**The Zagi Retail Company case.**

**Soudabeh Rafieisakhaei**

Table of Contents

[Relational Model 2](#_Toc168009016)

[ER Diagram 2](#_Toc168009017)

[Relational schema 3](#_Toc168009018)

[Data Dictionary 4](#_Toc168009019)

[Building the database 6](#_Toc168009020)

[Populating the database 9](#_Toc168009021)

[SQL queries 13](#_Toc168009022)

[WHERE 13](#_Toc168009023)

[ORDER BY 14](#_Toc168009024)

[JOIN 14](#_Toc168009025)

[LEFT OUTER JOIN 15](#_Toc168009026)

[HAVING 15](#_Toc168009027)

[Set operations (UNION, INTERSECT, EXCEPT) 16](#_Toc168009028)

[Aggregate functions 18](#_Toc168009029)

[Database updates using INSERT, UPDATE, and DELETE 18](#_Toc168009030)

[ER Diagram: Adding New Requirements to the database 20](#_Toc168009031)

[Relational schema for the updated ER model 21](#_Toc168009032)

[Building a relational database using the updated relational schema 21](#_Toc168009033)

[Populating the new features of the database with data 23](#_Toc168009034)

[SQL queries based on the newly added information. 24](#_Toc168009035)

[Dimensional Model 25](#_Toc168009036)

[Dimensional Model Diagram 25](#_Toc168009037)

[Building the tables according to the dimensional model 26](#_Toc168009038)

[Populating the tables of the dimensional model 28](#_Toc168009039)

[Analytical SQL queries using the dimensional model. 30](#_Toc168009040)

# Relational Model

## ER Diagram

Based on the requirements of the Zagi Retail Company case.

Diagram

Description automatically generated

## Relational schema

According to the ER model created in the first step.

Diagram

Description automatically generated

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Dictionary | | | | | | | |
| **Region Table** | | | | | | | |
| **Field** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| RegionID | Region’s identifying number | Character | 1 numeric character |  | Yes | Yes | PK |
| Region Name | Region’s name | Character | 25 alpha-numeric-characters |  | Yes | No |  |
| **Store Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| StoreID | Store identifying number | VarChar | 3 alpha-numeric characters |  | Yes | Yes | PK |
| StoreZip | Store’s zip code | Character | 5 numeric characters |  | Yes | No |  |
| RegionID | Region’s identifying number | Character | 1 numeric character |  | Yes | Yes | Fk |
| Foreign key RegionID refers to the primary key RegionID in Region table | | | | | | | |
| **Product Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| ProductID | Product’s identifying number | Character | 3 numeric characters |  | Yes | No | PK |
| ProductPrice | Product’s price | Numeric | (7,2) format, number with 7 digits, 2 of which are after the decimal point | (1,0) | Yes | No |  |
| ProductName | Product’s name | VarChar | 25 alpha-numeric characters |  | Yes | No |  |
| VendorID | Vendor’s identifying number | Character | 2 numeric characters |  | Yes | Yes | FK |
| CategoryID | Category’s identifying number | Character | 2 numeric characters |  | Yes | Yes | FK |
| Foreign key VendorID refers to the primary key VendorID in Vendor table  Foreign key CategoryID refers to the primary key CategoryID in Category table | | | | | | | |
| **Includes Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| ProductID | Product’s identifying number | Character | 3 numeric characters |  | Yes | Yes | FK/PK |
| TID | Sales Transactions’ identifying number | VarChar | 8 alpha-numeric characters |  | Yes | Yes | FK/PK |
| Quantity | Number of products the sales transactions include | Integer | 22 digits number of items | 0 | Yes | No |  |
| Foreign key ProductID refers to the primary ProductID in Product table  Foreign key TID refers to the primary key TID in Sales Transactions table  Foreign key ProductID and foreign key TID form a composite primary key in Includes table. | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sales Transaction Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| TID | Sales Transactions’ identifying number | VarChar | 8 alpha-numeric characters |  | Yes | Yes | PK |
| TDate | Transactions’ Date | Date | MM/DD/YYYY format |  | No | No |  |
| CustomerID | Customer’s identifying number | Character | 7 numeric characters |  | Yes | Yes | FK |
| StoreID | Store identifying number | VarChar | 3 alpha-numeric characters |  | Yes | Yes | FK |
| Foreign key CustomerID refers to the primary key CustomerID in Customer table  Foreign key StoreID refers to the primary key StoreID in Store table | | | | | | | |
| **Vendor Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| VendorID | Vendor’s identifying number | Character | 2 numeric characters |  | Yes | Yes | PK |
| VendorName | Vendor’s name | VarChar | 25 alpha-numeric characters |  |  |  |  |
| **Category Table** | | | | | | | |
|  |  |  |  |  |  |  |  |
| CategoryID | Category’s identifying number | Character | 2 numeric characters |  | Yes | Yes | PK |
| CategoryName | Category’s Name | VarChar | 25 alpha-numeric characters |  |  |  |  |
| **Customer Table** | | | | | | | |
| **Field Name** | **Description** | **Type** | **Specifications** | **Default** | **Required** | **Unique** | **Key(s)** |
| CustomerID | Customer’s identifying number | Character | 7 numeric characters |  | Yes | Yes | PK |
| CustomerZip | Customer’s zip code | Character | 5 numeric characters |  | Yes | No |  |
| CustomerName | Customer’s name | VarChar | 15 alpha-numeric characters |  | Yes | No |  |



## Building the database

Accrding to the relational schema created in step 2.

CREATE DATABASE zagi;

use zagi;

CREATE TABLE region (

regionid char(1) NOT NULL,

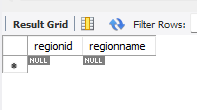
regionname varchar(25) NOT NULL,

PRIMARY KEY (regionid)

);

SELECT \* FROM region;





CREATE TABLE store (

storeid varchar(3) NOT NULL,

storezip char(5) NOT NULL,

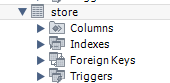
regionid char(1) NOT NULL,

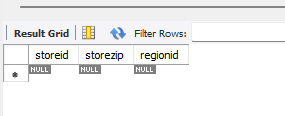
PRIMARY KEY (storeid),

FOREIGN KEY (regionid) REFERENCES region (regionid)

);

SELECT \* FROM table;





CREATE TABLE vendor (

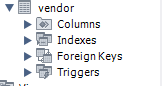
vendorid char(2) NOT NULL,

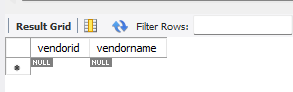
vendorname varchar(25) NOT NULL,

PRIMARY KEY (vendorid)

);

SELECT \* FROM vendor;





CREATE TABLE category (

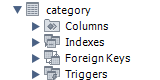
categoryid char(2) NOT NULL,

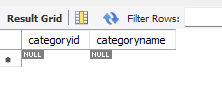
categoryname varchar(25) NOT NULL,

PRIMARY KEY (categoryid)

);

SELECT \* FROM category;





CREATE TABLE customer (

customerid char(7) NOT NULL,

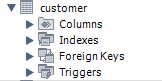
customername varchar(15) NOT NULL,

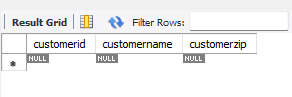
customerzip char(5) NOT NULL,

PRIMARY KEY (customerid)

);

SELECT \* FROM customer;





CREATE TABLE product (

productid char(3) NOT NULL,

productname varchar(25) NOT NULL,

productprice decimal(7,2) NOT NULL,

vendorid char(2) NOT NULL,

categoryid char(2) NOT NULL,

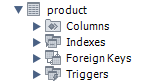
PRIMARY KEY (productid),

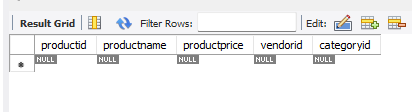
FOREIGN KEY (vendorid) REFERENCES vendor (vendorid),

FOREIGN KEY (categoryid) REFERENCES category (categoryid)

);

SELECT \* FROM product;





CREATE TABLE salestransaction (

tid varchar(8) NOT NULL,

customerid char(7) NOT NULL,

storeid varchar(3) NOT NULL,

tdate date NOT NULL,

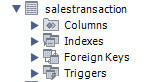
PRIMARY KEY (tid),

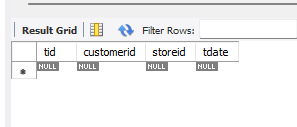
FOREIGN KEY (customerid) REFERENCES customer (customerid),

FOREIGN KEY (storeid) REFERENCES store (storeid)

);

SELECT \* FROM salestransaction;





CREATE TABLE includes (

productid char(3) NOT NULL,

tid varchar(8) NOT NULL,

quantity int NOT NULL,

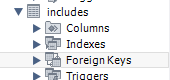
PRIMARY KEY (productid,tid),

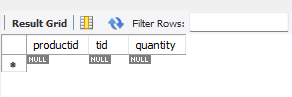
FOREIGN KEY (productid) REFERENCES product (productid),

FOREIGN KEY (tid) REFERENCES salestransaction (tid)

);

SELECT \* FROM includes;





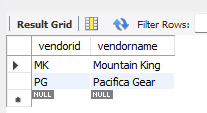
## Populating the database

INSERT INTO Vendor VALUES ('MK','Mountain King');

INSERT INTO Vendor VALUES ('PG','Pacifica Gear');

SELECT \* FROM vendor ORDER BY vendorid desc;

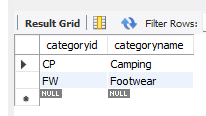
SELECT \* FROM vendor;



INSERT INTO category VALUES ('CP','Camping');

INSERT INTO category VALUES ('FW','Footwear');

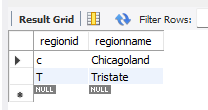
SELECT \* FROM category;



INSERT INTO region VALUES ('c','Chicagoland');

INSERT INTO region VALUES ('T','Tristate');

SELECT \* FROM region;

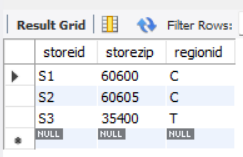


INSERT INTO store VALUES ('S1','60600','C');

INSERT INTO store VALUES ('S2', '60605','C');

INSERT INTO store VALUES ('S3','35400','T');

SELECT \* FROM store;

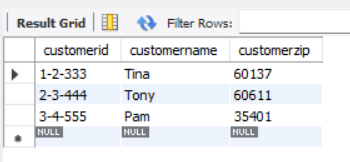


INSERT INTO customer VALUES ('1-2-333','Tina','60137');

INSERT INTO customer VALUES ('2-3-444','Tony','60611');

INSERT INTO customer VALUES ('3-4-555','Pam','35401');

SELECT \* FROM customer;



INSERT INTO salestransaction VALUES ('T111','1-2-333','S1','2020-01-01');

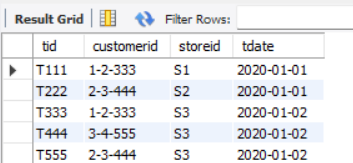
INSERT INTO salestransaction VALUES ('T222','2-3-444','S2','2020-01-01');

INSERT INTO salestransaction VALUES ('T333','1-2-333','S3','2020-01-02');

INSERT INTO salestransaction VALUES ('T444','3-4-555','S3','2020-01-02');

INSERT INTO salestransaction VALUES ('T555','2-3-444','S3','2020-01-02');

SELECT \* FROM salestransaction;



INSERT INTO product VALUES ('1X1','Zzz Bag',100,'PG','CP');

INSERT INTO product VALUES ('2X2','Easy Boot',70,'MK','FW');

INSERT INTO product VALUES ('3X3','Cosy Sock',15,'MK','FW');

INSERT INTO product VALUES ('4X4','Dura Boot',90,'PG','FW');

INSERT INTO product VALUES ('5X5','Tiny Tent',150,'MK','CP');

INSERT INTO product VALUES ('6X6','Biggy Tent',250,'MK','CP');

SELECT \* FROM product;



INSERT INTO includes VALUES ('1X1','T111',1);

INSERT INTO includes VALUES ('2X2','T222',1);

INSERT INTO includes VALUES ('3X3','T333',5);

INSERT INTO includes VALUES ('1X1','T333',1);

INSERT INTO includes VALUES ('4X4','T444',1);

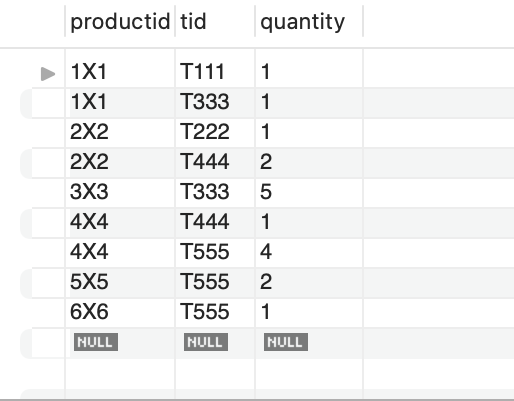
INSERT INTO includes VALUES ('2X2','T444',2);

INSERT INTO includes VALUES ('4X4','T555',4);

INSERT INTO includes VALUES ('5X5','T555',2);

INSERT INTO includes VALUES ('6X6','T555',1);

SELECT \* FROM includes;



## SQL queries

### WHERE

SELECT productid, productname,productprice,categoryid

FROM product

WHERE productprice < 100;

Table

Description automatically generated

### ORDER BY

SELECT productid, productname,productprice

From product

ORDER BY productprice;

Table

Description automatically generated

### JOIN

SELECT productid,productname,productprice,vendorname

FROM product

JOIN vendor

ON product.vendorid = vendor.vendorid

ORDER BY productid;

Table

Description automatically generated

### LEFT OUTER JOIN

SELECT customername,customerzip,storeid,tid,tdate

FROM customer c LEFT OUTER JOIN salestransaction s

ON c.customerid = s.customerid

GROUP BY customername,customerzip,storeid,tid,tdate;

Table

Description automatically generated

### HAVING

SELECT productname,productprice,tid,quantity

FROM includes i LEFT OUTER JOIN product p

ON i.productid = p.productid

GROUP BY productname,productprice,tid,quantity

HAVING quantity > 2;

Table

Description automatically generated

### Set operations (UNION, INTERSECT, EXCEPT)

CREATE VIEW customer\_in\_S3 AS

SELECT customername,customerzip,storeid,tdate

FROM customer c LEFT OUTER JOIN salestransaction s

ON c.customerid = s.customerid WHERE storeid = 's3'

GROUP BY customername,customerzip,storeid,tdate;

CREATE VIEW customer\_in\_S1 AS

SELECT customername,customerzip,storeid,tdate

FROM customer c LEFT OUTER JOIN salestransaction s

ON c.customerid = s.customerid WHERE storeid = 's1'

GROUP BY customername,customerzip,storeid,tdate;

SELECT \* FROM customer\_in\_S3

UNION

SELECT \* FROM customer\_in\_S1;

Table

Description automatically generated

SELECT \* FROM customer\_in\_S3

INTERSECT

SELECT \* FROM customer\_in\_S1;

MySQL does not support the INTERSECT operator; thus, we don’t have an output here .INTERSECT returns only common rows returned by the two SELECT statements.

SELECT \* FROM customer\_in\_S3

MINUS

SELECT \* FROM customer\_in\_S1;

MySQL does not support the EXCEPT or MINUS operator. It is supposed to return values that are in the first Select statement that are not in common with the second one.

* 1. Subqueries

SELECT productid, productname, productprice,categoryid

FROM product

WHERE productid IN

(SELECT productid

FROM includes

GROUP BY productid

HAVING SUM(quantity)=2);

Table

Description automatically generated

### Aggregate functions

SELECT productid, productname, productprice,vendorid,categoryid

FROM product

WHERE productprice > (SELECT AVG(productprice) FROM product);

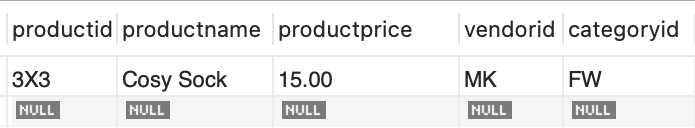
Table

Description automatically generated

SELECT productid, productname, productprice,vendorid,categoryid

FROM product

WHERE productprice = (SELECT min(productprice) FROM product);



### Database updates using INSERT, UPDATE, and DELETE

INSERT INTO product VALUES ('7X7','Water bottle',25,'PG','CP');

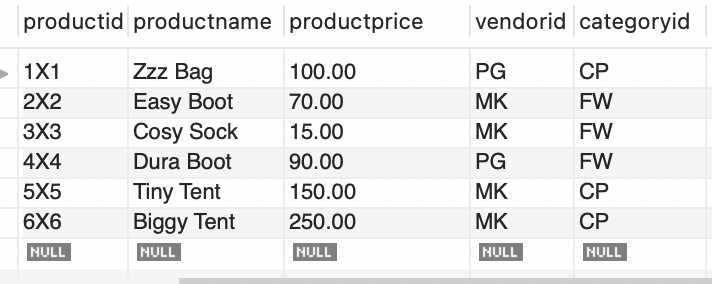
SELECT \* FROM product;

Table

Description automatically generated

DELETE FROM product WHERE productid = '7X7';

SELECT \* FROM product;

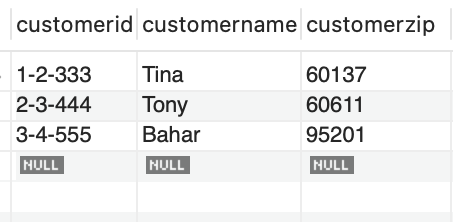


UPDATE customer

SET customername = 'Bahar', customerzip = '95201'

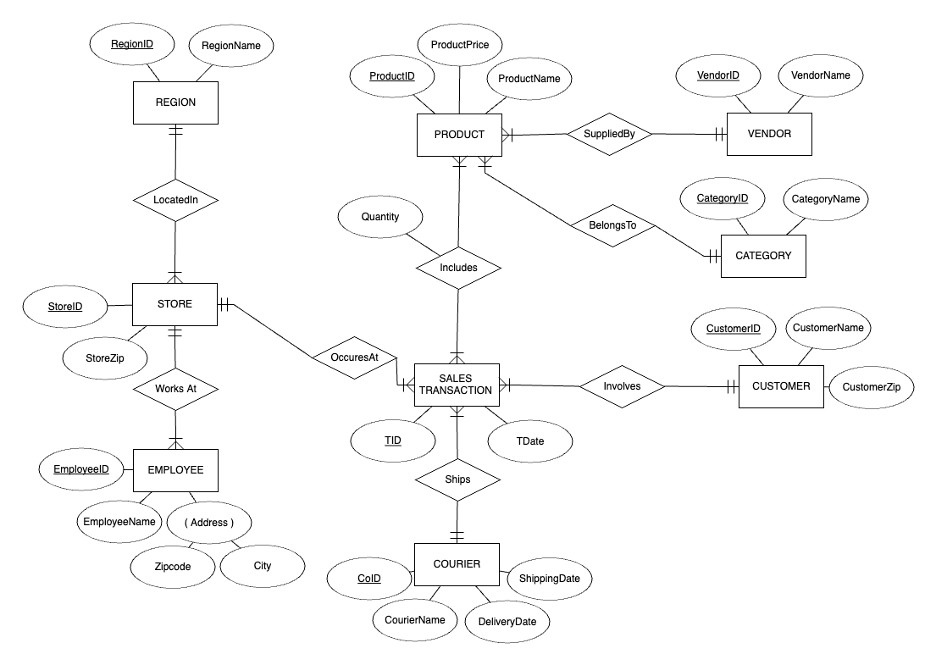
WHERE customerid = '3-4-555';

Select \* From customer;

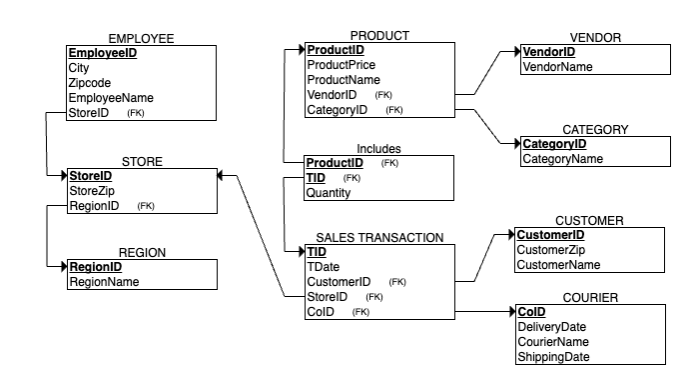


## ER Diagram: Adding New Requirements to the database

Add *your* requirements for the Zagi Retail Company case and the ER diagram for a scenario with two entities (both with at least three different types of attributes) involved in 1:M relationship.



## Relational schema for the updated ER model



## Building a relational database using the updated relational schema

All the tables are created the same as question 4, except sales transaction that requires a new foreign key to reference courier table. The code is included in this section and the it is populated in next question along with employee and courier tables.

CREATE TABLE employee(

employeeID char(7)NOT NULL,

employeeName varchar(15) NOT NULL,

city varchar(15) NOT NULL,

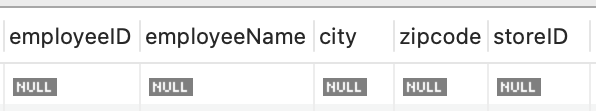
zipcode char(5) NOT NULL,

storeID varchar(3)NOT NULL,

PRIMARY KEY (employeeID),

FOREIGN KEY (storeID) REFERENCES STORE(storeID)

);



CREATE TABLE courier(

coID char(3) NOT NULL,

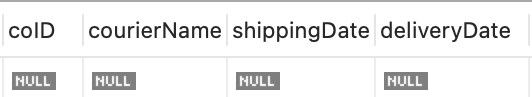
courierName varchar(15) NOT NULL,

shippingDate date NOT NULL,

deliveryDate date NOT NULL,

PRIMARY KEY (CoID)

);



CREATE TABLE salestransaction (

tid varchar(8) NOT NULL,

customerid char(7) NOT NULL,

storeid varchar(3) NOT NULL,

tdate date NOT NULL,

coID char(3) NOT NULL,

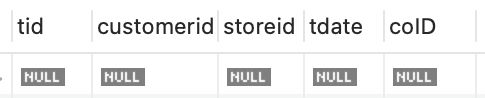
PRIMARY KEY (tid),

FOREIGN KEY (customerid) REFERENCES customer (customerid),

FOREIGN KEY (storeid) REFERENCES store (storeid),

FOREIGN KEY (coID) REFERENCES courier (coID)

);



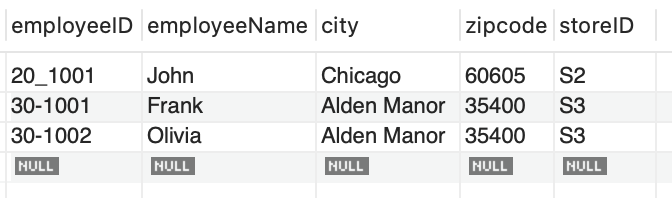
## Populating the new features of the database with data

INSERT INTO employee VALUES ('20\_1001','John','Chicago','60605','S2');

INSERT INTO employee VALUES ('30-1001','Frank','Alden Manor','35400','S3');

INSERT INTO employee VALUES ('30-1002','Olivia','Alden Manor','35400','S3');

SELECT \* FROM employee;

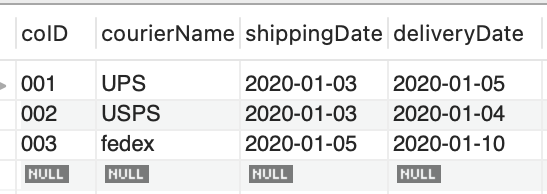


INSERT INTO courier VALUES ('001','UPS','2020-01-3','2020-01-5');

INSERT INTO courier VALUES ('002','USPS','2020-01-3','2020-01-4');

INSERT INTO courier VALUES ('003','fedex ','2020-01-5','2020-01-10');

SELECT \* FROM courier;



INSERT INTO salestransaction VALUES ('T111','1-2-333','S1','2020-01-01','001');

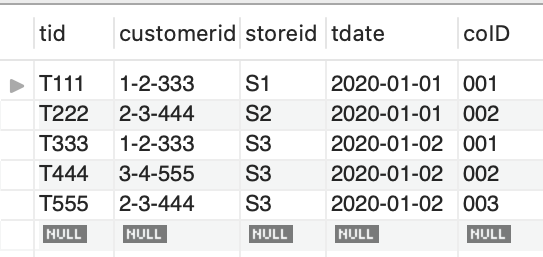
INSERT INTO salestransaction VALUES ('T222','2-3-444','S2','2020-01-01','002');

INSERT INTO salestransaction VALUES ('T333','1-2-333','S3','2020-01-02','001');

INSERT INTO salestransaction VALUES ('T444','3-4-555','S3','2020-01-02','002');

INSERT INTO salestransaction VALUES ('T555','2-3-444','S3','2020-01-02','003');

SELECT \* FROM salestransaction;



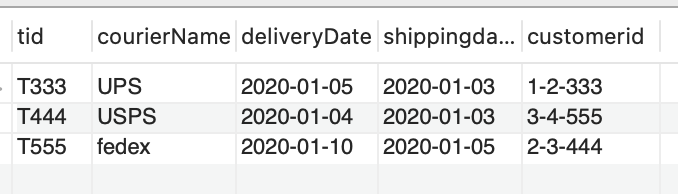
## SQL queries based on the newly added information.

SELECT tid,courierName,deliveryDate,shippingdate,customerid

FROM courier c RIGHT OUTER JOIN salestransaction s

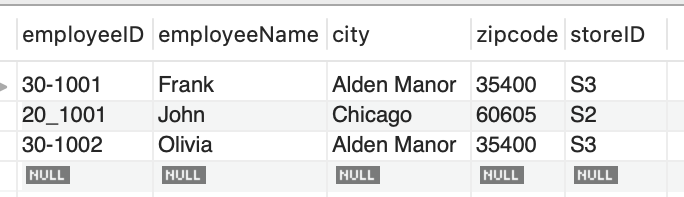
ON c.coid = s.coid WHERE storeid = 'S3'

GROUP BY tid,courierName,deliveryDate,shippingDate,customerid;



SELECT \* FROM employee

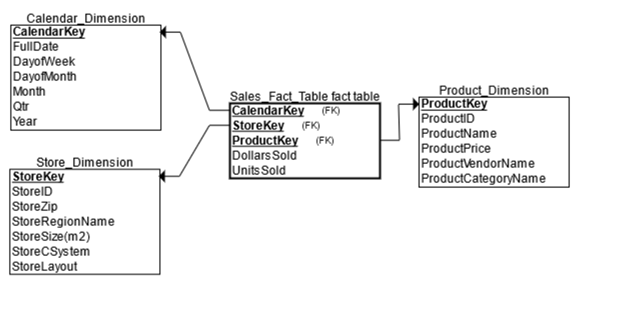
ORDER BY employeeName;



# Dimensional Model

## Dimensional Model Diagram

A dimensional model containing an aggregated fact table, where a fact table shows a summary of units sold and dollars sold for daily purchases of each product in each store.



## Building the tables according to the dimensional model

A dimensional model with an *aggregated* sales fact table: Sales\_Fact\_Table (Units Sold, Dollars Sold for daily purchase of each product in each store.

**SQL**

CREATE TABLE CALENDAR\_Dimension

(

CalendarKey INT NOT NULL,

FullDate DATE NOT NULL,

DayOfWeek CHAR(15) NOT NULL,

DayOfMonth INT NOT NULL,

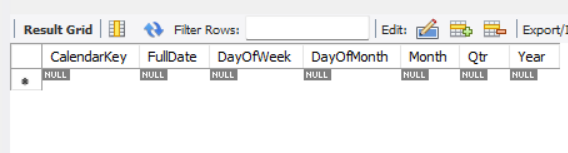
Month CHAR(10) NOT NULL,

Qtr CHAR(21) NOT NULL,

Year INT NOT NULL,

PRIMARY KEY (CalendarKey)

);



CREATE TABLE STORE\_Dimension

(

StoreKey INT NOT NULL,

StoreID CHAR(15) NOT NULL,

StoreZip CHAR(15) NOT NULL,

StoreRegionName CHAR(15) NOT NULL,

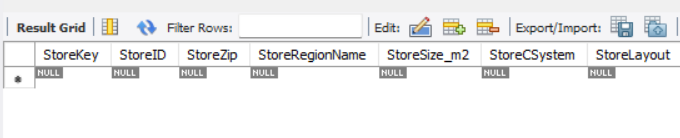
StoreSize\_m2 INT NOT NULL,

StoreCSystem CHAR(15) NOT NULL,

StoreLayout CHAR(15) NOT NULL,

PRIMARY KEY (StoreKey)

);



# For StoreSize\_m2 my sql did not support(m2)

CREATE TABLE PRODUCT\_Dimension

(

ProductKey INT NOT NULL,

ProductID CHAR(15) NOT NULL,

ProductName CHAR(25) NOT NULL,

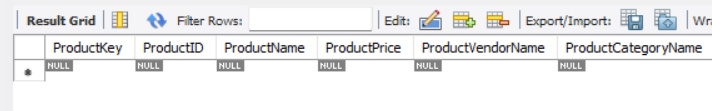
ProductPrice DECIMAL(7,2) NOT NULL,

ProductVendorName CHAR(20) NOT NULL,

ProductCategoryName CHAR(25) NOT NULL,

PRIMARY KEY (ProductKey)

);



CREATE TABLE Sales\_Fact\_Table

(

DollarsSold DECIMAL(7,2) NOT NULL,

UnitsSold INT NOT NULL,

CalendarKey INT NOT NULL,

StoreKey INT NOT NULL,

ProductKey INT NOT NULL,

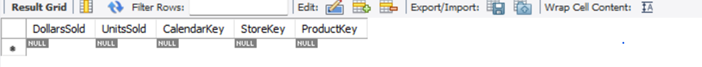
PRIMARY KEY (CalendarKey, StoreKey, ProductKey),

FOREIGN KEY (CalendarKey) REFERENCES Calendar\_Dimension(CalendarKey),

FOREIGN KEY (StoreKey) REFERENCES Store\_Dimension(StoreKey),

FOREIGN KEY (ProductKey) REFERENCES Product\_Dimension(ProductKey)

);



## Populating the tables of the dimensional model

CALENDAR Dimension

INSERT INTO CALENDAR\_Dimension VALUES(1,'2020-01-01','wednesday',1,'January','Q1',2020);

INSERT INTO CALENDAR\_Dimension VALUES(2,'2020-01-02','Thursday',2,'January','Q1',2020);

INSERT INTO CALENDAR\_Dimension VALUES(3, '2020-01-03','friday',3,'January','Q1',2020);

Table

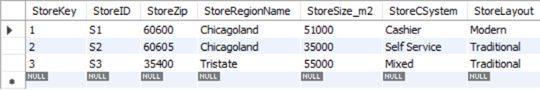
Description automatically generated

STORE Dimension

INSERT INTO STORE\_Dimension VALUES(1,'S1',60600,'Chicagoland',51000,'Cashier','Modern');

INSERT INTO STORE\_Dimension VALUES(2,'S2',60605,'Chicagoland',35000,'Self Service','Traditional');

INSERT INTO STORE\_Dimension VALUES(3,'S3',35400,'Tristate',55000,'Mixed','Traditional');



PRODUCT Dimension

INSERT INTO PRODUCT\_Dimension VALUES(1,'1X1','Zzz','100','Pacifica Gear','Camping');

INSERT INTO PRODUCT\_Dimension VALUES(2,'2X2','Easy Boot','70','Mountain King','Footwear');

INSERT INTO PRODUCT\_Dimension VALUES(3,'3X3','Cozy Sock','15','Mountain King','Footwear');

INSERT INTO PRODUCT\_Dimension VALUES(4,'4X4','Dura Boot','90','Pacifica Gear','Footwear');

INSERT INTO PRODUCT\_Dimension VALUES(5,'5X5','Tiny Tent','150','Mountain King','Camping');

INSERT INTO PRODUCT\_Dimension VALUES(6,'6X6','Biggy Tent','250','Mountain King','Camping');

Table

Description automatically generated

SALES FACT TABLE

INSERT INTO SALES\_Fact\_Table VALUES('100',1,1,3,1);

INSERT INTO SALES\_Fact\_Table VALUES('300',2,1,3,5);

INSERT INTO SALES\_Fact\_Table VALUES('250',1,1,3,6);

INSERT INTO SALES\_Fact\_Table VALUES('70',1,2,1,2);

INSERT INTO SALES\_Fact\_Table VALUES('45',3,2,2,3);

INSERT INTO SALES\_Fact\_Table VALUES('180',2,2,2,4);

INSERT INTO SALES\_Fact\_Table VALUES('120',8,3,2,3);

INSERT INTO SALES\_Fact\_Table VALUES('90',1,3,2,4);

INSERT INTO SALES\_Fact\_Table VALUES('60',4,3,3,3);

SELECT \* FROM SALES\_Fact\_Table;

Table

Description automatically generated

## Analytical SQL queries using the dimensional model.

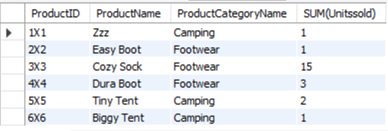
1)

SELECT ProductID, ProductName, ProductCategoryName, SUM(Unitssold)

FROM PRODUCT\_Dimension P, SALES\_Fact\_Table SA

WHERE P.ProductKey = SA.ProductKey

GROUP BY ProductID, ProductName, ProductCategoryName;



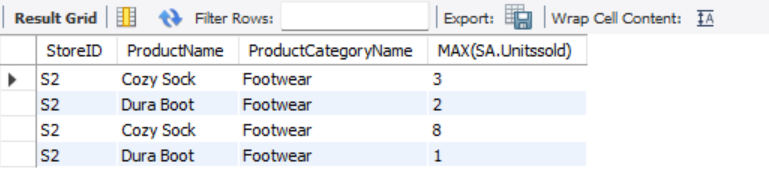
2)

SELECT S.StoreID, P.ProductName, P.ProductCategoryName, MAX(SA.Unitssold)

From STORE\_Dimension S,PRODUCT\_Dimension P, SALES\_Fact\_Table SA

WHERE S.StoreKey = SA.StoreKey AND P.ProductKey = SA.ProductKey AND S.STOREID='S2'

GROUP BY S.StoreID, P.ProductName, P.ProductCategoryName, SA.Unitssold;



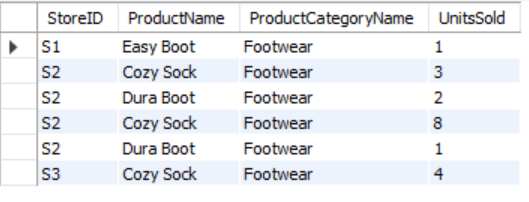
3)

SELECT S.StoreID, P.ProductName, P.ProductCategoryName, SA.UnitsSold

From STORE\_Dimension S,PRODUCT\_Dimension P, SALES\_Fact\_Table SA

WHERE S.StoreKey = SA.StoreKey AND P.ProductKey = SA.ProductKey AND P.ProductCategoryName='Footwear'

ORDER BY StoreID;



4)

SELECT P.ProductCategoryName, C.DayofWeek,C.FullDate, SA.Dollarssold

FROM PRODUCT\_Dimension P, CALENDAR\_Dimension C,SALES\_Fact\_Table SA

WHERE C.CalendarKey = SA.CalendarKey AND P.ProductKey = SA.ProductKey

GROUP BY P.ProductCategoryName, C.DayofWeek,C.FullDate, SA.Dollarssold;

Table

Description automatically generated

5)

SELECT S.StoreID,C.DayofWeek,P.ProductName, P.ProductCategoryName,sum(Unitssold)FROM PRODUCT\_Dimension P, CALENDAR\_Dimension C,STORE\_Dimension S, SALES\_Fact\_Table SA

WHERE P.ProductKey = SA.ProductKey AND C.CalendarKey = SA.CalendarKey

AND S.StoreKey = SA.StoreKey AND S.StoreID = 'S3' AND C.DayofWeek= 'Wednesday'

GROUP BY S.StoreID,C.DayofWeek,P.ProductName, P.ProductCategoryName;

