I. INTRODUCTION

I have chosen my Aunt's small retail store as the basis for my capstone project. She operates two store, each staffed with one employee. Each store usually serves 20 to 100 customers daily. Currently, they use a partially customized system for sales transactions, which involves barcode scanning but is not yet fully developed. This is why I have selected this project for my capstone.

My goal is to develop a Retail Sales Analysis System for my Aunt's business. The purpose of this system is to provide valuable insights and support data-driven decision-making for her retail operation. This system is intended to assist in monitoring and optimizing various aspects of the business, including sales performance and inventory management. By utilizing data from sales transactions, customer interactions, and product inventory, the system will enable us to make informed choices to improve profitability, enhance customer satisfaction, and foster business growth.

5 BUSINESS QUESTIONS

1. What are the total sales for each product?
2. Which products have the highest and lowest sales volume?
3. Which products generate the highest and lowest profit?
4. How does stock availability impact sales?
5. How can we optimize inventory levels for different product?

INTENDED REPORTS

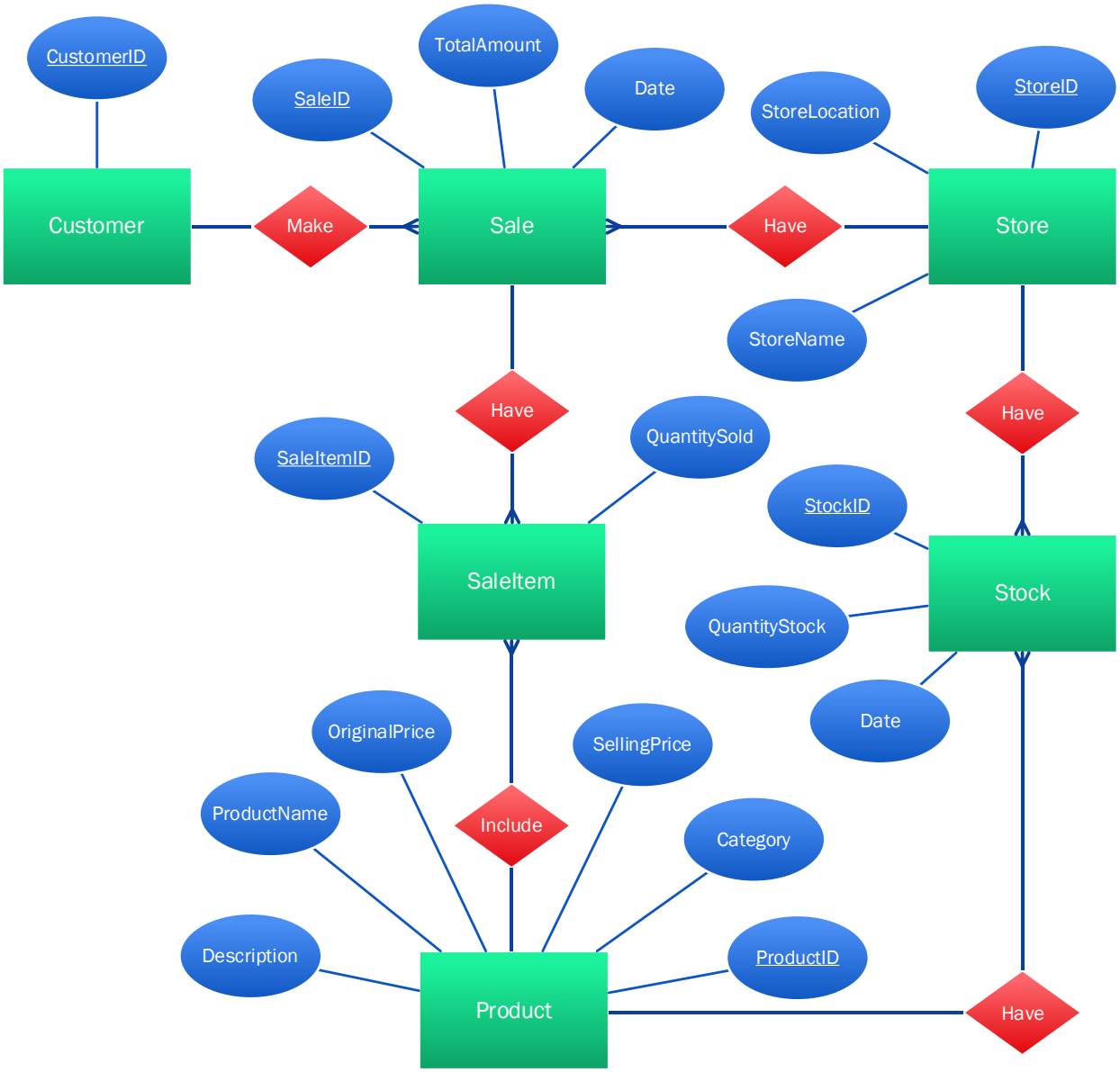
1. SALES PERFORMANCE

This report is related to the first 3 business questions. It is to gain insights into the Retail Store's sold product performance, allowing for informed decision-making, resource allocation, and strategy adjustments to maximize sales and profitability. The first question is about the breakdown of total sales per product that contribute the most to overall revenue. The second one is to identify the products that have highest and lowest quantities sold for inventory and product strategy. The third one is to determine which products are the most and least financially rewarding that will guide us on pricing strategy.

2. INVENTORY MANAGEMENT

This report is about optimizing inventory levels for different products to improve sales. It is related to the last 2 business questions. It aims to understand how to use data and analysis to ensure that the Retail Store maintains an appropriate amount of each product in its inventory. This helps in avoiding problems like overstocking or understocking, which can affect the Retail Store's operations and profitability. In simple terms, it's about making sure you have the right amount of products on hand to meet customer demand and minimize unnecessary costs.

Figure 1. ENTITY RELATIONSHIP DIAGRAM [RETAIL SALES ANALYSIS SYSTEM]



The Retail Store System will have 4 main entities

1. *Customer* – represents the customers who make purchases
   Attributes: *CustomerID*
   CustomerID only cause normally, the store doesn't actually take info from customer
2. *Product* – represents the products available
   Attributes: *ProductID, ProductName, Description, Category, OriginalPrice, SellingPrice*
3. *Store* – represents the stock information
   Attributes: *StoreID, StoreName, StoreLocation*
4. *Sale* – represents sales transactions
   Attributes: *SaleID, TotalAmount, Date*

After elimination of recursive and redundant relationship, we also need to include other entities such as

5. *Stock* – represents the stock information
   Attributes: *StockID, QuantityStock, Date*
6. *SaleItem* – represent the product sold
   Attributes: *SaleItemID, QuantitySold*

We could add more but for the project, I stop here to keep it small, simple and easy to understand.

Assumptions:

1. Each product has a unique ProductID.
2. Each stock entry has a unique StockID.
3. Each customer has a unique CustomerID.
4. Each sale has a unique SaleID.
5. Each sale item has a unique SaleItemID.
6. Each store has a unique StoreID.

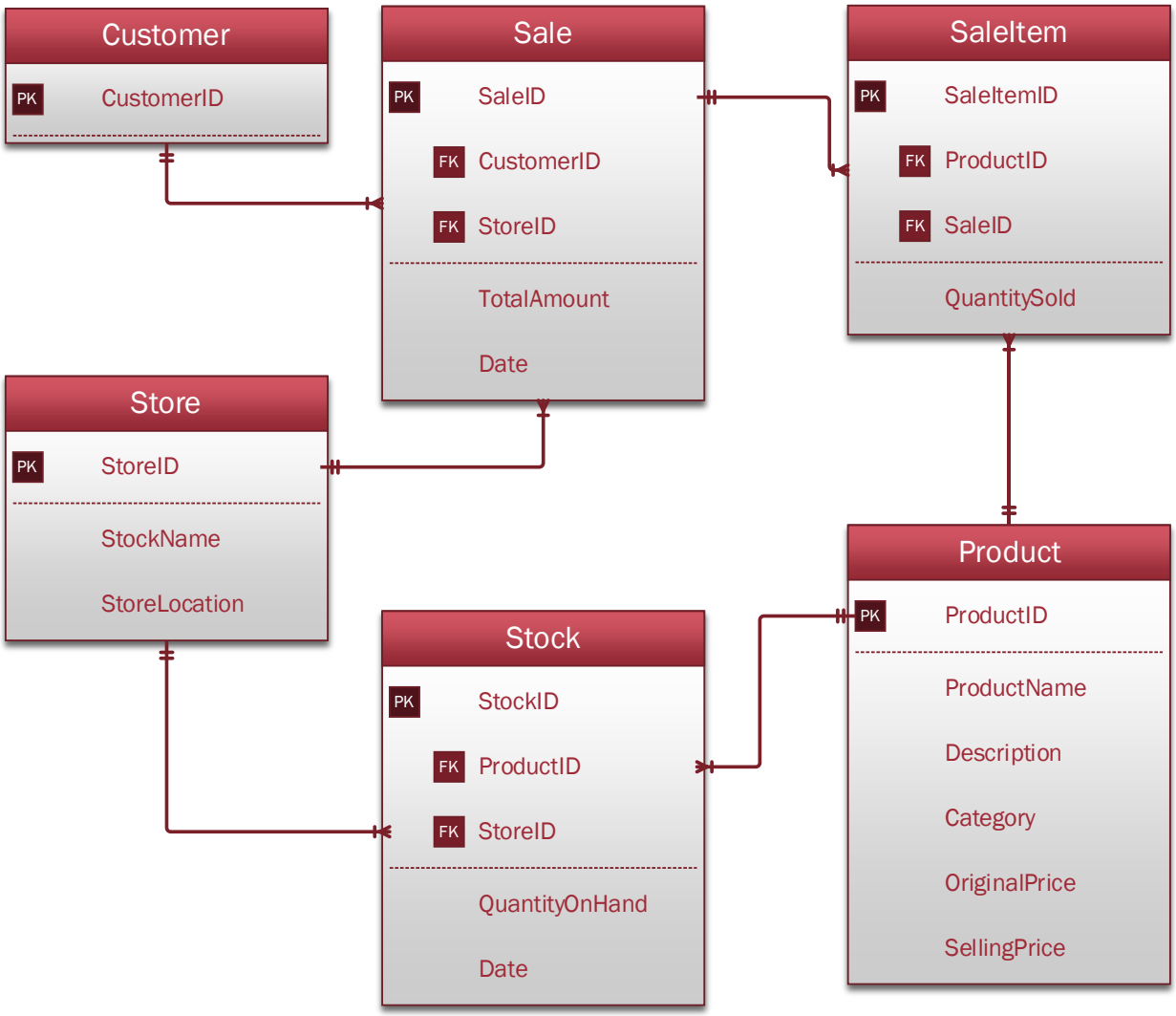
Cardinality:

1. Product (One) – Stock (Many): One Product can have multiple Stock entries.
2. Product (One) – Sale Item (Many): One Product can be included in multiple Sale Items.
3. Customer (One) – Sale (Many): One Customer can make multiple Sales.
4. Sale (One) – Sale Item (Many): One Sale can have multiple Sale Items.
5. Store (One) – Stock (Many): One Store can have multiple Stock entries.
6. Store (One) – Sale (Many): One Store can have multiple Sales.

II. NORMALIZED PROCESS

First Normal Form (1NF): In the design above, each column of the table contains atomic values and that each row is uniquely identifiable.
Second Normal Form (2NF): Remove partial dependencies. A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the entire primary key.
In our design, the tables are already in 2NF because all attributes in each table are fully functionally dependent on the primary key.
Third Normal Form (3NF): Remove transitive dependencies. A table is in 3NF if it is in 2NF and it does not have transitive dependencies. In our design, it is already in 3NF.
After the normalized process done, the design from relational model haven't change to normalized model as shown in Figure 2.

Figure 2. RELATIONAL MODEL AND NORMALIZED MODEL



III. DIMENSIONAL PROCESS

A. The 4-step Dimensional Modeling Process

Step 1: Select the business process

Business processes are the operational activities performed by the organization.

a. Sales – represented by Sale_Fact table
b. Inventory Management – represented by Stock_Fact table

Step 2: Declare the grain

Grain refers to the lowest level at which data is captured by a given business process.

a. Sales
    1. Sales by product by store by day
    2. Sales volume by product by store by day
b. Inventory Management
    1. Stock by product by store by day

Step 3: Identify the Dimensions

Dimensions provide the "who, what, where, when, why, and how" context surrounding a business process event.

a. Sales
    1. Customer_Dim [who]
    2. Product_Dim [what]
    3. Store_Dim [where]
    4. Date_Dim [when]
b. Inventory Management
    1. Product_Dim [what]
    2. Store_Dim [where]
    3. Date_Dim [when]

Step 4: Identify the Facts

Facts are the measurements that result from a business process event and are almost always numeric.

a. Sales
    1. TotalAmount – will be the fact for the Sales
    2. QuantitySold – will be the fact for the Sales volume
b. Inventory Management
    1. QuantityStock – will be the fact for the Stock

B. Creation of Fact Table

A *fact table* contains the numeric measures produced by an operational measurement event in the real world. At the lowest grain, a fact table row corresponds to a measurement event and vice versa. Thus the fundamental design of a fact table is entirely based on a physical activity

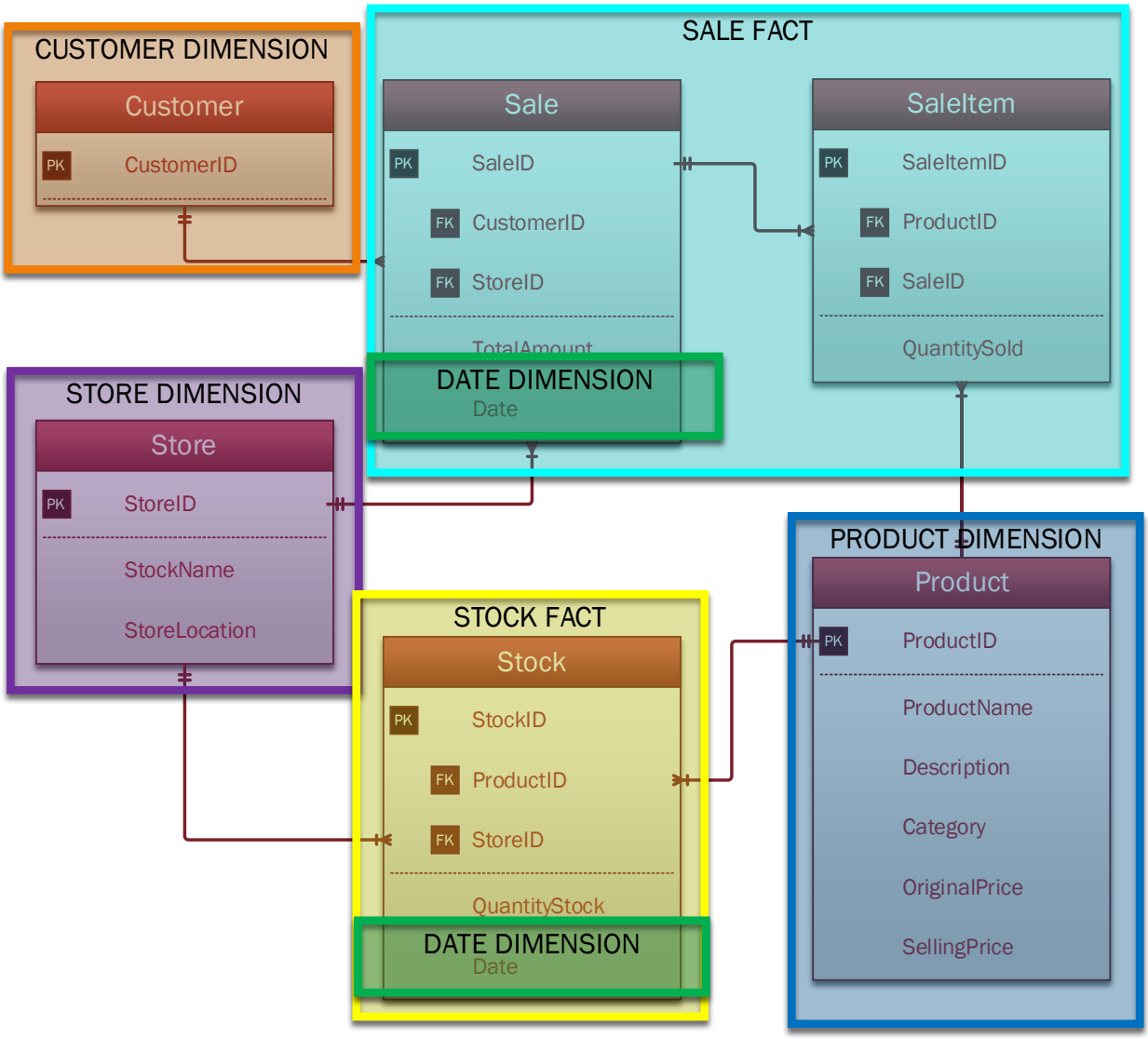
and is not influenced by the eventual reports that may be produced. In addition to numeric measures, a fact table always contains foreign keys for each of its associated dimensions, as well as optional degenerate dimension keys and date/time stamps. Fact tables are the primary target of computations and dynamic aggregations arising from queries. – Kimball Group

With the information above and the 4-step Dimensional Modeling, then we create the fact table as shown in Figure 3.

Figure 3. Fact Table

C. Source to Target Map

Figure 4. SOURCE TO TARGET MAP



Source-to-target mapping refers to the process of identifying and defining how data is transferred and transformed from a source system to a target system. Figure 4 is the Source-to-target mapping for the Retail Sales Analysis System.

D. Designing Dimension Table Using Dimension Normal Form

Step 1 (DNF1): Normalized Model (3NF)

We have previously completed the task, as illustrated in Figure 2.

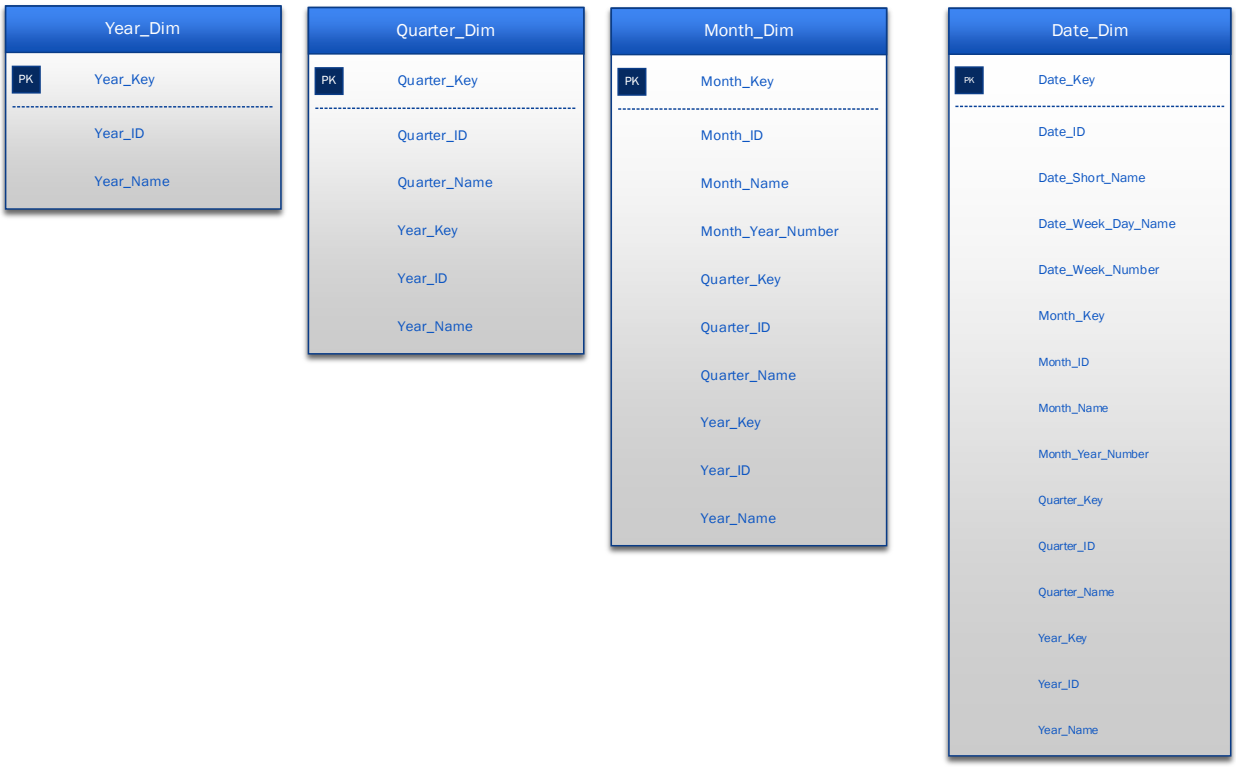
Step 2 (DNF2): Denormalized into the Dimensions

First, we need to denormalized Customer_Dim, Product_Dim, and Store_Dim as shown in Figure 5.

Figure 5. Denormalized Customer_Dim, Product_Dim, and Store_Dim

We also need to create dimension and denormalized Date_Dim as shown in Figure 5.

Figure 6. Denormalized Date_Dim



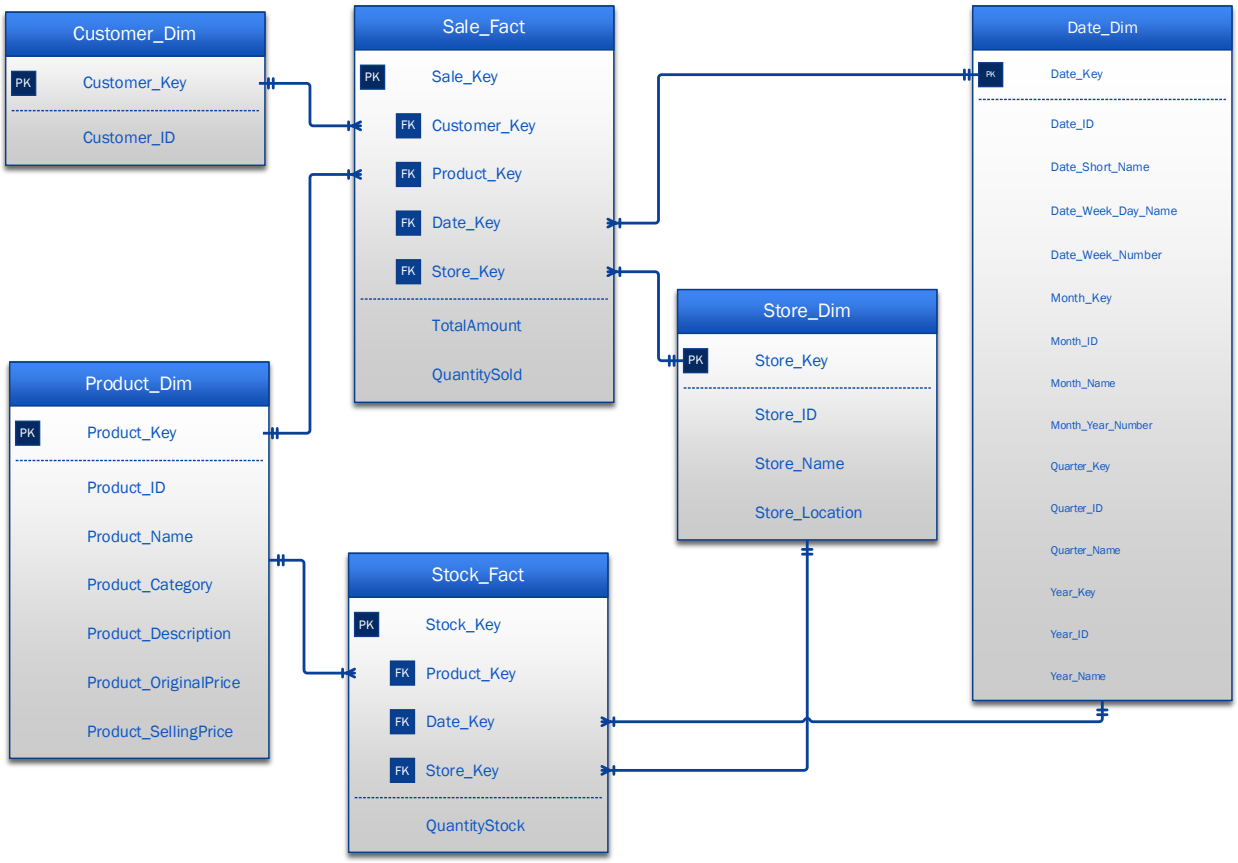
Process Summary of DNF2

- Product Table → Product_Dim
- Customer Table → Customer_Dim
- Store Table → Store_Dim
- Creating Date_Dim

Step 3 (DNF3): Attach Dimensions to Fact Table

We have previously completed the task, as illustrated in Figure 3.

Figure 3. DIMENSIONAL MODEL



IV. SQL SCRIPT FOR CREATION OF NORMALIZED MODEL

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace = '';

SET default_with_oids = false;


----------------------------------------
---          Drop Tables          ---
----------------------------------------
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS sale;
DROP TABLE IF EXISTS saleitem;
DROP TABLE IF EXISTS customer;
DROP TABLE IF EXISTS store;
DROP TABLE IF EXISTS stock;

--------------------------------------------------------------------------------
-- Name: product; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------

CREATE TABLE product (
    productid smallint NOT NULL PRIMARY KEY,
    productname varchar(255) NOT NULL,
    description text,
    category varchar(30),
    originalprice numeric(10, 2),
    sellingprice numeric(10, 2)
);
```

```
--------------------------------------------------------------------------------
-- Name: store; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
CREATE TABLE store (
   storeid smallint NOT NULL PRIMARY KEY,
   storename varchar(40) NOT NULL,
   storelocation varchar(255)
);

--------------------------------------------------------------------------------
-- Name: customer; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
CREATE TABLE customer (
   customerid smallint NOT NULL PRIMARY KEY
);

--------------------------------------------------------------------------------
-- Name: sale; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
CREATE TABLE sale (
   saleid smallint NOT NULL PRIMARY KEY,
   customerid smallint NOT NULL REFERENCES customer(customerid),
   storeid smallint NOT NULL REFERENCES store(storeid),
   totalamount numeric(10, 2),
   saledate date
);

--------------------------------------------------------------------------------
-- Name: saleitem; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
CREATE TABLE saleitem (
   saleitemid smallint NOT NULL PRIMARY KEY,
   productid smallint NOT NULL REFERENCES product(productid),
   saleid smallint NOT NULL REFERENCES sale(saleid),
   quantitysold smallint
);
```

```
--------------------------------------------------------------------------------
-- Name: stock; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
CREATE TABLE stock (
   stockid smallint NOT NULL PRIMARY KEY,
   productid smallint NOT NULL REFERENCES product(productid),
   storeid smallint NOT NULL REFERENCES store(storeid),
   quantitystock smallint,
   stockdate date
);
```

V. SQL SCRIPT FOR CREATION OF DIMENSIONAL TABLE

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace = '';

SET default_with_oids = false;



--------------------------------------------------------------------------------
---                    Drop Tables                             ---
--------------------------------------------------------------------------------
DROP TABLE IF EXISTS product_dim;
DROP TABLE IF EXISTS customer_dim;
DROP TABLE IF EXISTS store_dim;
DROP TABLE IF EXISTS date_dim;

--------------------------------------------------------------------------------
--product_dim
```

```
-------------------------------------------------------------------------
CREATE TABLE product_dim(
product_key int PRIMARY KEY,
product_id int,
product_name varchar (255),
product_category varchar(40),
product_description text,
product_originalprice numeric(10, 2),
product_sellingprice numeric(10, 2)
);

-------------------------------------------------------------------------
--date_dim
-------------------------------------------------------------------------
CREATE TABLE date_dim (
date_key int PRIMARY KEY,
date_id varchar(30),
date_short_name varchar(30),
date_day_of_week_name  varchar(60),
date_day_of_week_number  varchar(30),
month_key int,
month_id varchar(30),
month_name varchar(30),
month_year_number varchar(30),
quarter_key int,
quarter_id varchar(30),
quarter_name varchar(30),
year_key int,
year_id varchar(30),
year_name varchar(30)
);

-------------------------------------------------------------------------
--customer_dim
-------------------------------------------------------------------------
CREATE TABLE customer_dim (
customer_key int PRIMARY KEY,
customer_id int
);
```

```
-------------------------------------------------------------------------
--store_dim
-------------------------------------------------------------------------
CREATE TABLE store_dim (
store_key int PRIMARY KEY,
store_id int,
store_name varchar (60),
store_location varchar(255)
);


-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
--dimension families
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------

-------------------------------------------------------------------------
--year_dim --date_dim family
-------------------------------------------------------------------------
CREATE TABLE year_dim (
year_key int PRIMARY KEY,
year_id varchar(10),
year_name varchar(10)
);

--ALTER TABLE year_dim
--ADD PRIMARY KEY (year_key);
-------------------------------------------------------------------------
--quarter_dim --date_dim family
-------------------------------------------------------------------------
CREATE TABLE quarter_dim (
quarter_key int PRIMARY KEY,
quarter_id varchar(30),
quarter_name varchar(30),
year_key int,
year_id varchar(10),
year_name varchar(10)
);
```

```
-------------------------------------------------------------------------
--month_dim --date_dim family
-------------------------------------------------------------------------
CREATE TABLE month_dim (
month_key int PRIMARY KEY,
month_id varchar(10),
month_name varchar(30),
month_year_number varchar(10),
quarter_key int,
quarter_id varchar(30),
quarter_name varchar(30),
year_key int,
year_id varchar(10),
year_name varchar(10)
);
```

VI. SQL SCRIPT FOR CREATION OF FACT TABLE

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace = '';

SET default_with_oids = false;
```

```
-------------------------------------------------------------------------
--- drop tables
-------------------------------------------------------------------------
DROP TABLE IF EXISTS sale_fact CASCADE;
-------------------------------------------------------------------------
--sale_fact
-------------------------------------------------------------------------
CREATE TABLE sale_fact (
                    sale_key serial PRIMARY KEY,
                    customer_key int REFERENCES customer_dim (customer_key),
                    product_key int REFERENCES product_dim (product_key),
                    date_key int REFERENCES date_dim (date_key),
                    store_key int REFERENCES store_dim (store_key),
                    totalamount numeric(10, 2),
                    quantitysold int
);

-------------------------------------------------------------------------
--- drop tables
-------------------------------------------------------------------------
DROP TABLE IF EXISTS stock_fact CASCADE;
-------------------------------------------------------------------------
--stock_fact
-------------------------------------------------------------------------
CREATE TABLE stock_fact (
                    stock_key serial PRIMARY KEY,
                    product_key int REFERENCES product_dim (product_key),
                    date_key int REFERENCES date_dim (date_key),
                    store_key int REFERENCES store_dim (store_key),
                    quantityonhold int
);
```

VII. SQL SCRIPT FOR DATA INSERT INTO NORMALIZED MODEL DATABASE

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace = '';

SET default_with_oids = false;


--------------------------------------------------------------------------------
-- Name: INSERT product; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO product VALUES (1, 'Piattos', 'Jack ''n Jill Piattos Cheese 40g', 'Chips', '15.00', '17.00');
INSERT INTO product VALUES (2, 'VCut', 'Jack ''n Jill VCut Onion and Garlic 60g', 'Chips', '36.00', '39.00');
INSERT INTO product VALUES (3, 'Chippy', 'Jack ''n Jill Chippy BBQ 27g', 'Chips', '6.00', '8.00');
INSERT INTO product VALUES (4, 'Mr Chips', 'Jack ''n Jill Mr Chips Nacho Cheese 26g', 'Chips', '6.00',
'8.00');
INSERT INTO product VALUES (5, 'Clover Chips', 'Clover Chips Barbecue 55g', 'Chips', '21.00', '25.00');
INSERT INTO product VALUES (6, 'Century Tuna', 'Century Tuna Flakes in Oil 155grams', 'Canned Goods',
'33.00', '38.00');
INSERT INTO product VALUES (7, '555 Sardines', '555 Sardines in Tomato Sauces 155g', 'Canned Goods',
'21.00', '24.00');
INSERT INTO product VALUES (8, 'Argentina Corned Beef', 'Argentina Corned Beef 150g', 'Canned
Goods', '30.50', '35.00');
INSERT INTO product VALUES (9, 'Skyflakes', 'Skyflakes Crackers Handy Pack 200g', 'Biscuit', '40.00',
'45.00');
INSERT INTO product VALUES (10, 'Fita', 'Fita Biscuits Singles 30G 10s', 'Biscuit', '60.00', '65.00');
```

```
--------------------------------------------------------------------------------
-- Name: INSERT INTO store; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO store VALUES (1, 'Chennai Mini Mart 1', 'Rodriguez Ave, Brgy. 34, Bacolod, Negros Occidental');
INSERT INTO store VALUES (2, 'Chennai Mini Mart 2', 'Eroreco Subd., Brgy. Mandalagan, Bacolod, Negros Occidental');

--------------------------------------------------------------------------------
-- Name: INSERT INTO customer; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO customer VALUES (1);
INSERT INTO customer VALUES (2);
INSERT INTO customer VALUES (3);
INSERT INTO customer VALUES (4);
INSERT INTO customer VALUES (5);

--------------------------------------------------------------------------------
-- Name: INSERT INTO sale; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO sale VALUES (1, 1, 1, 85.00, '2023-10-12');
INSERT INTO sale VALUES (2, 1, 1, 380.00, '2023-10-12');
INSERT INTO sale VALUES (3, 1, 1, 325.00, '2023-10-12');
INSERT INTO sale VALUES (4, 2, 1, 273.00, '2023-10-12');
INSERT INTO sale VALUES (5, 2, 1, 120.00, '2023-10-12');
INSERT INTO sale VALUES (6, 2, 1, 175.00, '2023-10-12');
INSERT INTO sale VALUES (7, 2, 1, 225.00, '2023-10-12');
INSERT INTO sale VALUES (8, 2, 1, 325.00, '2023-10-12');
INSERT INTO sale VALUES (9, 3, 2, 760.00, '2023-10-12');
INSERT INTO sale VALUES (10, 3, 2, 480.00, '2023-10-12');
INSERT INTO sale VALUES (11, 4, 2, 80.00, '2023-10-12');
INSERT INTO sale VALUES (12, 4, 2, 125.00, '2023-10-12');
INSERT INTO sale VALUES (13, 5, 1, 450.00, '2023-10-12');
INSERT INTO sale VALUES (14, 5, 1, 650.00, '2023-10-12');
```

```
--------------------------------------------------------------------------------
-- Name: INSERT INTO saleitem; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO saleitem VALUES (1, 1, 1, 5);
INSERT INTO saleitem VALUES (2, 6, 2, 10);
INSERT INTO saleitem VALUES (3, 10, 3, 5);
INSERT INTO saleitem VALUES (4, 2, 2, 7);
INSERT INTO saleitem VALUES (5, 3, 2, 15);
INSERT INTO saleitem VALUES (6, 8, 2, 5);
INSERT INTO saleitem VALUES (7, 9, 2, 5);
INSERT INTO saleitem VALUES (8, 10, 2, 5);
INSERT INTO saleitem VALUES (9, 6, 3, 20);
INSERT INTO saleitem VALUES (10, 7, 3, 20);
INSERT INTO saleitem VALUES (11, 4, 4, 10);
INSERT INTO saleitem VALUES (12, 5, 4, 5);
INSERT INTO saleitem VALUES (13, 9, 5, 10);
INSERT INTO saleitem VALUES (14, 10, 5, 10);

--------------------------------------------------------------------------------
-- Name: INSERT INTO stock; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO stock VALUES (1, 1, 1, 200, '2023-10-01');
INSERT INTO stock VALUES (2, 2, 1, 200, '2023-10-01');
INSERT INTO stock VALUES (3, 3, 1, 350, '2023-10-01');
INSERT INTO stock VALUES (4, 4, 1, 400, '2023-10-01');
INSERT INTO stock VALUES (5, 5, 1, 200, '2023-10-01');
INSERT INTO stock VALUES (6, 6, 1, 400, '2023-10-01');
INSERT INTO stock VALUES (7, 7, 1, 500, '2023-10-01');
INSERT INTO stock VALUES (8, 8, 1, 350, '2023-10-01');
INSERT INTO stock VALUES (9, 9, 1, 300, '2023-10-01');
INSERT INTO stock VALUES (10, 10, 1, 300, '2023-10-01');
INSERT INTO stock VALUES (11, 1, 2, 200, '2023-10-07');
INSERT INTO stock VALUES (12, 2, 2, 200, '2023-10-07');
INSERT INTO stock VALUES (13, 3, 2, 350, '2023-10-07');
INSERT INTO stock VALUES (14, 4, 2, 400, '2023-10-07');
INSERT INTO stock VALUES (15, 5, 2, 200, '2023-10-07');
INSERT INTO stock VALUES (16, 6, 2, 400, '2023-10-07');
INSERT INTO stock VALUES (17, 7, 2, 500, '2023-10-07');
INSERT INTO stock VALUES (18, 8, 2, 350, '2023-10-07');
INSERT INTO stock VALUES (19, 9, 2, 300, '2023-10-07');
INSERT INTO stock VALUES (20, 10, 2, 300, '2023-10-07');
```

VIII. SQL SCRIPT FOR DATA INSERT INTO DIMENSIONAL TABLE

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace = '';

SET default_with_oids = false;


--------------------------------------------------------------------------------
-- Name: INSERT INTO product_dim; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO product_dim VALUES (1, 1, 'Piattos', 'Jack ''n Jill Piattos Cheese 40g', 'Chips', '15.00',
'17.00');
INSERT INTO product_dim VALUES (2, 2, 'VCut', 'Jack ''n Jill VCut Onion and Garlic 60g', 'Chips', '36.00',
'39.00');
INSERT INTO product_dim VALUES (3, 3, 'Chippy', 'Jack ''n Jill Chippy BBQ 27g', 'Chips', '6.00', '8.00');
INSERT INTO product_dim VALUES (4, 4, 'Mr Chips', 'Jack ''n Jill Mr Chips Nacho Cheese 26g', 'Chips',
'6.00', '8.00');
INSERT INTO product_dim VALUES (5, 5, 'Clover Chips', 'Clover Chips Barbecue 55g', 'Chips', '21.00',
'25.00');
INSERT INTO product_dim VALUES (6, 6, 'Century Tuna', 'Century Tuna Flakes in Oil 155grams', 'Canned
Goods', '33.00', '38.00');
INSERT INTO product_dim VALUES (7, 7, '555 Sardines', '555 Sardines in Tomato Sauces 155g', 'Canned
Goods', '21.00', '24.00');
INSERT INTO product_dim VALUES (8, 8, 'Argentina Corned Beef', 'Argentina Corned Beef 150g', 'Canned
Goods', '30.50', '35.00');
INSERT INTO product_dim VALUES (9, 9, 'Skyflakes', 'Skyflakes Crackers Handy Pack 200g', 'Biscuit',
'40.00', '45.00');
INSERT INTO product_dim VALUES (10, 10, 'Fita', 'Fita Biscuits Singles 30G 10s', 'Biscuit', '60.00', '65.00');
```

```
--------------------------------------------------------------------------------
-- Name: INSERT INTO store_dim; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO store_dim VALUES (1, 1, 'Chennai Mini Mart 1', 'Rodriguez Ave, Brgy. 34, Bacolod, Negros Occidental');
INSERT INTO store_dim VALUES (2, 2, 'Chennai Mini Mart 2', 'Eroreco Subd., Brgy. Mandalagan, Bacolod, Negros Occidental');

--------------------------------------------------------------------------------
-- Name: INSERT INTO customer_dim; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO customer_dim VALUES (1, 1);
INSERT INTO customer_dim VALUES (2, 2);
INSERT INTO customer_dim VALUES (3, 3);
INSERT INTO customer_dim VALUES (4, 4);
INSERT INTO customer_dim VALUES (5, 5);

--------------------------------------------------------------------------------
-- Name: INSERT INTO date_dim; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO date_dim VALUES (1, 1, '20231001', 'Sunday', 39, 1, 1, 'October', 10, 1, 1, 'Third', 1, 1, '2023');
INSERT INTO date_dim VALUES (2, 2, '20231007', 'Saturday', 40, 2, 2, 'October', 10, 2, 2, 'Third', 2, 2, '2023');
INSERT INTO date_dim VALUES (3, 3, '20231012', 'Thursday', 41, 3, 3, 'October', 10, 3, 3, 'Third', 3, 3, '2023');
```

IX. SQL SCRIPT FOR DATA INSERT INTO FACT TABLE

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
```

```
SET default_tablespace = '';

SET default_with_oids = false;


--------------------------------------------------------------------------------
-- Name: INSERT INTO sale_fact; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO sale_fact VALUES (1, 1, 1, 3, 1, 85.00, 5);
INSERT INTO sale_fact VALUES (2, 1, 6, 3, 1, 380.00, 10);
INSERT INTO sale_fact VALUES (3, 1, 10, 3, 1, 325.00, 5);
INSERT INTO sale_fact VALUES (4, 2, 2, 3, 1, 273.00, 7);
INSERT INTO sale_fact VALUES (5, 2, 3, 3, 1, 120.00, 15);
INSERT INTO sale_fact VALUES (6, 2, 8, 3, 1, 175.00, 5);
INSERT INTO sale_fact VALUES (7, 2, 9, 3, 1, 225.00, 5);
INSERT INTO sale_fact VALUES (8, 2, 10, 3, 1, 325.00, 5);
INSERT INTO sale_fact VALUES (9, 3, 6, 3, 2, 760.00, 20);
INSERT INTO sale_fact VALUES (10, 3, 7, 3, 2, 480.00, 20);
INSERT INTO sale_fact VALUES (11, 4, 4, 3, 2, 80.00, 10);
INSERT INTO sale_fact VALUES (12, 4, 5, 3, 2, 125.00, 5);
INSERT INTO sale_fact VALUES (13, 5, 9, 3, 1, 450.00, 10);
INSERT INTO sale_fact VALUES (14, 5, 10, 3, 1, 650.00, 10);

--------------------------------------------------------------------------------
-- Name: INSERT INTO stock_fact; Type: TABLE; Schema: public; Owner: -; Tablespace:
--------------------------------------------------------------------------------
INSERT INTO stock_fact VALUES (1, 1, 1, 1, 200);
INSERT INTO stock_fact VALUES (2, 2, 1, 1, 200);
INSERT INTO stock_fact VALUES (3, 3, 1, 1, 350);
INSERT INTO stock_fact VALUES (4, 4, 1, 1, 400);
INSERT INTO stock_fact VALUES (5, 5, 1, 1, 200);
INSERT INTO stock_fact VALUES (6, 6, 1, 1, 400);
INSERT INTO stock_fact VALUES (7, 7, 1, 1, 500);
INSERT INTO stock_fact VALUES (8, 8, 1, 1, 350);
INSERT INTO stock_fact VALUES (9, 9, 1, 1, 300);
INSERT INTO stock_fact VALUES (10, 10, 1, 1, 300);
INSERT INTO stock_fact VALUES (11, 1, 2, 2, 200);
INSERT INTO stock_fact VALUES (12, 2, 2, 2, 200);
INSERT INTO stock_fact VALUES (13, 3, 2, 2, 350);
INSERT INTO stock_fact VALUES (14, 4, 2, 2, 400);
INSERT INTO stock_fact VALUES (15, 5, 2, 2, 200);
```

```
INSERT INTO stock_fact VALUES (16, 6, 2, 2, 400);
INSERT INTO stock_fact VALUES (17, 7, 2, 2, 500);
INSERT INTO stock_fact VALUES (18, 8, 2, 2, 350);
INSERT INTO stock_fact VALUES (19, 9, 2, 2, 300);
INSERT INTO stock_fact VALUES (20, 10, 2, 2, 300);
```