Session 16 - SQL Query Optimization

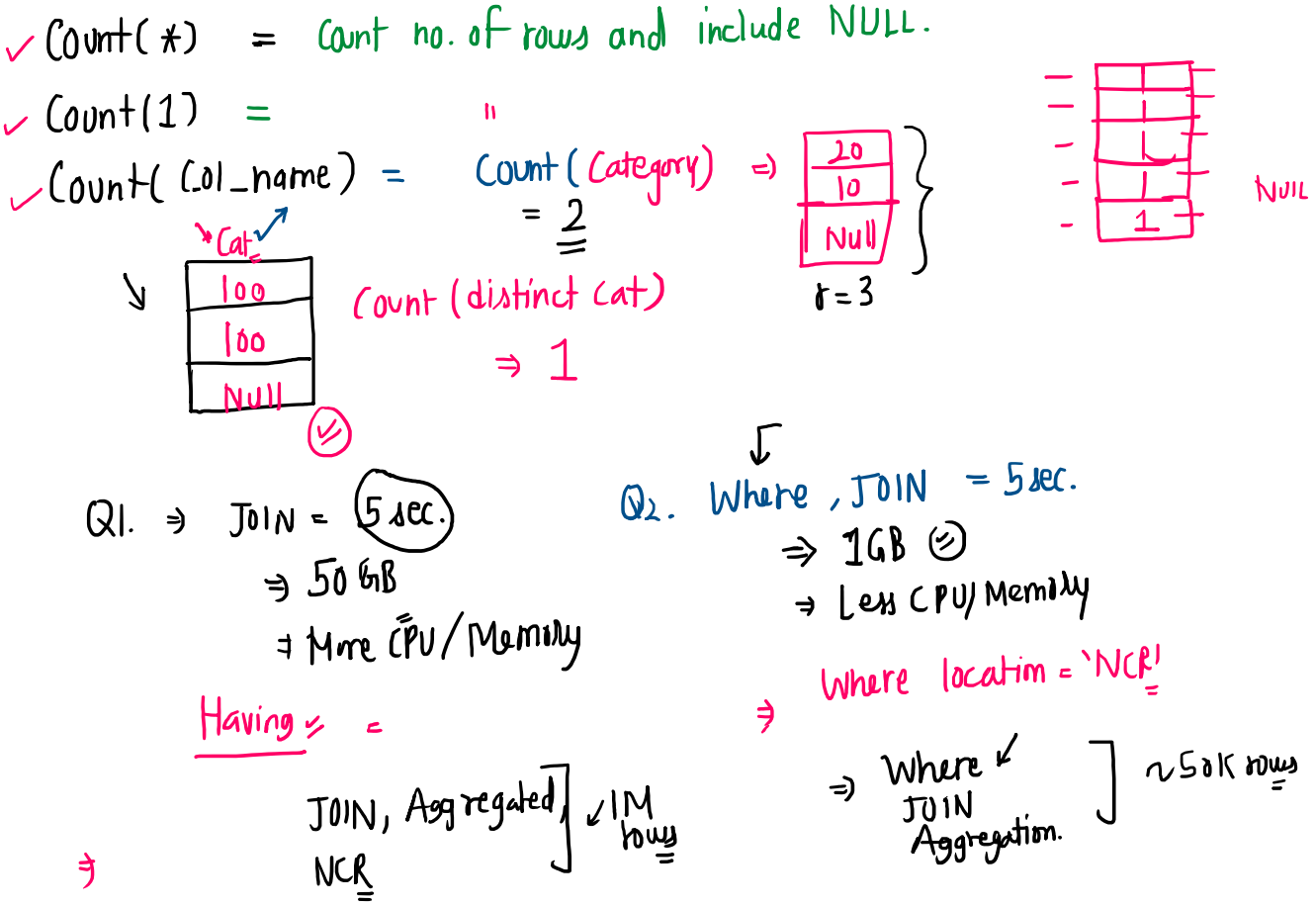
26 September 2024 21:49

DSML Aug24 Beginner 2

Agenda

1. Problem Statement
2. Busting a common myth about the COUNT()
3. Only select the columns that you really need
4. LIMIT is a trap
5. Use EXISTS() instead of COUNT()
6. Use APPROX_COUNT_DISTINCT instead of COUNT(DISTINCT)
7. Replace Self-Join with Windows Function
8. Trim your data early and often
9. Use MAX() instead of RANK()
10. Order your JOINs from larger tables to smaller tables
11. Does WHERE sequence matter?
12. Should we push ORDER BY to the end of the query?

1. Busting a common myth about the COUNT() function



2. Only select the columns that you really need

```
SELECT
    timestamp,
    number,
    transactions_root, state_root, receipts_root,
    miner,
    difficulty, total_difficulty,
    size,
    extra_data,
    gas_limit, gas_used,
    transaction_count,
```

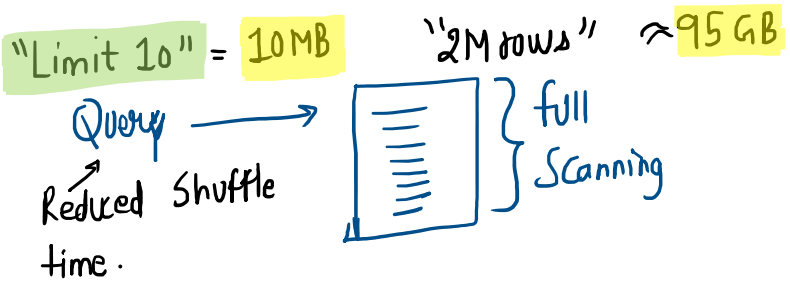
bytes shuffled reduced by a factor of 1/4.

```
    base_fee_per_gas
FROM bigquery-public-data.crypto_ethereum.blocks;
```

```
SELECT * FROM
  bigquery-public-data.crypto_ethereum.blocks;
```

3. LIMIT is a trap

↳ Speed up performance.
↳ Doesn't reduce cost.



4. Use EXISTS() instead of COUNT()

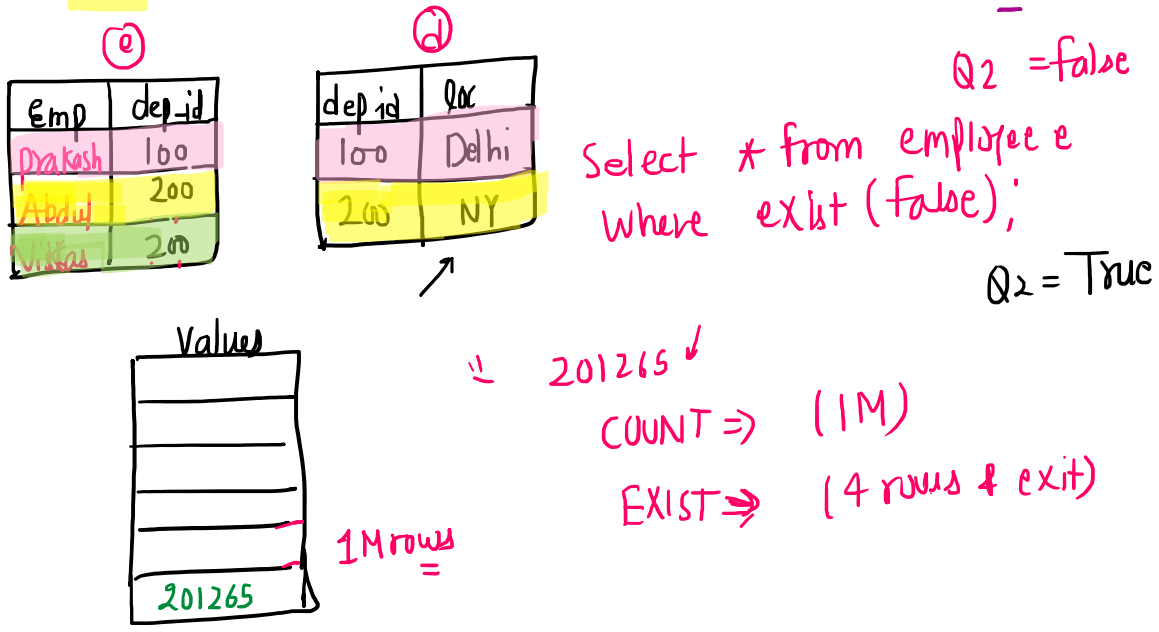
. Show all emp where dep_loc = "NY" ✓

```
Select * from employee
Where dep_id IN ( select id from department where loc = "NY");
```

More time

```
Select * from employee e
Where EXIST (select 1 from department d where d.dep_id = e.dep_id and d.loc = "NY")
```

Less time Q2



Q2 = false

Select * from employee e
Where exist (false);

Q2 = True

201265

COUNT ⇒ (1M)

EXIST ⇒ (4 rows & exit)

5. Use APPROX_COUNT_DISTINCT instead of COUNT(DISTINCT) for large datasets

↘ Count(DISTINCT u_name)

↘ ⇒ APPROX_COUNT_DISTINCT ( )

```
SELECT
  APPROX_COUNT_DISTINCT(miner)
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  timestamp BETWEEN '2015-01-01' AND '2023-12-31';
```

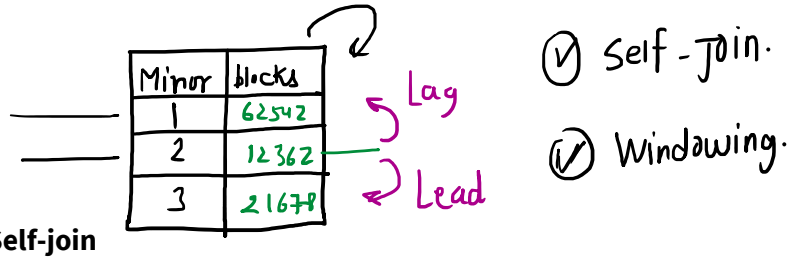
```
SELECT
  COUNT(DISTINCT miner)
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  timestamp BETWEEN '2015-01-01' AND '2023-12-31'
```

Break Till : 22:32

6. Replace Self-Join with Window Functions

Self-join usually requires more reads than window functions, therefore slower.

Question: We want to know the difference between the number of Ethereum blocks mined today and yesterday by each miner.



Self-join

```
WITH cte_table AS (
        SELECT
          DATE(timestamp) AS date,
          miner,
          COUNT(DISTINCT number) AS block_count
        FROM bigquery-public-data.crypto_ethereum.blocks
        WHERE
          DATE(timestamp) BETWEEN "2015-01-01" AND "2023-12-31"
        GROUP BY 1,2 )
      SELECT
        a.miner,
        a.date AS today,
        a.block_count AS today_count,
        b.date AS tmr,
        b.block_count AS tmr_count,
        b.block_count - a.block_count AS diff
      FROM cte_table a
      LEFT JOIN cte_table b
      ON
        DATE_ADD(a.date, INTERVAL 1 DAY) = b.date
        AND a.miner = b.miner
      ORDER BY
        a.miner, a.date
```

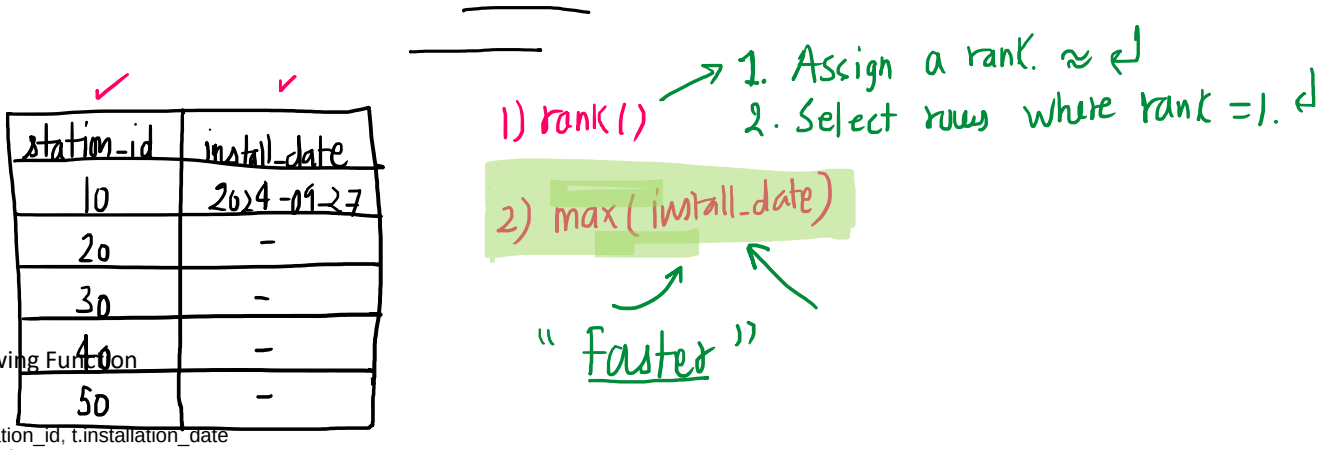
Windowing

```
WITH cte_table AS (
        SELECT
        DATE(timestamp) AS date,
        miner,
        COUNT(DISTINCT number) AS block_count
      FROM bigquery-public-data.crypto_ethereum.blocks
      WHERE
        DATE(timestamp) BETWEEN "2015-01-01" AND "2023-12-31"
      GROUP BY 1,2 )
      SELECT
        miner,
        date AS today,
        block_count AS today_count,
        LEAD(date, 1) OVER (PARTITION BY miner ORDER BY date) AS tmr,
        LEAD(block_count, 1) OVER (PARTITION BY miner ORDER BY date) AS tmr_count,
        LEAD(block_count, 1) OVER (PARTITION BY miner ORDER BY date) -
      block_count AS diff
      FROM cte_table a
```

7. **Trim your data early and often**
   Filter as early as possible

8. **Use MAX() instead of RANK()**

Question: The team has a general assumption that the older the establishment the more popular it'll be. To verify the same assumption, fetch the station ids and their respective date of installation in order starting from the one installed most recently.



Windowing Function

```
SELECT
    t.station_id, t.installation_date
  FROM (
    SELECT station_id,
        installation_date,
    RANK() OVER(PARTITION BY station_id ORDER BY installation_date DESC) AS rnk
    FROM bigquery-public-data.san_francisco.bikeshare_stations) t
  WHERE rnk = 1
  ORDER BY t.installation_date DESC;
```

Max Function

```
SELECT
    station_id,
    MAX(installation_date) AS doi
  FROM bigquery-public-data.san_francisco.bikeshare_stations
  GROUP BY 1
  ORDER BY doi DESC;
```

9. **Order your JOINs from larger tables to smaller tables**

Question: San Francisco is a big city and usually has a good number of bike trips. Find the number of bikes and docks currently available at all stations in San Francisco so that proper restocking can be done.

t1 JOIN t2

Larger

Customer = IM

Orders = ✓ Large

Bike_station =

bikeshare_status =

Case 1:

```
SELECT
   t1.station_id,
   t1.name,
   t2.bikes_available,
   t2.docks_available
FROM
    `bigquery-public-data.san_francisco.bikeshare_stations` t1
JOIN
   `bigquery-public-data.san_francisco.bikeshare_status` t2
ON
   t1.station_id = t2.station_id
WHERE
   t1.landmark = 'San Francisco';
```

Case 2: Larger table on Left side of join.

```
 SELECT
     t2.station_id,
     t2.name,
     t1.bikes_available,
     t1.docks_available
  FROM bigquery-public-data.san_francisco.bikeshare_status t1
  JOIN bigquery-public-data.san_francisco.bikeshare_stations t2
  ON t1.station_id = t2.station_id
  WHERE t2.landmark = 'San Francisco';
```

10. Does WHERE sequence matters?

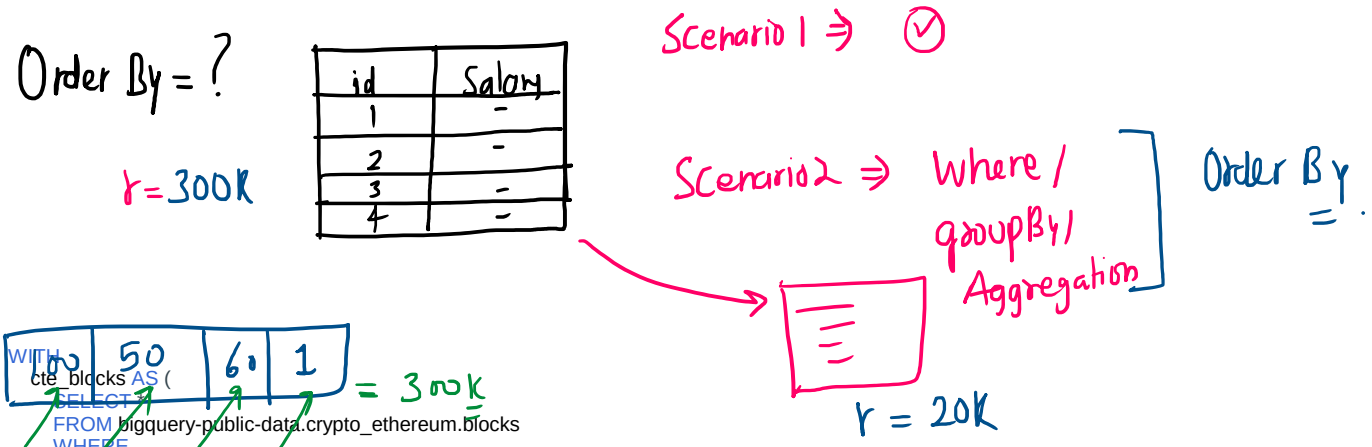
Select * from table
where miner like "%.a" Q1
and miner like "%.b"
and miner = "1001"

Select * from table
where miner = "1001" Q2
and miner like "%.a"
and miner like "%.b"

```
SELECT
  miner
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  miner LIKE '%a%'
  AND miner LIKE '%b%'
  AND miner = '0xc3348b43d3881151224b490e4aa39e03d2b1cdea';
```

```
SELECT
  miner
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  miner = '0xc3348b43d3881151224b490e4aa39e03d2b1cdea'
  AND miner LIKE '%a%'
  AND miner LIKE '%b%';
```

11. Should we push ORDER BY to the end of the query?




```
WITH
  cte_blocks AS (
    SELECT *
    FROM bigquery-public-data.crypto_ethereum.blocks
    WHERE
      DATE(timestamp) BETWEEN '2021-03-01' AND '2021-03-31'
    ORDER BY 1,2,3,4,5,6),
  cte_contracts AS (
    SELECT *
    FROM bigquery-public-data.crypto_ethereum.contracts
    WHERE
      DATE(block_timestamp) BETWEEN '2021-03-01' AND '2021-03-31'
    ORDER BY 1,2,4,5,6,7)
  SELECT *
  FROM cte_blocks b
  LEFT JOIN cte_contracts c
  ON c.block_number = b.number
  ORDER BY size, block_hash;
```

```
WITH
  cte_blocks AS (
    SELECT *
    FROM `bigquery-public-data.crypto_ethereum.blocks`
    WHERE
      DATE(timestamp) BETWEEN '2021-03-01' AND '2021-03-31'),
  cte_contracts AS (
    SELECT *
    FROM `bigquery-public-data.crypto_ethereum.contracts`
    WHERE
      DATE(block_timestamp) BETWEEN '2021-03-01' AND '2021-03-31')
  SELECT *
  FROM cte_blocks b
  LEFT JOIN cte_contracts c
  ON c.block_number = b.number
  ORDER BY size, block_hash;
```

https://www.scaler.com/hire/test/problem/23553/

```
select      jh.employee_id,
            concat(first_name,' ',last_name) 'full_name',
            job_title
from employees emp join job_history  jh
on  jh.employee_id = emp.employee_id
join jobs job
on  jh.job_id = job.job_id
where (datediff(end_date,start_date) /365) < 1
order by employee_id, job_title;
```

Sanket
exists if returns once it find the condition then what will happen to the rows below which needs to be counted

Nancy Aspin
count(*)=1 explain sir