Assignment 1 (OLAP Cube)

Abdelmageed Ahmed Abdelmageed Hassan

Fundamental of applied data science

Part 1

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

```
# import the data

bank_data<-read.csv("bank-additional-full.csv", header =TRUE, sep =";")
View(bank_data)
str(bank_data)

# 1- select the predictors
data <- data.frame(age =bank_data$age , education = bank_data$education ,
                   previous = bank_data$previous, pdays = bank_data$pdays ,
                   response = bank_data$y)

View(data)
```

age	education	previous	pdays	response
56	basic.4y	0	999	no
57	high.school	0	999	no
37	high.school	0	999	no
40	basic.6y	0	999	no
56	high.school	0	999	no
45	basic.9y	0	999	no
59	professional.course	0	999	no
41	unknown	0	999	no
24	professional.course	0	999	no
25	high.school	0	999	no
41	unknown	0	999	no
25	high.school	0	999	no
29	high.school	0	999	no
57	basic.4y	0	999	no

```
> str(data)
'data.frame':   41188 obs. of  5 variables:
 $ age      : int  56 57 37 40 56 45 59 41 24 25 ...
 $ education: chr  "basic.4y" "high.school" "high.school" "basic.6y" .
 $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
 $ pdays    : int  999 999 999 999 999 999 999 999 999 999 ...
 $ response : chr  "no" "no" "no" "no" ...
>
```

2 - Change the field value 999 to "NA" to represent missing values.

```
data$pdays[data$pdays == 999] <- 'NA'
View(data)
```

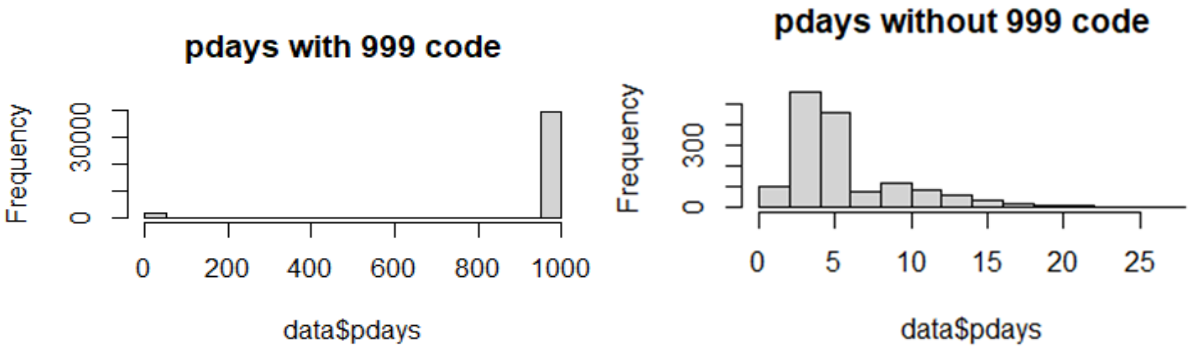
age	education	previous	pdays	response
56	basic.4y	0	NA	no
57	high.school	0	NA	no
37	high.school	0	NA	no
40	basic.6y	0	NA	no
56	high.school	0	NA	no
45	basic.9y	0	NA	no

```
> str(data)
'data.frame':   41188 obs. of  5 variables:
 $ age      : int  56 57 37 40 56 45 59 41 24 25 ...
 $ education: chr  "basic.4y" "high.school" "high.school" "basic.6y" ...
 $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
 $ pdays    : chr  "NA" "NA" "NA" "NA" ...
 $ response : chr  "no" "no" "no" "no" ...
>
```

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

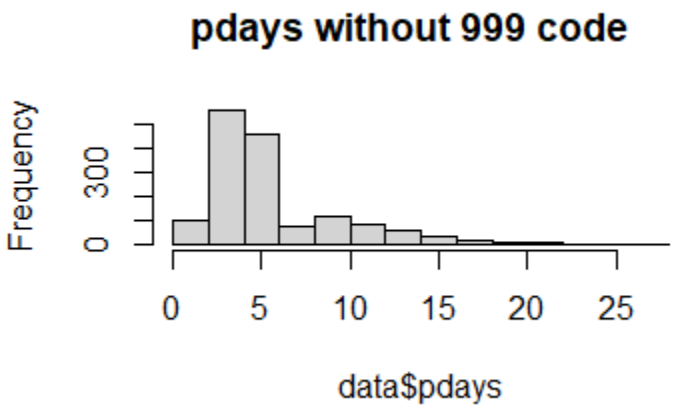
If we keep the code 999 in the data it will be considered as outliers By looking to the histogram before and after consider 999 as NA it indicate that the pdays columns can't give us insights until removing the 999 code .

There are 39673 Na value out of 41178 value so most of pdays values in missing .



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

```
# 4 - Create a histogram of the pdays variable showing the missing value excluded

data$pdays = sapply(data$pdays ,as.numeric)
sapply(data, mode) # show columns type

hist(data$pdays , na.rm = T ,  main = "pdays without 999 code  ")
```



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

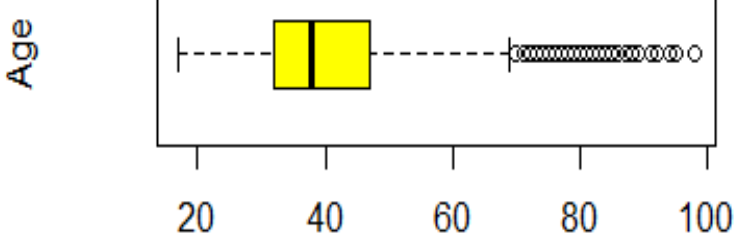
```
#5. Transform the data values of the education field into numeric values

unique(data$education)
# replace education values

data$education[data$education == 'illiterate'] <- 0
data$education[data$education == 'basic.4y'] <- 4
data$education[data$education == 'basic.6y'] <- 6
data$education[data$education == 'basic.9y'] <- 9
data$education[data$education == 'high.school'] <-  12
data$education[data$education == 'professional.course'] <- 14
data$education[data$education == 'university.degree'] <- 16
data$education[data$education == 'unknown'] <- NA
data$education = sapply(data$education ,as.numeric)
sapply(data, mode)
View(data)
#------------------------------------------------------------------------------
```

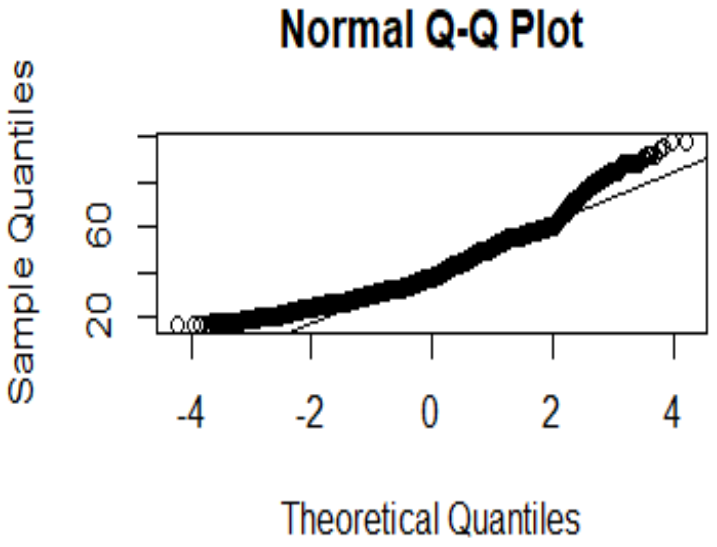
age	education	previous	pdays	response
56	4	0	NA	no
57	12	0	NA	no
37	12	0	NA	no
40	6	0	NA	no
56	12	0	NA	no
45	9	0	NA	no
59	14	0	NA	no

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

```
# 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.

install.packages("modeest")
library(modeest)

mean <- mean(data$age)
print(mean)

median <- median(data$age)
print(median)

mode<- mfv(data$age)
print(mode)

boxplot(data$age,
        main = 'boxplot for age column ',
        col = 'yellow',
        ylab = 'Age',
        horizontal = TRUE )

summary(data$age)
qqnorm(data$age)
qqline(data$age , datax = FALSE , distribution =qnorm , probs = c(0.25,0.75))

#-------------------------------------------------------------------------------
```

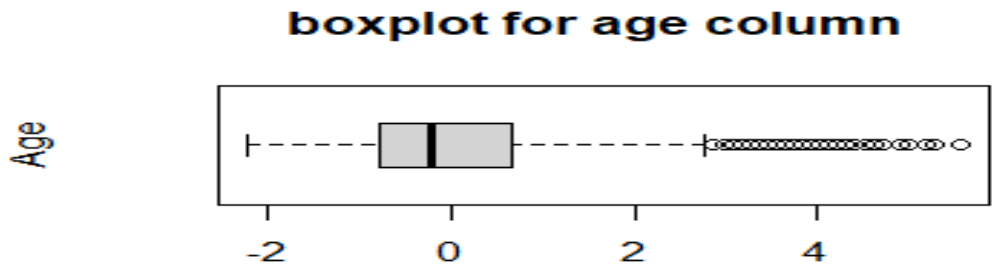
```
> print(mean)
[1] 40.02406
> print(median)
[1] 38
> print(mode)
[1] 31
> summary(data$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  17.00   32.00   38.00   40.02   47.00   98.00
```





7 - Standardize the age variable and save it as a new variable, age_z.

```
#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
data$age_z <- scale(data$age)

boxplot(data$age_z,
        main = 'boxplot for age column ',
        ylab = 'Age',
        horizontal = TRUE )
```




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

outliers <- boxplot.stats(data$age_z)$out
print(outliers)
```

```
> length(outliers)
[1] 469
> |
```

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

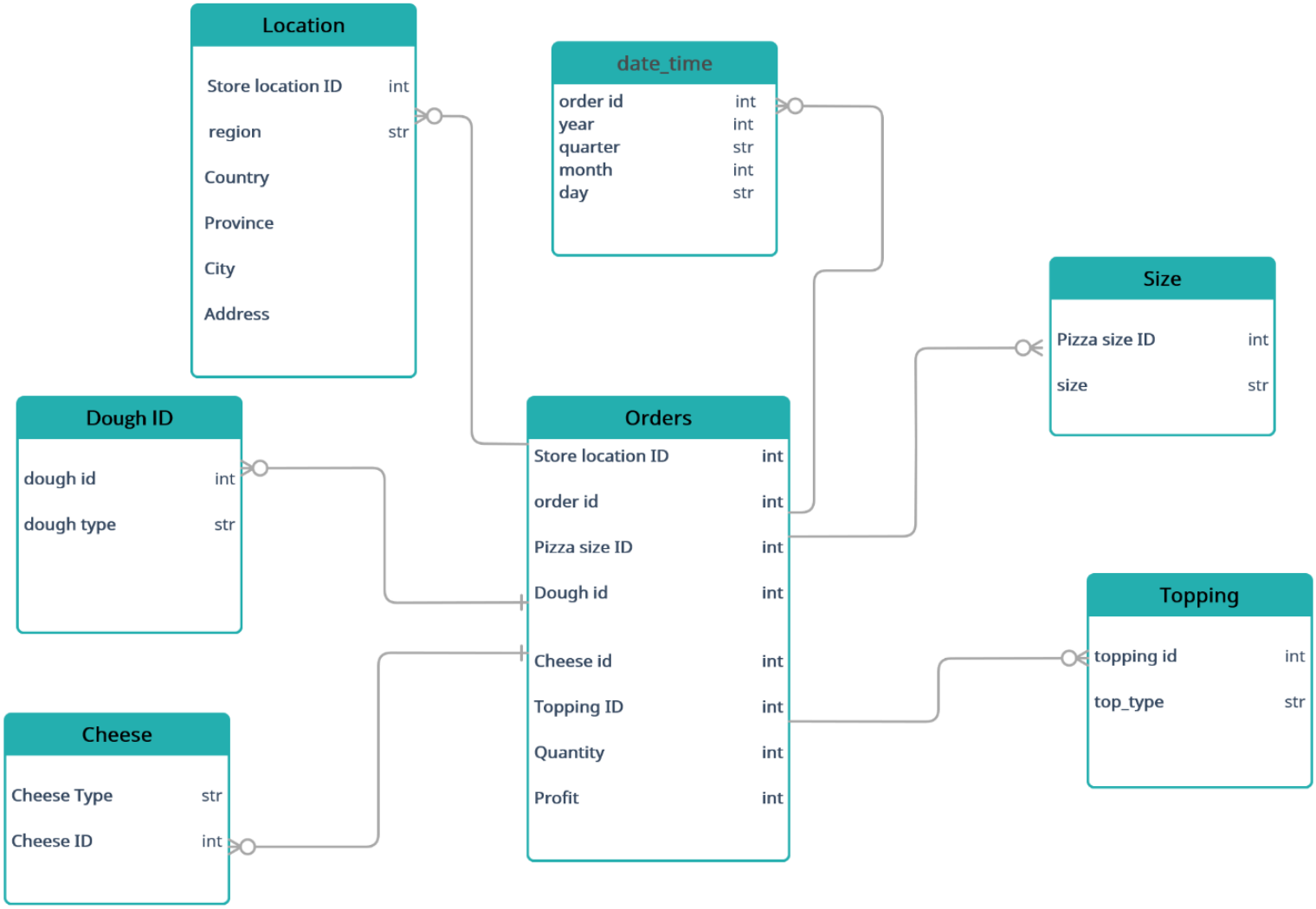
```
  [1] 2.876425 3.452171 3.164298 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665 4.603665
 [15] 4.603665 4.603665 5.275369 2.876425 2.876425 2.876425 3.548129 3.356213 2.876425 2.876425 3.164298 3.836002 3.836002 3.836002
 [29] 3.068340 3.068340 4.027918 3.164298 2.972382 2.876425 2.876425 2.876425 2.972382 2.876425 2.876425 2.972382 3.356213 3.164298
 [43] 3.164298 2.972382 3.356213 3.356213 3.644087 3.356213 2.876425 3.644087 3.356213 3.644087 4.315791 4.315791 4.315791 3.836002
 [57] 2.972382 4.315791 4.315791 3.740045 3.548129 4.123876 3.931960 2.972382 3.931960 3.164298 2.972382 2.972382 4.603665 3.931960
 [71] 3.931960 3.931960 2.972382 4.123876 3.356213 3.644087 4.603665 3.548129 3.068340 3.740045 3.740045 3.260256 3.356213 3.068340
 [85] 3.068340 2.876425 3.740045 3.260256 3.260256 3.260256 3.260256 3.452171 3.452171 4.027918 3.356213 2.876425 3.164298 3.164298
 [99] 3.164298 3.164298 3.452171 2.876425 4.315791 3.836002 2.876425 3.260256 3.260256 3.452171 4.603665 3.260256 3.931960 3.452171
[113] 3.260256 3.452171 3.164298 3.068340 2.876425 2.972382 2.876425 2.876425 3.452171 3.068340 3.164298 3.836002 3.260256 2.972382
[127] 3.260256 3.836002 3.260256 3.164298 3.260256 2.972382 3.452171 3.452171 4.507707 3.740045 2.876425 3.260256 4.603665 3.931960
[141] 3.836002 3.836002 3.644087 2.972382 2.972382 3.068340 3.164298 3.164298 3.164298 3.740045 3.068340 2.972382 4.123876 3.452171
[155] 3.452171 3.068340 2.972382 2.972382 3.452171 3.356213 2.972382 3.644087 3.644087 4.891538 4.891538 3.452171 3.164298 3.452171
[169] 3.164298 2.972382 4.123876 2.876425 2.876425 2.876425 2.876425 2.876425 2.876425 2.876425 3.931960 2.876425 2.876425 3.644087
[183] 2.972382 2.972382 3.356213 4.123876 2.972382 4.027918 4.027918 4.027918 3.836002 3.836002 4.411749 3.548129 3.356213 3.548129
[197] 2.876425 3.836002 2.972382 2.972382 2.972382 3.836002 3.836002 2.972382 2.972382 3.260256 3.260256 3.260256 4.123876 4.315791
[211] 2.876425 2.876425 2.876425 4.603665 4.603665 2.972382 5.563242 5.563242 3.164298 3.068340 2.972382 2.972382 3.931960 2.972382
[225] 3.356213 3.931960 3.164298 3.836002 3.356213 3.068340 4.027918 4.027918 3.644087 3.644087 3.068340 3.644087 2.972382 3.740045
[239] 2.972382 3.164298 3.931960 3.931960 3.452171 2.972382 2.972382 3.740045 3.740045 3.068340 3.068340 3.068340 3.068340 3.644087
[253] 3.836002 3.548129 3.548129 3.644087 3.356213 3.068340 3.548129 3.068340 4.027918 4.123876 4.123876 5.179411 3.644087 4.411749
[267] 3.644087 3.068340 3.548129 3.164298 3.931960 3.452171 4.123876 4.123876 3.836002 3.164298 4.411749 2.876425 2.972382 3.931960
[281] 3.260256 4.315791 4.219833 4.315791 2.876425 2.876425 3.452171 3.452171 3.452171 3.452171 3.452171 3.164298 3.164298 3.644087
[295] 3.068340 4.315791 3.260256 3.164298 3.836002 3.164298 4.027918 3.356213 2.876425 3.356213 3.836002 3.644087 3.644087 4.411749
[309] 4.219833 3.452171 4.027918 3.356213 2.972382 2.972382 3.836002 3.260256 4.027918 4.027918 3.260256 3.260256 3.260256 4.315791
[323] 4.315791 4.315791 3.356213 4.315791 2.876425 3.836002 3.836002 3.836002 3.836002 3.836002 3.548129 3.836002 3.164298 2.972382
[337] 2.972382 2.972382 3.740045 3.740045 3.068340 3.644087 4.027918 3.548129 3.548129 4.987496 2.876425 3.452171 3.548129 2.972382
[351] 3.260256 3.836002 3.740045 3.452171 3.548129 4.987496 4.219833 2.876425 3.740045 3.260256 3.068340 3.068340 3.452171 3.452171
[365] 4.699622 3.644087 3.644087 2.876425 3.260256 4.123876 3.931960 3.452171 3.452171 4.411749 4.123876 4.123876 3.356213 3.164298
[379] 3.068340 3.068340 3.068340 3.740045 3.164298 4.123876 4.123876 3.164298 3.644087 3.644087 3.836002 2.876425 3.644087 3.068340
[393] 3.068340 3.644087 3.644087 4.027918 3.068340 3.068340 2.972382 4.219833 3.548129 3.260256 3.452171 3.836002 2.972382 2.972382
[407] 3.931960 3.548129 3.356213 2.972382 3.548129 2.972382 3.068340 3.548129 3.164298 3.164298 4.987496 2.876425 3.548129 4.987496
[421] 3.356213 2.972382 2.876425 3.452171 3.452171 3.068340 3.452171 3.068340 2.972382 2.876425 3.931960 2.972382 3.644087 3.356213
[435] 4.603665 3.452171 4.603665 4.603665 4.315791 4.699622 4.411749 3.836002 4.123876 4.123876 4.027918 2.972382 4.219833 4.027918
[449] 4.411749 4.411749 4.027918 3.548129 3.356213 2.876425 3.260256 2.972382 2.972382 3.931960 2.876425 3.931960 3.260256 4.219833
[463] 3.260256 3.260256 4.219833 3.931960 3.836002 3.164298 3.260256
> |
```

Part 2

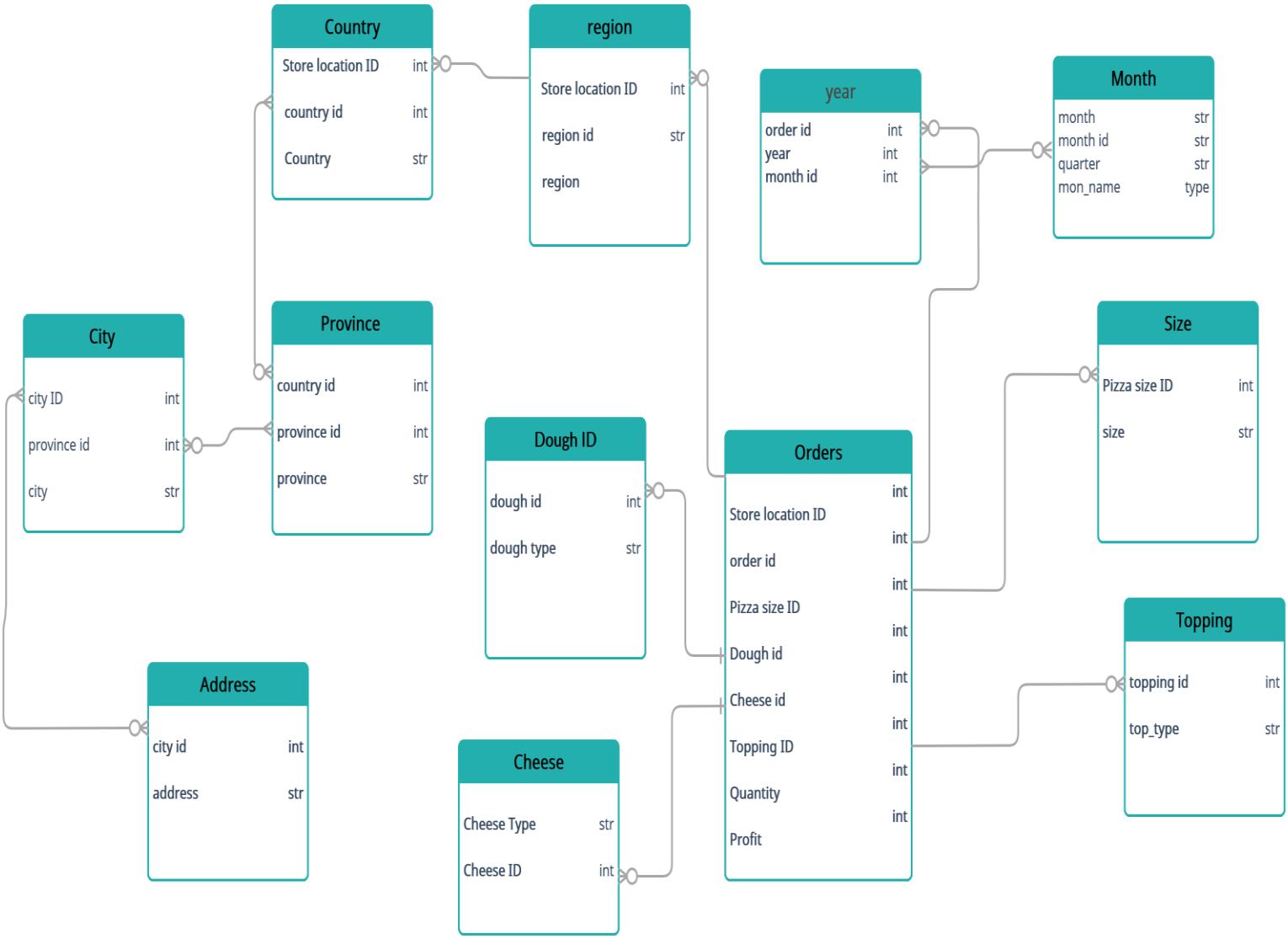
all csv file , schemas and source code here

1 – Sketch Star and Snow Schema

1.1 – Star Schema



1.2 – Snowflake Schema



Import csv file

```
# Part 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

library(readr)

Cheese<-read.csv("cheese.csv")

City<-read.csv("City.csv", header =TRUE, sep =",")

country<-read.csv("country.csv", header =TRUE, sep =",")

year<-read.csv("year.csv",)

Dough<-read.csv("Dough.csv", header =TRUE, sep =",")

Province<-read.csv("Province.csv", header =TRUE, sep =",")

Region<-read.csv("Region.csv", header =TRUE, sep =",")

size<-read.csv("size.csv", header =TRUE, sep =",")

Topping<-read.csv("Topping.csv", header =TRUE, sep =",")

Address<-read.csv("Address.csv", header =TRUE, sep =",")

month<-read.csv("month.csv", header =TRUE, sep =",")
.
```

Create Function to Generate Fact Table

```
# Function to generate the Sales table
gen_orders <- function(n) {

  # Generate transaction data randomly
  city_id <- sample(City$city_id, n, replace=T, prob=c(7,10,9,5,4,2,1,6,3,1,2,3,1,2,3,4,2,3,2))
  Province_ID <- sample(Province$province_id , n ,replace = T ,prob = c(8,5,7,3,2,1,6,3))
  Country_ID <- sample(country$country_id , n ,replace = T ,prob = c(8,6,4,3,2))
  REGION_ID <- sample(region$store_loc_id , n ,replace = T ,prob = c(8,6,3))
  year_id <- sample(year$year_id , n, replace=T ,prob = c(3 , 4,1 ))
  Month_id <- sample(month$month_id, n, replace=T)
  p_size <- sample(size$size_id ,n , replace = T )
  Dough_id <- sample(Dough$dough_id, n, replace=T, prob=c(1, 3, 2))
  cheese_id <- sample(Cheese$cheese_id, n, replace=T, prob=c(5,1, 3))
  topping_id <- sample(Topping$topping_id, n , replace = T ,prob = c(5 ,2 ,1,4))

  quantity <- sample(x= c(1,2,3,4) , n ,replace =T ,prob= c(10,6,4,2))

  orders <- data.frame(MONTH_ID=Month_id,
                       YEAR_ID=year_id,
                       CITY_ID = city_id ,
                       Province_ID = Province_ID ,
                       Country_ID = Country_ID ,
                       REGION_ID = REGION_ID ,
                       P_SIZE= p_size,
                       DOUGH_ID=Dough_id,
                       CHEESE_ID=cheese_id,
                       TOPPING_ID = topping_id,
                       QUANTITY = quantity )

  # Sort the records by time order
  orders <- orders[order(orders$YEAR_ID, orders$MONTH_ID),]
  row.names(orders) <- NULL
  return(orders)
}
# Now create the sales fact table
orders_fact <- gen_orders(1000)
```

Merge (Join) dimensions table to Fact table to create Pizza dataframe

based on the Snowflake schema we will join fact table (orders_fact) with dimensions tables and construct the dataframe that will be used in building OLAP Cube

```
#*********************************merge data set ************************************
library(dplyr)
# 1 - join orders_fact table with Topping table in a new table (profit table)
profit_fact <- left_join( orders_fact,Topping,by = c( "TOPPING_ID" = "topping_id"))
# 2 - join with year table by year id
profit_fact <- left_join( profit_fact,year,by = c( "YEAR_ID"="year_id"))
# 3- join with month table by month id
profit_fact <- left_join( profit_fact,month,by = c( "MONTH_ID"="month_id"))
# 4- join with size table by p_size and size_id
profit_fact <- left_join( profit_fact,size,by = c( "P_SIZE"="size_id"))
# 5 - join with dough table by dough_id
profit_fact <- left_join( profit_fact,Dough,by = c( "DOUGH_ID"="dough_id"))
# 6 - join with cheese table by cheese_id
profit_fact <- left_join( profit_fact,Cheese,by = c( "CHEESE_ID"="cheese_id"))

# nested join
# join the address ,city  ,province ,country and region on Location_table
Location_table <- left_join(City ,Address , by= c("city_id" = "city_id"))
Location_table <- left_join(Location_table ,Province , by= c("province_id" = "province_id"))
Location_table <- left_join(Location_table ,country , by= c("country_id" = "country_id"))
Location_table <- left_join(Location_table ,region , by= c("region_id" = "store_loc_id"))

# join Location_table with profit_fact to construct Pizza data Frame
Pizza_df <- left_join(profit_fact ,Location_table , by= c("REGION_ID" = "region_id"))
Pizza_df$profit <- Pizza_df$QUANTITY * Pizza_df$size_price

View(Pizza_df)

#*****************************************************************************************
#*****************************************************************************************
```

Building a Cube of profit and show subset of cells

```
# Build up a cube
profit_cube <-
  tapply(Pizza_df$profit,
         Pizza_df[,c("size_type", "year","month","top_type" , "dough_type" , "cheese_type"  )],
         FUN=function(x){return(sum(x))})

profit_cube
dimnames(profit_cube)
```

Showing the dimensions names of the cube

```
> dimnames(profit_cube)
$size_type
[1] "large"    "medium"   "personal" "small"    "xlarge"

$year
[1] "2018" "2019" "2020"

$month
 [1] "apr" "aug" "dec" "feb" "jan" "jul" "jun" "mar" "may" "nov" "oct" "sep"

$top_type
[1] "onions"   "pepper"   "pepperon" "tomatoes"

$dough_type
[1] "stuffed crust"    "white regular"    "whole wheat thin"

$cheese_type
[1] "cheddar"    "Mozzarella" "Swiss"

> |
```

Showing Subset of Cells

```
> profit_cube[ "large",  , , "tomatoes" , "stuffed crust" , "Mozzarella" ]
      month
year   apr aug dec feb jan jul jun mar may nov oct sep
  2018  NA  NA  NA 286  NA  NA  NA  NA  NA  NA  NA  NA
  2019  NA  NA  NA  NA  NA  NA  NA  NA  NA  52  NA 312
  2020  NA 312  65  NA 156  NA  NA  NA  NA  NA  NA  NA
> profit_cube[ "large",  , , "pepperon" , "stuffed crust" , "Mozzarella"]
      month
year   apr aug dec feb jan jul jun mar may nov oct sep
  2018  NA  NA  NA  NA  NA 195  NA  NA  NA  NA  NA 156
  2019  NA  NA  NA  NA 468  NA  NA  NA  NA  NA  NA  NA
  2020  NA  NA  NA 312 156 156 130  NA  NA  NA  65  NA
> profit_cube[ "large",  , , "pepper"   , "stuffed crust" , "Mozzarella" ]
      month
year   apr aug dec feb jan jul jun mar may nov oct sep
  2018  NA  NA 624  NA  NA  NA  NA  NA  NA  NA  NA  NA
  2019  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  2020  NA  NA  NA  NA  NA  NA  NA  NA 624  NA 468  NA
> profit_cube[ "large",  , , "onions"   , "white regular" , "Mozzarella" ]
      month
year   apr aug dec feb jan jul jun mar may nov oct sep
  2018  NA  NA  NA  NA  NA  NA  NA  NA 156  NA  NA  NA
  2019  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  2020  NA  NA  NA 468  NA  NA  NA  NA  NA  NA  NA  NA
```

Exploratory Data Analysis

1- Profit analysis based on Topping Type

1.1 Roll Up

By rolling up through the Profit cube for Topping Type and year dimensions we found that the "tomatoes" Topping is most preferred on followed by "pepperon" topping

```
> apply(profit_cube, c("year", "top_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
      top_type
year   onions pepper pepperon tomatoes
  2018   3585   4313    10014    12285
  2019   1894   9363    13202    18057
  2020   6090  12407    28608    26996
> 
```

1.2 Drill Down

By drilling down through the profit cube we can found the profit by each Topping for each month per year

```
> apply(profit_cube, c("year", "month", "top_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , top_type = onions

      month
year   apr aug dec  feb jan jul  jun mar may nov oct sep
  2018 426 360 260  300 215 180    0 575 176 490  75 528
  2019 184  60 695  449   0   0    0 150 120   0  60 176
  2020 465 995  60 1036 686  60 1172 384 228 498 311 195

, , top_type = pepper

      month
year   apr  aug  dec  feb  jan  jul  jun  mar  may  nov oct sep
  2018  50    0 1352  120  560   75  627  320  172  324 205 508
  2019 469 1149  678  624  661   60 1407 1058  460 1602 310 885
  2020 726 1213 1564 1250 1508 1535  899   50 1332 1302 608 420

, , top_type = pepperon

      month
year   apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018 1403  329 1251  633  720 1031 1238  110 1479  567  378  875
  2019 1334  615  733 2180 1649 1645  815 1488 1126  290  696  631
  2020 1123 1911 1652 4991 2708 3019 1764 4486 2412 1847 1663 1032

, , top_type = tomatoes

      month
year   apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018  205 1124 1023 1208 1624 1949  825  215  542  602 1024 1944
  2019 2746 1205 1727  594 1576 1077 1381  955 1331 2161 2151 1153
  2020 1569 2148 2472 2220 3176 1731 2351 1940 3285 1588 3220 1296
```

2- ***Profit analysis based on Cheese Type***

2.1 ***Roll Up***

By rolling up through the Profit cube for Cheese Type and year dimensions we found that the "Swiss cheese" is most preferred one and achieve highest revenue

```
> apply(profit_cube, c("year", "cheese_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
      cheese_type
year   cheddar Mozzarella Swiss
  2018    4771       8420 17006
  2019    7266      15133 20117
  2020    9022      24227 40852
>
```

2.2 ***Drill Down***

By drilling down through the profit cube we can found the profit by each Cheese for each month per year.

```
> apply(profit_cube, c("year", "month", "cheese_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , cheese_type = cheddar

      month
year    apr  aug dec  feb jan jul jun  mar may nov  oct sep
  2018  360  438 628  396 642 776 223  130 305 769    0 104
  2019 1374  672 838  286 644 115 221  732 746 342 1043 253
  2020  446 1544 627 1453 631 586 472 1303 824 497  398 241

, , cheese_type = Mozzarella

      month
year    apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018  643   50 1029  470  740 1220  390  235  699  314  749 1881
  2019 1422  852  814  777 2104  403 1449  976 1113 2651  741 1831
  2020 1300 1938 1159 3360 3163 1125 2564 2605 2515 1710 1943  845

, , cheese_type = Swiss

      month
year    apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018 1081 1325 2229 1395 1737 1239 2077  855 1365  900  933 1870
  2019 1937 1505 2181 2784 1138 2264 1933 1943 1178 1060 1433  761
  2020 2137 2785 3962 4684 4284 4634 3150 2952 3918 3028 3461 1857
```

3- ***Profit analysis based on Dough Type***

3.1 Roll Up

By rolling up through the Profit cube for Dough Type and year dimensions we found that the "White regular" dough is most preferred one and achieves highest revenue

```
> apply(profit_cube, c("year", "dough_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
      dough_type
year   stuffed crust white regular whole wheat thin
  2018          10752         14964                4481
  2019          10535         24609                7372
  2020          25503         37736               10862
```

3.2 Drill Down

By drilling down through the profit cube we can found the profit by each Dough for each month per year.

```
> apply(profit_cube, c("year", "month", "dough_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , dough_type = stuffed crust

      month
year    apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018  664 1037 1482 1023  925 1681  370  585  453   84  600 1848
  2019 2002  416  446 1284 1632  465 1446  415  201  592  974  662
  2020 1007 1282 1169 5223 2464 1800 2598 1814 3224 1344 2476 1102

, , dough_type = white regular

      month
year    apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018  596  614 1734 1198 2119 1323 1414  405 1674 1194  866 1827
  2019 2141 1893 2265 2345 1614 1800 1192 2595 2349 2947 1925 1543
  2020 2136 3979 3671 2984 4942 3285 2677 4589 3502 2637 2073 1261

, , dough_type = whole wheat thin

      month
year    apr  aug  dec  feb jan  jul jun mar may  nov  oct sep
  2018 824  162  670   40  75  231 906 230 242  705  216 180
  2019 590  720 1122  218 640  517 965 641 487  514  318 640
  2020 740 1006  908 1290 672 1260 911 457 531 1254 1253 580
```

4- Profit analysis based on Pizza Size

4.1 – Roll UP

Reducing the dimension by Roll up to get the general overview about the impact of the Pizza size in profit which indicates that the demands of large Pizza is increasing .

```
apply (profit_cube, c ("year", "size_type"), FUN=function(x) {return (sum (x, na.rm=TRUE))})
```

```
> apply(profit_cube, c("year", "size_type"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
     size_type
year   large medium personal small xlarge
  2018  9295   5740     3220  3857   8085
  2019 12142   6710     4810  7154  11700
  2020 21645  10900     6440 12271  22845
>
```

We notice that the large and xlarge sizes make the higher revenue than other types for last three years in an increasing manner which means that customers are beginning to prefer bigger pizzas.

4.2 Drill down

Drilling down to get more details for each Pizza size sales' revenues by year and month

```
apply (profit_cube, c ("year", "month", "size_type"), FUN=function(x) {return (sum (x, na.rm=TRUE))})
```

```
, , size_type = large

      month
year     apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct sep
  2018   221  390 1703  975 1079 1521 1235  260  533    0  494 884
  2019 1040 1495  299  754  962 1313 2379  806  533 1352  377 832
  2020 1274 2132 1417 2847 1989 1924 1872 3445 2041  403 1521 780

, , size_type = medium

      month
year     apr aug  dec  feb  jan jul jun  mar may  nov oct  sep
  2018   680  50  150  400  600 370 600  580 390  540  70 1310
  2019 1180 270  280  670  340 290 120 1250 530  290 650  840
  2020  650 440 1050 1450 1090 740 960 1110 670 1390 490  860

, , size_type = personal

      month
year    apr aug dec feb jan jul jun mar may nov oct sep
  2018 120 195 335 300 540 290 195   0  60 505 400 280
  2019 600 450 300 860 340 160 340 230 310 530 645  45
  2020 160 950 375 715 335 495 780 230 710 665 615 410

, , size_type = small

      month
year    apr  aug  dec  feb  jan jul  jun  mar  may nov oct sep
  2018 553   98  378  406    0 154  630   35  441 728 238 196
  2019 938  154  539  693  714 539  434 1050  329 756 735 273
  2020 644 1050 1736 1365 1379 546 1134  560 1841 917 896 203

, , size_type = xlarge

      month
year     apr  aug  dec  feb  jan  jul  jun  mar  may  nov  oct  sep
  2018   510 1080 1320  180  900  900   30  345  945  210  480 1185
  2019   975  660 2415  870 1530  480  330  315 1335 1125  810  855
  2020 1155 1695 1170 3120 3285 2640 1440 1515 1995 1860 2280  690
```