Section – A: Theory Questions

1. **What are the different types of window functions in SQL?**
   Window functions in SQL are used to perform calculations across a set of rows related to the current row in a query result. There are 11 different types of window functions in SQL.
   Rank, Dense_rank, Row_number, first_value, last_value, nth_value, ntile, percent_rank, cume_dist, lead, lag

2. **What is the purpose of the "RANK", "DENSE_RANK", and "ROW_NUMBER" functions in SQL?**
   RANK: This function returns the rank of a row within a result set, with the same rank assigned to rows with the same values. The rank of the next row is increased by the number of tied rows.
   DENSE_RANK: This function works similarly to RANK, but it doesn't skip any rank values even if there are tied rows.
   ROW_NUMBER: This function returns the unique number assigned to each row within a result set. It doesn't assign the same number to rows with the same values.

   Syntax:
   SELECT employee_id, salary, RANK() OVER (ORDER BY salary DESC) as salary_rank FROM employees;
   SELECT employee_id, salary, DENSE_RANK() OVER (ORDER BY salary DESC) as salary_rank FROM employees;
   SELECT employee_id, salary, ROW_NUMBER() OVER (ORDER BY salary DESC) as salary_rank FROM employees;

3. **What are the "FIRST_VALUE", "LAST_VALUE", and "NTH_VALUE" functions in SQL and how do they work?**
   FIRST_VALUE: This function returns the value of the first row in the specified partition.
   LAST_VALUE: This function returns the value of the last row in the specified partition.
   NTH_VALUE: This function returns the value of the nth row in the specified partition.

   Syntax:
   SELECT employee_id, salary, FIRST_VALUE(salary) OVER (PARTITION BY department_id ORDER BY hire_date) as first_salary FROM employees;
   SELECT employee_id, salary, LAST_VALUE(salary) OVER (PARTITION BY department_id ORDER BY hire_date) as last_salary FROM employees;
   SELECT employee_id, salary, NTH_VALUE(salary, 2) OVER (PARTITION BY department_id ORDER BY hire_date) as second_highest_salary FROM employees;

4. **What is the "NTILE" function in SQL, and how does it work?**
   The "NTILE" function in SQL is used to divide the result set into a specified number of groups, or tiles, and then assign a unique number to each tile. For example, if you have a result set with 100 rows and use NTILE(10), then the result set would be divided into 10 groups of 10 rows each, and each group would be assigned a unique number from 1 to 10.

   Syntax:
   SELECT column1, column2, NTILE(4) OVER (ORDER BY column2) as GroupNumber FROM table_name

5. **What is the "PERCENT_RANK" function in SQL, and how does it work?**
   The "PERCENT_RANK" function in SQL calculates the relative rank of each row within a result set.
   For example, if you have a result set with 100 rows and use PERCENT_RANK(), then the result set would be ranked from 1 to 100, and each row would be assigned a rank value that represents its relative rank within the result set.
   Formula:
   rank-1 / nrow-1

   Syntax:
   SELECT column1, column2, PERCENT_RANK() OVER (ORDER BY column2) as PercentRank FROM table_name;

6. **What is the "CUME_DIST" function in SQL, and how does it work?**
   The "CUME_DIST" function in SQL calculates the cumulative distribution of the values in a result set. For example, if you have a result set with 100 rows and use CUME_DIST(), then the result set would be ranked from 1 to 100, and each row would be assigned a cumulative distribution value that represents the fraction of rows that are equal to or less than that row.

   Syntax:
   SELECT column1, column2, CUME_DIST() OVER (ORDER BY column2) as CumulativeDistribution FROM table_name;

7. **What are the "LEAD" and "LAG" functions in SQL, and how do they work?**
   The "LEAD" and "LAG" functions in SQL are used to access the values in a result set that are located either ahead or behind the current row.
   For example, if you have a result set with 100 rows and use LEAD(column_name, n), then the result set would return the value of the specified column for the nth row ahead of the current row.
   Similarly, if you use LAG(column_name, n), then the result set would return the value of the specified column for the nth row behind the current row.

   Syntax:
   select column1, lag(column1) over() as "previous", lead(column1) over() as "next" from table_name;
   Note: If the value doesn't exist, it returns null

8. **What is the purpose of the "OVER()" clause in a window function in SQL, and when should we use "Range" and "Row"?**
   The "OVER()" clause in a window function in SQL is used to specify the window or set of rows that the function operates on.
   Whereas, partition by column order by column are the two clauses

   Syntax:
   select column1, sum(column) over(partition by column order by column) as "calculated_field" from table_name;
   select column1, last_value(column) over(partition by column order by column range between unbounded preceding and unbounded following) as "calculated_field" from table_name;

   Extra notes:
   For the Frame Clause Such as Rows between, Range between are mostly used for last_value, nth_value window functions
   range between unbounded preceding and current row: by default, changes in current row -> unbounded following
   range between unbounded preceding and unbounded following: select the whole range of partition

   # range between 2 preceding and 2 following

   - selection starts from 2 indexes from the first index 0 (2 preceding)
   - after 2 indexes 2 rows ends (2 followings)

   # difference between range and rows

range between unbounded preceding and current row: if there are duplicates, returns the last_value

rows between unbounded preceding and current row: if there are duplicated, returns the first_value of the last_value function

```
# shortend code (after: from, where | before: order by)
select *, last_value(col) over w as 'least col'
from table_name where col = 'val1' window w as (range between unbounded preceding and unbounded following)
```