`

MET CS 779 TERM PROJECT

Table of Contents

[1. Introduction 1](#_heading=h.1fob9te)

[2. Data Set Information 1](#_heading=h.3znysh7)

[1) Topic 1](#_heading=h.2et92p0)

[2) Source 1](#_heading=h.tyjcwt)

[3) Description of Dataset 1](#_heading=h.bo8vlssw1mix)

[3. Goals 2](#_heading=h.3dy6vkm)

[4. Technicals 2](#_heading=h.rg0pcdksqbqy)

[1) Tools 2](#_heading=h.1t3h5sf)

[2) Languages 2](#_heading=h.iyp8sus110ao)

[3) Advanced Database Area 2](#_heading=h.4d34og8)

[5. Data Pre-Processing 2](#_heading=h.f6rbdtmj6m43)

[6. Staging Tables 3](#_heading=h.umxvp4dqicax)

[7. Constellation Dimensional Design 4](#_heading=h.5dnoq4ciwm6o)

[8. SCD Explanation 5](#_heading=h.r2z03d8h0myw)

[9. Fact Table Explanation and Application 6](#_heading=h.v0n0c446u84c)

[10. Data Management and Maintenance 13](#_heading=h.j0btnng5pe6q)

[11. Conclusion 16](#_heading=h.pldz6vfmz11u)

[Appendices 18](#_heading=h.3rdcrjn)

[References 18](#_heading=h.rdb6jvpnjy4n)

# Introduction

This is a project to use Advanced Data Management technology to do the datasets from the real world in Kaggle.com.

# Data Set Information

## Topic

Supermarket Ordering, Invoicing and Sales

## Source

Supermarket Ordering and Sales: <https://www.kaggle.com/datasets/thedevastator/supermarket-ordering-invoicing-and-sales-analysi?select=Invoices.csv>

Location Source (Fast Food Restaurant Across America) : <https://www.kaggle.com/datasets/datafiniti/fast-food-restaurants>

## Description of Dataset

This data set provides an in-depth look into the ordering, invoicing and sales processes at a supermarket. With information ranging from customers' meal choices to the value of their orders and whether they were converted into sales, this dataset opens up endless possibilities to uncover consumer behavior trends and engagement within the business. From understanding who is exchanging with the company and when, to seeing what types of meals are most popular with consumers, this rich collection of data will allow us to gain priceless insights into consumer actions and habits that can inform strategic decisions. Dive deep into big data now by exploring Invoices.csv, OrderLeads.csv and SalesTeam.csv for invaluable knowledge about your customers!

There are 3 files/tables in the dataset: Invoices, orderLeads, salesTeam. Total 18 columns.

The OrderLeads.csv file contains data about orders placed at a supermarket, including company name, date, order value, and whether or not the order was converted into a sale. We can consider it as a pre-order file.

* Date: The date the order was placed. (Date)
* Company Name: The name of the company associated with the order. (String)
* Order Value: The total value of the order. (Float)
* Converted: Whether or not the order was converted into a sale. (Boolean)

The Invoices.csv file contains data about the invoices generated for orders placed at the supermarket, including served meal prices converted from orders to actual sales.

* Date: The date the order was placed. (Date)
* Date of Meal: The date the meal was served. (Date)
* Participants: The number of people who participated in the meal. (Integer)
* Meal Price: The cost of the meal. (Float)
* Type of Meal: The type of meal that was ordered. (String)

The SalesTeam.csv file contains information about the sales representatives of supermarkets, their IDs, the companies they are associated with, and the company IDs.

Although this topic is marked as a supermarket, it is more like a food or restaurants department in the supermarkets since it only focuses on the meal. Companies are customers of supermarkets, making pre-orders from sales representatives, and will convert the part or full pre-order to served meal, and generate an invoice for it. The dataset provides three datasets, but the location information is missing, so I get the location information from another dataset: Fast Food Restaurant Across America dataset and use it as the company's address of the supermarket dataset.

# Goals

I will use ETL: extract, transform, load to process all the dataset to provide all information to the project. Do data clearing, data integration, ERD design and use the knowledge of constellation dimensional design learned in CS779 to to create a data warehouse. By analyzing the data in the data warehouse, we can understand:

1. Which type of meal is top sold?
2. Who are the top 10 sales representatives that made the most sales in 2017?
3. How many of total pre-orders converted to sale in December of 2018?
4. Which state has the highest values of sell?

# Technicals

## Tools

Draw.io, DBeaver, Visual Studio Code/Jupyter notebook, Tableau

## Languages

SQL, Python

## Advanced Database Area

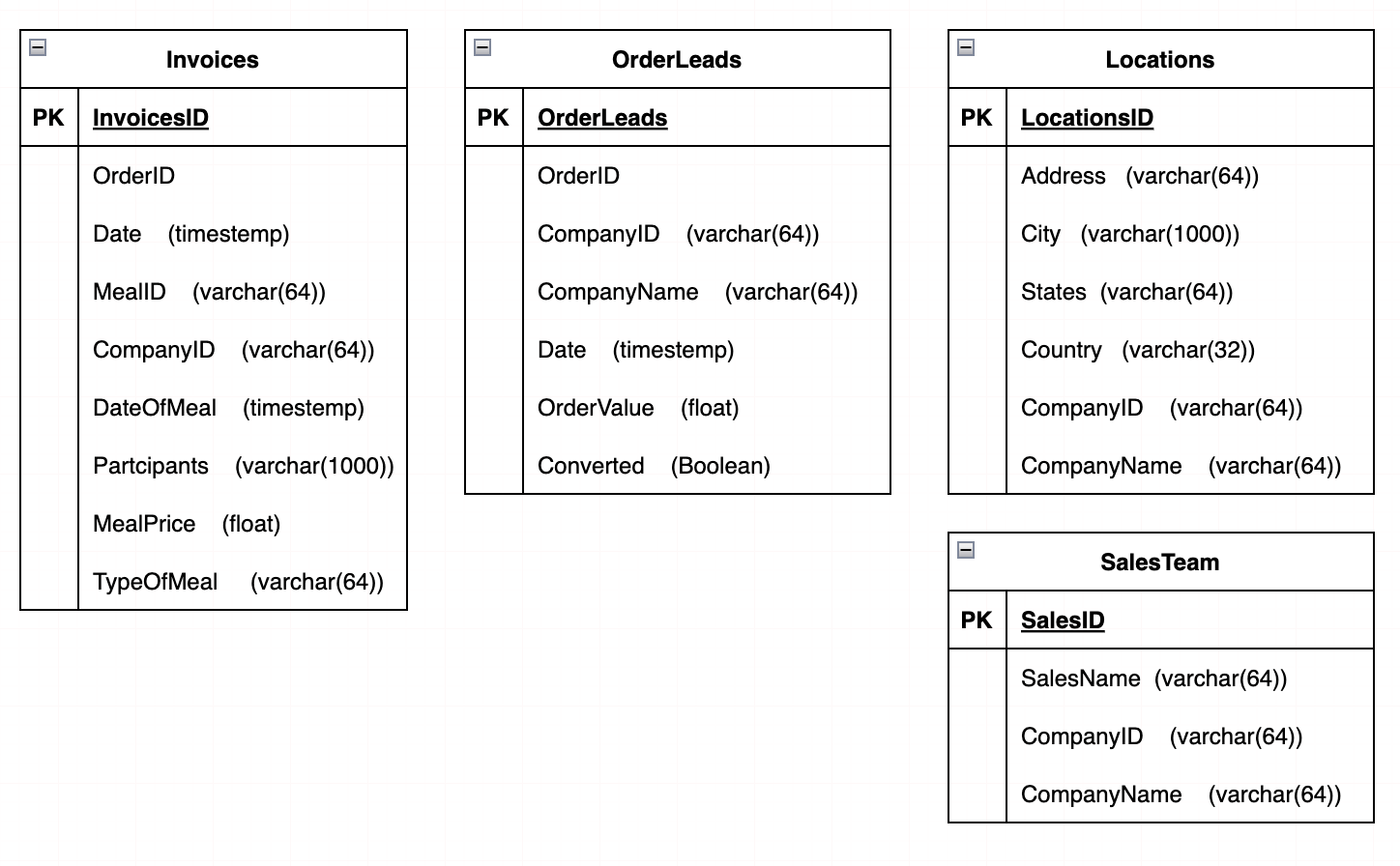
ERD modeling design, Constellation dimensional design and creation, structure of data warehouse or advanced database area’s techniques learned and applied later.

# Data Pre-Processing

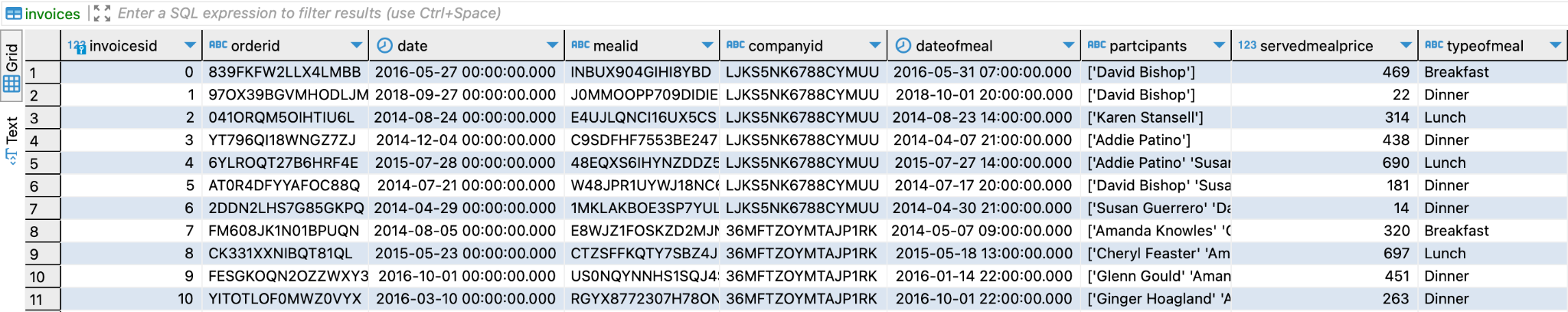
1. Extract data files from two websites to local.
2. Use python to do data cleaning and correct the data type of date from ‘dd-mm-yyyy’ to ‘yyyy-mm-dd’ and change the data type from object to timestamp in OrderLeads and Invoices tables.
3. Extract address, city, states and country column from Fast Food Restaurant Across America, set them to companies ID and names of Supermarket dataset as the location information dataset.
4. Create and transfer all the data tables to the new database/server: superMarket.

The data in the database is ready to be used by SQL.

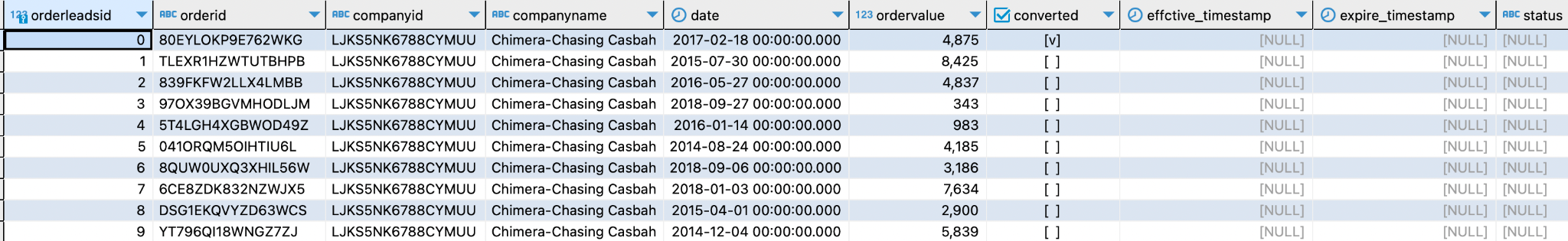
# Staging Tables



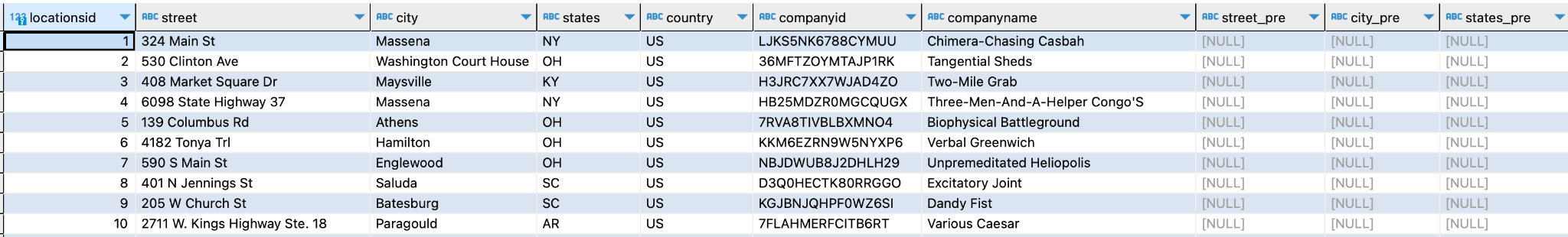
The screenshots of staging tables: Invoices table



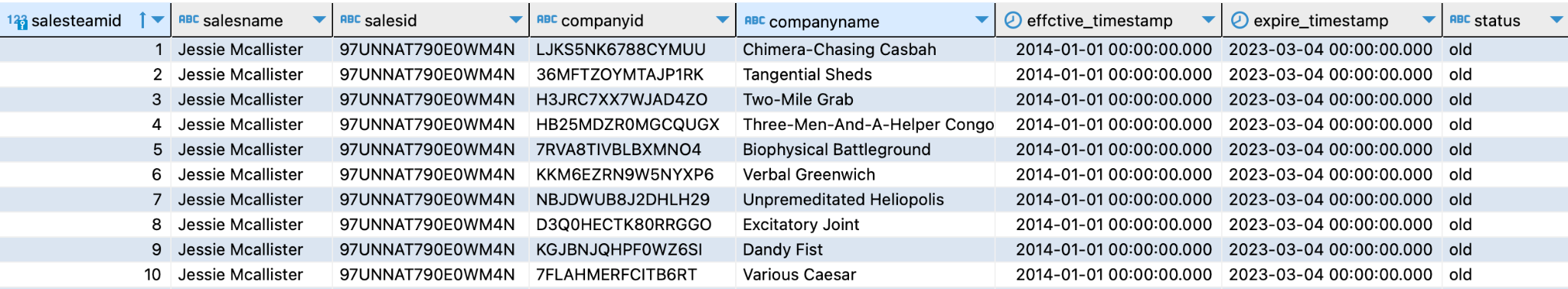
OrderLeads table:



Locations table:



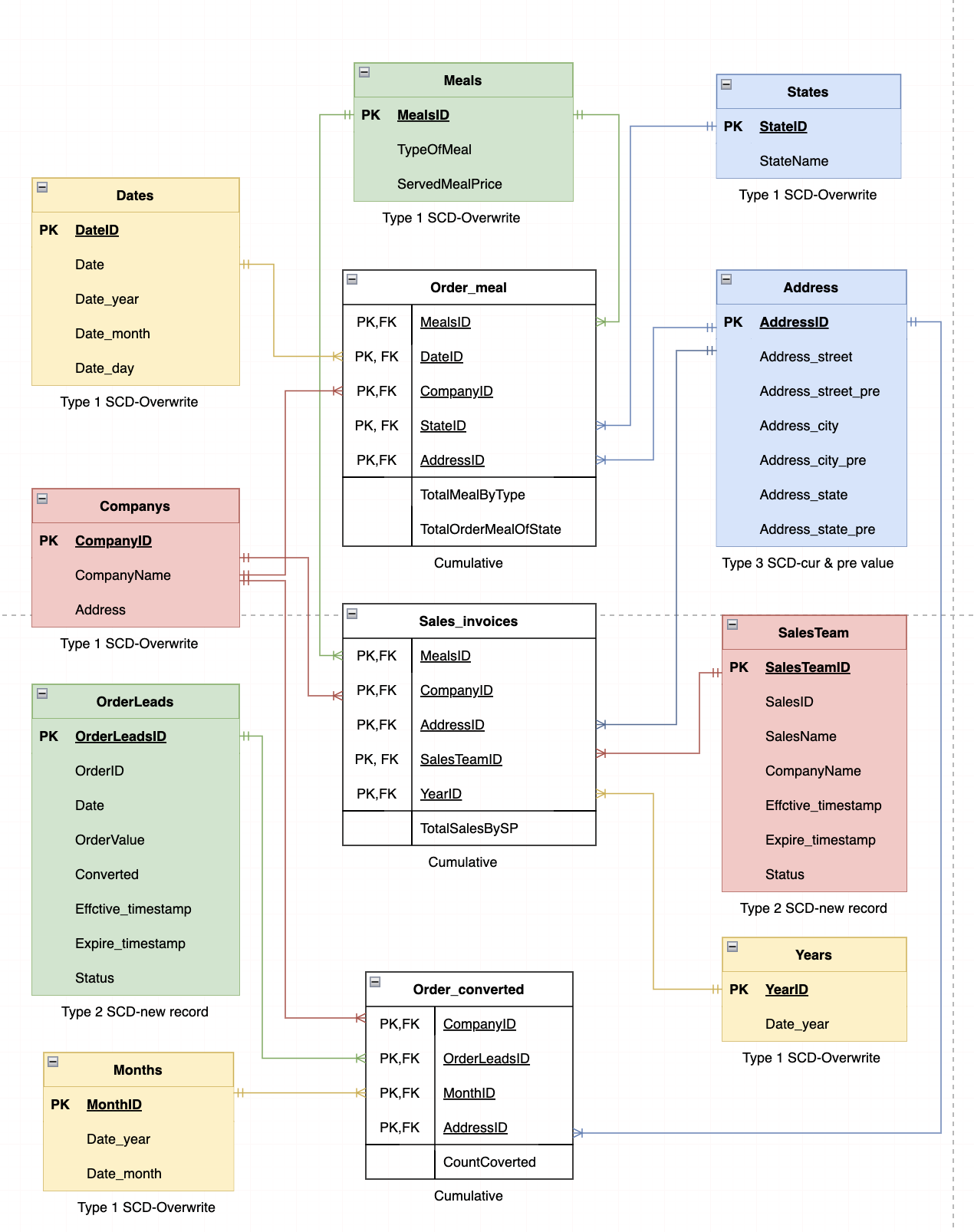
SalesTeam table:



# Constellation Dimensional Design

The constellation dimension model contains dimension tables and fact tables. The dimension table is used to supplement the fact table to describe and restore the scene when the fact occurs. Fact tables contain dimensions and measures, and all data can be generated through calculations based on such fact tables.

The ERD of design:



# SCD Explanation

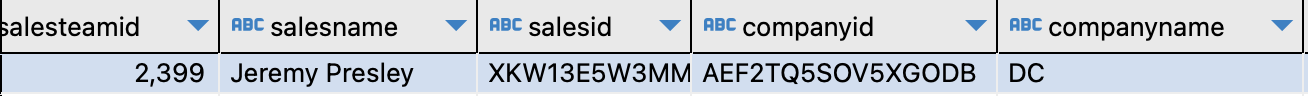
There are 3 types of slow changing dimensions in my constellation dimension design:

1. Type 1 SCDs\_overwriting: tables are Meals, Date, Companies, Month, Year and States. Total seven table’s data can be overwritten by new data. For example: the company name changed to a new company name, the old name will be overwritten by new like a updated the record:

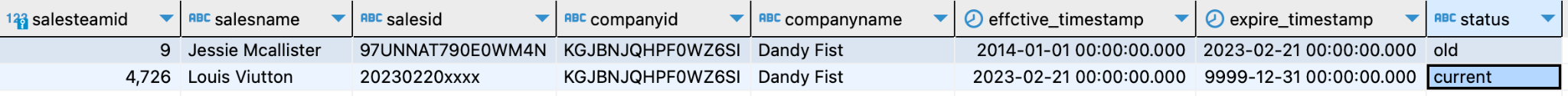
old company name:



new company name:

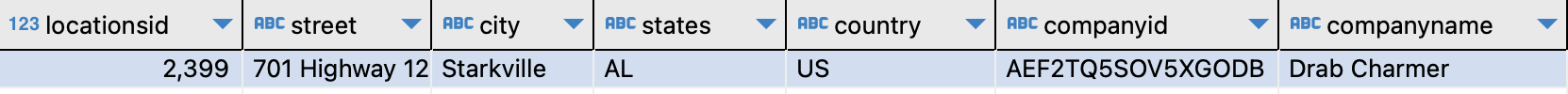


1. Type 2 SCDs\_new record: tables are OrderLeads and SalesTeam. Each record stores the whole history of the selected attribute. For example: We assigned a new salesperson: louis Viutton to run a company ‘Dandy Fist’ that was previously run by another salesperson Jessie Mcallister, so we have to add a new row and add both effective date and expire date on the two sales names of this company and mark the status of previous record as previous and current record as current. The records of history:

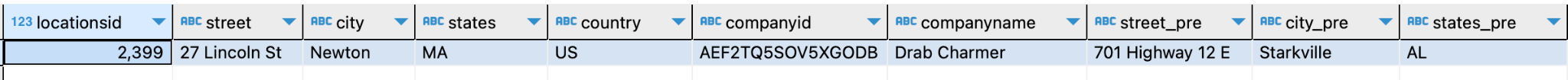


1. Type 3 SCDs\_current and previous value: tables are Address. The address is the address of the companies, usually a type 1 SCDs, but since it relates to sales in the city and state in which it is located, we must document its history of city and state change. When the value of any of the selected attributes changes, the current value is stored as the previous(old) value and the value becomes the current value. For example:

The old address of company ‘Drab Charmer’ is:



After company moved to MA, the record becomes:



Type 2 and 3 SCDs need to be maintained to keep the data consistent by two stored procedures.

# Fact Table Explanation and Application

There is one type of fact table in my design: cumulative. Cumulative table describes what has happened over a period of time. This type’s fact tables are mostly additive facts: tables are Order\_meal and Order\_converted and Sales\_invoices.

I created dimension tables and fact tables. Some example of them shows below:

Dimension tables: address:

Code:

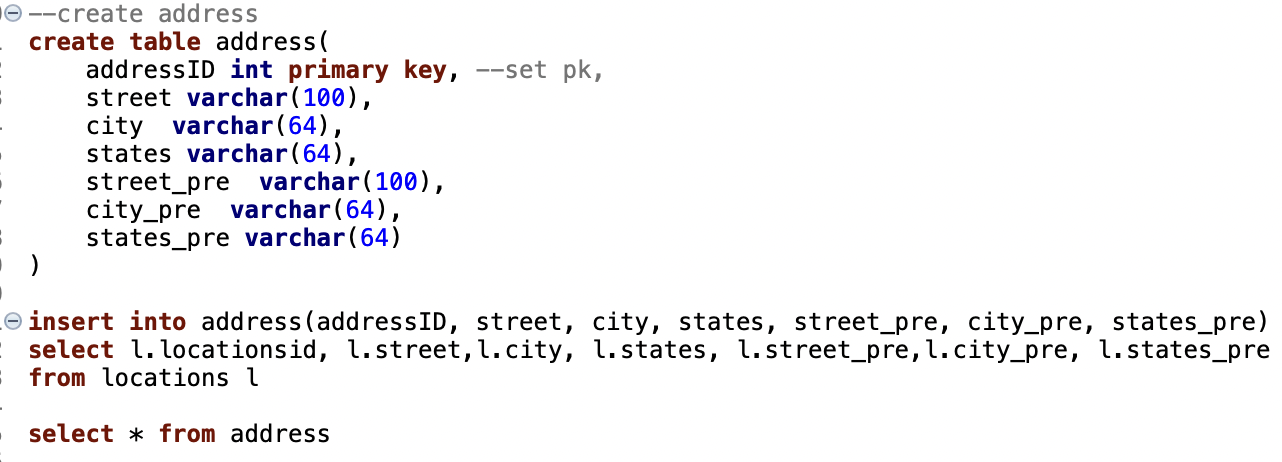
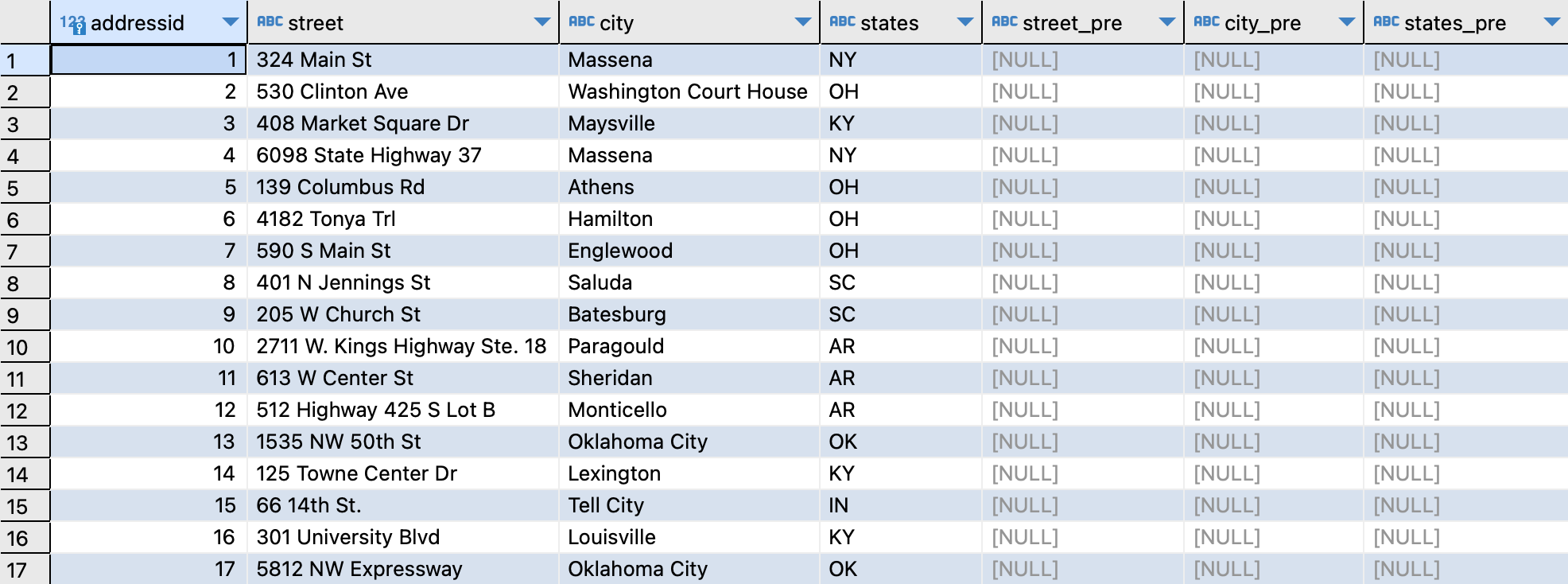


Table:



Dimension tables: Company

Code:

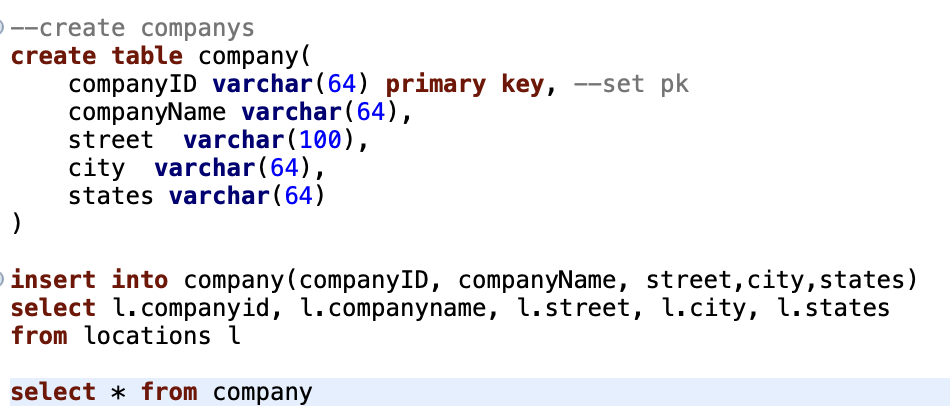
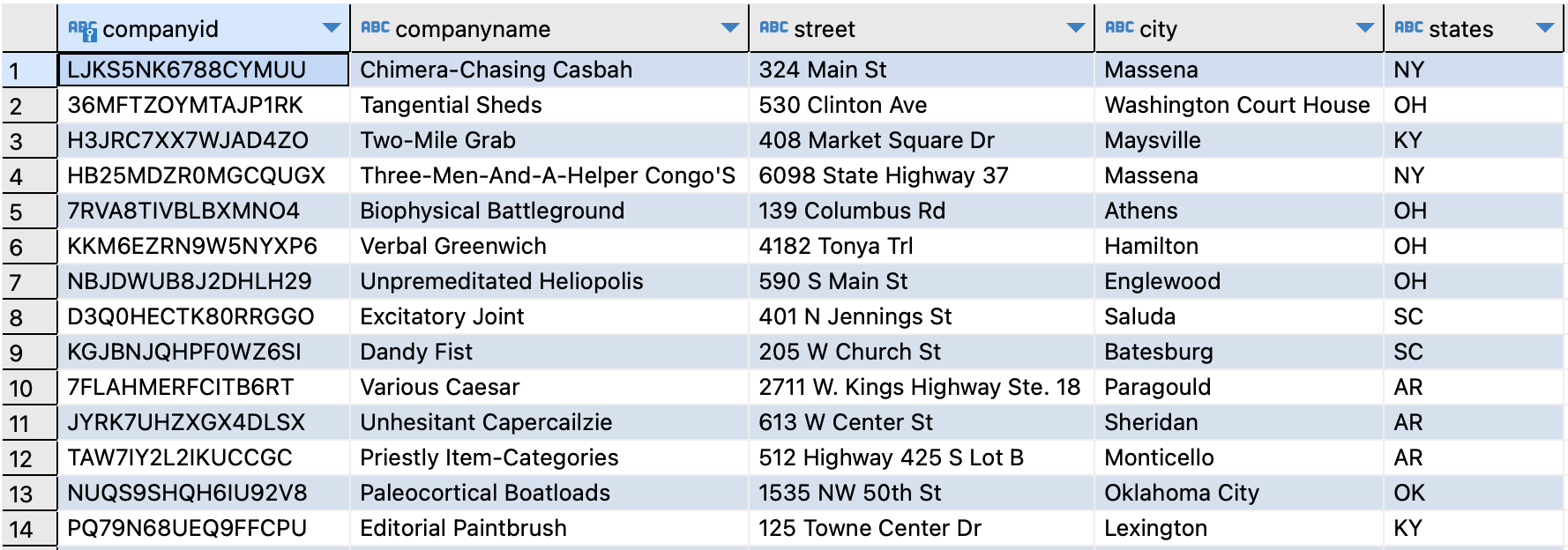


Table:

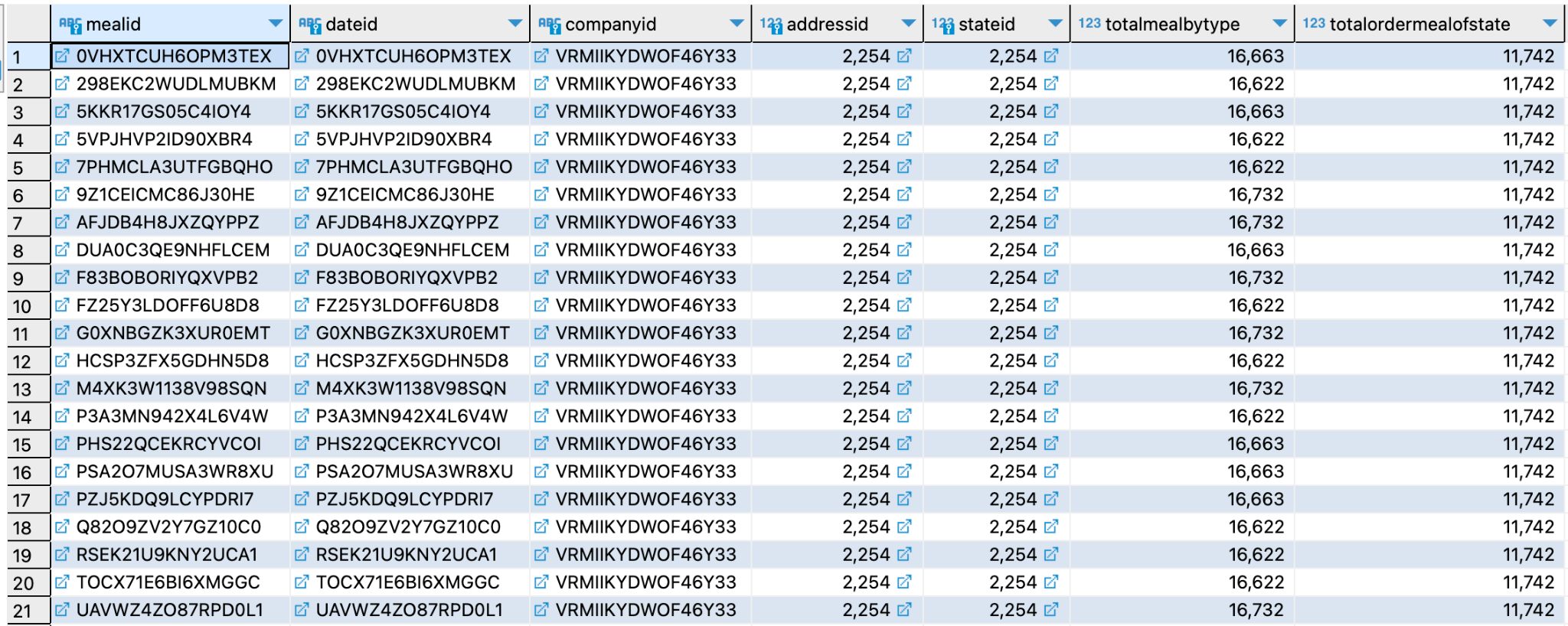


Fact table: order\_meal

Code:



Table:

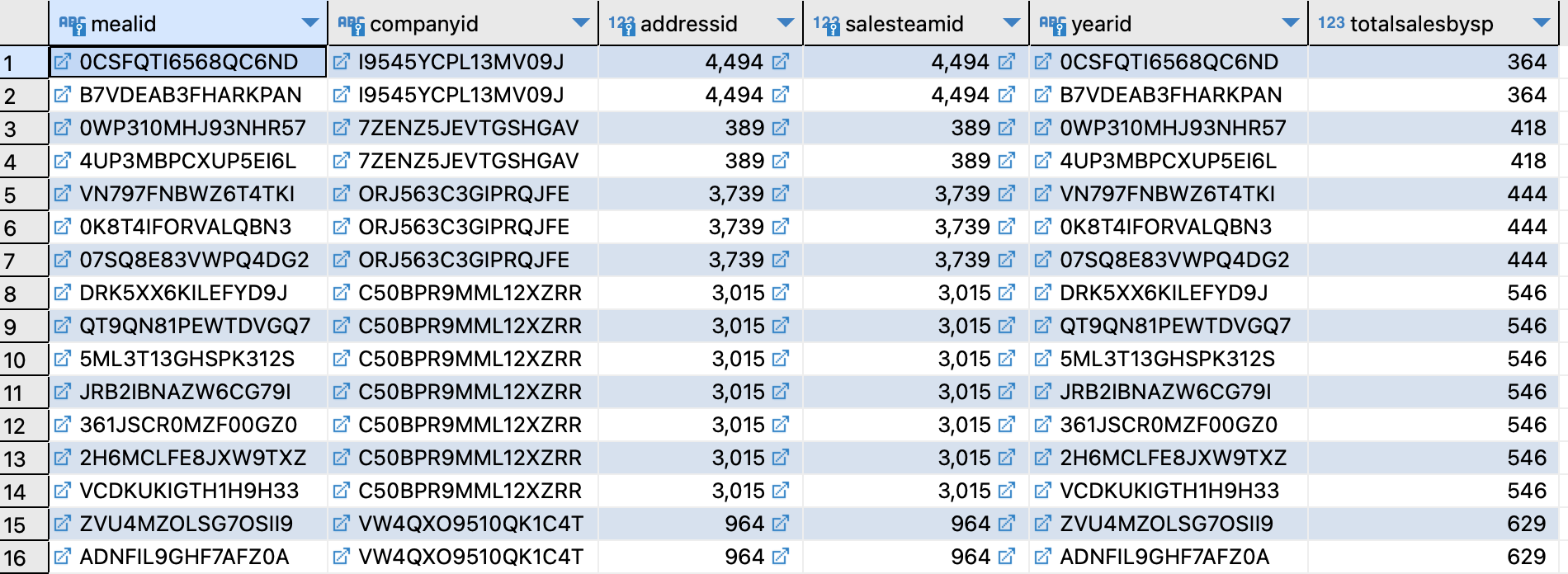


Fact table: sales\_invoices

Code:



Table:



Fact table:order\_converted

Code:

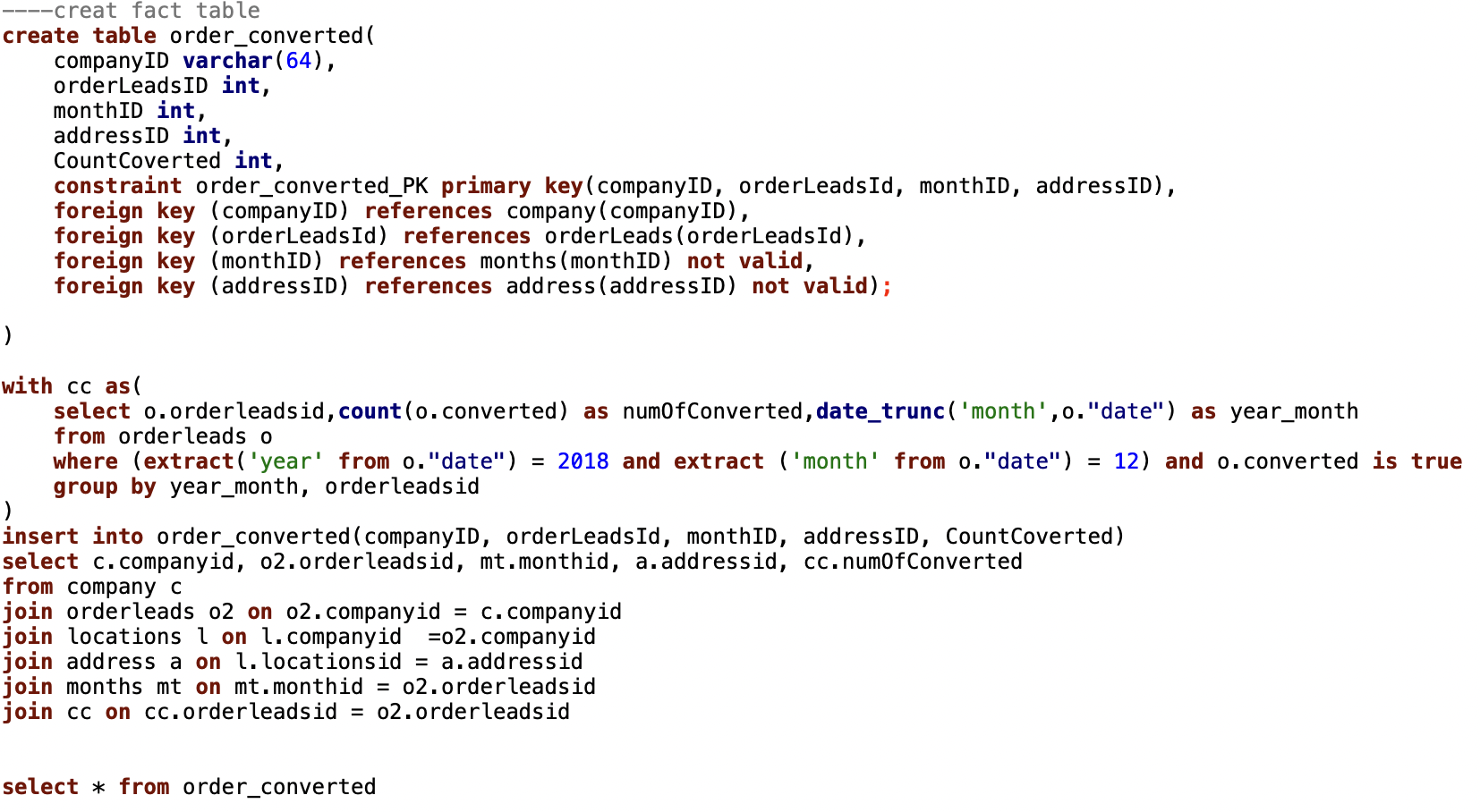
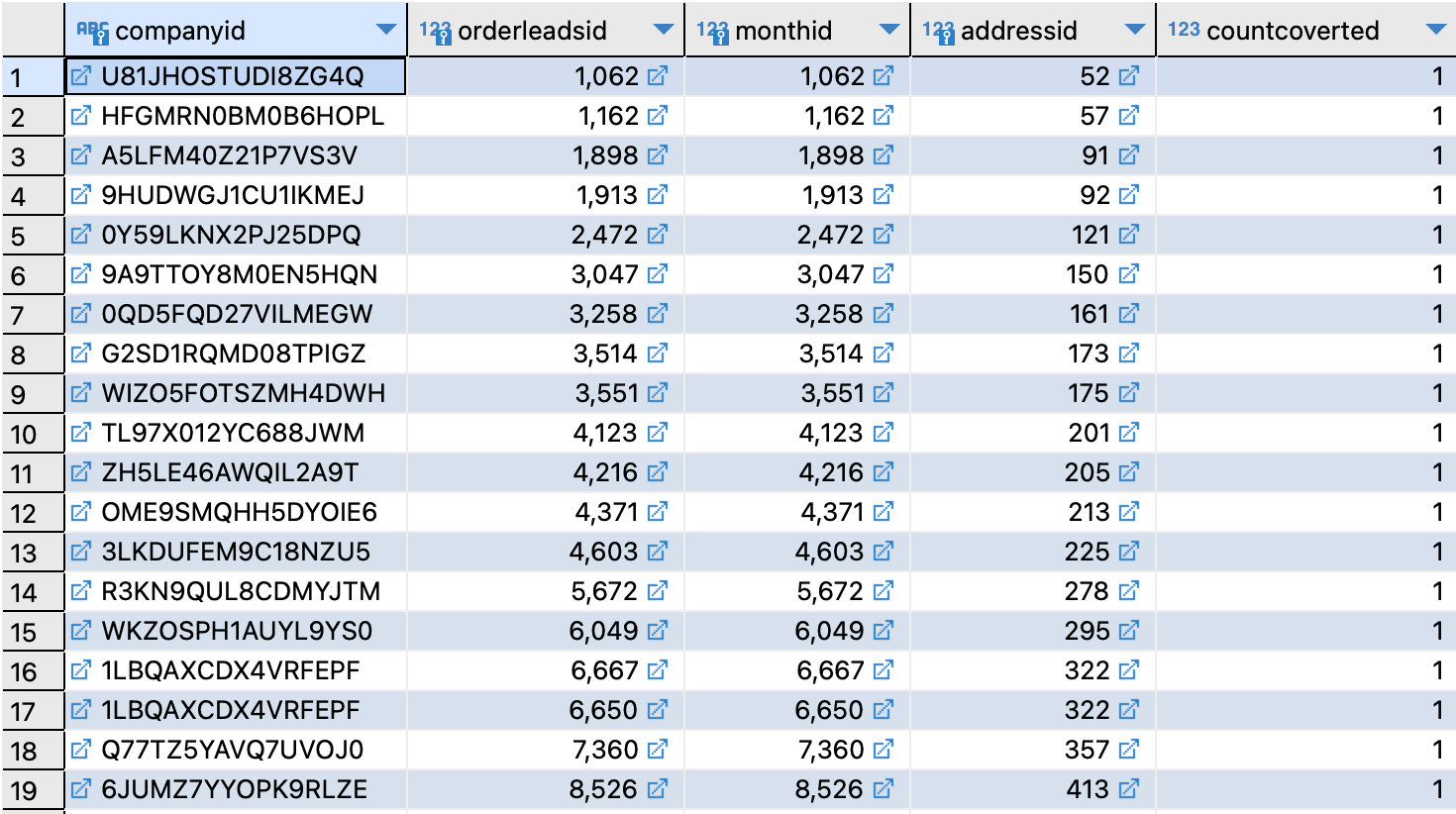


Table:



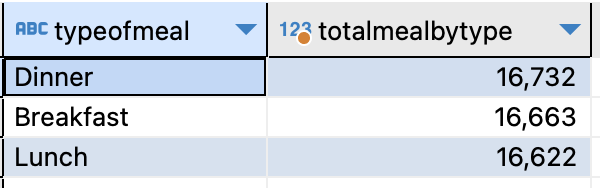
I attached the code files in Appendix, please check the order in that part.

To provide more detail of each fact table in my design, I will describe and get the results by code to implement how does each one works based on my concerned questions as below:

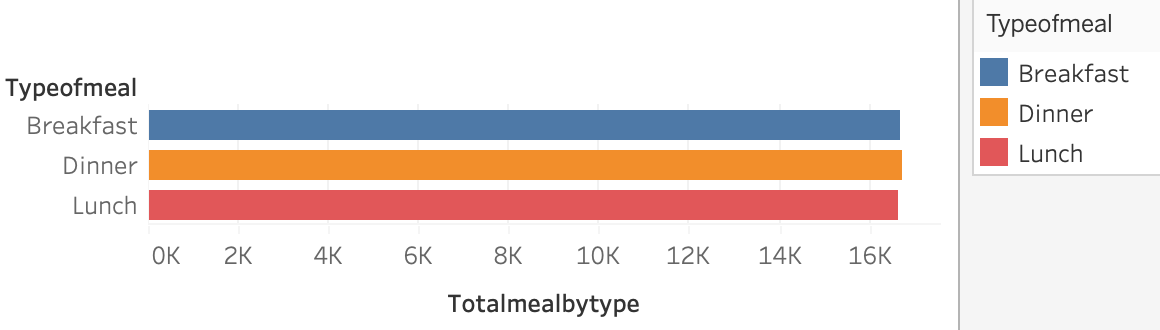
1. Which type of meal is top sold?

TotalMealByType in Order\_meal cumulative fact table gets information of type of meal and served meal price from invoices table. Sum the total served meal price by group each type of meal.

The result of sorted sold meal type list below:



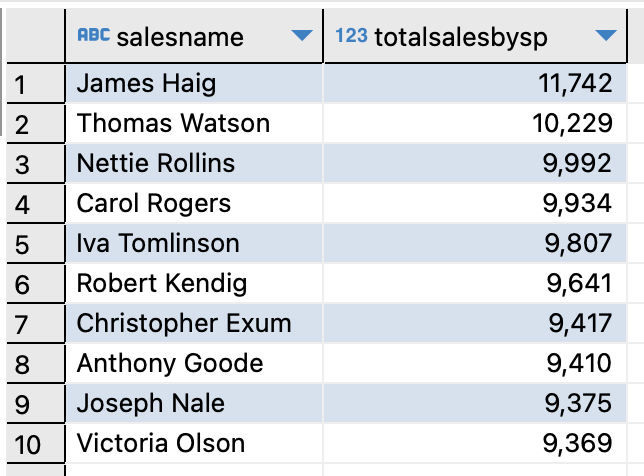
The top selling type of meal is dinner or we can say the most popular meal type is breakfast. The total sold by meal type for 2014-2018 is $16,732. We can find even breakfast is the top selling type of meal, but there is not a big gap between it and the other two types. Knowing the situation of the type of meal selling can help the supermarket to figure out some better sales plans. The visualization shows below:

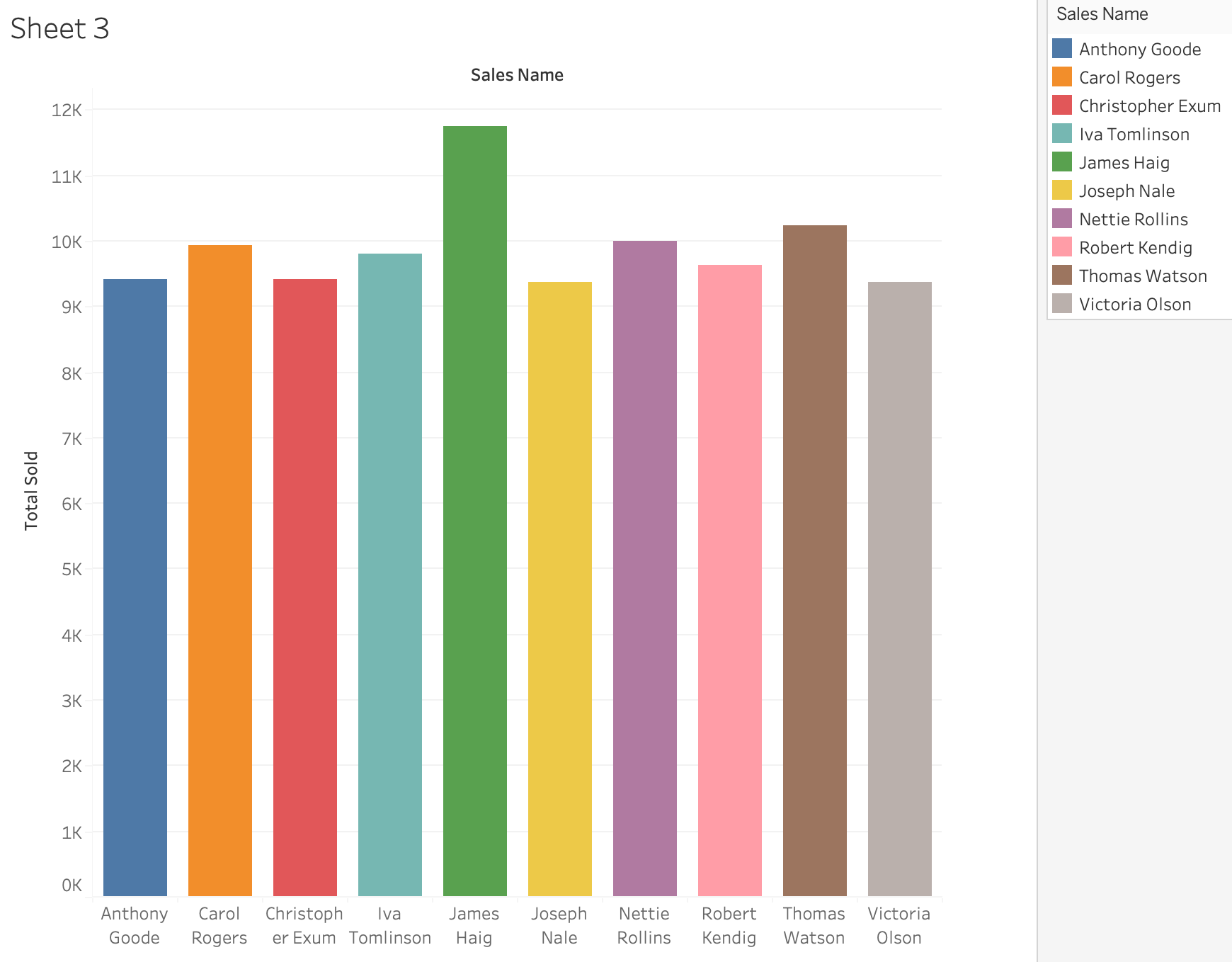


1. Who are the top 10 sales representatives that made the most sales in 2017?

If the pre-order is converted to sale, it will show in the invoice table with the served meal price. TotalSalesBySP in Sales\_invoices snapshot fact table extracts the served meal price information from Meals which extract from invoices table, get the sales name from SalesTeam table, sum the total served meal price on companyid group by sales name.

The result of top 10 sales representatives in 2017 list below:



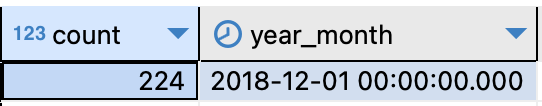
The data visualization shows below:

We can also find the top sales representatives in any specific year in the dataset. This type of analysis helps companies create incentives to encourage salespeople to perform better.

1. How many of the total pre-order converted to sale in December of 2018?

Order\_converted cumulative fact table extracts information of if pre-order converted to sale from OrderLeads table. Count the total true conversion when it happened in Dec. 2018.

The result:

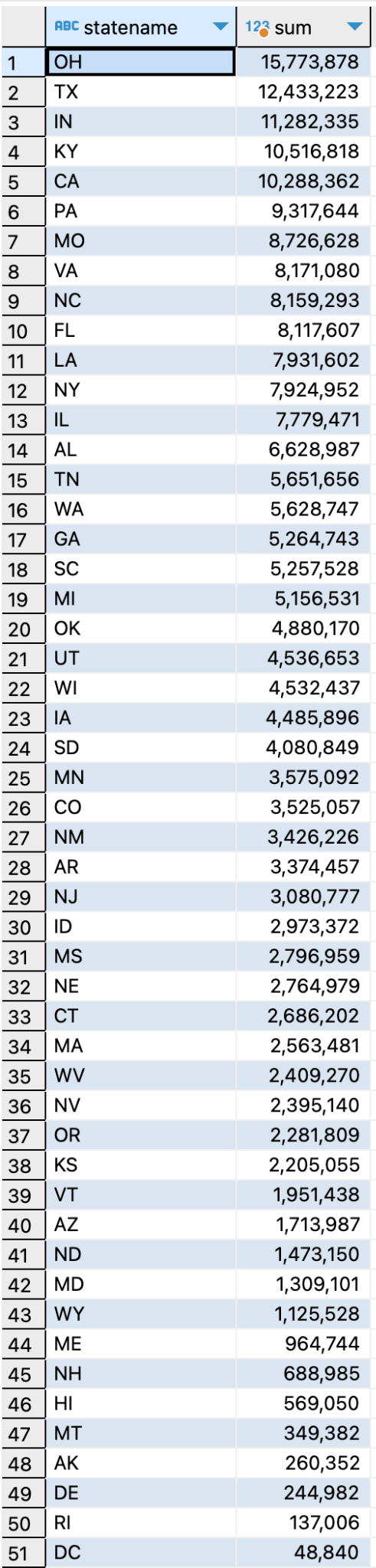


Total converted served meals from order leads in December of 2018 is 224. We can also specify years or months to get the number of converted served meals from order leads.

1. Which state has the highest values of selling?

TotalOrderMealOfState in Order\_meal extracts information of meal value from Invoices table and also extracts the states from Locations table. Sum the order value on companyid by group states.

The rankings of states by total sell values and the data visualization shows below:



The top sold state is OH.

# Data Management and Maintenance

To manage and maintain the data, I will set a few stored procedures to make the data consistent whenever specific data is changed in my slow changing dimension tables. Note: The stored procedures maintain the data on staging tables in this project.

1. Changing in Type 2 SCDs table. There are two situations we have to take care of:

1). The supermarket assigned a new salesperson to be responsible for the company or any relation changed between salesperson and company.

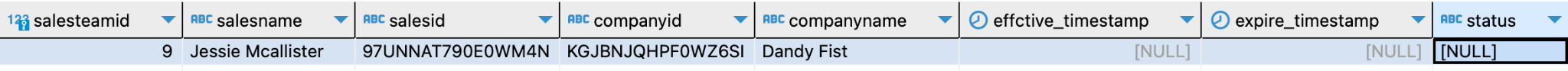
2). The company no longer works with the supermarket.

Those two situations relate to changing sales names or changing companies status.

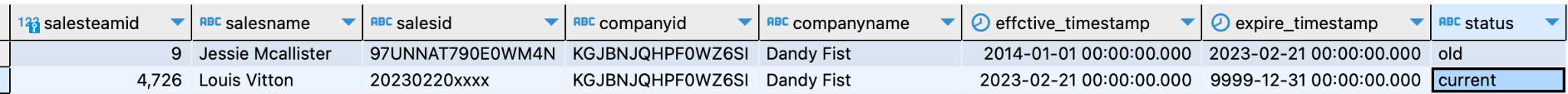
After calling the stored procedure with the new record, the stored procedures will check if a matching record exists. if yes, it will update the record with new timestamps values and status is ‘old’. And then check if it is the first situation in which the input parameter isCompanyLeft is False, then create a new record with set 2 timestamps and status is ‘current’. Otherwise, we will not create a new record until the salesperson has been assigned to a new company.

An example of first situation: Sales team table, if we assign a new sales to run company ‘Dandy Fist’ which is used to run by Jessie Mcallister, the stored procedure will find the record with the company name is ‘Dandy Fist’, company id is ‘KGJBNJQHPF0WZ6SI’ and then update the current record as status ‘old’ and give the timestamp of expired is now and set the effective time as the earliest time we can find in the database. And then insert the new record with the new sales man with the company information, set the effective is now and expire is no exit date instead of null and status is ‘current’. So that all the records are stored with the timestamp as a history.

Before:



After:

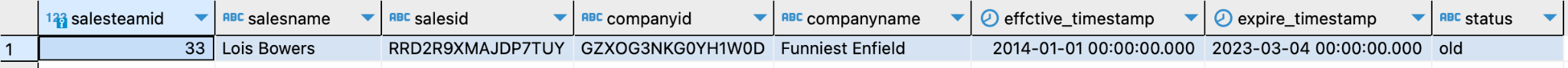


An example of the second situation: the company ‘Funniest Enfield’ stops work with supermarkets,then isCompanyLeft is true. The stored procedure will find the record with company id is ‘GZXOG3NKG0YH1W0D’ and then update the current record as status ‘old’ and give the timestamp of expired is now and set the effective time as the earliest time we can find in the database. It will not add a new record with ‘current’ status until a new company is assigned to the salesperson.

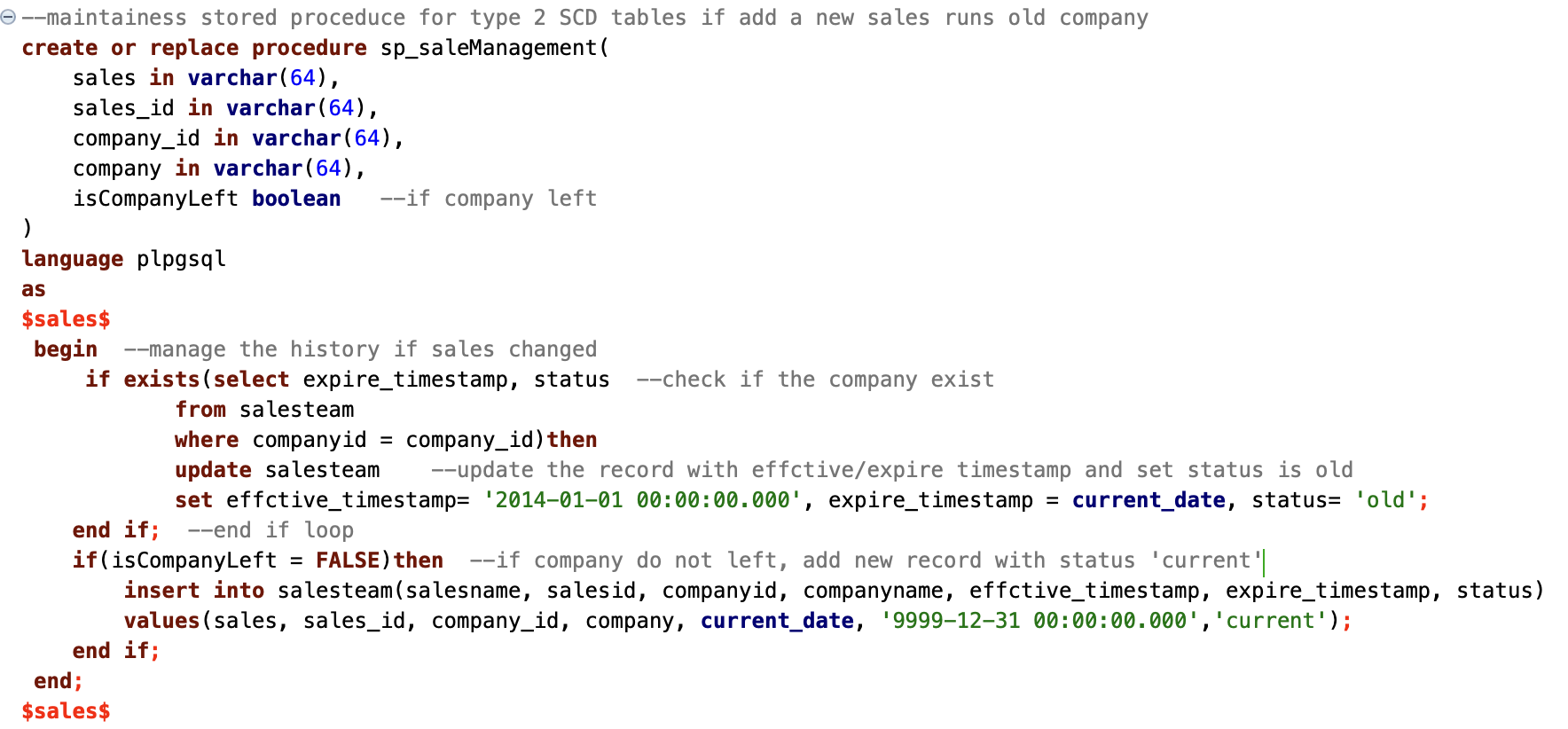
Before:



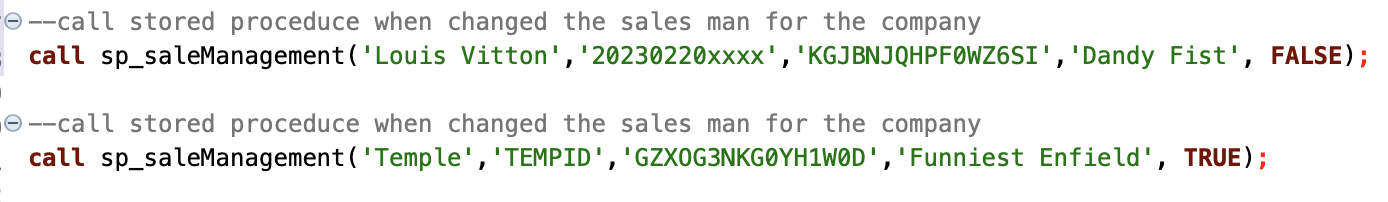
After:



Code:

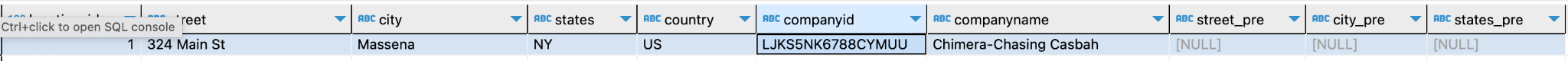


Change date by call:



1. Changing in Type 3 SCDs table.There is only one situation that is address changed. The stored procedures will check if a matching record exists. If yes, check if the value in any column changed, set the previous street, city and state in street\_pre, city\_pre and states\_pre, and new values are placed in the corresponding columns. If not, insert a new record. Usually, the address attributes will change multiple together , so I set three attributes to move together when a new address is in. For example: the company of company id is ’LJKS5NK6788CYMUU’ moved from NY to MA with a new address:

Before:

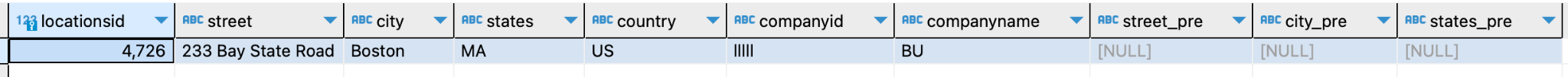


After move to a new address '3333 high st, Lexington MA':



The original address moves in corresponding columns with ‘\_pre’, and the new address replaces the old one.

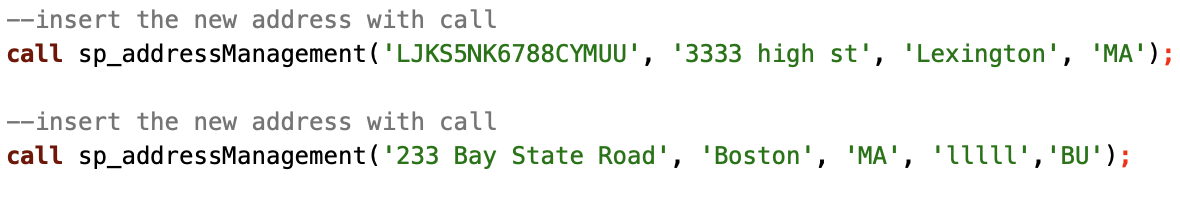
If there is a new company with a new address, the function cannot find the existing record, it will add in to the next new row. For example, company BU add in a new row with address ‘233 Bay State Road, Boston MA’ and company id is ‘lllll' after didn’t find the existing company id:



Code:



Change the address by call:



# Conclusion

In this project, I used ETL, designed and created a constellation dimensional model to create a data warehouse, using stored procedures to maintain it. The data warehouse is to assist decision-making and is closely related to business processes. The above analysis is obtained by analyzing the data in the data warehouse established by the constellation dimension model. The model uses the smallest granularity as a fact table, which is very flexible for reflecting business processes and realizing analysis requirements, and also has scalability to meet unknown query requirements, so as to meet all changes with the same.

By analyzing data in the data warehouse, I get the answers to my concerned questions. We can find that the data is evenly distributed: the type of meal sales is uniform; almost all customer companies will purchase more than two times; the total price of the reservations of the ranked companies is slowly decreasing. Although we have not obtained customer rating data, through the above analysis, we can know that customers should be satisfied with the quality of meals and willing to continue to buy. For the customer companies with the highest order volume, a reward mechanism can be adopted to thank customers for their long-term support, and at the same time, a virtuous circle of cooperation can be established. For sales representatives with top sales performance, supermarkets can also provide incentive systems to motivate sales representatives and improve morale.The customer distribution and sales situation in cities and states also play a reference role in the future decision-making of supermarkets.

# Appendices

1. CS779 term project.ipynb
2. createTables\_bw.sql
3. maintenance\_bw.sql
4. TERM PROJECT\_BeiWang.ppt

# References

1. CS779 modules and sliders.
2. PostgreSQL. <https://www.postgresql.org/docs/15/index.html>
3. When and how you should denormalization a relational database <https://rubygarage.org/blog/database-denormalization-with-examples>