# SQL - First 50 Questions Solution

1. select * from CITY where COUNTRYCODE = ‘USA’;
2. select NAME from CITY where COUNTRYCODE = ‘USA’ and POPULATION > 120000;
3. select * from CITY;
4. select * from CITY where ID = 1661;
5. select * from CITY where COUNTRYCODE = ‘JPN’;
6. select DISTRICT from CITY where COUNTRYCODE = ‘JPN’;

7. select CITY, STATE from STATION;
8. select CITY from STATION where (ID%2 = 0);
9. select count(CITY) - count distinct(CITY) from STATION;

10. with temp_table1 as (select CITY, char_length(CITY) as length_city  from STATION
    order by length_city desc, CITY asc limit 1
                **UNION**
            select CITY, char_length(CITY) as length_city  from STATION
    order by length_city, CITY asc limit 1 )

    select * from temp_table1;

    **#CHAR_LENGTH() function measures the string length in 'characters' whereas LENGTH() function measures the string length in 'bytes'**

    **OR**

    select CITY, char_length(CITY) as length_city  from STATION where char_length(CITY) = (select max(char_length(CITY) from STATION) order by CITY limit 1
    union
    select CITY, char_length(CITY) as length_city  from STATION where char_length(CITY) = (select min(char_length(CITY) from STATION);

11. select distinct CITY from STATION
    where lower(substr(city, 1,1)) IN ('a','e','i','o','u');

    OR

    select distinct CITY from STATION

where left(city, 1) IN ('a','e','i','o','u');

12. select distinct CITY from STATION
    where lower(substr(CITY, -1,1)) IN  (‘a’,’e’,’i’,’o’,’u’);
13. select distinct CITY from STATION
    where lower(substr(city, 1,1)) NOT IN ('a','e','i','o','u');
14. select distinct CITY from STATION
    where lower(substr(CITY, -1,1)) NOT IN  (‘a’,’e’,’i’,’o’,’u’);

15. select distinct CITY from STATION
    where left(city, 1) NOT IN ('a','e','i','o','u')
    OR
    where right(city, 1) NOT IN (‘a’,’e’,’i’,’o’,’u’);

16. select distinct CITY from STATION
    where left(city,1) NOT IN ('a','e','i','o','u')
    AND
    where right(city, 1) NOT IN (‘a’,’e’,’i’,’o’,’u’);

17. select product_id, product_name from Product P
    Inner Join Sales S on P.product_id = S.product_id
    where DATE BETWEEN 2019-01-01 and 2019-03-31 ;

    OR
    Date <=___ and Date>=___;

18. select author_id from Views where author_id = viewer_id order by author_id;

    immediate orders —> count —> % = subtract from actual/actual

19. select round
    (
    (select count(*) from Delivery
    where order_date = customer_pref_delivery_date)/count(*)*100 ,2
    ) as immediate_percentage from Delivery;

20. **NOT SURE whether correct or not:**
    with cte as
    (
    select count(*) as ignored_count from Ads where action = ‘Ignored’
                    UNION
    select count(*) as clicked_count from Ads where action = ‘Clicked’

```
                    UNION
select count(*) as viewed_count from Ads where action = ‘Viewed’
                    UNION
select ad_id,
case when (clicked_count+ viewed_count =0) then 0
         else (clicked_count/(clicked_count+ viewed_count))*100
end as ctr
from Ads
)

select ad_id, ctr  from CTE
group by ad_id
sort by ctr desc, ad_id asc;
```

**Correct answer:**

```
 select distinct ad_id, ifnull(round(
       sum(action = 'Clicked') / (sum(action = 'Clicked') + sum(action = 'Viewed'))
* 100, 2
      ) , 0) as ctr
    from Ads
    group by ad_id
    order by ctr desc, ad_id;
```

21. select employee_id, (select count(*) as team_size from Employee
    group by team_id ) as team_size from Employee;

    OR

```
select employee_id, count(team_id) over(partition by (team_id)) as team_size
from Employee;
```

22.

```
     WITH cte AS (
   SELECT country_id, SUM(weather_state) OVER (PARTITION BY country_id) AS
weather_st
   FROM Weather
   WHERE     YEAR(day) = 2019 AND MONTH(day) = 11
)
SELECT distinct country_name,
   CASE
      WHEN cte.weather_st <= 15 THEN 'Cold'
      WHEN cte.weather_st >= 25 THEN 'Hot'
```

```
        ELSE 'Warm'
    END AS weather_type
FROM Countries c
INNER JOIN cte ON c.country_id=cte.country_id;
```

23.

```
with cte as
(
select p.product_id, (p.price*u.units) as total_price
from Prices p inner join UnitsSold u on p.product_id = u.product_id
where u.purchase_date BETWEEN p.start_date AND p.end_date
)
select product_id, round(avg(total_price),2) as average_price from cte
group by product_id;
```

OR

select **p.product_id,round(sum(unit*price)/sum(unit),2)** as average_price
from prices p **inner join** unitssold u **on** p.product_id=u.product_id **and**
 u.purchase_date **between** p.start_date **and** p.end_date **group by product_id;**

24. select player_id ,min(event_date) as first_login_date from activity group by player_id;

25. select player_id ,device_id from Activity a
    where event_date =
    (select min(event_date) from Activity group by player_id HAVING player_id = a.player_id);

26. select p.product_name, sum(o.unit)
    from Products p Inner Join Orders o on p.product_id = o.product_id
    WHERE year(o.order_date) = 2020 and month(o.order_date) = 2
    group by p.product_name
    HAVING sum(o.unit) >= 100;

Products table:

| product_id | product_name | product_catego ry |
|---|---|---|
| 1 | Leetcode Solutions | Book |
| 2 | Jewels of Stringology | Book |
| 3 | HP | Laptop |

| | | |
|---|---|---|
| 4 | Lenovo | Laptop |
| 5 | Leetcode Kit | T-shirt |

Orders table:

| product_id | order_date | unit |
|---|---|---|
| 1 | 2020-02-05 | 60 |
| 1 | 2020-02-10 | 70 |
| 2 | 2020-01-18 | 30 |
| 2 | 2020-02-11 | 80 |
| 3 | 2020-02-17 | 2 |
| 3 | 2020-02-24 | 3 |
| 4 | 2020-03-01 | 20 |
| 4 | 2020-03-04 | 30 |
| 4 | 2020-03-04 | 60 |
| 5 | 2020-02-25 | 50 |
| 5 | 2020-02-27 | 50 |
| 5 | 2020-03-01 | 50 |

Output:

| product_name | unit |
|---|---|
| Leetcode Solutions | 130 |
| Leetcode Kit | 100 |

27. Input: Users table:

| user_id | name | mail |
|---|---|---|
| 1 | Winston | winston@leetc ode.com |
| 2 | Jonathan | jonathanisgreat |
| 3 | Annabelle | bella-@leetcod e.com |
| 4 | Sally | sally.come@lee tcode.com |
| 5 | Marwan | quarz#2020@le etcode.com |
| 6 | David | david69@gmail .com |
| 7 | Shapiro | .shapo@leetco de.com |

Output:

| user_id | name | mail |
|---|---|---|
| 1 | Winston | winston@leetc ode.com |

| | | |
|---|---|---|
| 3 | Annabelle | bella-@leetcod e.com |
| 4 | Sally | sally.come@leetcode.com |

select user_id, name, mail from Users where mail like ‘^[a-zA-Z0-9_\-\.]+ @leetcode[\.]com’ ;

28. **Write an SQL query to report the customer_id and customer_name of customers who have spent at least $100 in each month of June and July 2020.**

Customers table:

| customer_id | name | country |
|---|---|---|
| 1 | Winston | USA |
| 2 | Jonathan | Peru |
| 3 | Moustafa | Egypt |

Product table:

| product_id | description | price |
|---|---|---|
| 10 | LC Phone | 300 |
| 20 | LC T-Shirt | 10 |
| 30 | LC Book | 45 |
| 40 | LC Keychain | 2 |

Orders table:

| order_id | customer_id | product_id | order_date | quantity |
|---|---|---|---|---|
| 1 | 1 | 10 | 2020-06-10 | 1 |
| 2 | 1 | 20 | 2020-07-01 | 1 |
| 3 | 1 | 30 | 2020-07-08 | 2 |
| 4 | 2 | 10 | 2020-06-15 | 2 |
| 5 | 2 | 40 | 2020-07-01 | 10 |
| 6 | 3 | 20 | 2020-06-24 | 2 |
| 7 | 3 | 30 | 2020-06-25 | 2 |
| 9 | 3 | 30 | 2020-05-08 | 3 |

Output:

| customer_id | name |
|---|---|
| 1 | Winston |

select o.customer_id, c.name from Orders o **INNER JOIN** Customers c **ON** o.customer_id = c.customer_id

**INNER JOIN** Product p **ON**
o.product_id = p.product_id
where (p.price*o.quantity) >=100 **AND** year(o.order_date)=2020 **AND**
month(o.order_date) **IN**(6,7);

29. **Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.**

Input: TVProgram table:

| program_date | content_id | channel |
|---|---|---|
| 2020-06-10 08:00 | 1 | LC-Channel |
| 2020-05-11 12:00 | 2 | LC-Channel |
| 2020-05-12 12:00 | 3 | LC-Channel |
| 2020-05-13 14:00 | 4 | Disney Ch |
| 2020-06-18 14:00 | 4 | Disney Ch |
| 2020-07-15 16:00 | 5 | Disney Ch |

Content table:

| content_id | title | Kids_content | content_type |
|---|---|---|---|
| 1 | Leetcode Movie | N | Movies |
| 2 | Alg. for Kids | Y | Series |
| 3 | Database Sols | N | Series |
| 4 | Aladdin | Y | Movies |
| 5 | Cinderella | Y | Movies |

Output:

| title |
|---|
| Aladdin |

select distinct title from Content c INNER JOIN TVProgram t on t.content_id =
c.content_id
WHERE c.Kids_content = ‘Y’ AND year(t.program_date) = 2020 AND
month(t.program_date) = 6 AND c.content_type=‘Movies’;

30.

Table: NPV (id, year) is the primary key of this table.
The table has information about the id and the year of each inventory and the corresponding net present value.

Table: Queries (id, year) is the primary key of this table.
The table has information about the id and the year of each inventory query.
Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order.
The query result format is in the following example.

NPV table:

| id | year | npv |
|---|---|---|
| 1 | 2018 | 100 |
| 7 | 2020 | 30 |
| 13 | 2019 | 40 |
| 1 | 2019 | 113 |
| 2 | 2008 | 121 |
| 3 | 2009 | 12 |
| 11 | 2020 | 99 |
| 7 | 2019 | 0 |

Queries table:

| id | year |
|---|---|
| 1 | 2019 |
| 2 | 2008 |
| 3 | 2009 |
| 7 | 2018 |
| 7 | 2019 |
| 7 | 2020 |
| 13 | 2019 |

Output:

| id | year | npv |
|---|---|---|
| 1 | 2019 | 113 |
| 2 | 2008 | 121 |
| 3 | 2009 | 12 |
| 7 | 2018 | 0 |
| 7 | 2019 | 0 |
| 7 | 2020 | 30 |
| 13 | 2019 | 40 |

select q.*, n.npv as npv from Queries q INNER JOIN NPV n on q. id = n. id and q.year = n.year;

31. select q. id, q.year, ifnull(n.npv, 0) from npv n RIGHT JOIN Queries q on n. id= q. id and n.year = q.year;

32. select eu.unique_id , e.name from Employees e **LEFT JOIN** EmployeeUNI eu
    **ON** e. id = eu. id
    where e. id = eu. id;

33. select t1.name, ifnull(sum(t2.distance),0) from Users t1 **LEFT JOIN** Rides t2 **on**
    t1. id = t2.user_id
    group by t1. id;

34. select p.product_id, sum(o.units) as units_sold, o.order_date from Products p
    **INNER JOIN** Orders o **ON** p.product_id =o.product_id
    **where** month(o.order_date) = 2 **AND** year(o.order_date)=2022
    **group by** p.product_id
    **HAVING** units_sold >= 100;

35.
    - Find the name of the user who has rated the greatest number of movies. In case of a tie,
    return the lexicographically smaller user name.

    select u. name from Users u **INNER JOIN** MovieRating m **ON** u. user_id =
    m. user_id
    **group by** u.name
    **order by** count(m.rating) desc, u.name asc
    limit 1;

    - Find the movie name with the highest average rating in February
2020. In case of a tie, return
the lexicographically smaller movie name.

select u. name from Users u **INNER JOIN** MovieRating m **ON** u. user_id = m.
user_id
**group by** u.name
**order by** avg(m.rating) desc, u.name asc
limit 1;

36. select U.name, ifnull(sum(R.travelled_distance),0)
    from Users U **INNER JOIN** Riders R **ON** U. id = R. user_id
    groupby U. id
    order by 2 desc, 1 asc

37. select ifnull(e2.unique_id, null), e1.name
    from Employees e1 **INNER JOIN** EmployeeUNI e2

```
ON e1. id = e2. id
```

38.
```
select S. id, S. name
from Students INNER JOIN Departments D
ON S. department_id = D. id
where S.department_id != D. id;
```

39.
```
select from_id as person1, to_id as person2,
count(*) as call_count, sum(duration) from Calls
where 1<2
groupby 1,2
```

40. units*price/ (# units) , product_id

```
select p.product_id,
round(sum(u.units*p.price)/ sum(u.units) ,2) AS average_price
from Prices p INNER JOIN UnitsSold u ON p.product_id = u.product_id
group by p.product_id;
```

41. w name, p volume = W*L*H

```
select w.name as warehouse_name, sum(p.Width * p.Length * p. Height) AS
volume
from Warehouse w INNER JOIN products p ON w.product_id =
p.product_id
group by w.name;
```

42.
```
select sale_date,
sum(case when fruit ='apples' then sold_num else 0 END)-
sum(case when fruit ='oranges' then sold_num else 0 END)
as diff
group by sale_date;
```

43.
```
with cte as
(
select player_id, event_date,
lead (event_date) over(partition by player_id order by event_date) as
next_date
from Activity
)

select round(
count(distinct case when datediff(next_date, event_date) = 1 then 1 else 0
end)/ count (distinct player_id)
```

**,2)**
AS fraction from cte;

44. SELECT Name FROM Employee where id IN
(
SELECT ManagerId FROM Employee
GROUP BY ManagerId
HAVING COUNT(ManagerId) >= 5
);

45. select d.dept_name, count(s.student_name) AS student_number
from Student AS s INNER JOIN Department AS d ON
s.student_id = d.dept_id
group by 1
order by 2 desc, 1 asc;

46. select distinct(c.customer_id)
**from Customer c INNER JOIN Product p ON c.product_key =**
**p.product_key**
where count (distinct c.product_key) =

(select count (distinct product_key) from Product)

group by c.customer_id;

47. with CTE AS
(
select p.project_id, e.employee_id, max(experience_years)
**from Project p INNER JOIN Employee e**
**ON e.employee_id = p.employee_id**
group by 2
order by 3
)

select p.project_id, e.employee_id from CTE;

OR

***select * from***
***(***
*select p.project_id, e.employee_id, e.experience_years, rank() over*
*(partitionby project_id order by experience_years desc )as*
*rank_experience*
*from Project p join employee eon p.employee_id = e.employee_id*

```
)
where rank_experience=1;
```

48.
```
select b.name from
(
select name, available_from from Books where available_from <
2019-05-23
) b
INNER JOIN
(
select book_id, dispatch_date from Orders where dispatch_date >
2018-06-23 AND
dispatch_date < 2019-06-23
group by book_id
HAVING sum(quantity) < 10
) o
ON b.book_id = o.book_id;
```

49.
```
select student_id, min(course_id) as course_id, grade from Enrollments
where (student_id, grade) IN
(
select student_id, max(grade) from Enrollments
group by student_id
)
order by student_id;
```