bloquear las direcciones IP sospechosas.

🚨 **Acción recomendada:**
Si la consulta devuelve múltiples intentos fallidos para un mismo usuario desde distintas ubicaciones, es posible que su cuenta esté siendo objeto de un **ataque de fuerza bruta**. Se recomienda **bloquear temporalmente la cuenta y notificar al usuario afectado**.

Retrieve login attempts on specific dates

Para investigar los intentos de inicio de sesión que ocurrieron los días **2022-05-09** y **2022-05-08**, podemos utilizar la cláusula `WHERE` en SQL para filtrar los registros por la columna `login_date`.

Consulta SQL:

```
SELECT event_id, username, login_date, login_time, country,
ip_address, success
FROM log_in_attempts
WHERE login_date IN ('2022-05-09', '2022-05-08');
```

Explicación de la consulta:

1. **`SELECT event_id, username, login_date, login_time, country, ip_address, success`**
    - Extraemos los datos más relevantes de la tabla `log_in_attempts` para el análisis.

2. `FROM log_in_attempts`
    - Especificamos la tabla de la que se tomarán los datos.
3. `WHERE login_date IN ('2022-05-09', '2022-05-08')`
    - Filtramos los registros para obtener solo aquellos intentos de inicio de sesión que ocurrieron en las fechas **2022-05-09** y **2022-05-08**.
    - La cláusula `IN` nos permite especificar múltiples valores de fecha sin necesidad de usar múltiples condiciones `OR`.

Posibles Mejoras y Extensiones:

Ordenar los resultados para facilitar el análisis:

```
ORDER BY login_date DESC, login_time DESC;
```

- Esto nos ayuda a ver primero los intentos más recientes.

Incluir solo intentos fallidos para detectar posibles ataques:

```
WHERE login_date IN ('2022-05-09', '2022-05-08')
AND success = 0;
```

- Esto filtrará solo los intentos fallidos (`success = 0`), lo cual es útil para detectar intentos de acceso no autorizados.

Ampliar el rango de fechas si es necesario:
Si queremos incluir más días antes o después, podemos usar `BETWEEN` en lugar de `IN`:

```
WHERE login_date BETWEEN '2022-05-07' AND '2022-05-09';
```

-

Conclusión

Esta consulta permite investigar actividad sospechosa en fechas específicas. Si se detectan múltiples intentos de inicio de sesión fallidos desde ubicaciones inusuales, esto podría indicar un posible **ataque de fuerza bruta o acceso no autorizado**, por lo que se recomienda realizar un análisis más detallado de las IPs y los usuarios involucrados. 🚨

Retrieve login attempts outside of Mexico

Para investigar los intentos de inicio de sesión que **no se originaron en México**, utilizaremos la cláusula `WHERE` con el operador `<>` o `NOT IN` para excluir a México de los resultados.

Consulta SQL:

```
SELECT event_id, username, login_date, login_time, country,
ip_address, success
FROM log_in_attempts
WHERE country <> 'Mexico';
```

Explicación de la consulta:

1. **`SELECT event_id, username, login_date, login_time, country, ip_address, success`**
   - Seleccionamos las columnas más relevantes de la tabla `log_in_attempts` para analizar los registros sospechosos.
2. **`FROM log_in_attempts`**
   - Especificamos la tabla de donde extraemos los datos.
3. **`WHERE country <> 'Mexico'`**
   - Filtramos los registros para excluir los intentos de inicio de sesión que provienen de México (`<>` significa "distinto de").
   - Solo se mostrarán los intentos de inicio de sesión que ocurrieron en otros países.

Alternativa con `NOT IN` (si hay más países a excluir)

Si en el futuro queremos excluir más países (por ejemplo, México y EE.UU.), podemos usar `NOT IN`:

```
WHERE country NOT IN ('Mexico', 'USA');
```

Posibles Mejoras y Extensiones:

Ordenar los intentos de inicio de sesión por fecha y hora para priorizar la investigación:

```
ORDER BY login_date DESC, login_time DESC;
```

-

Incluir solo intentos fallidos para detectar posibles ataques:

```
WHERE country <> 'Mexico'
AND success = 0;
```

- Esto filtrará solo los intentos fallidos (`success = 0`), lo cual es útil para detectar accesos no autorizados.

Ampliar la consulta para ver intentos en un rango de fechas específico:

```
WHERE country <> 'Mexico'
AND login_date BETWEEN '2024-02-01' AND '2024-02-29';
```

-

Cruzar datos con la tabla `employees` para obtener información adicional sobre los usuarios:

```
SELECT l.event_id, l.username, e.department, l.login_date,
l.login_time, l.country, l.ip_address
FROM log_in_attempts l
JOIN employees e ON l.username = e.username
WHERE l.country <> 'Mexico';
```

- Esto ayuda a identificar si ciertos departamentos están siendo objetivo de intentos de acceso desde el extranjero.

Conclusión

Esta consulta nos permitirá detectar intentos de inicio de sesión **fuera de México**, lo que puede indicar actividad sospechosa si los empleados normalmente inician sesión solo desde este país. Si se identifican intentos recurrentes desde ubicaciones desconocidas, puede ser señal de un **ataque de fuerza bruta o de credenciales comprometidas**, lo que requeriría tomar medidas como **bloquear las IPs sospechosas o forzar cambios de contraseñas**. 🚨

Retrieve employees in Marketing

Para obtener información sobre los empleados del departamento de **Marketing** que trabajan en cualquier oficina del **edificio Este**, podemos utilizar la cláusula `WHERE` con las condiciones adecuadas en SQL.

Consulta SQL:

```
SELECT employee_id, device_id, username, department, office
FROM employees
WHERE department = 'Marketing'
AND office LIKE 'Este-%';
```

Explicación de la consulta:

1. **`SELECT employee_id, device_id, username, department, office`**
   - Seleccionamos las columnas relevantes de la tabla `employees` para identificar a los empleados de Marketing y sus dispositivos.
2. **`FROM employees`**
   - Especificamos la tabla de donde se tomarán los datos.
3. **`WHERE department = 'Marketing'`**
   - Filtramos para obtener solo los empleados cuyo departamento es "Marketing".
4. **`AND office LIKE 'Este-%'`**
   - La columna `office` contiene valores como **"Este-170", "Este-320"**, etc.
   - `LIKE 'Este-%'` selecciona todas las oficinas que **comienzan con "Este-"**, asegurando que solo obtenemos los empleados del **edificio Este**, sin importar el número de oficina.

Posibles Mejoras y Extensiones:

Ordenar los resultados para facilitar la revisión:

```
ORDER BY office, username;
```

- Esto organizará la lista por oficina y por nombre de usuario.

Cruzar datos con la tabla `machines` para obtener detalles sobre los dispositivos de estos empleados:

```
SELECT e.employee_id, e.username, e.department, e.office,
m.device_id, m.operating_system, m.os_patch_date
FROM employees e
JOIN machines m ON e.device_id = m.device_id
WHERE e.department = 'Marketing'
AND e.office LIKE 'Este-%';
```

- Esto nos permitirá obtener información adicional sobre los sistemas operativos y parches de seguridad de los equipos asignados a estos empleados.

Conclusión

Esta consulta nos ayudará a identificar a los empleados del **departamento de Marketing en el edificio Este**, lo que es clave para coordinar actualizaciones de seguridad en sus dispositivos. Si encontramos empleados con sistemas desactualizados, podemos priorizar sus máquinas para recibir los **parches de seguridad** lo antes posible. 🚀

Retrieve employees in Finance or Sales

🔍 Consulta SQL para recuperar empleados en Finanzas o Ventas:

```
SELECT employee_id, username, department, device_id, office
FROM employees
WHERE department IN ('Ventas', 'Finanzas');
```

🛠️ Explicación de la consulta:

1. **`SELECT employee_id, username, department, device_id, office`**
   Extraemos las columnas relevantes de la tabla `employees` para identificar los empleados y sus dispositivos asignados.
2. **`FROM employees`**
   Especificamos que la información proviene de la tabla `employees`.

3. `WHERE department IN ('Ventas', 'Finanzas')`
   Filtramos los registros para incluir solo a los empleados cuyo departamento sea **Ventas** o **Finanzas**.
   - La condición `IN ('Ventas', 'Finanzas')` es equivalente a usar `department = 'Ventas' OR department = 'Finanzas'`, pero más eficiente y fácil de leer.

🔎 Posibles Mejoras y Análisis Adicional:

✅ Ordenar los resultados por oficina para facilitar la gestión:

```
ORDER BY office, username;
```

Esto agrupa los resultados por ubicación de la oficina y los ordena alfabéticamente por usuario.

✅ Verificar empleados sin un `device_id` asignado:

```
WHERE department IN ('Ventas', 'Finanzas') AND device_id IS NULL;
```

Esto ayuda a identificar empleados que no tienen un dispositivo registrado, lo cual puede ser un riesgo de seguridad.

✅ Cruzar con la tabla `log_in_attempts` para detectar accesos sospechosos:

```
SELECT e.employee_id, e.username, e.department, l.ip_address,
l.login_date, l.success
FROM employees e
JOIN log_in_attempts l ON e.username = l.username
WHERE e.department IN ('Ventas', 'Finanzas')
AND l.success = 0;
```

Esto permite ver qué empleados de estos departamentos han tenido intentos fallidos de inicio de sesión recientemente.

🚨 Conclusión

Esta consulta permite identificar a los empleados de **Ventas y Finanzas** para aplicar actualizaciones de seguridad específicas en sus dispositivos. Además, cruzar esta información con los intentos de acceso puede revelar posibles **riesgos de seguridad**.

⚠ Acción recomendada:

- Aplicar parches de seguridad en los dispositivos de estos empleados.
- Investigar intentos de inicio de sesión fallidos en estos departamentos.
- Asegurar que cada empleado tenga un **device_id** registrado en la base de datos. 🚀

Retrieve all employees not in IT

🔍 Consulta SQL para recuperar a todos los empleados que no estén en Tecnología de la Información:

```
SELECT employee_id, username, department, device_id, office
FROM employees
WHERE department <> 'Tecnología de la Información';
```

🛠️ Explicación de la consulta:

1. **`SELECT employee_id, username, department, device_id, office`**
   Extraemos las columnas clave de la tabla `employees` para identificar a los empleados y sus dispositivos asignados.
2. **`FROM employees`**
   Indicamos que la información proviene de la tabla `employees`.
3. **`WHERE department <> 'Tecnología de la Información'`**
   - Excluimos a los empleados que pertenecen al departamento de **Tecnología de la Información**.
   - El operador `<>` significa **"distinto de"**, por lo que solo se seleccionarán empleados que no estén en este departamento.

🔎 Posibles Mejoras y Análisis Adicional:

✅ Ordenar los resultados por departamento para facilitar la gestión:

```
ORDER BY department, office, username;
```

Esto agrupa los empleados por departamento y oficina, permitiendo aplicar la actualización de manera organizada.

✅ **Verificar empleados sin `device_id` asignado para evitar problemas en la actualización:**

```
WHERE department <> 'Tecnología de la Información' AND device_id IS NULL;
```

Esto ayuda a identificar empleados que no tienen un dispositivo registrado, lo cual puede requerir una revisión antes de la actualización.

✅ **Cruzar con la tabla `log_in_attempts` para verificar intentos de acceso recientes:**

```
SELECT e.employee_id, e.username, e.department, l.ip_address, l.login_date, l.success
FROM employees e
JOIN log_in_attempts l ON e.username = l.username
WHERE e.department <> 'Tecnología de la Información'
AND l.success = 0;
```

Esto permite identificar si alguno de estos empleados ha tenido **intentos fallidos de inicio de sesión**, lo que podría indicar problemas de seguridad.

🚨 Conclusión

Esta consulta permite **identificar a los empleados que aún necesitan la actualización** porque no pertenecen a Tecnología de la Información. Además, se pueden realizar análisis adicionales para detectar **posibles riesgos de seguridad antes de aplicar la actualización**.

⚠ Acción recomendada:

- Aplicar la actualización en las máquinas de estos empleados.
- Revisar empleados sin `device_id` registrado.
- Identificar intentos de acceso sospechosos antes de proceder. 🚀

Summary

Se analizaron los datos de intentos de inicio de sesión y empleados para identificar posibles riesgos de seguridad dentro de la organización. A través de consultas SQL, se detectaron accesos sospechosos, como intentos fallidos fuera del horario laboral y conexiones desde ubicaciones inusuales. También se identificaron empleados que requieren actualizaciones en sus dispositivos, asegurando que solo aquellos fuera del departamento de TI reciban los parches correspondientes. Estas consultas permiten mejorar la seguridad del sistema y optimizar la gestión de accesos y dispositivos en la empresa. 🚀