Must know question types and concepts to crack SQL Interviews –

1) ‘Cumulative sum’ type questions -

e.g. Given, employee id, month and salary columns in a table, you should be able to find the cumulative sum of an employee’s salary over the last few months. Make sure you are well versed with ‘common table expressions’ and how to apply them to make your code more readable and also in some cases avoiding use of repeated subqueries. Having an in-depth knowledge of different windows functions is a must. In this example, one of the solutions would be –

```
with e as
(select Id, Month, max(Month) over (partition by Id) as max_mon, Salary
   from Employee)
select e.Id, e.Month, sum(e.Salary) over (partition by Id order by Month rows 2 preceding) as Salary
from e
where e.Month < max_mon
order by e.Id asc, e.Month desc
```

2) Finding *top ‘n’ salaries* in a department –
Here, its important to understand the use of the *dense_rank* function and how it differentiates 'from the *rank* function. We can use the ‘*TOP’* function as well, but it will not account for ties in salaries of employees. In this case, given a department and employee salary table, your SQL query would look something like –

```
select t.dept_name as department, t.name as employee, t.salary
from
(select e.*,d.name as dept_name
      ,dense_rank() over (partition by d.name order by salary desc) as rank
from employee e
left join department d on e.departmentid=d.id
)t
where t.rank<=3
and t.dept_name is not null
```

3) Find *‘median’* employee salary –
Given employee table containing employee id, company and salary, you should be able to design a query to pull the median salary of an employee in each department. Again, advanced knowledge of windows functions like *‘row_number*’ would help to solve this quickly and efficiently -

```
select t.id,t.company,t.salary
from
(select e.* ,row_number() over (partition by company order by salary asc) as rn
      ,count(id) over (partition by company) as total
from employee e)t
where rn between (total*1.0/2) and (total*1.0/2)+1
```

4) Learning to join table with itself – Important to know how to do this, an example would be finding the employees who earn more than their manager (given a single employee table with the corresponding manager name and salary)

5) Find continuous available seats in a cinema hall – Given a seat_id and whether its free or not, we have to find consecutive available seats. A sample solution would look like -

```
select a.seat_id
from cinema a left join cinema b on a.seat_id=b.seat_id+1
where a.free=1 and b.free=1

UNION

select b.seat_id
from cinema a left join cinema b on a.seat_id=b.seat_id+1
where a.free=1 and b.free=1
```

There are a bunch of other question types as well, but once you have mastered the above question types, you will find that you have a broad set of tools to tackle a lot of other questions that may be thrown at you during the interviews. Few other important functions and points to remember would be –

1) CASE WHEN statement
2) COALESCE (or CASE WHEN xyz is NULL)
3) NULLIF
4) ISNULL
5) DECODE
6) TOP, LIMIT
7) Difference between UNION ALL and UNION
8) Difference between using Count(CASE WHEN xyz) vs CASE WHEN COUNT(xyz). This difference is very important.
9) Remember, order of query execution –
   FROM->WHERE->GROUP BY->HAVING->SELECT->ORDER BY
10) Learn how to use common table expressions (CTE's) as much as possible
11) Avoid using windows functions on large data pulls, it would be a severe drag on query time. Use sub-queries/CTE's instead.
12) Learn date functions e.g. getdate(), datediff() and use of the formatting functions like FORMAT(), ROUND()
13) Perform mathematical operations on output of 2 SQL queries (division, multiplication, addition, subtraction)
14) Using count(distinct xyz) vs count(xyz)
15) Learning how to use derived tables is very important. You can of course choose to use CTE's instead depending on which method is more suited for you.
16) Learning using of 'LIKE' , '!=' and '=' operators, along with basic regex knowledge - 'A-Za-z0-9'
17) Finally, understand the intricacies in using inner join, left join and full outer join and when we use one join over the other. This is probably the most important concept you need to understand while formulating advanced SQL queries.