

```

select * from guests;
select * from bookings;
select * from rooms;

-- 40 You want to update the status of rooms to 'High Demand' if the
room has been booked more than 5 times.

set sql_safe_updates = 0;

update rooms r
set r.status = 'High Demand' where exists
(select b.roomnumber, count(*) as c
from bookings b
where b.roomnumber = r.roomnumber
group by b.roomnumber
having c > 5);

update bookings c
set c.bookingstatus = 'High demand' where exists
(select roomnumber from
(select roomnumber, count(*) as c
from bookings b
group by roomnumber
having c > 5) as Raj
where Raj.roomnumber = c.roomnumber);

/* IMP - MySQL doesn't allow reading from the same table we are
updating in a subquery
like this because it can cause ambiguous or unstable results. (we can
use derived table) */

update rooms r
set r.status = 'High Demand' where r.roomnumber in
(select b.roomnumber, count(*) as c
from bookings b
where b.roomnumber = r.roomnumber
group by b.roomnumber
having c > 5);

select roomtype, roomrate
from rooms where (roomnumber, roomtype) in
(select roomnumber, roomtype
from rooms
where roomrate < 200);

-- 41 You want to calculate the total amount spent by each guest and
room type
select * from guests;
select * from bookings;
select * from rooms;

select g.guestid, r.roomtype, sum(b.totalamount) as sumtotal
from guests g join bookings b
on g.guestid = b.guestid
join rooms r
on b.roomnumber = r.roomnumber
group by g.guestid, r.roomtype;

```

-- 42 You want to rank guests based on the total amount they have spent, and retrieve only those who are ranked in the top 5.

```
with top5 as
(select guestid, sum(totalamount) as sumtotal,
dense_rank() over(order by sum(totalamount) desc) as rn
from bookings
group by guestid)
select * from top5
where rn in (1,2,3,4,5);
```

Rank :> always use aggregated data (SUM, COUNT, etc.) + window function.

-- 43 You want to update the booking status based on the total amount: if it exceeds 500, the status should be 'Premium', otherwise 'Standard'.

```
update bookings
set bookingstatus = case when totalamount > 500 then 'Premieum' else
'Standard' end;
```

-- 44 "You want to find the guests who have booked more than 5 rooms.

```
select guestid
from bookings
group by guestid
having count(roomnumber) > 5;
```

-- 45 You want to rank rooms based on their total revenue for each room type.

```
select * from guests;
select * from bookings;
select * from rooms;
```

```
select r.roomnumber, r.roomtype, sum(b.totalamount) as sumtotal,
rank() over(partition by r.roomtype order by sum(b.totalamount) desc)
as rn
from rooms r join bookings b
on r.roomnumber = b.roomnumber
group by r.roomnumber, r.roomtype;
```

-- 46 You want to classify rooms as 'High Revenue' if their total booking amount exceeds \$3000 and 'Low Revenue' otherwise

```
select roomnumber,
case
when sum(totalamount) > 3000 then 'High Revenue'
else 'Low revenue'
end as Category_Rates
from bookings
group by roomnumber;
```

```

use internshala;

-- 47 You want to delete all bookings for rooms that are no longer
available.

select * from guests;
select * from bookings;
select * from rooms;

set sql_safe_updates = 0;

delete b
from bookings b left join rooms r
on b.roomnumber = r.roomnumber
where r.roomnumber is null;

-- 48 You want to update the booking status of all bookings where the
total amount is below the average booking amount.

update bookings
set bookingstatus = 'low' where totalamount <
(select average from
(select avg(totalamount) as average
from bookings) as Raj);

select roomnumber, c from
(select roomnumber, count(*) as c
from bookings b
group by roomnumber
having c > 5) as Raj
where roomnumber in (201,402);

/* Find all products that belong to categories whose average sale price
is above
the overall average sale price, and whose brands have above-average
ratings.
Return product, category, brand, sale_price, and rating. (Bigbasket
data, not related to final test task) */

select product, category, brand, sale_price, rating from bigbasket
where category in
(select category from bigbasket
group by category having avg(sale_price) >
(select avg(sale_price) from bigbasket))
and brand in
(select brand from bigbasket
group by brand
having avg(rating) >
(select avg(rating)
from bigbasket));

-- 49 To display the number of bookings each guest made and include the
average total amount per booking for each guest, which query is
correct?
select * from guests;
select * from bookings;
select * from rooms;

```

```

select g.guestid, count(b.bookingid) as numberofbooking,
avg(b.totalamount) as averagetotal
from guests g join bookings b
using(guestid)
group by g.guestid;

describe bookings;

alter table bookings
add column dateCheckIn date;

update bookings
set dateCheckIn = str_to_date(checkindate, '%Y-%m-%d');

-- 50 You want to find pairs of guests who stayed in the same room on
different dates.

select b.guestid, c.guestid
from bookings b join bookings c
on b.roomnumber = c.roomnumber
where b.datecheckin != c.datecheckin
and b.guestid != c.guestid;

-- 30 You want to label rooms based on the number of beds: 'Single' for
rooms with 1 bed, 'Double' for rooms with 2 beds, and 'Suite' for rooms
with more than 2 beds.

use internshala;
select * from guests;
select * from bookings;
select * from rooms;

select roomnumber, count(bedtype) as c,
case
when count(bedtype) = 1 then 'Single'
when count(bedtype) = 2 then 'double'
else 'suite'
end as 'category'
from rooms
group by roomnumber;

select * from bigbasket;

/*You want to update the product type based on sale_price:

If sale_price > 500 → 'Premium'

If sale_price between 200 and 500 → 'Mid-Range'

If sale_price < 200 → 'Budget'

Write an SQL query to update the table. (Bigbasket dataset) */

update bigbasket
set type = case when sale_price > 500 then 'Premieum'
when sale_price between 200 and 500 then 'Mid-range'
when sale_price < 200 then 'Budget'
else 'low' end;

```

-- 31 To find room types that have been booked more than 5 times, which query is correct?

```
select * from guests;
select * from bookings;
select * from rooms;
```

```
select r.roomtype, count(b.bookingid) as b
from rooms r join bookings b
on r.roomnumber = b.roomnumber
group by r.roomtype
having b > 5;
```

-- 32 You want to retrieve the name of guests and the type of room they booked

```
select distinct(concat(g.firstname, ' ', g.lastname)) as full_name,
r.roomtype
from guests g join bookings b
on g.guestid = b.guestid
join rooms r
on b.roomnumber = r.roomnumber;
```

-- 33 You want to create a CTE that ranks guests based on the total amount they have spent on bookings.

```
with guests_rank as
(select distinct(concat(g.firstname, ' ', g.lastname)) as full_name,
sum(b.totalamount) as sumtotal
from guests g join bookings b
using(guestid)
group by full_name)
select full_name, sumtotal,
rank() over(order by sumtotal desc) as rn
from guests_rank;
```

-- 34 You want to compare each guest's total booking amount to the next guest's total booking amount.

```
with compare as
(select guestid, sum(totalamount) as sumtotal
from bookings
group by guestid)
select guestid, sumtotal, lead(sumtotal, 1) over(order by guestid) as
'next guest'
from compare;
```

-- 35 You want to create a temporary table that stores the details of guests who have spent more than \$1000 on bookings

```
create temporary table high_spenders as
select g.guestid, concat(g.firstname, ' ', g.lastname) as full_name,
sum(b.totalamount) as sumtotal
from guests g join bookings b
using(guestid)
group by g.guestid, full_name
```

```
having sumtotal > 1000;
```

```
SELECT * FROM high_spenders;
```

```
-- You want to update the product category to 'High Demand' if the  
product has been ordered more than 20 times. (Bigbasket dataset)  
select * from bigbasket;
```

```
update bigbasket b  
set b.category = 'High Demand' where exists  
(select product, c from  
(select product, count(product) as c  
from bigbasket  
group by product  
having c > 20) as raj  
where raj.product = b.product);
```

```
select product, category  
from bigbasket  
where category = 'High Demand';
```

```
-- 36 You want to find the booking amount for each guest and compare it  
to both the previous and next bookings.
```

```
use internshala;  
select * from guests;  
select * from bookings;  
select * from rooms;
```

```
select guestid, totalamount, lag(totalamount,1) over(order by  
bookingid) as previous_booking, lead(totalamount, 1) over(order by  
bookingid) as next_booking  
from bookings;
```

```
-- 38 You want to find all guests who have not made any bookings in the  
past year.
```

```
select c.guestid, c.datecheckin from bookings c where not exists  
(select b.guestid  
from bookings b  
where b.datecheckin > subdate(curdate(), interval 1 year) and b.guestid  
= c.guestid);
```

```
-- 39 You want to classify guests as 'Frequent' if they have made more  
than 3 bookings, and 'Infrequent' otherwise.
```

```
select guestid, count(bookingid) as c,  
case  
when count(bookingid) > 3 then 'frequent'  
else 'Infrequent'  
end as Category  
from bookings  
group by guestid;
```

```
use internshala;
```

```

-- 20 To classify bookings based on their total amount as 'Low',
'Medium', or 'High'.

select * from guests;
select * from bookings;
select * from rooms;

select bookingid, sum(totalamount) as sumtotal,
case
when sum(totalamount) > 500 then 'High'
when sum(totalamount) > 300 then 'Medium'
else 'low'
end as Category
from bookings
group by bookingid;

-- 21 To find the average room rate for each room type, which query
should you use?

select roomtype, avg(roomrate) as 'average rate'
from rooms
group by roomtype;

-- 22 To retrieve room types that have more than 3 bookings, which
query is correct?

select r.roomtype, count(b.bookingid) as numbers
from rooms r join bookings b
using(roomnumber)
group by r.roomtype
having count(b.bookingid) > 3;

/* 23 You want to create a CTE that calculates the total amount spent
by each guest
and then retrieves only those guests who spent more or equal to $500.
Which query is correct? */

with total as
(select guestid, sum(totalamount) as sumtotal
from bookings
group by guestid)
select guestid, sumtotal
from total
where sumtotal >= 500;

-- 24 You want to compare guests who have booked the same room on
different dates. Which query is correct?

select * from bookings;

select b.guestid, c.guestid, b.datecheckin, c.datecheckin
from bookings b join bookings c
on b.roomnumber = c.roomnumber
where b.datecheckin != c.datecheckin
and b.guestid != c.guestid;

```

-- 25 To find room types where the total amount spent on bookings exceeds 1000, which query is correct?

```
select r.roomtype, sum(b.totalamount) as tot
from rooms r join bookings b
using(roomnumber)
group by r.roomtype
having sum(b.totalamount) > 1000;
```

-- 26 You want to retrieve all bookings, including those without corresponding guest information. Which query should be used?

```
select *
from bookings b left join guests g
on b.guestid = g.guestid;
```

-- 27 You want to find all rooms that have been booked by guests who are older than 40. Which query is correct?

```
select * from guests;
select * from bookings;
select * from rooms;
```

```
select r.roomnumber, g.age
from rooms r join bookings b
on r.roomnumber = b.roomnumber
join guests g
on b.guestid = g.guestid
where g.age > 40;
```

-- 28 You want to create a CTE that calculates the total bookings for each guest and then retrieves guests who made more than 3 bookings

```
with total_bookings as
(select guestid, count(bookingid) as countbookings
from bookings
group by guestid)
select * from total_bookings
where countbookings > 3;
```

-- 29 To calculate the difference in days between a guest's current booking check-in date and the next booking's check-in date

```
with book as
(select guestid, datecheckin as present, lead(datecheckin, 1)
over(order by datecheckin) as nextbooking
from bookings)
select guestid, present, nextbooking, datediff(nextbooking, present) as
diffrence
from book;
```

use internshala;

/* 30 You want to update the rooms table by converting text-based bed types into numeric values:

'Twin' should become 2

'Queen' should become 1

'King' should become 3 */

```
select * from rooms;
```

```
update rooms
```

```
set bedtype = case when bedtype = 'Twin' then 2
```

```
when bedtype = 'Queen' then 1
```

```
when bedtype = 'King' then 3
```

```
end;
```