



Vimanyu Chaturvedi
@vimanyuchaturvedi

SQL



Optimization

#1

Use 'regexp_like' to replace 'LIKE' clauses



```
SELECT *  
FROM  
    table1  
WHERE  
    lower(item_name) LIKE '%samsung%' OR  
    lower(item_name) LIKE '%xiaomi%' OR  
    lower(item_name) LIKE '%iphone%' OR  
    lower(item_name) LIKE '%huawei%'  
--and so on
```



```
SELECT *  
FROM  
    table1  
WHERE  
    REGEXP_LIKE(lower(item_name),  
        'samsung|xiaomi|iphone|huawei')
```

#2

Use 'regexp_extract' to replace 'Case-when Like'



```
SELECT  
CASE  
  WHEN concat(' ',item_name,' ') LIKE '%acer%' then 'Acer'  
  WHEN concat(' ',item_name,' ') LIKE '%advance%' then 'Advance'  
  WHEN concat(' ',item_name,' ') LIKE '%alfalink%' then 'Alfalink'  
...  
AS brand  
FROM item_list
```



```
SELECT  
  regexp_extract(item_name,'(asus|lenovo|hp|acer|dell|zyrex|...)')  
AS brand  
FROM item_list
```

#3

Convert long list of IN clause into a temporary table



```
SELECT *  
FROM Table1 as t1  
WHERE  
    itemid in (3363134,  
5189076, ..., 4062349)
```



```
SELECT *  
FROM Table1 as t1  
JOIN (  
    SELECT  
        itemid  
    FROM (  
        SELECT  
            split('3363134, 5189076, ..., ', ', ')  
            as bar  
        )  
    CROSS JOIN  
        UNNEST(bar) AS t(itemid)  
    ) AS Table2 as t2  
ON  
    t1.itemid = t2.itemid
```

#4

Always order your JOINS from largest tables to smallest tables



```
SELECT
  *
FROM
  small_table
JOIN
  large_table
ON small_table.id = large_table.id
```



```
SELECT
  *
FROM
  large_table
JOIN
  small_table
ON small_table.id = large_table.id
```

#5

Use simple equi-joins

Two tables with date string e.g., '2020-09-01', but one of the tables only has columns for year, month, day values



```
SELECT *  
FROM  
    table1 a  
JOIN  
    table2 b  
ON a.date = CONCAT(b.year, '-',  
b.month, '-', b.day)
```



```
SELECT *  
FROM  
    table1 a  
JOIN (  
    select  
        name, CONCAT(b.year, '-', b.month, '-', b.day) as date  
    from  
        table2 b  
    ) new  
ON a.date = new.date
```

#6

Always "GROUP BY" by the attribute/column with the largest number of unique entities/values



```
select
  main_category,
  sub_category,
  itemid,
  sum(price)
from
  table1
group by
  main_category, sub_category, itemid
```



```
select
  main_category,
  sub_category,
  itemid,
  sum(price)
from
  table1
group by
  itemid, sub_category, main_category
```

#7

Avoid subqueries in WHERE clause



```
select
  sum(price)
from
  table1
where
  itemid in (
    select itemid
    from table2
  )
```



```
with t2 as (
  select itemid
  from table2
)

select
  sum(price)
from
  table1 as t1
join
  t2
on t1.itemid = t2.itemid
```


#8

Use Max instead of Rank



```
SELECT *  
from (  
    select  
        userid,  
        rank() over (order by prdate desc) as rank  
    from table1  
)  
where ranking = 1
```



```
SELECT userid, max(prdate)  
from table1  
group by 1
```

#9

Other Tips

- Use `approx_distinct()` instead of `count(distinct)` for very large datasets
- Use `approx_percentile(metric, 0.5)` for median
- Avoid UNIONs where possible
- Use WITH statements vs. nested subqueries