Output Data

SQL injection attacks can occur to a poorly protected database. An SQL injection is where a database’s queries are interfered with by an attacker. The attacker will take advantage of any web security vulnerabilities in the database and attempt to sabotage or manipulate the existing data. A lack of proper coding can make a person vulnerable to a SQL injection. Parameters, field values and cookies can all be vulnerable to attack. One measure to combat SQL attacks is to have stored procedures. Stored procedures are SQL statements that produce a logical unit to perform a specific task. Like tables, they are database objects. Advantages of stored procedures include effective validation, encapsulation of business logic in a single entity and faster exception handling. A stored procedure is used in the database for its protection.

# Queries

## Query 1 | 2.2

SELECT Orders.Order\_Time, Customer.\*

FROM Orders

RIGHT JOIN Customer ON Orders.Customer\_Email = Customer.Customer\_Email;

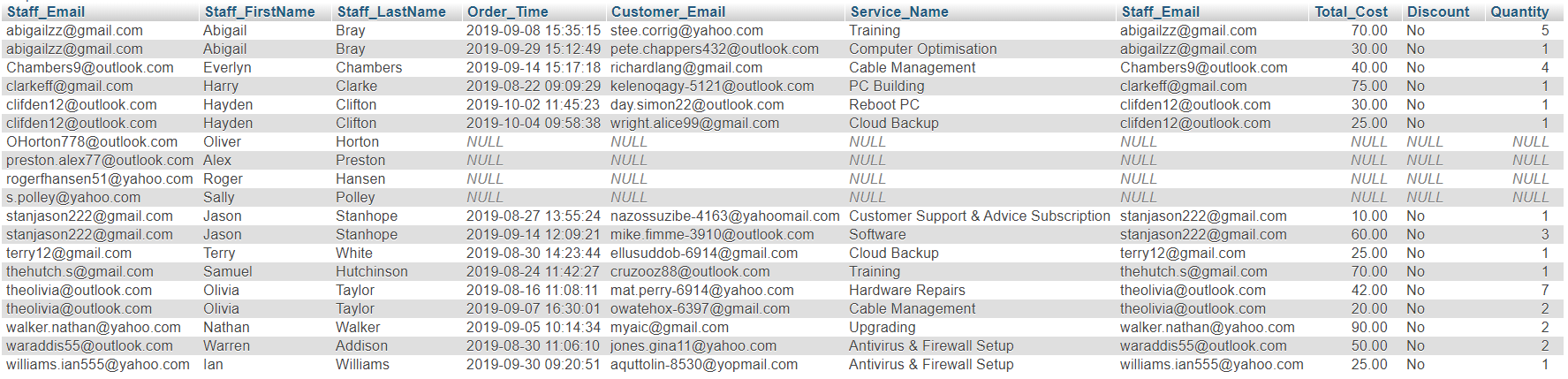


## Query 2 | 2.3

SELECT Staff.\*, Orders.\*

FROM Staff

LEFT JOIN Orders ON Staff.Staff\_Email = Orders.Staff\_Email;

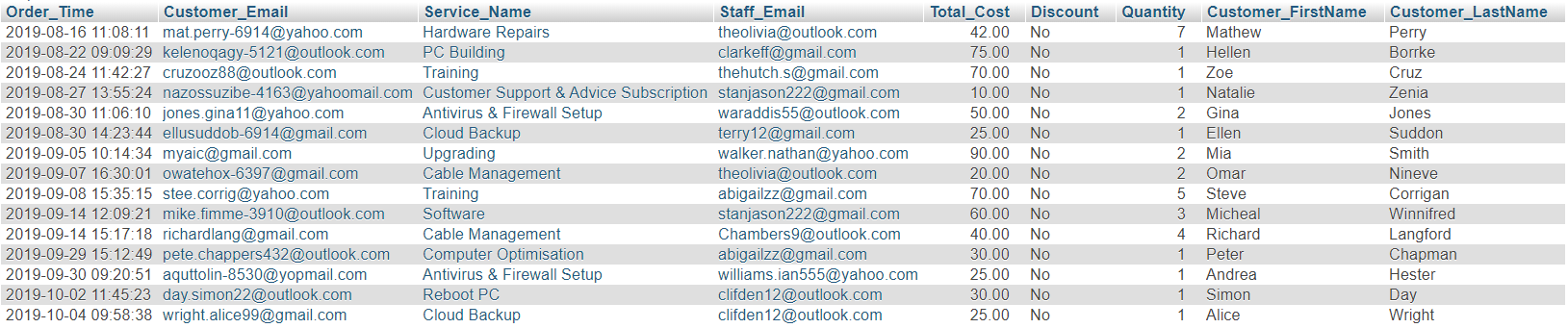


## Query 3 | 2.4

SELECT Orders.\*, Customer.Customer\_FirstName, Customer.Customer\_LastName

FROM Orders

INNER JOIN Customer ON Orders.Customer\_Email = Customer.Customer\_Email;



# Stored Procedure

DELIMITER $$

CREATE PROCEDURE `selectOrders`() NOT DETERMINISTIC CONTAINS SQL SQL SECURITY DEFINER SELECT \* FROM Orders;

CALL `selectOrders`();

SET @p0='2019-09-05 10:14:34';

CALL `selectOrders`(@p0, @p1);

SELECT @p1 AS `Quantity2`;

END$$

DELIMITER;

Call selectOrders;

The main advantage of this stored procedure is that it requires the input of a specific date and time to access the quantity of the order. This makes it less likely to be accessed by an intruder as they are unlikely to have the exact date and time of the order they want to alter. The images below show its results:

