Fundamentals of Applied Data Science with R	
	Data Preparation and Data warehouse
	Individual Assignment 1 Report
Name	Ahmed Shehata Mahmoud AboMoustafa
E-Mail	aabom018@uOttawa.ca

Part 1: Data Preparation

1. Import the data set into RStudio and reduce the dataset to only four predictors (age, education, previous, and pdays), and the target, response.

```
# Read Dataset
bank_df <-read.csv("/media/shehata/Data/R_Task/bank-additional-full.csv", header =TRUE, sep =";")
View(bank_df)

# reduce the dataset to only four predictors [age, education, previous, and pdays]
# and the target [response].
reduced_df = subset(bank_df, select = c(age, education, previous, pdays,y))
View(reduced_df)
```

Result after selection of specific columns

	age	education	previous	pdays	y
1	56	basic.4y	0	999	no
2	57	high.school	0	999	no
3	37	high.school	0	999	no
4	40	basic.6y	0	999	no
5	56	high.school	0	999	no
6	45	basic.9y	0	999	no

2. The field pdays is a count of the number of days since the client was last contacted from a previous campaign. The code 999 in the value represents customers who had not been contacted previously. Change the field value 999 to “NA” to represent missing values.

Here, I take a copy of the reduced dataset
Then, I use replace method to replace 999 code with NA

```
reduced_copy<-data.frame(reduced_df)

# change 999 to NA
reduced_copy <- reduced_copy %>% replace(.== 999, NA)
```

Before converting 999 code to NA

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

After Converting 999 code to NA

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

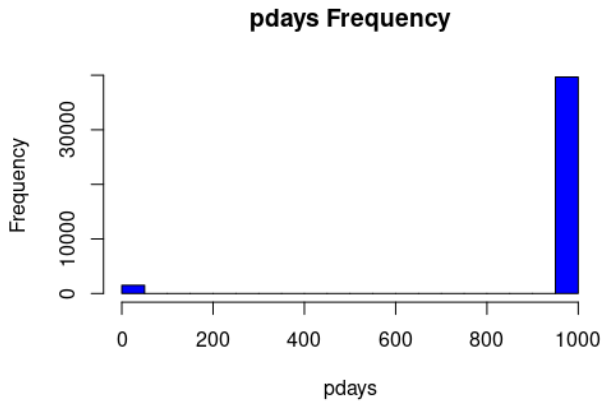
3. Explain why the field pdays is essentially useless until you handle the 999 code

In the pdays column, there are 39673 values of the 999 code, so most of the values are 999 code value which donates to missing values besides that the 999 code represent a missing value, it's from integer type which will affect any transformation we will try to do because there will be a bias toward the 999.
So we need to assign the 999 code value to Na value which donates to null values in R after that the values of age will be distributed correctly.

4. Create a histogram of the pdays variable showing the missing value excluded.

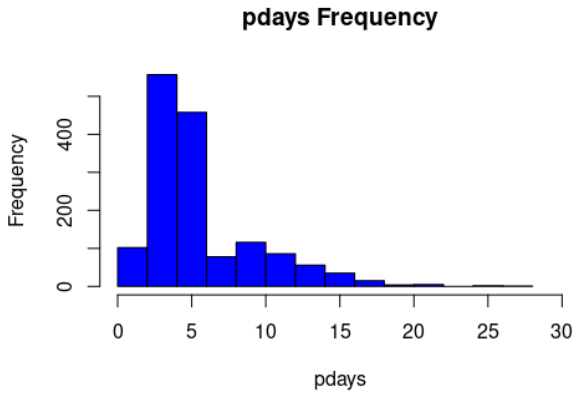
Histogram before exclude 999

```
hist(reduced_df$pdays,
     col= 'blue',
     xlim = c(0,1000),
     main= 'pdays Frequency',
     xlab = 'pdays',
     ylab = 'Frequency')
```



Histogram after exclude 999

```
hist(reduced_copy$pdays,
     col= 'blue',
     xlim = c(0,30),
     main= 'pdays Frequency',
     xlab = 'pdays',
     ylab = 'Frequency')
```



5. Transform the data values of the education field into numeric values.

Her, I use revalue methods and replace parameters to convert each value of education to each corresponding value.

```
# change the values
reduced_copy$education <- revalue(x = reduced_copy$education,
                replace= c("illiterate" = 0,"basic.4y" = 4, "basic.6y" = 6, "basic.9y" = 9,
                           "high.school" = 12, "professional.course" = 14,
                           "university.degree" = 16, "unknown" = NA))

# convert education type to numeric
reduced_copy$education <- as.numeric(reduced_copy$education)
```

This is the output of this step after replacing the values

```
  [1]  1  4  4  2  4  3  6 NA  6  4 NA  4  4  1  2  3  2  2  3  3  4  1  4  4  4  7 NA  3  7  7 NA NA NA  1
 [35]  1  1  4  3  7  1  7  4  6  7  7  6  7  3  7  3  4  1  4  6  1  6  7  4  6  7  7  3  2  2  4  7  3 NA
 [69]  3  1  2  7  7 NA  7  1  7  4 NA  3  1  1  4  7  7  4  2  7  3  4  7 NA NA  3  2  6  7  7  6  2 NA  1
[103] NA  7  7  4  2  4 NA NA  7  1  4  3  7  4  7 NA NA  1  4  4  4  4  7  7  2  6  1  6  2  6 NA  2  7  7
[137]  1  1  4  3  7 NA  7  1  4  1  1  4  7  1  3 NA  4 NA NA  3 NA NA  6  4  7  3  3  3  4  2  4 NA NA  7
[171]  4  4  7  7  4  3  4  2  4  3  4  2  3  7  1  7  2  4  1  7  6  3  7  7  3  1  4  3  3  2  4  2  4 NA
[205]  3  7  7  6  1  7  3  7  7 NA  6  1 NA NA  1  4  4 NA  2  1  3  7  7  7  1  1  6  6  1  4  7  6  1  4
[239]  4  7  4  6  7  3  7 NA  6  4  4  4  6  1  2  6 NA  4  3  3 NA  4  4  2  4  3  7  2  7  7  7  1  4  7
[273]  3  4  4  3  4  1  3  6  2  1  3  3  3  2  3  4 NA  4  7  4  6  1  6  6  4  7  4 NA  4  4  6 NA  4  2
[307] NA  1  7  1  7  7  4  6  7  3  4  2  3  4  3  2 NA  2  6  6  7  4  3  4  4  3  4  7  4  4  6  4  3  7
[341]  6  4  3 NA  1  4  6  3  7  1  2  6  7  4  7  7  4  3  1  1  4  7  4  1  7  7  4  3  3  4  3  3  4  3
[375]  4 NA  2  7  3  7 NA  3  7 NA  1  1  7  2 NA  4  7  4  7  4  3  6  4  1  7  3  4  4  4  7  1  7  2  4
[409]  4  3  4  1  3  1  1  1  3  4  4  7  1  1  1  7  1  3  2  6 NA  4  7  3  7  6 NA  6  3  6  7 NA  6  7
```

6. Compute the mean, median & mode of the age variable. Using a box plot, give the five- number summary of the data. Plot the quantile information.

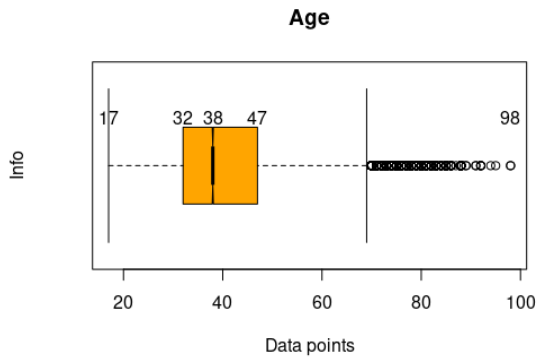
1. I compute mean, median, and mode using summary method

```
#mode
mode_age <- mlv(reduced_copy$age, method=mfv)
mode_age

# summary
summary(reduced_copy$age)

quantile(reduced_copy$age, c(0.25,0.5,0.75))
```

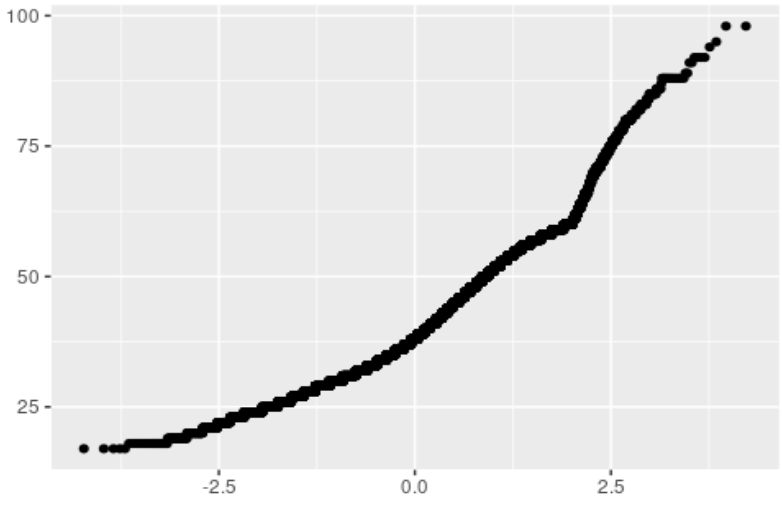
```
> mode_age
[1] 31
> # summary
> summary(reduced_copy$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  17.00   32.00   38.00   40.02   47.00   98.00
> quantile(reduced_copy$age, c(0.25,0.5,0.75))
25% 50% 75%
 32  38  47
```

2. I plot box plot to show outliers in the age

```
boxplot(reduced_copy$age,
        main = "Age",
        xlab = "Data points",
        ylab = "Info",
        col = "orange",
        horizontal = TRUE,
        notch = TRUE,
        axes = TRUE,
        staplewex = 2)

text(x=fivenum(reduced_copy$age), labels =fivenum(reduced_copy$age), y=1.25)
```



3. Her, I Plot the quantile information using qplot method



7. Some machine learning algorithms perform better when the numeric fields are standardized. Standardize the age variable and save it as a new variable, age_z.

Code

```
age_z <- scale(x = reduced_copy$age)
age_z
# add the standard age as a new column to dataframe
reduced_copy$age_z <- age_z
reduced_copy
```

Output

```
age education previous pdays  y        age_z
 56         1        0    NA no  1.533015677
 57         4        0    NA no  1.628973456
 37         4        0    NA no -0.290182119
 40         2        0    NA no -0.002308783
 56         4        0    NA no  1.533015677
 45         3        0    NA no  0.477480111
```

8.Obtain a listing of all records that are outliers according to the field age_z.

Code

```
# list of all outlires in age_z
outlires <- boxplot.stats(reduced_copy$age_z)$out
head(outlires)
```

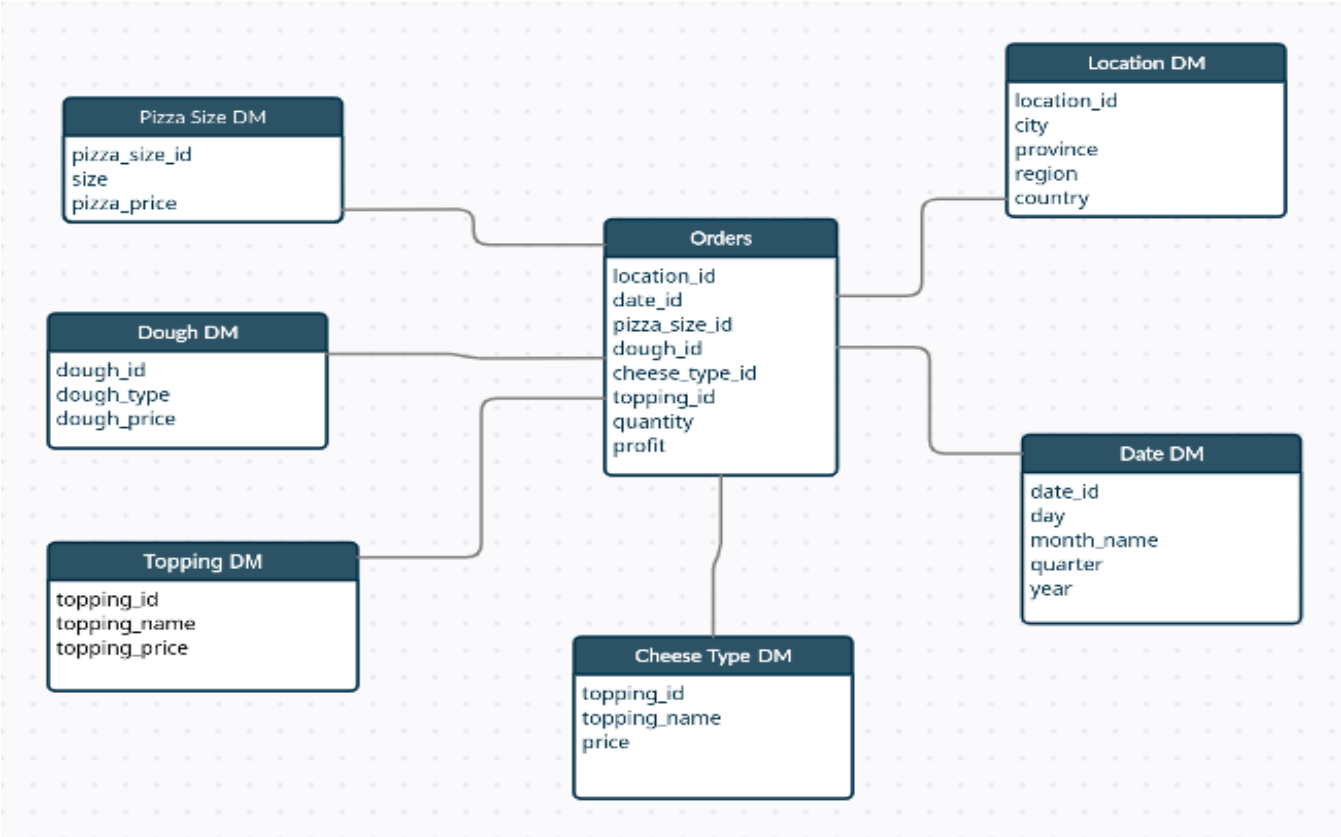
Output

```
 [1] 2.876425 3.452171 3.164298 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665
[12] 4.603665 4.603665 4.603665 4.603665 4.603665 5.275369 2.876425 2.876425 2.876425 3.548129 3.356213
[23] 2.876425 2.876425 3.164298 3.836002 3.836002 3.836002 3.068340 3.068340 4.027918 3.164298 2.972382
[34] 2.876425 2.876425 2.876425 2.972382 2.876425 2.876425 2.972382 3.356213 3.164298 3.164298 2.972382
[45] 3.356213 3.356213 3.644087 3.356213 2.876425 3.644087 3.356213 3.644087 4.315791 4.315791 4.315791
[56] 3.836002 2.972382 4.315791 4.315791 3.740045 3.548129 4.123876 3.931960 2.972382 3.931960 3.164298
[67] 2.972382 2.972382 4.603665 3.931960 3.931960 3.931960 2.972382 4.123876 3.356213 3.644087 4.603665
[78] 3.548129 3.068340 3.740045 3.740045 3.260256 3.356213 3.068340 3.068340 2.876425 3.740045 3.260256
[89] 3.260256 3.260256 3.260256 3.452171 3.452171 4.027918 3.356213 2.876425 3.164298 3.164298 3.164298
[100] 3.164298 3.452171 2.876425 4.315791 3.836002 2.876425 3.260256 3.260256 3.452171 4.603665 3.260256
[111] 3.931960 3.452171 3.260256 3.452171 3.164298 3.068340 2.876425 2.972382 2.876425 2.876425 3.452171
[122] 3.068340 3.164298 3.836002 3.260256 2.972382 3.260256 3.836002 3.260256 3.164298 3.260256 2.972382
[133] 3.452171 3.452171 4.507707 3.740045 2.876425 3.260256 4.603665 3.931960 3.836002 3.836002 3.644087
```

Part 2: Data Warehouse

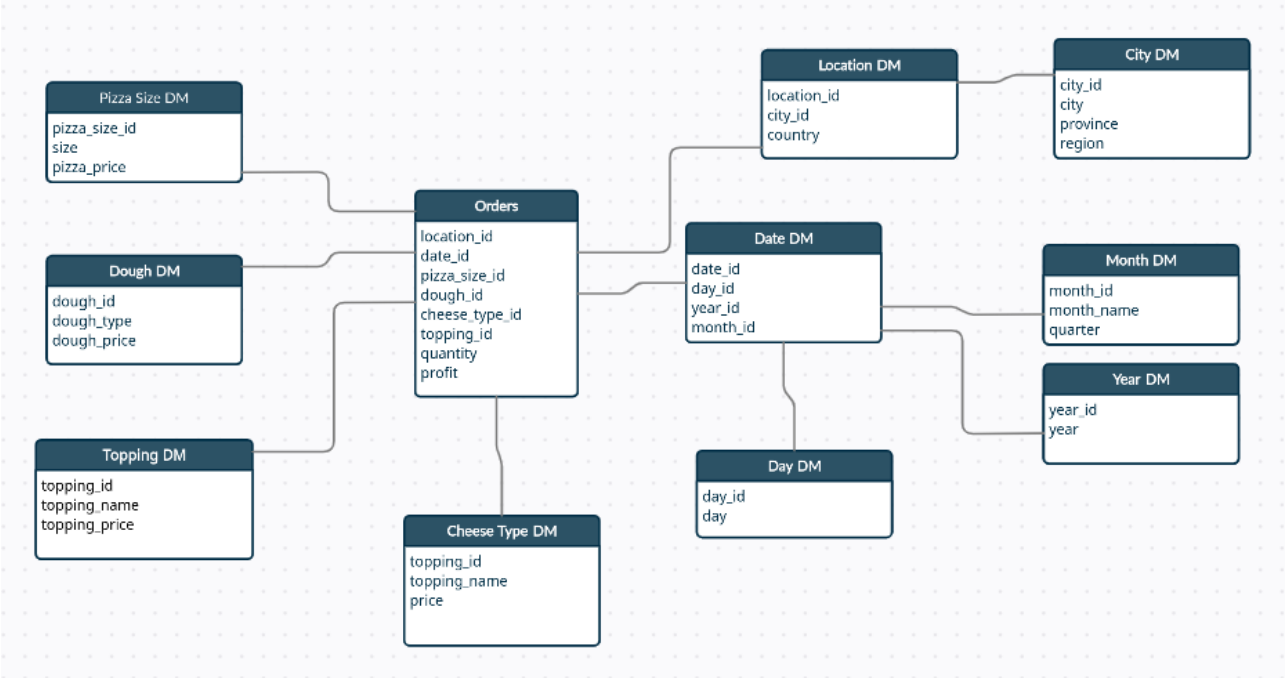
1 a. Sketch a star schema that represents this problem.

Here, I created the star schema, consisting of one fact table and 6 dimensions tables.



1 b. Sketch a snowflake schema that represents this problem.

Here, I create new 3 dimensions from date refers to day, month, and year and 1 new dimension from location table refers to city DM in order to normalize the data.



1 c. Generate a set of sample data stored in csv files for the dimensions and fact table for the snowflake schema in c.

- Here, I create 9 Excel files with the data of columns name in each table in snowflake.
- The data in **date** table generated random in the code using the data in the dimension related to the date table (day, month, and year)

Code

```
# step 2.2
# Function to generate the date table
gen_rondom_date <- function(no_of_data) {
  # Generate transaction data randomly
  date <- expand.grid(1:no_of_data)
  day <- sample(day_table$day_id, no_of_data, replace=T, prob=c(3,2,2,1,4,1,2))
  month <- sample(month_table$month_id, no_of_data, replace=T)
  year <- sample(year_table$year, no_of_data, replace=T, prob=c(3,2,2))

  date_orders <- data.frame(date_id = date,
                            day_id = day,
                            month_id = month,
                            year_id = year)
  return(date_orders)
}
```

Output

	date_id	day_id	month_id	year_id
1	1	30	9	2010
2	2	10	10	2010
3	3	50	3	2010
4	4	30	7	2010
5	5	50	4	2010

- The data in **Orders** table generated random in the code using the data in the dimension related to the Orders table

Code **Output**

```
# step 2.3
# Function to generate order table
gen_rondom_orders <- function(no_of_data) {
  # Generate transaction data randomly
  loc <- sample(location_table$location_id, no_of_data, replace=T, prob=c(3,2,1))
  city <- sample(city_table$city_id, no_of_data, replace=T, prob = c(2,4,1))
  dough <- sample(dough_table$dough_id, no_of_data, replace=T, prob = c(2,4,1))
  cheese <- sample(cheese_table$cheese_id, no_of_data, replace=T, prob=c(3,2,2))
  pizza <- sample(pizza_Size_table$pizza_size_id, no_of_data, replace=T, prob=c(1,2,4,9,12))
  topping <- sample(topping_table$topping_id, no_of_data, replace=T, prob=c(3,2,2,5))
  ddate <- date_fact$date_id
  quantitty <- sample(c(1:10), no_of_data, replace=T, prob = c(1:10))
  profitt <- quantitty * pizza_Size_table[as.factor(pizza),]$pizza_price
  orders <- data.frame(location_id = loc,
                       city_id = city,
                       date_id = ddate,
                       pizza_size_id = pizza,
                       dough_id = dough,
                       cheese_id = cheese,
                       topping_id = topping,
                       quantity = quantitty,
                       revenue = profitt)
  return(orders)
}
```

	location_id	city_id	date_id	pizza_size_id	dough_id	cheese_id	topping_id	quantity	revenue
1	1001	10	1	5	2	3	1	7	490
2	1001	20	2	4	1	2	4	8	400
3	1002	20	3	2	1	1	4	7	210
4	1001	20	4	4	2	1	4	8	400
5	1002	20	5	4	2	1	4	8	400
6	1002	10	6	4	1	1	4	7	350
7	1001	50	7	5	1	2	1	9	630
8	1002	20	8	4	2	1	4	9	450
9	1002	20	9	5	2	1	4	4	280
10	1003	10	10	4	3	3	3	6	300

2. Using R, read the dimensions files and the profit fact table. Build an OLAP cube for your revenue and show the cells of a subset of the cells

Here, I read the Excel files that I created for each dimension table

```
# step 2.1 Read files
library(readxl)

location_table <- read_excel("Dataset/location.xlsx")
city_table <- read_excel("Dataset/city.xlsx")

dough_table <- read_excel("Dataset/dough.xlsx")
cheese_table <- read_excel("Dataset/cheese.xlsx")
topping_table <- read_excel("Dataset/topping.xlsx")
```

Her, I created the OLAP cube using the order table created from the random generation in the code based on revenue

```
# step 2.4
# Build up a cube of revenue based on order_fact and date_fact
order_by_date <- merge(order_fact, date_fact, by.x = "date_id", by.y = "date_id")
order_by_date_month <- merge(order_by_date, month_table, by.x = "month_id", by.y = "month_id")
View(order_by_date_month)

revenue_cube_date <-
  tapply(order_by_date_month$revenue,
         order_by_date_month[,c("location_id", "city_id", "pizza_size_id","quarter",
                                "quantity", "month_name", "year_id")],
         FUN=function(x){return(sum(x))})
```

1. I show subset of the quantity based on month

Code

```
# Quantity and months
apply(revenue_cube_date, c("quantity", "month_name"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

Output

```
          month_name
quantity   Apr   Aug   Dec   Feb   Jan  Jul   Jun  Mar   May   Nov   Oct   Sep
       1     0    50    50    90   140   40    50   70   300   190   500     0
       2  1000   560   820   500   520  160   480    0  1160   700     0   540
       3  1740  1680  1260  1380  1920  870  2160  480  2730  2490  1560  1290
       4  3560  2320  3480  3000  1480  880  2040  440  5040  1720  3680  1480
       5  5350  3050  4350  4250  1700 2000  5000    0  7350  4650  8000  3600
       6  7860  7560  7380  6120  5040 3720  5040 2100 11280  6780  9000  6240
       7 12600  5390  6090  6930  5320 4270 10010 1050 17850  7070 11830 11620
       8 11360  8640  7440 10160  6080 3200 15280 2880 18560 10160 15600  9200
       9 13860 14580 13320 14040 12060 9810 20700 4500 29880 10800 14220 16650
       10 17200 17100 15600 16400 12200 6500 23300 5300 28600 16900 26800 16000
```

2. I show here a subset of the pizza size based on quarter

Code

```
# Pizza size and quarter
apply(revenue_cube_date, c("pizza_size_id", "quarter"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

Output

```
             quarter
pizza_size_id    Q1     Q2    Q3     Q4
            1  1420   3020  1840   2660
            2  4590  12090  5160   9330
            3 11840  26400 17200  21720
            4 34100  80300 43450  64000
            5 74200 159530 91350 114730
```

3. Suppose that we want to examine the data of the above store to find trends, and thus to predict which Pizza components the store should order more of. Describe a series of drill-down and roll-up operations that would lead to the conclusion that customers are beginning to prefer bigger pizzas.

- **First part**
  - I created a data frame from merged tables of pizza, topping, dough, and cheese to merge all columns in one DF.

```
order_by_date <- merge(order_fact, date_fact, by.x = "date_id", by.y = "date_id")
components1 <- merge(order_fact, pizza_Size_table, by.x = "pizza_size_id", by.y = "pizza_size_id" )
components2 <- merge(components1, topping_table, by.x = "topping_id", by.y = "topping_id" )
components3 <- merge(components2, dough_table, by.x = "dough_id", by.y = "dough_id" )
components4 <- merge(components3, cheese_table, by.x = "cheese_id", by.y = "cheese_id" )
View(components4)
```

- Then, I created an OLAP cube (quantity_cube) from the merged tables based on quantity with columns
  - Pizza_size
  - Dough_type
  - Date_id
  - cheese_type
  - topping_name

```
# step 3.1
quantity_cube <-
  tapply(components4$quantity,
         components4[,c("date_id", "size", "dough_type", "cheese_type", "topping_name")],
         FUN=function(x){return(sum(x))})
```

- Then I make an apply function on each column on the cube to sum all the occurrence of all types in each component to calculate which Pizza components the store should order more of

```
# find trends and thus to predict which Pizza components the store should order more of
pizza_size <- apply(quantity_cube, c("size"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
dough <- apply(quantity_cube, c("dough_type"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
cheese <- apply(quantity_cube, c("cheese_type"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
topping <- apply(quantity_cube, c("topping_name"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

- Then the output show that the most Pizza components the store should order more of is :

  dough of type white regular with value 7983

```
Browse[1]> pizza_size
   large   medium personal    small   xlarge
    4437     1929      447     1039     6283
Browse[1]> dough
   stuffed crust    white regular whole wheat thin
            2200             7983             3952
Browse[1]> cheese
   cheddar Mozzarella      Swiss
      3883       3963       6289
Browse[1]> topping
   onions    pepper pepperoni  tomatoes
     2250      2401      5891      3593
```

- **Second Part**
  - To calculate the operations that would lead to the conclusion that customers are beginning to prefer bigger pizzas
  - I make a new cube (revenue_cube2) based on the merged tables of pizza components based on **revenue** in order to calculate the amount of revenue based on each type of pizza
  - Then I make a 2 **Rollup** operations on the (**revenue_cube2 and quantity_cube**) based on pizza size.
    This output show that **xlarge pizza** is the most ordered and profitable type with **439810** in revenue and **6283** in quantity**.**

Code

```
# step 3.2
# Rollup
apply(revenue_cube2, c("size"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})

apply(quantity_cube, c("size"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

Output

```
Browse[1]> apply(revenue_cube2, c("size"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
   large   medium personal    small   xlarge
  221850    77160     8940    31170   439810
Browse[1]> apply(quantity_cube, c("size"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
   large   medium personal    small   xlarge
    4437     1929      447     1039     6283
Browse[1]> map(function(x) rnorm(10, pizza_size))
```

- Then I make a **DrillDown** operation on quantity_cube aggregated by pizza size and quarter in order to reach to the conclusion that customers are beginning to prefer bigger pizzas
- The output here show that the **Xlarge** pizza is the most ordered by customers specially in Q2 in the year

Code

```
# DrillDown
apply(quantity_cube, c("size","quarter"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

Output

```
Browse[1]> apply(quantity_cube, c("size","quarter"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
          quarter
size         Q1   Q2   Q3   Q4
  large     682 1606  869 1280
  medium    296  660  430  543
  personal   71  151   92  133
  small     153  403  172  311
  xlarge   1060 2279 1305 1639
Browse[1]> map(function(x) rnorm(10, pizza_size))
```