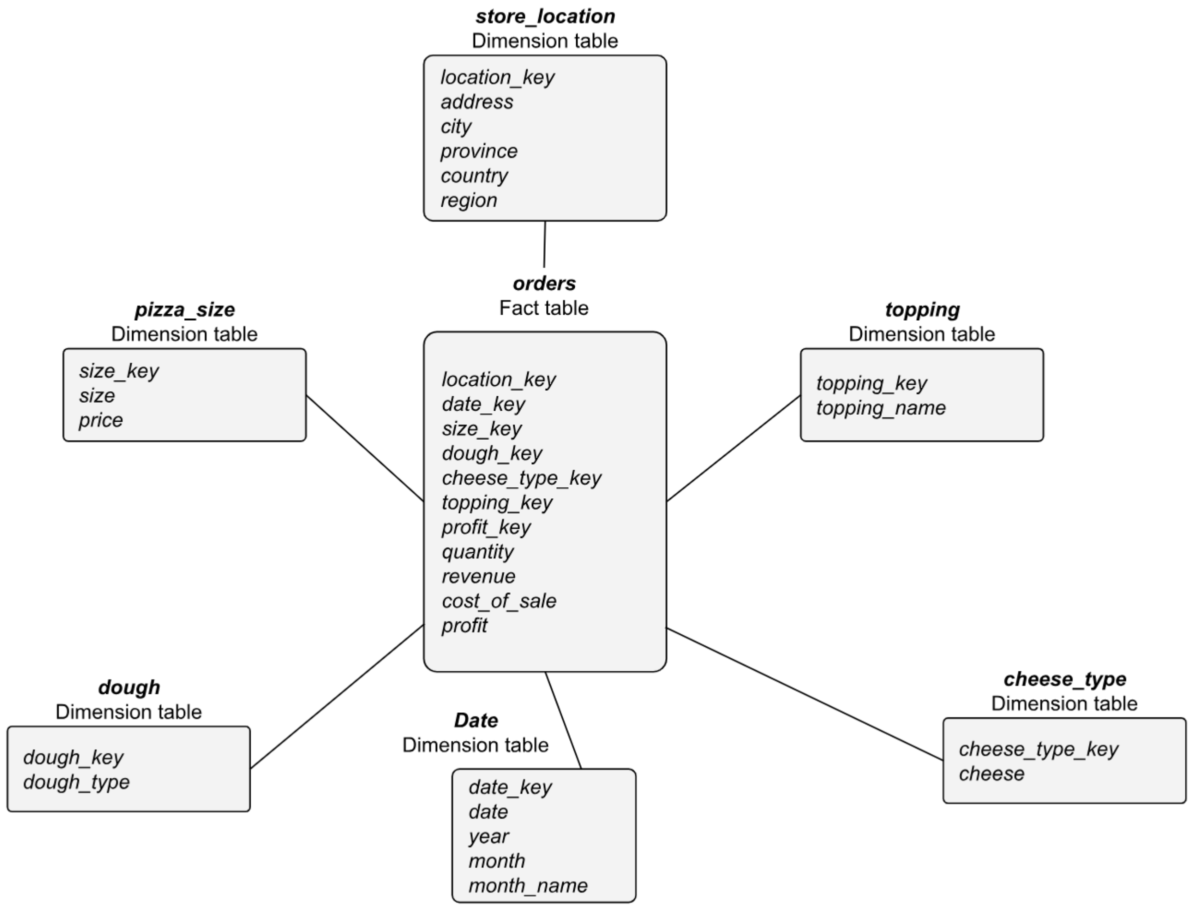
<u>Data Warehousing, OLAP Operations, Slicing and Dicing Activities</u>

Part A

1a. Star schema

The following star schema includes Orders (Fact) as well as all of its dimensions. A date dimension has been added as well for the ease of calculations in subsequent questions.

It should be noted that the size dimension (labeled pizza_size) also includes the prices of each pizza size. For the purpose of this activity, we are assuming that the price of the pizza is dependent on its size. Therefore, the other attributes of the pizza (topping, dough, cheese) do not contribute to the total price of the pizza. Revenue and profit attributes are added to the schema as they calculate the revenue and profit of the order, using another attribute called cost-of-sale (i.e., profit = revenue - cost of sale).

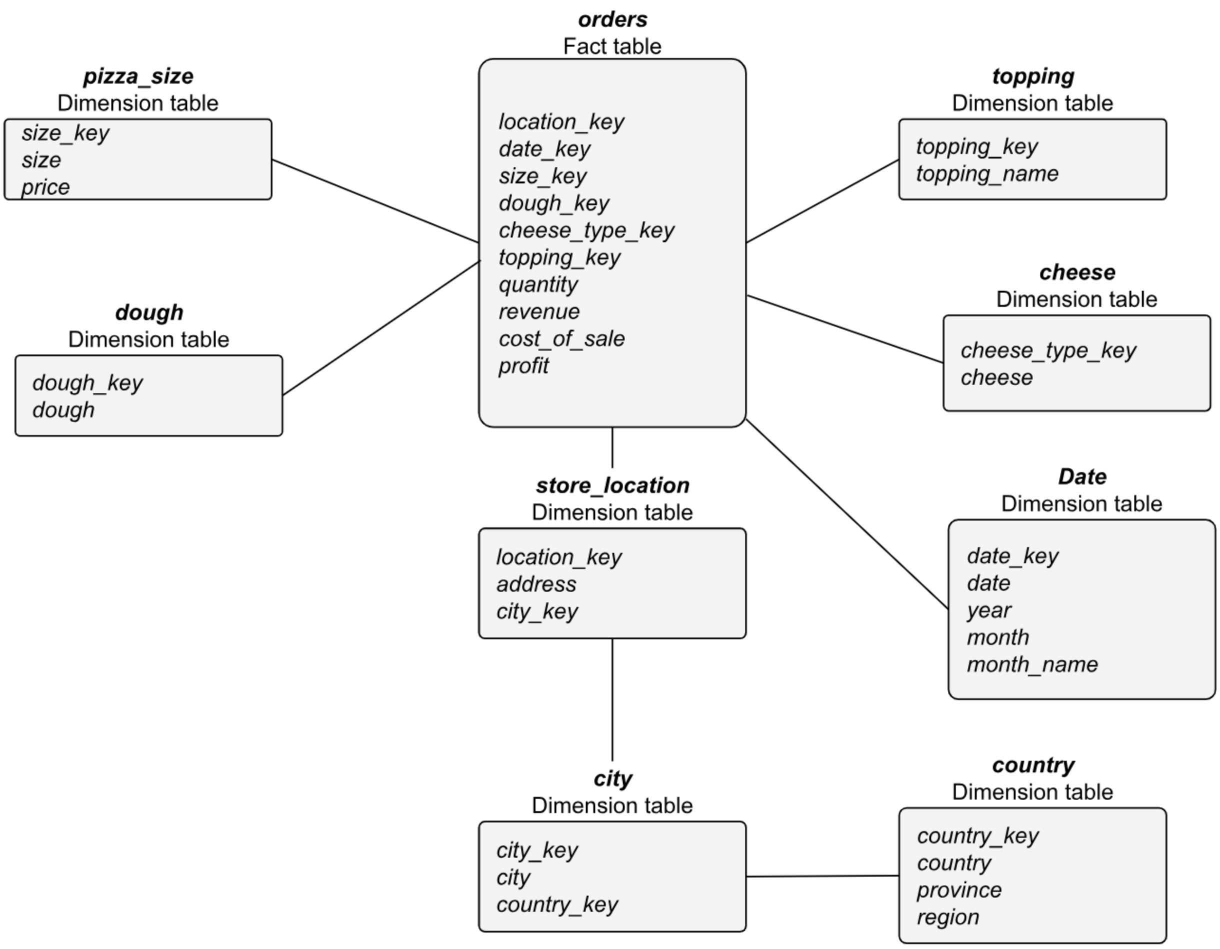


1b. Snowflake schema

In the above star schema the store_location dimension table is not normalized. Hence, we have introduced the city dimension table and country dimension table to make a snowflake, to normalize the snowflake schema and reduce redundancies. Such a table makes it easy to maintain and saves processing capabilities and thus resources.
We included 'city' and 'country' as separate dimension tables because there are thousands of cities and hundreds of countries. Furthermore, the presence of region and province attributes would require the table to be further normalized and split.
The reason for linking the country dimension table to the city dimension table is to optimize the information retrieval from the data warehouse.

Again, note that we have assumed that the price of the pizza is dependent on only the size of the pizza ordered. Same is reflected in the dimension tables.

Points to note:
- In the above diagram, we altered the fact table to add a few more attributes, which are calculated fields (i.e., revenue, profit and cost of sale), in order to improve the quality of the fact table and the ultimate analysis that would be carried out using it.We also used the same schema in our R files to generate our dimensions and facts.
- In addition, we introduce the price and cost_of_sale into the orders fact table, to calculate revenue and profit, respectively.



- We also introduced quantity and profit fields in order to provide deeper analysis of the orders. These fields are derived fields and are populated during fact population.
- Revenue is a value derived using price x quantity (in orders table).
- Profit is a value derived using revenue - cost of sale.

The R file attached provides the code that was used to build the OLAP.

The following sections provide the details of the tasks completed.

Note: Even though the keys for each dimension table are supposed to be numeric, for the purpose of better understanding our cube, we have replaced numeric values with strings.

1c. Generating data

We first created a date dimension having dates in the first 15 days of October 2023. This range can be expanded to generate dates of multiple months or years, and the code attached would simply split it into relevant months, years, month names and dates accordingly. We experimented with this and generated a sample dataset for 1 year of data, however, we proceeded with 15 days of data to keep our analysis straightforward and to increase the readability of our results.

Next, we populated the rest of our dimensions.

```
pizza_size <-
  data.frame(size_key=c("P", "S", "M", "L", "XL"),
             size=c("personal", "small", "medium", "large", "xlarge"),
             price=c(5,15,20,25,30))

topping <-
  data.frame(topping_key=c("tomatoes", "pepper", "onions", "pepperoni"),
             topping_name=c("tomatoes", "pepper", "onions", "pepperoni"))

cheese <-
  data.frame(cheese_key=c("swiss", "cheddar", "mozzarella", "parmesan"),
             cheese=c("swiss", "cheddar", "mozzarella", "parmesan"))

dough <-
  data.frame(dough_key=c("whole weat thin", "white regular", "stuffed crust", "regular"),
             dough=c("whole weat thin", "white regular", "stuffed crust", "regular"))

store_location <-
  data.frame(location_key=c("1","2","3","4"),
             address=c("336 Rideau St","458 Rideau St","471 Yonge St", "124 Fulton St"),
             city_key=c("2", "2", "2", "1"))

city <-
  data.frame(city_key = c("1","2"),
             city = c("New York", "Ottawa"),
             country_key = c("1","2"))

country <-
  data.frame(country_key = c("1","2"),
             country = c("United States of America","Canada"),
             province = c("New York","Ontario"),
             region = c("region a", " region b"))


profit <-
  data.frame(profit_key=c("1", "2", "3"),
             profit=c(22.5, 5.70, 11.20),
             revenue = c(45, 12, 32))

quantity <- c(1,2,3)
```

All of the dimensions were with respective fields. In the case of topping, cheese, and dough our primary key was the same as the name to enhance the readability of our cube. In the case where we added the numeric values to the key, we proceeded to merge the fact with the respective dimension to retrieve the non numeric values (cheese names, dough name etc.) to improve readability. However, we have not included this into our code at this point.

The quantity field was introduced here but used while sampling.

Next step was to create a function to generate orders which would populate our Orders fact. We completed this by using the following code:

```r
# Function to generate the Orders table
gen_orders <- function(no_of_recs) {
  # Generate transaction data randomly
  location_ <- sample(store_location$location_key, no_of_recs, replace=T, prob=c(2,1,1,2))
  order_date_ <- sample(date$date_key, no_of_recs, replace=T)
  order_month_ <- sample(date$Month_Name, no_of_recs, replace=T)
  size_ <- sample(pizza_size$size_key, no_of_recs, replace=T, prob=c(1, 3, 2, 3, 1))
  dough_ <- sample(dough$dough_key, no_of_recs, replace=T, prob=c(1, 3, 1, 3))
  cheese_ <- sample(cheese$cheese_key, no_of_recs, replace=T, prob=c(2, 3, 1, 2))
  topping_ <- sample(topping$topping_key, no_of_recs, replace=T, prob=c(3, 2, 1, 1))
  #profit_ <- sample(profit$profit_key, no_of_recs, replace=T, prob=c(1, 3, 2))
  quantity_ <- sample(quantity, no_of_recs, replace=T, prob=c(3, 2, 1))
  price <- pizza_size$price[match(size_, pizza_size$size_key)]
  revenue <- price * quantity_
  cost_of_sales <- (sample(c(0.8,0.85), no_of_recs, replace=T, prob=c(3, 2))) * revenue
  profit_ <- revenue - cost_of_sales


  orders <- data.frame(location_      #= location_,
                      ,order_date_    #= order_date_,
                      ,order_month_
                      ,size_          #= size_,
                      ,dough_         #= dough_,
                      ,cheese_        #= cheese_,
                      ,topping_       #= topping_,
                      #,profit_        #= profit_,
                      ,quantity_       #= quantity
                      ,price
                      ,revenue
                      ,cost_of_sales
                      ,profit_
                      )

  # Sort the records by time order
  orders <- orders[order(orders$order_date_),]
  row
  return(orders)
}

# Creating the orders_fact using function
orders_fact <-  gen_orders(500)

```

The fields revenue, cost_of_sales and profit_ are all calculated fields which have been calculated while populating the fact. This would certainly help in keeping the formulas for all of these fields constant. Even though this might be a costly activity in terms of resources, this approach makes the most sense since we are generating our own data and do not have data from a database providing us with these values.

2. Once generated, we used the Orders fact to build our revenue cube.

This revenue_cube had the dimensions of size, order date, location, dough, cheese, topping and order month. To analyze other aspects of the order, for example profits, quantities sold or cost_of_sales, other cubes can also be built. We have included some of these in our code which we used for verification of our cube.

```r
# Build up a cube for revenue
revenue_cube <-
  tapply(orders_fact$revenue,
         orders_fact[,c("size_","order_date_", "location_","dough_","cheese_", "topping_","order_month_")]
         function(x){return(sum(x))})

# Showing the cells of the cube
revenue_cube

```

With our OLAP cube complete, we moved on to analyzing the data in it.
We started with applying a series of roll-up operations to bifurcate the fact data into dimension, leading us to finding some extremely interesting trends in our dataset.

```
#3.

#Roll-up 1 - Revenue in terms of size and topping
apply(revenue_cube, c("size_", "topping_"),
        FUN=function(x) {return(sum(x, na.rm=TRUE))})

#Roll-up 2 - Revenue in terms of size and dough
apply(revenue_cube, c("size_", "dough_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})

#Roll-up 3 - Revenue in terms of size and cheese
apply(revenue_cube, c("size_", "cheese_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Roll-up 4 - Revenue in terms of dough and cheese
apply(revenue_cube, c("dough_", "cheese_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Roll-up 5 - Revenue in terms of dough and cheese
apply(revenue_cube, c("dough_", "topping_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Roll-up 6 - Revenue in terms of size and location
apply(revenue_cube, c("size_", "location_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})

```

```
> #Roll-up 1 - Revenue in terms of size and topping
> apply(revenue_cube, c("size_", "topping_"),
+         FUN=function(x) {return(sum(x, na.rm=TRUE))})
     topping_
size_ onions pepper pepperoni tomatoes
   L   1050   1600      1025     2050
   M    540    920       560     1480
   P    115    105        50      185
   S    570    915       780     1665
   XL   420    780       450     1260
>
> #Roll-up 2 - Revenue in terms of size and dough
> apply(revenue_cube, c("size_", "dough_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     dough_
size_ regular stuffed crust white regular whole weat thin
   L     1825           800          2175             925
   M     1260           400          1360             480
   P      215            30           165              45
   S     1260           510          1635             525
   XL    1350           330           900             330
>
> #Roll-up 3 - Revenue in terms of size and cheese
> apply(revenue_cube, c("size_", "cheese_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     cheese_
size_ cheddar mozzarella parmesan swiss
   L     2225        850     1525  1125
   M     1520        460      560   960
   P      150         30      150   125
   S     1260        555     1140   975
   XL    1380        570      420   540
>
```

```
>
> #Roll-up 4 - Revenue in terms of dough and cheese
> apply(revenue_cube, c("dough_", "cheese_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
                  cheese_
dough_             cheddar mozzarella parmesan swiss
  regular             2350        870     1245  1445
  stuffed crust        815        275      540   440
  white regular       2280       1010     1625  1320
  whole weat thin     1090        310      385   520
>
>
> #Roll-up 5 - Revenue in terms of dough and cheese
> apply(revenue_cube, c("dough_", "topping_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
                  topping_
dough_             onions pepper pepperoni tomatoes
  regular            1095   1530       860     2425
  stuffed crust       510    480       345      735
  white regular       895   1675      1160     2505
  whole weat thin     195    635       500      975
>
>
> #Roll-up 6 - Revenue in terms of size and location
> apply(revenue_cube, c("size_", "location_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     location_
size_    1    2   3    4
   L  2125 1075 800 1725
   M  1000  380 800 1320
   P   160   75  55  165
   S   915  615 885 1515
   XL  660  660 540 1050
> |
```

With our roll-up operations, we quickly analyzed the low revenue generated by "Personal" pizza size across all dimensions. And in every roll-up operation, the highest revenue was brought in by "Large" pizza size orders.

Roll-up analysis

1. In the size vs toppings roll-up, Large pizzas with tomatoes generated the highest revenue, while Personal pizzas with pepperoni topping was the least popular. Overall, the tomato toppings generated the most revenue across all pizza sizes.

2. Similarly, in size vs dough comparison, we noticed that regular and white regular dough performed equally as well, competing with each other very closely. While both stuffed crust and whole weat thin lagged behind, with stuffed crust performing the worst.

3. The most interesting roll-up results were brought by rollup 6 comparing size and location. It seemed that while all locations performed almost equally well, the Personal pizza performed the worst in all locations
.

4. It is worth noting, that in all of the roll up operations performed, personal pizzas were the worst performing pizza sizes. Where as the large and extra large pizzas were the most sold regardless of the location, topping, order data. Therefore, customers are preferring larger pizzas over small or medium sized pizzas.

```
#Drill-down 1 - Revenue in terms of order_month_ and location_ and size_
apply(revenue_cube, c("order_month_", "location_", "size_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Drill-down 2 - Revenue in terms of order_month_ and topping_ and size_
apply(revenue_cube, c("order_month_", "topping_", "size_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Drill-down 3 - Revenue in terms of order_month_ and topping_ and size_
apply(revenue_cube, c("order_month_", "location_", "size_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Drill-down 4 - Revenue in terms of order_month_ and topping_ and size_
apply(revenue_cube, c("dough_", "cheese_", "size_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


#Drill-down 5 - Revenue in terms of order_month_ and topping_ and size_
apply(revenue_cube, c("size_", "dough_", "location_"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
> #Drill-down 1 - Revenue in terms of order_month_ and location_ and size_
> apply(revenue_cube, c("order_month_", "location_", "size_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , size_ = L

            location_
order_month_    1    2   3    4
     October 2125 1075 800 1725

, , size_ = M

            location_
order_month_    1   2   3    4
     October 1000 380 800 1320

, , size_ = P

            location_
order_month_   1  2  3   4
     October 160 75 55 165

, , size_ = S

            location_
order_month_   1   2   3    4
     October 915 615 885 1515

, , size_ = XL

            location_
order_month_   1   2   3    4
     October 660 660 540 1050
```

```
> #Drill-down 2 - Revenue in terms of order_month_ and topping_ and size_
> apply(revenue_cube, c("order_month_", "topping_", "size_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , size_ = L

            topping_
order_month_ onions pepper pepperoni tomatoes
     October   1050   1600      1025     2050

, , size_ = M

            topping_
order_month_ onions pepper pepperoni tomatoes
     October    540    920       560     1480

, , size_ = P

            topping_
order_month_ onions pepper pepperoni tomatoes
     October    115    105        50      185

, , size_ = S

            topping_
order_month_ onions pepper pepperoni tomatoes
     October    570    915       780     1665

, , size_ = XL

            topping_
order_month_ onions pepper pepperoni tomatoes
     October    420    780       450     1260

> #Drill-down 3 - Revenue in terms of order_month_ and topping_ and size_
> apply(revenue_cube, c("order_month_", "location_", "size_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , size_ = L

            location_
order_month_    1    2   3    4
     October 2125 1075 800 1725

, , size_ = M

            location_
order_month_    1   2   3    4
     October 1000 380 800 1320

, , size_ = P

            location_
order_month_   1  2  3   4
     October 160 75 55 165

, , size_ = S

            location_
order_month_   1   2   3    4
     October 915 615 885 1515

, , size_ = XL

            location_
order_month_   1   2   3    4
     October 660 660 540 1050
```

```
> #Drill-down 4 - Revenue in terms of order_month_ and topping_ and size_
> apply(revenue_cube, c("dough_", "cheese_", "size_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , size_ = L

                 cheese_
dough_             cheddar mozzarella parmesan swiss
  regular              825        275      400   325
  stuffed crust        275         25      325   175
  white regular        675        350      650   500
  whole weat thin      450        200      150   125

, , size_ = M

                 cheese_
dough_             cheddar mozzarella parmesan swiss
  regular              460        160      200   440
  stuffed crust        200         40      100    60
  white regular        640        240      220   260
  whole weat thin      220         20       40   200

, , size_ = P

                 cheese_
dough_             cheddar mozzarella parmesan swiss
  regular               75         15       75    50
  stuffed crust         10          0       10    10
  white regular         50         15       35    65
  whole weat thin       15          0       30     0

, , size_ = S

                 cheese_
dough_             cheddar mozzarella parmesan swiss
  regular              390         90      360   420
  stuffed crust        180        180      105    45
  white regular        525        195      570   345
  whole weat thin      165         90      105   165

, , size_ = XL

                 cheese_
dough_             cheddar mozzarella parmesan swiss
  regular              600        330      210   210
  stuffed crust        150         30        0   150
  white regular        390        210      150   150
  whole weat thin      240          0       60    30
```

```
> #Drill-down 5 - Revenue in terms of order_month_ and topping_ and size_
> apply(revenue_cube, c("size_", "dough_", "location_"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , location_ = 1

     dough_
size_ regular stuffed crust white regular whole weat thin
   L      800           275           850             200
   M      380            60           360             200
   P       70            20            60              10
   S      180           135           495             105
   XL     240           150           120             150

, , location_ = 2

     dough_
size_ regular stuffed crust white regular whole weat thin
   L      475           100           300             200
   M      180            60            80              60
   P       30             0            35              10
   S      150           150           225              90
   XL     360            90           210               0

, , location_ = 3

     dough_
size_ regular stuffed crust white regular whole weat thin
   L      150           125           325             200
   M      300           100           400               0
   P       35             0            20               0
   S      285            75           345             180
   XL     180            30           270              60

, , location_ = 4

     dough_
size_ regular stuffed crust white regular whole weat thin
   L      400           300           700             325
   M      400           180           520             220
   P       80            10            50              25
   S      645           150           570             150
   XL     570            60           300             120
```

Drill-down analysis

1. Through our drill down operations, we were able to find staggering analysis patterns. In our first operation we drilled down into the OLAP cube on the 3 dimensions of order month, location and size. We found that as per our roll-up operations, Personal pizzas were sold extremely less than any other pizza. Therefore, the store should not focus on getting more components of Personal sized pizza and instead buy large and extra large in bulk quantities as they prove to be the most selling and revenue generating sizes.
2. When we drilled down into order month vs topping vs size, we analyzed that pepperoni and onion generated the least revenue in the entire month for all pizza sizes. Whereas tomatoes were the most favored. It would thus be advised to the store to buy more tomato topping components than pepperoni. Pepper toppings were closely following tomatoes and would require to be replenished.
3. Our most important finding was in drilling down dough vs cheese vs size. In this operation we noticed that pizzas with mozzarella cheese and stuffed crust were never ordered. Same was the case with whole wheat thin dough and mozzarella and swiss cheese. Therefore, these components should not be bought or bought in very less amounts as they did not generate ANY revenue for the store. Furthermore regular dough with cheddar cheese is the most selling pizza, and must be replenished in high amounts as that would generate the highest revenue for the store.
4. Much like our roll0up analysis, the drop-down operations confirmed that the personal pizza sizes were the least bought. Whereas the large and extra large were generating the most revenue. It can thus be concluded that the large and XL pizzas are preferred by he customers and the store should prefer buying components for these size.s

Note: Please note that, in Part A, question 1C and 2 (first part), the data was generated using R, as per the confirmation on email given below. We are mentioning this point in this report just to make sure there is no confusion in the assignment question and the corresponding responses presented in this document.

Re: DTI 5126 Assignment 2 questions

mailto:lpaiv023@uottawa.ca

Pouya Khodaee <pkhod015@uottawa.ca>
Sat 11/4/2023 7:40 PM
To:Lakshika Josephine Paiva <lpaiv023@uottawa.ca>
Cc:Usman Bashir <mbash028@uottawa.ca>

Hi,

Yes, just get the idea from tutorial material and do not copy-paste. No .CSV file is available for the first part.

Best,
Pouya

From: Lakshika Josephine Paiva <lpaiv023@uottawa.ca>
Date: Saturday, November 4, 2023 at 6:06 PM
To: Pouya Khodaee <pkhod015@uottawa.ca>
Cc: Usman Bashir <mbash028@uottawa.ca>
Subject: Re: DTI 5126 Assignment 2 questions

Hi Pouya,

Thanks.
For Part A, Q1-c, can we generate the data for the dimension files using the data generation code taught in tutorial 2?

Or should we just create it manually/ offline in CSV files and then read them using R code?

Thanks and regards,
Lakshika

Part B

Prior to the tasks in the assignment, the data was imported into RStudio. Then the .csv file with delimiters, was converted to a table to ensure better readability in rows and columns. R code used for the process is shown:

Import:

```
> library(readr)
> bank_additional_full <- read_csv("bank-additional-full.csv")
Rows: 41188 Columns: 1
── Column specification ──────────────────────────────────────
Delimiter: ","
chr (1): age;job;marital;education;default;housing;loan;contact;month;d...
```

Imported data:

bank_additional_full × | bank_additional_full.df ×

column 1: character

Filter

	age;job;marital;education;default;housing;loan;contact;month;day_of_week;duration;campaign;pdays;previous;poutcome;emp.var
1	56;"housemaid";"married";"basic.4y";"no";"no";"no";"te...
2	57;"services";"married";"high.school";"unknown";"no";"...
3	37;"services";"married";"high.school";"no";"yes";"no";"t...
4	40;"admin.";"married";"basic.6y";"no";"no";"no";"teleph...
5	56;"services";"married";"high.school";"no";"no";"yes";"t...
6	45;"services";"married";"basic.9y";"unknown";"no";"no"...
7	59;"admin.";"married";"professional.course";"no";"no";...
8	41;"blue-collar";"married";"unknown";"unknown";"no";...

Showing 1 to 9 of 41,188 entries, 1 total columns

Converted data into a table and assigned to a data frame.

> bank_additional.df = read.csv(file="bank-additional-full.csv", sep=";", header=T)

Converted data:

bank_additional_full × | bank_additional_full.df ×

Filter

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
1	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon
2	57	services	married	high.school	unknown	no	no	telephone	may	mon
3	37	services	married	high.school	no	yes	no	telephone	may	mon
4	40	admin.	married	basic.6y	no	no	no	telephone	may	mon
5	56	services	married	high.school	no	no	yes	telephone	may	mon
6	45	services	married	basic.9y	unknown	no	no	telephone	may	mon
7	59	admin.	married	professional.course	no	no	no	telephone	may	mon
8	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon

Showing 1 to 9 of 41,188 entries, 21 total columns

Answers to questions:

1. The data columns which are required were preserved and other columns were dropped.
   > bank_simple.df <- bank_additional_full.df[-c(2:3, 5:12, 15:20)]
   > View(bank_simple.df)

bank_additional_full × | bank_additional_full.df × | bank_simple ×

Filter

	age	education	pdays	previous	y
1	56	basic.4y	999	0	no
2	57	high.school	999	0	no
3	37	high.school	999	0	no
4	40	basic.6y	999	0	no
5	56	high.school	999	0	no
6	45	basic.9y	999	0	no
7	59	professional.course	999	0	no
8	41	unknown	999	0	no
9	24	professional.course	999	0	no
10	25	high.school	999	0	no

Showing 1 to 11 of 41,188 entries, 5 total columns

In addition, the "y" column was renamed to "response" as it was not descriptive of the data in the column.

> names(bank_simple.df)[names(bank_simple.df) == "y"] <- "response"

onal_full.df × | bank_additional.df × | bank_additional_full × | bank_simple.df ×

Filter

	age	education	pdays	previous	response
1	56	basic.4y	999	0	no
2	57	high.school	999	0	no
3	37	high.school	999	0	no
4	40	basic.6y	999	0	no
5	56	high.school	999	0	no
6	45	basic.9y	999	0	no
7	59	professional.course	999	0	no
8	41	unknown	999	0	no

Showing 1 to 8 of 41,188 entries, 5 total columns

2. The value "999" is replaced with "NA". The 'before' and 'after' output is shown below.

Before:

```
> table(bank_simple.df$pdays)

    0     1    10    11    12    13    14    15    16    17    18    19 
   15    26    52    28    58    36    20    24    11     8     7     3 
    2    20    21    22    25    26    27     3     4     5     6     7 
   61     1     2     3     1     1     1   439   118    46   412    60 
    8     9   999 
   18    64 39673 
```

bank_simple.df$pdays[bank_simple.df$pdays == "999"] <- "NA"

After:

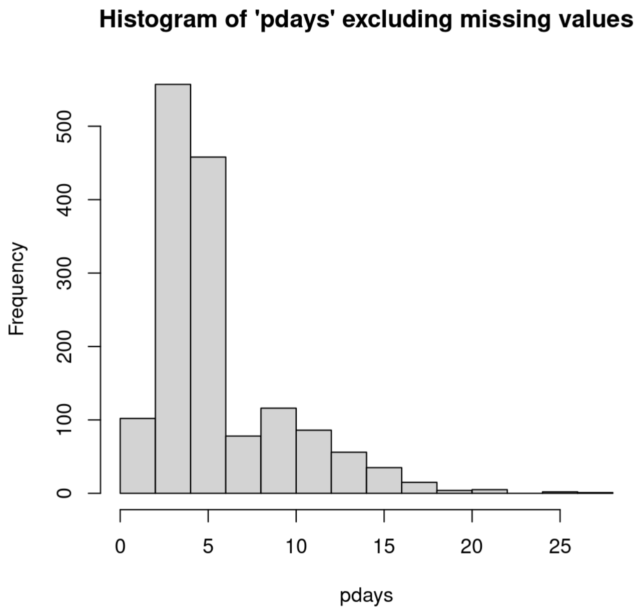
```
> table(bank_simple.df$pdays)

    0     1    10    11    12    13    14    15    16    17    18    19 
   15    26    52    28    58    36    20    24    11     8     7     3 
    2    20    21    22    25    26    27     3     4     5     6     7 
   61     1     2     3     1     1     1   439   118    46   412    60 
    8     9    NA 
   18    64 39673 
```

3. In reality '999' denotes missing values. Because the 'pdays' column is numeric, having 999 in comparison to the other values which are drastically smaller, will make '999' look like part of the data and affect/skew the data in calculations. For example, if the mean is calculated including '999' values, the result will take an abnormal value with 999 in the calculation, whereas actually it is significantly lesser than that. Hence, it is better to substitute it with NA.

4. See the code and histogram below:

```
> bank_simple.df$pdays[bank_simple.df$pdays == 999] <- "NA"
> bank_simple.df$pdays <- as.numeric(as.character(bank_simple.df$pdays))
> hist(bank_simple.df$pdays[!is.na(bank_simple.df$pdays)], xlab="pdays",
ylab="Frequency", main="Histogram of pdays excluding missing values")
```

The histogram is self-explanatory, and it shows the frequency of pdays.



5. Converting categorical values to numerical values in the education column.

```
> bank_simple.df$education[bank_simple.df$education=="illiterate"]<-"0"
> bank_simple.df$education[bank_simple.df$education=="basic.4y"]<-"4"
> bank_simple.df$education[bank_simple.df$education=="basic.6y"]<-"6"
> bank_simple.df$education[bank_simple.df$education=="basic.9y"]<-"9"
> bank_simple.df$education[bank_simple.df$education=="high.school"]<-"12"
> bank_simple.df$education[bank_simple.df$education=="professional.course"]<-"12a"
> bank_simple.df$education[bank_simple.df$education=="university.degree"]<-"16"
> bank_simple.df$education[bank_simple.df$education=="unknown"]<-"NA"
> count(bank_simple.df$education)
```

```
    x  freq
1   0    18
2  12  9515
3 12a  5243
4  16 12168
5   4  4176
6   6  2292
7   9  6045
8  NA  1731
>
```

6. Following are the mean, median and mode calculations of the ‘age’ field:

```
> mean(bank_simple.df$age)
[1] 40.02406
> median(bank_simple.df$age)
[1] 38
> library(DescTools)
> Mode(bank_simple.df$age)
[1] 31
attr(,"freq")
[1] 1947
```

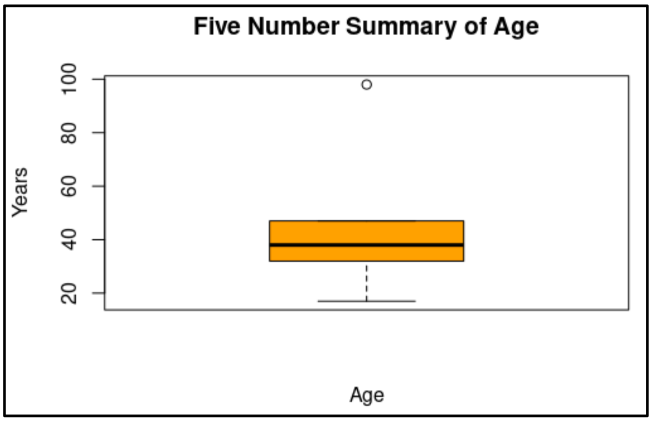
Please note that, ‘mode’ was calculated after using the library (DescTools)

Calculation of five number summary to show the data, and assigning it to a variable.

```
> age_fivenum <- fivenum(bank_simple.df$age)
> age_fivenum
[1] 17 32 38 47 98
```

Boxplot of five number summary of age:

```
> boxplot(age_fivenum,
+         data=age_fivenum,
+         main="Five Number Summary of Age",
+         xlab="Age",
+         ylab="Years",
+         col="orange",
+         border="black"
+ )
```

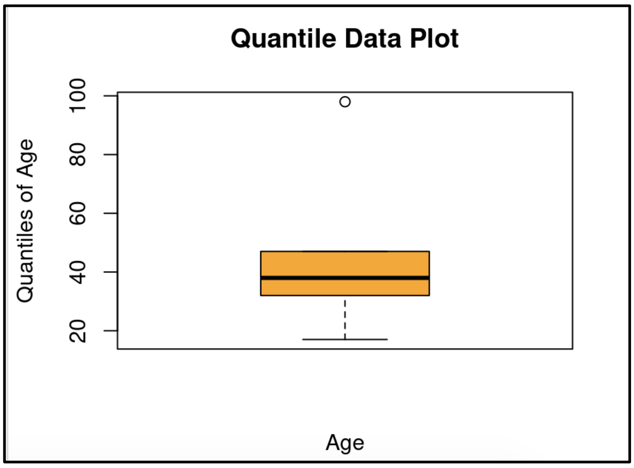


Quantile information:

```
> quantile(bank_simple.df$age)
  0%  25%  50%  75% 100% 
  17   32   38   47   98 
```

Following is the boxplot for quantiles in age.

```
> boxplot(quantile_plot, data=quantile_plot, main="Quantile Data Plot",
xlab="Age", ylab="Quantiles of Age", col="orange", border="black")
```



7. Age variable standardized and saved in “age_z”

```
> age_z <- scale(bank_simple.df$age)
```

8. Detecting outliers in age_z.

```
> age_z <- scale(bank_simple.df$age)
> out <- boxplot.stats(age_z)$out
> out_ind <- which(age_z %in% c(out))
> bank_simple.df[out_ind, ]
```

According to the output, individuals aged 70 and above are considered outliers in this dataset.

```
      age        education pdays previous response
27714  70         basic.4y    NA        0      yes
27758  76 university.degree    NA        0       no
27781  73 university.degree    NA        1       no
27801  88         basic.4y    NA        0       no
27803  88         basic.4y    NA        0      yes
27806  88         basic.4y    NA        0      yes
27809  88         basic.4y    NA        0       no
27811  88         basic.4y    NA        0      yes
27812  88         basic.4y    NA        0      yes
27813  88         basic.4y    NA        0       no
27814  88         basic.4y    NA        0      yes
27815  88         basic.4y    NA        0       no
27816  88         basic.4y    NA        0       no
27817  88         basic.4y    NA        0      yes
27818  88         basic.4y    NA        0      yes
27819  88         basic.4y    NA        0      yes
27827  95         basic.6y    NA        0       no
27838  70         basic.4y    NA        1       no
27839  70         basic.4y    NA        1       no
27845  70         basic.4y    NA        0       no
27852  77          unknown    NA        0      yes
27876  75         basic.9y    NA        0       no
27880  70 university.degree    NA        0       no
27903  70         basic.4y    NA        1       no
27931  73 university.degree    NA        0      yes
27951  80         basic.4y    NA        0       no
27952  80         basic.4y    NA        0       no
27964  80 professional.course    NA        0      yes
28221  72         basic.4y    NA        0       no
28222  72         basic.4y    NA        0       no
28313  82          unknown    NA        0      yes
28457  73         basic.4y    NA        0      yes
28505  71         basic.4y    NA        0       no
28531  70         basic.4y    NA        0      yes
28541  70         basic.4y    NA        1      yes
28587  70         basic.4y    NA        0       no
28620  71      high.school    NA        0       no
28733  70          unknown    NA        0      yes
28774  70          unknown    NA        0       no
29226  71          unknown    NA        0      yes
29264  75         basic.4y    NA        0       no
29499  73         basic.4y     6        1       no
29626  73         basic.4y    NA        0       no
29669  71 university.degree    NA        0      yes
29683  75         basic.4y    NA        0      yes
29974  75         basic.4y    NA        0       no
29978  78         basic.4y    NA        0      yes
29982  75         basic.4y    NA        1      yes
29988  70         basic.6y    NA        0       no
```

29991	78	basic.4y	NA	0	no
30001	75	basic.4y	NA	1	yes
30005	78	basic.4y	NA	0	yes
30007	85	basic.4y	NA	0	yes
30073	85	basic.4y	NA	0	no
30079	85	basic.4y	NA	0	no
30080	80	high.school	NA	3	no
30089	71	university.degree	NA	0	no
30104	85	basic.4y	NA	0	no
30111	85	basic.4y	NA	0	no
30134	79	basic.9y	NA	0	yes
30172	77	basic.4y	NA	1	no
30215	83	basic.4y	NA	0	no
30226	81	professional.course	NA	0	no
30228	71	university.degree	5	1	no
30242	81	professional.course	NA	0	no
30335	73	university.degree	NA	1	no
30336	71	basic.4y	NA	1	no
30391	71	basic.4y	NA	0	no
30431	88	high.school	NA	0	no
30461	81	basic.9y	NA	1	no
30590	81	basic.4y	NA	0	yes
35834	81	unknown	NA	0	yes
35849	71	unknown	NA	1	no
35857	83	basic.9y	3	3	no
35879	75	basic.4y	NA	0	no
35974	78	high.school	3	2	no
36184	88	basic.4y	NA	0	no
36286	77	basic.9y	NA	1	no
36312	72	university.degree	NA	0	no
36384	79	high.school	NA	1	no
36385	79	high.school	NA	1	no
36817	74	basic.4y	NA	0	yes
36999	75	basic.4y	NA	0	no
37137	72	university.degree	NA	0	no
37138	72	university.degree	NA	0	no
37171	70	basic.4y	NA	0	no
37187	79	unknown	NA	0	no
37191	74	high.school	NA	0	no
37193	74	high.school	NA	0	no
37194	74	high.school	NA	1	yes
37196	74	high.school	NA	0	no
37207	76	basic.4y	NA	0	yes
37208	76	basic.4y	NA	0	yes
37214	82	university.degree	NA	2	no
37220	75	basic.4y	NA	0	yes
37228	70	basic.4y	NA	0	yes
37236	73	basic.4y	NA	0	yes
37238	73	basic.9y	15	1	no
37240	73	basic.4y	NA	0	no
37258	73	basic.4y	NA	0	no
37261	76	university.degree	NA	0	yes
37317	70	basic.4y	NA	1	no
37342	85	professional.course	NA	0	yes
37356	80	illiterate	6	1	yes
37372	70	high.school	NA	0	no
37404	74	university.degree	NA	1	yes
37455	74	professional.course	NA	0	no
37456	76	basic.4y	NA	0	no
37473	88	basic.4y	NA	0	no
37480	74	basic.4y	NA	1	no

37494	81	basic.6y	4	1	no
37506	76	basic.6y	NA	1	no
37510	74	professional.course	NA	1	yes
37513	76	university.degree	NA	1	no
37526	73	professional.course	NA	0	no
37533	72	basic.4y	NA	1	no
37546	70	professional.course	NA	1	no
37569	71	basic.4y	NA	0	yes
37571	70	professional.course	NA	0	yes
37587	70	professional.course	NA	0	no
37598	76	professional.course	15	1	yes
37602	72	basic.4y	NA	0	no
37603	73	basic.4y	NA	0	yes
37605	80	high.school	NA	0	yes
37636	74	basic.9y	NA	1	yes
37662	71	basic.4y	NA	0	yes
37676	74	basic.4y	13	1	yes
37680	80	basic.4y	NA	0	no
37691	74	university.degree	NA	0	yes
37693	73	basic.4y	NA	0	no
37716	74	basic.9y	NA	1	no
37717	71	university.degree	6	2	yes
37736	76	basic.4y	NA	0	no
37737	76	basic.4y	NA	1	no
37744	87	basic.4y	NA	0	yes
37757	79	basic.4y	3	2	yes
37766	70	university.degree	NA	1	no
37770	74	basic.4y	6	1	yes
37776	88	university.degree	NA	0	no
37785	81	high.school	NA	0	no
37819	80	basic.4y	NA	0	yes
37820	80	basic.4y	3	2	no
37821	78	basic.4y	NA	0	no
37826	71	university.degree	NA	0	no
37827	71	university.degree	NA	0	no
37862	72	unknown	NA	1	yes
37869	73	university.degree	NA	0	no
37871	73	basic.4y	NA	1	yes
37874	73	high.school	NA	0	no
37906	79	basic.9y	NA	1	yes
37921	72	unknown	NA	0	no
37936	71	basic.4y	NA	0	no
37947	83	unknown	NA	0	no
37952	76	unknown	NA	0	no
37953	76	unknown	NA	0	yes
37955	72	basic.4y	3	2	yes
37959	71	professional.course	NA	0	no
37998	71	basic.4y	NA	0	yes
38000	76	basic.4y	NA	0	no
38006	75	unknown	NA	1	yes
38008	71	basic.4y	NA	0	no
38020	78	unknown	NA	0	no
38021	78	unknown	NA	1	yes
38023	91	university.degree	NA	2	no
38033	91	university.degree	NA	0	no
38034	76	basic.4y	3	1	yes
38046	73	basic.4y	NA	0	no
38053	76	university.degree	NA	1	no
38055	73	basic.4y	NA	0	no
38061	71	unknown	NA	0	yes
38066	83	unknown	NA	0	no

38072	70	basic.4y	3	2	yes
38075	70	basic.4y	NA	2	yes
38082	70	basic.4y	3	1	yes
38089	70	basic.4y	3	1	no
38126	70	high.school	NA	0	no
38128	70	university.degree	NA	0	yes
38130	70	university.degree	NA	0	no
38137	81	basic.4y	NA	0	yes
38145	70	basic.4y	NA	1	no
38146	70	basic.4y	NA	0	no
38167	78	basic.9y	NA	0	no
38170	71	basic.4y	NA	2	no
38176	71	basic.4y	NA	1	yes
38179	75	basic.4y	NA	0	yes
38180	83	professional.course	4	1	yes
38184	71	basic.9y	NA	0	no
38185	82	university.degree	NA	0	yes
38192	82	university.degree	NA	1	no
38193	82	university.degree	NA	0	no
38194	80	basic.4y	NA	0	no
38196	80	basic.4y	NA	0	no
38207	86	basic.4y	NA	0	no
38230	77	unknown	NA	0	no
38242	75	basic.9y	NA	0	no
38247	77	university.degree	NA	0	yes
38248	70	university.degree	NA	1	no
38253	80	basic.4y	NA	1	no
38254	71	university.degree	NA	0	no
38255	71	university.degree	NA	0	no

[reached 'max' / getOption("max.print") -- omitted 269 rows]