**SQL Statements:**

Customer Table:

FILL:

SELECT Customer\_PhoneNo, Customer\_LastName, Customer\_FirstName, Customer\_Email, Customer\_Points

FROM dbo.tblCustomer

The above sql query selects all customer information from tblCustomers. This allows us to view tblCustomer in its entirety which can prove very useful.

FILLBYPHONENUMBER:

SELECT Customer\_PhoneNo, Customer\_LastName, Customer\_FirstName, Customer\_Email, Customer\_Points

FROM tblCustomer

WHERE (Customer\_PhoneNo LIKE @Param1)

The above sql query selects all customer information from tblCustomers where the customers phone number is similar to a specified phone number. This can prove useful when used as a search as it allows us to search for customers by there phone number. This is useful as it will save time when using the system.

GETLOYALTYPOINTS:

SELECT Customer\_Points

FROM tblCustomer

WHERE (Customer\_PhoneNo = @Param1)

The above sql query returns a customers’ loyalty points based on their phone number. This is useful as it allows us to view and manipulate a customers’ loyalty points.

GETNUMBERCUSTOMERS:

SELECT COUNT(\*) AS Expr1

FROM tblCustomer

WHERE (Customer\_PhoneNo = @Param1)

The above sql query counts the number of customers where their number is equal to the one stored in the database. This is useful as we can use the value that it returns to determine if the customer exists in the database and if we need to add it to it.

SETPOINTSTOZERO:

UPDATE tblCustomer

SET Customer\_Points = 0

The above sql query sets all customers’ loyalty points to zero. This is useful as it allows us to set the customers’ loyalty points to zero when a new year begins.

UPDATELOYALTYPOINTS:

UPDATE tblCustomer

SET Customer\_Points = @Param1

WHERE (Customer\_PhoneNo = @Param2)

The above sql query allows us to update customers’ loyalty points based on their phone number. This is useful as it allows us to update a customers loyalty points after a transaction or during a point later in time.

Invoice Table:

FILL:

SELECT Invoice\_No, Invoice\_Amount, Amount\_Paid, Invoice\_Date, Staff\_ID

FROM dbo.tblInvoice

The above sql query selects all invoice information from tblInvoice. This allows us to view tblInvoice in its entirety which can prove very useful.

FILLBYINVOICENUMBER:

SELECT Invoice\_No, Invoice\_Amount, Amount\_Paid, Invoice\_Date, Staff\_ID

FROM tblInvoice

WHERE (Invoice\_No LIKE @Param1)

The above sql query selects all invoice information from table based on a specified value similar to the Invoice\_No stored it tblInvoice. This is useful as it allows us to search and view invoice information based on a specified invoice number.

GETAVERAGEAMOUNTPAID:

SELECT AVG(Amount\_Paid)

FROM tblInvoice

The above sql statement returns the average amount paid by the customer during a transaction. This is useful as it can be used to work out how much change on average the business is giving to customers which can then determine how large a float they should keep at a time.

GETAVERAGEDATE:

SELECT CAST(AVG(CAST(Invoice\_Date AS FLOAT)) AS DATETIME) AS Expr1

FROM tblInvoice

The above sql query returns the average date of which the majority of transaction occurred. This is useful as it allows the business owners to see when their business is most productive and can thus make sure more staff and products are available on those days.

GETAVERAGEINVOICEAMOUNT:

SELECT AVG(Invoice\_Amount)

FROM tblInvoice

The above sql query returns the average amount per transaction from tblInvoice. This is useful as it allows the business owners to see how much they are earning on average per transaction.

GETNUMBEROFTRANSACTIONS:

SELECT COUNT(\*)

FROM tblInvoice

The above sql query returns the number of total transactions processed which can prove useful when evaluating business performance.

INSERTNEWINVOICE:

INSERT INTO [dbo].[tblInvoice] ([Invoice\_Amount], [Amount\_Paid], [Invoice\_Date], [Staff\_ID]) VALUES (@Invoice\_Amount, @Amount\_Paid, @Invoice\_Date, @Staff\_ID);

The above sql statement allows us to insert a new row into table invoice which is useful as it allow as to add a new transaction.

Staff Table:

FILL:

SELECT Staff\_ID, Staff\_LastName, Staff\_FirstName, Staff\_Passcode, Staff\_Role

FROM dbo.tblStaff

The above sql query selects all staff information from tblStaff. This allows us to view tblCustomer in its entirety which can prove very useful.

CHECKLOGIN:

SELECT COUNT(\*) AS Expr1

FROM tblStaff

WHERE (Staff\_ID = @Param1) AND (Staff\_Passcode = @Param2)

The above sql query allows us to check if login information is correct and if so proceed with the program.

FILLBYNAME:

SELECT Staff\_ID, Staff\_LastName, Staff\_FirstName, Staff\_Passcode, Staff\_Role

FROM tblStaff

WHERE (Staff\_LastName LIKE @Param1) OR

(Staff\_FirstName LIKE @Param2)

The above sql query selects all staff information where Staff\_LastName or Staff\_FirstName is similar to a specified value. This allows us to search for a specific employee based on either their first or last name.

GETMOSTPRODUCTIVEEMPLOYEE:

SELECT Staff\_ID

FROM tblStaff

WHERE (Staff\_ID =

(SELECT AVG(Staff\_ID) AS Expr1

FROM tblInvoice))

The above sql query returns the employee with the most transactions processed. This is useful as the managers/owners can use it to determine the most productive employee.

GETROLE:

SELECT Staff\_Role

FROM tblStaff

WHERE (Staff\_ID = @Param1)

The above sql statement returns the role of a specified staff member based on their Staff\_ID. This is useful as we can use it to determine the access level the user of the program has in regard to the program.

GETSTAFFFIRSTNAME:

SELECT Staff\_FirstName

FROM tblStaff

WHERE (Staff\_ID = @Param1)

The above sql statement returns the first name of a specified staff member based on their Staff\_ID.

GETSTAFFLASTNAME:

SELECT Staff\_LastName

FROM tblStaff

WHERE (Staff\_ID = @Param1)

The above sql statement returns the last name of a specified staff member based on their Staff\_ID.

Product Table:

FILL:

SELECT Product\_ID, Product\_Description, Price, Amount\_Sold

FROM dbo.tblProduct

The above sql statement selects all product information from tblProduct. This allows us to view tbProduct in its entirety which can prove very useful.

FILLBYDESCRIPTION:

SELECT Product\_ID, Product\_Description, Price, Amount\_Sold

FROM tblProduct

WHERE (Product\_Description LIKE @Param1)

The above sql statement selects all product information based on a specified value similar to Product\_Description. This is useful as it allows us to search for a specific product based on it description.