03_Exploratory_Analysis_sql (SQL)

Import notebook

Gold Layer: CVE Exploratory Data Analysis

This notebook performs the final analysis on our clean, normalized Silver-layer tables. The goal is to derive actionable business intelligence and identify key trends in the 2024 cybersecurity vulnerability data.

Each query below represents a "Gold" analysis, ready for reporting or visualization.

0. Verify Tables

```
SELECT 'CVEs Table' AS table_name, COUNT(*) AS record_count FROM cve_silver.cves
UNION ALL
SELECT 'Affected Products Table', COUNT(*) FROM cve_silver.affected_products;
```

▸ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [table_name: string, record_count: long]

Table

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

1. Top 25 Vendors

```
-- Top 25 Vendors by Vulnerability Count
SELECT
    vendor,
    COUNT(DISTINCT cve_id) AS vulnerability_count
FROM cve_silver.affected_products
WHERE vendor IS NOT NULL
    AND vendor NOT IN ('n/a', 'N/A')
GROUP BY vendor
ORDER BY vulnerability_count DESC
LIMIT 25;
```

▸ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [vendor: string, vulnerability_count: long]

Table

;

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

2. Severity Distribution

```
-- Vulnerability Severity Distribution
SELECT
    cvss_severity,
    COUNT(*) AS count
FROM cve_silver.cves
WHERE cvss_severity IS NOT NULL
GROUP BY cvss_severity
ORDER BY count DESC;
```

▸ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [cvss_severity: string, count: long]

Table

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

3. Monthly Trends

```
-- Vulnerability Disclosures Per Month
SELECT
    DATE_TRUNC('MONTH', date_published) AS publication_month,
    COUNT(*) AS count
FROM cve_silver.cves
GROUP BY publication_month
ORDER BY publication_month;
```

▸ _sqldf: pyspark.sql.connect.dataframe.DataFrame = [publication_month: timestamp, count: long]

Table

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

4. Top 10 Products

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [vendor: string, product: string ... 1 more field]

Table

This result is stored as `_sqldf` and can be used in other Python and SQL cells.