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

Table of SQL

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.

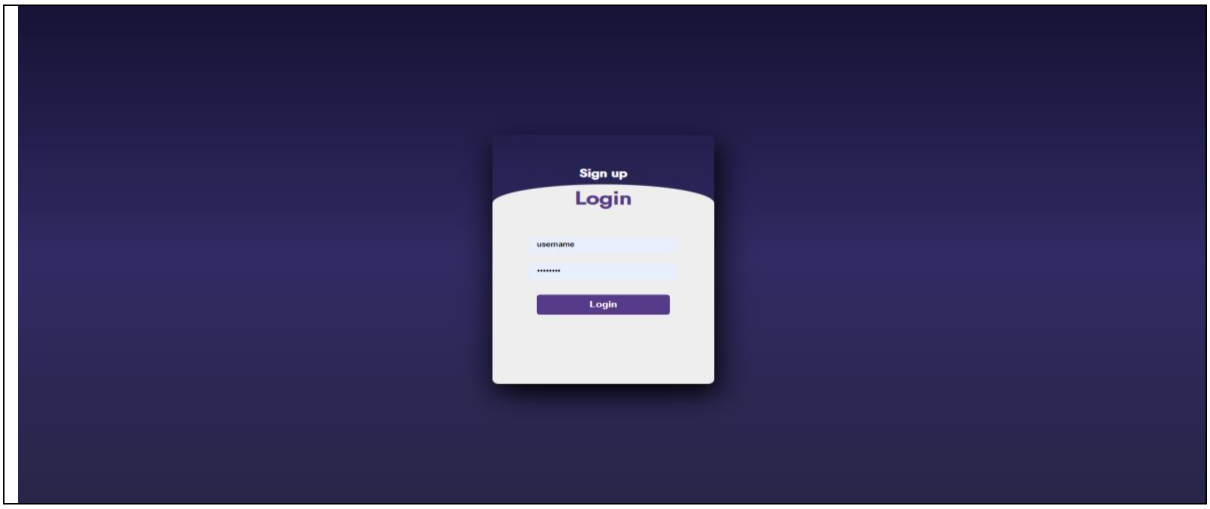
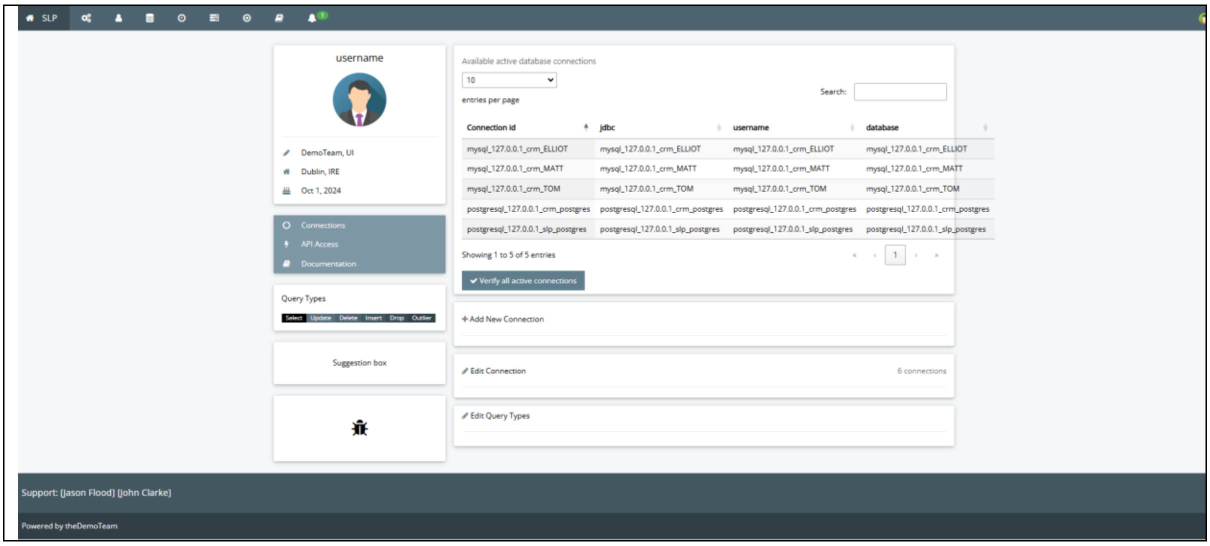


Figure 1 SLP Login

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.



DemoTeam, UI
Dublin, IRE
Oct 1, 2024

Available active database connections

10 entries per page

Connection id	jdbc	username	database
mysql_127.0.0.1_crm_ELLIOT	mysql_127.0.0.1_crm_ELLIOT	mysql_127.0.0.1_crm_ELLIOT	mysql_127.0.0.1_crm_ELLIOT
mysql_127.0.0.1_crm_MATT	mysql_127.0.0.1_crm_MATT	mysql_127.0.0.1_crm_MATT	mysql_127.0.0.1_crm_MATT
mysql_127.0.0.1_crm_TOM	mysql_127.0.0.1_crm_TOM	mysql_127.0.0.1_crm_TOM	mysql_127.0.0.1_crm_TOM
postgresql_127.0.0.1_crm_postgres	postgresql_127.0.0.1_crm_postgres	postgresql_127.0.0.1_crm_postgres	postgresql_127.0.0.1_crm_postgres
postgresql_127.0.0.1_slp_postgres	postgresql_127.0.0.1_slp_postgres	postgresql_127.0.0.1_slp_postgres	postgresql_127.0.0.1_slp_postgres

Showing 1 to 5 of 5 entries

6 connections

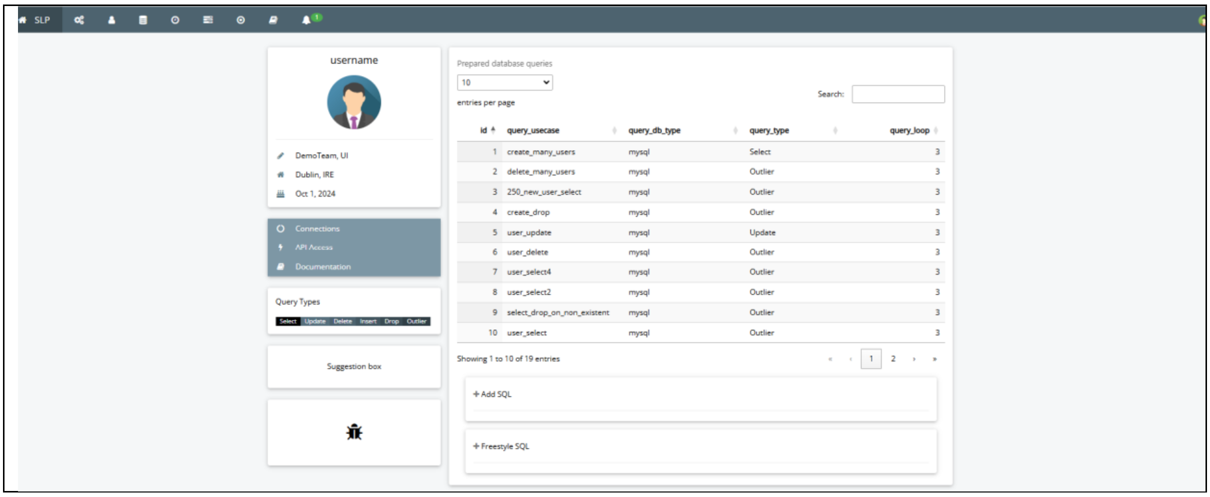
Support: [Jason Flood] [John Clarke]

Powered by theDemoTeam

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



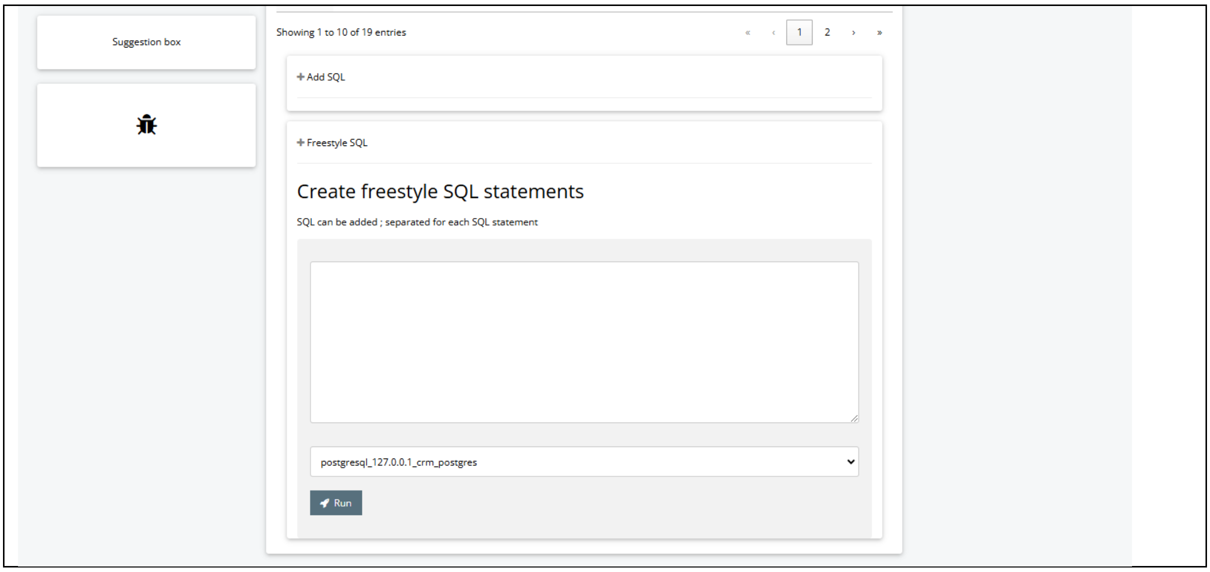
id	query_usecase	query_db_type	query_type	query_loop
1	create_many_users	mysql	Select	3
2	delete_many_users	mysql	Outlier	3
3	250_new_user_select	mysql	Outlier	3
4	create_drop	mysql	Outlier	3
5	user_update	mysql	Update	3
6	user_delete	mysql	Outlier	3
7	user_select4	mysql	Outlier	3
8	user_select2	mysql	Outlier	3
9	select_drop_on_non_existent	mysql	Outlier	3
10	user_select	mysql	Outlier	3

Showing 1 to 10 of 19 entries

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.



Showing 1 to 10 of 19 entries

Create freestyle SQL statements

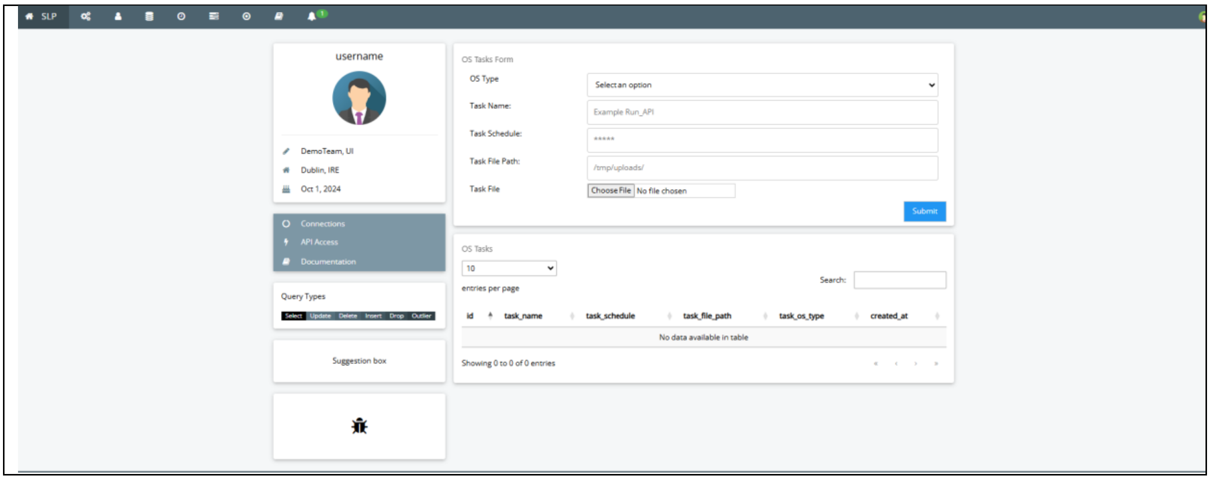
SQL can be added ; separated for each SQL statement

postgresql_127.0.0.1_crm_postgres

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



OS Tasks Form

OS Type	Select an option
Task Name:	Example Run_API
Task Schedule:	*****
Task File Path:	/tmp/uploads/
Task File	Choose File No file chosen

OS Tasks

10

entries per page

Search:

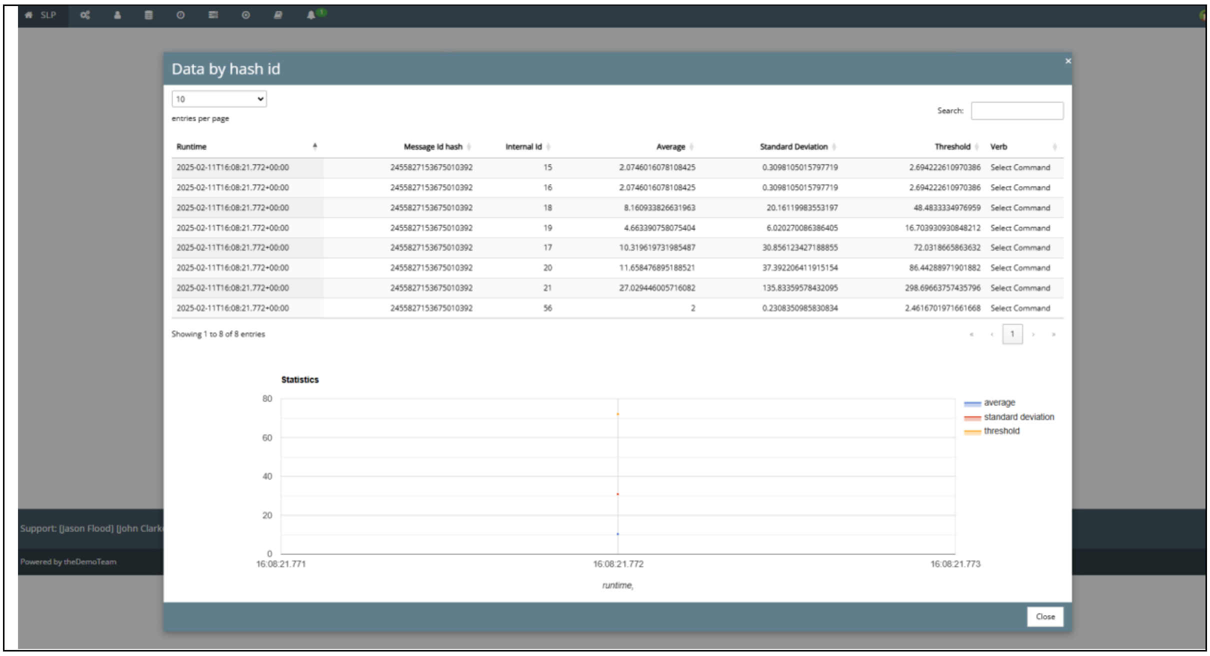
id	task_name	task_schedule	task_file_path	task_os_type	created_at
No data available in table					

Showing 0 to 0 of 0 entries

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



Data by hash id

10

entries per page

Search:

Runtime	Message Id hash	Internal Id	Average	Standard Deviation	Threshold	Verb
2025-02-11T16:08:21.772+00:00	2455827153675010392	15	2.0746016078108425	0.3098105015797719	2.694222610970386	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	16	2.0746016078108425	0.3098105015797719	2.694222610970386	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	18	8.160933826631963	20.16119983553197	48.4833334976959	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	19	4.663390758075404	6.020270086386405	16.703930930848212	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	17	10.319619731985487	30.856123427188855	72.0318665863632	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	20	11.658476895188521	37.392206411915154	86.44288971901882	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	21	27.029446005716082	135.83359578432095	298.69663757435796	Select Command
2025-02-11T16:08:21.772+00:00	2455827153675010392	56	2	0.2308350985830834	2.4616701971661668	Select Command

Showing 1 to 8 of 8 entries

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.

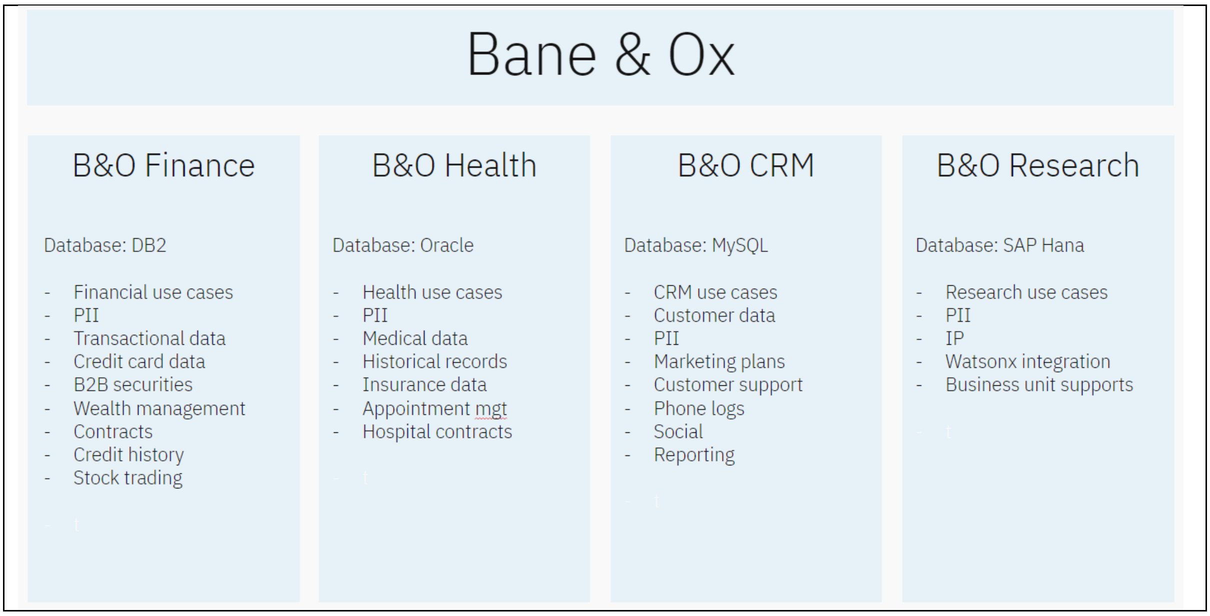
Use case	**Outliers type**	**Details**
Excessive data extraction	High volume	Verb= select
Excessive data modification (dml)	High volume	Verb= insert or update
Access from a new connection	New	Source program (new app) New OS New client IP
Excessive activity from a connection	High volume	Source program (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 **Error! Reference source not found.**. More BU's can be added to suit a specific customer Demo if needed.



Bane & Ox

B&O Finance

Database: DB2

- Financial use cases
- PII
- Transactional data
- Credit card data
- B2B securities
- Wealth management
- Contracts
- Credit history
- Stock trading

B&O Health

Database: Oracle

- Health use cases
- PII
- Medical data
- Historical records
- Insurance data
- Appointment mgt
- Hospital contracts

B&O CRM

Database: MySQL

- CRM use cases
- Customer data
- PII
- Marketing plans
- Customer support
- Phone logs
- Social
- Reporting

B&O Research

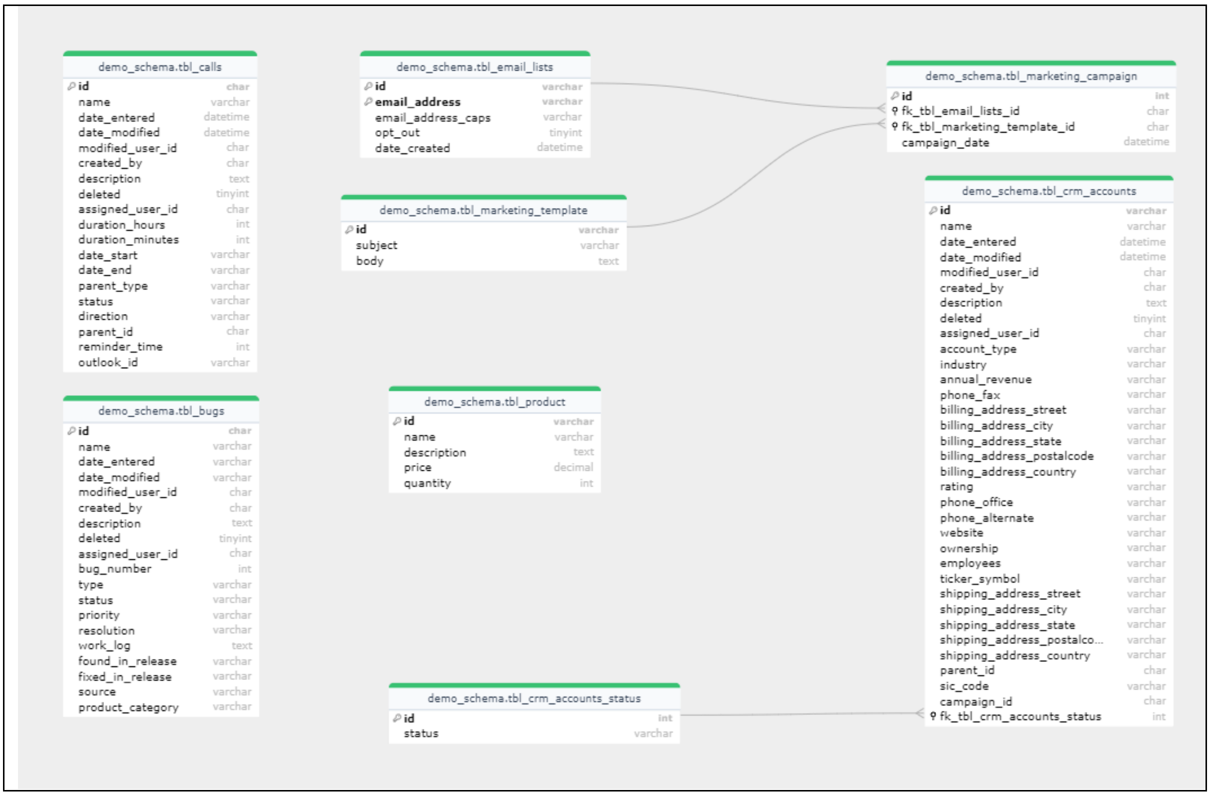
Database: SAP Hana

- Research use cases
- PII
- IP
- Watsonx integration
- Business unit supports

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 **Error! Reference source not found.**, show the tables and columns of the CRM BU.



demo_schema.tbl_calls	
id	char
name	varchar
date_entered	datetime
date_modified	datetime
modified_user_id	char
created_by	char
description	text
deleted	tinyint
assigned_user_id	char
duration_hours	int
duration_minutes	int
date_start	varchar
date_end	varchar
parent_type	varchar
status	varchar
direction	varchar
parent_id	char
reminder_time	int
outlook_id	varchar

demo_schema.tbl_email_lists	
id	varchar
email_address	varchar
email_address_caps	varchar
opt_out	tinyint
date_created	datetime

demo_schema.tbl_marketing_campaign	
id	int
fk_tbl_email_lists_id	char
fk_tbl_marketing_template_id	char
campaign_date	datetime

demo_schema.tbl_marketing_template	
id	varchar
subject	varchar
body	text

demo_schema.tbl_crm_accounts	
id	varchar
name	varchar
date_entered	datetime
date_modified	datetime
modified_user_id	char
created_by	char
description	text
deleted	tinyint
assigned_user_id	char
account_type	varchar
industry	varchar
annual_revenue	varchar
phone_fax	varchar
billing_address_street	varchar
billing_address_city	varchar
billing_address_state	varchar
billing_address_postalcode	varchar
billing_address_country	varchar
rating	varchar
phone_office	varchar
phone_alternate	varchar
website	varchar
ownership	varchar
employees	varchar
ticker_symbol	varchar
shipping_address_street	varchar
shipping_address_city	varchar
shipping_address_state	varchar
shipping_address_postalco...	varchar
shipping_address_country	varchar
parent_id	char
sic_code	varchar
campaign_id	char
fk_tbl_crm_accounts_status	int

demo_schema.tbl_bugs	
id	char
name	varchar
date_entered	varchar
date_modified	varchar
modified_user_id	char
created_by	char
description	text
deleted	tinyint
assigned_user_id	char
bug_number	int
type	varchar
status	varchar
priority	varchar
resolution	varchar
work_log	text
found_in_release	varchar
fixed_in_release	varchar
source	varchar
product_category	varchar

demo_schema.tbl_product	
id	varchar
name	varchar
description	text
price	decimal
quantity	int

demo_schema.tbl_crm_accounts_status	
id	int
status	varchar

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,N
ULL,'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,NUL
L,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

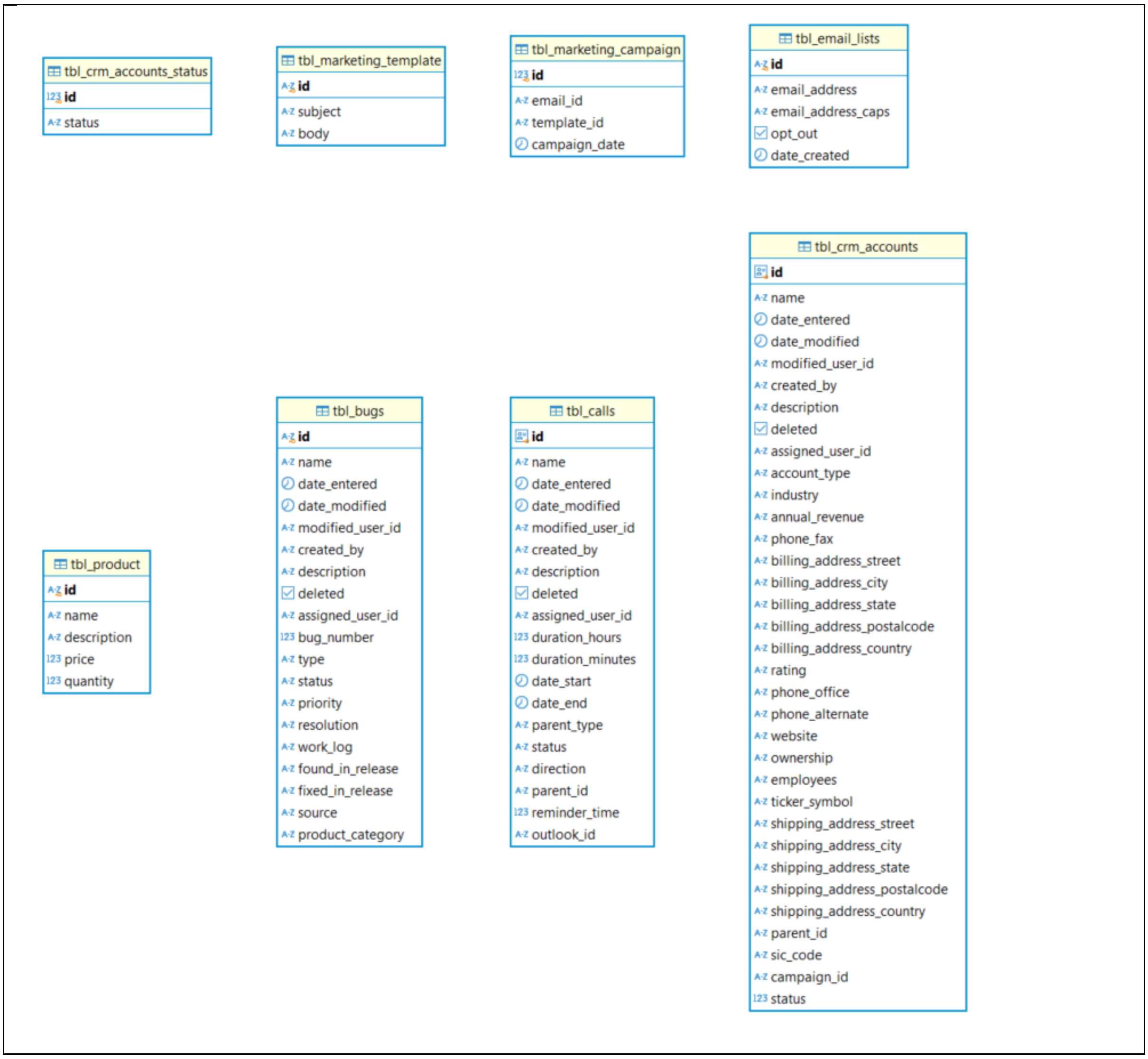


Figure 9 Bane & Ox CRM Postgres ERD

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 9, shows the tables and columns of the Finance BU.

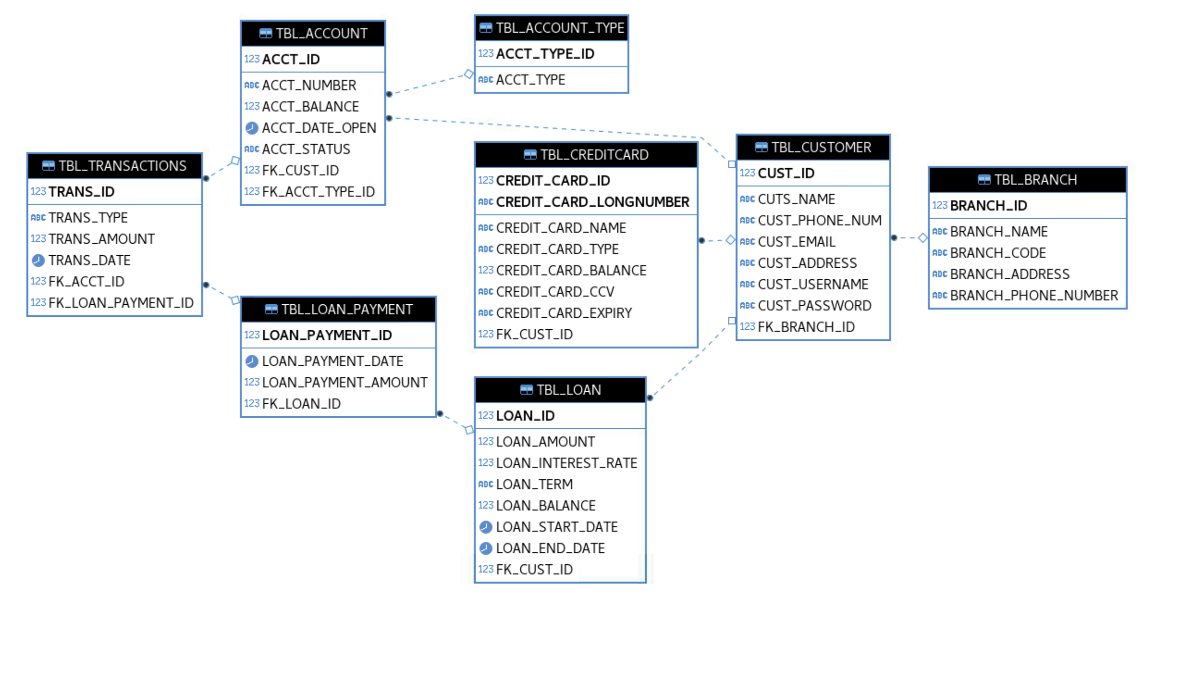


Figure 9 Bane & Ox Finance DB2 ERD

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

```sql
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

```sql
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

```sql
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,NUL
L,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

Appendix

Queries for tbl_crm_accounts

```sql
/* 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,N
ULL,'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
/* 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