

Student ID: 2136685

IM Coursework 2

Student ID: 2136685

Table of Contents

1. Case 1.....	3
2. Case 2.....	3
2.1. Login.....	3
2.2. Inserting a New Employee.....	4
2.2.1. Why Inserting a New Employee Might Fail.....	4
2.3. Changing a User's Email Address.....	4
2.4. Access to a User's Password.....	5
2.5. How to Secure Against SQL Injection Attacks.....	5
3. Case 3.....	5
3.1. Importance of Auditing.....	5
3.2. Overtime Manager Scenario.....	6
3.2.1. The Threats.....	6
3.2.2. Auditing Script.....	6
References.....	7
Appendix.....	7

Table of Figures

Figure 1: SQL Injection into login fields.....	7
Figure 2: SQL Injection on login was successful.....	7
Figure 3: Who each employee number is associated to.....	8
Figure 4: Choosing what user to login as.....	8
Figure 5: Logged in as number 3.....	8
Figure 6: SQL Injection for inserting a new employee.....	9
Figure 7: SQL Injection gives error.....	9
Figure 8: Attempting to login with created account.....	9
Figure 9: Account creation with SQL Injection succeeded.....	9
Figure 10: SQL Injection for seeing employee ID and email associated.....	10
Figure 11: Result of SQL Injection from Figure 10.....	10
Figure 12: SQL Injection for changing Martin Taylor's email.....	10
Figure 13: SQL Injection to show login table.....	11
Figure 14: Martin Taylor's email was changed successfully.....	11
Figure 15: SQL Injection for showing password based on an email.....	11
Figure 16: Result of SQL Injection in Figure 15 shows the password.....	12
Figure 17: Testing the password from Figure 16.....	12
Figure 18: The password was correct and the login was successful.....	12
Figure 19: Allow-list Input Validation (OWASP, 2021).....	12

Index of Tables

Table 1: Table showing all row-level and column-level security*	13
---	----

1. Case 1

In Table 1, I show the column-level and row-level security implemented for the “Employee” table. For the row-level security, the roles allowed to access the information can only do so with the views assigned to their roles. For example, a store manager can only access the views in the “store_manager” schema and none more, this is consistent to all of the roles defined.

The views take into account which user is logged into the database, therefore it can determine between the stores and show the employees of that specific store only. This is displayed in Table 1 and the employees are distinguished by the fact that store 2 employees have two digits in their data, for example “Employee 10” and “Address 10”, whereas store 1 employees would have only one digit in their name, for example “Employee 1” and “Address 1”.

To ensure continuity for future updates and perhaps alterations to the “Employee” table, for example adding another column, even if the finance manager can see all of the columns at the time, the finance manager role does not get access to the whole table and only to the columns specified.

Moreover, for the column-level security, all of the tables are set to a schema called “private”, which no role has access too, except for the “postgres” user.

2. Case 2

2.1. Login

To log into the website, an SQL injection attack, as seen in Figure 1, can be inputted in the username and password fields. As the fields expect an actual username and password, we can assume that the data entered will be used in an SQL statement in the form of “SELECT 1 FROM login WHERE username='<username>' AND password='<password>'”, therefore using the “OR 1=1” in the SQL injection statement, we force the statement to always show 1, as 1=1 is true, and true is 1. The “--” at the end shows that everything after the “--” is a comment, therefore making the “AND password='<password>'” part of the statement not get checked. The attack was successful and we were logged in as user “alice@warwick.ac.uk”, as seen in Figure 2.

On the other hand, on the /search page, we can just press the submit button and all the employees show up without being logged in, as seen in Figure 3. We assume that each number in front of each user is associated to that user.

Instead of using “OR 1=1”, as in Figure 1, we can try to use the statement “OR emp_id=3”, as seen in Figure 4, which should log us in as Martin Taylor. Seen in Figure 5, the test successfully logged me in as “martintaylor@warwick.ac.uk”, confirming that each number is associated with their user and is their emp_id.

2.2. Inserting a New Employee

To insert a new user, we can attempt an SQL injection with the statement “abcd' OR 1=1; INSERT INTO login VALUES (5, 'test5', 'test5');--”, seen in Figure 6. The “OR 1=1” part will take care of the login, whereas “INSERT INTO login VALUES (5, 'test5', 'test5');--” will actually insert the new user “test5”, with password “test5” and an emp_id of 5, into the “login” table. This should allow us to log in with the credentials “test5” and “test5”.

The SQL statement gives an error, seen in Figure 7. However, trying the credentials allows me to log into the website, as seen in Figures 8 and 9.

2.2.1. Why Inserting a New Employee Might Fail

If the website has protection against SQL Injection attacks, the new employee addition will fail. The website can have protections such as replacing spaces, for example “SELECT * FROM login ...” to “SELECT/**/ /**/FROM/**/login/**/...”. Also, the attack might fail if the website doesn’t use single quotes for SQL statements. For example, the website can use “WHERE username=\$test5\$” instead of “WHERE username='test5'”. Moreover, a website can use parameterized queries, where instead of putting the raw input into the SQL statement, the websites validates the user input and instead of, for example bypassing checks and commenting out statements with “SELECT 1 FROM login WHERE username= ‘abcd’ OR 1=1;--”, the website would instead attempt to query for someone with the username “abcd’ OR 1=1;--”.

2.3. Changing a User’s Email Address

As was seen in Figure 5, employee ID 3 is associated to Martin Taylor. Moreover, the “Welcome <email>” in Figure 5 shows the user’s email. Figure 10 and 11 confirm that it is indeed their email. I used the SQL injection “%'; SELECT emp_id, email FROM LOGIN; --” on the /search page to not have to fill two input forms. The search SQL statement uses a LIKE query to find anything that is related to the input form, therefore a “LIKE %<input>%” must have been used. Using the “%';” completes the statement.

I will change Martin Taylor’s email from “martintaylor@warwick.ac.uk” to “changed@email.com”. To do so, I will use the following SQL injection statement, “%'; UPDATE login SET email='changed@email.com' WHERE emp_id=3; --”. This changes the email linked to emp_id=3, which we confirmed to be associated to Martin Taylor in section 2.1. To confirm the change, I will check the “login” table with the SQL injection statement seen in Figure 12. Figure 13 confirms that the email was changed.

2.4. Access to a User's Password

Seen in Figure 15, I inserted the SQL injection statement “%'; SELECT password FROM login WHERE email='changed@email.com'; --” in the search field, which selects the password to the email we have just changed in section 2.3. This password would be Martin Taylor's password. The query returns “taylor” as the result, as seen in Figure 16. I tested the password in Figure 17 and it allowed me to log in to the account, seen in Figure 18, confirming that “taylor” is Martin Taylor's account.

2.5. How to Secure Against SQL Injection Attacks

A website owner can block the usage of single quotes by using statements such as “... WHERE email= \$\$changed@email.com\$\$;” instead of “... WHERE email='changed@email.com';”

Also, a website owner can block the usage of spaces. Instead of “SELECT email FROM login;” the statement would be “SELECT/**/email/**/FROM/**/login;” where “/**/” is equivalent to a space.

Moreover, a website owner can ensure that the input is validated by an allow-list. For simple queries, for example sorting by ascending or descending, one can make it so that the only allowed inputs are “ASC” or “DESC” and nothing else, as shown in Java in Figure 19 (OWASP, 2021).

The best defence against SQL Injection would be to use prepared statements with variable binding (parameterized queries) which make the developer define the SQL code structure first and then add each parameter needed to the query later. Using prepared statements allows the database to separate the code (the statement) to the data (user input), regardless of the input supplied. The statements don't allow attackers to change the intent of the query even if they would attempt to enter an SQL injection statement such as “%' OR 1=1;’--”, the SQL query would not be vulnerable as it would look for a username that matches the entire string “%' OR 1=1;’--” (OWASP, 2021).

3. Case 3

3.1. Importance of Auditing

Logging employee actions is crucial to the security of an organisation's information. Logging data such as when database users log in and out of the database or whenever they do an action on a table in a database, for example INSERT, SELECT, UPDATE, DROP etc., can help quickly identifying security incidents (e.g. database exports), policy violations (e.g. unauthorised deletions of tables), fraudulent activity (e.g. manufacturing records) and operational problems (e.g. improper storing of passwords). These logs can be brought together in an audit to be analysed to ensure compliance with the laws and regulations of the government, and to ensure that certain security standard specifications, for example ISO standard certifications, are still in accordance to the governing body's specification (OWASP, 2009).

3.2. Overtime Manager Scenario

3.2.1. The Threats

If a manager was recorded as in their office late one night, this can be interpreted multiple ways. If the manager was recorded through access card logs, then a threat actor might have stolen or cloned their access card, or the manager themselves was there late. If the manager was recorded in the office with CCTV, then we can pinpoint this access to the manager. Moreover, if they were recorded through the database login logs, perhaps they forgot to clear their username and someone else with the clerical staff password tried to input it but they couldn't access the database due to mismatched credentials - whether the password access is malicious or not is inconclusive, as this could have been an actual clerical staff or it could have been a threat actor. In addition, I don't know if being in the office past 10 PM is unusual for the employees, because this detail is not disclosed in the scenario.

Overall, this could be an insider threat attempting to, for example, sell data. Also, it could be an outside entity - a threat actor - with access to the clerical staff password and the manager computer, or username (not specified in the scenario). On the other hand, it could be a legitimate attempt to retrieve data for work by a clerical staff. I can not say for certain that this is a malicious access attempt or a mistake.

3.2.2. Auditing Script

The auditing script is presented in the zip file, in the "C3" folder. The only instructions for the script are that the one running the script replaces their "postgresql.conf" file with the one provided, to enable logging effectively. Secondly, the script run with sudo to allow the restart of the postgresql service and to allow for chmod of the log folder, so that the log can be copied to the folder with the script.

References

OWASP (2009). *OWASP Logging Guide SUMMARY*. [online] Available at: https://owasp.org/www-pdf-archive/OWASP_Logging_Guide.pdf [Accessed 14 May 2023].

OWASP (2021). *SQL Injection Prevention · OWASP Cheat Sheet Series*. [online] Owasp.org. Available at: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html [Accessed 12 May 2023].

Appendix

Enter login details

Enter user name

Enter password

Note: Your email is your username.

Figure 1: SQL Injection into login fields

Welcome alice@warwick.ac.uk

1	Alice Smith	22 Baker Street	1954-01-06	123456	123456789	650000
2	Robert Holmes	2 morvenside Street	1994-01-01	789976	567891234	450000
3	Martin Taylor	34 down Street	1988-10-05	546789	678976584	250000

Figure 2: SQL Injection on login was successful

Employee

1	Alice Smith	22 Baker Street	1954-01-06	123456	123456789	650000
2	Robert Holmes	2 morvenside Street	1994-01-01	789976	567891234	450000
3	Martin Taylor	34 down Street	1988-10-05	546789	678976584	250000

Figure 3: Who each employee number is associated to

Enter login details

Enter user name

Enter password

Note: Your email is your username.

Figure 4: Choosing what user to login as

Welcome martintaylor@warwick.ac.uk

1	Alice Smith	22 Baker Street	1954-01-06	123456	123456789	650000
2	Robert Holmes	2 morvenside Street	1994-01-01	789976	567891234	450000
3	Martin Taylor	34 down Street	1988-10-05	546789	678976584	250000

Figure 5: Logged in as number 3

Enter login details

Enter user name

Enter password

Note: Your email is your username.

Figure 6: SQL Injection for inserting a new employee

ProgrammingError

psycopg2.ProgrammingError: no results to fetch

Traceback (most recent call last)

Figure 7: SQL Injection gives error

Enter login details

Enter user name

Enter password

Note: Your email is your username.

Figure 8: Attempting to login with created account

Welcome test5

1	Alice Smith	22 Baker Street	1954-01-06	123456	123456789	650000
2	Robert Holmes	2 morvenside Street	1994-01-01	789976	567891234	450000
3	Martin Taylor	34 down Street	1988-10-05	546789	678976584	250000

Figure 9: Account creation with SQL Injection succeeded

Enter search parameters

Use the searchstr form variable to search for products.

Enter search string

Figure 10: SQL Injection for seeing employee ID and email associated

Employee

1	alice@warwick.ac.uk
2	bobholmes@warwick.ac.uk
5	test5
3	martintaylor@warwick.ac.uk

Figure 11: Result of SQL Injection from Figure 10

Enter search parameters

Use the searchstr form variable to search for products.

Enter search string

Figure 12: SQL Injection for changing Martin Taylor's email

Enter search parameters

Use the searchstr form variable to search for products.

Enter search string

Figure 13: SQL Injection to show login table

Employee

1	alice@warwick.ac.uk	alicebaker				
2	bobholmes@warwick.ac.uk	holmesdrive				
5	test5	test5				
3	changed@email.com	taylor				

Figure 14: Martin Taylor's email was changed successfully

Enter search parameters

Use the searchstr form variable to search for products.

Enter search string

Figure 15: SQL Injection for showing password based on an email

Employee

Figure 16: Result of SQL Injection in Figure 15 shows the password

Enter login details

Enter user name

Enter password

Note: Your email is your username.

Figure 17: Testing the password from Figure 16

Welcome changed@email.com

1	Alice Smith	22 Baker Street	1954-01-06	123456	123456789	650000
2	Robert Holmes	2 morvenside Street	1994-01-01	789976	567891234	450000
3	Martin Taylor	34 down Street	1988-10-05	546789	678976584	250000

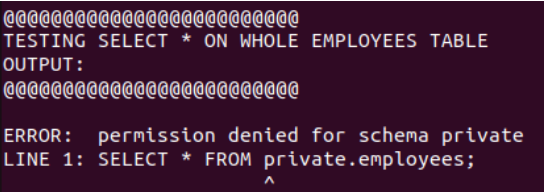
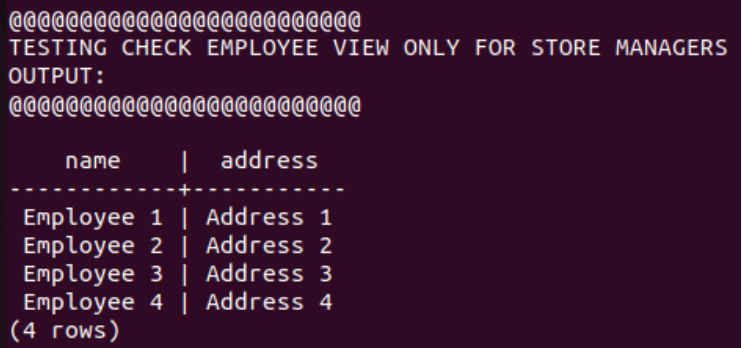
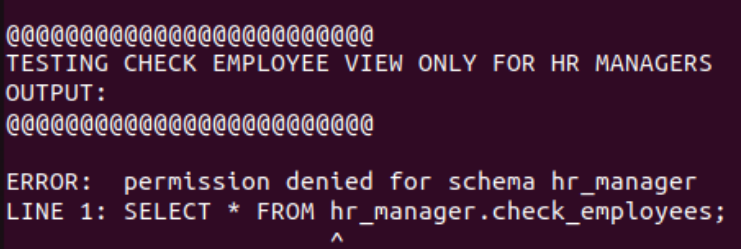
Figure 18: The password was correct and the login was successful

```
public String someMethod(boolean sortOrder) {  
    String SQLquery = "some SQL ... order by Salary " + (sortOrder ? "ASC" : "DESC");  
    ...  
}
```

Figure 19: Allow-list Input Validation (OWASP, 2021)

Table 1: Table showing all row-level and column-level security*

*Store 1 employees have ONE digit in their data whereas store 2 employees have TWO.

Employee Role	Description of Test	Screenshot
Store Manager (Store 1)	Store manager (store 1) attempting to access the employees table	 <pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Store manager (store 1) attempting to access the check employee view for store managers	 <pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name address -----+----- Employee 1 Address 1 Employee 2 Address 2 Employee 3 Address 3 Employee 4 Address 4 (4 rows) </pre>
	Store manager (store 1) attempting to access the check employee view for HR managers	 <pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>

	Store manager (store 1) attempting to access the check employee view for admins	<pre> @@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	Store manager (store 1) attempting to access the check employee view for finance managers	<pre> @@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	Store manager (store 1) attempting to access the check employee view for area managers	<pre> @@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
Store Manager (Store 2)	Store manager (store 2) attempting to access the employees table	<pre> @@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Store manager (store 2) attempting to access the check employee view for store managers	<pre> @@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@ name address -----+----- Employee 10 Address 10 Employee 20 Address 20 Employee 30 Address 30 Employee 40 Address 40 (4 rows) </pre>

	Store manager (store 2) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Store manager (store 2) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	Store manager (store 2) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	Store manager (store 2) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
HR Manager (Store 1)	HR manager (store 1) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	HR manager (store 1) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>

	HR manager (store 1) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name address date_of_birth salary -----+-----+-----+----- Employee 1 Address 1 1976-12-12 30000 Employee 2 Address 2 1997-02-02 40700 Employee 3 Address 3 1985-10-10 50670 Employee 4 Address 4 1986-07-30 64660 (4 rows) </pre>
	HR manager (store 1) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	HR manager (store 1) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	HR manager (store 1) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
HR Manager (Store 2)	HR manager (store 2) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>

	HR manager (store 2) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	HR manager (store 2) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name address date_of_birth salary -----+-----+-----+----- Employee 10 Address 10 1976-12-12 40000 Employee 20 Address 20 1997-02-02 50700 Employee 30 Address 30 1985-10-10 60670 Employee 40 Address 40 1986-07-30 74660 (4 rows) </pre>
	HR manager (store 2) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	HR manager (store 2) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	HR manager (store 2) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>

Admin (Store 1)	Admin (store 1) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Admin (store 1) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	Admin (store 1) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Admin (store 1) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name ----- Employee 1 Employee 2 Employee 3 Employee 4 (4 rows) </pre>
	Admin (store 1) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>

	Admin (store 1) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
Admin (Store 2)	Admin (Store 2) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Admin (Store 2) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	Admin (Store 2) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Admin (Store 2) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name ----- Employee 10 Employee 20 Employee 30 Employee 40 (4 rows) </pre>

	Admin (Store 2) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	Admin (Store 2) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
Finance Manager (Store 1)	Finance manager (store 1) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Finance manager (store 1) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	Finance manager (store 1) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>

	Finance manager (store 1) attempting to access the check employee view for admins	<pre>@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^</pre>
	Finance manager (store 1) attempting to access the check employee view for finance managers	<pre>@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@ name address date_of_birth sort_code bank_account_number salary ----- ----- ----- ----- ----- ----- Employee 1 Address 1 1976-12-12 123456 65765865 30000 Employee 2 Address 2 1997-02-02 123456 87745455 40700 Employee 3 Address 3 1985-10-10 123456 35765987 50670 Employee 4 Address 4 1986-07-30 123456 65765472 64660 (4 rows)</pre>
	Finance manager (store 1) attempting to access the check employee view for area managers	<pre>@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^</pre>
Finance Manager (Store 2)	Finance manager (store 2) attempting to access the employees table	<pre>@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^</pre>
	Finance manager (store 2) attempting to access the check employee view for store managers	<pre>@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^</pre>

	Finance manager (store 2) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Finance manager (store 2) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	Finance manager (store 2) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name address date_of_birth sort_code bank_account_number salary -----+-----+-----+-----+-----+----- Employee 10 Address 10 1976-12-12 123456 65765665 40000 Employee 20 Address 20 1997-02-02 123456 87745255 50700 Employee 30 Address 30 1985-10-10 123456 35765287 60670 Employee 40 Address 40 1986-07-30 123456 65765172 74660 (4 rows) </pre>
	Finance manager (store 2) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema area_manager LINE 1: SELECT * FROM area_manager.check_employees; ^ </pre>
Area Manager (Store 1)	Area manager (store 1) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>

	Area manager (store 1) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	Area manager (store 1) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Area manager (store 1) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	Area manager (store 1) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>
	Area manager (store 1) attempting to access the check employee view for area managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ name address -----+----- Employee 1 Address 1 Employee 2 Address 2 Employee 3 Address 3 Employee 4 Address 4 (4 rows) </pre>

Area Manager (Store 2)	Area manager (store 2) attempting to access the employees table	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING SELECT * ON WHOLE EMPLOYEES TABLE OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema private LINE 1: SELECT * FROM private.employees; ^ </pre>
	Area manager (store 2) attempting to access the check employee view for store managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR STORE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema store_manager LINE 1: SELECT * FROM store_manager.check_employees; ^ </pre>
	Area manager (store 2) attempting to access the check employee view for HR managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR HR MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema hr_manager LINE 1: SELECT * FROM hr_manager.check_employees; ^ </pre>
	Area manager (store 2) attempting to access the check employee view for admins	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR ADMINS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema admin LINE 1: SELECT * FROM admin.check_employees; ^ </pre>
	Area manager (store 2) attempting to access the check employee view for finance managers	<pre> @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TESTING CHECK EMPLOYEE VIEW ONLY FOR FINANCE MANAGERS OUTPUT: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ ERROR: permission denied for schema finance_manager LINE 1: SELECT * FROM finance_manager.check_employees; ^ </pre>

Area manager (store 2) attempting to access the check employee view for area managers

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
TESTING CHECK EMPLOYEE VIEW ONLY FOR AREA MANAGERS
OUTPUT:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

      name      | address
-----+-----
Employee 10 | Address 10
Employee 20 | Address 20
Employee 30 | Address 30
Employee 40 | Address 40
(4 rows)

```