1. SELECT **concat**(customer\_first\_name," ", customer\_last\_name) as full\_name, \* FROM `scaler-ds-ml-de.farmers\_market.customer` limit 10
2. SELECT **upper**(concat(customer\_first\_name," ", customer\_last\_name)) as full\_name, \* FROM `scaler-ds-ml-de.farmers\_market.customer` limit 10
3. SELECT

concat(lower (customer\_first\_name) ," ", substr(lower(customer\_last\_name), 1,1))

as full\_name, \* FROM `scaler-ds-ml-de.farmers\_market.customer` limit 10

1. SELECT \* from `farmers\_market.customer`

where customer\_zip=22802

SELECT \* from `farmers\_market.customer`

where customer\_first\_name='Bob'

order by customer\_id asc

SELECT \* from `farmers\_market.customer`

where customer\_id between 10 and 20

order by customer\_id asc

SELECT \* from `farmers\_market.customer`

where customer\_id >10

and customer\_id <20

order by customer\_id asc

SELECT \* from `farmers\_market.customer`

where

(customer\_id >10 and customer\_id <20)

or customer\_id =10

order by customer\_id asc

SELECT \* from `farmers\_market.customer\_purchases\_date`

where market\_date not in ('2020-08-08', '2020-09-16')

SELECT \* from `farmers\_market.customer\_purchases\_date`

where market\_date  in ('2020-08-08', '2020-09-16')

SELECT \* from `farmers\_market.customer`

where customer\_first\_name like "%er%"

SELECT \* from `farmers\_market.customer`

where customer\_first\_name like "%er\_"

—-------------

SELECT distinct customer\_first\_name, customer\_last\_name FROM `farmers\_market.customers`

select \* from `farmers\_market.product` where product\_size IS NULL

or trim(product\_size) =""

select \* from `farmers\_market.product` where product\_size IS NOT NULL

AND trim(product\_size) !=""

select \* from `farmers\_market.customer\_purchases`

where market\_date in (SELECT market\_date FROM `farmers\_market.market\_date\_info`

WHERE market\_rain\_flag=1)

select \* from `farmers\_market.product`

where product\_category\_id in

(select product\_category\_id from `farmers\_market.product\_category`

where lower(product\_category\_name) like '%fresh%'

)

select \*,

case when product\_size='small' then 1

when product\_size='medium' then 2

when product\_size='large' then 3

else 0

end as flag

from `farmers\_market.product`

—------

select \*,

if( product\_size='small', 1 , if(product\_size='medium',2,0)) as if\_flag

from `farmers\_market.product`

—-----

# workaround to rename the column as the same name–-

select product\_id,product\_size\_fixed as product\_size, product\_category\_id from (

select \*,

if( product\_size is null, "byebye", product\_size) as product\_size\_fixed

from `farmers\_market.product`

)

—--

select \*,

case when trim(product\_size)="" or product\_size is null then "byebye" else product\_size

end as case\_when\_flag

from `farmers\_market.product`

SELECT

\* ,

CASE

WHEN quantity \* cost\_to\_customer\_per\_qty < 5

THEN "Under $5"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 5 AND 9.99

THEN "$5 - $9.99"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 10 AND 19.99

THEN "$10 - $19.99"

ELSE "Above $20"

END AS price\_basket

FROM `farmers\_market.customer\_purchases`

—---------

SELECT

\* ,

CASE

WHEN revenue < 5

THEN "Under $5"

WHEN revenue BETWEEN 5 AND 10

THEN "$5 - $10"

WHEN revenue BETWEEN 10 AND 19.99

THEN "$10 - $19.99"

ELSE "Above $20"

END AS price\_basket

from (

select \*, quantity \* cost\_to\_customer\_per\_qty as revenue

FROM `farmers\_market.customer\_purchases` )abcx

—--

Aggregation

select sum(original\_price) as totals, count (original\_price) as counter,

min(original\_price) as min\_, max(original\_price) as max\_

from `farmers\_market.vendor\_inventory`

—---

Groups

select vendor\_id, sum(quantity \* original\_price) as total from `farmers\_market.vendor\_inventory`

where vendor\_id <> 7

group by vendor\_id

order by total asc

select vendor\_id, product\_id ,

sum( quantity \* original\_price) as totals

from `farmers\_market.vendor\_inventory`

group by vendor\_id, product\_id

select vendor\_id, product\_id ,

sum( quantity \* original\_price) as totals

from `farmers\_market.vendor\_inventory`

group by vendor\_id, product\_id

order by vendor\_id asc, product\_id desc

select vendor\_id, product\_id ,

sum( quantity \* original\_price) as totals

from `farmers\_market.vendor\_inventory`

where vendor\_id <>4

group by vendor\_id, product\_id

having totals>5000

order by vendor\_id asc, product\_id desc

select vendor\_id, product\_id ,

sum( quantity \* original\_price) as totals

from `farmers\_market.vendor\_inventory`

where vendor\_id <>4

group by vendor\_id, product\_id

having totals>5000

order by vendor\_id asc, product\_id desc

—---------

Filter out vendors who brought at least 100 items from the farmer’s market over the period - 2019-05-02 and 2019-05-16.

(Also, second highest vendor by item size)

select vendor\_id, sum(quantity) as items from `farmers\_market.vendor\_inventory`

where market\_date between '2019-05-02' and '2019-05-16'

group by vendor\_id

having sum(quantity)>=100

order by sum(quantity) desc, vendor\_id asc

limit 1

offset 1

Vendor details as well

select \* from `farmers\_market.vendor` where vendor\_id in (

select vendor\_id

from `farmers\_market.vendor\_inventory`

where market\_date between '2019-05-02' and '2019-05-16'

group by vendor\_id

having sum(quantity)>=100

order by sum(quantity) desc, vendor\_id asc

limit 1

offset 1

)

Joins

select A.\*, B.vendor\_name from `farmers\_market.vendor\_inventory` as A

left join `farmers\_market.vendor` as B

on A.vendor\_id= B.vendor\_id

select A.\*, B.vendor\_name

from `farmers\_market.vendor\_inventory` as A

full join `farmers\_market.vendor` as B

on A.vendor\_id= B.vendor\_id

—------------

Joining 3 tables

select a.\* , b.customer\_first\_name , v.vendor\_name, p.product\_name

from `farmers\_market.customer\_purchases` a

left join `farmers\_market.customer` b

on a.customer\_id = b.customer\_id

left join `farmers\_market.vendor` v

on a.vendor\_id= v.vendor\_id

left join `farmers\_market.product` p

on a.product\_id= p.product\_id

—---------

Aggregations when join is used

select b.customer\_first\_name , b.customer\_last\_name, v.vendor\_name, round(sum (a.quantity),0) as total

from `farmers\_market.customer\_purchases` a

left join `farmers\_market.customer` b

on a.customer\_id = b.customer\_id

left join `farmers\_market.vendor` v

on a.vendor\_id= v.vendor\_id

left join `farmers\_market.product` p

on a.product\_id= p.product\_id

group by b.customer\_first\_name , b.customer\_last\_name, v.vendor\_name

Question: Retrieve the vendor details for all vendors who have at least one item in their inventory with a quantity greater than 5 and price under Rs. 20.

select \* from `farmers\_market.vendor` where vendor\_id in (

select vendor\_id from `farmers\_market.vendor\_inventory` a where

a.quantity > 5 and a.original\_price<20)

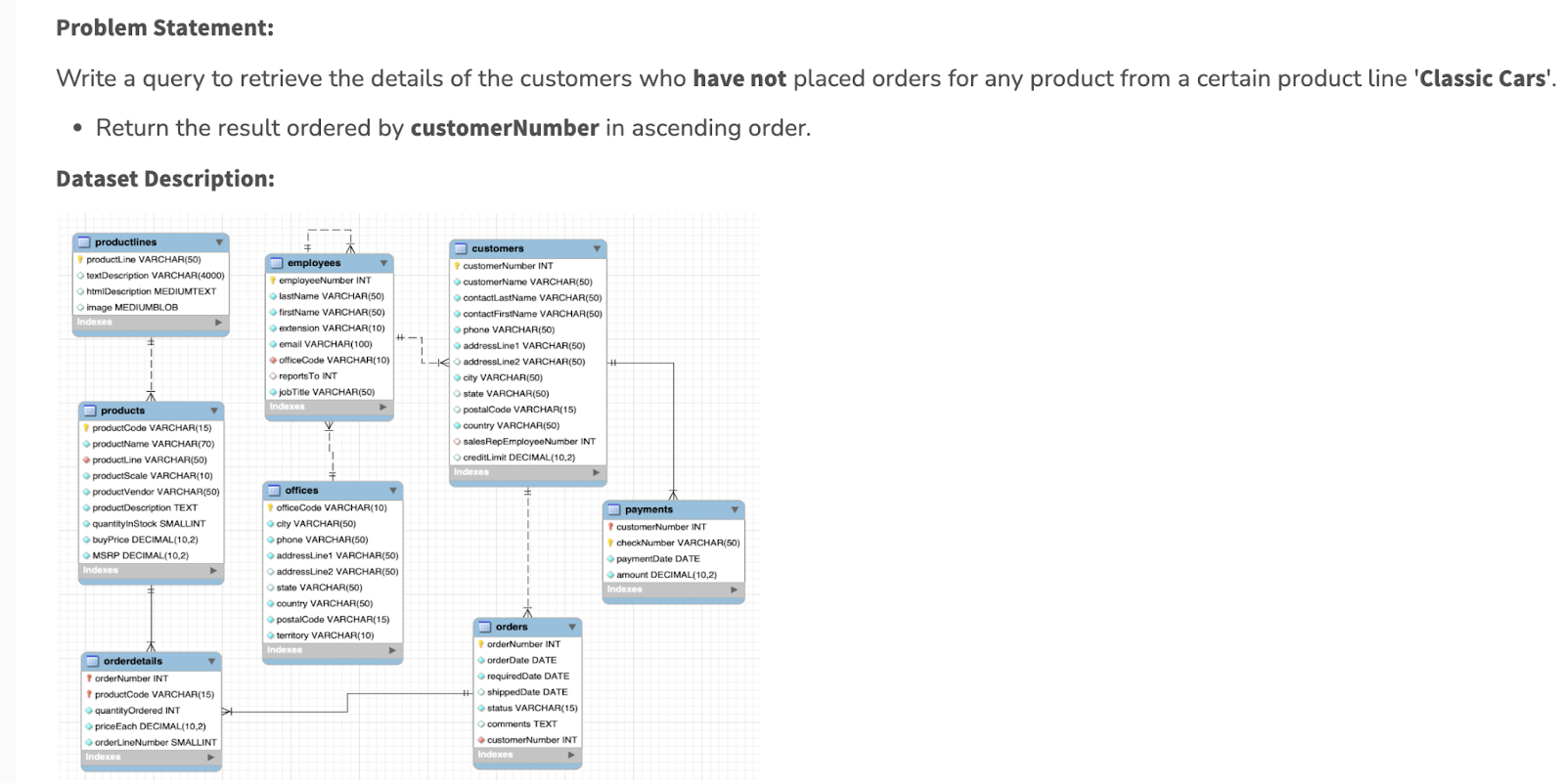
—- or—------

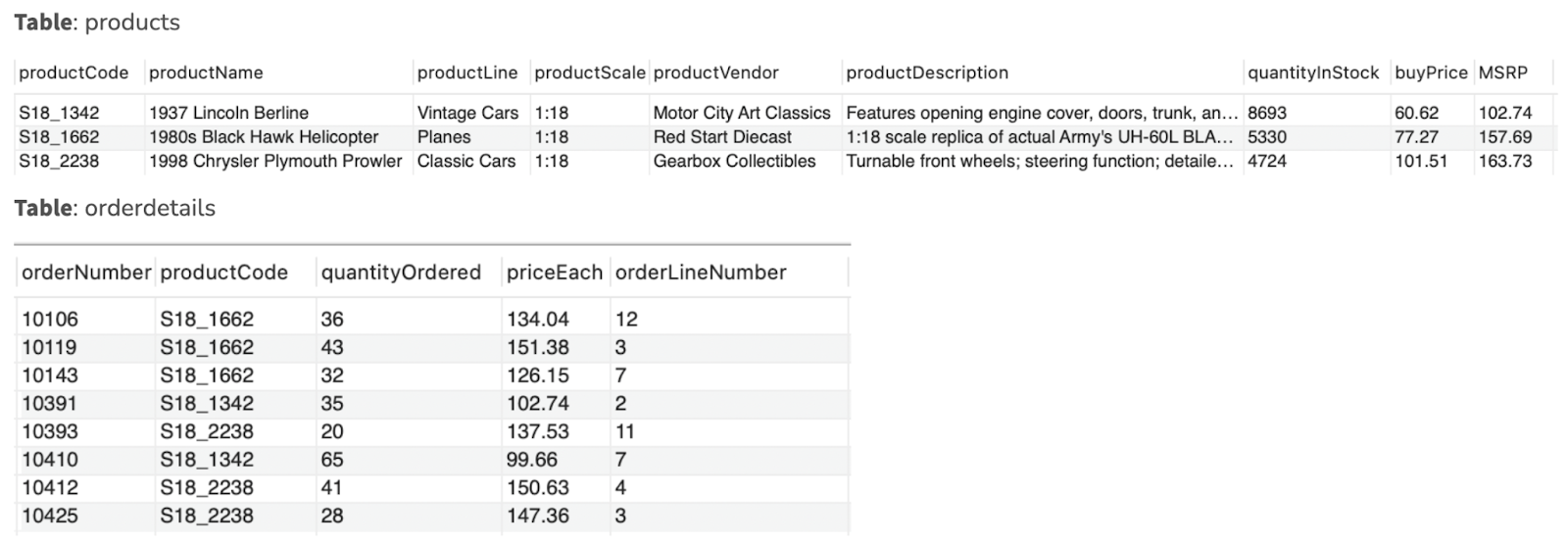
select \* from `farmers\_market.vendor` b where exists

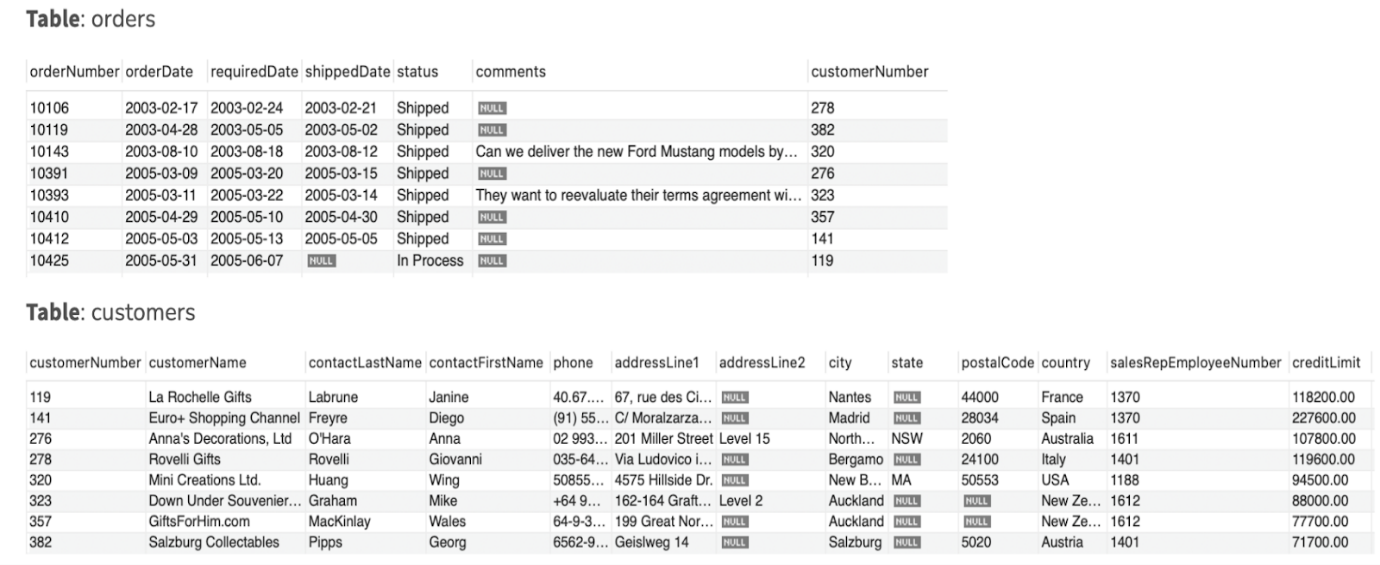
(select b.vendor\_name from `farmers\_market.vendor\_inventory` a where

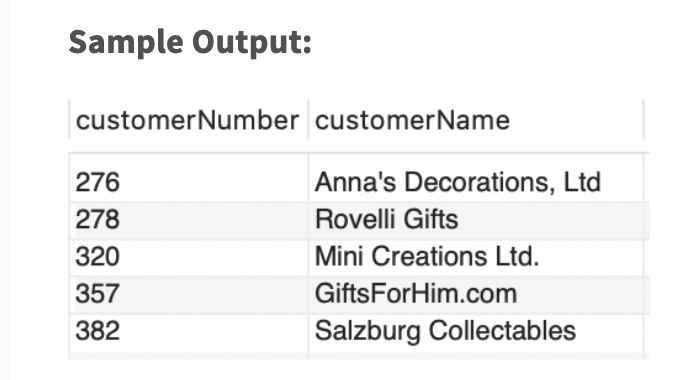
b.vendor\_id=a.vendor\_id and

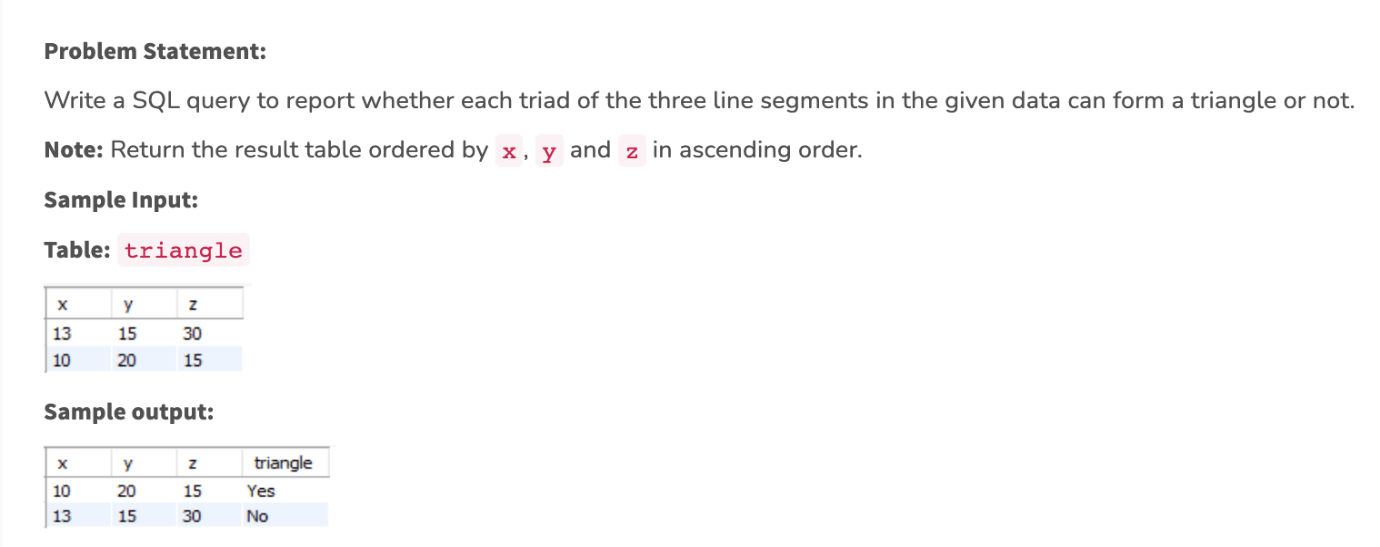
a.quantity > 5 and a.original\_price<20)

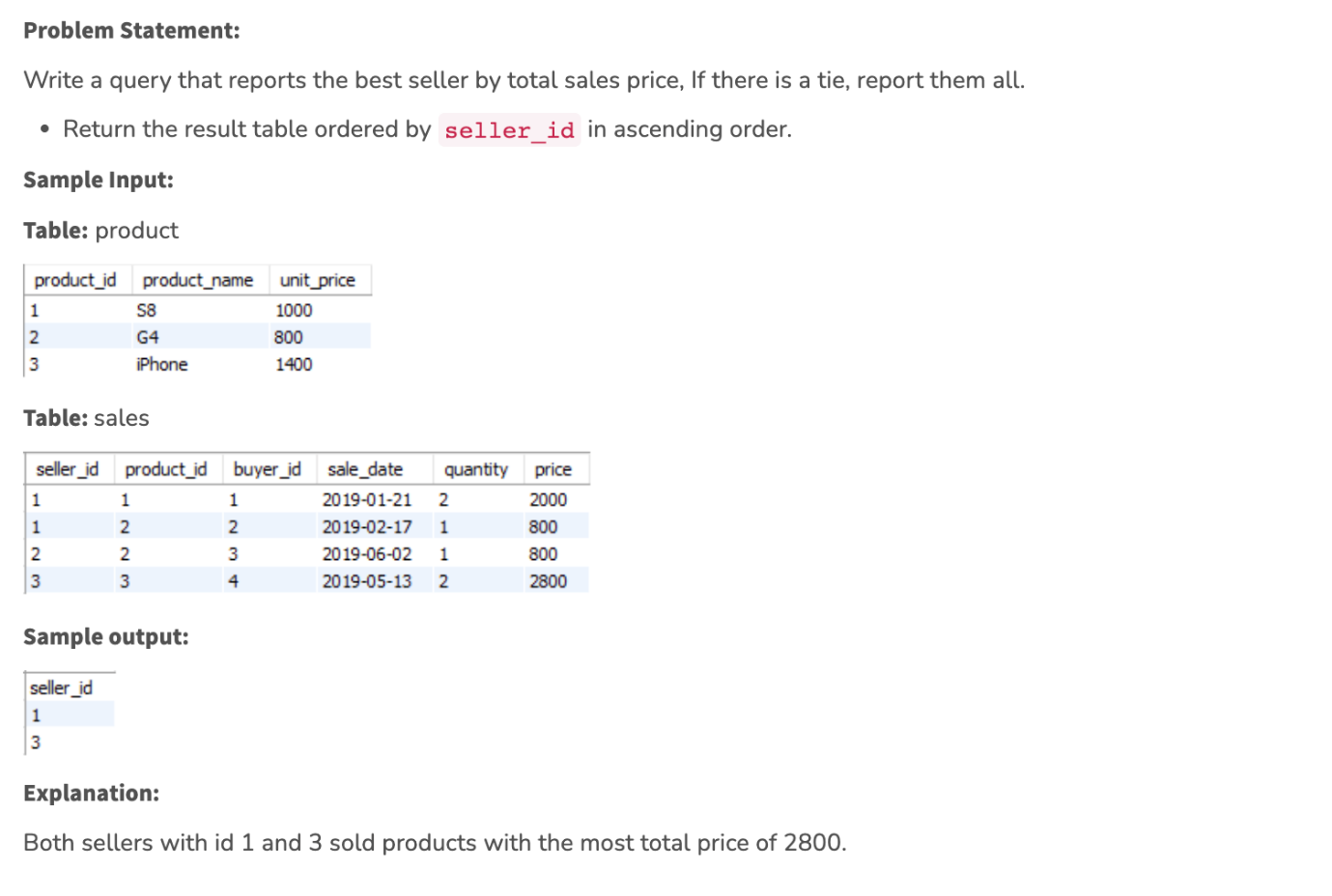


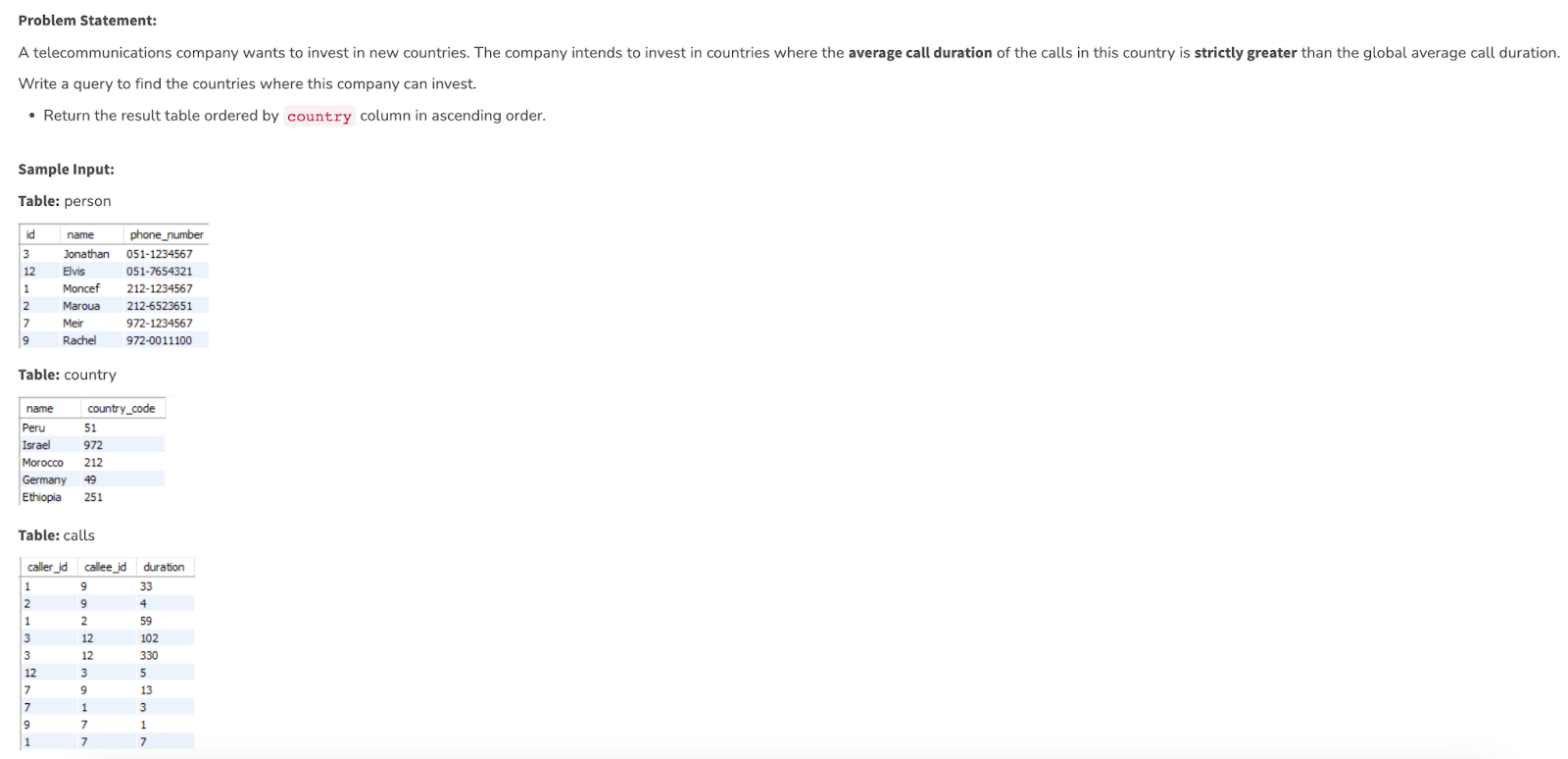


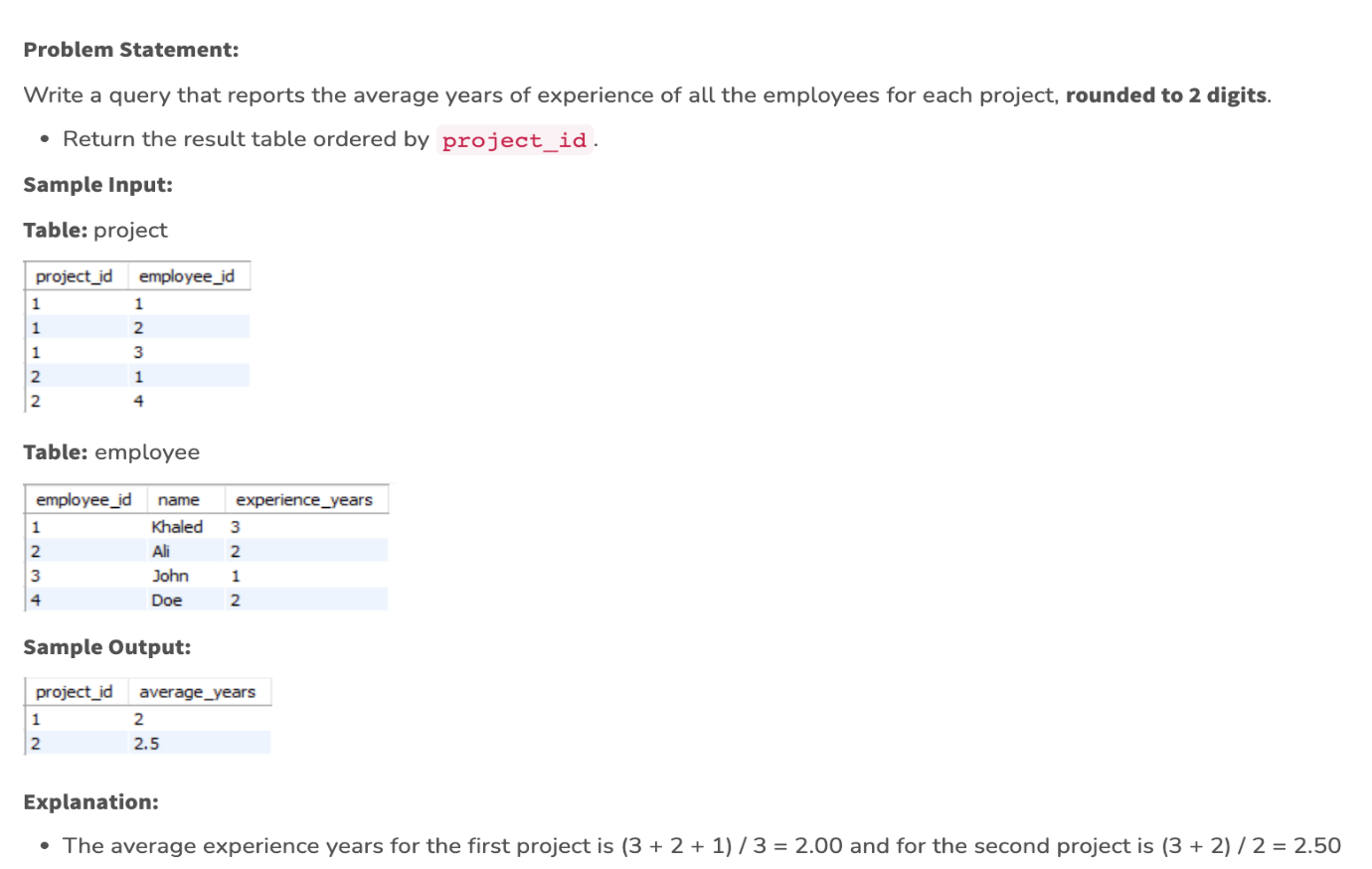


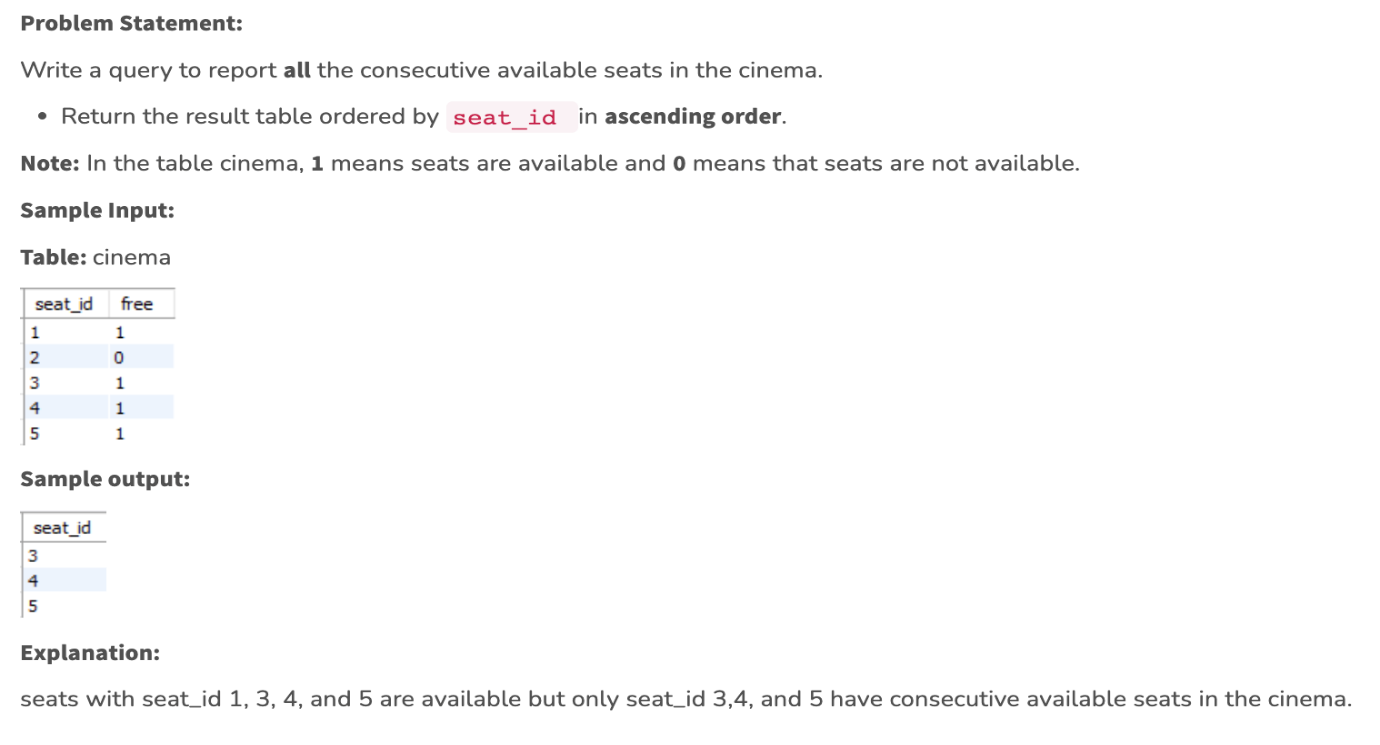








..



select \* from (

select vendor\_id, market\_date, quantity ,

max(quantity) over (partition by vendor\_id) as max\_quantity

from `farmers\_market.vendor\_inventory`

) abc

where quantity = max\_quantity

Last value of market date for each vendor and their max\_quantity

select \* from (

select \*, row\_number () over (partition by vendor\_id order by market\_date desc) as

row\_num

from (

select vendor\_id, market\_date, quantity ,

max(quantity) over (partition by vendor\_id) as max\_quantity

from `farmers\_market.vendor\_inventory`

) abc

where quantity = max\_quantity

)def

where row\_num=1

—-----Window frames

select \*, round(sum(sales) over (partition by customer\_id order by market\_date

rows unbounded preceding),2) as cum\_sum from

(

select customer\_id,market\_date, sum(quantity\*cost\_to\_customer\_per\_qty) as sales

from `farmers\_market.customer\_purchases\_date`

group by 1,2

order by customer\_id, market\_date asc

)

Last 5 mov avg:

select \*, round(sum(sales) over (partition by customer\_id order by market\_date

rows unbounded preceding),2) as cum\_sum,

round(avg(sales) over (partition by customer\_id order by market\_date

rows 4 preceding), 2) as last5mov\_Avg,

from

(

select customer\_id,market\_date, sum(quantity\*cost\_to\_customer\_per\_qty) as sales

from `farmers\_market.customer\_purchases\_date`

group by 1,2

order by customer\_id, market\_date asc

)

—----------- consolidated query—----

select \*, round(sum(sales) over (partition by customer\_id order by market\_date

rows unbounded preceding),2) as cum\_sum,

round(avg(sales) over (partition by customer\_id order by market\_date

rows 4 preceding), 2) as last5mov\_Avg,

ntile(3) over (partition by customer\_id order by market\_date) as ntilerrr,

round(avg(sales) over (partition by customer\_id order by market\_date

rows between 4 preceding and 2 following), 2) as last4precand2followingmov\_Avg,

from

(

select customer\_id,market\_date, sum(quantity\*cost\_to\_customer\_per\_qty) as sales

from `farmers\_market.customer\_purchases\_date`

group by 1,2

order by customer\_id, market\_date asc

)

where customer\_id=20

—-----

select \*, lead(sales, 1, 0) over (partition by customer\_id order by market\_date asc) as next\_day\_sales,

lag(sales, 1, 0) over (partition by customer\_id order by market\_date asc) as prev\_day\_sales,

first\_value(value\_expr)-------

from

(

select customer\_id,market\_date, round(sum(quantity\*cost\_to\_customer\_per\_qty),2)as sales

from `farmers\_market.customer\_purchases\_date`

where customer\_id=20

group by 1,2

order by customer\_id, market\_date asc

)

—---------------------

Extracting from Timestamp

select market\_start\_datetime,

extract (day from market\_start\_datetime) as day,

extract (month from market\_start\_datetime) as month,

extract (year from market\_start\_datetime) as year,

extract (quarter from market\_start\_datetime) as quarter,

extract (week from market\_start\_datetime) as week,

extract (dayofweek from market\_start\_datetime) as quadayofweek,

extract (hour from market\_start\_datetime) as hour

from `farmers\_market.datetime\_demo`

—--------------

select (parse\_datetime("%Y-%m-%d %H:%M:%S", timer\_stamp)),

extract ( month from (parse\_datetime("%Y-%m-%d %H:%M:%S", timer\_stamp))) as month

from

( select market\_date, transaction\_time,

concat (market\_date," ", transaction\_time) as timer\_stamp

from `farmers\_market.customer\_purchases`) abc

select parse\_datetime("%Y-%m-%d %H:%M:%S","2019-07-13 08:58:00") as timer\_stamp

—--

select market\_start\_datetime,

date\_add(market\_start\_datetime, INTERVAL 15 DAY) as new\_market\_start\_datetime,

date\_add(market\_start\_datetime, INTERVAL 15 HOUR) as new\_market\_start\_datetime\_HOUR,

date\_add(market\_start\_datetime, INTERVAL 30 MINUTE) as new\_market\_start\_datetime\_MINUTE,

date\_add(market\_start\_datetime, INTERVAL -15 DAY) as new\_market\_start\_datetime\_sub,

date\_sub(market\_start\_datetime, INTERVAL 15 DAY) as new\_market\_start\_datetime\_sub\_1,

from `farmers\_market.datetime\_demo`

—------

**What does the query do? select extract(day from date), sum(sales) from tbl where date between current\_date() and date\_sub(current\_date(), interval 60 days) group by extract(day from date)**

**—---**

**--Average time taken by a customer to come back/ Average Turn around time of the store**

**select avg(average) as TAT from (**

**select customer\_id, sum(days\_between)/count(days\_between) as average from (**

**select \*, date\_diff(next\_day,market\_date, day) as days\_between from (**

**select \*,**

**lead(market\_date,1 ) over (partition by customer\_id order by market\_date asc) as next\_day**

**from (**

**select customer\_id, market\_date from `farmers\_market.customer\_purchases`**

**group by 1,2**

**order by customer\_id asc, market\_date asc) abc)**

**) base**

**group by 1**

**)**

**—--------- CTEs**

**with base as (**

**select a.\*, b.vendor\_name**

**from `farmers\_market.customer\_purchases` a**

**left join**

**`farmers\_market.vendor` b**

**on a.vendor\_id=b.vendor\_id),**

**every\_day\_sales as (**

**select vendor\_name, market\_date, sum (quantity\* cost\_to\_customer\_per\_qty) as sales from base**

**group by 1,2 )**

**select a.\*, b.market\_start\_datetime from every\_day\_sales as a**

**inner join**

**`farmers\_market.datetime\_demo` b**

**on a.market\_date=b.market\_date**

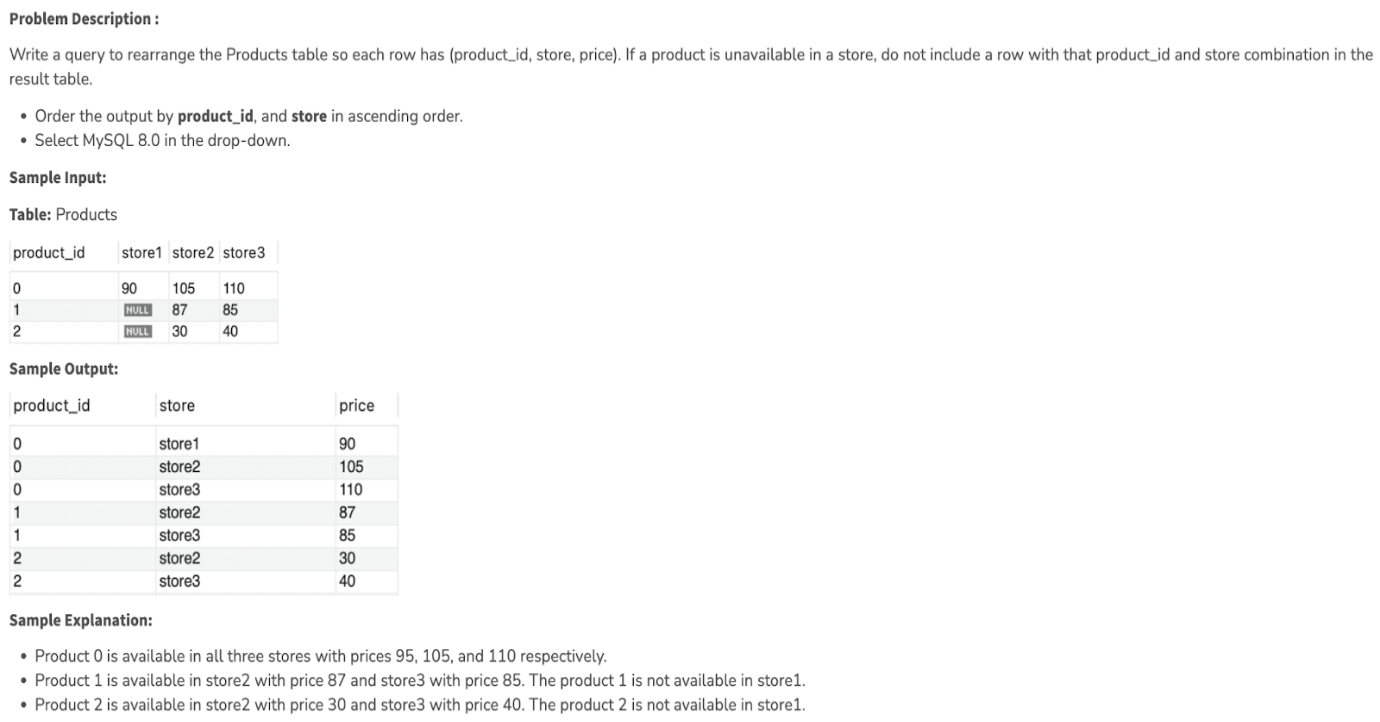
**Policy for sharing projects?**

Many of the case studies in the Scaler DSML course are built in partnership with other companies. These partner companies have shared their datasets and problem statements with Scaler and, in doing so, they have given us a huge responsibility to keep their assets protected. Hence, we ask all students to sign an NDA in the beginning of the course.

At Scaler, we also understand that our students who work hard to solve these business cases would like to showcase their solutions on their Github, Kaggle, and Resume.

As such, here are the guidelines:

1. You may **not** post the Business Case Study problem statement anywhere.
2. You may **not** share any datasets anywhere, except for the publicly available datasets from Kaggle or public domain.
3. You may upload your solutions to your Github or Kaggle profiles as long as the solution:
4. does not contain any confidential information or intellectual property owned by the partner company or Scaler.
5. does not compromise Scaler’s confidentiality agreement with the partner company
6. does not compromise the NDA signed by the students
7. does not leak the shared datasets.
8. It is acceptable if small snippets or snapshots of the datasets may appear in your solution, as long as it does not compromise the overall privacy of the dataset, the partner company, or the stakeholders of the partner company.
9. You may list the project titles, a short description of the problem, and a short summary of your insights on your resume and on relevant platforms.
10. We recommend that instead of sharing the names of partner companies you use a generic description that makes you look professional. eg:
11. Instead of saying “worked on a project from Uber”, you can say “worked on a project from **a top ride-sharing company**”



**—--------------------------**



—--- MYSQL

create database hospital;

use  hospital;

Create table patients (patient\_name varchar(30)

, phone int

);

select \* from patients;

INSERT INTO patients VALUES ('HIMANSHU','997');

INSERT INTO patients VALUES ('Junaid','abc');

INSERT INTO patients VALUES ('Suraaj Hasija',9971840678);

alter table patients

modify phone varchar(30);

alter table patients

modify patient\_name bigint;

alter table patients

rename COLUMN patient\_name TO p\_name;

alter table patients

ADD email varchar(30) unique;

INSERT INTO patients VALUES ('kuntal',8888888888, 'kuntal@scaler.com');

INSERT INTO patients VALUES ('kuntal1',8888888888, 'kuntal@scaler.com');

INSERT INTO patients VALUES ('kuntal1',8888888888, null );

Create table doctors (doctor\_name varchar(30)

, phone int primary key

);

INSERT INTO doctors VALUES ('kuntal',null);

INSERT INTO doctors VALUES ('kuntal',889988999);

update patients

set email='himanshu@scaler.com'

where p\_name='HIMANSHU';

RENAME TABLE patients to Patient;

#Delete from Patient where p\_name='suraaj';

#TRUNCATE TABLE Patient;

#drop table patient;

—------------------

use hospital;

drop table bills;

create table bills (

patient\_id INT NOT NULL,

    name varchar(40),

    bill\_date Date primary key,

    amount decimal (8,2) not null)

    PARTITION BY RANGE (YEAR(bill\_date))

(

        PARTITION P1 VALUES LESS THAN (2016),

        PARTITION P2 VALUES LESS THAN (2017),

         PARTITION P3 VALUES LESS THAN (2018),

          PARTITION P4 VALUES LESS THAN (2021))

          ;

  insert into bills values

  ( 1,'suraaj', '2015-02-10', 1000.2) ,

    ( 2,'suraaj1', '2016-02-10', 1980.2) ,

      ( 1,'suraaj2', '2017-02-10', 10300.2) ,

        ( 1,'suraaj3', '2019-02-10', 10200.2) ,

          ( 1,'suraaj4', '2018-02-10', 1500.2) ,

            ( 1,'suraaj5', '2019-03-10', 1600.2) ,

              ( 1,'suraaj6', '2020-02-10', 1700.2) ;

create table category (

category\_id INT primary key ,           01-01, 01-02

    name varchar(40),

    bill\_date Date not null,

    amount decimal (8,2) not null)

    PARTITION BY list(category\_id)

(

        PARTITION cat1 VALUES in (0,1),

        PARTITION cat2 VALUES in (2),

         PARTITION cat3 VALUES in (3),

          PARTITION cat4 VALUES in (4));

insert into category values

  ( 0,'suraaj', '2015-02-10', 1000.2) ,

    ( 1,'suraaj1', '2016-02-10', 1980.2) ,

      ( 2,'suraaj2', '2017-02-10', 10300.2) ,

        ( 3,'suraaj3', '2019-02-10', 10200.2) ,

             ( 4,'suraaj3', '2019-02-10', 10200.2)   ;

#hashing

create table hashing (

category\_id INT primary key ,

    name varchar(40),

    bill\_date Date ,

    amount decimal (8,2))

    PARTITION BY hash(category\_id)

partitions 5;

    insert into hashing values

  ( 0,'suraaj', '2015-02-10', 1000.2) ,

    ( 1,'suraaj1', '2016-02-10', 1980.2) ,

      ( 2,'suraaj2', '2017-02-10', 10300.2) ,

        ( 3,'suraaj3', '2019-02-10', 10200.2) ,

             ( 4,'suraaj3', '2019-02-10', 10200.2)  ,

                ( 5,'suraaj3', '2019-02-10', 10200.2),

                   ( 6,'suraaj3', '2019-02-10', 10200.2)   ,

                      ( 719,'suraaj3', '2019-02-10', 10200.2)   ,

                         ( 440,'suraaj3', '2019-02-10', 10200.2)   ,

                            ( 2209,'suraaj3', '2019-02-10', 10200.2)   ,

  ( 23,'suraaj3', '2019-02-10', 10200.2)   ,

          ( 15,'suraaj3', '2019-02-10', 10200.2)   ,

             ( 17,'suraaj3', '2019-02-10', 10200.2)   ,

                ( 65,'suraaj3', '2019-02-10', 10200.2)   ,

                   ( 08,'suraaj3', '2019-02-10', 10200.2)   ;

—----------

SELECT TABLE\_NAME, PARTITION\_NAME, TABLE\_ROWS, AVG\_ROW\_LENGTH, DATA\_LENGTH

FROM INFORMATION\_SCHEMA.PARTITIONS

where table\_schema='suraaj' and table\_name='bills'

select \* from bills where year(bill\_date)=2016;

select \* from category

SELECT TABLE\_NAME, PARTITION\_NAME, TABLE\_ROWS, AVG\_ROW\_LENGTH, DATA\_LENGTH

FROM INFORMATION\_SCHEMA.PARTITIONS

where table\_schema='suraaj' and table\_name='category';

Select \* from hashing;

SELECT TABLE\_NAME, PARTITION\_NAME, TABLE\_ROWS, AVG\_ROW\_LENGTH, DATA\_LENGTH

FROM INFORMATION\_SCHEMA.PARTITIONS

where table\_schema='suraaj' and table\_name='hashing';

—--- INDEXING—-

select \* from indexing;

explain select \* from indexing where first\_name='suraaj';

create index first\_name\_index on indexing(first\_name);

explain select \* from indexing where first\_name='suraaj';

—----------

select Concat(firstname, ' ', lastname) as full\_name, hiredate

from `cochin.employees`

where LOWER(title) like 'sales representative%'

—-------

select productid from `cochin.products` where unitsinstock < reorderlevel

—-----

#Find and display the details of customers who have placed more than 5 orders.

select c.customerid, c.contactname, c.companyname, c.phone, count(distinct orderid) as orders from `cochin.customers` c

inner join

`cochin.orders` o

on c.customerid=o.customerid

group by c.customerid,

having count(distinct orderid)>5

order by count(distinct orderid) desc

-------

select customerid,companyname,contactname,phone from cochin\_traders.customers where customerid in (

select customerid from cochin\_traders.orders group by customerid having count(orderid)> 5)

-------

**H/W**

An employee of ours (Margaret Peacock, EmployeeID 4) has the record of completing most orders. However, there are some customers who've never placed an order with her. Show such customers.

select

\*

from `cochin.customers` c

inner join

(select

distinct customerid

from `cochin.orders`

where employeeid <> 4)abc

on abc.customerid=c.customerid

------

select orderid, count(\*) as line\_items

from `cochin.order\_details`

group by 1

order by line\_items desc

limit 10

------

select \* from `cochin.products` limit 10

------------

select p.productname, abc.total\_quantity from `cochin.products` p

inner join

(select productid, sum(quantity) as total\_quantity

from `cochin.order\_details`

group by 1

order by total\_quantity desc

limit 5)abc

on p.productid=abc.productid

—-

with base as (

select

extract (year from o.orderdate) as year\_,

extract (month from o.orderdate) as month\_,

round(sum(od.quantity \* od.unitprice),2) as revenue

from

`cochin.order\_details` od

inner join

`cochin.orders` o

on od.orderid=o.orderid

group by 1,2

order by 1,2

),

base\_2 as (

select \*, lead(revenue, 1) over (partition by year\_ order by month\_ asc) as next\_rev from base)

select \*, round((next\_rev-revenue)/revenue\*100,2) as per\_inc from base\_2