ERM Initial Concept Sketches

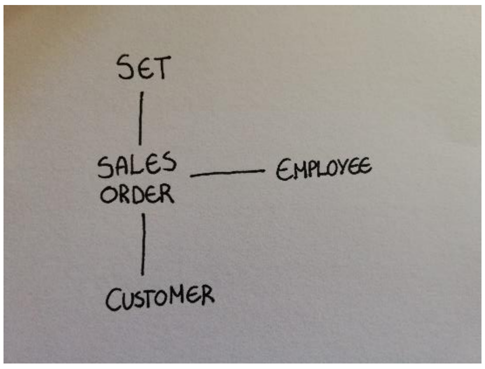


Figure 1 - Customer order processing sketch.

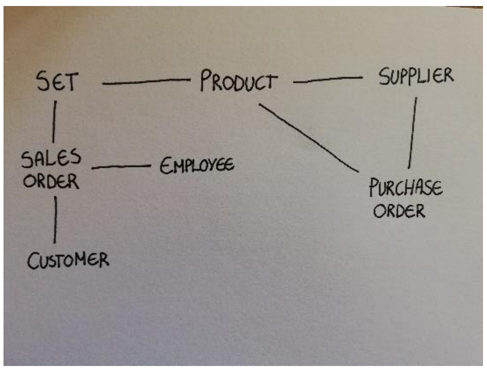


Figure 2 - Full system sketch.

Full System ERM

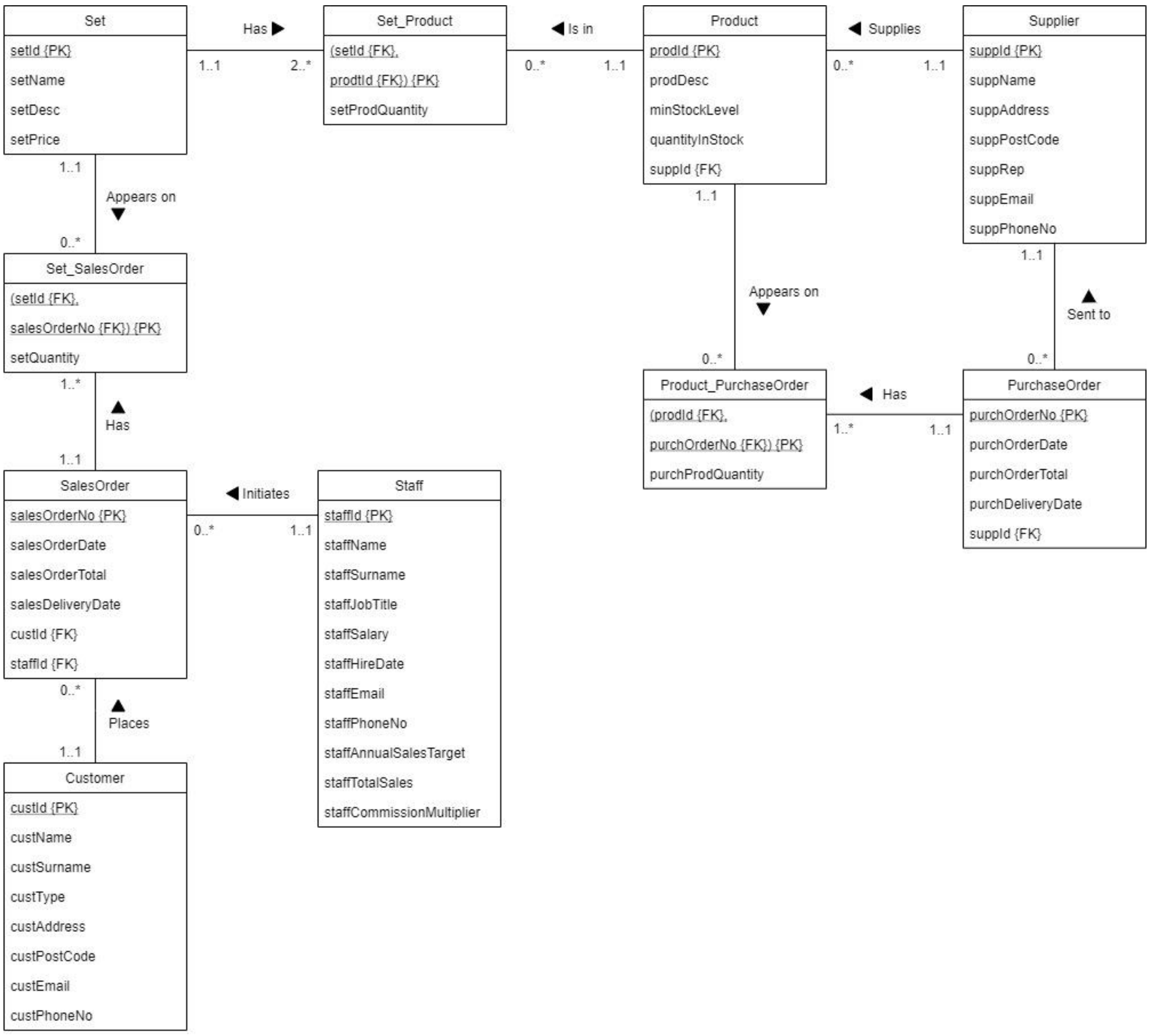


Figure 3 - Full system ERM.

Entity	**Primary Key {PK}**	**Foreign Keys {FK}**	**Rationale**
Customer	custId	N/A	The customers need a unique identifier to avoid any ambiguity/duplication during the ordering process. An alternative could be a combination of the name, surname and address, although a unique, numeric ID is more scalable and practical for database applications.
Staff	staffId	N/A	Similar reasons to customer ID, it is a unique identifier for every staff member.
SalesOrder	salesOrderNo	custID, staffID	Unique number for quick referencing/identification of individual orders. Contains foreign keys for customers; as they place the orders, and for staff; as sales are made by the sales team (who also get commissions from them).
Set	setId	N/A	Set names/descriptions may have commonalities, as such a unique identifier is needed for the primary key.
Set_SalesOrder	(setId, salesOrderNo)	setId, salesOrderNo	This entity was created to resolve the many-to-many multiplicity between Set and Sales Order. As such, the primary key is created from a combination of the foreign keys from Set and Sales Order.
Product	prodId	suppId	The company mentioned in the case study that they give products unique id numbers, so I thought that these would be appropriate for the primary key. A foreign key from Supplier is present, as each product is supplied by a supplier.
Set_Product	(setId, prodId)	setId, prodId	Once again, an entity created to resolve a many-to-many multiplicity. Primary key created from the foreign keys from Set and Product.

Supplier	suppId	N/A	Similar rationale to Customer and Staff, a unique identifier for every individual supplier.
PurchaseOrder	purchOrderNo	suppId	Similar rationale to SalesOrder, a unique reference number for every purchase order. A foreign key linking to Supplier, as that is where the purchase orders get sent.
Product_PurchaseOrder	(prodId, purchOrderNo)	prodId, purchOrderNo	Resolving another many-to-many multiplicity between product and purchase order. Primary key created from their foreign keys.

SQL Entire Script Link

Here is a link to the entire script for Table Creation, Data Insertion and 4 Example Queries:
https://pastecode.xyz/view/c5bdfd10

It may be easier to view in that format than as pasted below.

SQL Table Creation Script

```
/*-----Drops all tables if they already exist-----*/

DROP TABLE IF EXISTS Product_PurchaseOrder;

DROP TABLE IF EXISTS BoxSet_Product;

DROP TABLE IF EXISTS BoxSet_SalesOrder;

Drop TABLE IF EXISTS PurchaseOrder;

Drop TABLE IF EXISTS SalesOrder;

Drop TABLE IF EXISTS Product;

Drop TABLE IF EXISTS Supplier;

Drop TABLE IF EXISTS BoxSet;

Drop TABLE IF EXISTS Staff;

Drop TABLE IF EXISTS Customer;


/*-----Creates the tables and constraints-----*/

CREATE TABLE Customer (

        custId INTEGER(6),

        custName VARCHAR(40) NOT NULL,
	custSurname VARCHAR(40) NOT NULL,

	custType VARCHAR(20) NOT NULL,

	custAddress VARCHAR(100) NOT NULL,

	custPostCode VARCHAR(12) NOT NULL,

	custEmail VARCHAR(100) NOT NULL,

	custPhoneNo VARCHAR(50),

	CONSTRAINT customer_custId_pk PRIMARY KEY (custId)

);


CREATE TABLE Staff (

	staffId INTEGER(6),

	staffName VARCHAR(40) NOT NULL,

	staffSurname VARCHAR(40) NOT NULL,

	staffJobTitle VARCHAR(80) NOT NULL,

	staffSalary DECIMAL(8,2) NOT NULL,

	staffHireDate DATE NOT NULL,

	staffEmail VARCHAR(100) NOT NULL,

	staffPhoneNo VARCHAR(50) NOT NULL,

	staffAnnualSalesTarget DECIMAL(10,2),

	staffTotalSales DECIMAL(10,2),

	staffCommisionMultiplier DECIMAL(3,2),

	CONSTRAINT staff_staffId_pk PRIMARY KEY (staffId)

);


/*"Set" is a reserved keyword, as such, i've renamed the Entity to "BoxSet" instead.*/

CREATE TABLE BoxSet (

	setId INTEGER(6),

	setName VARCHAR(80) NOT NULL,

	setDesc VARCHAR(1200),

	setPrice DECIMAL(6,2) NOT NULL,

	CONSTRAINT set_setId_pk PRIMARY KEY (setId)
```

```
);

CREATE TABLE Supplier (
        suppId INTEGER(6),
        suppName VARCHAR(80) NOT NULL,
        suppAddress VARCHAR(100) NOT NUll,
        suppPostCode VARCHAR(12) NOT NULL,
        suppRep VARCHAR(80),
        suppEmail VARCHAR(100) NOT NULL,
        suppPhoneNo VARCHAR(50) NOT NULL,
        CONSTRAINT supplier_suppId_pk PRIMARY KEY (suppId)
);

CREATE TABLE Product (
        prodId INTEGER(6),
        prodDesc VARCHAR(800),
        minStockLevel INTEGER(4) NOT NULL,
        quantityInStock INTEGER(6)NOT NULL,
        suppId INTEGER(6) NOT NULL,
        CONSTRAINT product_prodId_pk PRIMARY KEY (prodId),
        CONSTRAINT product_suppId_fk FOREIGN KEY (suppId)
        REFERENCES Supplier (suppId)
);

CREATE TABLE SalesOrder (
        salesOrderNo INTEGER(10),
        salesOrderDate DATE NOT NULL,
        salesOrderTotal DECIMAL(10,2) NOT NULL,
        salesDeliveryDate DATE,
        custId INTEGER(6) NOT NULL,
        staffId INTEGER(6),
        CONSTRAINT salesOrder_salesOrderNo_pk PRIMARY KEY (salesOrderNo),
        CONSTRAINT salesOrder_custId_fk FOREIGN KEY (custId)
        REFERENCES Customer (custId),
        CONSTRAINT salesOrder_staffId_fk FOREIGN KEY (staffId)
        REFERENCES Staff (staffId)
);

CREATE TABLE PurchaseOrder (
        purchOrderNo INTEGER(10),
        purchOrderDate DATE NOT NULL,
        purchOrderTotal DECIMAL(10,2) NOT NULL,
        purchDeliveryDate DATE,
        suppId INTEGER(6) NOT NULL,
        CONSTRAINT purchaseOrder_purchOrderNo_pk PRIMARY KEY (purchOrderNo),
        CONSTRAINT purchaseOrder_suppId_fk FOREIGN KEY (suppId)
        REFERENCES Supplier (suppId)
);

CREATE TABLE BoxSet_SalesOrder (
        setId INTEGER(6),
        salesOrderNo INTEGER(10),
        setQuantity INTEGER(4) NOT NULL,
        CONSTRAINT setSalesOrder_setId_salesOrderNo_pk PRIMARY KEY (setId, salesOrderNo),
        CONSTRAINT setSalesOrder_setId_fk FOREIGN KEY (setId)
        REFERENCES BoxSet (setId),
        CONSTRAINT setSalesOrder_salesOrderNo_fk FOREIGN KEY (salesOrderNo)
        REFERENCES SalesOrder (salesOrderNo)
);

CREATE TABLE BoxSet_Product (
        setId INTEGER(6),
	prodId INTEGER(6),
	setProdQuantity INTEGER(4) NOT NULL,
	CONSTRAINT setProduct_setId_prodId_pk PRIMARY KEY (setId, prodId),
	CONSTRAINT setProduct_setId_fk FOREIGN KEY (setId)
	REFERENCES BoxSet (setId),
	CONSTRAINT setProduct_prodId_fk FOREIGN KEY (prodId)
	REFERENCES Product (prodId)
);

CREATE TABLE Product_PurchaseOrder (
	prodId INTEGER(6),
	purchOrderNo INTEGER(10),
	purchOrderQuantity INTEGER(4) NOT NULL,
	CONSTRAINT productPurchaseOrder_prodId_purchOrderNo_pk PRIMARY KEY (prodId,
purchOrderNo),
	CONSTRAINT productPurchaseOrder_prodId_fk FOREIGN KEY (prodId)
	REFERENCES Product (prodId),
	CONSTRAINT productPurchaseOrder_purchOrderNo_fk FOREIGN KEY (purchOrderNo)
	REFERENCES PurchaseOrder (purchOrderNo)
);
```

SQL Data Insertion Script

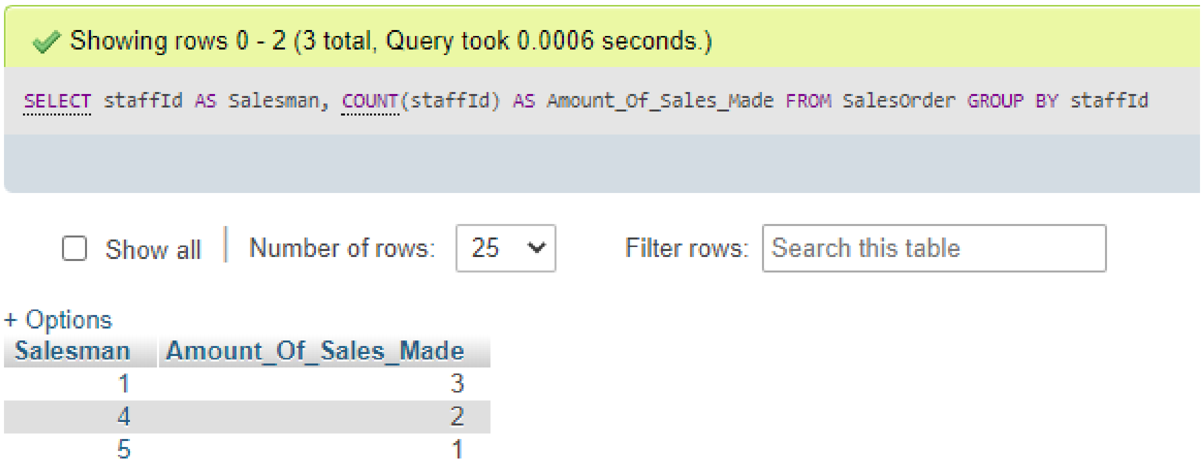
```
/*-----Inserts data into the database-----*/
INSERT INTO Customer (
	custId, custName, custSurname, custType,
	custAddress, custPostCode, custEmail, custPhoneNo)
	VALUES
(000001, "Jerry", "McBerry", "Corporate", "14 Cloth Avenue, Yerobe", "YB28 9TT",
"JerryMcBerry@mail.com", "07846573219"),
(000002, "Steve", "Grafield", "Individual", "9 Fork Street, Kent", "QT4 7PO",
"steveMaster5000@mail.com", "07812545279"),
(000003, "Bazinta", "Manabe", "Individual", "11 Ochre Road, Srpingston", "GH12 15HT",
"Babababaziiiinta@mail.co.uk", NULL),
(000004, "Jack", "Brown", "Corporate", "88 Wharf Side, Steeple", "YZ23 4HE",
"JackinstonBoy@mail.com", "07143473227"),

(000005, "John", "Sheeple", "Corporate", "5 Tree Avenue, Kent", "SE99 8RZ",
"WakeUpSheeple@mail.com", NULL),

(000006, "Angela", "Wolinska", "Individual", "23 Bazinga Street, Place", "SR5 24BB",
"Aaaangie@mail.com", NULL),

(000007, "Ahmed", "Queeple", "Corporate", "16 Jerry Avenue, Andrea", "YT69 4BZ",
"AHQueeps@mail.com", "07844476232"),

(000008, "Alex", "Spowksi", "Individual", "18 Long Street, Place", "BE24 7OP", "AxSpow@mail.com",
"07977754212");


INSERT INTO Staff (

        staffId, staffName, staffSurname, staffJobTitle,

        staffSalary, staffHireDate, staffEmail, staffPhoneNo,

        staffAnnualSalesTarget, staffTotalSales, staffCommisionMultiplier)

        VALUES

(000001, "Jericho", "Melander", "Salesman", 60000.00, "2017-07-24", "j.melander@mail.com",
"07166488233", 120000.00, 7000.00, 0.10),

(000002, "Anna", "Burger", "Assembler", 33000.00, "2018-11-12", "a.burger@mail.com",
"07854663288", NULL, NULL, NULL),

(000003, "Yennefer", "Jackson", "Assembler", 48000.00, "2018-12-19", "y.jackson@mail.com",
"07946123498", NULL, NULL, NULL),

(000004, "Agatha", "Spicy", "Salesman", 51000.00, "2018-09-18", "a.spicy@mail.com",
"07866351722", 140000.00, 44000.00, 0.10),

(000005, "Severus", "Servitor", "Salesman", 25000.00, "2019-07-22", "s.servitor@mail.com",
"07835462718", 60000.00, 79000.00, 0.20),

(000006, "Anika", "Yander", "Accountant", 57000.00, "2017-04-03", "a.yander@mail.com",
"07998364772", NULL, NULL, NULL);


INSERT INTO BoxSet (

        setId, setName, setDesc, setPrice)

        VALUES

(020114, "Fragrant Smelling Nice Box", "This set smells nicer than any other, because it has
fragrance!", 49.99),

(020115, "GetCLEAN Wash Set", "When you gotta really scrub to get yourself fresh!", 129.99),
(020116, "SoftCushions Moisturiser Set", "Stay hydrated with your moisture application station!", 34.99),

(020117, "EagleEye Pretty Spy Collection", "This set makes your eye's pretty; allows you to see through walls too.", 692.99),

(020118, "Lip Gloss Collection-Toss Box", "Makes your lips puffy and glossy, like... glass!", 79.99),

(020119, "RedVengence Foundation Set", "Make-up yourself with the blood of your enemies with this spectacular foundation!", 49.99);


INSERT INTO Supplier (

        suppId, suppName, suppAddress, suppPostCode,

        suppRep, suppEmail, suppPhoneNo)

        VALUES

(000334, "JacksonSupplies", "Box 3 Bazinga Road, London", "DE31 2QB", "Steve Naan", "supplies@jackson.com", "07934565456"),

(000335, "MakeUpStation", "36 Dumpster Street, Birmingham", "BS17 8NT", "Roxanne Ezay", "corporate@makeupstation.com", "07736645289"),

(000336, "JustMOISTURISER", "21 Jacksonville, Kent", "CY21 44UP", "Jacquline Arbor", "services@moisture.com", "07884663251"),

(000337, "SummerCollectionsGMBH", "9 Steinhart, Berlin", "ZQ8992T", "Gunther Weiss", "sales@sc.de", "04834111436");


INSERT INTO Product (

        prodId, prodDesc, minStockLevel, quantityInStock,

        suppId)

        VALUES

(043431, "Moisturiser No.293", 800, 3600, 000336),

(043432, "Lip Balm No.088", 1200, 14000, 000335),

(043433, "Hair Gel No.014", 240, 17, 000334),

(043434, "Styling Foam No.921", 12, 37, 000334),

(043435, "Make-up Brush No.004", 60, 1300, 000335),

(043436, "Extra Moisturiser No.292", 200, 4000, 000336),

(043437, "Foundation No.113", 75, 13, 000337),

(043438, "Hair Comb No.420", 20, 3, 000337);
```

```
INSERT INTO SalesOrder (
        salesOrderNo, salesOrderDate, salesOrderTotal, salesDeliveryDate,
        custId, staffId)
        VALUES
(0012456781, "2020-04-10", 159.98, NULL, 000006, 000004),
(0012456782, "2020-03-11", 692.99, NULL, 000002, 000004),
(0012456783, "2020-04-12", 49.99, "2020-06-12", 000003, 000001),
(0012456784, "2020-03-22", 79.99, NULL, 000003, 000005),
(0012456785, "2020-02-17", 99.98, "2020-04-21", 000001, 000001),
(0012456786, "2020-02-07", 69.98, NULL, 000007, 000001);

INSERT INTO PurchaseOrder (
        purchOrderNo, purchOrderDate, purchOrderTotal, purchDeliveryDate,
        suppId)
        VALUES
(0000453622, "2020-01-12", 500.00, NULL, 000334),
(0000453623, "2020-02-13", 2500.00, NULL, 000337);

INSERT INTO BoxSet_SalesOrder (
        setId, salesOrderNo, setQuantity)
        VALUES
(020118, 0012456781, 2),
(020117, 0012456782, 1),
(020114, 0012456783, 1),
(020118, 0012456784, 1),
(020114, 0012456785, 1),
(020119, 0012456785, 1),
(020116, 0012456786, 2);

INSERT INTO BoxSet_Product (
        setId, prodId, setProdQuantity)
        VALUES
(020114, 043431, 1),
(020114, 043432, 2),
(020115, 043431, 1),
(020115, 043433, 1),
(020115, 043434, 1),
(020116, 043433, 1),
(020116, 043432, 2),
(020117, 043431, 1),
(020117, 043436, 1),
(020118, 043436, 1),
(020118, 043437, 2),
(020119, 043434, 1),
(020119, 043435, 2),
(020119, 043432, 1);

INSERT INTO Product_PurchaseOrder (
        prodId, purchOrderNo, purchOrderQuantity)
        VALUES
(043433, 0000453622, 400),
(043437, 0000453623, 3000),
(043438, 0000453623, 2000);
```

SQL Queries Script

```
/*Below are 4 useful queries*/

/*Finding out how many recent sales each salesman has initiated

-NOTE: didn't have permission to use COUNT with INNER JOIN, so I was not able to display salesman name instead of id...*/

SELECT staffId AS Salesman, COUNT(staffId) AS Amount_Of_Sales_Made

FROM SalesOrder
```

GROUP BY staffId;



✓ Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)

```
SELECT staffId AS Salesman, COUNT(staffId) AS Amount_Of_Sales_Made FROM SalesOrder GROUP BY staffId
```

Salesman	Amount_Of_Sales_Made
1	3
4	2
5	1

Figure 4 - First query to find amount of sales made by each salesman.

/*Finding the highest spending customers

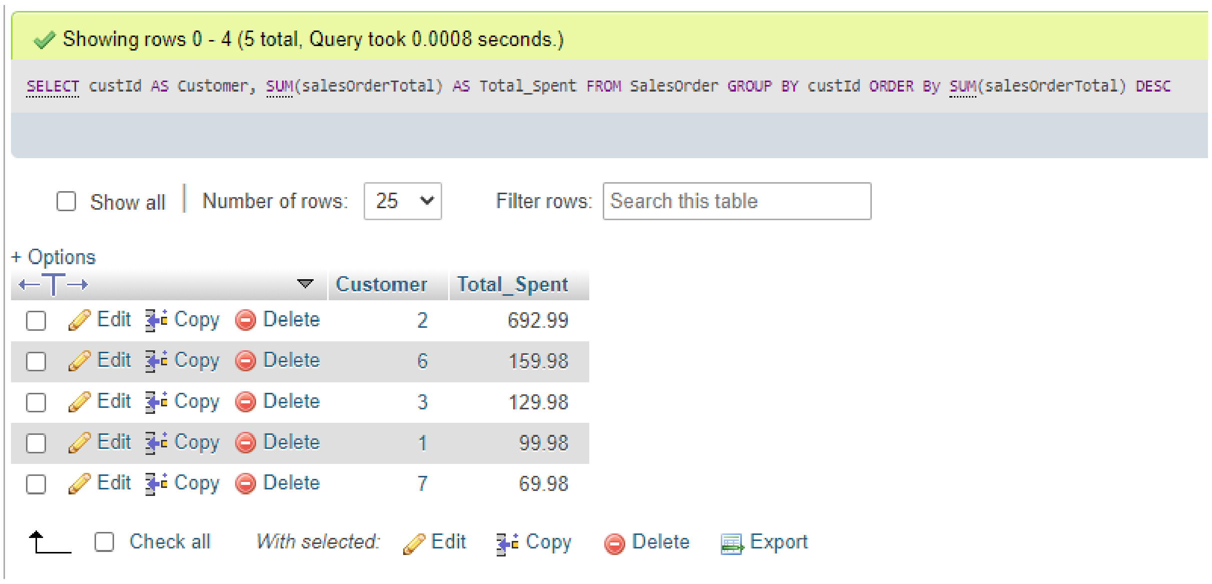
-NOTE: Same issue as above, tried to insert customer names but I do not have permission for phpMyAdmin...*/

SELECT custId AS Customer, SUM(salesOrderTotal) AS Total_Spent

FROM SalesOrder

GROUP BY custId

ORDER By SUM(salesOrderTotal) DESC;



✓ Showing rows 0 - 4 (5 total, Query took 0.0008 seconds.)

```
SELECT custId AS Customer, SUM(salesOrderTotal) AS Total_Spent FROM SalesOrder GROUP BY custId ORDER By SUM(salesOrderTotal) DESC
```

Customer	Total_Spent
2	692.99
6	159.98
3	129.98
1	99.98
7	69.98

Figure 5 - Second query to find high spending customers.

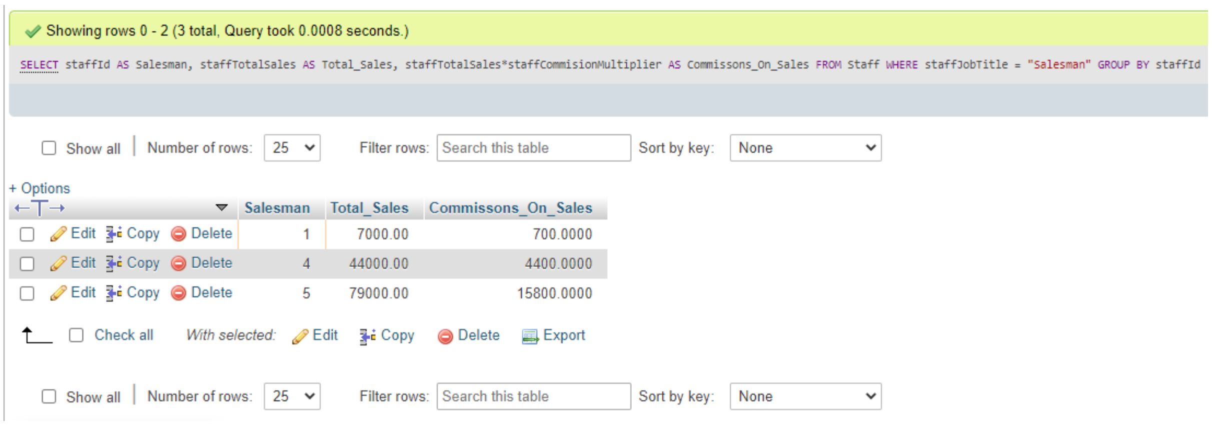
/*Finding how much commission each salesman made on total sales*/

SELECT staffId AS Salesman, staffTotalSales AS Total_Sales, staffTotalSales*staffCommisionMultiplier AS Commissons_On_Sales

FROM Staff

WHERE staffJobTitle = "Salesman"

GROUP BY staffId;



✓ Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

```
SELECT staffId AS Salesman, staffTotalSales AS Total_Sales, staffTotalSales*staffCommisionMultiplier AS Commissons_On_Sales FROM Staff WHERE staffJobTitle = "Salesman" GROUP BY staffId
```

Salesman	Total_Sales	Commissons_On_Sales
1	7000.00	700.0000
4	44000.00	4400.0000
5	79000.00	15800.0000

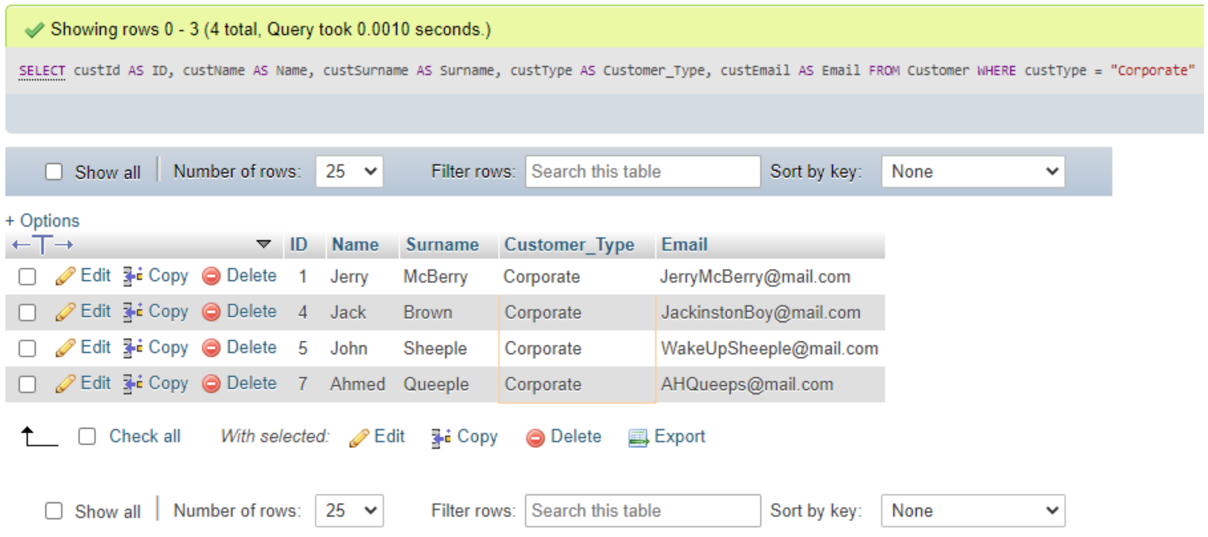
Figure 6 - Finding the commission earned by each employee for their total sales.

/*Retrieving the emails of corporate customers*/

SELECT custId AS ID, custName AS Name, custSurname AS Surname, custType AS Customer_Type, custEmail AS Email

FROM Customer

WHERE custType = "Corporate";



✓ Showing rows 0 - 3 (4 total, Query took 0.0010 seconds.)

```
SELECT custId AS ID, custName AS Name, custSurname AS Surname, custType AS Customer_Type, custEmail AS Email FROM Customer WHERE custType = "Corporate"
```

ID	Name	Surname	Customer_Type	Email
1	Jerry	McBerry	Corporate	JerryMcBerry@mail.com
4	Jack	Brown	Corporate	JackinstonBoy@mail.com
5	John	Sheeple	Corporate	WakeUpSheeple@mail.com
7	Ahmed	Queeple	Corporate	AHQueeps@mail.com

Figure 7 - Retrieving emails of corporate customers.

SQL Query Rationale

Query Number	**Purpose & Rationale**
1 – Total individual sales made per salesman	This query counts the amount of individual orders that a salesman has been linked with to identify how many individual customers/orders they are bringing in for the business.
2 – Highest spending customers	This query identifies how much each customer has spent from their recently placed orders. This could be used to identify customers for potential promotions.
3 – Commissions made by the salesmen	This query uses a salesman total sales and their commissions multiplier to display how much money they have earned through commissions on their total sales. This information could be used, for example, for accounting & tax purposes.
4 – Retrieving emails of corporate customers	This query selects the emails and names of all corporate-type customers. This could be useful for promotional newsletter deals and tax accounting purposes.

Once again, I have tried to join a few tables for all of these queries; however, I have found an issue with permissions on phpMyAdmin where I was not able to write Entity.COUNT(attribute). This is necessary to use the INNER JOIN, which I was not able to test since the system did not allow me to use commands this way.

If I were to improve these queries, I would have used said INNER JOIN to link the queried tables with their corresponding, linked tables to switch the ids for names. This would also allow me to create more intricate and useful queries.