Consultas SQL Aplicadas a la Ciberseguridad

1. Subconsultas y Análisis de Patrones Complejos

Consulta para detectar usuarios con múltiples intentos fallidos desde distintas IPs en 24 horas:

```
SELECT username, COUNT(DISTINCT ip_address) AS unique_ips, COUNT(*) AS total_attempts
FROM log_in_attempts
WHERE success = 0
  AND login_date >= CURRENT_DATE - INTERVAL 1 DAY
GROUP BY username
HAVING unique_ips > 3 OR total_attempts > 10;
```

Explicación:
- Selecciona usuarios con intentos fallidos.
- Cuenta IPs únicas y total de intentos.
- Filtra usuarios con >3 IPs o >10 intentos.
- Útil para detectar ataques de fuerza bruta o distribuidos.

Consultas SQL Aplicadas a la Ciberseguridad

2. Funciones de Ventana para Análisis Temporal

Consulta para identificar picos sospechosos por hora:

```
WITH hourly_attempts AS (
  SELECT DATE_FORMAT(login_time, '%Y-%m-%d %H:00') AS hour,
        COUNT(*) AS attempts,
        AVG(COUNT(*)) OVER (ORDER BY DATE(login_time)
          RANGE BETWEEN 7 PRECEDING AND CURRENT ROW) AS moving_avg
  FROM log_in_attempts
  WHERE success = 0
  GROUP BY hour
)
SELECT hour, attempts, moving_avg
FROM hourly_attempts
WHERE attempts > moving_avg * 3;
```

Permite detectar aumentos súbitos de intentos fallidos, como en ataques DDoS o escaneos masivos.

Consultas SQL Aplicadas a la Ciberseguridad

3. Optimización de Consultas con Índices

Mejorar rendimiento de búsquedas por fecha y usuario:

```
CREATE INDEX idx_login_date_user ON log_in_attempts (login_date, username);
```

Facilita y acelera búsquedas frecuentes por estas columnas.

Consultas SQL Aplicadas a la Ciberseguridad

4. Seguridad en Consultas: Prevención de Inyección SQL

Ejemplo en Python:

```
cursor.execute(
  """
  SELECT * FROM employees
  WHERE username = %s AND department = %s
  """,
  (user_input, department_input)
);
```

Uso de marcadores (%s) para evitar inyecciones SQL. Separación de datos y estructura.

Consultas SQL Aplicadas a la Ciberseguridad

5. Análisis Forense con Triggers y Auditorías

Auditoría de cambios en tabla employees:

```
CREATE TABLE employees_audit (
  action VARCHAR(10), old_data JSON, new_data JSON,
  changed_by VARCHAR(50), change_time TIMESTAMP
);

CREATE TRIGGER audit_employees AFTER UPDATE ON employees
FOR EACH ROW BEGIN
  INSERT INTO employees_audit
  VALUES ('UPDATE', JSON_OBJECT(OLD), JSON_OBJECT(NEW), CURRENT_USER(), NOW());
END;
```

Permite rastrear cambios sospechosos o no autorizados.

Consultas SQL Aplicadas a la Ciberseguridad

6. Integración con Herramientas de Monitoreo (SIEM)

Consulta para exportar datos a SIEM como Splunk:

```
SELECT l.ip_address, l.username, l.login_time, e.department, m.os_patch_date
FROM log_in_attempts l
JOIN employees e ON l.username = e.username
JOIN machines m ON e.device_id = m.device_id
WHERE l.country NOT IN ('Mexico', 'USA')
  AND m.os_patch_date < '2024-01-01';
```

Permite detectar accesos desde ubicaciones sospechosas con sistemas desactualizados.

Consultas SQL Aplicadas a la Ciberseguridad

7. Manejo de Grandes Volúmenes de Datos

Particionado de tabla para acelerar consultas:

```
ALTER TABLE log_in_attempts
PARTITION BY RANGE (YEAR(login_date)) (
  PARTITION p2022 VALUES LESS THAN (2023),
  PARTITION p2023 VALUES LESS THAN (2024),
  PARTITION p2024 VALUES LESS THAN (2025)
);
```

Mejora rendimiento de búsquedas por año y facilita archivado de datos antiguos.