Day 20

### Join on more than 2 tables

1)I want to conduct , email marketing in canada

get me the list of customers first\_name & last\_name who

resides in canada

use below tables

-customer(customer\_id,store\_id,first\_name,last\_name,email,

address\_id,....)

-address(address\_id,address,city\_id)

-city (city\_id,city,country\_id)

-country (country\_id,country)

>select c.customer\_id,c.first\_name,c.last\_name,cntry.country from customer c

join address a

on c.address\_id=a.address\_id

join city ct

on a.city\_id=ct.city\_id

join country cntry

on ct.country\_id=cntry.country\_id

where cntry.country='Canada';

Important scenario based questions

1)SQL query Execution order

>select d.dname,sum(e.salary) as salary from employ1 e

join dept1 d

on e.deptno=d.deptno

where e.empid!=109

group by d.dname

having sum(e.salary)>4000

order by sum(e.salary) desc

limit 1;

Mysql-->limit 1

Oracle-->fetch first rows only;

-->

1)from

2)join

3)where

4)group by

5)having

6)select

7)distinct

8)order by

9)limit

2)SQL Optimization techniques

ex- 1 query is taking too much time to execute, i want to

reduce its execution time, what are the ways to optimize this

query?

-->

1)select only the columns you need instead of using \*

2)Try to filter out unnecessary data if possible

using where clause

3)Use inner join if possible.

4)Always order your join from largest table to smallest table

large\_table join small\_table

ex- world\_covid\_data join india\_covid\_data

5)Use Indexes for faster retrival of data

6)Avoid select distinct if possible

(because distinct requires lot of processing power)

7)Run your large query during non busy hours DBA

### Index

Indexes are database objects which you can create over one

or more columns of the table..

It helps you in increasing the performance of data retrival.

used to retrive data from table more quickly..

syntax-->

create index index\_name

on table\_name(column1,column2,...);

>create index my\_index

on emp(name);

>create index my\_index

on emp(name,sal);

Indexing is nothing but one of the optimization technique

task

1)Types of Indexes

2)What is Normalization in SQL

### Fact & Dimension tables

Fact column: Contains data that is being analyze or

you can perform calculations on this columns

ex--> profit,sales,revenue,salary

Dimension column : contains values that is not useful for

calculations.

means only contains categorical information.

ex--> address,product\_name,description

Fact table--> Fact columns + Foreign key column

Dimension table --> Dimension columns + Primary key column

Types of schema

1)star schema

All the dimesnion tables connected to 1 fact table

(fact table contains foreign key column)

90% used

2)snowflake schema :

Same as start schema , but atleast 1 dimension table is not

connected to fact table

3)Galaxy schema

It is nothing but collection of star schema

1)

Cust\_id flight\_id origin destination

1 flight1 delhi hyderabad

1 flight2 hyderabad kochi

1 flight3 kochi manglore

2 flight1 mumbai ayodhya

2 flight2 ayodhya gorakhpur

output

cust\_id origin destination

1 delhi manglore

2 mumbai gorakhpur

>select distinct cust\_id,first\_value(origin)over(partition by cust\_id) as origin,

last\_value(destination) over(partition by cust\_id) as destination from flight

order by cust\_id;

first\_value--> It returns the first value from partition data

last\_value -->It returns the last value from partition data

2)find department name and its employees

(create a list of employees for each department)

>select d.dname,listagg(e.ename,',') as employees from employ1 e

join dept1 d

on e.deptno=d.deptno

group by d.dname;

#lead & lag window functions

lag function used to access value from previous row

lead function used to access value from next row

>select e.\*,lag(salary,2) over(order by empid) as prev\_salary from employ1 e;

>select e.\*,lead(salary,2) over(order by empid) as prev\_salary from employ1 e;

ex-->

>select empid,ename,salary,

case when salary>prev\_salary then 'more salary than previous employ'

when salary<prev\_salary then 'less salary than previous employ'

when salary=prev\_salary then 'same as previous employ salary'

else 'no previous employ salary'

end as remark from (

select e.\*,lag(salary)over(order by empid) as prev\_salary from employ1 e);

##

Input

val

10

2

2

null

100

null

500

null

>select val,coalesce(val,lag(val)over(order by rownum)) as result from nxtval;

-->

10

2

2

2

100

100

500

500