

# Easy

solve following here: [https://www.naukri.com/code360/problem-lists/top-100-sql-problems?difficulty\[\]=Easy](https://www.naukri.com/code360/problem-lists/top-100-sql-problems?difficulty[]=Easy)

## 1. BIG COUNTRIES

```
SELECT name, population, area
FROM World
WHERE area > 3000000
OR population > 25000000;
```

## 2. COMBINE TWO TABLES

```
SELECT FirstName, LastName, City, State
FROM Person
LEFT JOIN Address
USING (PersonId);
```

## 3. STUDENTS DB

```
INSERT INTO students VALUES (3, 'Kim', 'F'), (4, 'Molina',
'F'), (5, 'Dev', 'M');
SELECT *
FROM students;
```

## 4. SPOTIFY SESSIONS

```
SELECT session_id
FROM Playback
LEFT JOIN Ads
ON Playback.customer_id = Ads.customer_id
```

```
AND timestamp BETWEEN start_time AND end_time  
WHERE ad_id IS NULL;
```

## 5. ARTICLE

```
SELECT viewer_id AS id  
FROM Views  
WHERE author_id != viewer_id  
GROUP BY viewer_id  
HAVING COUNT(article_id) > 1  
ORDER BY id;
```

## 6. DUPLICATE EMAILS

```
SELECT Email  
FROM Person  
GROUP BY Email  
HAVING COUNT(*) > 1;
```

## 7. MARVEL CITIES

```
SELECT *  
FROM City  
WHERE population > 100000  
AND CountryCode = 'Marv';
```

## 8. WAREHOUSE MANAGER

```
select name as warehouse_name,sum((Width*Length*Height)*uni  
ts) as volume  
from warehouse  
JOIN  
Products
```

```
ON warehouse.product_id = Products.product_id  
group by name;
```

## 9. SALES EXECUTIVE

```
SELECT name  
FROM SalesPerson AS s  
WHERE sales_id NOT IN (  
    SELECT sales_id  
    FROM Orders  
    LEFT JOIN Company  
    USING (com_id)  
    WHERE Company.name = 'RED'  
);
```

## 10. NPV QUERIES

```
SELECT  
    q.id,  
    q.year,  
    COALESCE(n.npv, 0) AS npv  
FROM  
    Queries AS q  
LEFT JOIN  
    NPV AS n  
ON  
    q.id = n.id AND q.year = n.year;
```

## 11. SWAP SALARY

```
UPDATE Salary  
SET sex = CASE WHEN sex = 'f' THEN 'm' ELSE 'f' END;
```

## 12. DIRECTOR'S ACTOR

```
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id, director_id
HAVING COUNT(*) >= 3;
```

### 13. LARGEST ORDER

```
SELECT customer_number
FROM Orders
GROUP BY 1
ORDER BY count(*) DESC;
```

### 14. IMDB MetaCritic Rating

```
SELECT Title, Rating
FROM IMDB
INNER JOIN earning
USING (Movie_id)
WHERE MetaCritic > 60
AND Domestic > 1000000000
AND Title LIKE '%2012%';
```

### 15. RANK SCORES

```
SELECT score,
       DENSE_RANK() OVER(ORDER BY Score DESC) AS Rank
FROM Scores;
```

### 16. MAX WEIGHT

```
SELECT genre , MAX((Rating + MetaCritic/10.0)/2) as weight
ed_rating
FROM imdb as i JOIN genre as g
USING(Movie_id)
```

```
WHERE i.title LIKE '%2014%'
AND genre IS NOT NULL
AND rating IS NOT NULL
AND MetaCritic IS NOT NULL
GROUP BY genre
ORDER BY genre ASC ;
```

## 17. TRIANGLE JUDGEMENT

```
SELECT x,
       y,
       z,
       CASE WHEN (x + y > z and y + z > x and z + x > y) TH
EN 'Yes' ELSE 'No' END AS triangle
FROM triangle;
```

## 18. The Most Frequently Ordered Products for Each Customer

```
WITH ProductCounts AS (
  -- Step 1 & 2: Count orders per product per customer and rank them
  SELECT
    customer_id,
    product_id,
    RANK() OVER (
      PARTITION BY customer_id
      ORDER BY COUNT(product_id) DESC
    ) as rnk
  FROM Orders
  GROUP BY customer_id, product_id
)
-- Step 3: Filter for the top rank and join for product details
SELECT
  pc.customer_id,
  pc.product_id,
```

```
    p.product_name
FROM ProductCounts pc
JOIN Products p ON pc.product_id = p.product_id
WHERE pc.rnk = 1
ORDER BY pc.customer_id;
```

## 19. Orders With Maximum Quantity Above Average

```
SELECT order_id
FROM OrdersDetails
GROUP BY order_id
HAVING MAX(quantity) > (
    SELECT SUM(quantity) / COUNT(DISTINCT product_id)
    FROM OrdersDetails
)
```

## 20. Product's Worth Over Invoices

```
SELECT name, SUM(rest) AS rest, SUM(paid) AS paid, SUM(canceled) AS canceled, SUM(refunded) AS refunded
FROM product
LEFT JOIN Invoice
USING (product_id)
GROUP BY name
ORDER BY name;
```

