ADBMS Practical

ADBMS LAB ASSIGNMENT 1

Name:- Atharva sawant Roll No:- 1451

Q1) Student Table

Query:

```
create table student (name varchar(30),rollno number(10), marks number(10))
partition by range (marks)(partition A values less than (25), partition B values less
than (30), partition C values less than (35),partition D values less than (40))
```

```
Table created.
```

```
insert into Student
values(414,'Rahul','05-Sep-2002',9329869499,'Dadar','Mumbai',400028)
```

```
select * from Student
```

ROLLNO	NAME	DOB	MOBILENO	ADDRESS	CITY	ZIPCODE
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
400	Nehal	23-JUN-02	9328866949	Bandra	Mumbai	400024
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
400	Nehal	23-JUN-02	9328866949	Bandra	Mumbai	400024
429	Ethan	16-MAY-03	9327669499	Virar	Mumbai	400024
429	Ethan	16-MAY-03	9327669499	Virar	Mumbai	400024
429	Ethan	16-MAY-03	9327669499	Virar	Mumbai	400024
429	Ethan	16-MAY-03	9327669499	Virar	Mumbai	400024

```
12 rows selected.
```

Q) Student (Partition A)

```
select * from Student partition(A)
```

ROLLNO	NAME	DOB	MOBILENO	ADDRESS	CITY	ZIPCODE
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
400	Nehal	23-JUN-02	9328866949	Bandra	Mumbai	400024
414	Rahul	05-SEP-02	9329869499	Dadar	Mumbai	400028
400	Nehal	23-JUN-02	9328866949	Bandra	Mumbai	400024

```
8 rows selected.
```

Q) Add Partition

alter table Student add partition D values less than(460)

```
Table altered.
```

alter table Student add partition E values less than(475)

```
Table altered.
```

Q2) SalesPart Table

Query:

create table SalesPart (sales_id number(5),sales_person_name varchar(20),DOB number(10),date_of_joining number(10),sales_person_region varchar(20))partition by list(sales_person_region)(partition R1_west values ('MH','GJ','GA','RJ'), partition R2_east values ('AS','NL','MZ'),partition R3_north values ('DL','JK','HP'))

insert into SalesPart values(1,'Ayush',05092002,12,'MH')

```
select * from SalesPart
```

SALES_ID	SALES_PERSON_NAME	DOB	DATE_OF_JOINING	SALES_PERSON_REGION
3	Yash	1022002	12	GJ
4	Nehal	5062002	12	GA
1	Ayush	5092002	12	MH
5	Rohan	5122002	12	AS
2	Ethan	3082003	11	JK

5 rows selected.

```
select * from SalesPart partition(R1_west)
```

SALES_ID	SALES_PERSON_NAME	DOB	DATE_OF_JOINING	SALES_PERSON_REGION
3	Yash	1022002	12	GJ
4	Nehal	5062002	12	GA
1	Ayush	5092002	12	MH

3 rows selected.

ADBMS LAB ASSIGNMENT 2

Q1) Employee Table

Query:

CREATE TABLE emp

(

```
empno NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,

ename VARCHAR2(10),

job VARCHAR2(9),

mgr NUMBER(4),

hiredate DATE,

sal NUMBER(7,2),

comm NUMBER(7,2),

deptno NUMBER(2)

);
```

```
Table created.
```

```
insert into emp values
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20)

insert into emp values
(7666,'AYUSH','SALESMAN',7902,to_date('17-12-1980','dd-mm-yyyy'),2000,NULL
,20)

insert into emp values
(7999,'ETHAN','CLERK',7902,to_date('17-12-1989','dd-mm-yyyy'),800,NULL,20)

insert into emp values
(7999,'ETHAN','CLERK',7902,to_date('17-12-1989','dd-mm-yyyy'),800,NULL,20)

insert into emp values
(7321,'AMRISH','CLERK',7902,to_date('17-12-1999','dd-mm-yyyy'),1800,NULL,20)
```

```
select *from emp
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7666	AYUSH	SALESMAN	7902	17-DEC-80	2000	-	20
7999	ETHAN	CLERK	7902	17-DEC-89	800	-	20
7599	NEHAL	CLERK	7902	17-DEC-70	800	-	20
7431	MEET	CLERK	7902	17-DEC-76	800	-	20
7321	AMRISH	CLERK	7902	17-DEC-99	1800	-	20

```
7 rows selected.
```

Q) Assign a dense rank to all employees of dept

Query:

select empno, deptno, sal, Rank()over(partition by deptno order by sal) as myRank from emp

EMPNO	DEPTNO	SAL	MYRANK
7599	20	800	1
7999	20	800	1
7431	20	800	1
7369	20	800	1
7321	20	1800	5
7666	20	2000	6
7645	20	5000	7
7888	20	9000	8
7123	20	9000	8
7499	30	1600	1

10 rows selected.

Q) Find min and max

Query:

select empno, deptno, sal, min(sal) Keep(Dense_Rank first order by sal) over(partition by deptno) as Lowest,

max (sal) Keep(Dense_Rank last order by sal) over(partition by deptno) as Highest from emp

EMPNO	DEPTNO	SAL	LOWEST	HIGHEST
7645	20	5000	800	9000
7888	20	9000	800	9000
7123	20	9000	800	9000
7369	20	800	800	9000
7431	20	800	800	9000
7666	20	2000	800	9000
7999	20	800	800	9000
7599	20	800	800	9000
7321	20	1800	800	9000
7499	30	1600	1600	1600

Q) Calculate the difference

Query:

select empno, ename, deptno, sal, Lag(sal, 1,0) over(order by sal) as Previous_sal, sal-lag(sal, 1,0)over(order by sal) as sal_difference from emp

EMPNO	ENAME	DEPTNO	SAL	PREVIOUS_SAL	SAL_DIFFERENCE
7431	MEET	20	800	0	800
7599	NEHAL	20	800	800	0
7999	ETHAN	20	800	800	0
7369	SMITH	20	800	800	0
7499	ALLEN	30	1600	800	800
7321	AMRISH	20	1800	1600	200
7666	AYUSH	20	2000	1800	200
7645	YASH	20	5000	2000	3000
7123	PARKER	20	9000	5000	4000
7888	NISHANT	20	9000	9000	0

10 rows selected

Q) Rollup Function

Query:

select deptno, job, count(*), sum(sal) from emp group by rollup(deptno, job)

DEPTNO	JOB	COUNT(*)	SUM(SAL)
20	CLERK	6	14000
20	MANAGER	2	14000
20	SALESMAN	1	2000
20	-	9	30000
30	SALESMAN	1	1600
30	-	1	1600
-	-	10	31600

Q) Cube Function

Query:

select deptno, job, count(*), sum(sal) from emp group by cube(deptno, job)

DEPTNO	JOB	COUNT(*)	SUM(SAL)
-	-	10	31600
-	CLERK	6	14000
-	MANAGER	2	14000
-	SALESMAN	2	3600
20	-	9	30000
20	CLERK	6	14000
20	MANAGER	2	14000
20	SALESMAN	1	2000
30	-	1	1600
30	SALESMAN	1	1600

ADBMS LAB ASSIGNMENT 3

Q1) Create type with name book_type by grouping the information ISBNno varchar2(20), Title varchar2(20), author varchar2(20)

Query:

```
Create type book_type as object
(
    isbn_no varchar2(30),
    title varchar2(20),
    author varchar2(30)
);
```

```
Type created.
```

Q) Pid number, book book_type,pdate date,amount number

Query:

```
create table purchase
(
    pid number,
    book book_type,
    pdata date,
    amount number
)
```

```
Type created.
```

Q) Insert Values

Query:

```
insert into purchase values(1,book_type(101,'Data Mining','AyushA'),'19-May-2020',500)

insert into purchase values(1,book_type('101','Data Mining','AyushA'),'19-May-2020',500)

insert into purchase values(1,book_type('102','Valorant','RohitD'),'10-Jan-2020',900)

insert into purchase
```

values(1,book_type('103','CSGO','EthanA'),'15-Mar-2020',200)

insert into purchase values(1,book_type('104','Apex Legends','NehalA'),'20-Dec-2020',100)

```
select pid, p.book.isbn_no,p.book.title,p.book.author,pdata,amount from purchase p
```

PID	BOOK.ISBN_NO	BOOK.TITLE	BOOK.AUTHOR	PDATA	AMOUNT
1	102	Valorant	RohitD	10-JAN-20	900
1	103	CSGO	EthanA	15-MAR-20	200
1	104	Apex Legends	NehalA	20-DEC-20	100
1	101	Data Mining	AyushA	19-MAY-20	500
1	101	Data Mining	AyushA	19-MAY-20	500

Q) Select * from purchase (Amount<500)

Query:

select pid, p.book.isbn_no,p.book.title,p.book.author,pdata,amount from purchase p where amount<500

PID	BOOK.ISBN_NO	BOOK.TITLE	BOOK.AUTHOR	PDATA	AMOUNT
1	103	CSGO	EthanA	15-MAR-20	200
1	104	Apex Legends	NehalA	20-DEC-20	100

Q) Write a query to update the author name of the book titled “Data Mining”.

Query:

update purchase p

set p.book.author = 'EthanA'

where p.book.title = 'Data Mining'

```
2 row(s) updated.
```

Query:

select pid, p.book.isbn_no,p.book.title,p.book.author,pdata,amount from purchase p where p.book.title = 'Data Mining'

PID	BOOK.ISBN_NO	BOOK.TITLE	BOOK.AUTHOR	PDATA	AMOUNT
1	101	Data Mining	EthanA	19-MAY-20	500
1	101	Data Mining	EthanA	19-MAY-20	500

2 rows selected.

Q) Create type person with fields name and address

Query:

```
create type landmark as object
(
    city varchar(50),
    pincode number
);
```

```
Type created.
```

```
create type address as object
(
    street landmark,
    country varchar(30),
    state varchar(50)
);
create table details
(
    name varchar(50),
    addr address,
```

mobNo number

)

```
Table created.
```

insert into details
values('Ayush',address(landmark('Mumbai',400028),'India','Maharashtra'),9320005555)

```
select * from details
```

NAME	ADDR	MOBNO
Ayush	[unsupported data type]	9320005555

Q)

Query:

select name, d.addr.street.city, d.addr.street.pincode, d.addr.country, d.addr.state, mobNo from details d

NAME	ADDR.STREET.CITY	ADDR.STREET.PINCODE	ADDR.COUNTRY	ADDR.STATE	MOBNO
Ayush	Mumbai	400028	India	Maharashtra	9320005555
Ayush	Mumbai	400028	India	Maharashtra	9320005555
Ethan	Surat	400019	India	Gujurat	9320005555
Nehal	Bangalore	400021	India	Karnataka	9393939393
Meet	Panaji	400020	India	Goa	9928938985

ADBMS LAB ASSIGNMENT 4

Q) Create a variable

var_num = 20

Values	
var_num	20

Q) Assign Operator

a=10

b=20

30 = var1

var1

```
> var1
[1] 30
```

c = 500

print(c)

```
> c = 500
> print(c)
[1] 500
```

Q) DataTypes in R

num = TRUE

num

```
> num = TRUE
> num
[1] TRUE
```

Q) converting into numeric type

num3 <- as.numeric(TRUE)

num3

class(num3)

```
> num3 <- as.numeric(TRUE)
> num3
[1] 1
> class(num3)
[1] "numeric"
```

Q) Converting into Integer

c<- as.integer(TRUE)

c

class(c)

```
> c<- as.integer(TRUE)
> c
[1] 1
> class(c)
[1] "integer"
```

Q) Converting into Complex Type

r <- as.complex('1234ayush')

r

class(r)

```
> r <- as.complex('1234ayush')
Warning message:
NAs introduced by coercion
> r
[1] NA
> class(r)
[1] "complex"
```

Q) Arthimetic Operator

b <- c(14,15,16,17)

b

```
> b <- c(14,15,16,17)
> b
[1] 14 15 16 17
```

Q) Display all the list of directory

dir()

```
> dir()
[1] "desktop.ini"    "My Music"        "My Pictures"
[4] "My Videos"      "OpenIV"          "Rockstar Games"
```

Q) print all the variable that has being declared

ls()

```
> ls()
[1] "b"       "c"       "num"     "num3"    "r"
[6] "var_num" "var1"
```

Q) creating Matrix

m <- matrix(c(11,12,13,14,15,16,17,18,19),nrow = 3, ncol = 3)

print(m)

dim(m)

m<- matrix(c(11,12,13,14,15,16,17,18,19), nrow = 3, ncol = 3, byrow = TRUE)

m

```
> m <- matrix(c(11,12,13,14,15,16,17,18,19),nrow = 3, ncol = 3)
> print(m)
     [,1] [,2] [,3]
[1,]   11   14   17
[2,]   12   15   18
[3,]   13   16   19
> dim(m)
[1] 3 3
> m<- matrix(c(11,12,13,14,15,16,17,18,19), nrow = 3, ncol = 3, byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   14   15   16
[3,]   17   18   19
```

Q) using rbind and cbind function

```
> a<- c(1,2,3)
> b<- c(10,20,30)
> cbind(a,b)
      a  b
[1,] 1 10
[2,] 2 20
[3,] 3 30
> rbind(a,b)
  [,1] [,2] [,3]
a    1    2    3
b   10   20   30
```

Q) multiplication by vector

m<- matrix(c(11,12,13,14,15,16,17,18,19), nrow = 3, ncol = 3, byrow = TRUE)

m

n<- matrix(c(14,12,13,14,15,16,17,18,18), nrow = 3, ncol = 3, byrow = TRUE)

print(n)

a<- m*n

a

```
> m<- matrix(c(11,12,13,14,15,16,17,18,19), nrow = 3, ncol = 3, byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   14   15   16
[3,]   17   18   19
> n<- matrix(c(14,12,13,14,15,16,17,18,18), nrow = 3, ncol = 3, byrow = TRUE)
> print(n)
     [,1] [,2] [,3]
[1,]   14   12   13
[2,]   14   15   16
[3,]   17   18   18
> a<- m*n
> a
     [,1] [,2] [,3]
[1,]  154  144  169
[2,]  196  225  256
[3,]  289  324  342
```

ADBMS LAB ASSIGNMENT 5

Install:

1. XLConnect
2. writexl
3. readxl
4. dplyr
5. Hmisc

Step 1: Create the file in XL add file into it then save file with .csv extension

Step 2: Place your file in current working directory

```
>
> data1 <- read.csv("noodles.csv")
> data1
   Country.Region X2018 X2019 X2020 X2021 X2022 Rank CCA3 Country.Territory        Capital
1           China 40250 41450 46360 43990 45070    1  CHN             China        Beijing
2       Indonesia 12540 12520 12640 13270 14260    4  IDN         Indonesia        Jakarta
3           India  6060  6730  6730  7560  7580    2  IND             India      New Delhi
4           Japan  5780  5630  5970  5850  5980   11  JPN             Japan          Tokyo
5     Philippines  3980  3850  4470  4440  4290   13  PHL       Philippines         Manila
6     South Korea  3820  3900  4130  3790  3950   29  KOR       South Korea          Seoul
7        Thailand  3460  3570  3710  3630  3870   20  THA          Thailand        Bangkok
8          Brazil  2390  2420  2720  2850  2830    7  BRA            Brazil       Brasilia
9         Nigeria  1820  1920  2460  2620  2790    6  NGA           Nigeria          Abuja
10         Russia  1850  1910  2000  2100  2200    9  RUS            Russia         Moscow
11          Nepal  1570  1640  1540  1590  1650   49  NPL             Nepal      Kathmandu
12       Malaysia  1370  1450  1570  1580  1550   45  MYS          Malaysia    Kuala Lumpur
13         Mexico  1120  1170  1160  1360  1510   10  MEX            Mexico     Mexico City
14         Taiwan   830   830   870   900   880   57  TWN            Taiwan          Taipei
15   Saudi Arabia   550   560   830   850   870   41  SAU      Saudi Arabia         Riyadh
16        Myanmar   600   620   660   760   770   26  MMR           Myanmar     Nay Pyi Taw
17   South Africa   260   280   350   410   480   24  ZAF      South Africa        Pretoria
18          Egypt   220   280   350   400   460   14  EGY             Egypt           Cairo
19      Australia   410   420   440   450   450   55  AUS         Australia        Canberra
20     Bangladesh   310   370   370   430   440    8  BGD        Bangladesh          Dhaka
21         Turkey    80   120   190   360   420   18  TUR            Turkey          Ankara
22       Cambodia   330   350   370   410   400   73  KHM          Cambodia      Phnom Penh
23        Germany   320   330   370   390   390   19  DEU           Germany          Berlin
24         Poland   310   310   330   340   380   37  POL            Poland          Warsaw
25     Kazakhstan   170   250   280   320   360   66  KAZ        Kazakhstan       Nursultan
26      Guatemala   230   250   260   270   280   68  GTM         Guatemala Guatemala City
27       Pakistan   190   200   220   230   240    5  PAK          Pakistan       Islamabad
28     Uzbekistan   170   210   210   210   210   43  UZB        Uzbekistan        Tashkent
29         Canada   190   190   190   200   200   39  CAN            Canada          Ottawa
```

Source code:

#Display first 6 row head

(data1)

#Display first 2 row

head(data1 , 2)

Output:

```
> head(data1)
  Country.Region X2018 X2019 X2020 X2021 X2022 Rank CCA3 Country.Territory   Capital Continent
1          China 40250 41450 46360 43990 45070    1  CHN             China   Beijing      Asia
2      Indonesia 12540 12520 12640 13270 14260    4  IDN         Indonesia   Jakarta      Asia
3          India  6060  6730  6730  7560  7580    2  IND             India New Delhi      Asia
4          Japan  5780  5630  5970  5850  5980   11  JPN             Japan     Tokyo      Asia
5    Philippines  3980  3850  4470  4440  4290   13  PHL       Philippines    Manila      Asia
6    South Korea  3820  3900  4130  3790  3950   29  KOR       South Korea     Seoul      Asia
  X2022.Population
1       1425887337
2        275501339
3       1417173173
4        123951692
5        115559009
6         51815810
> head(data1 , 2)
  Country.Region X2018 X2019 X2020 X2021 X2022 Rank CCA3 Country.Territory Capital Continent
1          China 40250 41450 46360 43990 45070    1  CHN             China Beijing      Asia
2      Indonesia 12540 12520 12640 13270 14260    4  IDN         Indonesia Jakarta      Asia
  X2022.Population
1       1425887337
2        275501339
>
```

dim(data1)

Output:

```
dim(data1)
[1] 53 12
```

#Display last rows

tail(data1)

#display last 2 rows

tail(data1 , 2)

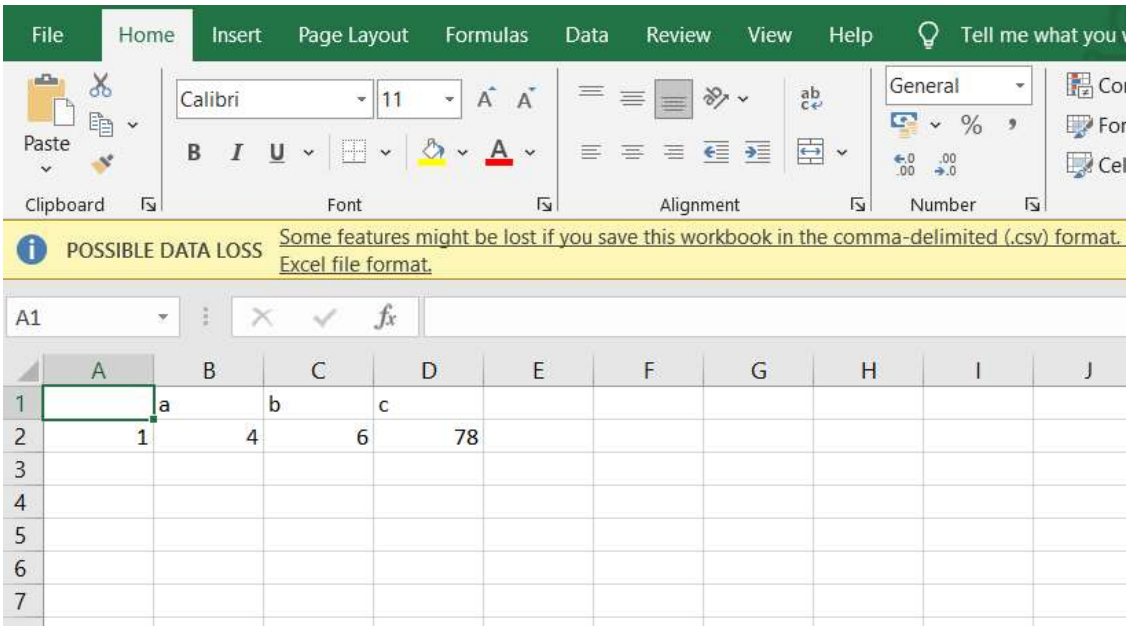
```
> tail(data1)
   Country.Region X2018 X2019 X2020 X2021 X2022 Rank CCA3 Country.Territory      Capital
48        Denmark    20    20    10    10    20  115  DNK           Denmark   Copenhagen
49        Finland    10    20    20    20    20  118  FIN           Finland     Helsinki
50    Switzerland    10    10    10    10    20  101  CHE       Switzerland         Bern
51      Argentina    10    10     0    20    10   33  ARG         Argentina Buenos Aires
52     Costa Rica    10    10    10    20    10  124  CRI        Costa Rica     San José
53        Ukraine   320   340   320   350    NA   38  UKR           Ukraine         Kiev
       Continent X2022.Population
48        Europe          5882261
49        Europe          5540745
50        Europe          8740472
51 South America         45510318
52 North America          5180829
53        Europe         39701739
> tail(data1 , 2)
   Country.Region X2018 X2019 X2020 X2021 X2022 Rank CCA3 Country.Territory  Capital
52     Costa Rica    10    10    10    20    10  124  CRI        Costa Rica San José
53        Ukraine   320   340   320   350    NA   38  UKR           Ukraine     Kiev
       Continent X2022.Population
52 North America          5180829
53        Europe         39701739
>
```

#Writing in the file

z <- data.frame(a=4 , b=6, c=78)

write.csv(z,file="data.csv")

Output:

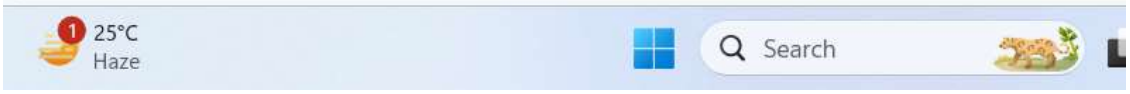


#Data Preprocessing techniques

#1. Naming & renaming the variable, adding a new variable

data1 <- mtcars data1

```
> g <- mtcars
> g
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```



Output

g <- mtcars[1:6,1:5]

g

```
> g <- mtcars[1:6,1:5]
> g
                   mpg cyl disp  hp drat
Mazda RX4         21.0   6  160 110 3.90
Mazda RX4 Wag     21.0   6  160 110 3.90
Datsun 710        22.8   4  108  93 3.85
Hornet 4 Drive    21.4   6  258 110 3.08
Hornet Sportabout 18.7   8  360 175 3.15
Valiant           18.1   6  225 105 2.76
> 
```

Output:

g <- rename(mtcars, display = disp)

g

Output:

```
> g <- rename(mtcars, display = disp)
> g
                   mpg cyl display  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6   160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6   160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4   108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6   258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8   360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6   225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8   360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4   146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4   140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6   167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C         17.8   6   167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE        16.4   8   275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL        17.3   8   275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC       15.2   8   275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8   472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8   460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8   440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128          32.4   4    78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic       30.4   4    75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla    33.9   4    71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona     21.5   4   120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger  15.5   8   318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin       15.2   8   304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28        13.3   8   350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird  19.2   8   400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9         27.3   4    79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2     26.0   4   120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa      30.4   4    95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L    15.8   8   351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino      19.7   6   145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora     15.0   8   301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E        21.4   4   121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

#adding column and data into it

g$newp1 <- g$hp * 0.5

g

```
> g$newp1 <- g$hp * 0.5
> g
                   mpg cyl display  hp drat    wt  qsec vs am gear carb newp1
Mazda RX4         21.0   6   160.0 110 3.90 2.620 16.46  0  1    4    4  55.0
Mazda RX4 Wag     21.0   6   160.0 110 3.90 2.875 17.02  0  1    4    4  55.0
Datsun 710        22.8   4   108.0  93 3.85 2.320 18.61  1  1    4    1  46.5
Hornet 4 Drive    21.4   6   258.0 110 3.08 3.215 19.44  1  0    3    1  55.0
Hornet Sportabout 18.7   8   360.0 175 3.15 3.440 17.02  0  0    3    2  87.5
Valiant           18.1   6   225.0 105 2.76 3.460 20.22  1  0    3    1  52.5
Duster 360        14.3   8   360.0 245 3.21 3.570 15.84  0  0    3    4 122.5
Merc 240D         24.4   4   146.7  62 3.69 3.190 20.00  1  0    4    2  31.0
Merc 230          22.8   4   140.8  95 3.92 3.150 22.90  1  0    4    2  47.5
Merc 280          19.2   6   167.6 123 3.92 3.440 18.30  1  0    4    4  61.5
Merc 280C         17.8   6   167.6 123 3.92 3.440 18.90  1  0    4    4  61.5
Merc 450SE        16.4   8   275.8 180 3.07 4.070 17.40  0  0    3    3  90.0
Merc 450SL        17.3   8   275.8 180 3.07 3.730 17.60  0  0    3    3  90.0
Merc 450SLC       15.2   8   275.8 180 3.07 3.780 18.00  0  0    3    3  90.0
Cadillac Fleetwood  10.4   8   472.0 205 2.93 5.250 17.98  0  0    3    4 102.5
Lincoln Continental 10.4   8   460.0 215 3.00 5.424 17.82  0  0    3    4 107.5
Chrysler Imperial   14.7   8   440.0 230 3.23 5.345 17.42  0  0    3    4 115.0
Fiat 128          32.4   4    78.7  66 4.08 2.200 19.47  1  1    4    1  33.0
Honda Civic       30.4   4    75.7  52 4.93 1.615 18.52  1  1    4    2  26.0
Toyota Corolla    33.9   4    71.1  65 4.22 1.835 19.90  1  1    4    1  32.5
Toyota Corona     21.5   4   120.1  97 3.70 2.465 20.01  1  0    3    1  48.5
Dodge Challenger  15.5   8   318.0 150 2.76 3.520 16.87  0  0    3    2  75.0
AMC Javelin       15.2   8   304.0 150 3.15 3.435 17.30  0  0    3    2  75.0
Camaro Z28        13.3   8   350.0 245 3.73 3.840 15.41  0  0    3    4 122.5
Pontiac Firebird  19.2   8   400.0 175 3.08 3.845 17.05  0  0    3    2  87.5
Fiat X1-9         27.3   4    79.0  66 4.08 1.935 18.90  1  1    4    1  33.0
Porsche 914-2     26.0   4   120.3  91 4.43 2.140 16.70  0  1    5    2  45.5
Lotus Europa      30.4   4    95.1 113 3.77 1.513 16.90  1  1    5    2  56.5
Ford Pantera L    15.8   8   351.0 264 4.22 3.170 14.50  0  1    5    4 132.0
Ferrari Dino      19.7   6   145.0 175 3.62 2.770 15.50  0  1    5    6  87.5
Maserati Bora     15.0   8   301.0 335 3.54 3.570 14.60  0  1    5    8 167.5
Volvo 142E        21.4   4   121.0 109 4.11 2.780 18.60  1  1    4    2  54.5
>
```

#check the column names

colnames(g)

Output:

```
colnames(g)

 [1] "mpg"     "cyl"     "display" "hp"      "drat"
     "wt"      "qsec"    "vs"      "am"
[10] "gear"    "carb"    "newp1"
```

#reading data from file and renaming the columns

data1 <- read.csv("data.csv", sep = "," , col.names = c("Sr.No.","DateOfJoining","Department"))data1

Output:

```
> data1 <- read.csv("data.csv", sep = "," , col.names = c("Sr.No.","DateOfJoining","Department"))
Warning message:
In read.table(file = file, header = header, sep = sep, quote = quote,  :
  header and 'col.names' are of different lengths
> data1
  Sr.No. DateOfJoining Department
1      4            6         78
```

```
> a <- c(1,2,3,NA,4)
> print(a)
```

Output:

```
[1]  1  2  3 NA  4
```

median(a)
median(a,na.rm = T)
is.na(a)
Output:

```
> median(a)
[1] NA
> median(a,na.rm = T)
[1] 2.5
> is.na(a)
[1] FALSE FALSE FALSE  TRUE FALSE
>
```

test<-read.csv("test.txt",sep = ',',header=T)

test

test1<-complete.cases(test)

test1

Output:

```
> test1<-complete.cases(test)
> test1
 [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[16]  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
>
> test<-read.csv("test.txt",sep = ',',header=T)
> test
   Campaign.Name       Date Spend..USD. X..of.Impressions  Reach X..of.Website.Clicks
1  Test Campaign  1.08.2019        3008             39550  35820                 3038
2  Test Campaign  2.08.2019        2542            100719  91236                 4657
3  Test Campaign  3.08.2019        2365             70263  45198                 7885
4  Test Campaign  4.08.2019        2710             78451  25937                 4216
5  Test Campaign  5.08.2019        2297            114295  95138                 5863
6  Test Campaign  6.08.2019          NA             42684  31489                 7488
7  Test Campaign  7.08.2019        2838             53986  42148                 4221
8  Test Campaign  8.08.2019        2916             33669  20149                 7184
9  Test Campaign  9.08.2019       45511             31598   8259                 2899
10 Test Campaign 10.08.2019        2790                NA  79632                 8125
11 Test Campaign 11.08.2019       83633             71286   3750                 2893
12 Test Campaign 12.08.2019        2831            124591  10598                 8264
13 Test Campaign       <NA>        1972             65827  49531                 7568
14 Test Campaign 14.08.2019          NA             56304  25982                 3993
15 Test Campaign 15.08.2019        2516             76219   4993                 2537
16 Test Campaign 16.08.2019        3076            106584  81389                 6800
17 Test Campaign 17.08.2019       95843             54389   1995                 1576
18 Test Campaign 18.08.2019        1979             53632  43241                 6909
19 Test Campaign 19.08.2019        2626             22521  10698                 7617
20 Test Campaign 20.08.2019        2712             39470  31893                 6050
21 Test Campaign 21.08.2019      133771            109834   5471                 1995
22 Test Campaign 22.08.2019        2899             34752  27932                 1983
23 Test Campaign 23.08.2019        2407             60286  49329                 5077
24 Test Campaign 24.08.2019        2078             36650  30489                 7156
25 Test Campaign 25.08.2019        2928            120576 105978                 3596
26 Test Campaign 26.08.2019        2311             80841  61589                 3820
27 Test Campaign 27.08.2019        2915            111469  92159                 6435
28 Test Campaign 28.08.2019        2247             54627  41267                 8144
29 Test Campaign 29.08.2019        2805             67444  43219                 7651
30 Test Campaign 30.08.2019        1977            120203  89380                 4399
   X..of.Searches X..of.View.Content X..of.Add.to.Cart X..of.Purchase
```

dataCompleteCases <- test[complete.cases(test),]

dataCompleteCases

Output:

```
> dataCompleteCases <- test[complete.cases(test),]
> dataCompleteCases
   Campaign.Name       Date Spend..USD. X..of.Impressions  Reach X..of.Website.Clicks
1  Test Campaign  1.08.2019        3008             39550  35820                 3038
3  Test Campaign  3.08.2019        2365             70263  45198                 7885
5  Test Campaign  5.08.2019        2297            114295  95138                 5863
7  Test Campaign  7.08.2019        2838             53986  42148                 4221
8  Test Campaign  8.08.2019        2916             33669  20149                 7184
12 Test Campaign 12.08.2019        2831            124591  10598                 8264
16 Test Campaign 16.08.2019        3076            106584  81389                 6800
18 Test Campaign 18.08.2019        1979             53632  43241                 6909
20 Test Campaign 20.08.2019        2712             39470  31893                 6050
23 Test Campaign 23.08.2019        2407             60286  49329                 5077
24 Test Campaign 24.08.2019        2078             36650  30489                 7156
25 Test Campaign 25.08.2019        2928            120576 105978                 3596
26 Test Campaign 26.08.2019        2311             80841  61589                 3820
27 Test Campaign 27.08.2019        2915            111469  92159                 6435
28 Test Campaign 28.08.2019        2247             54627  41267                 8144
29 Test Campaign 29.08.2019        2805             67444  43219                 7651
30 Test Campaign 30.08.2019        1977            120203  89380                 4399
   X..of.Searches X..of.View.Content X..of.Add.to.Cart X..of.Purchase
1            1946               1069              894             255
3            2572               2367             1268             578
5            2106                858              956             768
7            2733               2182             1301             890
8            2867               2194             1240             431
12           2081               1992             1382             709
16           2661               2594             1059             487
18           2824               2522              461             257
20           2061               1894             1047             730
23           2592               2004              632             473
24           2687               2427              327             269
25           2937               2551             1228             651
26           2037               1046              346             284
27           2976               2552              992             771
28           2432               1281             1009             721
29           1920               1240             1168             677
```

Install Hmisc package

#imputation

x <-c(1,2,3,NA,4,4,NA)

x

Output:

```
> x <-c(1,2,3,NA,4,4,NA)
> x
[1]  1  2  3 NA  4  4 NA
>
```

x<- impute(x, fun=mean)

x

Output:

```
> x<- impute(x, fun=mean)
Error in impute(x, fun = mean) : could not find function "impute"
> x
[1]  1  2  3 NA  4  4 NA
>
```

ADBMS LAB ASSIGNMENT 6

Classification Algorithm

Naive Bayes:

#Loading library e1071

Query:

```
install.packages("e1071")
library(e1071)
```

#Loading library e1071

Query:

```
install.packages("klaR")
library(klaR)
```

#Loading package MASS

Query:

```
install.packages("caret")
library(caret)
```

#Loading package lattice

Query:

```
install.packages("lattice")
library(lattice)
```

#Loading package ggplot2

Query:

```
install.packages("ggplot2")
library(ggplot2)
```

Query:

data("iris")

head(iris)

Output:

```
> data("iris")
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

unique(iris$Species)

Output:

```
> unique(iris$Species)
[1] setosa     versicolor virginica
Levels: setosa versicolor virginica
```

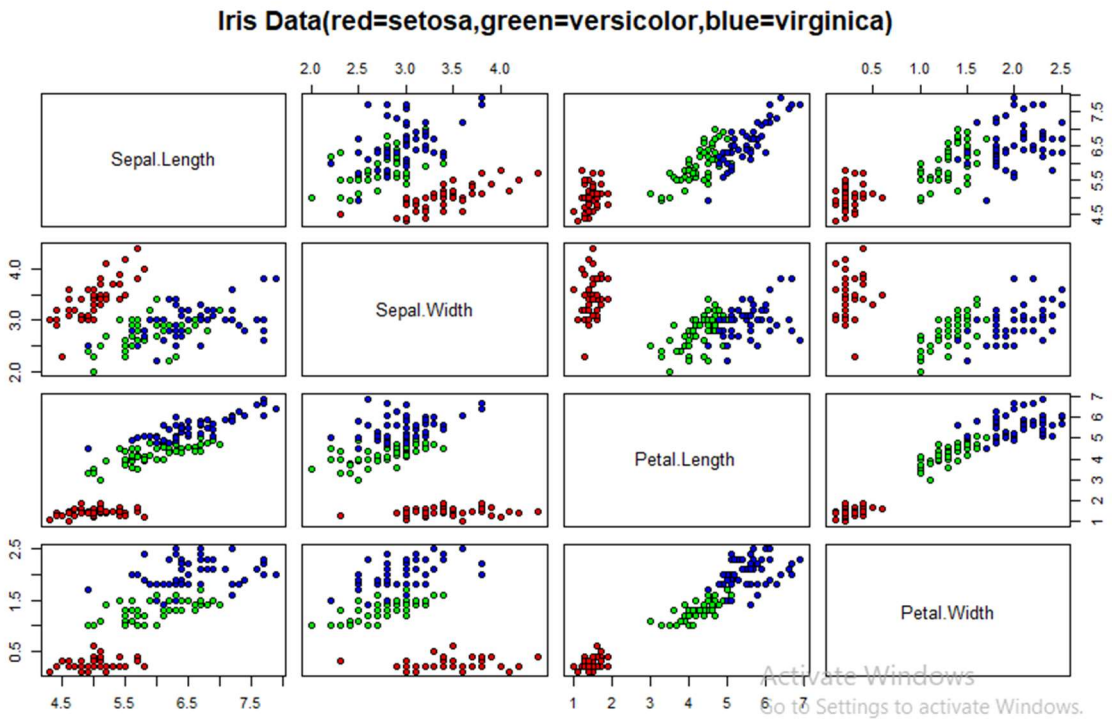
#Plot Graph:

Query:

pairs(iris[1:4],main="Iris

Data(red=setosa,green=versicolor,blue=virginica)",pch=21,bg=c("red","green","blue")[unclass(iris$Species)])

Output:



#Training a Naive Bayes Model:

Query:

```
index = sample(nrow(iris),floor(nrow(iris)*0.7))

train = iris[index,]

test = iris[-index,]

xTrain = train[,-5]

yTrain = train$Species

xtest = test[,-5]

ytest = test$Species
```

Query:

```
model <- train(xTrain,yTrain,'nb',trControl = trainControl(method = 'cv',number = 10))

model
```

Output:

```
> model
Naive Bayes

105 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 95, 94, 95, 95, 95, 94, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9427273  0.9136913
   TRUE      0.9609091  0.9410216

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

#Table gives frequency table:

Query:

prop.table(table(predict(model$finalModel,xtest)$class,ytest))

Output:

```
> prop.table(table(predict(model$finalModel,xtest)$class,ytest))
            ytest
                 setosa versicolor  virginica
  setosa     0.28888889 0.00000000 0.00000000
  versicolor 0.00000000 0.31111111 0.02222222
  virginica  0.00000000 0.04444444 0.33333333
```

ADBMS LAB ASSIGNMENT 7

 # Generate random IQ values with mean = 30 and sd = 2

```
IQ<-rnorm(40,30,2)
```

 # Sorting IQ level in ascending order

```
IQ<-sort(IQ)
```

 # Generate vector with pass and fail values of 40 students

```
result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
            1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
            0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
            1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
```

 # Data Frame

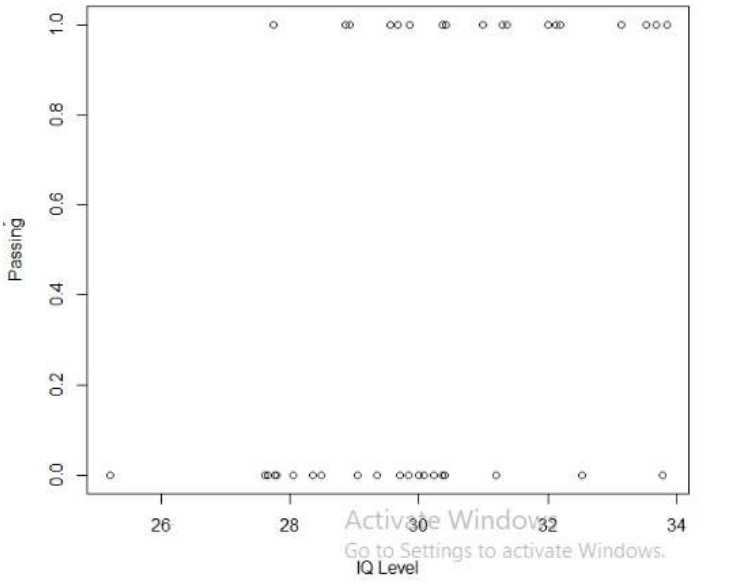
```
df <- as.data.frame(cbind(IQ, result))

df
```

```
          IQ result
1   25.20448      0
2   27.60490      0
3   27.65810      0
4   27.75078      1
5   27.76795      0
6   27.79150      0
7   28.05809      0
8   28.35858      0
9   28.49228      0
10  28.85503      1
11  28.93388      1
12  29.03965      0
13  29.05135      0
14  29.34614      0
15  29.55027      1
16  29.66554      1
17  29.71479      0
18  29.84423      0
19  29.85895      1
20  29.98851      0
21  30.07295      0
22  30.23719      0
23  30.35870      1
24  30.36794      0
25  30.39693      0
26  30.41384      1
27  30.98814      1
28  31.19469      0
29  31.30255      1
30  31.35771      1
31  32.00863      1
32  32.12619      1
33  32.19718      1
34  32.53706      0
35  33.13059      1
36  33.14557      1
37  33.52262      1
38  33.67926      1
```

Plotting IQ on x-axis and result on y-axis

plot(IQ, result, xlab = "IQ Level", ylab = "Probability of Passing")

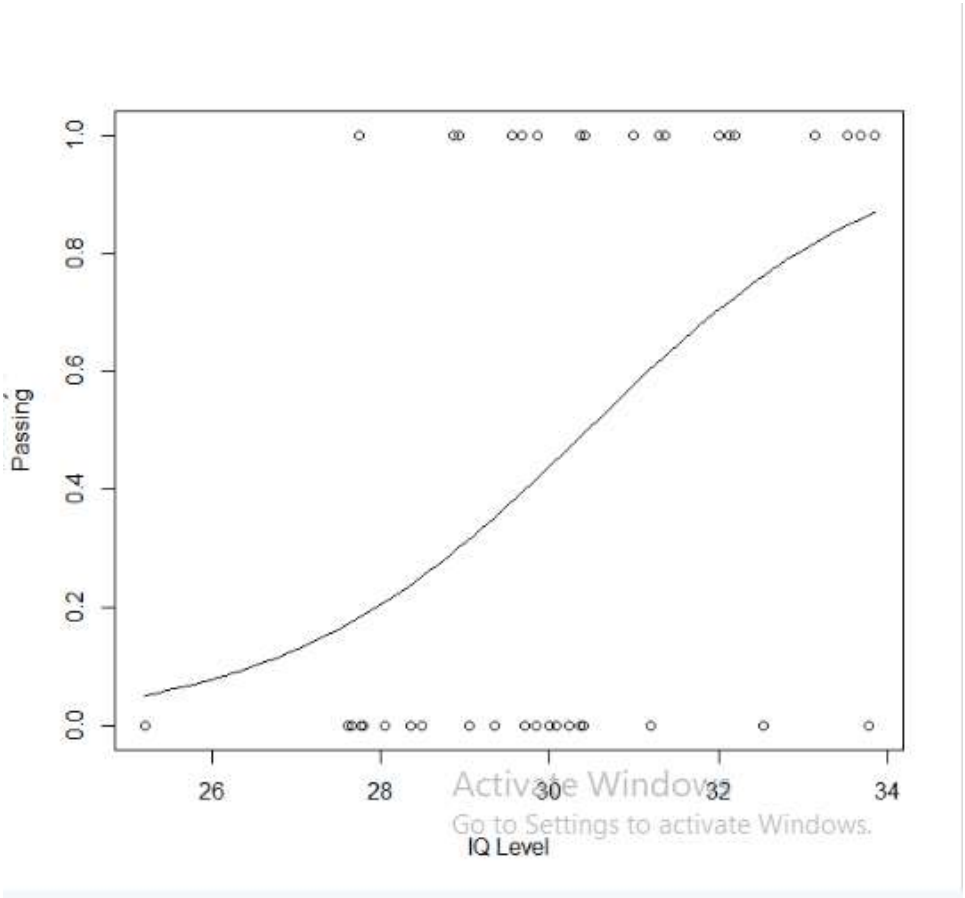


Create a logistic model

```
g = glm(result~IQ, family=binomial, df)
```

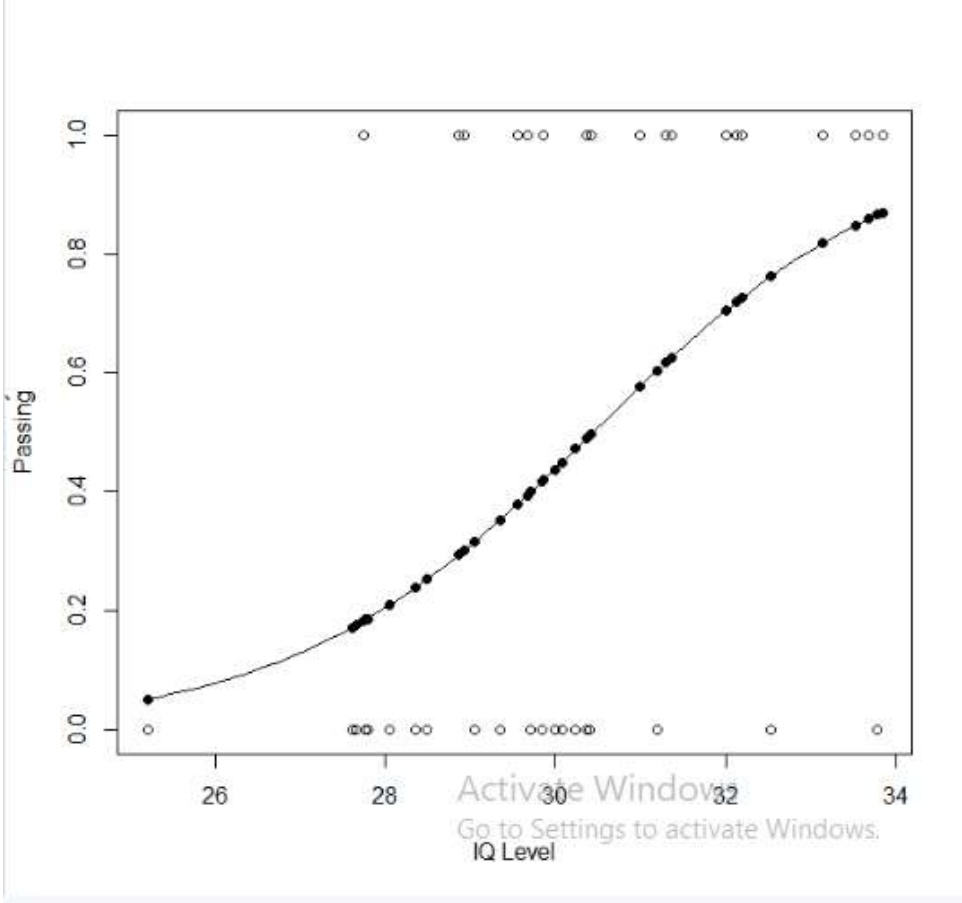
Create a curve based on prediction using the regression method

```
curve(predict(g, data.frame(IQ=x), type="resp"),
      add=TRUE)
```



Based on fit to the regression model

points(IQ, fitted(g), pch=16)



Summary of Regression model

new_IQ1 <- rnorm(10, 30, 2)

Create data frame with the new IQ values

new_data <- data.frame(IQ = new_IQ1)

Make Predictions on the new IQ values

```
predicted_probabilities <- predict(g, newdata =
                                     new_data, type = "response")
```

If you want to get predicted classes based on threshold (e.g 0.5)

```
threshold <- 0.5
predicted_classes <- ifelse(predicted_probabilities >=
                              threshold, 1, 0)
```

The variable predicted_probabilities contains the probability of passing (1)

The variable predicted_classes contains the binary predictions (0 or 1)

predicted_classes

```
> predicted_classes
 1  2  3  4  5  6  7  8  9 10
 0  1  1  1  0  0  1  0  1  1
```

new_IQ1

```
> new_IQ1
 [1] 25.09038 30.84977 30.83017 33.59615 28.34178 27.68488 31.35100 27.88464 30.68417 30.90905
```

ADBMS LAB ASSIGNMENT 8

#Apriori Algorithm:

```
data<- read.csv("data_apriori.csv")
data
```

Output:

```
> data<- read.csv("data_apriori.csv")
> data
   Customer_Id  Products
1            1     bread
2            1    butter
3            1      eggs
4            1      milk
5            4     bread
6            4    butter
7            4      eggs
8            4      milk
9            4      soda
10           2      beer
11           2     bread
12           2    cheese
13           2     chips
14           2      mayo
15           2      soda
16           3     bread
17           3    butter
18           3      eggs
19           3      milk
20           3   oranges
21           5      buns
22           5     chips
23           5      beer
24           5   mustard
25           5   pickels
26           5      soda
27           6     bread
28           6    butter
29           6 chocolate
30           6      eggs
```

#Splitting:

```
trans <- split(data$Products, data$Customer_Id, "trancsactions")
trans
```

Output:

```
> trans <- split(data$Products, data$Customer_Id, "trancsactions")
> trans
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"

$`4`
[1] "bread"  "butter" "eggs"   "milk"   "soda"

$`5`
[1] "buns"    "chips"   "beer"    "mustard" "pickels" "soda"

$`6`
[1] "bread"     "butter"    "chocolate" "eggs"      "milk"

$`7`
[1] "banana"    "chocolate" "eggs"      "milk"      "oranges"

$`8`
[1] "beer"      "bread"     "buns"      "cheese"    "chips"     "chocolate" "mayo"      "mustard"
[9] "soda"

$`9`
[1] "bread"   "butter"  "eggs"    "milk"    "milk"    "oranges" "soda"
```

#Printing first 6 records:

```
head(trans)
```

Output:

```
> head(trans)
$`1`
[1] "bread"  "butter" "eggs"   "milk"

$`2`
[1] "beer"   "bread"  "cheese" "chips"  "mayo"   "soda"

$`3`
[1] "bread"   "butter"  "eggs"    "milk"    "oranges"

$`4`
[1] "bread"  "butter" "eggs"   "milk"   "soda"

$`5`
[1] "buns"    "chips"   "beer"    "mustard" "pickels" "soda"

$`6`
[1] "bread"     "butter"    "chocolate" "eggs"      "milk"
```

#Install Package: arules

```
install.packages("arules")
library(arules)
```

#Rules and inspect:

```
rules = apriori(trans, parameter = list(support=0.5, confidence =
0.9, maxlen=3,minlen=2))
inspect(rules)
```

Output:

```
> inspect(rules)
     lhs                rhs      support   confidence coverage  lift     count
[1]  {eggs}          => {milk}   0.6000000 1          0.6000000 1.666667 9
[2]  {milk}          => {eggs}   0.6000000 1          0.6000000 1.666667 9
[3]  {butter}        => {bread}  0.6000000 1          0.6000000 1.250000 9
[4]  {butter, eggs}  => {milk}   0.5333333 1          0.5333333 1.666667 8
[5]  {butter, milk}  => {eggs}   0.5333333 1          0.5333333 1.666667 8
[6]  {bread, eggs}   => {milk}   0.5333333 1          0.5333333 1.666667 8
[7]  {bread, milk}   => {eggs}   0.5333333 1          0.5333333 1.666667 8
[8]  {butter, eggs}  => {bread}  0.5333333 1          0.5333333 1.250000 8
[9]  {bread, eggs}   => {butter} 0.5333333 1          0.5333333 1.666667 8
[10] {butter, milk}  => {bread}  0.5333333 1          0.5333333 1.250000 8
[11] {bread, milk}   => {butter} 0.5333333 1          0.5333333 1.666667 8
```