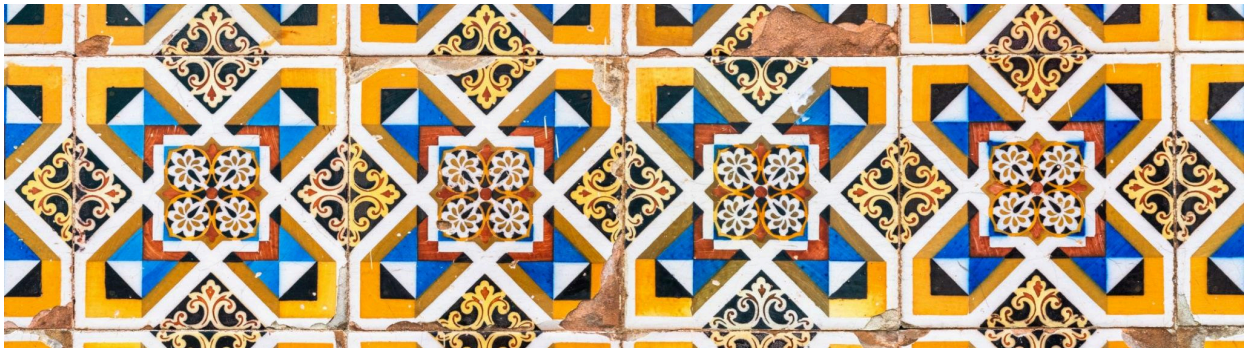

Complex Interview Questions

Q1. Derive Point table for icc tournament.

INPUT		
Team_1	Team_2	Winner
India	SL	India
SL	Aus	Aus
SA	Eng	Eng
Eng	NZ	NZ
Aus	India	India

Output			
Team_Name	Matches_played	no_of_wins	no_of_losses
India	2	2	0
SL	2	0	2
SA	1	0	1
Eng	2	1	1
Aus	2	1	1
NZ	1	1	0

-- Solution 1

```
SELECT TEAM_Name , COUNT(*) Matched_played,
        SUM(CASE WHEN TEAM_NAME= Winner THEN 1 ELSE 0 END)
No_of_Wins,
        SUM(CASE WHEN TEAM_NAME != Winner THEN 1 ELSE 0 END)
No_of_Losses
FROM
(select distinct (team_1) as Team_Name from icc_world_cup
UNION
select distinct (team_2) from icc_world_cup) a
JOIN icc_world_cup b ON a.Team_Name = b.Team_1 OR a.Team_Name =
b.Team_2
GROUP BY Team_Name;
```

Code for creating table :

```
create table icc_world_cup
(
```

```
Team_1 varchar(20),
Team_2 varchar(20),
Winner varchar(20)
);

INSERT INTO icc_world_cup values('India','SL','India');
INSERT INTO icc_world_cup values('SL','Aus','Aus');
INSERT INTO icc_world_cup values('SA','Eng','Eng');
INSERT INTO icc_world_cup values('Eng','NZ','NZ');
INSERT INTO icc_world_cup values('Aus','India','India');

select * from icc_world_cup;
```

Q2. Find new and repeat customers everyday.

output - order_date, no_of_new_customers, no_of_repeat_customers, revenue_new_customers, revenue_repeat_customers

– Solution

```
SELECT order_date,
        COALESCE(SUM(CASE WHEN days = 0 THEN 1 END),0)
no_of_new_customers,
        COALESCE(SUM(CASE WHEN days != 0 THEN 1 END),0)
no_of_repeat_customers,
        COALESCE(SUM(CASE WHEN days = 0 THEN order_amount
END),0) Revenue_new_customers,
        COALESCE(SUM(CASE WHEN days != 0 THEN order_amount
END),0) Revenue_repeat_customers
FROM(
        select b.order_date,b.order_amount,
DATEDIFF(day,a.first_order_date,b.order_date) days
        FROM(
        select customer_id , min(order_date) first_order_date from
customer_orders group by customer_id) a
        JOIN customer_orders b ON b.customer_id =a.customer_id
        )aa
GROUP BY order_date
;
```

Output:

	order_date	no_of_new_customers	no_of_repeat_customers	Revenue_new_customers	Revenue_repeat_customers
1	2022-01-01	3	0	6600	0
2	2022-01-02	2	1	4900	2000
3	2022-01-03	1	2	3000	4000

Code for creating table :

```
create table customer_orders (
order_id integer,
customer_id integer,
order_date date,
order_amount integer
);

insert into customer_orders values(1,100,cast('2022-01-01' as
date),2000),(2,200,cast('2022-01-01' as
date),2500),(3,300,cast('2022-01-01' as date),2100)
,(4,100,cast('2022-01-02' as date),2000),(5,400,cast('2022-01-02' as
date),2200),(6,500,cast('2022-01-02' as date),2700)
,(7,100,cast('2022-01-03' as date),3000),(8,400,cast('2022-01-03' as
date),1000),(9,600,cast('2022-01-03' as date),3000)
;

select * from customer_orders
```

Q3. Scenario based Interviews Question for Product companies.

Find total visits by customers, most_visited_floor by customer, resources used.

	name	address	email	floor	resources
1	A	Bangalore	A@gmail.com	1	CPU
2	A	Bangalore	A1@gmail.com	1	CPU
3	A	Bangalore	A2@gmail.com	2	DESKTOP
4	B	Bangalore	B@gmail.com	2	DESKTOP
5	B	Bangalore	B1@gmail.com	2	DESKTOP
6	B	Bangalore	B2@gmail.com	1	MONITOR

	name	total_visits	most_visited_floor	resources_used
1	A	3	1	CPU,DESKTOP
2	B	3	2	DESKTOP,MONITOR

– Solution

```
WITH CTE AS(
        SELECT * , RANK() OVER(partition by floor order by no_visit_floor
desc) rn
        FROM
        (
        select *
                ,count(*) over(partition by name) total_visit
                ,count(*) over(partition by name,floor) no_visit_floor
        from entries
        )a
),
distinct_res as
 (select distinct name, resources from entries)

SELECT a.name, total_visit, floor as most_visited_floor
        ,string_agg(resources,',')
FROM
        (
        SELECT name, floor, total_visit
        FROM CTE
        WHERE rn=1
        GROUP BY name, floor, total_visit
        ) a
INNER JOIN distinct_res b ON a.name = b.name
group by a.name, total_visit, floor
;
```

Code for creating table :

```
create table entries (
name varchar(20),
address varchar(20),
email varchar(20),
floor int,
resources varchar(10));

insert into entries
values ('A','Bangalore','A@gmail.com',1,'CPU'),
```

```
('A','Bangalore','A1@gmail.com',1,'CPU'),
('A','Bangalore','A2@gmail.com',2,'DESKTOP')
,('B','Bangalore','B@gmail.com',2,'DESKTOP'),
('B','Bangalore','B1@gmail.com',2,'DESKTOP'),
('B','Bangalore','B2@gmail.com',1,'MONITOR');

Select * from entries;
```

Q4. Write a query to provide the date for nth occurance of sunday in the future from given data

```
-- datepart
/* sunday -1  monday -2 ….. friday -6  saturday -7 */
```

– Solution

```
declare @today_date as date = GETDATE();
declare @n int =3;
--set @today_date = GETDATE(); --today's date
--set @n = 3;
select dateadd(week, @n-1,dateadd(day,8
-datepart(weekday,@today_date),@today_date));
```

Q5. The pareto principle

The pareto principle states that for many outcomes roughly 80% of consequences comes from 20% of cause.ex.

```
--1 . 80 % of the productivity come from 20% of the employee
--2. 80% of sales comes from 20% of customers
--3. 80% of decisions in a meeting are made in 20% of the time.
--4. 80% of sales comes from 20% of products and services.
```

– Solution

```
WITH product_80_pct_sales as (
SELECT product_id, product_sale
        ,SUM(product_sale) over() total_sale
        ,0.8*SUM(product_sale) over() total_sale_80_pct
        ,SUM(product_sale) over(order by product_sale desc
                ROWS between unbounded preceding and current row)
running_sum
        , ROUND((SUM(product_sale) over(order by product_sale desc
```

```
                ROWS between unbounded preceding and current row)) /
(SUM(product_sale) over())*100,0)
                as sales_running_sum_pct
        ,(ROW_number() over(order by product_sale desc)*1.0/count(*)
over())*100 product_count_pct
FROM(
        select product_id, ROUND(sum(sales),2) product_sale
                from orders
        group by product_id
        )a
)
SELECT product_id,
        product_sale,
        running_sum as sales_running_sum,
        sales_running_sum_pct,
        product_count_pct
FROM product_80_pct_sales
WHERE sales_running_sum_pct <= 80;
```

Q6. Write a query to find person id, name, number of friends, sum of marks of person who have friends with total score greater than 100.

– Solution

```
 select a.personid,a.name, count(a.fid) no_of_friends, sum(b.score)
sum_friends_score
FROM
(
select p.personid,p.name, f.fid
FROM person p
JOIN friend f
ON p.personid = f.pid
) a
JOIN person b
ON a.fid = b.personid
GROUP by a.personid, a.name
HAVING sum(b.score) > 100;
```

Code for creating table :

```
 drop table friend
Create table friend (pid int, fid int)
insert into friend (pid , fid ) values ('1','2');
insert into friend (pid , fid ) values ('1','3');
insert into friend (pid , fid ) values ('2','1');
insert into friend (pid , fid ) values ('2','3');
insert into friend (pid , fid ) values ('3','5');
insert into friend (pid , fid ) values ('4','2');
insert into friend (pid , fid ) values ('4','3');
insert into friend (pid , fid ) values ('4','5');
drop table person
create table person (PersonID int,Name varchar(50),  Score int)
insert into person(PersonID,Name ,Score) values('1','Alice','88')
insert into person(PersonID,Name ,Score) values('2','Bob','11')
insert into person(PersonID,Name ,Score) values('3','Devis','27')
insert into person(PersonID,Name ,Score) values('4','Tara','45')
insert into person(PersonID,Name ,Score) values('5','John','63')

select * from person
select * from friend
```

Q7. trip and users (LeetCode Hard questions)

Write a query to find the cancellation rate of requests with unbanned users (both client and drivers must not be banned) each day between "2013-10-01" and "2013-10-03". Round cancellation rate to two decimal points.
The cancellation rate is computed by dividing the number of canceled (by client or driver) requests with unbanned users by the total number of requests with unbanned users on that day.

– Solution

```
select request_at,
        count(*) total_trips,
        count(CASE WHEN status in
('cancelled_by_driver','cancelled_by_client')
                                THEN 1 ELSE null END)cancelled_trips,
        1.0*count(CASE WHEN status in
('cancelled_by_driver','cancelled_by_client')
                                THEN 1 ELSE null END) / count(*) *100
cancelled_rate_pct
```

```
from trips t
INNER JOIN users c ON c.users_id =t.client_id
INNER JOIN users d ON d.users_id =t.driver_id
WHERE c.banned = 'No' and d.banned = 'No'
group by request_at;
```

Code for creating table :

```
Create table  Trips (id int, client_id int, driver_id int, city_id int, status varchar(50), request_at varchar(50));
Create table Users (users_id int, banned varchar(50), role varchar(50));

insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('1', '1', '10', '1', 'completed', '2013-10-01');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('2', '2', '11', '1', 'cancelled_by_driver', '2013-10-01');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('3', '3', '12', '6', 'completed', '2013-10-01');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('4', '4', '13', '6', 'cancelled_by_client', '2013-10-01');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('5', '1', '10', '1', 'completed', '2013-10-02');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('6', '2', '11', '6', 'completed', '2013-10-02');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('7', '3', '12', '6', 'completed', '2013-10-02');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('8', '2', '12', '12', 'completed', '2013-10-03');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('9', '3', '10', '12', 'completed', '2013-10-03');
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values ('10', '4', '13', '12', 'cancelled_by_driver', '2013-10-03');

insert into Users (users_id, banned, role) values ('1', 'No', 'client');
insert into Users (users_id, banned, role) values ('2', 'Yes', 'client');
insert into Users (users_id, banned, role) values ('3', 'No', 'client');
insert into Users (users_id, banned, role) values ('4', 'No', 'client');
insert into Users (users_id, banned, role) values ('10', 'No', 'driver');
insert into Users (users_id, banned, role) values ('11', 'No', 'driver');
insert into Users (users_id, banned, role) values ('12', 'No', 'driver');
insert into Users (users_id, banned, role) values ('13', 'No', 'driver');
```

Q8. Write a sql query to find the winner in each group.

The winner in each group is the player who scored maximum total points within the group. In case of a tie, the lowest player_id wins

```
select * from players;
select * from matches;
```

100 %

Results | Messages

	player_id	group_id
2	25	1
3	30	1
4	45	1
5	10	2
6	35	2
7	50	2
8	20	3
9	40	3

	match_id	first_player	second_player	first_score	second_score
1	1	15	45	3	0
2	2	30	25	1	2
3	3	30	15	2	0
4	4	40	20	5	2
5	5	35	50	1	1

– Solution

```
WITH cte as(
        SELECT player_id, SUM(score) total_score
        FROM(
                select match_id, first_player as player_id, first_score as
score
                FROM matches
                UNION ALL
                select match_id, second_player, second_score
                FROM matches
                )a
        GROUP BY player_id
),
cte1 as(
         SELECT cte.*, p.group_id,
                RANK() OVER(partition by group_id order by total_score
desc,cte.player_id) as rnk
```

```
        from cte
        JOIN players p
        ON cte.player_id = p.player_id
       )
SELECT group_id, player_id,total_score
FROM cte1
WHERE rnk =1;
```

	group_id	player_id	total_score
1	1	15	3
2	2	35	1
3	3	40	5

Code for creating table :

```
create table players
(player_id int,
group_id int);

insert into players values (15,1);
insert into players values (25,1);
insert into players values (30,1);
insert into players values (45,1);
insert into players values (10,2);
insert into players values (35,2);
insert into players values (50,2);
insert into players values (20,3);
insert into players values (40,3);

create table matches
(
match_id int,
first_player int,
second_player int,
first_score int,
second_score int);

insert into matches values (1,15,45,3,0);
insert into matches values (2,30,25,1,2);
insert into matches values (3,30,15,2,0);
insert into matches values (4,40,20,5,2);
```

```
insert into matches values (5,35,50,1,1);

select * from players;
select * from matches;
```

Q9. MARKET ANALYSIS

Write a SQL query to find each seller , wether the brand of the second items (by date) they sold is their favourite brand. if a seller sold less than two items, report the answer for that seller as no. o/p

seller_id	2nd_item_fav_brand
1	yes/ no
2	yes/ no

Users , orders, items table as below:

	user_id	join_date	favorite_brand
1	1	2019-01-01	Lenovo
2	2	2019-02-09	Samsung
3	3	2019-01-19	LG
4	4	2019-05-21	HP

	order_id	order_date	item_id	buyer_id	seller_id
1	1	2019-08-01	4	1	2
2	2	2019-08-02	2	1	3
3	3	2019-08-03	3	2	3
4	4	2019-08-04	1	4	2
5	5	2019-08-04	1	3	4
6	6	2019-08-05	2	2	4

	item_id	item_brand
1	1	Samsung
2	2	Lenovo
3	3	LG
4	4	HP

– Solution

```
-- seller sold multiple brand
 -- each seller has their fav brand
 -- find if second item sold by seller is fav brand
 -- if seller sold less than 2 items then no o/p
 WITH order_cte as (
        select seller_id, item_id
        FROM(
```

```
                    select *
                            ,RANK() OVER (Partition by seller_id order by
order_date asc) as rnk
                    from orders) a
          WHERE rnk =2)
select u.user_id,
         CASE WHEN i.item_brand = u.favorite_brand THEN 'Yes' ELSE
'NO' END as nd_item_fav_brand
from order_cte oc
JOIN items i on oc.item_id = i.item_id
RIGHT JOIN users u on u.user_id = oc.seller_id;
```

	user_id	nd_item_fav_brand
1	1	NO
2	2	Yes
3	3	Yes
4	4	NO

Code for creating table :

```
DROP TABLE if exists users
create table users (
user_id        int     ,
 join_date      date    ,
 favorite_brand  varchar(50));

 DROP TABLE if exists orders
 create table orders (
 order_id       int     ,
 order_date     date    ,
 item_id        int     ,
 buyer_id       int     ,
 seller_id      int  );

create table items (
 item_id        int     ,
 item_brand     varchar(50));

 insert into users values
(1,'2019-01-01','Lenovo'),(2,'2019-02-09','Samsung'),(3,'2019-01-19','LG'),(
4,'2019-05-21','HP');
 insert into items values (1,'Samsung'),(2,'Lenovo'),(3,'LG'),(4,'HP');
```

```
insert into orders values
(1,'2019-08-01',4,1,2),(2,'2019-08-02',2,1,3),(3,'2019-08-03',3,2,3),(4,'2019
-08-04',1,4,2)
 ,(5,'2019-08-04',1,3,4),(6,'2019-08-05',2,2,4);

select * from users;
select * from orders;
select * from items;
```

Q10. Arrange the table according to status and dates as start and end dates

	date_value	status
1	2019-01-01	success
2	2019-01-02	success
3	2019-01-03	success
4	2019-01-04	Fail
5	2019-01-05	Fail
6	2019-01-06	success

Input Table

	start_date	end_date	status
1	2019-01-01	2019-01-03	success
2	2019-01-04	2019-01-05	Fail
3	2019-01-06	2019-01-06	success

Out Put

– Solution

```
WITH cte as (
        select *
                ,ROW_NUMBER() over(partition by status order by
date_value asc) rn
                ,dateadd(day,-1 * ROW_NUMBER() over(partition by status
order by date_value asc), date_value) as group_date
        from tasks
)
select min(date_value) start_date
        ,max(date_value) end_date
```

```
        ,status
from cte
group by status, group_date
order by start_date
;
```

Code for creating table :

```
drop table if exists tasks
create table tasks(
        date_value date,
        status varchar(10)
        );
insert into tasks
values('2019-01-01', 'success'),
('2019-01-02', 'success'),
('2019-01-03', 'success'),
('2019-01-04', 'Fail'),
('2019-01-05', 'Fail'),
('2019-01-06', 'success');
select * from tasks;
```

Q.11

– Solution

Code for creating table :

Q12.

– Solution

Code for creating table :