EXAMPLE 34 : LISTS

VECTORS, ARRAYS, MATRICES

ARE ALL HOMOGENOUS, I.E. ALL ELEMENTS ARE OF THE **SAME TYPE**

A LIST

IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

LOGICAL

(FALSE, 1, "LIST", (3, FALSE, "SUBLIST"))

VECTORS, ARRAYS, MATRICES

ARE ALL HOMOGENOUS, I.E. ALL ELEMENTS ARE OF THE **SAME TYPE**

A LIST

IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

LOGICAL

(FALSE, 1, "LIST", (3, FALSE, "SUBLIST"))

NUMERIC

VECTORS, ARRAYS, MATRICES

ARE ALL HOMOGENOUS, I.E. ALL ELEMENTS ARE OF THE **SAME TYPE**

A LIST

IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

LOGICAL
CHARACTER
(FALSE, 1, "LIST", (3, FALSE, "SUBLIST"))
NUMERIC

VECTORS, ARRAYS, MATRICES

ARE ALL HOMOGENOUS, I.E. ALL ELEMENTS ARE OF THE **SAME TYPE**

A LIST

IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

LOGICAL CHARACTER LOGICAL CHARACTER

(FALSE, 1, "LIST", (3, FALSE, "SUBLIST"))

NUMERIC NUMERIC

LIST WITHIN A LIST

EXAMPLE 34 : LISTS

A LIST IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

```
family <- list("Mom","Dad", c("Kid1","Kid2","Kid3"),3,c(4,5,7))
```

THE LIST() FUNCTION CAN BE USED TO CREATE A LIST

```
names(family) <- c("Mother","Father","Kids","no.Kids","ages.of.Kids")
```

COMPONENTS OF A LIST CAN BE INDEXED USING [], OR [[]], OR NAMES OF THE COMPONENTS

```
family[1]
$Mother
[1] "Mom"
family[[1]]
[1] "Mom"
family$Mother
[1] "Mom"
```

EXAMPLE 34 : LISTS

A LIST IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

```
family <- list("Mom","Dad", c("Kid1","Kid2","Kid3"),3,c(4,5,7))
names(family) <- c("Mother","Father","Kids","no.Kids","ages.of.Kids")
family[1]
$Mother
[1] "Mom"
family[[1]]
[1] "Mom"
family$Mother
[1] "Mom"
```

WHEN YOU INDEX WITH [] THE RESULT IS A LIST

EXAMPLE 34 : LISTS

A LIST IS A COLLECTION OF ELEMENTS, WHICH CAN BE OF DIFFERENT TYPES INCLUDING LISTS

```
family <- list("Mom","Dad", c("Kid1","Kid2","Kid3"),3,c(4,5,7))
names(family) <- c("Mother","Father","Kids","no.Kids","ages.of.Kids")
family[1]
$Mother
[1] "Mom"
family[[1]]
[1] "Mom"
family$Mother
[1] "Mom"
```

WHEN YOU INDEX WITH [[]] OR WITH THE NAME THE RESULT IS THE COMPONENT OF THE LIST (A VECTOR, OR ARRAY OR LIST ETC)

EXAMPLE 35 : CREATING A DATAFRAME

A DATAFRAME IS DATA ARRANGED IN ROWS AND COLUMNS

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

ROWS REPRESENT 1 UNIT OR 1 OBSERVATION

NAME	AGE	PURCHASE DATE	AMOUNT
"A"	20	Jan 1	100
"B"	54	Feb 3	340
"C"	36	Jan 20	700
"D"	32	Dec 14	650
"E"	45	Nov 18	321
"F"	22	May 10	789

COLUMNS REPRESENT VARIABLES

EXAMPLE 35 : CREATING A DATAFRAME

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

LET'S CREATE A DATA FRAME TO REPRESENT SOME PLAYERS' SCORES

FIRST, WE CREATE 3 VECTORS TO REPRESENT THE VARIABLES

NAME	TEAM	SCORE
Sachin	India	120
Dravid	India	108
McGrath	Aus	98
Warne	Aus	45
Jayawardane	Sri Lanka	115
Drone	India	May 10

```
playerNames <- c("Sachin","Dravid","Mcgrath","Warne","Jayawardane","Dhoni")
teams <- c("India","India","Australia","Australia","Sri Lanka","India")
scores <- c(120, 108, 98, 45, 115, 100)
```

EXAMPLE 35 : CREATING A DATAFRAME

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

```
playerNames <- c("Sachin","Dravid","Mcgrath","Warne","Jayawardane","Dhoni")
teams <- c("India","India","Australia","Australia","Sri Lanka","India")
scores <- c(120, 108, 98, 45, 115, 100)
```

```
cricInfo <- data.frame(playerNames, teams, scores)
```

EXAMPLE 35 : CREATING A DATAFRAME

```
playerNames <- c("Sachin","Dravid","Mcgrath","Warne","Jayawardane","Dhoni")
teams <- c("India","India","Australia","Australia","Sri Lanka","India")
scores <- c(120, 108, 98, 45, 115, 100)
```

```
cricInfo <- data.frame(playerNames, teams, scores)
cricInfo
```

```
  playerNames      teams scores
1      Sachin      India    120
2      Dravid      India    108
3     Mcgrath  Australia     98
4       Warne  Australia     45
5 Jayawardane  Sri Lanka    115
6       Dhoni      India    100
```

THE 3 VECTORS FORM THE COLUMNS OF THE DATA FRAME

EXAMPLE 35 : CREATING A DATAFRAME

```
playerNames <- c("Sachin","Dravid","Mcgrath","Warne","Jayawardane","Dhoni")
teams <- c("India","India","Australia","Australia","Sri Lanka","India")
scores <- c(120, 108, 98, 45, 115, 100)
cricInfo <- data.frame(playerNames, teams, scores)
cricInfo
```

```
nrow(cricInfo)
[1] 6
ncol(cricInfo)
[1] 3
```

	playerNames	teams	scores
1	Sachin	India	120
2	Dravid	India	108
3	Mcgrath	Australia	98
4	Warne	Australia	45
5	Jayawardane	Sri Lanka	115
6	Dhoni	India	100

NROW() AND NCOL() WILL GIVE YOU THE NUMBER OF ROWS AND COLUMNS

EXAMPLE 35 : CREATING A DATAFRAME

```
names(cricInfo)
[1] "playerNames" "teams"
rownames(cricInfo)
[1] "1" "2" "3" "4" "5" "6"
```

```
  playerNames       teams scores
1      Sachin       India    120
2      Dravid       India    108
3     Mcgrath Australia       98
4       Warne Australia       45
5 Jayawardane Sri Lanka      115
6       Dhoni       India    100
```

NAMES() AND ROWNAMES() WILL GIVE YOU VECTORS CONTAINING THE COLUMN NAMES AND ROW NAMES

EXAMPLE 35 : CREATING A DATAFRAME

```
names(cricInfo)
[1] "playerNames" "teams"
"scores"
rownames(cricInfo)
[1] "1" "2" "3" "4" "5" "6"
```

	playerNames	teams	scores
1	Sachin	India	120
2	Dravid	India	108
3	Mcgrath	Australia	98
4	Warne	Australia	45
5	Jayawardane	Sri Lanka	115
6	Dhoni	India	100

NAMES() AND ROWNAMES() WILL GIVE YOU VECTORS CONTAINING THE COLUMN NAMES AND ROW NAMES

EXAMPLE 35 : CREATING A DATAFRAME

WHAT HAPPENS IF THE VECTORS USED TO CREATE THE DATA FRAME ARE NOT OF EQUAL LENGTH?

```
cricInfoNew <- data.frame(playerNames, teams,
                          scores = c(scores,20))
```

THERE IS AN EXTRA ELEMENT IN SCORES

```
Error in data.frame(playerNames, teams, scores = c(scores,
20)): arguments imply differing number of rows: 6, 7
```

EXAMPLE 36 : READING DATA FROM FILES

EXAMPLE 36 : READING DATA FROM FILES

IF YOU HAVE A FILE WITH DATA ARRANGED IN ROWS AND COLUMNS, YOU CAN DIRECTLY READ IT INTO A DATA FRAME

```
u.user     -- Demographic information about the users; this is a tab
              separated list of
              user id | age | gender | occupation | zip code
              The user ids are the ones used in the u.data data set.
```

U.USER IS A FILE WITH SOME USER RELATED DATA

EXAMPLE 36 : READING DATA FROM FILES

U.USER IS A FILE WITH SOME USER RELATED DATA

```
u.user     -- Demographic information about the users; this is a tab
              separated list of
              user id | age | gender | occupation | zip code
              The user ids are the ones used in the u.data data set.
```

THE COLUMN NAMES

NOTE: THESE ARE NOT INCLUDED IN THE FILE

EXAMPLE 36 : READING DATA FROM FILES

U.USER IS A FILE WITH SOME USER RELATED DATA

HERE IS WHAT THE DATA LOOKS LIKE IN A TEXT EDITOR

THE COLUMNS ARE SEPARATED BY "|"

```
user id | age | gender | occupation | zip code
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
12|28|F|other|06405
```

EXAMPLE 36 : READING DATA FROM FILES

U.USER IS A FILE WITH SOME USER RELATED DATA

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
```

U.USER HOLDS THE FILE PATH

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
```

```
header <- c("userId","age","gender","occupation","zipCode")
```

WE'LL SET UP A HEADER VARIABLE TO TELL THE DATA FRAME WHAT THE COLUMN NAMES ARE

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")
```

```
userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

READ.TABLE() WILL READ THE DATA INTO A DATA FRAME

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")

userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

THE FILE TO READ FROM

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")

userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

THERE IS NO HEADER ROW IN THE FILE

THIS IS THE DEFAULT OPTION

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")

userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

THERE COLUMNS ARE SEPARATED BY "|"

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")

userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

STRINGS ARE ENCLOSED IN "" OR ''

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")

userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

TAKE THE COLUMN NAMES FROM THE HEADER VARIABLE

EXAMPLE 36 : READING DATA FROM FILES

```
u.user <- "/Users/swethakolalapudi/Downloads/ml-100k/u.user"
header <- c("userId","age","gender","occupation","zipCode")
userInfo <- read.table(u.user,header = FALSE, sep ="|",
                       quote = "\"'",col.names = header)
```

```
head(userInfo,10)
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

ONCE YOU'VE READ THE FILE YOU CAN CHECK IF EVERYTHING WENT AS PLANNED

THIS WILL PRINT THE FIRST 10 ROWS OF THE DATA FRAME

EXAMPLE 37 : INDEXING A DATAFRAME

EXAMPLE 37 : INDEXING A DATAFRAME

```
head(userInfo,10)
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

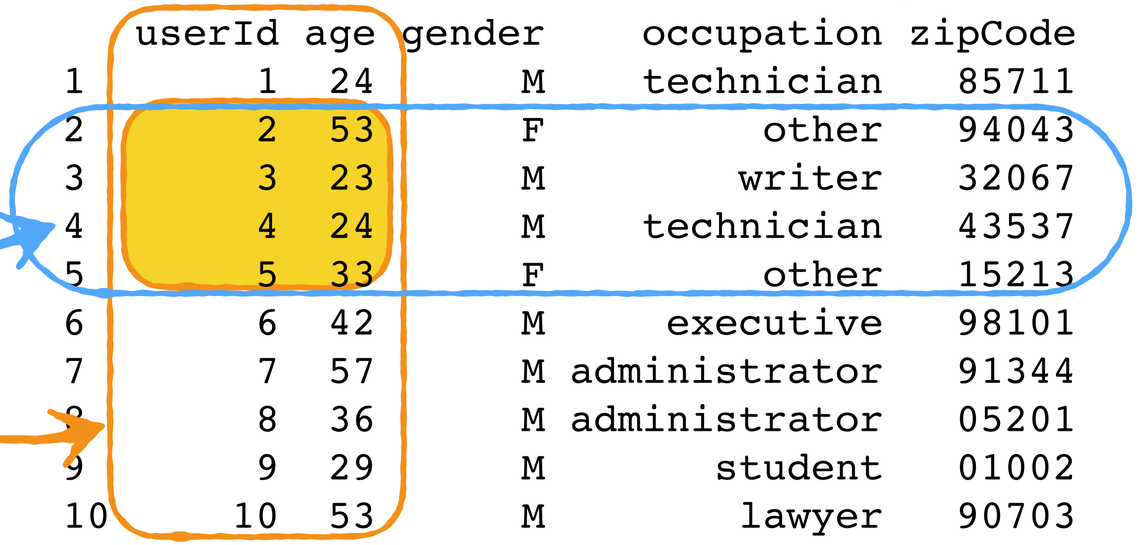
USERINFO IS A DATAFRAME WITH SOME USER DEMOGRAPHIC DATA

YOU CAN INDEX A DATAFRAME LIKE YOU INDEX A MATRIX/ARRAY

EXAMPLE 37 : INDEXING A DATAFRAME

```
userInfo[2:5,1:2]
```

	userId	age
2	2	53
3	3	23
4	4	24
5	5	33



	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

YOU CAN INDEX A DATAFRAME LIKE YOU INDEX A MATRIX/ARRAY

EXAMPLE 37 : INDEXING A DATAFRAME

```
userInfo[2:5,1:2]
```

	userId	age
2	2	53
3	3	23
4	4	24
5	5	33

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

THE RESULT IS A DATA FRAME ALSO, WITH THE ROW NAMES AND COLUMN NAMES PRESERVED

EXAMPLE 37 : INDEXING A DATAFRAME

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

YOU CAN ACCESS THE ELEMENTS OF THE LIST USING THEIR NAMES

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

EACH ELEMENT IS A COLUMN OF THE DATAFRAME

EXAMPLE 37 : INDEXING A DATAFRAME

```
userInfo$age[1:10]
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

THIS ACCESSES THE AGE COLUMN

UNLIKE THE PREVIOUS CASE, THE RESULT OF THIS INDEXING IS A VECTOR (NOT A DATAFRAME)

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

EXAMPLE 37 : INDEXING A DATAFRAME

```
userInfo$age[1:10]
```

```
 [1] 24 53 23 24 33 42 57 36 29 53
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

THIS GETS US THE FIRST 10 ELEMENTS OF THAT VECTOR

A DATAFRAME IS LIKE A LIST (OF VECTORS OR OTHER LISTS)

EXAMPLE 37 : INDEXING A DATAFRAME

```
class(userInfo$gender)
[1] "factor"
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

WHEN YOU CREATE A DATAFRAME, COLUMNS WHICH CONTAIN STRINGS ARE AUTOMATICALLY TREATED AS FACTORS

I.E. THE COLUMN WILL BE AWARE OF THE DISTINCT SET OF STRING VALUES IT CONTAINS (LEVELS)

EXAMPLE 37 : INDEXING A DATAFRAME

```
length(levels(userInfo$occupation))
```

THE OCCUPATION COLUMN WHICH IS A FACTOR (SINCE IT CONTAINS STRINGS)

LET'S FIND THE DISTINCT NUMBER OF OCCUPATIONS IN THE DATASET

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

EXAMPLE 37 : INDEXING A DATAFRAME

```
length(levels(userInfo$occupation))
```

A VECTOR CONTAINING THE DISTINCT SET OF OCCUPATIONS

LET'S FIND THE DISTINCT NUMBER OF OCCUPATIONS IN THE DATASET

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

EXAMPLE 22 : INDEXING A DATAFRAME

```
length(levels(userInfo$occupation))
```

```
[1] 21
```

A VECTOR CONTAINING THE DISTINCT SET OF OCCUPATIONS

THE LENGTH OF THAT VECTOR

LET'S FIND THE DISTINCT NUMBER OF OCCUPATIONS IN THE DATASET

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

```
head(userInfo,10)
```

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

USERINFO IS A DATAFRAME WITH SOME USER DEMOGRAPHIC DATA

LIKE WITH AN SQL TABLE, YOU CAN PERFORM GROUPING WITH A DATA FRAME

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

LIKE WITH AN SQL TABLE, YOU CAN PERFORM GROUPING WITH A DATA FRAME

I.E. YOU CAN DIVIDE THE DATA INTO GROUPS AND THEN COUNT, SUM ETC THE VALUES

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS?

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender, userInfo, mean)
```

FROM THE DATA FRAME USERINFO

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS?

FROM THE DATA FRAME USERINFO

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender, userInfo, mean)
```

TAKE EACH DISTINCT VALUE OF THE GENDER

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS?

FROM THE DATA FRAME USERINFO TAKE EACH DISTINCT VALUE OF THE GENDER

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender, userInfo, mean)
```

```
gender      age
1      F 33.81319
2      M 34.14925
```

AND COMPUTE THE MEAN OF THE AGE COLUMN

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS IN EACH OCCUPATION?

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender + occupation,
          userInfo, mean)
```

FROM THE DATA FRAME USERINFO

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS IN EACH OCCUPATION?

FROM THE DATA FRAME USERINFO

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender + occupation,
          userInfo, mean)
```

TAKE EACH DISTINCT COMBINATION OF GENDER AND OCCUPATION

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: WHAT IS THE AVERAGE AGE OF MALE VS FEMALE USERS IN EACH OCCUPATION?

FROM THE DATA FRAME USERINFO

TAKE EACH DISTINCT COMBINATION OF GENDER AND OCCUPATION

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
aggregate(age ~ gender + occupation,
          userInfo, mean)
```

	gender	occupation	age
1	F	administrator	40.63889
2	M	administrator	37.16279
3	F	artist	30.30769
4	M	artist	32.33333
5	M	doctor	43.57143
6	F	educator	39.11538
7	M	educator	43.10145

AND COMPUTE THE MEAN OF THE AGE COLUMN

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: FIND THE TOP 3 OCCUPATIONS OF THE USERS

FIRST FIND THE NUMBER OF USERS IN EACH OCCUPATION

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
occupationCounts <- aggregate(userId ~ occupation,
                              userInfo, length)
```

FOR EACH DISTINCT OCCUPATION,

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: FIND THE TOP 3 OCCUPATIONS OF THE USERS

FIRST FIND THE NUMBER OF USERS IN EACH OCCUPATION

	userId	age	gender	occupation	zipCode
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002
10	10	53	M	lawyer	90703

```
occupationCounts <- aggregate(userId ~ occupation,
                              userInfo, length)
```

FIND THE NUMBER OF USER IDS

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: FIND THE TOP 3 OCCUPATIONS OF THE USERS

FIRST FIND THE NUMBER OF USERS IN EACH OCCUPATION

```
occupationCounts <- aggregate(userId ~ occupation,
                              userInfo, length)
```

OCCUPATIONCOUNTS IS A DATAFRAME

	occupation	userId
1	administrator	79
2	artist	28
3	doctor	7
4	educator	95
5	engineer	67
6	entertainment	18
7	executive	32
8	healthcare	16
9	homemaker	7

YOU CAN SORT IT BY THE #USERIDS COLUMN

```
head(occupationCounts[order(occupationCounts$userId, decreasing = TRUE),],3)
```

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: FIND THE TOP 3 OCCUPATIONS OF THE USERS

FIRST FIND THE NUMBER OF USERS IN EACH OCCUPATION

```
occupationCounts <- aggregate(userId ~ occupation,
                              userInfo, length)
```

	occupation	userId
1	administrator	79
2	artist	28
3	doctor	7
4	educator	95
5	engineer	67
6	entertainment	18
7	executive	32
8	healthcare	16
9	homemaker	7

OCCUPATIONCOUNTS IS A DATAFRAME

YOU CAN SORT IT BY THE #USERIDS COLUMN

```
head(occupationCounts[order(occupationCounts$userId, decreasing = TRUE),],3)
```

IN DESCENDING ORDER

EXAMPLE 38 : AGGREGATES USING DATA FRAMES

OBJECTIVE: FIND THE TOP 3 OCCUPATIONS OF THE USERS

FIRST FIND THE NUMBER OF USERS IN EACH OCCUPATION

```
occupationCounts <- aggregate(userId ~ occupation,
                              userInfo, length)
```

```
head(occupationCounts[order(occupationCounts$userId, decreasing = TRUE),],3)
```

PRINT THE TOP 3 ROWS

```
   occupation userId
19    student    196
14      other    105
4    educator     95
```

WHEN YOU SORT A DATAFRAME, THE ROW NAMES ARE ALSO SORTED

EXAMPLE 39 : MERGING DATA FRAMES

EXAMPLE 39 : MERGING DATA FRAMES

WE HAVE 2 DATA FRAMES WITH CUSTOMER RELATED DATA

```
df1 <- data.frame(custId = 1:3, custName = c("Vitthal","Janani","Navdeep"))
df2 <- data.frame(custId = 1:3, custAge = c(36,36,26))
```

DF1

	custId	custName
1	1	Vitthal
2	2	Janani
3	3	Navdeep

DF2

	custId	custAge
1	1	36
2	2	36
3	3	26

EXAMPLE 39 : MERGING DATA FRAMES

DF1

	custId	custName
1	1	Vitthal
2	2	Janani
3	3	Navdeep

DF2

	custId	custAge
1	1	36
2	2	36
3	3	26

```
df3 <- merge(df1, df2, by="custId")
```

MERGE WILL COMBINE THE 2 DATA FRAMES BY MATCHING THE ROWS BASED ON THE SPECIFIED COLUMN

EXAMPLE 39 : MERGING DATA FRAMES

DF1

```
  custId custName
1      1  Vitthal
2      2   Janani
3      3  Navdeep
```

DF2

```
  custId custAge
1      1      36
2      2      36
3      3      26
```

```
df3 <- merge(df1, df2, by="custId")
```

```
  custId custName custAge
1      1  Vitthal      36
2      2   Janani      36
3      3  Navdeep      26
```

EXAMPLE 39 : MERGING DATA FRAMES

```
df3 <- merge(df1, df2, by="custId")
```

THIS IS PRETTY SIMILAR TO A JOIN IN SQL
OR A LOOKUP IN EXCEL

EXAMPLE 39 : MERGING DATA FRAMES

```
df3 <- merge(df1, df2, by="custId")
```

YOU CAN SPECIFY MULTIPLE COLUMNS TO MERGE ON

EXAMPLE 39 : MERGING DATA FRAMES

```
df3 <- merge(df1, df2, by="custId")
```

IF YOU DON'T SPECIFY THE BY , IT WILL MERGE BASED THE COLUMN NAMES WHICH ARE COMMON

EXAMPLE 39 : MERGING DATA FRAMES

```
df3 <- merge(df1, df2, by="custId")
```

```
df3 <- merge(df1, df2, by.x="custId",by.y="custId")
```

YOU CAN SPECIFY THE COLUMNS TO MATCH FROM EACH DATA FRAME SEPARATELY