Ecommerce Site – Data Management System

Company: Ecommerce Site
Client: owner

Initial consultation from the client:

We are **selling** out quickly of some of our **inventory**, while other items have gotten close to a period where we might no longer be able to sell them. We need to know which SKU's move faster than others and the point of **reorder**.
We are also struggling with not having enough information about our **clients**. We need their **addresses**, their **social media**, their **age** even? to understand if some of our products have a better fit depending on certain characteristics, but also to understand how we can better communicate our promotions.

Speaking of **promotions**, I would also like to know how they are working out. Are people actually purchasing more because of them? Are they being used even? If they have, which **codes** worked better? The ones we sent via email? text? or social media?

I think that's a good start, please get back to us as soon as possible.

Email later that week—

Something I would like to add to this first stage working with you is specifically about our onsite **support button**… is it working? do people use it?

Thank you!

eCommerce Site owner

- PART 1 – FLOW OF THE SYSTEM -

Dear,

After our initial consultation and follow up and focusing on understanding what are the aspects that concern you and what kind of reports you would like to see drawn out of your system, our team was able to build a comprehensive first step to your database structure.
Composed of 7 tables, the flow of your database would look as follows:

I. Products

The entire system revolves around your products. Without these, none of the other tables could exist. It is important to keep the information in this Master Table concise. The Product ID is the unique identifier for each product, given that the names are so similar in between them.

II. Customers

You mentioned your need to capture a bigger scope of data related to your customers. We understood this need and based on it and on the logic of your business, your customer table accompanies products as the center of the database. All following tables are directly or indirectly related to their information, therefore the amount of data collected from them will be crucial.
With this amount of information, the possibility of focusing your business on a customer centric approach is higher, which takes us to our following table

III. Marketing

The flow of the system brings us to the Marketing table. The way you promote your product to the customers or potential customers will be directly related to the information you gather from them. The relation between this tables could potentially allow you to understand the average age of your customers, their gender, when they buy, why, the frequency of their purchases. Based on this information, we created a table called Promotions.

a. Promotions

This table will allow the creation of different promotions and the tracking of each of them. We included the channel of distribution of promotion, so you can actively track which channel is bringing the most customers and potentially strengthen your efforts to that specific channel.

IV. Orders

This category is divided into 2 tables, orders and order details. The idea behind this division is that you will be able to have conciseness when only using the Orders table.

Customer ID and Promotion ID show as the relation the Orders table has to these ID's. There would be no Order without a Customer to create it and the Promotion ID would just come through if the Customer used a promotion with their order.

a. Order Details

Order Details comes in with 2 columns of its own, Order Quantity referring to the number of SKU's drawn out of your inventory and Order Amount which is the total payment from the customer for their specific order. Accompanying these columns, the table is fed with the Order Number from the Orders table and Product ID from the Products table.

V. Employees

As we discussed, you have few employees at the moment but having them in a system is important, and more so if you will keep growing at the pace you have been in the last few years. To this category, there are 2 tables to it, the first one is Employees, with all their identifying details and the Department they belong to.

The second table is Support and this is related to the last email you sent to us after our initial conversation.

a. Support

The support table is created based on the need of tracking the success of the support button on your website. Having this table will allow you to create reports such as the number of issued you get in a day/week/month, how fast you are responding to those issues, what agent is being the most successful, and also what type of customer is using this button and for what purpose.

This table has 4 columns of its own and 2 others that are fed from the Employees table and the Customers table. The Email, which is fed from the Customers table, is a column that will potentially let you know the type of customer that is using the support button.

In the appendices, you will have an example of the output dashboards of the system.

- PART 2 – DATABASE STRUCTURE -

I. **Customers** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Customer_ID	VARCHAR(10)	NO	PRIMARY	Unique identifier of the customer
First_Name	VARCHAR(14)	NO		Customer first name
Last_Name	VARCHAR(16)	NO		Customer Last name
Birth_Date	DATE	NO		Customer Birth Date
Gender	ENUM(M,F)	NO		Customer gender
Address	VARCHAR(255)	NO		Customer Address
Zip_code	INT(5)	NO		Customer Zip code
City	VARCHAR(15)	NO		Customer City
Country	VARCHAR(15)	NO		Customer Country
Email	VARCHAR(60)	NO		Customer email
Phone_number	INT	NO		Customer phone number
Customer_Type	ENUM(OT,S)	YES		Is the customer a one-time purchaser (OT) or a subscriber to our program (S)?

II. **Products** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Product_ID	VARCHAR(10)	NO	PRIMARY	Unique identifier of the product
Product_Family	VARCHAR(14)	NO		Product family (either Full-size bars, Minis bars, Gift card)
Product_Name	VARCHAR(16)	NO		Product name (representing the flavour of the bars for Full-size and Minis)
Product_Price	DECIMAL(4,2)	NO		Individual product price

III. **Promotions** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Promotion_ID	VARCHAR(22)	NO	PRIMARY	Unique code used in the promotion
Promotion_Type	VARCHAR(14)	NO		Promotion channel (email, Instagram, Facebook, Twitter, Tik-Tok, SMS)
Promotion_Title	VARCHAR(16)	NO		Name of the promotion and which products it is targeting

IV. **Orders** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Order_No	INT	NO	PRIMARY	Unique number of the order
Order_date	DATE	NO		Date of the order
Shipped_order	DATE	YES		Date of the shipment of the order
Order_Amount	DECIMAL(5,2)	NO		Total amount of the order
Customer_ID	VARCHAR(10)	NO	FOREIGN	Unique identifier of the customer
Promotion_ID	INT(10)	NO	FOREIGN	Unique code used in the promotion

V. **Order Details** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Order_No	VARCHAR(10)	NO	PRIMARY	Unique number of the order
Quantity	INT(8)	NO		Number of products in the order
Amount	DECIMAL(5,2)	NO		Amount of the order in USD
Product_ID	VARCHAR(10)	NO	FOREIGN	Unique identifier of the product

VI. **Employees** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Employee_No	INT(10)	NO	PRIMARY	Unique identifier of the employee
Employee_First_Name	VARCHAR(14)	NO		Employee first name
Employee_Last_Name	VARCHAR(16)	NO		Employee last name
Birth_date	DATE	NO		Employee birth date
Hire_date	DATE	NO		Employee hire date
Employee_department	VARCHAR(40)	NO		Department where the employee currently works

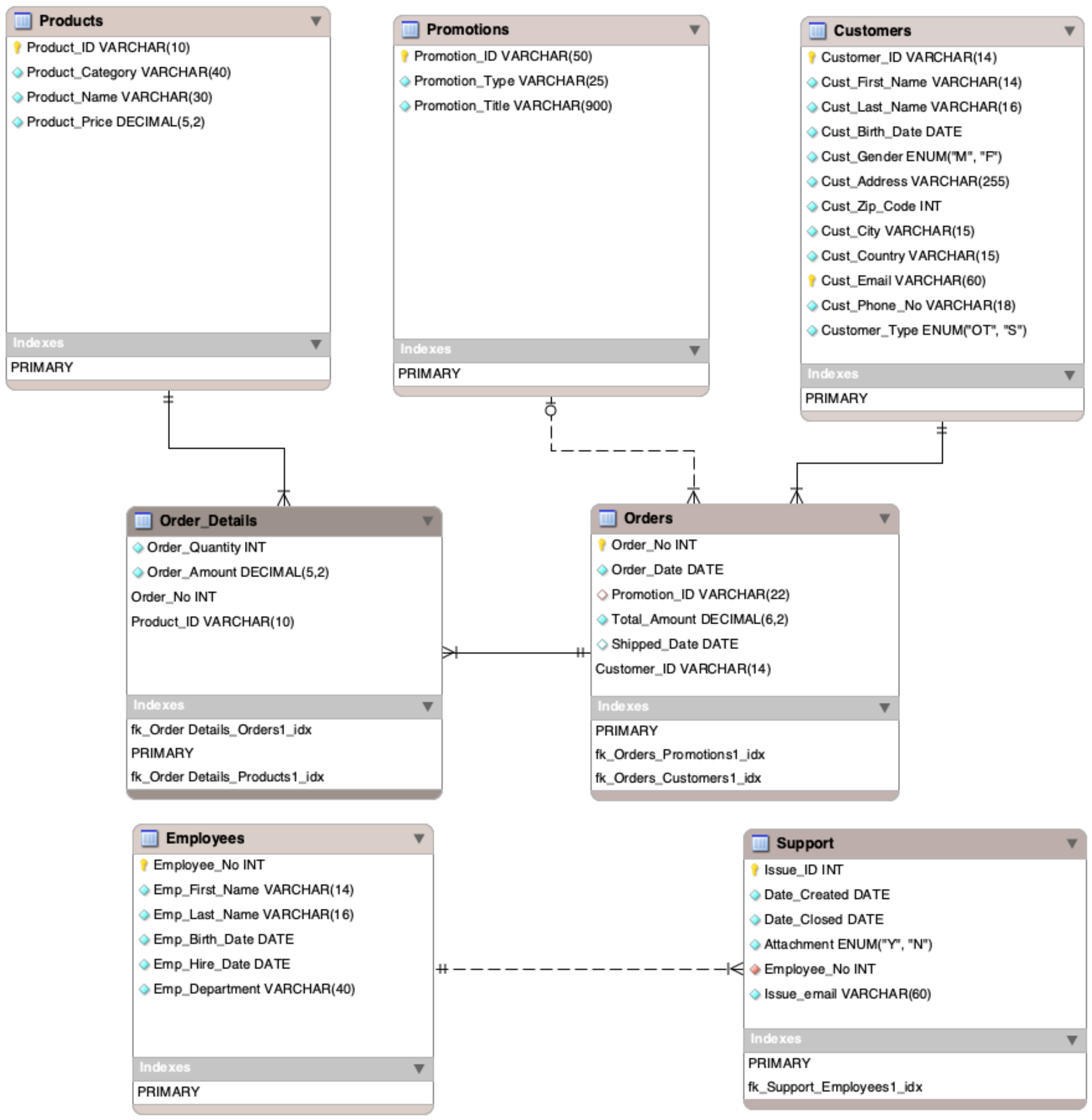
VII. **Support** Table

FIELD	DATATYPE	NULL	KEY	DESCRIPTION
Issue_ID	INT(10)	NO	PRIMARY	Unique identifier of the reported issue
Date_Created	DATE	NO		Date of creation of the issue
Date_Closed	DATE	NO		Date of closure of the issue
Attachment	ENUM(Y,N)	NO		Were there files attached to the reported issue? (Y/N)
Customer_email	VARCHAR(60)	YES	FOREIGN	Does the email mentioned in the issue report match one of customers?
Emp_No	INT(10)	NO	FOREIGN	Unique identifier of the employee who answered the issue

This is a database structure that can always be updated

- PART 3 – ENTITY-RELATIONAL MODEL -

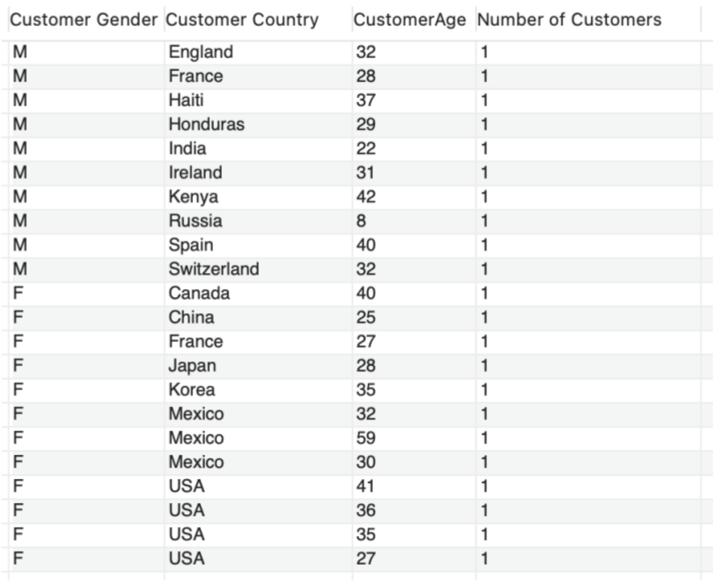
Please find the Model script in the appendices.



- PART 4 – SQL QUERIES TO BUILD THE REPORT -

What are your customers' demographics? (age, gender, country)

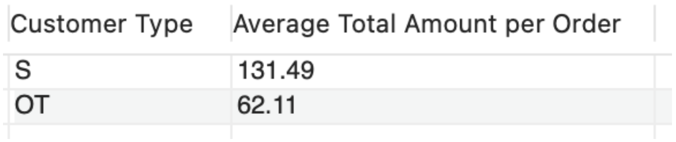
```
SELECT cust_gender AS "Customer Gender", cust_country AS "Customer Country",
        (YEAR(CURDATE()) - YEAR(cust_birth_date)) AS "CustomerAge",
        COUNT(customer_id) AS "Number of Customers"
FROM customers
GROUP BY cust_country, cust_gender, customerage
ORDER BY cust_gender, cust_country ASC;
```



Customer Gender	Customer Country	CustomerAge	Number of Customers
M	England	32	1
M	France	28	1
M	Haiti	37	1
M	Honduras	29	1
M	India	22	1
M	Ireland	31	1
M	Kenya	42	1
M	Russia	8	1
M	Spain	40	1
M	Switzerland	32	1
F	Canada	40	1
F	China	25	1
F	France	27	1
F	Japan	28	1
F	Korea	35	1
F	Mexico	32	1
F	Mexico	59	1
F	Mexico	30	1
F	USA	41	1
F	USA	36	1
F	USA	35	1
F	USA	27	1

What is the average purchase amount for one-time purchasers vs. subscribers?

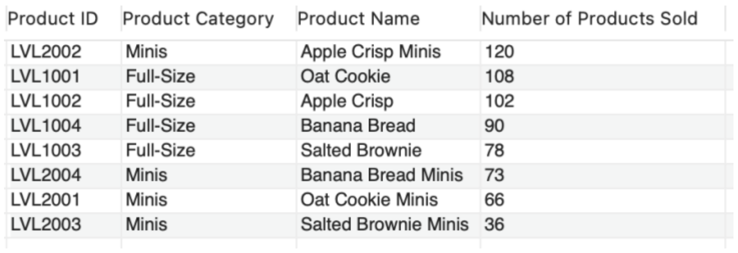
```
SELECT c.customer_type AS “Customer Type”, ROUND(AVG(total_amount),2) AS “Average
        Total Amount per Order”
FROM orders AS o, customers AS c
WHERE o.customer_id = c.customer_id
GROUP BY c.customer_type;
```



Customer Type	Average Total Amount per Order
S	131.49
OT	62.11

Which type of bars are sold the most?

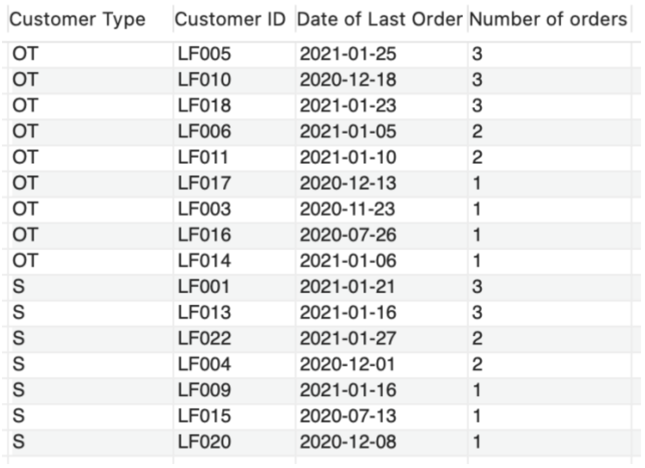
```
SELECT od.product_id AS "Product ID", p.product_category AS "Product Category",
        p.product_name AS "Product Name", SUM(od.order_quantity) AS "Number of
        Products Sold"
FROM order_details AS od, products AS p
WHERE od.product_id = p.product_id
AND UPPER(p.product_category) NOT LIKE '%GIFT%'
GROUP BY od.product_id
ORDER BY SUM(od.order_quantity) DESC;
```



Product ID	Product Category	Product Name	Number of Products Sold
LVL2002	Minis	Apple Crisp Minis	120
LVL1001	Full-Size	Oat Cookie	108
LVL1002	Full-Size	Apple Crisp	102
LVL1004	Full-Size	Banana Bread	90
LVL1003	Full-Size	Salted Brownie	78
LVL2004	Minis	Banana Bread Minis	73
LVL2001	Minis	Oat Cookie Minis	66
LVL2003	Minis	Salted Brownie Minis	36

Do our customers order several times from us?

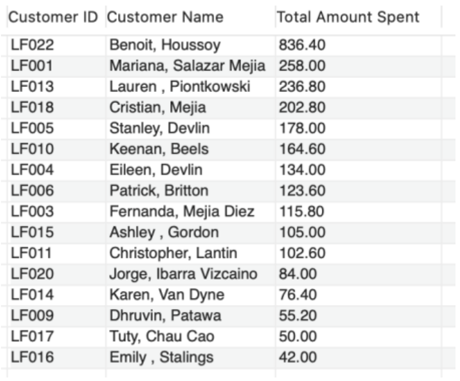
```
SELECT c.customer_type AS "Customer Type", c.customer_id AS "Customer ID",
        MAX(order_date) AS "Date of Last Order", COUNT(order_no) AS "Number of orders"
FROM customers AS c, orders AS o
WHERE c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY c.customer_type, COUNT(order_no) DESC;
```



Customer Type	Customer ID	Date of Last Order	Number of orders
OT	LF005	2021-01-25	3
OT	LF010	2020-12-18	3
OT	LF018	2021-01-23	3
OT	LF006	2021-01-05	2
OT	LF011	2021-01-10	2
OT	LF017	2020-12-13	1
OT	LF003	2020-11-23	1
OT	LF016	2020-07-26	1
OT	LF014	2021-01-06	1
S	LF001	2021-01-21	3
S	LF013	2021-01-16	3
S	LF022	2021-01-27	2
S	LF004	2020-12-01	2
S	LF009	2021-01-16	1
S	LF015	2020-07-13	1
S	LF020	2020-12-08	1

What is the total amount spent per customer?

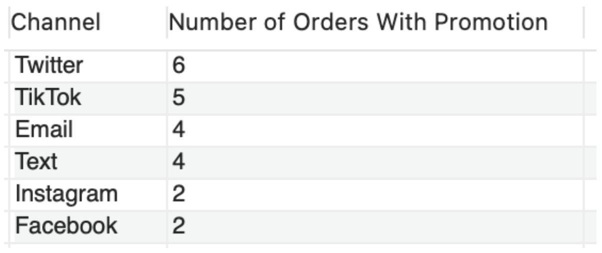
```
SELECT c.customer_id AS "Customer ID", concat_ws(", ", c.cust_first_name,
        c.cust_last_name) AS "Customer Name", SUM(o.total_amount) AS "Total Amount
        Spent"
FROM customers AS c, orders AS o
WHERE c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY SUM(o.total_amount) DESC;
```



Customer ID	Customer Name	Total Amount Spent
LF022	Benoit, Houssoy	836.40
LF001	Mariana, Salazar Mejia	258.00
LF013	Lauren , Piontkowski	236.80
LF018	Cristian, Mejia	202.80
LF005	Stanley, Devlin	178.00
LF010	Keenan, Beels	164.60
LF004	Eileen, Devlin	134.00
LF006	Patrick, Britton	123.60
LF003	Fernanda, Mejia Diez	115.80
LF015	Ashley , Gordon	105.00
LF011	Christopher, Lantin	102.60
LF020	Jorge, Ibarra Vizcaino	84.00
LF014	Karen, Van Dyne	76.40
LF009	Dhruvin, Patawa	55.20
LF017	Tuty, Chau Cao	50.00
LF016	Emily , Stalings	42.00

Which channels work best for our promotions and create most orders?

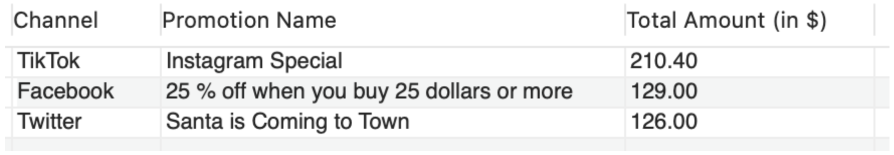
```
SELECT pm.promotion_type AS "Channel", COUNT(o.order_no) as "Number of Orders with
        Promotion"
FROM promotions AS pm, orders AS o
WHERE pm.promotion_id = o.promotion_id
AND o.promotion_id IS NOT NULL
GROUP BY pm.promotion_type
ORDER BY COUNT(o.order_no) DESC;
```



Channel	Number of Orders With Promotion
Twitter	6
TikTok	5
Email	4
Text	4
Instagram	2
Facebook	2

What is the top 3 of the most revenue generating promotions?

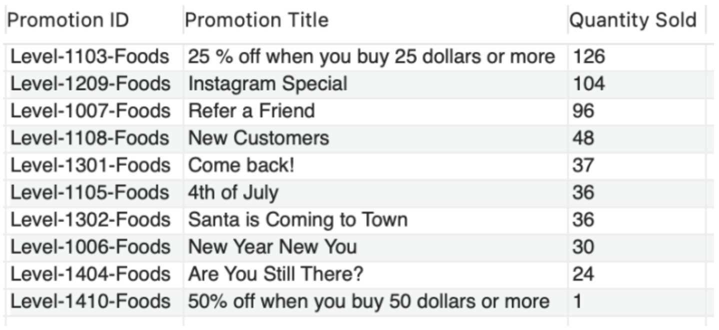
```
SELECT pm.promotion_type AS "Channel", pm.promotion_title AS "Promotion Name",
        SUM(o.total_amount) AS "Total Amount (in $)"
FROM promotions AS pm, orders AS o
WHERE pm.promotion_id = o.promotion_id
GROUP BY o.promotion_id
ORDER BY SUM(o.total_amount) DESC
LIMIT 3;
```



Channel	Promotion Name	Total Amount (in $)
TikTok	Instagram Special	210.40
Facebook	25 % off when you buy 25 dollars or more	129.00
Twitter	Santa is Coming to Town	126.00

Do customers buy significantly more with a promotion code? (quantity)

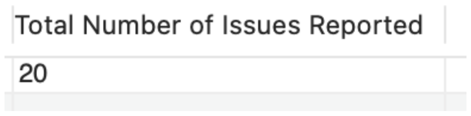
```
SELECT pm.promotion_id AS "Promotion ID", pm.promotion_title AS "Promotion Title",
        SUM(od.order_quantity) AS "Quantity Sold"
FROM promotions AS pm, orders AS o, order_details AS od
WHERE pm.promotion_id = o.promotion_id
AND o.order_no = od.order_no
GROUP BY pm.promotion_title
ORDER BY SUM(od.order_quantity) DESC;
```



Promotion ID	Promotion Title	Quantity Sold
Level-1103-Foods	25 % off when you buy 25 dollars or more	126
Level-1209-Foods	Instagram Special	104
Level-1007-Foods	Refer a Friend	96
Level-1108-Foods	New Customers	48
Level-1301-Foods	Come back!	37
Level-1105-Foods	4th of July	36
Level-1302-Foods	Santa is Coming to Town	36
Level-1006-Foods	New Year New You	30
Level-1404-Foods	Are You Still There?	24
Level-1410-Foods	50% off when you buy 50 dollars or more	1

How many issues are reported through the support button?

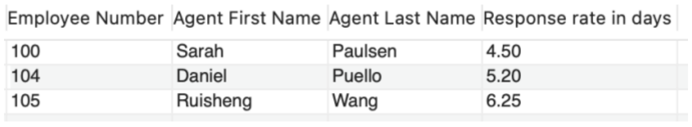
```
SELECT COUNT(issue_id) AS "Total Number of Issues Reported"
FROM support;
```



Total Number of Issues Reported
20

What is the average response rate? How fast are the issues being responded to?

```
SELECT e.employee_no AS "Employee Number", e.emp_first_name AS "Agent First Name",
          e.emp_last_name AS "Agent Last Name", ROUND(AVG(s.date_closed-
          s.date_created),2) AS "Response rate in days"
FROM support AS s, employees AS e
WHERE s.employee_no = e.employee_no
AND UPPER(e.emp_department) = "SUPPORT"
GROUP BY e.employee_no;
```



Employee Number	Agent First Name	Agent Last Name	Response rate in days
100	Sarah	Paulsen	4.50
104	Daniel	Puello	5.20
105	Ruisheng	Wang	6.25

- PART 5 – SQL PROCEDURE TO BUILD THE REPORT -

I. More details on the Support Button

The first built procedure aims to know if when an issue is reported through the support button it is done by one of the current customers or when it is done by a potential customer. Given the current structure of the button, the only possibility to get this information is through the email mentioned when the user filled in the query.
We have added a column mentioning if Yes or No the issue was reported by a current customer.

```
ALTER TABLE support
ADD Customer_yn  VARCHAR(3) NULL;
```

This procedure has to be used once the issue is reported in the system by inputting the issue ID. If the email mentioned in the support table is the same as the one from the customers table, then the "Customer_yn" column is updated as "Yes". If not, then the value is updated to "No".

Input: issue ID as INT

Functionality

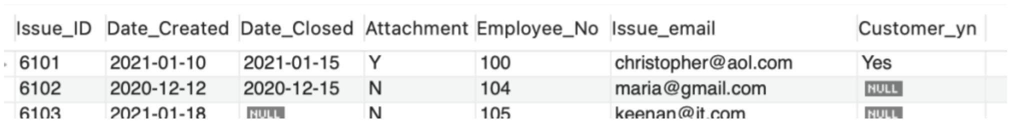
> ***Step 1*** – Declare a variable to store the following query. Declared as an integer: v_count
> ***Step 2*** – Set the variable as equal to 0
> ***Step 3*** – Query where the issue_email matches the cust_email from the customers table for the issue number selected as an input. If there is a match, the variable is equal to 1, if not it stays as 0.
> ***Step 4*** – If statement as:
> If v_count is greater than 0, then the "Customer_yn" column from support table is updated as "Yes" for the input issue ID. Else, the column is updated to "No" for the input issue ID

Please find below the code of the procedure we called 'support_cust_yn_prc'

```
CREATE PROCEDURE `support_cust_yn_prc`(IN in_issue_no INT)

BEGIN
    DECLARE v_count INT;
    SET v_count = 0;

    SELECT count(1)
    INTO v_count
    FROM support AS s
    LEFT JOIN customers AS c
    ON s.issue_email = c.cust_email
    WHERE issue_id = in_issue_no
    AND s.issue_email = c.cust_email;

    IF v_count > 0
         THEN UPDATE support
                   SET customer_yn = "Yes"
       WHERE issue_id = in_issue_no;
    ELSE
         UPDATE support
         SET customer_yn = "No"
         WHERE issue_id = in_issue_no;
    END IF;

END
```

Here is the output for the issue ID 6101:

```
CALL support_cust_yn_prc(6101);

SELECT *
FROM support;
```



Issue_ID	Date_Created	Date_Closed	Attachment	Employee_No	Issue_email	Customer_yn
6101	2021-01-10	2021-01-15	Y	100	christopher@aol.com	Yes
6102	2020-12-12	2020-12-15	N	104	maria@gmail.com	NULL
6103	2021-01-18	NULL	N	105	keenan@it.com	NULL

However, we believe there is a possibility that a current customer uses a different email from the one registered in the Customers table. Therefore, to increase the chances of having the correct information, we recommend you add a new feature to the support button, such as:

"Have you already ordered from us?" **Yes** **No**
"If yes, please use the email you used for your order"

II. Is the promise “3-day shipment” made?

Customer satisfaction is the most important as customer is at the heart of the company. Therefore, it is essential to know if the 3-day shipment promise is met when a customer order from us.
This procedure checks if the day difference between the shipped date and order date for an order. If the days difference is less or equal to 3 days, then a value should return “3 days or less”. If not, then a value should return “More than 3 days!”.
We know that some orders have not been shipped yet. Therefore, for these orders, the days difference is calculated based on the current date.

Input: order number as INT

Functionality

> ***Step 1*** – Declare a variable to store the following query. Declared as VARCHAR: v_check
> ***Step 2*** – Query selecting the days difference between shipped date and order date into v_check and where the order number is equal to the input
> ***Step 3*** – If statement as:
> If v_check (days difference) is less or equal to 3, then the output is set as “3 days or less”.
> If v_check is greater than 3, then the output is set as “More than 3 days!”

Output: “3 days or less” or “More than 3 days!” VARCHAR(30)

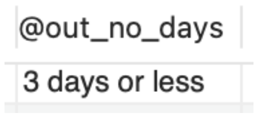
Please find below the code of the procedure we called ‘days_ship_yn’

```
CREATE PROCEDURE `days_ship_yn`(IN in_order_no INT, OUT out_3day_ship_yn
VARCHAR(30))
BEGIN

   DECLARE v_check VARCHAR(3);

   SELECT CASE
     WHEN shipped_date is not null THEN DATEDIFF(shipped_date, order_date)
     WHEN shipped_date is null THEN DATEDIFF(curdate(), order_date)
     END
   INTO v_check
   FROM orders
   WHERE order_no = in_order_no;

   IF v_check <= 3
        THEN SET out_3day_ship_yn = "3 days or less";
   ELSE
        SET out_3day_ship_yn = "More than 3 days!";
```

```
    END IF;

END
```

Here is the output for the order number 3

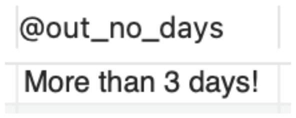
```
CALL days_ship_yn(3, @out_no_days);
SELECT @out_no_days;
```



```
@out_no_days
3 days or less
```

Here is the output for the order number 12, which has not been shipped yet

```
CALL days_ship_yn(12, @out_no_days);
SELECT @out_no_days;
```

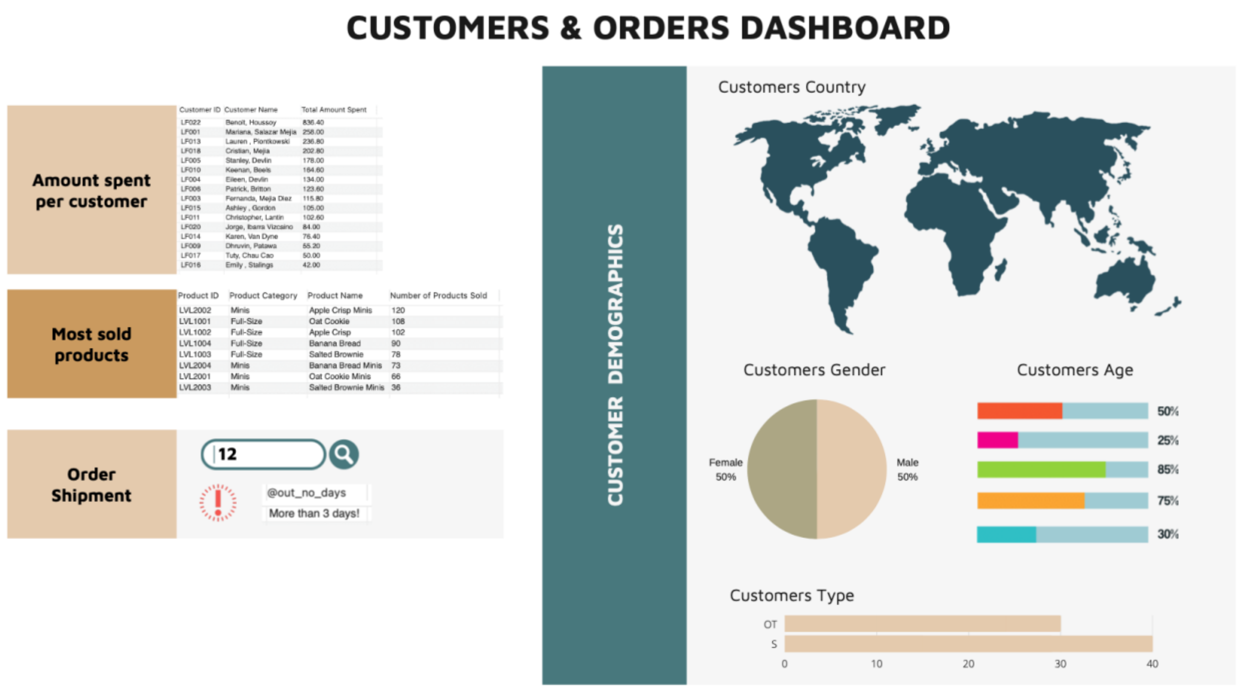


```
@out_no_days
More than 3 days!
```

Then, for orders that have not been shipped yet, and where it has been more than 3 days, the information will show up on the dashboard as a warning. Therefore, it will be possible for you to check why the shipment has been delayed, but also to contact the customer to let him/her know.

- APPENDICES -

System Dashboards Examples



Amount spent per customer

Customer ID	Customer Name	Total Amount Spent
LF022	Benoit, Houssoy	836.40
LF001	Mariana, Salazar Mejia	258.00
LF013	Lauren , Piontkowski	236.80
LF018	Cristian, Mejia	202.80
LF005	Stanley, Devlin	178.00
LF010	Keenan, Beels	164.60
LF004	Eileen, Devlin	134.00
LF006	Patrick, Britton	123.60
LF003	Fernanda, Mejia Diez	115.80
LF015	Ashley , Gordon	105.00
LF011	Christopher, Lantin	102.60
LF020	Jorge, Ibarra Vizcaino	84.00
LF014	Karen, Van Dyne	76.40
LF009	Dhruvin, Patawa	55.20
LF017	Tuty, Chau Cao	50.00
LF016	Emily , Stalings	42.00

Most sold products

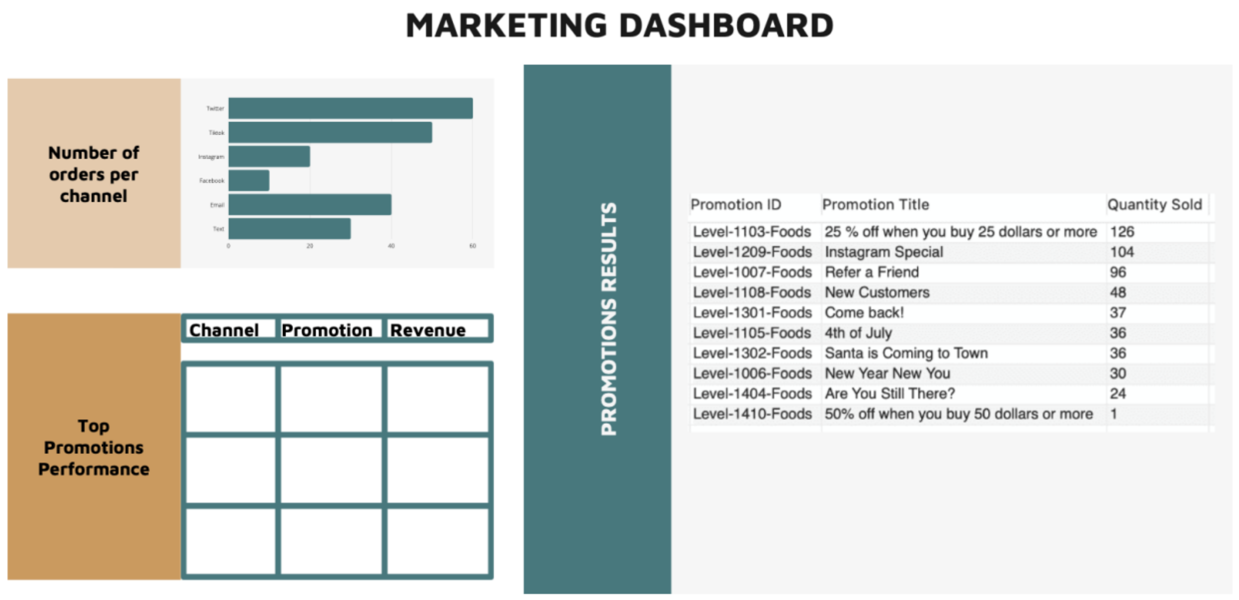
Product ID	Product Category	Product Name	Number of Products Sold
LVL2002	Minis	Apple Crisp Minis	120
LVL1001	Full-Size	Oat Cookie	108
LVL1002	Full-Size	Apple Crisp	102
LVL1004	Full-Size	Banana Bread	90
LVL1003	Full-Size	Salted Brownie	78
LVL2004	Minis	Banana Bread Minis	73
LVL2001	Minis	Oat Cookie Minis	66
LVL2003	Minis	Salted Brownie Minis	36

Order Shipment

12

@out_no_days

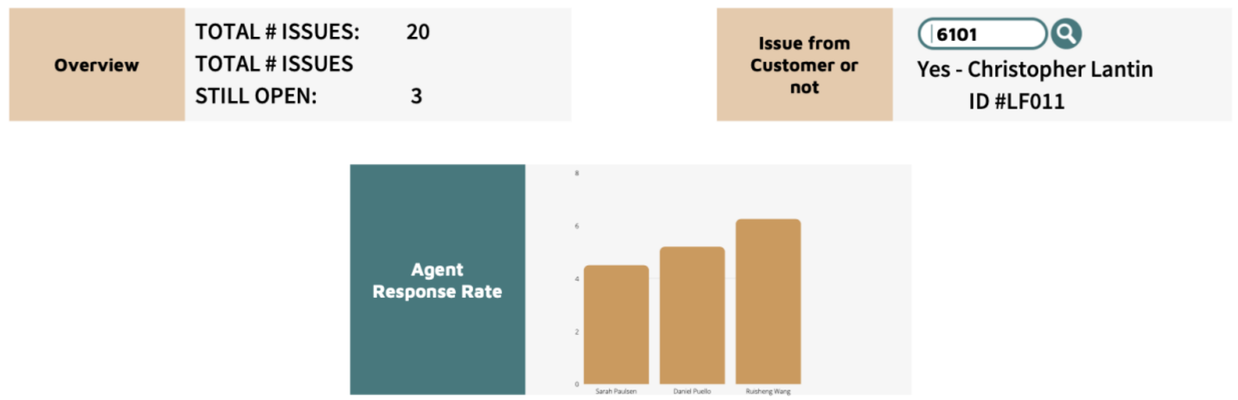
More than 3 days!



Channel	Promotion	Revenue

Promotion ID	Promotion Title	Quantity Sold
Level-1103-Foods	25 % off when you buy 25 dollars or more	126
Level-1209-Foods	Instagram Special	104
Level-1007-Foods	Refer a Friend	96
Level-1108-Foods	New Customers	48
Level-1301-Foods	Come back!	37
Level-1105-Foods	4th of July	36
Level-1302-Foods	Santa is Coming to Town	36
Level-1006-Foods	New Year New You	30
Level-1404-Foods	Are You Still There?	24
Level-1410-Foods	50% off when you buy 50 dollars or more	1

SUPPORT DASHBOARD



Overview	TOTAL # ISSUES:	20
	TOTAL # ISSUES STILL OPEN:	3

Issue from Customer or not	6101
	Yes - Christopher Lantin ID #LF011

Agent Response Rate

ER Model Code – Table scripts

```
-- -----------------------------------------------------
-- Schema final_project
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `final_project` DEFAULT CHARACTER SET utf8 ;
USE `final_project` ;

-- -----------------------------------------------------
-- Table `final_project`.`Products`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Products` (
  `Product_ID` VARCHAR(10) NOT NULL,
  `Product_Category` VARCHAR(40) NOT NULL,
  `Product_Name` VARCHAR(30) NOT NULL,
  `Product_Price` DECIMAL(5,2) NOT NULL,
  PRIMARY KEY (`Product_ID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `final_project`.`Promotions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Promotions` (
  `Promotion_ID` VARCHAR(50) NOT NULL,
  `Promotion_Type` VARCHAR(25) NOT NULL,
  `Promotion_Title` VARCHAR(900) NOT NULL,
  PRIMARY KEY (`Promotion_ID`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `final_project`.`Customers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Customers` (
  `Customer_ID` VARCHAR(14) NOT NULL,
  `Cust_First_Name` VARCHAR(14) NOT NULL,
  `Cust_Last_Name` VARCHAR(16) NOT NULL,
  `Cust_Birth_Date` DATE NOT NULL,
  `Cust_Gender` ENUM("M", "F") NOT NULL,
  `Cust_Address` VARCHAR(255) NOT NULL,
  `Cust_Zip_Code` INT NOT NULL,
  `Cust_City` VARCHAR(15) NOT NULL,
  `Cust_Country` VARCHAR(15) NOT NULL,
  `Cust_Email` VARCHAR(60) NOT NULL,
  `Cust_Phone_No` VARCHAR(18) NOT NULL,
  `Customer_Type` ENUM("OT", "S") NOT NULL,
  PRIMARY KEY (`Customer_ID`, `Cust_Email`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `final_project`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Orders` (
  `Order_No` INT NOT NULL,
  `Order_Date` DATE NOT NULL,
  `Promotion_ID` VARCHAR(22) NULL,
  `Total_Amount` DECIMAL(6,2) NOT NULL,
  `Shipped_Date` DATE NULL,
  `Customer_ID` VARCHAR(14) NOT NULL,
  PRIMARY KEY (`Order_No`, `Customer_ID`),
  INDEX `fk_Orders_Promotions1_idx` (`Promotion_ID` ASC) VISIBLE,
  INDEX `fk_Orders_Customers1_idx` (`Customer_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Orders_Promotions1`
    FOREIGN KEY (`Promotion_ID`)
    REFERENCES `final_project`.`Promotions` (`Promotion_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Orders_Customers1`
    FOREIGN KEY (`Customer_ID`)
    REFERENCES `final_project`.`Customers` (`Customer_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `final_project`.`Employees`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Employees` (
  `Employee_No` INT NOT NULL,
  `Emp_First_Name` VARCHAR(14) NOT NULL,
  `Emp_Last_Name` VARCHAR(16) NOT NULL,
  `Emp_Birth_Date` DATE NOT NULL,
  `Emp_Hire_Date` DATE NOT NULL,
  `Emp_Department` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`Employee_No`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `final_project`.`Order_Details`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Order_Details` (
  `Order_Quantity` INT NOT NULL,
  `Order_Amount` DECIMAL(5,2) NOT NULL,
  `Order_No` INT NOT NULL,
  `Product_ID` VARCHAR(10) NOT NULL,
  INDEX `fk_Order Details_Orders1_idx` (`Order_No` ASC) VISIBLE,
  PRIMARY KEY (`Order_No`, `Product_ID`),
  INDEX `fk_Order Details_Products1_idx` (`Product_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Order Details_Orders1`
    FOREIGN KEY (`Order_No`)
    REFERENCES `final_project`.`Orders` (`Order_No`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order Details_Products1`
    FOREIGN KEY (`Product_ID`)
    REFERENCES `final_project`.`Products` (`Product_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `final_project`.`Support`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `final_project`.`Support` (
  `Issue_ID` INT NOT NULL,
  `Date_Created` DATE NOT NULL,
  `Date_Closed` DATE NOT NULL,
  `Attachment` ENUM("Y", "N") NOT NULL,
  `Employee_No` INT NOT NULL,
  `Issue_email` VARCHAR(60) NOT NULL,
  PRIMARY KEY (`Issue_ID`),
  INDEX `fk_Support_Employees1_idx` (`Employee_No` ASC) VISIBLE,
  CONSTRAINT `fk_Support_Employees1`
    FOREIGN KEY (`Employee_No`)
    REFERENCES `final_project`.`Employees` (`Employee_No`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Insert scripts

Here are some of our Insert scripts for the different tables. The data not mentioned in the scripts below has been added through csv files that we will be more than happy to share with you if needed.

a. Parents Tables

Customer Table

```
INSERT INTO `final_project`.`Customers`
(`Customer_ID`, `Cust_First_Name`, `Cust_Last_Name`, `Cust_Birth_Date`, `Cust_Gender`,
`Cust_Address`, `Cust_Zip_Code`, `Cust_City`, `Cust_Country`, `Cust_Email`,
`Cust_Phone_No`, `Customer_Type`)
VALUES
("LF001", "Mariana", "Salazar Mejia", "1989-06-16", "F", "1 Leighton St", 35622, "Merida",
"Mexico", "mariana@gmail.com", “8567771829”, "S"),
("LF002", "Maria", "Salazar Mejia", "1981-11-19", "F", "56 Wareham St", 83489, "Toronto",
"Canada", "maria@gmail.com", “8579998814”, "S"),
("LF003", "Fernanda", "Mejia Diez", "1962-02-21", "F", "Av. Mexico 129", 37400, "Mexico
City", "Mexico", "fernanda@gmail.com", “6174568813”, "OT"),
("LF004", "Eileen", "Devlin", "1980-10-31", "F", "549 Franklin St", 20141, "Boston", "USA",
"eileen@gmail.com", "7619998823", "S"),
("LF005", "Stanley", "Devlin", "2013-12-31", "M", "1 Crawford Av", 20155, "Moscow",
"Russia", "stanley@gmail.com", " 8742839001", "OT"),
("LF006", "Patrick", "Britton", "1981-10-10", "M", " 45 Newbury St", 20384, "Madrid",
"Spain", "patrick@gmail.com", " 9298887234", "OT"),
("LF007", "James", "Marvel", "1979-05-04", "M", " 78 Kinnaird St", 73849, "Nairobi", "Kenya",
"james@gmail.com", " 7652838892", "OT"),
("LF008", "Maithily", "Erande", "1985-04-16", "F", " 43 Sudbury Av", 88374, "New York",
"USA", "maithily@gmail.com", " 7384491234", "S"),
("LF009", "Dhruvin", "Patawa", "1999-09-04", "M", " 51 Marlboro St", 82092, "Mumbai",
"India", "dhruvin@gmail.com", " 7387772912", "S"),
("LF010", "Keenan", "Beels", "1989-07-27", "M", " 3 Education Circle", 98485, "London",
"England", "keenan@gmail.com", " 4526667189", "OT");
```

Products Table

```
INSERT INTO `final_project`.`Products`
(`Product_ID`, `Product_Category`, `Product_Name`, `Product_Price`)
VALUES
("LVL1001", "Full-Size", "Oat Cookie", 3.50),
("LVL1002", "Full-Size", "Apple Crisp", 3.50),
("LVL1003", "Full-Size", "Salted Brownie", 3.50),
("LVL1004", "Full-Size", "Banana Bread", 3.50),
("LVL2001", "Minis ", "Oat Cookie Minis", 2.20),
("LVL2002", "Minis ", "Apple Crisp Minis", 2.20),
("LVL2003", "Minis ", "Salted Brownie Minis", 2.20),
("LVL2004", "Minis ", "Banana Bread Minis", 2.20),
("LVL0020", "Gift Card ", "Level Digital Gift Card $20", 20.00),
("LVL0050", "Gift Card ", "Level Digital Gift Card $50", 50.00),
("LVL0100", "Gift Card ", "Level Digital Gift Card $100", 100.00);
```

Promotions Table

```
INSERT INTO `final_project`.`Promotions`
(`Promotion_ID`, `Promotion_Type`, `Promotion_Title`)
VALUES
("Level-9001-Foods", "Facebook", "Come back! "),
("Level-9002-Foods", "Facebook", "Santa is Coming to Town"),
("Level-9003-Foods", "Facebook", "25 % off when you buy 25 dollars or more"),
("Level-9004-Foods", "Facebook", "Are You Still There?"),
("Level-9005-Foods", "Facebook", "4th of July"),
("Level-9006-Foods", "Facebook", "New Year New You"),
("Level-9007-Foods", "Facebook", "Refer a Friend"),
("Level-9008-Foods", "Facebook", "New Customers"),
("Level-9009-Foods", "Facebook", "Instagram Special"),
("Level-9010-Foods", "Facebook", "50% off when you buy 50 dollars or more");
```

Employees Table

```
INSERT INTO `final_project`.`Employees`
(`Employee_No`, `Emp_First_Name`, `Emp_Last_Name`, `Emp_Birth_Date`, `Emp_Hire_Date`,
`Emp_Department`)
VALUES
(100, "Sarah", "Paulsen", "1987-01-23", "2021-08-01", "Support"),
(101, "Deborah", "Wang", "1997-03-26", "2019-03-06", "HR"),
(102, "Julia", "Cambridge", "1990-04-01", "2021-01-01", "Operations"),
(103, "Bjorn", "Nordwall", "1989-03-02", "2019-12-01", "IT"),
(104, "Daniel", "Puello", "1996-06-09", "2020-01-12", "Support"),
(105, "Ruisheng", "Wang", "2000-01-01", "2020-02-02", "Support"),
(106, "Ilan", "Green", "1981-03-04", "2021-01-25", "Marketing"),
(107, "Patrick", "Heaton", "1994-10-05", "2021-01-10", "Marketing"),
(108, "Jeff", "Schiebe", "1955-11-19", "2020-06-16", "Operations"),
(109, "Rajendra", "Shirole", "1975-09-02", "2020-09-30", "Finance"),
(110, "Morgan", "McClure", "1994-07-03", "2020-11-19", "Sales");
```

b. Child Tables

Orders Table

```
INSERT INTO `final_project`.`Orders`
(`Order_No`, `Order_Date`, `Promotion_ID`, `Total_Amount`, `Customer_ID`)
VALUES
(12, "2021-01-25", "Level-9008-Foods", 39.60, "LF005");

INSERT INTO `final_project`.`Orders`
(`Order_No`, `Order_Date`, `Total_Amount`, `Shipped_Date`, `Customer_ID`)
VALUES
(29, "2021-01-27", 710.40, "LF022");
```

Support Table

```
INSERT INTO `final_project`.`Support`
(`Issue_ID`, `Date_Created`, `Attachment`, `Employee_No`, `Issue_email`)
VALUES
(6103, "2021-01-18", "N", 105, "keenan@it.com"),
(6105, "2020-12-01", "N", 105, "fernanda@gmail.com"),
(6109, "2021-01-09", "N", 105, "james@gmail.com"),
(6114, "2020-12-12", "N", 104, "myname@yourname.com"),
(6119, "2021-01-10", "N", 100, "arline@gmail.com");
```