Professional Master's in Artificial Intelligence
Fundamentals for Applied Data Science (DTI5126)

Subject: Assignment 1 (Data Preparation & Data Warehousing)

By

Mohamed Sayed Abdelwahab Hussein

(300273145)

Mhuss073@uottawa.ca

Under Supervision
Dr. Olubisi Runsewe

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 the data set
bank_df <- read_delim("bank-additional-full.csv", delim = ";")

#select specific columns from the original data frame
df <- bank_df[,c("age", "education", "previous", "pdays", "y")]
View(df)
```

Assignment 1.R × | df ×

Filter

	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
7	59	professional.course	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.

```
#change the value of 999 to NA as a missing values
df$pdays = na_if(df$pdays, 999)
```

Assignment 1.R × | df ×

Filter

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

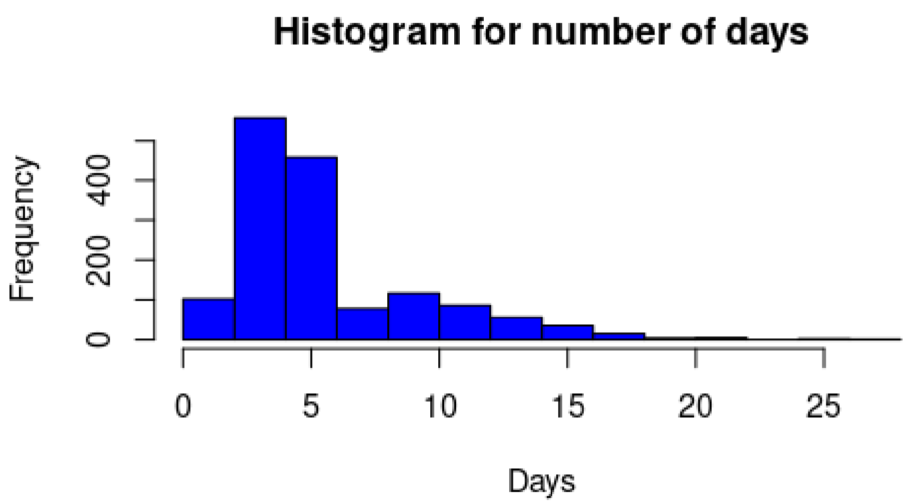
3. Explain why the field pdays is essentially useless until you handle the 999 code.

sum(is.na(df$pdays))

it has a lot of missing values (39673), so we can`t work with this column.

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

```
#plot histogram of pdays
hist(df$pdays, main="Histogram for number of days",
     xlab="Days", border="black", col="blue", breaks=10)
```



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

```
#Transform the data values of the education field into numeric values
df$education <- revalue(df$education, replace= c("illiterate" = 0,
                                                 "basic.4y" = 4,
                                                 "basic.6y" = 6,
                                                 "basic.9y" = 9,
                                                 "high.school" = 12,
                                                 "professional.course" = 14,
                                                 "university.degree" = 16,
                                                 "unknown" = "Missing" ))

#change the value of Missing to NA as a missing values
df$education = na_if(df$education, "Missing")
unique(df$education)
```

```
> unique(df$education)
[1] "4"  "12" "6"  "9"  "14" NA   "16" "0"
> ""
```

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

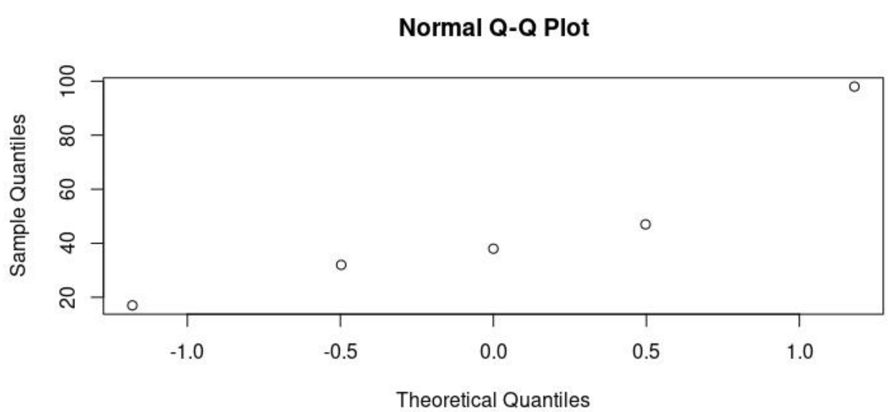
```
#mean of age
mean(df$age)
#median of age
median(df$age)
#function to calac the mode
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
#mode of age
getmode(df$age)
```

- Mean = 40.02406
- Median = 38
- Mode = 31

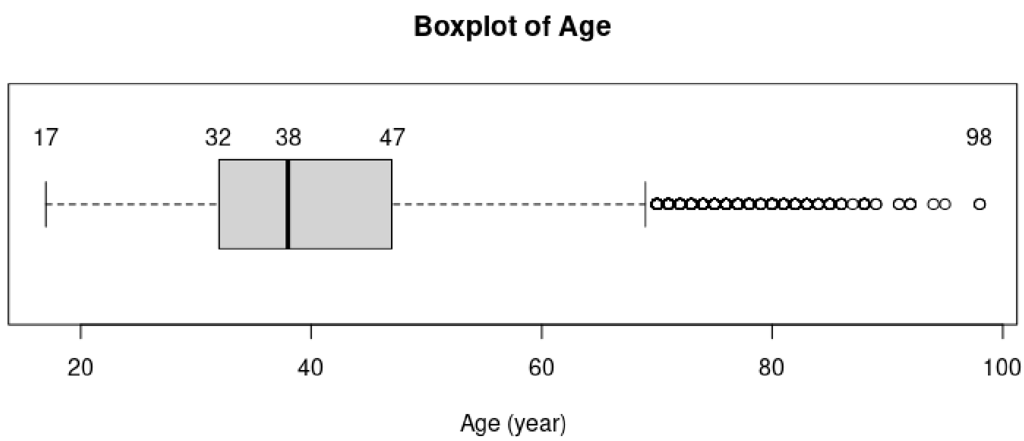
```
> #mean of age
> mean(df$age)
[1] 40.02406
> #median of age
> median(df$age)
[1] 38
> #function to calac the mode
> getmode <- function(v) {
+   uniqv <- unique(v)
+   uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> #mode of age
> getmode(df$age)
[1] 31
```

```
> summary(df$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  17.00   32.00   38.00   40.02   47.00   98.00 
```

```
#Q-Q Plot of age (quantile)
qqnorm(quantile(df$age))
```



```
#Boxplot
boxplot(df$age, main="Boxplot of Age", xlab="Age (year)", horizontal = TRUE)
text(x=fivenum(df$age),labels=fivenum(df$age),y=1.3)
```



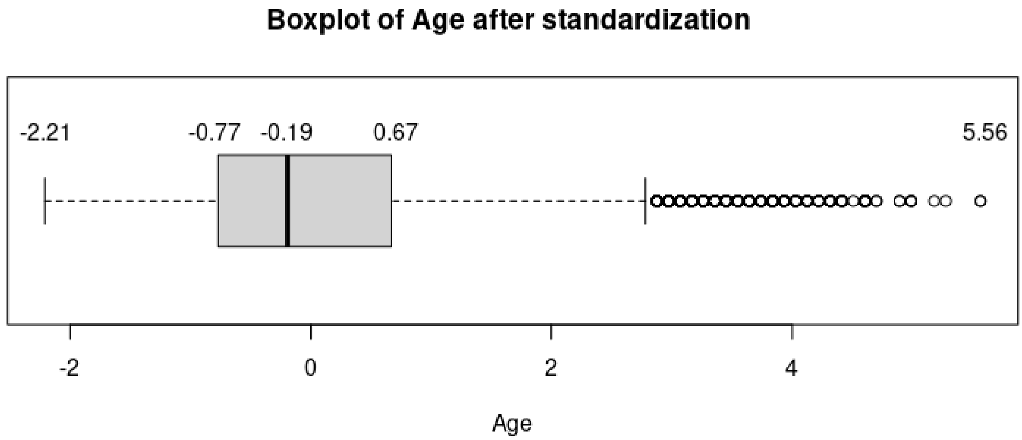
7. Standardize the age variable and save it as a new variable, age_z.

```
#Standardize the age variable
df$age_z = scale(x = df$age)
```

Assignment 1.R × | df × | Filter

	age	education	previous	pdays	y	age_z
1	56	4	0	NA	no	1.533015677
2	57	12	0	NA	no	1.628973456
3	37	12	0	NA	no	-0.290182119
4	40	6	0	NA	no	-0.002308783
5	56	12	0	NA	no	1.533015677
6	45	9	0	NA	no	0.477480111
7	59	14	0	NA	no	1.820889013

```
#Boxplot on age after standardization
boxplot(df$age_z, main="Boxplot of Age after standardization ", xlab="Age", horizontal = TRUE)
text(x=fivenum(round(df$age_z,digits=1)),labels=fivenum(round(df$age_z,digits=2)),y=1.3)
```



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

```
#Obtain a listing of all records that are outliers according to the field age_z
age_outliers <-  df[ which(df$age_z < -3 | df$age_z > 3), ]
age_outliers
```

```
# A tibble: 369 × 6
     age education previous pdays y     age_z[,1]
   <dbl> <chr>        <dbl> <dbl> <chr>     <dbl>
 1    76 16               0    NA no         3.45
 2    73 16               1    NA no         3.16
 3    88 4                0    NA no         4.60
 4    88 4                0    NA yes        4.60
 5    88 4                0    NA yes        4.60
 6    88 4                0    NA no         4.60
 7    88 4                0    NA yes        4.60
 8    88 4                0    NA yes        4.60
 9    88 4                0    NA no         4.60
10    88 4                0    NA yes        4.60
# … with 359 more rows
```

Part 2 (Data Warehousing & OLAP):

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

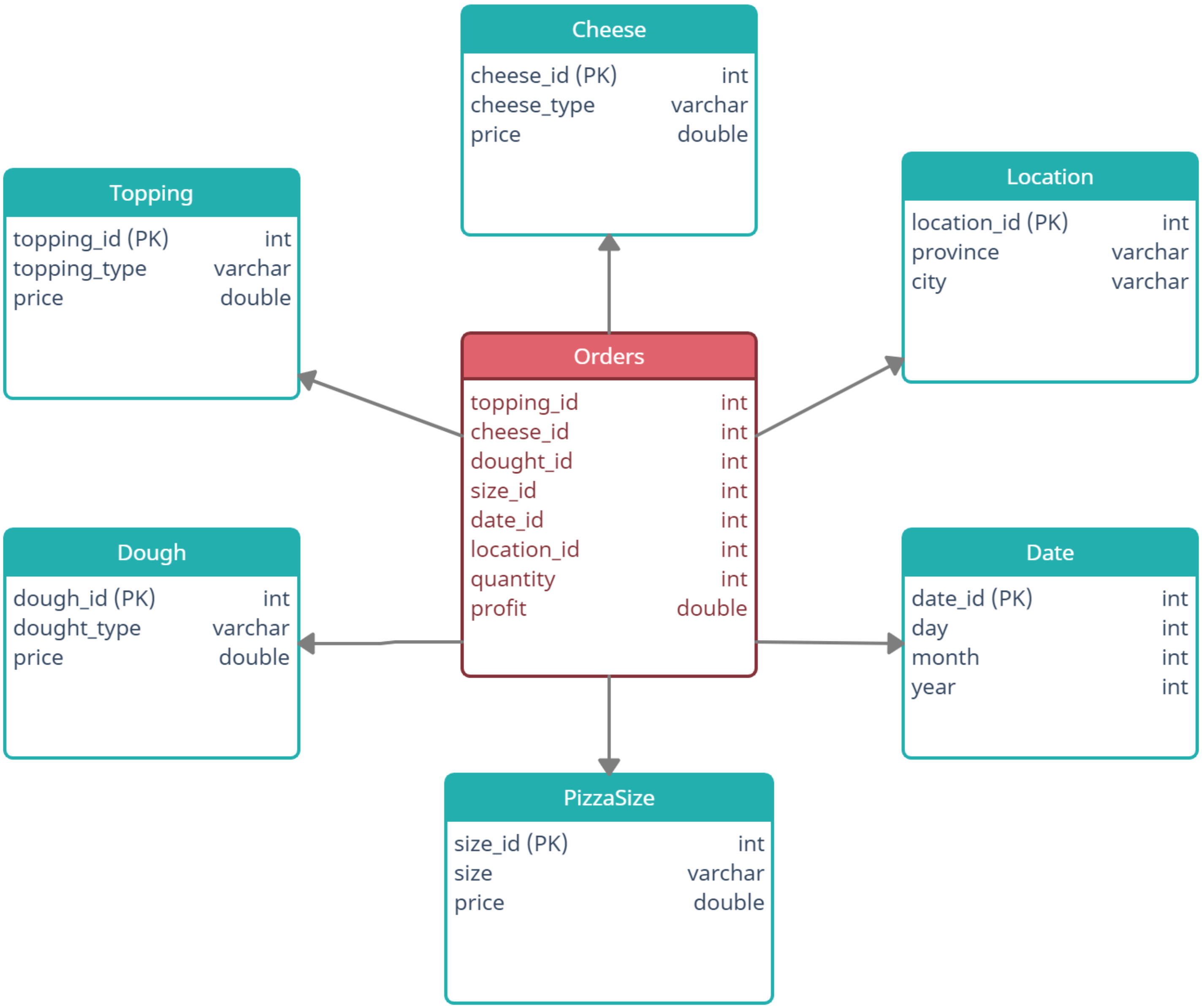


Figure 1 Star Schema

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

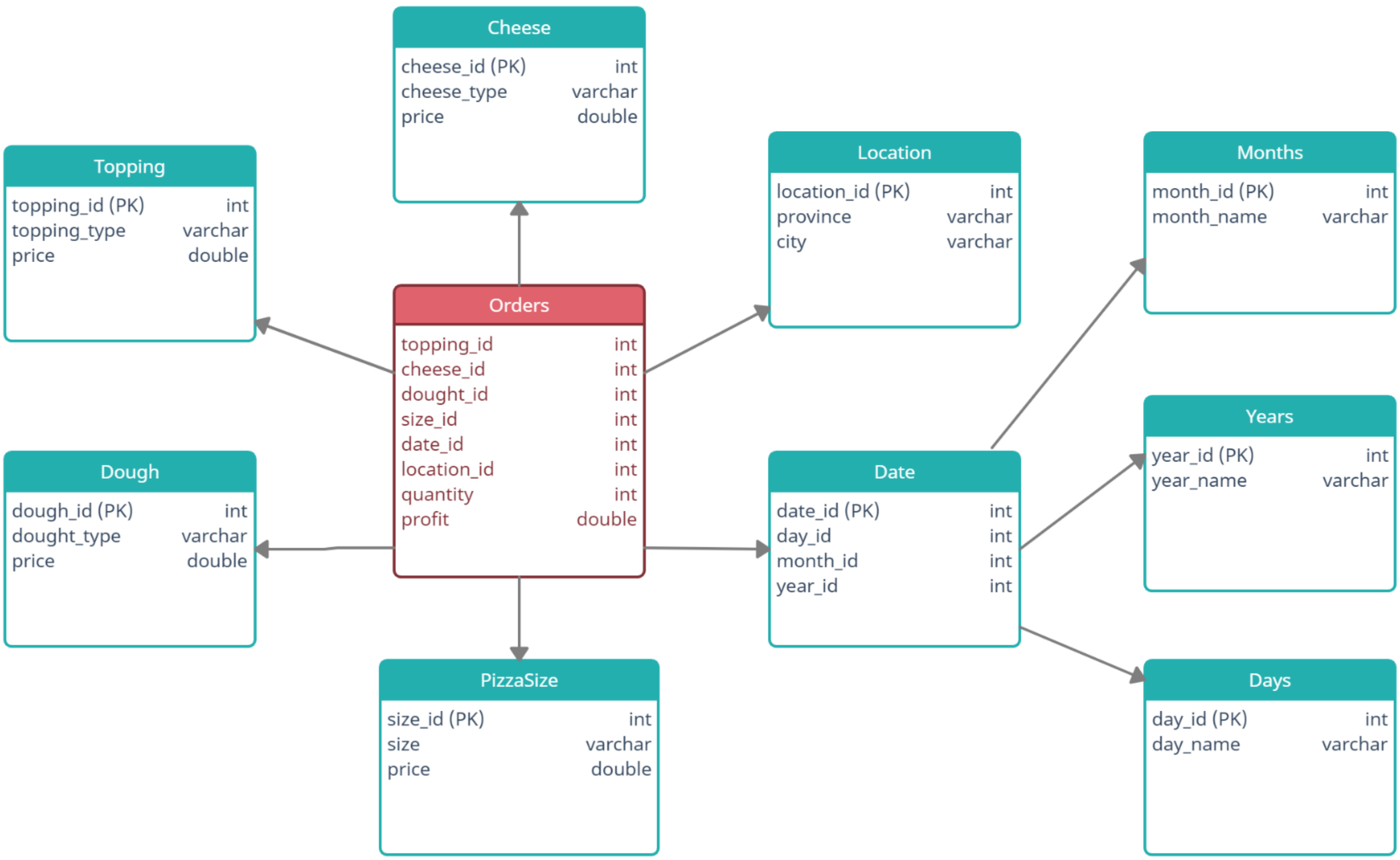


Figure 2 Snowflake Schema

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

    - **Read dimension files in R: -**

      As you can see from figure 2 there are 10 tables so I will create 10 excel files, each file represents a table.

Cloud > project > tables

Name	Size	Modified
..		
Cheese.xlsx	7.9 KB	Oct 5, 2
Date.xlsx	92.9 KB	Oct 5, 2
Days.xlsx	7.9 KB	Oct 5, 2
Dough.xlsx	7.9 KB	Oct 5, 2
Location.xlsx	8 KB	Oct 5, 2
Months.xlsx	8 KB	Oct 5, 2
Orders.xlsx	7.8 KB	Oct 5, 2
PizzaSize.xlsx	7.9 KB	Oct 5, 2
Topping.xlsx	7.9 KB	Oct 5, 2
Years.xlsx	7.9 KB	Oct 5, 2

Then read them in R environment

Assignment 1(Part2).R

Run

```
install.packages("writexl")
install.packages("readxl")
library("writexl")
library("readxl")
Cheese <- read_excel("tables/Cheese.xlsx")
Date <- read_excel("tables/Date.xlsx")
Days <- read_excel("tables/Days.xlsx")
Dough <- read_excel("tables/Dough.xlsx")
Location <- read_excel("tables/Location.xlsx")
Months <- read_excel("tables/Months.xlsx")
Orders <- read_excel("tables/Orders.xlsx")
PizzaSize <- read_excel("tables/PizzaSize.xlsx")
Topping <- read_excel("tables/Topping.xlsx")
Years <- read_excel("tables/Years.xlsx")

```

- **Generate and Fill data in tables using R: -**

  There are 2 empty tables (Date, Orders) which get their data from dimension tables, so we will use the next function to generate **<u>5000</u>** data sample from dimension table and fill these 2 tables.

For Date Table:

Assignment 1(Part2).R

Run Sour

```r

# Function to generate the Date table
gen_date <- function(no_of_recs) {
  # Generate transaction data randomly
  # fill date_id start from 1 to number of sequences
  s_date_id <- 1:no_of_recs
  # fill the day_id from the dimension table Days
  s_day_id <- sample(Days$day_id, no_of_recs,
              replace=T)
  # fill the month_id from the dimension table Months
  s_month_id <- sample(Months$month_id, no_of_recs, replace=T)
  # fill the year_id from the dimension table Years
  s_year_id <- sample(Years$year_id, no_of_recs, replace=T)
  # create data frame of all generated attributes to represent our Date table
  date_table <- data.frame(date_id= s_date_id,
                           day_id=s_day_id,
                           month_id=s_month_id,
                           year_id=s_year_id)

   return (date_table)
}

```

For Orders Table

```r

# Function to generate the Orders table
gen_orders <- function(no_of_recs) {
  # Generate transaction data randomly

  # fill the topping_id from the dimension table Topping
  s_topping_id <- sample(Topping$topping_id, no_of_recs,
                   replace=T)
  # fill the cheese_id from the dimension table Cheese
  s_cheese_id <- sample(Cheese$chesse_id, no_of_recs, replace=T)
  # fill the dough_id from the dimension table Dough
  s_dough_id <- sample(Dough$dough_id, no_of_recs, replace=T)
  # fill the size_id from the dimension table PizzaSize
  s_size_id <- sample(PizzaSize$size_id, no_of_recs, replace=T)
  # fill the date_id from the dimension table Date
  s_date_id <- sample(Date$date_id, no_of_recs, replace=T)
  # fill the location_id from the dimension table Location
  s_location_id <- sample(Location$location_id, no_of_recs, replace=T)
  # fill the dough_id from the dimension table Dough
  s_dough_id <- sample(Dough$dough_id, no_of_recs, replace=T)
  s_quantity <- sample(c(1,2,3,4,5,6,7), no_of_recs, replace=T)
  s_profit <- s_quantity * (Cheese[s_cheese_id,]$price + Topping[s_topping_id,]$price
                          + Dough[s_dough_id,]$price + PizzaSize[s_size_id,]$price)
  # create data frame of all generated attributes to represent our Date table
  orders_table <- data.frame(topping_id= s_topping_id,
                             cheese_id=s_cheese_id,
                             dough_id=s_dough_id,
                             size_id=s_size_id,
                             date_id=s_date_id,
                             location_id=s_location_id,
                             quantity=s_quantity,
                             profit=s_profit)

  return (orders_table)
}
```

2. Build an OLAP cube for profit and show the cells of a subset of the cells

```r
#Build an OLAP cube for profit and show the cells of a subset of the cells
profit_cube <-
  tapply(Orders$profit,
         Orders[,c("size_id", "quantity", "dough_id", "cheese_id")],
         FUN=function(x){return(sum(x))})
profit_cube
dimnames(profit_cube)
```

```
, , dough_id = 2, cheese_id = 3

       quantity
size_id    1    2    3    4     5     6     7
      1 1165 1966 2559 3940  7025  8826  5593
      2 1283 2142 2334 5156  6810  6414  4984
      3 1380 2756 4407 4104  8190  9810 11424
      4 1919 3848 4845 5636  8120 12192  8519
      5 2663 4134 6570 8228 10300  6522 13566

, , dough_id = 3, cheese_id = 3

       quantity
size_id    1    2    3    4     5     6     7
      1 1215 2858 4080 5400  6785  5544  8001
      2 1141 2430 5814 5816  3630  7836  7952
      3 1253 3634 5481 6940  4360  8670 10808
      4 1225 3564 5028 7976 10575 12012 15575
      5 2615 2108 6279 8964  9820 15756 12908
```

```
> dimnames(profit_cube)
$size_id
[1] "1" "2" "3" "4" "5"

$quantity
[1] "1" "2" "3" "4" "5" "6" "7"

$dough_id
[1] "1" "2" "3"

$cheese_id
[1] "1" "2" "3"
```

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 drilldown and roll-up operations that would lead to the conclusion that customers are beginning to prefer bigger pizzas.

- Slice (Xlarge size and Mozzarella cheese)

```r
# Slice
# cube data with xlarge size and Mozzarella cheese
profit_cube["5", , ,"3"]

92:1  (Top Level) ÷
Console  Terminal ×  Jobs ×
R 4.1.0 · /cloud/project/
```

```
        dough_id
quantity     1     2     3
       1  2203  2663  2615
       2  4406  4134  2108
       3  6279  6570  6279
       4  7452  8228  8964
       5  6415 10300  9820
       6 12516  6522 15756
       7  9800 13566 12908
```

- Dice (medium & Large) size and (2,4) quantity

```
# Dice
# cube data with (medium large) size and (2,4) quantity
profit_cube[c(3,4), c(2,4), ,]

102:1  (Top Level)

Console  Terminal  Jobs
R 4.1.0 · /cloud/project/

, , dough_id = 2, cheese_id = 3

       quantity
size_id    2    4
      3 2756 4104
      4 3848 5636

, , dough_id = 3, cheese_id = 3

       quantity
size_id    2    4
      3 3634 6940
      4 3564 7976
```

- Rollup (Size & Pizza component)

```
# Rollup
apply(profit_cube, c("size_id", "dough_id"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})


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

```
> # Rollup
> apply(profit_cube, c("size_id", "dough_id"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
       dough_id
size_id      1      2      3
      1  57743  78946  87544
      2  77646  82099 103445
      3 101411 103853 115180
      4 123647 130490 159784
      5 151863 155408 167810
> apply(profit_cube, c("size_id", "cheese_id"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
       cheese_id
size_id      1      2      3
      1  65512  72883  85838
      2  71886  97545  93759
      3  97752  99320 123372
      4 124763 143660 145498
      5 152174 163403 159504
```

- Drill Down

```
#Drill Down
apply(profit_cube, c("size_id", "dough_id", "cheese_id"),
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
, , cheese_id = 1

       dough_id
size_id     1     2     3
      1 17932 23985 23595
      2 20884 24239 26763
      3 30461 31311 35980
      4 36440 38629 49694
      5 52930 49164 50080

, , cheese_id = 2

       dough_id
size_id     1     2     3
      1 18930 23887 30066
      2 26745 28737 42063
      3 30795 30471 38054
      4 42743 46782 54135
      5 49862 54261 59280

, , cheese_id = 3

       dough_id
size_id     1     2     3
      1 20881 31074 33883
      2 30017 29123 34619
      3 40155 42071 41146
      4 44464 45079 55955
      5 49071 51983 58450
```

Resources: -

- This project is implemented on the Rstudio Cloud, you can access the project using the following link: https://rstudio.cloud/project/2974593

- The schema diagrams created using Creatly website (creatly.com).