Q1) Given an Order Table with the schema (id, user_id, total, created). Write a SQL Query to create a retention plot. The format for the raw data and output are given.

Week Start Date is the 1st Week in which the User_Id Placed the order, Week 0 is Unique User ids who placed their 1st Order in this week. Out of those ids, Week 1 is unique users who placed an order in 1st Week + 1, Then Week 2 is 1st Week + 2 and so on till Week 10.

Query :

```
Select W4.weekdates as
week_start_date,W3.week0,W3.week1,W3.week2,W3.week3,W3.week4,W3.week5,W3.wee
k6,W3.week7,W3.week8,W3.week9,W3.week10
from
(Select distinct W2.firstweekno, sum(case when W2.weekno -W2.firstweekno = 0 then
W2.users else null end) as 'week0',
sum(case when W2.weekno -W2.firstweekno=1 then W2.users else null end) as 'week1',
sum(case when W2.weekno -W2.firstweekno = 2 then W2.users else null end) as 'week2',
sum(case when W2.weekno -W2.firstweekno = 3 then W2.users else null end) as 'week3',
sum(case when W2.weekno -W2.firstweekno = 4 then W2.users else null end) as 'week4',
sum(case when W2.weekno -W2.firstweekno = 5 then W2.users else null end) as 'week5',
sum(case when W2.weekno -W2.firstweekno = 6 then W2.users else null end) as 'week6',
sum(case when W2.weekno -W2.firstweekno = 7 then W2.users else null end) as 'week7',
sum(case when W2.weekno -W2.firstweekno = 8 then W2.users else null end) as 'week8',
sum(case when W2.weekno -W2.firstweekno = 9 then W2.users else null end) as 'week9',
sum(case when W2.weekno -W2.firstweekno = 10 then W2.users else null end) as
'week10'


 from

(Select W1.firstweekno,W1.weekno,COUNT(Distinct userId) as users from
(Select Z.userId,case when Y.firstweekno <0 then 42+Y.firstweekno+10
when Y.firstweekno >=0 then Y.firstweekno end as firstweekno ,Z.date, case when
Z.weekno <0 then 42+Z.weekno+10
when Z.weekno >=0 then Z.weekno end as weekno from
(Select distinct T2.users,T1.weekno as firstweekno from
(select distinct date(created) as weekdates,strftime('%W', created, 'weekday 0', '-6
days')-10 as weekno from orders
where date(created)>= '2017-03-06' and date(created) <='2018-03-04')T1
inner JOIN
(select distinct userId as users,min(date(created)) as firstdate from orders group by 1 )T2
```

```
ON T2.firstdate=T1.weekdates
order by 1)Y
inner JOIN(Select  userId,date(created) as date,strftime('%W', created, 'weekday 0', '-6
days')-10 as weekno
from orders
where date(created)between '2017-03-06' and '2018-03-04')Z
ON Y.users=Z.userId order by 2)W1
group by 1,2)W2

group by 1)W3
left JOIN
(select distinct min(date(created)) as weekdates,strftime('%W', created, 'weekday 0', '-6
days')-10 as weekno from orders
where date(created)>= '2017-03-06' and date(created) <='2018-03-04' group by 2) W4
ON W3.firstweekno=W4.weekno
```

Output :first 10 rows of output

	week_start_date	week0	week1	week2	week3	week4	week5	week6	week7	week8	week9	week10
1	2017-03-06	32	5	2	6	6	4	5	5	4	4	8
2	2017-03-13	23	5	3	2	6	7	3	7	8	3	5
3	2017-03-20	28	8	2	7	3	2	4	2	2	4	6
4	2017-03-27	30	7	5	3	4	4	3	3	2	2	4
5	2017-04-03	26	10	6	5	5	6	7	3	4	6	3
6	2017-04-10	39	9	9	3	3	7	8	6	4	8	3
7	2017-04-17	39	3	6	6	4	3	5	2	2	4	4
8	2017-04-24	64	12	19	7	7	7	9	13	9	8	10
9	2017-05-01	84	12	11	9	11	12	5	7	9	3	5
10	2017-05-08	85	11	12	9	19	9	9	11	13	7	14

Q2) Given the tables Order_Timeline(schema id,order_id, message, created) & Order_Shipment Table(schema id, order_id,actual_dispatch_date,created) , write a SQL Query to find

a. % orders shipped before first message date(OTIF)
b. % orders shipped on first message date+1(OTIF+1)
c. % orders shipped on first message date+2(OTIF+2)
d. %orders shipped after that(OTIF+>2)

Order_Timeline contains the message for expected dispatch date, Order_shipment gives you the real dispatch date. They are combined using order_id.

Query :

```
Select X.OTIF_val,round(100* (1.0*X.orders)/(1.0*Y.orders),2) as'%orders' from (Select
case when date(P.actual_dispatch_date)<date(P.firstmessage_date) then 'OTIF'
when julianday(P.actual_dispatch_date)-julianday(P.firstmessage_date) in (0,1) then
'OTIF+1'
when julianday(P.actual_dispatch_date)-julianday(P.firstmessage_date) ==2 then 'OTIF+2'
when julianday(P.actual_dispatch_date)-julianday(P.firstmessage_date) > 2 then
'OTIF+>2' end as 'OTIF_val', COUNT( distinct P.order_id) as orders from
(Select T1.order_id,T1.firstmessage_date,date(T2.actual_dispatch_date) as
actual_dispatch_date from
(select distinct order_id,min(date(json_extract(message,'$.dispatch_date'))) as
firstmessage_date from order_timeline  group by 1)T1
inner join order_shipment T2 on T1.order_id=T2.order_id
where T2.actual_dispatch_date is not null)P
group by 1) X join
(Select COUNT( distinct P.order_id) as orders from
(Select T1.order_id,T1.firstmessage_date,date(T2.actual_dispatch_date) as
actual_dispatch_date from
(select distinct order_id,min(date(json_extract(message,'$.dispatch_date'))) as
firstmessage_date from order_timeline  group by 1)T1
inner join order_shipment T2 on T1.order_id=T2.order_id
where T2.actual_dispatch_date is not null)P) Y
ON 1=1
```

Output:

	OTIF_val	%orders
1	OTIF	90.14
2	OTIF+1	6.85
3	OTIF+2	1.15
4	OTIF+>2	1.86

Q3) A company record its employees movement In and Out of office in a table with 3 columns
(Employee id, Action (In/Out), Created)
There is NO sample data for this question. You only need to submit the queries

Employee id	Action	Created
1	In	2019-04-01 12:00:00
1	Out	2019-04-01 15:00:00
1	In	2019-04-01 17:00:00
1	Out	2019-04-01 21:00:00

- First entry for each employee is **"In"**
- Every **"In"** is succeeded by an **"Out"**
- No data gaps and, employee can work across days

1. **Find number of employees inside the Office at current time**

Query:

```
Select COUNT(distinct P.employees) as employees from (Select distinct EmployeeId
as employees, Sum(case when action='in' then 1 Else 0 End) as 'IN',
Sum(case when action='out' then 1 Else 0 End) as 'OUT' from emp
where created<=datetime('now')
group by 1)P
where P.'IN' >P.OUT
```

2. **Find number of employees inside the Office at "2019-05-01 19:05:00"**

Query :

```
Select COUNT(distinct P.employees) as employees from (Select distinct EmployeeId
as employees, Sum(case when action='in' then 1 Else 0 End) as 'IN',
Sum(case when action='out' then 1 Else 0 End) as 'OUT' from emp
where created<=datetime('2019-05-01 19:05:00')
```

```
group by 1)P
where P.'IN' >P.OUT
```

3. **Measure amount of hours spent by each employee inside the office since the day they started (Account for current shift if she/he is working)**

Query:

```
Select Z.employee,SUM(Z.hrs) as totalhrsspent from(Select distinct
X.employee,(julianday(X.intime)-julianday(Y.outtime))*24.0 as hrs FROM
(Select * from
(Select distinct EmployeeId as employee, case when action='in' then created Else
NULL End as 'intime',
case when action='out' then created Else NULL End as 'outtime' from emp)P
where P.intime is not null)X
inner JOIN
(Select * from
(Select distinct EmployeeId as employee, case when action='in' then created Else
NULL End as 'intime',
case when action='out' then created Else NULL End as 'outtime' from emp)P
where P.outtime is not null)Y
ON X.employee=Y.employee)Z
group by 1
```

4. **Measure amount of hours spent by each employee inside the office between "2019-04-01 14:00:00" and "2019-04-02 10:00:00"**

Query :

```
Select Z.employee,SUM(Z.hrs) as totalhrsspent from(Select distinct
X.employee,(julianday(X.intime)-julianday(Y.outtime))*24.0 as hrs FROM
(Select * from
```

```
(Select distinct EmployeeId as employee, case when action='in' then created Else
NULL End as 'intime',
case when action='out' then created Else NULL End as 'outtime' from emp where
datetime(created) between '2019-04-01 14:00:00' and '2019-04-02 10:00:00')P
where P.intime is not null )X
inner JOIN
(Select * from
(Select distinct EmployeeId as employee, case when action='in' then created Else
NULL End as 'intime',
case when action='out' then created Else NULL End as 'outtime' from emp where
datetime(created) between '2019-04-01 14:00:00' and '2019-04-02 10:00:00')P
where P.outtime is not null)Y
ON X.employee=Y.employee)Z
group by 1
```