Database Management Systems

Final Project

August Delbert and Max Chis

Submitted to Professor Russell Timco

INFSCI2710

August 3, 2021

**Table of Contents**

[**Database Overview**](#_x6wx2u9kmx3s) **4**

[**Assumptions**](#_vqdcp4zb262q) **7**

[**Graphical Schema**](#_5zhi28j3kd7u) **9**

[**Relational Schema Set**](#_43zsblmvg7xd) **10**

[**Analysis of the Normal Form**](#_e1i0abumkkp) **11**

[**Integrity Constraints**](#_a2d6nx6adzb5) **13**

[**DDL Code**](#_wsbqzm6jzm01) **14**

[**Queries Used to Implement functionality**](#_mc900ga3x9kt) **18**

[Search for a List of All Available Products](#_qgnh0y8uimt8) 18

[Search for products owned by a specific customer](#_oy3a4w8j8hvm) 18

[Search for all customers with or including given name](#_jkwloof2mpdj) 18

[Search for personal and contact information about a customer](#_hgdyc0alks6a) 18

[See general information for a specific case given the case number](#_8m16k3ltfee2) 18

[Search for open cases by status and timeframe](#_br815ybg4cjh) 19

[Search for closed cases by status and timeframe](#_r5ggaljtyrx3) 19

[Search for common resolutions by product](#_m2pomc48be37) 19

[Search for cases by product](#_9gdnz4wn04jv) 19

[Search for cases by customer](#_v823a56cz779) 19

[Number of Support Cases by Product (ordered by largest)](#_g2vkdx20he3o) 20

[Number of Support Cases Closed by Employee (ordered by largest)](#_c6e2prvi03nw) 20

[Number of Support Cases by Customer (ordered by largest)](#_iomv01vhfwnq) 20

[Number of Support cases by customer’s company(ordered by largest)](#_rkx5wolxdcsc) 20

[Total number of products sold, grouped by product](#_s3pd44ry33vx) 20

[The total number of products that have been sold by individual salespeople, grouped by product](#_6r737h1l50q8) 21

[The total amount of money a specific company has spent on products](#_4rixarmfrvnn) 21

[The number of cases a specific employee currently has opened and the number that employee has closed](#_79ei6u25gns) 21

[**Queries for Creation of New Data Entries/Tuples**](#_7yhzygm3txh1) **22**

[Insert new salesperson info](#_64e7dklp2o8t) 22

[Insert new customer information](#_x9v9g8sxzkcc) 22

[Insert new product information](#_tqpymu4hgpdn) 22

[Create a record of product ownership](#_mztyqtwhj5yx) 22

[Insert new case](#_m7je8ou5nz4k) 22

[Insert new case comments](#_fd636kulbhyr) 23

[Insert new resolution](#_850tkp49eqae) 23

[Insert new sell info](#_jgndzmd9syxi) 23

[**Queries to Modify and Delete Existing Data Entries/Tuples**](#_del4tcht01b7) **24**

[Update address of customer](#_5jtt4vah9ywi) 24

[Delete salesperson from table](#_3gq6e4ukkr86) 24

[Close case](#_r85ix2oecvjz) 24

[**Testing Efforts and Erroneous Cases**](#_7hqcs6htr5lp) **25**

[**Design Limitations and Possibilities for Improvements**](#_2dqi2vzalbxl) **32**

[**Github Repository**](#_udfxjs25mk0c) **35**

## 

## Database Overview

This database includes information on the company’s products and on information relevant to products, including which salespeople sell them, which customers have purchased them, issues related to the products, and resolutions to the issues. Assuming no user-based restrictions (such as customers only being able to look up information on the product catalog and regarding products they have purchased), users can look up a wide array of relevant information. Examples of the types of information that can be queried include: customers and the products they have purchased, cases and their resolutions, common resolutions by products, etc.

**Description of the Company and Its Products**

The company that has commissioned the design of this database, MaxAug Enterprises, produces useless inventions. The products are unique in that they serve no true purpose or benefit to society other than their entertainment value as useless inventions. They currently have five products on the market: 1) Alphalax, 2) Icetex, 3) Volt Tondom, 4) Zenlux, 5) Span Tinfind. Notably, the names of these products were chosen by use of a random name generator, because it helps to emphasize the senseless nature of their purpose.

**General Assumptions Regarding Application and Use of the Database**

For general use purposes, it is assumed that different portals and applications will be developed to enable different users to search for, access, and browse data that is relevant and of interest to them.

As a baseline, it is assumed that a generic, all-access portal would be available for all constituents where they could access a web page online and then search for and browse information related to all products offered by the company. Beyond this, different constituencies would be able to log in on the general website using unique credentials to access information and features that are relevant and specific to their role.

For example, after a customer has logged in using their unique credentials, they would have access to a customer portal that would enable them to see information about products they have already purchased. In addition to this, they would be able to access information about support cases they have requested related to the products they have purchased.

Similarly, salespersons and support employees would have their own respective portals where they could log in and browse for information related to their job duties and responsibilities. Salespersons would be able to search for information about customers they have been assigned to work with. This could include customers’ contact information, information about products customers have already purchased, and the amount of money the customers’ companies have already spent on products.

Likewise, employees would be able to log in and search for information about support cases. They could search for and browse information related to cases that have been opened or closed, comments embedded within cases, and resolutions for any specific cases.

Additionally, executives would have their own portal, or business intelligence system wherein they would be able to conduct queries regarding the number of products sold as well as the amount of income that has been grossed.

**Types of Users Accessing Database**

The types of database users are listed below, followed by descriptions of the ways in which they will be accessing and utilizing the database:

Potential Customers:

Potential customers will be able to use the database to browse a list of all products that are available for purchase.

Existing Customers:

In addition to the search features that are accessible for potential customers, existing customers will also be able to conduct queries to search for information about support cases they have requested for products that they own as well as resolutions that relate to those cases.

Company Salespersons:

Company salespersons will use the database to search and browse information about products and customers. A salesperson will also be able to update attribute information for customers who they are serving (name, address, state, zip, company name).

Database Administrators:

The database administrators will serve as stewards of the database. They will be able to add and alter tables and relations as necessary. They will also be able to enter new customer, product, and employee data into the system.

Company Employees:

Company employees will be able to create new cases whenever a new case is requested by a customer. Employees will also be able to note comments for specific cases and add resolutions for specific cases. Company employees will also be able to change case attributes. For example, they will be able to update a case’s STATUS to indicate if it is open (TRUE) or closed (FALSE).

Company IT Professionals:

Company Information Technology professionals will regularly review performance and operation of the database system to identify, diagnose, and address any possible issues. They will review data and test new features from architectural, design, and development perspectives. Accordingly, they will have access to all portals and levels of access within the system. This means they will also have the ability to create, edit, and update tables, queries, data entries, etc.

Company Executives:

Company executives will use the database to search for and review information related to general operations, efficiency, and effectiveness of the company’s sales and support processes. Specifically, they will be able to search for information about the total number of each different product that has been sold, the total number of products that have been sold by individual salespeople, and the amount of money a customers’ company has spent on products. They will also be able to track the number of cases opened and closed by specific employees.

-----------

Note: All of the user functions described in this section have been used to inform the development of queries to support functionality of the database system. For more detailed information, please refer to the section of this report titled Queries Used to Implement Functionality.

## 

## Assumptions

Current working assumptions include:

* All salespersons currently sell all products.
* Anyone who uses the database will create data entries in good faith and will not be creating entries that are nonsensical or improper. For example, numbers will not be entered into Customer or Employee NAME fields, or in a Customer’s COMPANY attribute.
* Some assumptions have been made regarding the relationships of different entities. These are outlined below:

|  |  |  |
| --- | --- | --- |
| **Relationship** | **Type** | **Notes** |
| Salesperson-Customer | One-to-Many | · One salesperson can serve many Customers  · Each customer will have one assigned Salesperson. |
| Salesperson-Product | Many-to-Many | · One Salesperson can sell many products  · One product can be sold by many different salespeople. |
| Customer-Case | One-to-Many | · A Customer may have requested zero support cases (a scenario where they have never had any issues with the products they have purchased) or they may have requested many cases for any number of products.  · A specific case will always be associated with a single customer. |
| Customer-Product | Many-to-Many | · A Customer may own zero products (a case in which they are working with a salesperson, but haven’t yet made any purchases) or they may own many of any of the different products.  · A specific product may not yet be owned by a customer, or it could have been purchased by a customer many different times. |
| Product-Case | One-to-Many | · Each case will be opened for one specific product. Each case will only ever be regarding a single product.  · Each product can have zero cases (perhaps when it is first put on the market and there may have not yet been any issues with it) or many cases. |
| Case-Resolution | One-to-One | · Each case will lead to one single resolution. It is possible for a case to have zero resolutions if it has not yet been closed.  · Each resolution will only ever be related to one case. |
| Employee-Case | One-to-Many | · Each case will be opened by and assigned to one employee.  · An employee may have opened zero cases (perhaps when they are first hired) or many cases. |
| Employee-Comment | One-to-Many | · An employee can note zero case comments or multiple case comments.  · A case comment will be noted by a single employee |
| Case-Comment | One-to-Many | · Each case comment relates to only one case.  · A single case may have zero or many related case comments. |

## 

## Graphical Schema

## Relational Schema Set

## Analysis of the Normal Form

**Normal Form of Database**

The Normal Form of our database is Second Normal Form. The reasons for why this is in 2NF, and the justifications for it being so, are detailed below.

Each of the tables included within the database is listed below along with a classification of its associated normal form:

|  |  |
| --- | --- |
| Table | Classification |
| SALESPERSONS | 2nd Normal Form\* |
| CUSTOMERS | 2nd Normal Form\* |
| PRODUCTS | Boyce Codd Normal Form |
| OWN | Boyce Codd Normal Form |
| SELL | Boyce Codd Normal Form |
| EMPLOYEES | 2nd Normal Form\* |
| CASES | Boyce Codd Normal Form |
| CASE\_COMMENTS | Boyce Codd Normal Form |
| RESOLUTIONS | Boyce Codd Normal Form |

**\*Note:** From a very literal standpoint, the tables that are designated with an asterisk are in second normal form, because they include both state and zip code information. Zip codes are functionally dependent upon states. If we wanted to convert each of these tables into BCNF / 3rd normal form, we could create an additional table of states and their associated zip codes. From there, within the Salespersons, Customers, and Employees tables, we could list the zip codes and remove the states. However, in terms of usage, this does not appear to be practical or useful. Accordingly, we believe it is more sensible to keep the states paired with the zip codes.

Additional analysis and discussion regarding the normal form within the overall context of this database is provided below:

Meets Criteria for First Normal Form

Our database meets the criteria for First Normal Form by simply not having multiple values within any single column. By the nature of being designed in SQL, it is quite difficult to have multiple values within a single column, absent some trickery like having multiple values contained within a comma-delimited string, which is not done here.

Meets Criteria for Second Normal Form

Our database similarly meets the criteria for Second Normal Form by having a single-column primary key and no other possible candidate keys. This can be identified with reference to the DDL statements, wherein no table holds more than one primary key. Foreign keys are included within some DDL statements, but for any table which contains a foreign key, different records within the table may hold the same foreign key even while other attributes in the two records differ from each other, rendering them incapable of being candidate keys.

Does Not Completely Meet Criteria for Third Normal Form

Our database, however, does not meet the criteria for Third Normal Form, as some columns in some tables exhibit a functional dependency. The clearest example of this is in the Salespersons table, where the columns **Zip Code** and **State** both exist, despite the fact that a zip code can be associated with only one state, and hence **State** can be said to be functionally dependent on **Zip Code**.

While this violates Third Normal Form, it is considered justifiable as addresses usually include both state and zip code in their most commonly-recognized format. While splitting these up into separate tables is possible, it would be unwieldy and contrary to how addresses are usually recorded. Additionally, it would make identifying and sorting Salespersons by locational attributes such as states and/or zip-code more difficult; if we had a table, for example, which solely linked states with zip codes, and a user wanted to find Salespersons by states, the query would first need to search the States-Zip Codes table to find corresponding zip codes, and then search a secondary table to find those salespersons whose zip codes correspond to those zip codes found. Such a measure would be doable, but unnecessarily complex.

Thus, by nature of this table failing to meet Third Normal Form, the database as a whole fails to meet Third Normal Form. However, as previously mentioned, this is considered defensible as the violations of Third Normal Form exist in a context where it is considered standard to include the two column types within the same table, even though a functional dependency does exist between them.

## Integrity Constraints

An array of integrity constraints (ICs) has been established and embedded within the DDL code to ensure that the database abides by certain principles and properties that reflect the design intentions and the real-life processes that are used by the company that has commissioned the development of this database. An overview of the ICs incorporated in the database is provided below:

* Regarding the need for proper formatting with phone numbers and emails, CONSTRAINTS have been added to the relevant tables (SALESPERSONS, CUSTOMERS, EMPLOYEES) to ensure that any new data entries abide by common formatting standards.
* Regarding the need to ensure essential attributes are not left blank or null, a “NOT NULL” IC was added to relevant fields along with a CONSTRAINT that the respective fields could not be entered as blank entries (e.g., **NAME != ''**).
* Regarding the need to enforce specific types of relationships between different entities (e.g., one-to-one, one-to-many, or many-to-many), tables were designed and coded to ensure that the relevant relationships would be enforced. Specifically:
  + For many-to-many relationships, an intermediary table was created to enable each of the respective entities to have as many different relationships with the other as necessary. This is specifically seen in the OWN table (which facilitates a many-to-many relationship between CUSTOMERS and PRODUCTS) and the SELL table (which facilitates a many-to-many relationship between SALESPERSONS and PRODUCTS).
  + For many-to-one and one-to-one relationships, any entity that is constrained to be associated with a single entry in another corresponding entity is set so that its ID is generated as an identity and then also so that its table includes the ID of the related table. This has the effect of ensuring that there will only ever be one unique version of the “one” entity and that that unique entry will only have one of the related entities that is able to be assigned to it at any given time. Conversely, if the corresponding entity is not also limited by a one-to-one relationship, then the related entity is not required to include the initial entity’s ID in its list of attributes. For example, each Customer is assigned only a single Salesperson whereas a Salesperson does not have any constraints about the number of Customers it can be related to.
* In some special instances, ICs were crafted to ensure that the table properties mirrored specific company processes. For example, with CASES, there can never be an instance of a Case that has a STATUS set to open but also has a TIMECLOSED attribute that is not null. Similarly, there can never be an instance of a Case that has its STATUS set to closed and a TIMECLOSED that is null or has not been specified. Accordingly, a constraint was developed within the CASES table to enforce these requirements.

## DDL Code

|  |
| --- |
| **CREATE TABLE SALESPERSONS(**  **SP\_ID INT GENERATED ALWAYS AS IDENTITY,**  **NAME VARCHAR(255) NOT NULL,**  **STREET VARCHAR(255),**  **CITY VARCHAR(255),**  **STATE VARCHAR(2),**  **ZIP VARCHAR(5),**  **EMAIL VARCHAR(255),**  **PHONE VARCHAR(25),**  **JOB\_TITLE VARCHAR(255),**  **PRIMARY KEY(SP\_ID),**  **CONSTRAINT CHK\_EMAIL CHECK (EMAIL LIKE '%\_@\_\_%.com' AND EMAIL NOT LIKE '%@@%' AND EMAIL NOT LIKE '%..%'),**  **CONSTRAINT CHK\_PHONE CHECK (PHONE LIKE '\_\_\_-\_\_\_-\_\_\_\_'),**  **CONSTRAINT CHK\_NAME CHECK (NAME != '')**  **);**  **CREATE TABLE CUSTOMERS(**  **CUST\_ID INT GENERATED ALWAYS AS IDENTITY,**  **SP\_ID INT NOT NULL,**  **NAME VARCHAR(255) NOT NULL,**  **STREET VARCHAR(255),**  **CITY VARCHAR(255),**  **STATE VARCHAR(2),**  **ZIP VARCHAR(5),**  **EMAIL VARCHAR(255),**  **PHONE VARCHAR(25),**  **COMPANY VARCHAR(50),**  **COMPANY\_GROSS\_INCOME\_DOLLARS FLOAT(25),**  **PRIMARY KEY(CUST\_ID),**  **FOREIGN KEY (SP\_ID) REFERENCES SALESPERSONS(SP\_ID),**  **UNIQUE (CUST\_ID, SP\_ID),**  **CONSTRAINT CHK\_NAME CHECK (NAME != ''),**  **CONSTRAINT CHK\_EMAIL CHECK (EMAIL LIKE '%\_@\_\_%.com' AND EMAIL NOT LIKE '%@@%' AND EMAIL NOT LIKE '%..%'),**  **CONSTRAINT CHK\_PHONE CHECK (PHONE LIKE '\_\_\_-\_\_\_-\_\_\_\_')**  **);**  **CREATE TABLE PRODUCTS(**  **P\_ID INT GENERATED ALWAYS AS IDENTITY,**  **NAME VARCHAR(255) NOT NULL,**  **DESCRIPTION VARCHAR(255) NOT NULL,**  **PRIMARY KEY(P\_ID),**  **UNIQUE (NAME),**  **CONSTRAINT CHK\_NAME CHECK (NAME != ''),**  **CONSTRAINT CHK\_DESCRIPTION CHECK (DESCRIPTION != '')**  **);**  **CREATE TABLE OWN(**  **O\_ID SERIAL,**  **CUST\_ID INT NOT NULL,**  **P\_ID INT NOT NULL,**  **PRIMARY KEY (O\_ID),**  **FOREIGN KEY (P\_ID) REFERENCES PRODUCTS(P\_ID),**  **FOREIGN KEY (CUST\_ID) REFERENCES CUSTOMERS(CUST\_ID),**  **CONSTRAINT OWNER UNIQUE (O\_ID, CUST\_ID)**  **);**  **CREATE TABLE EMPLOYEES(**  **EMP\_ID INT GENERATED ALWAYS AS IDENTITY,**  **NAME VARCHAR(255) NOT NULL,**  **STREET VARCHAR(255),**  **CITY VARCHAR(255),**  **STATE VARCHAR(2),**  **ZIP VARCHAR(5),**  **PHONE VARCHAR(25),**  **EMAIL VARCHAR(255),**  **PRIMARY KEY(EMP\_ID),**  **CONSTRAINT CHK\_NAME CHECK (NAME != ''),**  **CONSTRAINT CHK\_EMAIL CHECK (EMAIL LIKE '%\_@\_\_%.com' AND EMAIL NOT LIKE '%@@%' AND EMAIL NOT LIKE '%..%'),**  **CONSTRAINT CHK\_PHONE CHECK (PHONE LIKE '\_\_\_-\_\_\_-\_\_\_\_')**  **);**  **CREATE TABLE CASES(**  **CASE\_ID INT GENERATED ALWAYS AS IDENTITY,**  **SUMMARY VARCHAR(255) NOT NULL,**  **DESCRIPTION VARCHAR(255),**  **STATUS BOOLEAN NOT NULL,**  **EMP\_ID INT NOT NULL,**  **O\_ID INT NOT NULL,**  **TIMEOPENED DATE NOT NULL,**  **TIMECLOSED DATE,**  **PRIMARY KEY(CASE\_ID),**  **FOREIGN KEY (O\_ID) REFERENCES OWN(O\_ID),**  **FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEES(EMP\_ID),**  **CONSTRAINT CHK\_SUMMARY CHECK (SUMMARY != ''),**  **CONSTRAINT CHK\_TIMECLOSED1 CHECK ((STATUS = TRUE AND TIMECLOSED IS NULL) OR (STATUS = FALSE AND TIMECLOSED IS NOT NULL))**  **);**  **CREATE TABLE CASE\_COMMENTS(**  **COM\_ID INT GENERATED ALWAYS AS IDENTITY,**  **CASE\_ID INT NOT NULL,**  **COMMENT\_TIME TIME NOT NULL,**  **COMMENT\_DATE DATE NOT NULL,**  **CASE\_COMMENT VARCHAR(255) NOT NULL,**  **PRIMARY KEY(COM\_ID),**  **FOREIGN KEY (CASE\_ID) REFERENCES CASES(CASE\_ID),**  **UNIQUE (CASE\_ID, COM\_ID),**  **CONSTRAINT CHK\_CASE\_COMMENT CHECK (CASE\_COMMENT != '')**  **);**  **CREATE TABLE RESOLUTIONS(**  **RES\_ID INT GENERATED ALWAYS AS IDENTITY,**  **RES\_NAME VARCHAR(255) NOT NULL,**  **STEPS VARCHAR(255) NOT NULL,**  **CASE\_ID INT NOT NULL,**  **PRIMARY KEY(RES\_ID),**  **FOREIGN KEY (CASE\_ID) REFERENCES CASES(CASE\_ID),**  **UNIQUE(CASE\_ID),**  **CONSTRAINT CHK\_RES\_NAME CHECK (RES\_NAME != ''),**  **CONSTRAINT CHK\_STEPS CHECK (STEPS != '')**  **);**  **CREATE TABLE SELL(**  **SELL\_ID SERIAL,**  **SP\_ID INT NOT NULL,**  **P\_ID INT NOT NULL,**  **PRIMARY KEY (SELL\_ID),**  **FOREIGN KEY (SP\_ID) REFERENCES SALESPERSONS(SP\_ID),**  **FOREIGN KEY (P\_ID) REFERENCES PRODUCTS(P\_ID)**  **);**  **/\* Creating Indexes \*/**  **CREATE INDEX IX\_CUSTOMERS1 on CUSTOMERS(CUST\_ID, NAME, SP\_ID, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS);**  **CREATE INDEX IX\_SALESPERSONS1 on SALESPERSONS(SP\_ID, NAME, JOB\_TITLE);**  **CREATE INDEX IX\_PRODUCTS1 on PRODUCTS(P\_ID, NAME);**  **CREATE INDEX IX\_EMPLOYEES1 on EMPLOYEES(EMP\_ID, NAME);**  **CREATE INDEX IX\_CASES1 on CASES(CASE\_ID, O\_ID, EMP\_ID, STATUS, TIMEOPENED, TIMECLOSED);**  **CREATE INDEX IX\_CASE\_COMMENTS1 on CASE\_COMMENTS(COM\_ID, CASE\_ID, COMMENT\_TIME, COMMENT\_DATE);**  **CREATE INDEX IX\_RESOLUTIONS1 on RESOLUTIONS(RES\_ID, CASE\_ID, RES\_NAME);**  **CREATE INDEX IX\_OWN1 on OWN(O\_ID, CUST\_ID, P\_ID);** |

## 

## Queries Used to Implement functionality

### Search for a List of All Available Products

|  |
| --- |
| SELECT \*  FROM PRODUCTS |

### Search for products owned by a specific customer

|  |
| --- |
| SELECT C.NAME, P.NAME AS PRODUCT, COUNT(P.NAME)  FROM PRODUCTS P, CUSTOMERS C, OWN O  WHERE O.P\_ID = P.P\_ID AND O.CUST\_ID = C.CUST\_ID  AND C.NAME LIKE '%Bart Simpson%'  GROUP BY P.NAME, C.NAME  ORDER BY C.NAME |

### Search for all customers with or including given name

|  |
| --- |
| SELECT \*  FROM CUSTOMERS  WHERE NAME LIKE '%Bart%' |

### Search for personal and contact information about a customer

|  |
| --- |
| SELECT NAME, STREET, CITY, STATE, ZIP, EMAIL, PHONE, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS  FROM CUSTOMERS  WHERE NAME LIKE '%Bart%' |

### See general information for a specific case given the case number

(status, customer name, product, case summary, case description, case comments)

|  |
| --- |
| SELECT DISTINCT CA.CASE\_ID, CA.STATUS, CU.NAME, P.NAME, CA.SUMMARY, CA.DESCRIPTION, CA.TIMEOPENED, CA.TIMECLOSED, CC.CASE\_COMMENT  FROM CASES CA, CUSTOMERS CU, RESOLUTIONS R, CASE\_COMMENTS CC, OWN O, PRODUCTS P  WHERE CA.CASE\_ID = 1  AND CA.O\_ID = O.O\_ID AND O.CUST\_ID = CU.CUST\_ID AND O.P\_ID = P.P\_ID AND CC.CASE\_ID = CA.CASE\_ID |

### Search for open cases by status and timeframe

|  |
| --- |
| SELECT \*  FROM CASES  WHERE STATUS = TRUE  AND TIMEOPENED >= '20210303' |

### Search for closed cases by status and timeframe

|  |
| --- |
| SELECT \*  FROM CASES  WHERE STATUS = FALSE  AND TIMEOPENED >= '20210303' AND TIMECLOSED <= '20210508' |

### Search for common resolutions by product

|  |
| --- |
| SELECT P.NAME, R.RES\_NAME, R.STEPS  FROM RESOLUTIONS R, CASES C, OWN O, PRODUCTS P  WHERE R.CASE\_ID = C.CASE\_ID AND C.O\_ID = O.O\_ID AND O.P\_ID = P.P\_ID  AND P.NAME = 'Alphalax' |

### Search for cases by product

|  |
| --- |
| SELECT \*  FROM CASES C, PRODUCTS P, OWN O  WHERE O.O\_ID = C.O\_ID AND O.P\_ID = P.P\_ID  AND P.NAME = 'Alphalax' |

### Search for cases by customer

|  |
| --- |
| SELECT CU.NAME, CA.CASE\_ID, CA.STATUS, CA.EMP\_ID, CA.SUMMARY, CA.DESCRIPTION, CA.TIMEOPENED, CA.TIMECLOSED  FROM CASES CA, CUSTOMERS CU, OWN O  WHERE CA.O\_ID = O.O\_ID AND O.CUST\_ID = CU.CUST\_ID  AND CU.NAME LIKE '%Bart Simpson%' |

### Number of Support Cases by Product (ordered by largest)

|  |
| --- |
| SELECT P.NAME as PRODUCT, COUNT(CASE\_ID) as CASE\_COUNT  FROM CASES C, PRODUCTS P, OWN O  WHERE C.O\_ID = O.O\_ID AND O.P\_ID = P.P\_ID  GROUP BY P.NAME  ORDER BY COUNT(CASE\_ID) DESC |

### Number of Support Cases Closed by Employee (ordered by largest)

|  |
| --- |
| SELECT E.NAME, COUNT(CASE\_ID) as CASE\_COUNT  FROM CASES C, EMPLOYEES E  WHERE C.EMP\_ID = E.EMP\_ID AND C.STATUS = FALSE  GROUP BY E.NAME  ORDER BY COUNT(CASE\_ID) DESC |

### Number of Support Cases by Customer (ordered by largest)

|  |
| --- |
| SELECT CU.NAME, COUNT(CASE\_ID) as CASE\_COUNT  FROM CASES CA, CUSTOMERS CU, OWN O  WHERE CA.O\_ID = O.O\_ID AND O.CUST\_ID = CU.CUST\_ID  GROUP BY CU.NAME  ORDER BY COUNT(CASE\_ID) DESC |

### Number of Support cases by customer’s company(ordered by largest)

|  |
| --- |
| SELECT CU.COMPANY, COUNT(CASE\_ID) as CASE\_COUNT  FROM CASES CA, CUSTOMERS CU, OWN O  WHERE CA.O\_ID = O.O\_ID AND O.CUST\_ID = CU.CUST\_ID  GROUP BY CU.COMPANY  ORDER BY COUNT(CASE\_ID) DESC |

### Total number of products sold, grouped by product

|  |
| --- |
| SELECT P.NAME, COUNT(O.P\_ID) as NUMBER\_SOLD  FROM OWN O, PRODUCTS P  WHERE O.P\_ID = P.P\_ID  GROUP BY P.NAME  ORDER BY NUMBER\_SOLD DESC |

### The total number of products that have been sold by individual salespeople, grouped by product

|  |
| --- |
| SELECT S.NAME, P.NAME, COUNT(O.O\_ID) AS Total\_Products\_Sold  FROM SALESPERSONS S, PRODUCTS P, OWN O, CUSTOMERS C  WHERE S.SP\_ID = C.SP\_ID AND C.CUST\_ID = O.CUST\_ID AND O.P\_ID = P.P\_ID  GROUP BY S.NAME, P.NAME  ORDER BY S.NAME, Total\_Products\_Sold DESC |

### The total amount of money a specific company has spent on products

|  |
| --- |
| SELECT distinct COMPANY, SUM(COMPANY\_GROSS\_INCOME\_DOLLARS) as Total\_Gross  FROM CUSTOMERS  GROUP BY COMPANY  ORDER BY Total\_Gross DESC |

### The number of cases a specific employee currently has opened and the number that employee has closed

|  |
| --- |
| SELECT C.STATUS, COUNT(CASE\_ID) as CASE\_COUNT  FROM CASES C  WHERE C.EMP\_ID = 1  GROUP BY C.STATUS  ORDER BY CASE\_COUNT |

### 

## Queries for Creation of New Data Entries/Tuples

### Insert new salesperson info

|  |
| --- |
| insert into SALESPERSONS(NAME, STREET, CITY, STATE, ZIP, EMAIL, PHONE, JOB\_TITLE)  VALUES('Jane Doe', '4739 Collins Avenue', 'TRACY', 'MN', 56175, ‘Vorcy1964@jourrapide.com', '657-444-5555', 'Salesperson') |

### Insert new customer information

|  |
| --- |
| INSERT INTO CUSTOMERS (NAME, SP\_ID, STREET, CITY, STATE, ZIP, EMAIL, PHONE, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)  VALUES('Bart Simpson', 3, '123 Main St.', 'Springfield', 'IL', '12345', 'bart@gmail.com', '555-555-5555', 'Springfield Elementary', 10) |

### Insert new product information

|  |
| --- |
| INSERT INTO PRODUCTS(NAME, DESCRIPTION)  VALUES('Alphalax', 'Never go on a rollercoaster again!') |

### Create a record of product ownership

|  |
| --- |
| INSERT INTO OWN(CUST\_ID, P\_ID)  VALUES(1,1) |

### Insert new case

|  |
| --- |
| insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)  VALUES('Broken valve', 'Valve breaks upon attempting to turn with high-powered drill.', TRUE, 1, 1, TO\_DATE('20210304','YYYYMMDD'), NULL) |

### 

### Insert new case comments

|  |
| --- |
| insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)  VALUES (1, '12:00:00', TO\_DATE('20210304','YYYYMMDD'), 'Maybe don''t use a high-powered drill to turn a valve, idiot') |

### Insert new resolution

|  |
| --- |
| insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)  VALUES('Valve Fixed', 'Customer was instructed to reverse the high-powered drill and run it in the opposite direction.', 1) |

### Insert new sell info

|  |
| --- |
| insert into SELL(SP\_ID, P\_ID)  VALUES(1,1) |

### 

### 

### 

### 

### 

## Queries to Modify and Delete Existing Data Entries/Tuples

Note: some select, representative queries have been provided below. An exhaustive list of update/deletion queries have not been provided. This is done with the understanding that the structure and principles applied to develop the queries below can easily be transferred and extended to any other tables where the updates or deletions may be necessary.

### Update address of customer

|  |
| --- |
| UPDATE CUSTOMERS  SET STREET = '123 MAIN STREET', CITY = 'EXAMPLETOWN', STATE = 'CA', ZIP = '60999'  WHERE C.CUST\_ID = 1 |

### Delete salesperson from table

|  |
| --- |
| DELETE FROM SALESPERSONS  WHERE SP\_ID = 1 |

### Close case

|  |
| --- |
| UPDATE CASES  SET TIMECLOSED = TO\_DATE('01/02/2021', 'MM/DD/YYYY'), STATUS=FALSE  WHERE CASE\_ID = 1 |

## 

## Testing Efforts and Erroneous Cases

The full suite of testing efforts can be found below. In the case of testing, no erroneous cases were discovered outside of those which would be expected given the previously acknowledged limitations of the database.

|  |
| --- |
| **//Test happy path insertion into each table**  **//Test that each constraint is properly catching whatever error it is designed to catch.**  **//Assumptions: System users will not enter numeric characters where it is not appropriate. For example, they will not enter numbers in the CUSTOMERS name, city, state, or company fields.**  **/\*Note: Wherever a test is marked as PASS, it indicates that the DDL code worked as expected.**  **Either 1) an error results where we expected to receive an error or**  **2) the code did not result in an error when we did not expect any errors. \*/**  **//TEST SALESPERSONS**  **//HAPPY PATH**  **insert into SALESPERSONS(NAME, STREET, CITY, STATE, ZIP, EMAIL, JOB\_TITLE)**  **VALUES('Jane Doe', '4739 Collins Avenue', 'TRACY', 'MN', 56175, 'Vorcy1964@jourrapide.com', 'Salesperson')**  **//TEST EMAIL\_FORMAT CHECK -- Email entries that are not in the proper format should result in an error**  **insert into SALESPERSONS(NAME, STREET, CITY, STATE, ZIP, EMAIL, JOB\_TITLE)**  **VALUES('Brian J Wooley', '1733 Nicholas Street', 'Concordia', 'KS', 66901, 'BadFormat', 'Salesperson')**  **//Test Result: PASS**  **//TEST BLANK AND NULL NAMES**  **//Blank or null entries in NAME field should result in an error.**  **insert into SALESPERSONS(NAME, STREET, CITY, STATE, ZIP, EMAIL, JOB\_TITLE)**  **VALUES('', '4739 Collins Avenue', 'TRACY', 'MN', 56175, 'Vorcy1964@jourrapide.com', 'Salesperson')**  **insert into SALESPERSONS(NAME, STREET, CITY, STATE, ZIP, EMAIL, JOB\_TITLE)**  **VALUES(NULL, '4739 Collins Avenue', 'TRACY', 'MN', 56175, 'Vorcy1964@jourrapide.com', 'Salesperson')**  **//Test Result: PASS**  **//TEST CUSTOMERS**  **//HAPPY PATH**  **insert into CUSTOMERS(SP\_ID, NAME, STREET, CITY, STATE, ZIP, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)**  **VALUES(1, 'John Smith', '43 Main Street', 'Wilkinsburg', 'PA', 14447, 'Bob''s Burgers', 250000)**  **//TEST NULL NAMES**  **//Blank or null entries in NAME field should result in an error.**  **insert into CUSTOMERS(SP\_ID, NAME, STREET, CITY, STATE, ZIP, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)**  **VALUES(1, '', '43 Main Street', 'Wilkinsburg', 'PA', 14447, 'Bob''s Burgers', 250000)**  **insert into CUSTOMERS(SP\_ID, NAME, STREET, CITY, STATE, ZIP, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)**  **VALUES(1, NULL, '43 Main Street', 'Wilkinsburg', 'PA', 14447, 'Bob''s Burgers', 250000)**  **//Test Result: PASS**    **//TEST BLANK AND NULL SP\_ID**  **//Blank or null entries in SP\_ID field should result in an error.**  **insert into CUSTOMERS(SP\_ID, NAME, STREET, CITY, STATE, ZIP, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)**  **VALUES('', 'Test Name', '43 Main Street', 'Wilkinsburg', 'PA', 14447, 'Bob''s Burgers', 250000)**  **insert into CUSTOMERS(SP\_ID, NAME, STREET, CITY, STATE, ZIP, COMPANY, COMPANY\_GROSS\_INCOME\_DOLLARS)**  **VALUES(NULL, 'Test Name', '43 Main Street', 'Wilkinsburg', 'PA', 14447, 'Bob''s Burgers', 250000)**  **//Test Result: PASS**  **//TEST PRODUCTS**  **//HAPPY PATH**  **insert into PRODUCTS(NAME, DESCRIPTION)**  **//TEST BLANK AND NULL NAMES**  **//Blank or null entries in NAME field should result in an error.**  **insert into PRODUCTS(NAME, DESCRIPTION)**  **VALUES('', 'Test Description')**  **insert into PRODUCTS(NAME, DESCRIPTION)**  **VALUES(NULL, 'Test Description')**  **//Test Result: PASS**  **//TEST BLANK AND NULL DESCRIPTIONS**  **//Blank or null entries in DESCRIPTION field should result in an error.**  **insert into PRODUCTS(NAME, DESCRIPTION)**  **VALUES('Test Name', '')**  **insert into PRODUCTS(NAME, DESCRIPTION)**  **VALUES('Test Name', NULL)**  **//Test Result: PASS**  **//TEST EMPLOYEES**  **//HAPPY PATH**  **insert into EMPLOYEES(NAME, STREET, CITY, STATE, ZIP, PHONE, EMAIL)**  **VALUES('Antonio M Burnside', '105 Woodland Avenue', 'New Orleans', 'LA', 70112, '202-555-0543', 'yicaxe9769@ovooovo.com')**    **//TEST NULL NAMES**  **//Blank or null entries in NAME field should result in an error.**  **insert into EMPLOYEES(NAME, STREET, CITY, STATE, ZIP, PHONE, EMAIL)**  **VALUES('', '105 Woodland Avenue', 'New Orleans', 'LA', 70112, '202-555-0543', 'yicaxe9769@ovooovo.com')**  **insert into EMPLOYEES(NAME, STREET, CITY, STATE, ZIP, PHONE, EMAIL)**  **VALUES(NULL, '105 Woodland Avenue', 'New Orleans', 'LA', 70112, '202-555-0543', 'yicaxe9769@ovooovo.com')**  **//Test Result: PASS**  **//TEST EMAIL\_FORMAT CHECK**  **insert into EMPLOYEES(NAME, STREET, CITY, STATE, ZIP, PHONE, EMAIL)**  **VALUES('James V Haman', '4009 Walkers Ridge Way', 'SHASTA', 'CA', 96087, '202-555-0708', 'badformat@@@oeee...cccc')**    **//TEST PHONE\_FORMAT CHECK**  **insert into EMPLOYEES(NAME, STREET, CITY, STATE, ZIP, PHONE, EMAIL)**  **VALUES('Cynthia M Hall', '410 Patterson Fork Road', 'Lombard', 'IL', 60148, '3126717922', 'yicaxe9769@ovooovo.com')**  **//TEST OWN**  **//HAPPY PATH**  **insert into OWN(CUST\_ID, P\_ID)**  **VALUES(1,1)**  **//TEST MANY-TO-MANY RELATIONSHP BETWEEN CUSTOMERS AND PRODUCTS**  **/\*A Customer should be able to own many different products.**  **Conversely, any individual product should be able to be owned by many different customers.**  **The addition of multiple entries should not result in any errors.\*/**  **//Test one(CUST) to many(P)**  **insert into OWN(CUST\_ID, P\_ID)**  **VALUES (1,1), (1,2)**  **//Test Result: PASS**    **//Test one(P) to many(CUST)**  **insert into OWN(CUST\_ID, P\_ID)**  **VALUES (1, 1), (2,1)**  **//Test Result: PASS**  **//TEST CASES**  **//HAPPY PATH**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Broken valve', 'Valve breaks upon attempting to turn with high-powered drill.', TRUE, 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **//TEST BLANK AND NULL SUMMARY**  **//Blank or null entries in SUMMARY field should result in an error.**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('', 'Test Case', TRUE, 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES(NULL, 'Test Case', TRUE, 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **//Test Result: PASS**  **//TEST BLANK OR NULL STATUS**  **//Blank or null entries in STATUS field should result in an error.**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', "", 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', NULL, 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **//Test Result: PASS**  **//TEST NON-BOOLEAN STATUS**  **//Non-boolean entries in STATUS field should result in an error.**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', 3, 1, 2, TO\_DATE('20210304','YYYYMMDD'), NULL)**  **//Test Result: PASS**  **//TEST BLANK AND NULL TIMEOPENED**  **//Blank or null entries in TIMEOPENED field should result in an error.**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', TRUE, 1, 2, "", NULL)**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', TRUE, 1, 2, NULL, NULL)**  **//Test Result: PASS**    **//TEST CONDITIONAL TIMECLOSED**  **//TIMECLOSED should not accept an entry unless that status of the case is set to closed.**  **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', TRUE, 1, 2, TO\_DATE('20210304','YYYYMMDD'), TO\_DATE('20210305','YYYYMMDD'))**  **//The above result should be a failure**    **insert into CASES(SUMMARY, DESCRIPTION, STATUS, EMP\_ID, O\_ID, TIMEOPENED, TIMECLOSED)**  **VALUES('Test Case', 'Test Case', FALSE, 1, 2, TO\_DATE('20210304','YYYYMMDD'), TO\_DATE('20210305','YYYYMMDD'))**  **//The above result should be a success**  **//Test Result: PASS**    **//TEST CASE\_COMMENTS**  **//HAPPY PATH**  **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '12:00:00', TO\_DATE('20210304','YYYYMMDD'), 'Maybe don''t use a high-powered drill to turn a valve, idiot')**  **//TEST BLANK AND NULL COMMENT\_TIME**  **//Blank or null entries in COMMENT\_TIME field should result in an error.**  **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '', TO\_DATE('20210304','YYYYMMDD'), 'Maybe don''t use a high-powered drill to turn a valve, idiot')**    **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, NULL, TO\_DATE('20210304','YYYYMMDD'), 'Maybe don''t use a high-powered drill to turn a valve, idiot')**  **//Test Result: PASS**  **//TEST BLANK AND NULL CASE\_DATE**  **//Blank or null entries in CASE\_DATE field should result in an error.**  **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '12:00:00', '', 'Maybe don''t use a high-powered drill to turn a valve, idiot')**    **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '12:00:00', NULL, 'Maybe don''t use a high-powered drill to turn a valve, idiot')**  **//Test Result: PASS**  **//TEST BLANK AND NULL CASE\_COMMENT**  **//Blank or null entries in CASE\_COMMENT field should result in an error.**  **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '12:00:00', TO\_DATE('20210304','YYYYMMDD'), '')**    **insert into CASE\_COMMENTS(CASE\_ID, COMMENT\_TIME, COMMENT\_DATE, CASE\_COMMENT)**  **VALUES(1, '12:00:00', TO\_DATE('20210304','YYYYMMDD'), NULL)**  **//Test Result: PASS**  **//TEST RESOLUTIONS**  **//HAPPY PATH**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES('Valve Adaptation', 'Customer was instructed to turn the valve with their hand rather than a high-powered drill. Customer reports product is now working effectively', 1)**    **//TEST BLANK AND NULL RES\_NAME**  **//Blank or null entries in RES\_NAME field should result in an error.**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES('', 'Test Steps', 2)**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES(NULL, 'Test Steps', 2)**  **//Test Result: PASS**  **//TEST BLANK AND NULL STEPS**  **//Blank or null entries in STEPS field should result in an error.**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES('Test Resolution', '', 2)**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES('Test Resolution', NULL, 2)**  **//Test Result: PASS**  **//TEST ONE-TO-ONE RELATIONSHIP WITH CASES**  **//A single case cannot have more than one resolution. Accordingly, the entry of a second resolution for a case should result in an error.**  **insert into RESOLUTIONS(RES\_NAME, STEPS, CASE\_ID)**  **VALUES('Test Resolution', 'Test Steps', 1), ('Test Resolution2', 'Test Steps2', 1)**  **//Test Result: PASS**  **//TEST SELL**  **//HAPPY PATH**  **insert into SELL(SP\_ID, P\_ID)**  **VALUES(1,1)**  **//TEST MANY-TO-MANY RELATIONSHP BETWEEN SALESPERSONS AND PRODUCTS**  **/\*A Salesperson should be able to sell many different products.**  **Conversely, any individual product should be able to be sold by many different salespeople.**  **The addition of multiple entries should not result in any errors.\*/**  **//Test one(SP) to many(P)**  **insert into SELL(SP\_ID, P\_ID)**  **VALUES (1,1), (1,2)**  **//Test Result: PASS**    **//Test one(P) to many(SP)**  **insert into SELL(SP\_ID, P\_ID)**  **VALUES (1, 1), (2,1)**  **//Test Result: PASS** |

## 

## Design Limitations and Possibilities for Improvements

As the initial version of the database has been developed, several limitations and possibilities for potential improvements have been identified. The identified limitations are listed below followed by recommendations for how the database system can be improved to better address the issue in the future.

**Limitations regarding management of data related to the amount of money grossed for each individual product:**

In the current implementation of the database, all information about COMPANY\_GROSS\_INCOME\_DOLLARS is aggregated into a single attribute within the CUSTOMERS entity. As we have worked with the system, we have realized that it does not provide a means to parse information about how much money has been spent by a customer’s company on individual products. From an executive’s standpoint, it would be useful to be able to see how much money a company is spending on different products and how much money each separate product is grossing. For the next version of the database, we would recommend adding the following features:

* An additional entity representing companies that have purchased products. This COMPANIES entity could include attributes related to the size, nature, and industry of the company which may be useful in conducting analysis about the sectors where demand for the products is greatest.
* Creation of additional attributes within either the COMPANIES, CUSTOMERS and/or PRODUCTS entities that would enable tracking of the amount of money grossed per product.
* A means to keep track of how much money each company has spent on different products in addition to the gross income dollars that are aggregated across all product purchases.
* Development of additional queries and associated indexes that would enable an executive to search for the gross income dollars per product, the gross income dollars per product per customer, and the gross income dollars per product per company.

**Limitations regarding ability to track and create meaningful queries related to types of cases and resolutions**

From an executive standpoint, it would be useful to conduct review cases and resolutions based on the type or classification of problems or issues that are occurring. For example, a customer may be requesting a support case because they do not know how to operate or use a product, or they may be requesting a case because the product is failing or not functioning in the way that they would expect it to.

Although the CASES and RESOLUTIONS have titles and descriptions, there is currently no means to track or analyze cases and resolutions by the nature and type of problems that have occurred. This ability to track and review cases based on the nature and type of problems that have occurred would be useful for several different constituencies. For example, SALESPERSONS could be aware of the frequency of common issues and use this as they are guiding and advising potential customers. EMPLOYEES would be able to search for CASES and RESOLUTIONS by classification and use that information as a reference that could help them resolve similar issues when brought up in different cases. Executives would also be able to use this information to track where the greatest needs for quality improvement and prioritization in future product development initiatives.

Accordingly, for the next version of the database, we would recommend the following:

* Addition of attributes within the CASES and/or RESOLUTIONS entities that would enable a specific case or resolution to be classified by a TYPE of problem or issue that is occurring.
* Creation of queries and associated indexes that would enable relevant stakeholders to search for CASES and/or RESOLUTIONS by type as well as the number of CASES and/or RESOLUTIONS by type.

**Limitations regarding deletion and removal of data entries**

As it has currently been designed, the system is not yet robust in the ways in which it would handle and account for the deletion of data entries. For example, there may be cases in which a customer decides to return a product. This would have ramifications for the system that have not yet been fully accounted for. In the case where a customer returns a product, they are presumably no longer the owner of the product. They may have also requested previous support cases for the product which they no longer own. If they are no longer an owner, this would affect the OWN table, which would contain data that represents the former relationship between the customer and the product they used to own. Additionally, the CASES table relies on a foreign key from the OWN table which is based on the UNIQUE combination of the customer’s ID (CUST\_ID) and the own ID (O\_ID).

Considering this, we would recommend the following:

* More closely review and understand the ramifications of the deletion of a tuple from the CUSTOMERS table.
* Create triggers that would help the system to effectively respond to the deletion of any CUSTOMERS tuples by appropriately modifying any tuples in other tables that are related to and constrained by the respective tuple that has been removed from the CUSTOMERS table.
* Review the SALESPERSONS and EMPLOYEES tables in a similar way to understand the ramifications of the deletion of any tuples and then follow the same processes described above to create adaptive triggers where necessary.

**Limitations regarding the definition and constraints of the SALESPERSONS to PRODUCTS and SALESPERSONS to CUSTOMERS relationships.**

Currently, it is assumed that all Salespersons can sell all products. There are no restrictions currently placed on the types of products that can be sold by a specific salesperson. However, in a scenario where the company continues to grow to a large enough size where salespeople may begin to specialize and sell only certain products, the system is not currently designed in a way that would enforce and restrict the sale of a specific product to a customer based on the salespeople who are designated as being able to sell the specified product.

With this in mind, we would recommend the following:

* In the next version of the database, explore ways to create sensible constraints for the CUSTOMERS, SALESPERSONS, and OWN tables to ensure that a Customer can only purchase products from a salesperson who is able to sell the product in question.
* Similarly, consider the relationship of CUSTOMERS to SALESPERSONS. In the current scenario, each customer can only have one designated salesperson. However, in a scenario where salespeople do not sell all products, the relationship between CUSTOMERS and SALESPERSONS may potentially need to be reconsidered so that customers could have multiple related salespeople. This is only a hypothetical scenario, but it’s something to keep in mind as the company would continue to grow and develop its processes and the associated systems.

## 

## Github Repository

To better assist with collaboration, our code was developed and hosted on the following Github Repository. This repository also provides a format by which our source code can be more easily accessed and viewed.

<https://github.com/maxachis/INFSCI2710-Project>