**SQL Interview Prep**

1: Write a SQL query to find the Nth highest salary from the Employees table. (e.g., find the 3rd highest salary).

Select name, salary from

(select name, salary, dense\_rank() over (order by salary desc) as rnk from employes)

Where rnk=3;

2. Write a SQL query to find all email addresses that appear more than once in the Employees table.

Select email, count(\*) from employes

Group by email having count (\*)>1

3. Write a SQL query to retrieve the first\_name and last\_name of all employees who work in the 'Sales' department. Assume you need to join Employees and Departments tables.

Select e.first\_name, e.last\_name,d.department\_name from

Employees e join Departments d on e. department\_id = d. department\_id

Where d.department\_name = ‘Sales’

4. Write a SQL query to find the employee\_id, first\_name, and salary of the employee with the highest salary in *each* department.

Select employee\_id, first\_name, salary from

(select employee\_id, first\_name, salary, dense\_rank over(partition by department\_id order by salary desc) as rnk from employes )

Where rnk=1

5.Write a SQL query to find the second highest salary from the Employees table *without* using window functions like DENSE\_RANK or keywords like LIMIT or TOP.

SELECT first\_name, last\_name, salary FROM Employees WHERE salary = (SELECT MAX(salary) FROM Employees WHERE salary < (SELECT MAX(salary) FROM Employees));

6.Write a SQL query to find all employees who are *not* assigned to any department (i.e., their department\_id is NULL or does not exist in the Departments table).

SELECT e.first\_name FROM Employees e LEFT JOIN Departments d ON e.department\_id = d.department\_id WHERE e.department\_id IS NULL OR d.department\_id IS NULL;

7. Write a SQL query to show each employee's first\_name, department\_name, salary, and the cumulative salary of employees within their department, ordered by salary (ascending) within each department.

SELECT e.first\_name, d.department\_name, e.salary, e.c\_salary FROM

( SELECT employee\_id, first\_name, department\_id, salary, SUM(salary) OVER (PARTITION BY department\_id ORDER BY salary ASC) AS c\_salary FROM Employees) e

JOIN Departments d ON e.department\_id = d.department\_id ORDER BY d.department\_name, e.salary ASC;

8. Write a SQL query to display each employee's first\_name along with their manager's first\_name. The Employees table has a manager\_id column that refers back to employee\_id. Handle cases where an employee has no manager.

Select e.first\_name,m.first\_name from Employes e ,Employes m where

e.manager\_id = m.employes\_id

9. You have a table EmailList (id INT, email VARCHAR). Write a SQL query to delete duplicate email entries, keeping only the row with the smallest id for each email.

DELETE FROM EmailList

WHERE id NOT IN

( SELECT MIN(id) FROM EmailList GROUP BY email );

10. Given a Sales table (SaleID, Product, Year, Amount), write a query to display the total sales Amount for each Product for the years 2023, 2024, and 2025, with each year as a separate column.

Select product,

Sum(case when year= 2023 then amount) as sales\_2023,

Sum(case when year= 2024 then amount) as sales\_2024,

Sum(case when year= 2025 then amount) as sales\_2025,

From

Sales

Where year in (2023,2024,2025)

Group by product

11. **Consecutive Logins:** Given a LoginLogs table (LogID, UserID, LoginDate DATE), find all UserIDs who logged in on at least three consecutive days.

WITH ConsecutiveLogins AS (

    SELECT

        UserID,

        LoginDate,

        LAG(LoginDate, 1) OVER (PARTITION BY UserID ORDER BY LoginDate) AS prev\_login,

        LAG(LoginDate, 2) OVER (PARTITION BY UserID ORDER BY LoginDate) AS prev\_login2

    FROM

        LoginLogs

)

SELECT DISTINCT UserID

FROM

    ConsecutiveLogins

WHERE

    DATEDIFF(LoginDate, prev\_login) = 1

    AND DATEDIFF(prev\_login, prev\_login2) = 1;

12. Find all employee\_ids from the Employees table who do *not* exist in a separate BonusRecipients table (employee\_id, bonus\_amount).

SELECT employee\_id FROM Employees

EXCEPT

SELECT employee\_id FROM BonusRecipients;

13. Write a SQL query to update the salary in the Employees table: give a 10% raise to employees in the 'IT' department and a 5% raise to employees in the 'HR' department. Use the Departments table to identify the department names.

 Employees (employee\_id INT PRIMARY KEY, first\_name VARCHAR, last\_name VARCHAR, email VARCHAR, hire\_date DATE, salary DECIMAL, department\_id INT, manager\_id INT)

 Departments (department\_id INT PRIMARY KEY, department\_name VARCHAR)

Update Employees

Set salary =

Case

When d.department\_name = ‘IT’ then e.salary \* 1.10

When d.department = ‘ HR’ then e.salary\*1.05

Else

e.salary

end

FROM Employees e JOIN Departments d ON e.department\_id = d.department\_id WHERE d.department\_name IN ('IT', 'HR');

14. Why are indexes used in databases? Write the SQL statement to create a non-clustered index named idx\_email on the email column of the Employees table.

CREATE NONCLUSTERED INDEX idx\_email

ON Employees (email);

15. Write a SQL query to rank employees within each department based on their salary (highest salary gets rank 1). If two employees have the same salary, they should receive the same rank, and the next rank should be skipped (use RANK()).

SELECT emp\_name, salary, ranked from

(select emp\_name,salary, rank() over(partitation by department order by salary desc) as ranked from employees)

16. Write a PL/SQL procedure named UpdateEmpSalary that accepts an p\_employee\_id (INT) and p\_raise\_percentage (NUMBER) as input. The procedure should update the employee's salary by the given percentage. Include exception handling for cases where the employee\_id does not exist (NO\_DATA\_FOUND).

DECLARE

Exp exception;

Create or replace PROCEDURE UpdateEmpSalary (p\_employee\_id IN int, p\_raise\_percentage IN number) IS BEGIN

Count\_updated\_employee number;

UPDATE employee

SET salary := salary+salary\*(p\_raise\_percentage/100)

Where employee. Employee\_id = p\_employee\_id;

Count\_update\_employee\_number := SQL%ROWCOUNT;

If Count\_update\_employee\_number=0 then

Raise exp

End if;

Exception

When exp then

Dbms\_output.put\_line(“no row effected”);

end;

/

17.Write a PL/SQL function named GetDeptEmployeeCount that takes a p\_department\_id (INT) as input and returns the total number of employees in that department (NUMBER).

Create or replace function GetDeptEmployeeCount(p\_department\_id in int)

Return number;

As

ctn number;

Begin

Select count(\*) into ctn

From employees

Where department\_id = p\_department\_id;

Return ctn;

End;

/

18. Write a PL/SQL anonymous block that uses an explicit cursor to fetch and display (using DBMS\_OUTPUT.PUT\_LINE) the first\_name and salary of all employees earning more than $100,000.

DECLARE

c\_name employees.firstname%type;

c\_salary employee.salary%type;

curser c\_customer is

select first\_name, salary from Employees where

salary >100000;

BEGIN

Open c\_customer

Loop

Fetch c\_customer into c\_name, c\_salary

Exit when c\_customer%notfound;

Dbms\_output.put\_line(‘Name ’|| c\_name|| ‘ Salary ‘ || c\_salary);

End loop;

Close c\_customer;

End;

/

19. Create a PL/SQL trigger named AuditEmployeeUpdates that fires *before* any UPDATE operation on the Employees table. The trigger should automatically set a last\_updated\_date column (assume it exists) to the current system date (SYSDATE).

CREATE OR REPLACE TRIGGER LogEmployeeUpdates

AFTER UPDATE ON Employees

FOR EACH ROW

BEGIN

INSERT INTO EmployeeAudit (employee\_id, update\_timestamp)

VALUES (:OLD.employee\_id, SYSDATE);

END; /

20. Write a PL/SQL anonymous block that attempts to select the salary of an employee with employee\_id = 999 into a variable. Include an exception handler specifically for TOO\_MANY\_ROWS (in case multiple employees somehow have the same ID, though unlikely with a PK) and another for NO\_DATA\_FOUND.

Declare

sal number;

begin

select salary into sal

from Employee where employee\_id = 999;

Exception

when TOO\_MANY\_ROWS then

dbms\_output.put\_line(‘multiple output’);

when NO\_DATA\_FOUND then

dbms\_output.put\_line(‘invalid emp ir’);

End;

/