***/\*Q1.Find the number of Players & purchasers of the above-mentioned games on year month & game level.***

***• Players should be populated after the release of the game***

***• Pre Orders should be considered for purchasers but should be merged with the release month of the game***

***Q2.For the total Players and Purchasers individually, find out the segment distribution at the Game Level.***

***Q3.Make a 5x5 matrix of the players of games, showing what are the common players between any of the two games.***

***Q4. There are two tables given below. Write a query to get the output table from the input table. You can import the input table to your local SQL Server.***

***\*/***

--Q1:

select a.year,a.month,a.[GAME NAME],No\_of\_player,No\_of\_Purchaser from

(select year,month, [GAME NAME],count(distinct ACCT\_ID) as No\_of\_player from(

select \*,YEAR(player\_date) as [year],MONTH(player\_date) as [month] from

(

select ACCT\_ID,p.[GAME NAME],p.[PLAY DATE] as Player\_date

from Players p

join [Game Release Information] q on p.[GAME NAME]=q.[GAME NAME]

where p.[PLAY DATE]>=q.[RELEASE DATE] ) A)b

group by year,month,[GAME NAME]

)a

join

(

select year,month, [GAME NAME],count(distinct ACCT\_ID) as No\_of\_purchaser from(

select \*,YEAR(purchase\_date) as [year],MONTH(purchase\_date) as [month] from(

select ACCT\_ID,p.[GAME NAME],[RELEASE DATE]

, case

when [PURCHASE DATE]<[RELEASE DATE] THEN [RELEASE DATE] else [PURCHASE DATE] end as purchase\_date

from Purchases p

join [Game Release Information] q on p.[GAME NAME]=q.[GAME NAME]) A)b

group by year,month,[GAME NAME]

)b

on a.year=b.year and

a.month=b.month

and a.[GAME NAME]=b.[GAME NAME]

order by year,month,[GAME NAME]

ans::: SELECT COUNT(DISTINCT p.acct\_id) as num\_players, COUNT(DISTINCT pr.acct\_id) as num\_purchasers,

MONTH(p.purchase\_date) as month, YEAR(p.purchase\_date) as year, p.game\_name

FROM Players p

JOIN Purchases pr ON p.acct\_id = pr.acct\_id AND p.game\_name = pr.game\_name

WHERE p.purchase\_date >= (SELECT release\_date FROM Games WHERE game\_name = p.game\_name)

GROUP BY MONTH(p.purchase\_date), YEAR(p.purchase\_date), p.game\_name

--Q2:

--for individual players

select distinct p.[GAME NAME] as game,s.[SEGMENT] as segment,

count([SEGMENT]) as seg\_distribution

from Players p

left join Segment s on s.[ACCT ID]=p.ACCT\_ID

GROUP BY p.[GAME NAME],s.[SEGMENT]

ORDER BY game asc

--for individual purchaser

select distinct q.[GAME NAME] as game,t.[SEGMENT] as segment,

count([SEGMENT]) as seg\_distribution

from Purchases q

left join Segment t on t.[ACCT ID]=q.[ACCT\_ID]

GROUP BY q.[GAME NAME],t.[SEGMENT]

ORDER BY game asc

ANS::: SELECT s.segment, COUNT(DISTINCT p.acct\_id) as num\_players, COUNT(DISTINCT pr.acct\_id) as num\_purchasers,

p.game\_name

FROM Players p

JOIN Purchases pr ON p.acct\_id = pr.acct\_id AND p.game\_name = pr.game\_name

JOIN Segment s ON s.acct\_id = p.acct\_id

WHERE p.purchase\_date >= (SELECT release\_date FROM Games WHERE game\_name = p.game\_name)

GROUP BY s.segment, p.game\_name

--Q3:

--Note : I tried some other queries to find player who plays two game with same id but there is no any id which are

-- equal cuz all the acctid are distinct so we did not any result other than similar games

select [GAME NAME], count(case when [GAME NAME]='GAME - 1'Then 1 else Null end) as [GAME - 1],

count(case when [GAME NAME]='GAME - 2' Then [ACCT\_ID] else Null end) as [GAME - 2],

count(case when [GAME NAME]='GAME - 3' Then [ACCT\_ID] else Null end )as [GAME - 3],

count(case when [GAME NAME]='GAME - 4' Then [ACCT\_ID] else Null end )as [GAME - 4],

count(case when [GAME NAME]='GAME - 5' Then [ACCT\_ID] else Null end) as [GAME - 5]

from Players

group by [GAME NAME]

ORDER BY [GAME NAME]

ANS:::SELECT game1.game\_name as game\_1, game2.game\_name as game\_2, COUNT(DISTINCT p.acct\_id) as num\_common\_players

FROM Players p

JOIN Players game1 ON p.acct\_id = game1.acct\_id AND p.game\_name <> game1.game\_name

JOIN Players game2 ON p.acct\_id = game2.acct\_id AND game1.game\_name = game2.game\_name

GROUP BY game1.game\_name, game2.game\_name

--select count(\*) from (SELECT (ACCT\_ID)

--FROM Players

--WHERE [GAME NAME]='GAME - 1'

--intersect

--SELECT ACCT\_ID

--FROM Players

--WHERE [GAME NAME]='GAME - 2')t

--Q4:

create table input

(

Room int,

Status\_Date Date,

Booked int

);

insert into input values (101,'1/1/2016',0)

insert into input values (101,'1/2/2016',1)

insert into input values (101,'1/3/2016',1)

insert into input values (101,'1/4/2016',0)

insert into input values (101,'1/5/2016',1)

insert into input values (101,'1/6/2016',1)

insert into input values (101,'1/7/2016',1)

insert into input values (102,'1/1/2016',0)

insert into input values (102,'1/2/2016',1)

insert into input values (102,'1/3/2016',0)

insert into input values (102,'1/4/2016',0)

insert into input values (102,'1/5/2016',0)

insert into input values (102,'1/6/2016',0)

insert into input values (102,'1/7/2016',0)

insert into input values (102,'1/8/2016',1);

select \* from input;

--ANS1:

select Room,end\_date as Start\_Date,Status\_Date as End\_Date,Booked as Status from(

select \*,case

when Booked<>lead(Booked) Over(order by Room,Status\_Date) AND Booked=lag(Booked) Over(order by Room,Status\_Date)

AND Booked=lag(Booked,2) Over(order by Room,Status\_Date) AND Booked=lag(Booked,3) Over(order by Room,Status\_Date)

AND Booked=lag(Booked,4) Over(order by Room,Status\_Date)

then lag(Status\_Date,4)over(order by Room,Status\_Date)

when Booked<>lead(Booked) Over(order by Room,Status\_Date) AND Booked=lag(Booked) Over(order by Room,Status\_Date)

AND Booked=lag(Booked,2) Over(order by Room,Status\_Date)

then lag(Status\_Date,2)over(order by Room,Status\_Date)

when Booked<>lead(Booked) Over(order by Room,Status\_Date) AND Booked=lag(Booked) Over(order by Room,Status\_Date)

then lag(Status\_Date)over(order by Room,Status\_Date)

when (Booked=0 or Booked=1) and

Booked<>lead(Booked) Over(order by Room,Status\_Date) Then Status\_Date

when lead(Booked) Over(order by Room,Status\_Date) is Null Then Status\_Date else null end as end\_date

from input)t

where end\_date is NOT null

--Hardcoded way answer

with cte as (select \*,case when (Booked=0 or Booked=1) and

Booked<>lead(Booked) Over(order by Room,Status\_Date) Then Status\_Date

when lead(Booked) Over(order by Room,Status\_Date) is Null Then Status\_Date else Null end as end\_date

from input

),cte2 as(

select \*,datediff(d,end\_date,lead(end\_date)over(order by Room,Status\_Date)) as diff from cte

where end\_date is NOT null

),cte3 as (select \*,iif(diff<0,DATEADD(day,-(lag(diff) over(order by Room,Status\_Date))+1,end\_date),

iif(lag(diff) over(order by Room,Status\_Date)<0,end\_date,

iif(diff>1,Status\_Date,DATEADD(day,-(lag(diff) over(order by Room,Status\_Date)),lead(Status\_Date)

over(order by Room,Status\_Date))))) as new\_start\_date

from cte2)

select Room,coalesce(new\_start\_date,Status\_Date) as Start\_Date,end\_date as End\_Date,Booked as Status from cte3

ANS :::

SELECT Room, MIN(Status\_Date) as start\_date, MAX(Status\_Date) as end\_date, Booked

FROM (

SELECT Room, Status\_Date, Booked,

ROW\_NUMBER() OVER (PARTITION BY Room ORDER BY Status\_Date) -

ROW\_NUMBER() OVER (PARTITION BY Room, Booked ORDER BY Status\_Date) as grp

FROM input

) t

GROUP BY Room, Booked, grp

ORDER BY 1,2,3