|  |
| --- |
| Technical Marketing |
| Scenario Launch Platform 2.0 Handbook |
| A document guide to demo enablement – version 0001 |

|  |
| --- |
| Jason Flood (CTO) / John Clarke (Principle)  [Q1 2025] |

Table of Contents

[Table of Figures 2](#_Toc191044920)

[Table of SQL 2](#_Toc191044921)

[Technical Marketing 4](#_Toc191044922)

[What is the function of the Technical Marketing? 4](#_Toc191044923)

[Scenario Launch Platform (SLP) 5](#_Toc191044924)

[What is the SLP? 5](#_Toc191044925)

[What is the purpose of Creating and editing connections? 5](#_Toc191044926)

[What is the purpose of Creating queries? 6](#_Toc191044927)

[What is the purpose of Freestyle mode? 6](#_Toc191044928)

[What is the purpose of Running OS Tasks mode? 7](#_Toc191044929)

[What is the purpose of Insights mode? 7](#_Toc191044930)

[What are Risks Events? 8](#_Toc191044931)

[Risk Event Categories 9](#_Toc191044932)

[What are Outliers? 11](#_Toc191044933)

[Scenario Databases 14](#_Toc191044934)

[Bane & Ox CRM - MySQL 15](#_Toc191044935)

[Example SQL Commands for CRM 15](#_Toc191044936)

[Inserts into CRM 15](#_Toc191044937)

[B&O CRM – Postgres 19](#_Toc191044938)

[B&O Finance - DB2 [Coming Soon] 19](#_Toc191044939)

[Example SQL Commands for Finance 20](#_Toc191044940)

[B&O Health - Oracle [Coming Soon] 22](#_Toc191044941)

[B&O Research – Sap Hana [Coming Soon] 23](#_Toc191044942)

[SQL to create MySQL CRM tables i](#_Toc191044943)

[SQL Queries to play with per table iv](#_Toc191044944)

[SAMPLE Injection queries viii](#_Toc191044945)

[SQL to create Finance tables viii](#_Toc191044946)

[SQL to create Research tables viii](#_Toc191044947)

[SQL to create Health tables viii](#_Toc191044948)

[Bibliography ix](#_Toc191044949)

## Table of Figures

[Figure 1 SLP Login 5](#_Toc191045236)

[Figure 2 SLP Demo Setup 5](#_Toc191045237)

[Figure 3 SLP Creating Queries Mode 6](#_Toc191045238)

[Figure 4 SLP Freestyle Mode 6](#_Toc191045239)

[Figure 5 SLP Running OS Tasks Mode 7](#_Toc191045240)

[Figure 6 Insights 7](#_Toc191045241)

[Figure 7Bane & Ox database by use case 14](#_Toc191045242)

[Figure 8 Bane & Ox CRM MySQL ERD 15](#_Toc191045243)

[Figure 9 Bane & Ox Finance DB2 ERD 20](#_Toc191045244)

## Table of SQL

[SQL Example 1 Insert command for tbl\_crm\_account\_status 15](#_Toc180657523)

[SQL Example 2 Insert command for tbl\_crm\_account 16](#_Toc180657524)

[SQL Example 3 Insert command for tbl\_calls 16](#_Toc180657525)

[SQL Example 4 Insert command for tbl\_product 16](#_Toc180657526)

[SQL Example 5 Insert command for tbl\_email\_lists 16](#_Toc180657527)

[SQL Example 6 Insert command for tbl\_marketing\_template 16](#_Toc180657528)

[SQL Example 7 Insert command for tbl\_marketing\_campaign 16](#_Toc180657529)

[SQL Example 8 Insert command for tbl\_bugs 16](#_Toc180657530)

[SQL Example 9 A Stored Procedure to create an outlier 18](#_Toc180657531)

[SQL Example 10 A further Stored Procedure to create an outlier 19](#_Toc180657532)

[SQL Example 11 Insert command for GOSALES.TBL\_ACCOUNT\_TYPE 20](#_Toc180657533)

[SQL Example 12 Insert command for GOSALES.TBL\_BRANCH 20](#_Toc180657534)

[SQL Example 13 Insert command for GOSALES.TBL\_CUSTOMER 21](#_Toc180657535)

[SQL Example 14 Insert command for GOSALES.TBL\_ACCOUNT 21](#_Toc180657536)

[SQL Example 15 Insert command for GOSALES.TBL\_CREDITCARD 21](#_Toc180657537)

[SQL Example 16 Insert command for GOSALES.TBL\_LOAN 21](#_Toc180657538)

[SQL Example 17 Insert command for GOSALES.TBL\_LOAN\_PAYMENT 21](#_Toc180657539)

[SQL Example 18 Insert command for GOSALES.TBL\_TRANSACTIONS 21](#_Toc180657540)

[SQL Example 19 SQL to create tbl\_crm\_accounts\_status i](#_Toc180657541)

[SQL Example 20 SQL to create tbl\_crm\_accounts i](#_Toc180657542)

[SQL Example 21 SQL to create tbl\_calls ii](#_Toc180657543)

[SQL Example 22 SQL to create tbl\_product ii](#_Toc180657544)

[SQL Example 23 SQL to create tbl\_email\_lists ii](#_Toc180657545)

[SQL Example 24 SQL to create tbl\_marketing\_template iii](#_Toc180657546)

[SQL Example 25 SQL to create tbl\_marketing\_campaign iii](#_Toc180657547)

[SQL Example 26 SQL to create tbl\_bug iii](#_Toc180657548)

[SQL Example 27 Queries for tbl\_bugs iv](#_Toc180657549)

[SQL Example 28 Queries for tbl\_calls iv](#_Toc180657550)

[SQL Example 29 Queries for tbl\_crm\_accounts v](#_Toc180657551)

[SQL Example 30 Queries for tbl\_accounts\_status v](#_Toc180657552)

[SQL Example 31 Queries for tbl\_email\_lists vi](#_Toc180657553)

[SQL Example 32 Queries for tbl\_marketing\_campaign vi](#_Toc180657554)

[SQL Example 33 Queries for tbl\_marketing\_template vii](#_Toc180657555)

[SQL Example 34 Queries for tbl\_product vii](#_Toc180657556)

[SQL Example 35 XSS Queries for tbl\_product viii](#_Toc180657557)

[SQL Example 36 SQL injections viii](#_Toc180657558)

# Technical Marketing

For technical marketing support please reach out to Jason Flood (CTO) or John Clarke (Principle).

## What is the function of the Technical Marketing?

The function of the technical marketing is to contribute technical skills and knowledge to product demos, marketing campaigns and launches. Technical marketing collaborates with sales and product management to develop materials. These materials ultimately become assets for product demonstrations and events.

Technical marketing builds assets to drive and assist sales teams and marketing in the following ways:

* Maintaining current demos and upgrading them to the latest version to keep abreast of new product features
* Supporting the existing Infrastructure and demo environment that provides content to the sales teams
* Building automation to make demoes and demo maintenance easier, more reliable and faster
* Creating new demo content, based on products, and security use cases. Building apps, UIs, and thinking of creative ways to exploit and market the portfolio
* Writing technical docs on how to use assets and technology
* Develop impactful demos and demo scripts that showcase industry use cases and workflows that embolden sales teams and marketing to deliver differentiated demos and successful proof of concepts
* Represent and assist and events and product demos, showcasing product portfolio and engage the community

# Scenario Launch Platform (SLP)

## What is the SLP?

The Scenario Launch Platform (SLP) is a demonstration enablement toolkit built to enable easy showcases of compelling use cases and product features. The SLP supports scenarios and demo scripts. The SLP endeavours to provide sales and marketing teams with a one click approach to drive a demonstration.

SLP provides users with the dynamic demonstration capability for clients or client facing events via a web portal. The SLP has several key functions that a user can leverage to achieve this. These functions are Creating and editing connections, Creating queries, Freestyle, Running crons, Insights and more.

|  |
| --- |
|  |

Figure 1 SLP Login

## What is the purpose of Creating and editing connections?

Creating and editing connections allows user to create a connection to known databases and schemas that have the existing users within the database. All tables are deleted and recreated. Once a user logs in they can access the Create and edit connections via the Settings cog in top left corner of page.

|  |
| --- |
|  |

Figure 2 SLP Demo Setup

## What is the purpose of Creating queries?

Creating queries allows a user to create any number of SQL queries to run against their stored connections. SLP come with preloaded queries to run against the demo schemas. However a user can create and store their own queries in the SLP database for further use. These can then be shared amongst other users

|  |
| --- |
|  |

Figure 3 SLP Creating Queries Mode

## What is the purpose of Freestyle mode?

Free Style mode allows users to enter their own queries and SQL directly into the database. With the added benefit of the SLP quick database pivot feature, users can quickly jump between databases to tell a compelling story.

|  |
| --- |
|  |

Figure 4 SLP Freestyle Mode

## What is the purpose of Running OS Tasks mode?

Running cron mode is where the SLP allows users upload files and run time crons or scheduled tasks. This give the user the flexibility to create narratives over time and upload use cases via cron. These scheduled tasks are stored in a database so that a use knows what active tasks they have running. They can also remove the tasks

|  |
| --- |
|  |

Figure 5 SLP Running OS Tasks Mode

## What is the purpose of Insights mode?

Insights mode allows a user to graph whats happening in their collector analytics. Showing over time how their Outliers threshold change

|  |
| --- |
|  |

Figure 6 Insights

# What are Risks Events?

A Risk Event is a collection of data points around an asset that shows a potential risk.

The Risk Event module evaluates assets every hour, looking for assets with potential risks. It examines the asset’s activities and other attributes and evaluates the asset’s risk. Assets are databases, database users, and operational system users.

A Risk Event is opened for assets with a high risk. It lists the findings (also called Leads) that indicate a risk over time, thus telling a broad story of the asset’s risk.

Findings might include the following examples:

* Outliers, or anomalies. Such as exceptionally high volumes of Select activities, Delete activities from a certain table, or new activities. (In other words, activities a user never did before).
* High severity policy violations.
* High volume of failed logins. Disabled by default. See details in Configuring Risk Event Leads.
* High volume of SQL exceptions. Disabled by default. See details in Configuring Risk Event Leads.

The following are some examples of Risk Events:

* A certain database had 300 failed logins during an hour, along with policy violations typical to SQL injection attacks. During the following hour, the same database has exceptionally high volume of SQL exceptions and exceptionally high volume of activities on the CUSTOMENT, CREDIT\_CARD and PURCHASE\_ORDER tables.
* A certain user had exceptional amounts of SELECT activities and an exceptional number of DELETE activities during an hour.

A list of terminology that the Risk Events module employs:

**Risk Event**

A Risk Event is a potential attack or breach with the following characteristics: asset, time frame, category, severity level, and findings.

**Asset**

The objects the Risk Events process observes when it is searching for Risk Events. The asset types are as follows: database, database user, and operating system user.

**Finding, lead**

data points, such as outliers and policy violations, that indicate a potential breach.

**Feature**

historical data about the asset that portrays an attribute of the asset. The Risk Events process uses a list of features to categorize the Risk Event and calculate its risk score and severity level. The following are some examples of features:

* Total number of high severity violations in the last hour: 23
* Number of failed log-in attempts in the last week: 744

## Risk Event Categories

**Abnormal or unexpected behaviour**

The asset is exhibiting deviations from normal activities. These may be anomalies, which are detected by comparing hourly activity with an average hour’s activity, or policy violations. Investigating these discrepancies is essential to identify whether the irregularities stem from legal activities or breaches of established policies.

**Brute force attack**

Hackers may attempt to access a database by trying common combinations of user names and passwords, such as 'ADMIN/ADMIN,' or by trying multiple variations of passwords for a specific user.

The occurrence of failed login attempts by multiple users, especially users such as 'ROOT', 'ADMIN', 'WP', and other common username can raise a suspicion of an attack. Moreover, multiple failed login attempts by a single user also raise suspicion of an attack. Other factors, such as excessive exceptions and policy violations, support this suspicion.

**Credential stealing / Account takeover**

There is a suspicion that an unauthorized user accessed the database. A new connection profile was used to access an account, exhibiting anomalies. This may indicate that a user’s credentials were stolen and misused. Errors that are associated with the account are reported.

**Cross-site scripting (XSS)**

Cross-site scripting (XSS) attacks attempt to insert malicious JavaScript code into the server through input fields and APIs. When such a script is stored in the database, it becomes persistent and is activated every time that a user accesses the data. SQL statements that include JavaScript may indicate an attempt to inject such malicious code.

**Data stealing / Data leak**

This attack is an attempt to retrieve data for unauthorized use. Data stealing is identified by abnormally high data retrieval activity. It may serve as the initial step in a ransomware attack, where the attacker steals the data and then removes it from the database altogether.

**Data tampering**

In this attack, the attacker accesses the database to modify or remove data. This may cause loss of data or disruption in system operations. Anomalies in the volume of data deletion or modification may indicate such an attack. Exceptions that are caused by missing data may support this suspicion.

**Distributed Denial-of-Service (DDoS) / Denial of Service**

Typically, Distributed Denial-of-Service (DDoS) attacks target a network, web server, or web service and should be detected at these levels. However, these attacks can cause a significant increase in data activity and overload the database.An extreme increase in data activity may indicate a DDoS attack. Also, certain SQL statement patterns can raise suspicion of such an attack, whether there is an increase in data activity or not.

**Massive grants**

A user granted many new privileges to various users. It might also be a user who typically does not grant privileges that has now granted a significant number of privileges.

**Massive user creation**

A user created many new users compared to normal. While this might appear as normal day-to-day activity, it should be investigated since the creation of many users is often the initial step in a Distributed Denial-of-Service (DDoS) attack.

**OS command injection**

OS command injection, also known as shell injection, is an attempt to run commands on the operating system, injected through an application. Certain patterns of SQL statements may indicate that the statement includes an OS command and can be suspected as an OS command injection attempt.

**Schema tampering**

Schema tampering refers to the malicious modification or removal of database elements, such as tables, views, and stored procedures. These modifications may cause excessive exceptions as applications fail to use these schema elements in a usual manner.An anomaly in the volume of activities that modify or remove schema elements may indicate such an attack. Exceptions that are related to schema elements that don’t exist support this suspicion.

**SQL Injection**

SQL injection attacks attempt to use application vulnerabilities by concatenating user input with SQL queries. If successful, these attacks can run malicious SQL commands by using the legitimate application connection. There are various SQL injection techniques. Explore policy violations and exceptions to understand which techniques, suggesting an SQL injection, were identified.

A more comprehensive explanation on risk events is available on IBMs documents for Guardium Insights. (IBM Corporation, n.d.)

# What are Outliers?

Guardium Insights uses Outliers to automatically identify abnormal server and user behavior, providing early detection of possible attacks.

The Outliers feature looks for anomalies in activity on a database or by a database user. It studies this activity over a certain period and builds a statistical model based on these observations. It then flags deviations in the statistical model to identify potential risks. These deviations are called outliers. They are based on the abnormality in the type of activity that is done, the time of the activity, the source of the activity, or a combination. For example, an outlier can be a user that usually queries the database 10 times a day and one day queries the database 1000 times instead.

In these examples or any others, it is the deviation of the activity that creates outliers. This way, the Outliers feature can indicate a security violation that is taking place, even if the activities themselves do not directly violate an existing security policy. Therefore, it can be a valuable tool in identifying danger in actions that are not inherently dangerous independently, that standard safeguards might overlook.

For example, if the Outliers engine alerts you to a high volume of failed logins or exceptions, these alerts can indicate brute force attacks or SQL injections. Perhaps from a disgruntled employee or hacked user account that is making bad changes or extracting much data.

Keep in mind that not all outliers flagged by the Outliers engine indicate a potential attack. Some abnormal activities are benign, but the Outliers engine flags them nevertheless. Such as when a user or application is doing maintenance work.

The statistical model that the Outliers is based on first takes an initial learning period to build itself. The default duration for that initial learning period is 7 days but it can be adjusted in settings. After the learning period is over, the statistical model does an iteration of calculations every hour, and continuously updates itself based on these hourly calculations.

Examples of user activities that can be detected as outliers include:

* When a user accesses a table for the first time.
* When a user selects specific data in a table that they never selected before.
* An exceptional volume of errors. For example, an application generates more SQL errors than it has in the past. This volume might indicate that an SQL injection attack is in progress.
* Activity that itself is unexceptional, but its volume is unusual.
* Activity that itself is unexceptional, but the time of activity is unusual. For example, a DBA is accessing a particular table more frequently than in the past. This frequency might indicate that the DBA is slowly downloading small amounts of data over time.

Example of database activities that can be detected as Outliers include:

* Exceptional volume of errors
* Activity that itself is unexceptional, but its volume is unusual.
* Activity that itself is unexceptional, but the time of activity is unusual.

The Outliers engine identifies the following five elements, per user and database, as indicators for potential outliers. It scans for these elements every hour:

**Verbs (commands)**

Actions that can indicate suspicious intent.

**Objects**

Objects on which the actions were run.

**Verbs + Objects**

For example, when a user makes certain selections in a certain table.

**Application**

The source program used to connect. For example, a user who usually connects from MySQL connected from a different client.

**Connection**

The client IP

When the Outliers engine finds outliers among those elements, it categorizes them into one or more of the following six outliers types:

**High volume of activities**

A high volume of activities far past the norm that the statistical model observed for user and database behavior.

**Exceptions**

A high volume of exceptions, or error types.

**New**

A high volume of new activity.

**Vulnerable activities**

A high volume of activity on object groups that Guardium considers vulnerable(You can configure what is considered vulnerable in settings).

**Diverse activities**

A high volume of diverse activities. To clarify, it is the high diversity of the activities, rather than the high volume of the activities, that characterizes this outliers type.

**Ongoing**

Each outlier is assigned an hourly score. If a particular outlier score is close to the threshold for a few hours straight, but doesn't quite pass the threshold, this outlier is of an ongoing type. This outlier type exists to detect activity that tries to hide under the radar.

The following table displays some examples of use cases that might display in the reports or Risk Events, each anchored to one of the Outliers types. Refer to the use case table to understand the kinds of risks that the Outliers that are listed in your summary and details reports might indicate.

|  |  |  |
| --- | --- | --- |
| **Use case** | **Outliers type** | **Details** |
| Excessive data extraction | High volume | Verb= select |
| Excessive data modification (dml) | High volume | Verb= insert or update |
| Access from a new connection | New | Source program (new app)  New OS  New client IP |
| Excessive activity from a connection | High volume | Source program (new app)  New OS  New client IP |
| Abnormal number of errors | Error | Verb= activity that caused the error. |
| Abnormal type of SQL operation | New | Any verb the SQL did not engage in before and now did run |
| Abnormal working hours | High volume,new,diverse | Verb |
| Excessive privileges granted by a user | High volume | Verb= grant |
| |  | | --- | | Schema tampering | | High volume,New | Verb= drop of schema elements (function, package, table, view, and others) |
| High number of different (unique) activity types | Diverse | Verb |

A more comprehensive explanation on outliers is available on IBMs documents for Guardium Insights. (IBM Corporation, n.d.)

# Scenario Databases

The Guardium demo databases stack will be built using several different types of databases. Each database will be associated with a different business. Each database type will consist of max 8 – 10 tables. These are configured to contain data types that will support Guardium rules. Distilling the databases down to their minimum necessary will make demonstration support and storytelling more manageable.

The premise for the demo environments revolves around a fictious company called Bane & Ox. This company has multiple business units (BU’s) such as Finance, Health, CRM and Research. Each business units assets will form the backbone of demo stories use cases as described in Figure 9. More BU’s can be added to suit a specific customer Demo if needed.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

Figure 7Bane & Ox database by use case

# Bane & Ox CRM - MySQL

The Bane & OX CRM is built using a MySQL Database. It consists of 8 tables housing the data for this BU. Data such as PII, customer support, products and marketing can be found within the tables. This data will be used to drive several of the features of Guardium using CRM use cases. The entity relationship diagram (ERD) shown in Figure 10, show the tables and columns of the CRM BU.

|  |
| --- |
|  |

Figure 8 Bane & Ox CRM MySQL ERD

### Example SQL Commands for CRM

To assist in understanding of the data, and also to enable the use of Freestyle mode in the SLP toolkit a number of example SQL commands are provided.

### Inserts into CRM

#### Insert into tbl\_crm\_account\_status

|  |
| --- |
| **INSERT** **INTO** tbl\_crm\_accounts\_status (id, status) **values** (1,**'lead'**),  (2,**'opportunity'**),  (3,**'customer/won'**),  (4,**'archive'**); |

SQL Example 1 Insert command for tbl\_crm\_account\_status

#### Insert into tbl\_crm\_accounts

|  |
| --- |
| **INSERT** **INTO** tbl\_crm\_accounts **VALUES** (**'df61978a-f4cc-ff64-8de0-53e90f19a56a'**,**'B.H. Edwards Inc'**,**'2024-09-22 03:11:33'**,**'2024-09-24 03:11:33'**,**'seed\_will\_id'**,**'1'**,**NULL**,0,**'seed\_max\_id'**,**'Customer'**,**'Technology'**,**NULL**,**NULL**,**'1715 Scott Dr'**,**'Alabama'**,**'CA'**,**'14882'**,**'USA'**,**NULL**,**'(847) 706-6877'**,**NULL**,**'www.devim.edu'**,**NULL**,**NULL**,**NULL**,**'1715 Scott Dr'**,**'Alabama'**,**'CA'**,**'14882'**,**'USA'**,**NULL**,**NULL**,**NULL**,1); |

SQL Example 2 Insert command for tbl\_crm\_account

#### Insert into tbl\_calls

|  |
| --- |
| **INSERT** **INTO** tbl\_calls **VALUES** (**'e854b40d-414e-6c8d-d2b7-53e90f7b0f77'**,**'Left a message'**,**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**),**NOW**(),**'1'**,**'1'**,**NULL**,0,**'seed\_max\_id'**,0,30,**'2014-12-28 09:30:00'**,**'2014-12-28 10:00:00'**,**'Accounts'**,**'Planned'**,**'Outbound'**,**'df61978a-f4cc-ff64-8de0-53e90f19a56a'**,-1,**NULL**); |

SQL Example 3 Insert command for tbl\_calls

#### Insert into tbl\_product

|  |
| --- |
| **INSERT** **INTO** ***tbl\_product*** **VALUES** (**'d67f8d9d'**,**'Detergent'**,**'Keep the clothes cleaner with this detergent'**,**'4.80'**,10) |

SQL Example 4 Insert command for tbl\_product

#### Insert into tbl\_email\_lists

|  |
| --- |
| **INSERT** **INTO** tbl\_email\_lists **VALUES** ('d67f8d9d-7c28-00df-47f1-53e90f54066f','jim@example.com','JIM@EXAMPLE.COM',0,**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**)) |

SQL Example 5 Insert command for tbl\_email\_lists

#### Insert into tbl\_marketing\_template

|  |
| --- |
| **INSERT** **INTO** tbl\_marketing\_template **VALUES** ('53e90f54066f','Free Detergent with new technology','Keep the clothes cleaner with this detergent') |

SQL Example 6 Insert command for tbl\_marketing\_template

#### Insert into tbl\_marketing\_campaign

|  |
| --- |
| **INSERT** **INTO** tbl\_marketing\_campaign (***email\_id***,***template\_id***,campaign\_date)**VALUES** ('d67f8d9d-7c28-00df-47f1-53e90f54066f','53e90f54066f',**DATE\_SUB**(**NOW**(), ***INTERVAL*** -10 **DAY**)) |

SQL Example 7 Insert command for tbl\_marketing\_campaign

#### Insert into tbl\_bugs

|  |
| --- |
| **INSERT** **INTO** tbl\_bugs **VALUES** ('e4f7505c-0a0e-f582-f406-53e90f8a5637','Error occurs while running count query',**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**),**NOW**(),'1','1',**NULL**,0,'seed\_max\_id',1,**NULL**,'Assigned','Medium',**NULL**,**NULL**,**NULL**,**NULL**,**NULL**,**NULL**); |

SQL Example 8 Insert command for tbl\_bugs

#### Stored Procedures

|  |
| --- |
| **drop procedure if exists salesDB.createOutlierTable;**  **DELIMITER //**  **CREATE PROCEDURE salesDB.createOutlierTable(IN var\_TableName VARCHAR(255))**  **BEGIN**  **DECLARE mytable VARCHAR(100);**  **SET @StartSQL = CONCAT('DROP TABLE IF EXISTS ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName));**  **PREPARE stmt FROM @StartSQL;**  **EXECUTE stmt;**  **DEALLOCATE PREPARE stmt;**    **SET @SQL = CONCAT('CREATE TABLE ',CONCAT('salesDB.tbl\_outlier\_',var\_TableName),' LIKE ',CONCAT('salesDB','.tbl\_product'));**  **PREPARE stmt1 FROM @SQL;**  **EXECUTE stmt1;**  **DEALLOCATE PREPARE stmt1;**  **SET @mytable = CONCAT('salesDB.tbl\_outlier\_', var\_TableName);**    **SET @sql2 = CONCAT('INSERT INTO ', @mytable, ' VALUES (?, ?, ? , ?, ?)');**  **PREPARE stmt2 FROM @sql2;**  **SET @value1 = 'd67f8d9d';**  **SET @value2 = 'Detergent 1';**  **SET @value3 = 'Keep the clothes cleaner with this detergent 1';**  **SET @value4 = '80.1';**  **SET @value5 = 10;**    **EXECUTE stmt2 USING @value1, @value2, @value3, @value4, @value5;**  **DEALLOCATE PREPARE stmt2;**  **SET @SQL3 = CONCAT('select \* from ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName));**  **PREPARE stmt3 FROM @SQL3;**  **EXECUTE stmt3;**  **DEALLOCATE PREPARE stmt3;**  **SET @FinsihSQL = CONCAT('DROP TABLE IF EXISTS ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName));**  **PREPARE stmt4 FROM @FinsihSQL;**  **EXECUTE stmt4;**  **DEALLOCATE PREPARE stmt4;**  **END//**  **call salesDB.createOutlierTable("john")** |

SQL Example 9 A Stored Procedure to create an outlier

#### Stored Procedures

|  |
| --- |
| **drop procedure if exists salesDB.createOutlierTableSpecificAccess;**  **DELIMITER //**  **CREATE PROCEDURE salesDB.createOutlierTableSpecificAccess(IN var\_TableName VARCHAR(255))**  **BEGIN**  **DECLARE mytable VARCHAR(100);**  **SET @StartSQL = CONCAT('DROP TABLE IF EXISTS ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName));**  **PREPARE stmt FROM @StartSQL;**  **EXECUTE stmt;**  **DEALLOCATE PREPARE stmt;**    **SET @SQL = CONCAT('CREATE TABLE ',CONCAT('salesDB.tbl\_outlier\_',var\_TableName),' LIKE ',CONCAT('salesDB','.tbl\_product'));**  **PREPARE stmt1 FROM @SQL;**  **EXECUTE stmt1;**  **DEALLOCATE PREPARE stmt1;**  **SET @mytable = CONCAT('salesDB.tbl\_outlier\_', var\_TableName);**  **SET @sql2 = CONCAT('INSERT INTO ', @mytable, ' VALUES (?, ?, ? , ?, ?)');**  **PREPARE stmt2 FROM @sql2;**    **SET @value1 = 'd67f8d9d';**  **SET @value2 = 'Detergent 1';**  **SET @value3 = 'Keep the clothes cleaner with this detergent 1';**  **SET @value4 = '80.1';**  **SET @value5 = 10;**    **-- Execute the prepared statement with the values**  **EXECUTE stmt2 USING @value1, @value2, @value3, @value4, @value5;**    **-- Clean up**  **DEALLOCATE PREPARE stmt2;**  **SET @SQL3 = CONCAT('select name from ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName, ' where quantity = 10'));**  **PREPARE stmt3 FROM @SQL3;**  **EXECUTE stmt3;**  **DEALLOCATE PREPARE stmt3;**  **SET @FinsihSQL = CONCAT('DROP TABLE IF EXISTS ', CONCAT('salesDB.tbl\_outlier\_',var\_TableName));**  **PREPARE stmt4 FROM @FinsihSQL;**  **EXECUTE stmt4;**  **DEALLOCATE PREPARE stmt4;**  **END//**  **call salesDB.createOutlierTableSpecificAccess("john");** |

SQL Example 10 A further Stored Procedure to create an outlier

# B&O CRM – Postgres

|  |
| --- |
|  |

Figure 9 Bane & Ox CRM Postgres ERD

# B&O Finance - DB2 [Coming Soon]

The Bane & OX Finance is built using a DB2 Database. It consists of 8 tables housing the data for this BU. Data such as PII, credit card, transactions and loans can be found within the tables. This data will be used to drive several of the features of Guardium using Finance use cases. The entity relationship diagram (ERD) in Figure 12, shows the tables and columns of the Finance BU.

|  |
| --- |
|  |

Figure 9 Bane & Ox Finance DB2 ERD

### Example SQL Commands for Finance

To assist in understanding of the data, and to enable the use of Freestyle mode in the SLP toolkit several example SQL commands are provided.

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_ACCOUNT\_TYPE **VALUES** ('2','Current'),('5','Savings'),('8','Student'); |

SQL Example 11 Insert command for GOSALES.TBL\_ACCOUNT\_TYPE

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_BRANCH **VALUES** ('1234','HASTINGS BRANCH','HBAO','UNIT 7 Main Street','678-098-654-321'),  ('1245','OCEAN BRANCH','OBOD','UNIT 12 Ocean Drive','678-098-654-333'),  ('1256','VIEWING BRANCH','VBCW','107 Cliff Walk Road','678-098-654-111'),  ('1267','TREES GREEN BRANCH','TGBG','45 Green Street','678-098-654-222'); |

SQL Example 12 Insert command for GOSALES.TBL\_BRANCH

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_CUSTOMER **VALUES** ('5','Billy Bob','491-999-000-123','billy\_bob@mail.com','10 Linen Drive','billy\_bob@mail.com','password','1234'),  ('8','Jason Flood','491-234-777-123','jason\_flood@mail.com','20 Watery Lane','jason\_flood@mail.com','password123','1234'),  ('11','John Clarke','400-321-888-555','john\_clarke@mail.com','32 Pond De Road','john\_clarke@mail.com','password111','1256'),  ('14','John Short','345-321-666-555','john\_short@mail.com','11 The Gable End','john\_short@mail.com','password111','1256'); |

SQL Example 13 Insert command for GOSALES.TBL\_CUSTOMER

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_ACCOUNT **VALUES** ('5','1284907823','300.23','2010-12-31','OPEN','5','8'),  ('8','1289990134','50000.11','2012-10-11','OPEN','8','5'),  ('11','1284567321','20.85','2014-12-21','OPEN','11','5'),  ('14','1284907333','4567.00','2009-11-24','OPEN','14','2'); |

SQL Example 14 Insert command for GOSALES.TBL\_ACCOUNT

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_CREDITCARD **VALUES** ('5275','Mr Billy Bob','Credit','1000.00','5488515324554846','838','04/29','5'),  ('5308','Mr Billy Bob','Debit','750.00','5276418415485044','035','11/27','5'),  ('5341','Mr Jason Flood','Credit','954.11','5362852720486583','439','10/30','8'),  ('5374','Mr John Short','Credit','3457.22','5488515324554846','124','07/26','14'); |

SQL Example 15 Insert command for GOSALES.TBL\_CREDITCARD

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_LOAN **VALUES** ('111','500.00','10.03','12 Months','462.00','2024-09-01','2025-09-01','11'),  ('146','500.00','09.01','12 Months','500.00','2024-10-01','2025-10-01','8'),  ('181','500.00','10.03','12 Months','462.00','2024-09-01','2025-09-01','14'),  ('216','500.00','10.03','12 Months','462.00','2024-09-01','2025-09-01','5'); |

SQL Example 16 Insert command for GOSALES.TBL\_LOAN

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_LOAN\_PAYMENT **VALUES** ('3000','2024-10-01','40.00','111'),  ('3021','2024-10-01','40.00','181'),  ('3042','2024-10-01','40.00','216'); |

SQL Example 17 Insert command for GOSALES.TBL\_LOAN\_PAYMENT

|  |
| --- |
| **INSERT** **INTO** GOSALES.TBL\_TRANSACTIONS **VALUES** ('3000','LOAN','40.00','2024-10-01','11'),  ('3021','LOAN','40.00','2024-10-01','14'),  ('3042','LOAN','40.00','2024-10-01','5'); |

SQL Example 18 Insert command for GOSALES.TBL\_TRANSACTIONS

# B&O Health - Oracle [Coming Soon]

# B&O Research – Sap Hana [Coming Soon]

### SQL to create MySQL CRM tables

#### Create tbl\_crm\_accounts\_status

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_crm\_accounts\_status***;  **CREATE** **TABLE** tbl\_crm\_accounts\_status (  id **int**(11) **NOT** **NULL** **AUTO\_INCREMENT**,  status **varchar**(64) **NOT** **NULL** **DEFAULT** '',  **PRIMARY** **KEY** (id)  ) **ENGINE**=InnoDB **DEFAULT** CHARSET=utf8;  **UNLOCK** TABLES; |

SQL Example 19 SQL to create tbl\_crm\_accounts\_status

#### Create tbl\_crm\_accounts

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_crm\_accounts***;  **CREATE** **TABLE** tbl\_crm\_accounts (  id **varchar**(36) **NOT** **NULL**,  name **varchar**(150) **DEFAULT** **NULL**,  date\_entered **datetime** **DEFAULT** **NULL**,  date\_modified **datetime** **DEFAULT** **NULL**,  modified\_user\_id **char**(36) **DEFAULT** **NULL**,  created\_by **char**(36) **DEFAULT** **NULL**,  description **text**,  deleted **tinyint**(1) **DEFAULT** '0',  assigned\_user\_id **char**(36) **DEFAULT** **NULL**,  account\_type **varchar**(50) **DEFAULT** **NULL**,  industry **varchar**(50) **DEFAULT** **NULL**,  annual\_revenue **varchar**(100) **DEFAULT** **NULL**,  phone\_fax **varchar**(100) **DEFAULT** **NULL**,  billing\_address\_street **varchar**(150) **DEFAULT** **NULL**,  billing\_address\_city **varchar**(100) **DEFAULT** **NULL**,  billing\_address\_state **varchar**(100) **DEFAULT** **NULL**,  billing\_address\_postalcode **varchar**(20) **DEFAULT** **NULL**,  billing\_address\_country **varchar**(255) **DEFAULT** **NULL**,  rating **varchar**(100) **DEFAULT** **NULL**,  phone\_office **varchar**(100) **DEFAULT** **NULL**,  phone\_alternate **varchar**(100) **DEFAULT** **NULL**,  website **varchar**(255) **DEFAULT** **NULL**,  ownership **varchar**(100) **DEFAULT** **NULL**,  employees **varchar**(10) **DEFAULT** **NULL**,  ticker\_symbol **varchar**(10) **DEFAULT** **NULL**,  shipping\_address\_street **varchar**(150) **DEFAULT** **NULL**,  shipping\_address\_city **varchar**(100) **DEFAULT** **NULL**,  shipping\_address\_state **varchar**(100) **DEFAULT** **NULL**,  shipping\_address\_postalcode **varchar**(20) **DEFAULT** **NULL**,  shipping\_address\_country **varchar**(255) **DEFAULT** **NULL**,  parent\_id **char**(36) **DEFAULT** **NULL**,  sic\_code **varchar**(10) **DEFAULT** **NULL**,  campaign\_id **char**(36) **DEFAULT** **NULL**,  status **int**(11) **unsigned** **NOT** **NULL**,  **PRIMARY** **KEY** (id),  KEY FK\_STATUS (status),  **CONSTRAINT** FK\_STATUS **FOREIGN** **KEY** (status) **REFERENCES** tbl\_crm\_accounts\_status (id)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8;  **UNLOCK** TABLES; |

SQL Example 20 SQL to create tbl\_crm\_accounts

#### Create tbl\_calls

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_calls***;  **CREATE** **TABLE** tbl\_calls (  id **char**(36) **NOT** **NULL**,  name **varchar**(50) **DEFAULT** **NULL**,  date\_entered **datetime** **DEFAULT** **NULL**,  date\_modified **datetime** **DEFAULT** **NULL**,  modified\_user\_id **char**(36) **DEFAULT** **NULL**,  created\_by **char**(36) **DEFAULT** **NULL**,  description **text**,  deleted **tinyint**(1) **DEFAULT** '0',  assigned\_user\_id **char**(36) **DEFAULT** **NULL**,  duration\_hours **int** **DEFAULT** **NULL**,  duration\_minutes **int** **DEFAULT** **NULL**,  date\_start **varchar**(50) **DEFAULT** **NULL**,  date\_end **varchar**(50) **DEFAULT** **NULL**,  parent\_type **varchar**(255) **DEFAULT** **NULL**,  status **varchar**(100) **DEFAULT** 'Planned',  direction **varchar**(100) **DEFAULT** **NULL**,  parent\_id **char**(36) **DEFAULT** **NULL**,  reminder\_time **int** **DEFAULT** '-1',  outlook\_id **varchar**(255) **DEFAULT** **NULL**,  **PRIMARY** **KEY** (id)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8;  **UNLOCK** TABLES; |

SQL Example 21 SQL to create tbl\_calls

#### Create tbl\_product

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_product***;  **CREATE** **TABLE** tbl\_product (  id **varchar**(25) **NOT** **NULL**,  name **varchar**(150) **DEFAULT** **NULL**,  description **text**,  price **DECIMAL**(10,2),  quantity **INT** **DEFAULT** '0',  **PRIMARY** **KEY** (id)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8; |

SQL Example 22 SQL to create tbl\_product

#### Create tbl\_email\_lists

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_email\_lists***;  **CREATE** **TABLE** tbl\_email\_lists (  id **varchar**(36) **NOT** **NULL** **UNIQUE**,  email\_address **varchar**(150) **NOT** **NULL** **UNIQUE**,  email\_address\_caps **varchar**(255) **DEFAULT** **NULL**,  opt\_out **tinyint**(1) **DEFAULT** '0',  date\_created **datetime** **DEFAULT** **NULL**,  **PRIMARY** **KEY** (id,email\_address)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8; |

SQL Example 23 SQL to create tbl\_email\_lists

#### Create tbl\_marketing\_template

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_marketing\_template***;  **CREATE** **TABLE** tbl\_marketing\_template (  id **varchar**(36) **NOT** **NULL**,  subject **varchar**(255) **NOT** **NULL**,  body **text**,  **PRIMARY** **KEY** (id)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8; |

SQL Example 24 SQL to create tbl\_marketing\_template

#### Create tbl\_marketing\_campaign

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_marketing\_campaign***;  **CREATE** **TABLE** tbl\_marketing\_campaign (  id **int** **NOT** **NULL** **AUTO\_INCREMENT**,  email\_id **char**(36) **NOT** **NULL**,  template\_id **char**(36) **NOT** **NULL**,  campaign\_date **datetime** **NOT** **NULL**,  **PRIMARY** **KEY** (id),  KEY FK\_TEMPLATE\_ID (template\_id),  KEY FK\_EMAIL\_ID (email\_id),  **CONSTRAINT** FK\_TEMPLATE\_ID **FOREIGN** **KEY** (template\_id) **REFERENCES** tbl\_marketing\_template (id),  **CONSTRAINT** FK\_EMAIL\_ID **FOREIGN** **KEY** (email\_id) **REFERENCES** tbl\_email\_lists (id)) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8;  **ALTER** **TABLE** ***tbl\_marketing\_campaign*** **AUTO\_INCREMENT**=100; |

SQL Example 25 SQL to create tbl\_marketing\_campaign

#### Create tbl\_bugs

|  |
| --- |
| **DROP** **TABLE** **IF** **EXISTS** ***tbl\_bugs***;  **CREATE** **TABLE** tbl\_bugs (  id **char**(36) **NOT** **NULL**,  name **varchar**(255) **DEFAULT** **NULL**,  date\_entered **varchar**(255) **DEFAULT** **NULL**,  date\_modified **varchar**(255) **DEFAULT** **NULL**,  modified\_user\_id **char**(36) **DEFAULT** **NULL**,  created\_by **char**(36) **DEFAULT** **NULL**,  description **text**,  deleted **tinyint**(1) **DEFAULT** '0',  assigned\_user\_id **char**(36) **DEFAULT** **NULL**,  bug\_number **int** **DEFAULT** '0',  type **varchar**(255) **DEFAULT** **NULL**,  status **varchar**(100) **DEFAULT** **NULL**,  priority **varchar**(100) **DEFAULT** **NULL**,  resolution **varchar**(255) **DEFAULT** **NULL**,  work\_log **text**,  found\_in\_release **varchar**(255) **DEFAULT** **NULL**,  fixed\_in\_release **varchar**(255) **DEFAULT** **NULL**,  source **varchar**(255) **DEFAULT** **NULL**,  product\_category **varchar**(255) **DEFAULT** **NULL**,  **PRIMARY** **KEY** (id)  ) **ENGINE**=MyISAM **DEFAULT** CHARSET=utf8;  **UNLOCK** TABLES; |

SQL Example 26 SQL to create tbl\_bug

### SQL Queries to play with per table

#### Queries for tbl\_bugs

|  |
| --- |
| /\* SQL queries for tbl\_bugs \*/  /\*SELECT\*/  **select** \* **from** tbl\_bugs;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_bugs  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_bugs  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_bugs  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_bugs  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_bugs  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_bugs **VALUES** ('aaaaaaaa-bbbb-cccc-dddd-111111111111','Error occurs while running count query',**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**),**NOW**(),'1','1',**NULL**,0,'seed\_max\_id',1,**NULL**,'Assigned','Medium',**NULL**,**NULL**,**NULL**,**NULL**,**NULL**,**NULL**);  /\*DELETE\*/  **DELETE** **FROM** tbl\_bugs **WHERE** id='aaaaaaaa-bbbb-cccc-dddd-111111111111'; |

SQL Example 27 Queries for tbl\_bugs

#### Queries for tbl\_calls

|  |
| --- |
| /\* SQL queries for tbl\_calls \*/  /\*SELECT\*/  **select** \* **from** tbl\_calls;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_calls  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_calls  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_calls  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_calls  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_calls  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_calls **VALUES** ('aaaaaaaa-bbbb-cccc-dddd-111111111111','Left a message',**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**),**NOW**(),'1','1',**NULL**,0,'seed\_max\_id',0,30,'2014-12-28 09:30:00','2014-12-28 10:00:00','Accounts','Planned','Outbound','df61978a-f4cc-ff64-8de0-53e90f19a56a',-1,**NULL**);  /\*DELETE\*/  **DELETE** **FROM** tbl\_calls **WHERE** id='aaaaaaaa-bbbb-cccc-dddd-111111111111'; |

SQL Example 28 Queries for tbl\_calls

#### Queries for tbl\_crm\_accounts

|  |
| --- |
| /\* SQL queries for tbl\_crm\_accounts \*/  /\*SELECT\*/  **select** \* **from** tbl\_crm\_accounts;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_crm\_accounts  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_crm\_accounts  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_crm\_accounts  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_crm\_accounts  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_crm\_accounts  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_crm\_accounts **VALUES** ('aaaaaaaa-bbbb-cccc-dddd-111111111111','B.H. Edwards Inc','2024-09-22 03:11:33','2024-09-24 03:11:33','seed\_will\_id','1',**NULL**,0,'seed\_max\_id','Customer','Technology',**NULL**,**NULL**,'1715 Scott Dr','Alabama','CA','14882','USA',**NULL**,'(847) 706-6877',**NULL**,'www.devim.edu',**NULL**,**NULL**,**NULL**,'1715 Scott Dr','Alabama','CA','14882','USA',**NULL**,**NULL**,**NULL**,1);  /\*DELETE\*/  **DELETE** **FROM** tbl\_crm\_accounts **WHERE** id='aaaaaaaa-bbbb-cccc-dddd-111111111111'; |

SQL Example 29 Queries for tbl\_crm\_accounts

#### Queries for tbl\_accounts\_status

|  |
| --- |
| /\* SQL queries for tbl\_crm\_accounts\_status \*/  /\*SELECT\*/  **select** \* **from** tbl\_crm\_accounts\_status;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_crm\_accounts\_status  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_crm\_accounts\_status  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_crm\_accounts\_status  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_crm\_accounts\_status  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_crm\_accounts\_status  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_crm\_accounts\_status (id, status) **values** (6,'No Lead');  /\*DELETE\*/  **DELETE** **FROM** tbl\_crm\_accounts\_status **WHERE** id=6; |

SQL Example 30 Queries for tbl\_accounts\_status

#### Queries for tbl\_email\_lists

|  |
| --- |
| /\* SQL queries for tbl\_email\_lists \*/  /\*SELECT\*/  **select** \* **from** tbl\_email\_lists;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_email\_lists  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_email\_lists  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_email\_lists  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_email\_lists  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_email\_lists  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_email\_lists **VALUES** ('aaaaaaaa-bbbb-cccc-dddd-111111111111','jay@example.com','JAY@EXAMPLE.COM',0,**DATE\_SUB**(**NOW**(), **INTERVAL** 2 **DAY**));  /\*DELETE\*/  **DELETE** **FROM** tbl\_email\_lists **WHERE** id='aaaaaaaa-bbbb-cccc-dddd-111111111111'; |

SQL Example 31 Queries for tbl\_email\_lists

#### Queries for tbl\_marketing\_campaign

|  |
| --- |
| /\* SQL queries for tbl\_marketing\_campaign \*/  /\*SELECT\*/  **select** \* **from** tbl\_marketing\_campaign;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_marketing\_campaign  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_marketing\_campaign  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_marketing\_campaign  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_marketing\_campaign  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_marketing\_campaign  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_marketing\_campaign (email\_id,template\_id,campaign\_date)**VALUES** ('aaaaaaaa-bbbb-cccc-dddd-111111111111','a1a1a1a1a1a1',**DATE\_SUB**(**NOW**(), ***INTERVAL*** -10 **DAY**))  /\*DELETE\*/  **DELETE** **FROM** tbl\_marketing\_campaign **where** email\_id='aaaaaaaa-bbbb-cccc-dddd-111111111111' **AND** template\_id='a1a1a1a1a1a1'; |

SQL Example 32 Queries for tbl\_marketing\_campaign

#### Queries for tbl\_marketing\_template

|  |
| --- |
| /\* SQL queries for tbl\_marketing\_template \*/  /\*SELECT\*/  **select** \* **from** tbl\_marketing\_template ;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_marketing\_template  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_marketing\_template  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_marketing\_template  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_marketing\_template  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_marketing\_template  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_marketing\_template **VALUES** ('a1a1a1a1a1a1','Free Detergent with new technology','Keep the clothes cleaner with this detergent')  /\*DELETE\*/  **DELETE** **FROM** tbl\_marketing\_template **WHERE** id='a1a1a1a1a1a1'; |

SQL Example 33 Queries for tbl\_marketing\_template

#### Queries for tbl\_product

|  |
| --- |
| /\* SQL queries for tbl\_product \*/  /\*SELECT\*/  **select** \* **from** tbl\_product;  /\*ALTER\*/  **ALTER** **TABLE** tbl\_product  **ADD** ***new\_col*** **varchar**(255);  **ALTER** **TABLE** tbl\_product  **DROP** new\_col ;  /\*ALTER & UPDATE & DROP\*/  **ALTER** **TABLE** tbl\_product  **ADD** ***new\_col\_up*** **varchar**(255);  **UPDATE** tbl\_product  **SET** ***new\_col\_up*** =**"Updated"**;  **ALTER** **TABLE** tbl\_product  **DROP** new\_col\_up ;  /\*INSERT\*/  **INSERT** **INTO** tbl\_product **VALUES** ('a1a1a1a1','Detergent','Keep the clothes cleaner with this detergent','4.80',10)  /\*DELETE\*/  **DELETE** **FROM** tbl\_product **WHERE** id='a1a1a1a1'; |

SQL Example 34 Queries for tbl\_product

### SAMPLE Injection queries

#### XSS Queries for tbl\_product

|  |
| --- |
| /\*INSERT\*/  **INSERT** **INTO** tbl\_product **VALUES** ('a1a1a1a1','IMG','<IMG SRC=javascript:alert(&quot;XSS&quot;)>','4.80',10);  /\*SELECT\*/  **select** \* **from** tbl\_product;  **SELECT** \* **FROM** tbl\_product **WHERE** description **LIKE** '%<IMG SRC%';  /\*INSERT\*/  **INSERT** **INTO** tbl\_product **VALUES** ('a2a2a2a2','Body','<BODY onload!#$%&()\*~+-\_.,:;?@[/|\\]^`=alert(\"XSS\")>','4.80',10);  /\*SELECT\*/  **select** \* **from** tbl\_product;  **SELECT** \* **FROM** tbl\_product **WHERE** description **LIKE** '%<BODY onload%';  /\*DELETE\*/  **DELETE** **FROM** tbl\_product **WHERE** id='a1a1a1a1';  /\*DELETE\*/  **DELETE** **FROM** tbl\_product **WHERE** id='a2a2a2a2'; |

SQL Example 35 XSS Queries for tbl\_product

#### SAMPLE SQL Injection (SQLI) Queries

|  |
| --- |
| select \* from salesDB.tbl\_crm\_accounts WHERE 1=1 AND 1=1;  select \* from salesDB.tbl\_crm\_accounts\_status where id=1 or 15=15;  select \* from salesDB.tbl\_product where id=1 or true--; |

SQL Example 36 SQL injections

### SQL to create Finance tables

### SQL to create Research tables

### SQL to create Health tables

# Bibliography

IBM Corporation. (n.d.). *Outliers*. Retrieved from ibm.com: https://www.ibm.com/docs/en/guardium-insights/3.x?topic=outliers

IBM Corporation. (n.d.). *Risk Events*. Retrieved from ibm.com: https://www.ibm.com/docs/en/guardium-insights/3.x?topic=risk-events