```
# RANK WITHOUT Window function,
SELECT a1.Name, a1.Sales, COUNT (a2.Sales) Sales_Rank
FROM Total_Sales a1, Total_Sales a2
WHERE a1.Sales < a2.Sales OR (a1.Sales=a2.Sales AND a1.Name = a2.Name)
GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;

# Shortest Distance in a Line
SELECT MIN(ABS(p1.x-p2.x)) as shortest FROM point p1 JOIN point p2 on p1.x <> p2.x
order by p1.x;

# Monthly Percentage Difference
select id, CONCAT(DATE_FORMAT(created_at, '%Y'), '-', DATE_FORMAT(created_at,
'%m')) AS 'year_month', ROUND((SUM(value) - LAG(SUM(value), 1, 0) OVER(ORDER BY
CONCAT(DATE_FORMAT(created_at, '%Y'), '-', DATE_FORMAT(created_at, '%m')))) /
LAG(SUM(value), 1, 0) OVER(ORDER BY CONCAT(DATE_FORMAT(created_at, '%Y'), '-',
DATE_FORMAT(created_at, '%m')))*100,2) AS avg FROM sf_transactions GROUP BY 2;

# Finding User Purchases
WITH cte AS (SELECT DISTINCT abc.user_id, DATEDIFF(abc.next_purchase_date,
abc.created_at) AS intervall FROM (select user_id, created_at, LEAD(created_at, 1,
0) OVER(PARTITION BY user_id ORDER BY created_at) AS next_purchase_date from
amazon_transactions order by user_id, created_at) AS abc WHERE
abc.next_purchase_date <> 0)
SELECT DISTINCT cte.user_id FROM cte WHERE cte.intervall <= 7;

# Calculate Special Bonus
SELECT employee_id,
CASE WHEN mod(employee_id ,2)=0 THEN 0
WHEN name LIKE 'M%' THEN 0
ELSE SALARY END AS bonus FROM employees;

# Tree Node
select
    id,
    case when p_id is null then 'Root'
    when id in (select p_id from tree) then 'Inner'
    else 'Leaf' end as type
from
    tree ;


# Total Sales Amount by Year
WITH RECURSIVE cte AS
(SELECT MIN(period_start) AS date
FROM sales
UNION ALL
SELECT DATE_ADD(date, INTERVAL 1 DAY)
FROM cte WHERE date <= (SELECT MAX(period_end) FROM SALES))

SELECT s.product_id, p.product_name, LEFT(c.date,4) AS report_year,
SUM(s.average_daily_sales) AS total_amount
FROM sales s JOIN product p
ON s.product_id = p.product_id
JOIN cte c
ON s.period_start <= c.date AND s.period_end >= c.date
GROUP BY 1,2,3
ORDER BY 1,3;
```

```
# Users By Average Session Time
WITH loi AS (SELECT user_id, DATE(timestamp) AS dat ,MAX(timestamp) as 'login_time'
FROM facebook_web_log WHERE action in ('page_load') GROUP BY user_id,
DATE(timestamp)
),
lof AS (SELECT user_id, DATE(timestamp) AS dat, MIN(timestamp) as 'logoff_time'
FROM facebook_web_log WHERE action in ('page_exit') GROUP BY user_id,
DATE(timestamp))

SELECT loi.user_id, avg(time_to_sec(timediff(lof.logoff_time,loi.login_time))) AS
avg FROM loi JOIN lof ON loi.user_id = lof.user_id AND loi.dat = lof.dat GROUP BY
loi.user_id;

# Highest Cost Orders
SELECT c.first_name, SUM(o.total_order_cost) as tot_sales_per_day, o.order_date
FROM customers c JOIN orders o ON c.id = o.cust_id WHERE o.order_date BETWEEN
'2019-02-01' AND '2019-05-01' GROUP BY c.first_name, o.order_date ORDER BY
tot_sales_per_day DESC LIMIT 1 OFFSET 0;

# Consecutive Numbers
SELECT DISTINCT num as ConsecutiveNums FROM (SELECT num, lead(num) OVER(order by
id) as up, lag(num) OVER(order by id) as down FROM logs) abc WHERE num = up and num
= down;

# Employees Earning More Than Their Managers
SELECT e1.name as employee from employee e1 where e1.salary > (SELECT e2.salary
from employee e2 where e1.managerid = e2.id)

#Median Employee Salary
with cte as (
                  select *, row_number() over (partition by company order by
salary) as row_id,
                                count(salary) over (partition by company) as cnt
            from employee)
select id, company, salary
from cte
where row_id between cnt/2.0 and cnt/2.0+1;

# Tournament Winners
WITH CTE AS (SELECT first_player as player, first_score as score FROM matches
UNION ALL
SELECT second_player as player, second_score as score FROM matches)

SELECT abc.player_id, abc.group_id FROM (SELECT c.player as player_id, p.group_id
as group_id,dense_rank() OVER (PARTITION BY p.group_id ORDER BY SUM(c.score) desc,
c.player asc) as rnk
FROM cte c join players p
ON c.player = p.player_id
group by c.player) abc
WHERE abc.rnk =1;

# Exchange Seats
SELECT  id,
       CASE
            WHEN id % 2 = 0 THEN lag(student, 1) OVER (ORDER BY id)
            ELSE COALESCE(lead(student, 1) OVER (ORDER BY id), student) END as
student
FROM Seat;
```

```
# Employees With Missing Information
SELECT employee_id FROM Employees WHERE employee_id NOT IN (SELECT employee_id FROM
Salaries)
UNION
SELECT employee_id FROM Salaries WHERE employee_id NOT IN (SELECT employee_id FROM
Employees)
ORDER BY 1 ASC;

# Triangle Judegement
SELECT x,y,z,
CASE WHEN x + y <= z THEN 'No'
     WHEN y + z <= x THEN 'No'
     WHEN z + x <= y THEN 'No'
      ELSE 'Yes' END AS 'triangle' FROM triangle
```