# Oracle SQL Coding Test – Master Question Bank

## Advanced SQL

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No | Question | Solution | Explanation |
| 41 | Find top 2 earners in each department | SELECT \* FROM ( SELECT EMP\_NAME, DEPT\_ID, SALARY, ROW\_NUMBER() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY DESC) AS RN FROM EMPLOYEES ) WHERE RN <= 2; | Top N per group using ROW\_NUMBER. |
| 42 | Calculate year-over-year sales | SELECT REGION, EXTRACT(YEAR FROM SALE\_DATE) AS YEAR, SUM(AMOUNT) AS TOTAL, LAG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY EXTRACT(YEAR FROM SALE\_DATE)) AS PREV\_YEAR, ROUND((SUM(AMOUNT) - LAG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY EXTRACT(YEAR FROM SALE\_DATE))) \* 100 / LAG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY EXTRACT(YEAR FROM SALE\_DATE)), 2) AS YOY\_GROWTH FROM SALES GROUP BY REGION, EXTRACT(YEAR FROM SALE\_DATE); | YoY using LAG over yearly aggregates. |
| 43 | Find median salary per department | SELECT DEPT\_ID, PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY SALARY) AS MEDIAN\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID; | Median via PERCENTILE\_CONT. |
| 44 | Find employees with above-average salary growth | WITH SAL\_AVG AS ( SELECT DEPT\_ID, AVG(SALARY) AS AVG\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID ) SELECT e.EMP\_NAME, e.SALARY, s.AVG\_SAL FROM EMPLOYEES e JOIN SAL\_AVG s ON e.DEPT\_ID = s.DEPT\_ID WHERE e.SALARY > s.AVG\_SAL; | Compare employee to dept avg via join. |
| 45 | Calculate 3-month moving average of sales | SELECT REGION, TRUNC(SALE\_DATE, 'MM') AS MONTH, AVG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY TRUNC(SALE\_DATE, 'MM') ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS MOVING\_AVG FROM SALES GROUP BY REGION, TRUNC(SALE\_DATE, 'MM'); | Moving average over monthly aggregates. |
| 46 | Identify employees without managers | SELECT EMP\_NAME FROM EMPLOYEES WHERE MANAGER\_ID IS NULL; | NULL check for manager. |
| 47 | Use CASE to assign performance bonus | SELECT EMP\_NAME, SALARY, CASE WHEN SALARY > 90000 THEN '10% BONUS' WHEN SALARY BETWEEN 70000 AND 90000 THEN '5% BONUS' ELSE 'NO BONUS' END AS BONUS\_CATEGORY FROM EMPLOYEES; | CASE mapping to bonus. |
| 48 | Find duplicate employees by name | SELECT EMP\_NAME, COUNT(\*) AS DUP\_COUNT FROM EMPLOYEES GROUP BY EMP\_NAME HAVING COUNT(\*) > 1; | Detect duplicates using GROUP BY HAVING. |
| 49 | Find employees hired after their manager | SELECT e.EMP\_NAME, m.EMP\_NAME AS MANAGER FROM EMPLOYEES e JOIN EMPLOYEES m ON e.MANAGER\_ID = m.EMP\_ID WHERE e.HIRE\_DATE > m.HIRE\_DATE; | Self-join comparison. |
| 50 | Rank employees globally by salary | SELECT EMP\_NAME, SALARY, RANK() OVER (ORDER BY SALARY DESC) AS GLOBAL\_RANK FROM EMPLOYEES; | Global ranking. |
| 51 | List employees and department average difference | SELECT EMP\_NAME, DEPT\_ID, SALARY, SALARY - AVG(SALARY) OVER (PARTITION BY DEPT\_ID) AS DIFF\_FROM\_AVG FROM EMPLOYEES; | Difference from dept avg using window. |
| 52 | Employees hired in the same month as their manager | SELECT e.EMP\_NAME, m.EMP\_NAME AS MANAGER FROM EMPLOYEES e JOIN EMPLOYEES m ON e.MANAGER\_ID = m.EMP\_ID WHERE TO\_CHAR(e.HIRE\_DATE, 'MM') = TO\_CHAR(m.HIRE\_DATE, 'MM'); | Compare month parts of dates. |
| 53 | Rank sales within each region by date | SELECT SALE\_ID, REGION, SALE\_DATE, AMOUNT, ROW\_NUMBER() OVER (PARTITION BY REGION ORDER BY SALE\_DATE) AS RN FROM SALES; | Per-region row numbering. |
| 54 | Calculate total & cumulative sales per product | SELECT PRODUCT, SUM(AMOUNT) AS TOTAL\_SALES, SUM(SUM(AMOUNT)) OVER (ORDER BY PRODUCT) AS CUM\_TOTAL FROM SALES GROUP BY PRODUCT; | Cumulative total across products. |
| 55 | Find departments with > average salary of company | SELECT DEPT\_ID, AVG(SALARY) AS AVG\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID HAVING AVG(SALARY) > (SELECT AVG(SALARY) FROM EMPLOYEES); | HAVING compares to company average. |
| 56 | Identify inactive employees (not in recent logs) | SELECT e.EMP\_NAME FROM EMPLOYEES e WHERE NOT EXISTS ( SELECT 1 FROM ATTENDANCE a WHERE a.EMP\_ID = e.EMP\_ID AND a.ATTEND\_DATE >= SYSDATE - 30 ); | NOT EXISTS correlated with attendance. |
| 57 | Calculate difference between hire date and today in months | SELECT EMP\_NAME, ROUND(MONTHS\_BETWEEN(SYSDATE, HIRE\_DATE)) AS MONTHS\_WORKED FROM EMPLOYEES; | MONTHS\_BETWEEN usage. |
| 58 | Use CTE to get department salary ranks | WITH DEPT\_SAL AS ( SELECT DEPT\_ID, EMP\_NAME, SALARY, RANK() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY DESC) AS RNK FROM EMPLOYEES ) SELECT \* FROM DEPT\_SAL WHERE RNK <= 3; | CTE with ranking. |
| 59 | Find employees who joined on weekends | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE TO\_CHAR(HIRE\_DATE, 'DY', 'NLS\_DATE\_LANGUAGE=ENGLISH') IN ('SAT','SUN'); | TO\_CHAR weekday extraction. |

## Basic SQL

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No | Question | Solution | Explanation |
| 1 | Find all employees | SELECT \* FROM EMPLOYEES; | Simple select all rows from EMPLOYEES. |
| 2 | Display employee names and salaries | SELECT EMP\_NAME, SALARY FROM EMPLOYEES; | Select specific columns. |
| 3 | Sort employees by salary descending | SELECT EMP\_NAME, SALARY FROM EMPLOYEES ORDER BY SALARY DESC; | Order by salary desc. |
| 4 | Find employees hired after 2020 | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE HIRE\_DATE > TO\_DATE('2020-01-01','YYYY-MM-DD'); | Filter by date. |
| 5 | Get unique department IDs | SELECT DISTINCT DEPT\_ID FROM EMPLOYEES; | Distinct values for DEPT\_ID. |
| 6 | Count employees in each department | SELECT DEPT\_ID, COUNT(\*) AS EMP\_COUNT FROM EMPLOYEES GROUP BY DEPT\_ID; | Aggregate count grouped by DEPT\_ID. |
| 7 | Find total salary for each department | SELECT DEPT\_ID, SUM(SALARY) AS TOTAL\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID; | SUM per department. |
| 8 | Get average salary of all employees | SELECT AVG(SALARY) AS AVG\_SAL FROM EMPLOYEES; | Average salary. |
| 9 | Display employees with salary > 50000 | SELECT EMP\_NAME, SALARY FROM EMPLOYEES WHERE SALARY > 50000; | Filter by salary threshold. |
| 10 | Show employees whose name starts with 'A' | SELECT EMP\_NAME FROM EMPLOYEES WHERE EMP\_NAME LIKE 'A%'; | Pattern match using LIKE. |
| 11 | Get current system date and timestamp | SELECT SYSDATE AS TODAY, SYSTIMESTAMP AS FULL\_TIMESTAMP FROM DUAL; | System date/time functions. |
| 12 | Display employee name and years of service | SELECT EMP\_NAME, FLOOR(MONTHS\_BETWEEN(SYSDATE, HIRE\_DATE)/12) AS YEARS\_WORKED FROM EMPLOYEES; | Compute years from HIRE\_DATE. |
| 13 | Round the current date to nearest month | SELECT ROUND(SYSDATE, 'MM') AS ROUNDED\_DATE FROM DUAL; | ROUND with 'MM' to nearest month. |
| 14 | Find employees hired this year | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE EXTRACT(YEAR FROM HIRE\_DATE) = EXTRACT(YEAR FROM SYSDATE); | Use EXTRACT for year comparison. |
| 15 | Convert date to formatted text | SELECT EMP\_NAME, TO\_CHAR(HIRE\_DATE, 'DD-Mon-YYYY') AS FORMATTED\_DATE FROM EMPLOYEES; | Format date using TO\_CHAR. |
| 16 | Use CASE to categorize salary levels | SELECT EMP\_NAME, SALARY, CASE WHEN SALARY < 50000 THEN 'LOW' WHEN SALARY BETWEEN 50000 AND 80000 THEN 'MEDIUM' ELSE 'HIGH' END AS SAL\_CATEGORY FROM EMPLOYEES; | CASE expression for salary buckets. |
| 17 | Display employees sorted by department and name | SELECT DEPT\_ID, EMP\_NAME FROM EMPLOYEES ORDER BY DEPT\_ID, EMP\_NAME; | Order by multiple columns. |
| 18 | Find number of employees joined each year | SELECT EXTRACT(YEAR FROM HIRE\_DATE) AS JOIN\_YEAR, COUNT(\*) AS EMP\_COUNT FROM EMPLOYEES GROUP BY EXTRACT(YEAR FROM HIRE\_DATE) ORDER BY JOIN\_YEAR; | Group by extracted year. |
| 19 | Find earliest hire date | SELECT MIN(HIRE\_DATE) AS FIRST\_JOINED FROM EMPLOYEES; | MIN aggregate. |
| 20 | Display employee names in uppercase | SELECT UPPER(EMP\_NAME) AS NAME\_UPPER FROM EMPLOYEES; | UPPER string function. |

## Intermediate SQL

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No | Question | Solution | Explanation |
| 21 | Get employees earning above department average | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES e WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | Correlated subquery comparing to dept average. |
| 22 | Show department with total salary > 100000 | SELECT DEPT\_ID, SUM(SALARY) AS TOTAL FROM EMPLOYEES GROUP BY DEPT\_ID HAVING SUM(SALARY) > 100000; | HAVING filter on aggregate. |
| 23 | Display top 3 highest paid employees | SELECT EMP\_NAME, SALARY FROM ( SELECT EMP\_NAME, SALARY, RANK() OVER (ORDER BY SALARY DESC) AS R FROM EMPLOYEES) WHERE R <= 3; | Ranking with analytic function inside inline view. |
| 24 | Show previous employee’s salary using LAG | SELECT EMP\_NAME, SALARY, LAG(SALARY) OVER (ORDER BY HIRE\_DATE) AS PREV\_SAL FROM EMPLOYEES; | LAG analytic function. |
| 25 | Compute salary difference from previous employee | SELECT EMP\_NAME, SALARY, SALARY - LAG(SALARY) OVER (ORDER BY HIRE\_DATE) AS SAL\_DIFF FROM EMPLOYEES; | Use LAG and arithmetic. |
| 26 | Rank employees by department salary | SELECT EMP\_NAME, DEPT\_ID, SALARY, DENSE\_RANK() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY DESC) AS SAL\_RANK FROM EMPLOYEES; | DENSE\_RANK partitioned by dept. |
| 27 | Find second highest salary using subquery | SELECT MAX(SALARY) AS SECOND\_HIGHEST FROM EMPLOYEES WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEES); | Aggregate subquery to get 2nd highest. |
| 28 | Join employee and department tables | SELECT e.EMP\_NAME, d.DEPT\_NAME FROM EMPLOYEES e JOIN DEPARTMENTS d ON e.DEPT\_ID = d.DEPT\_ID; | Inner join between tables. |
| 29 | Left join to include departments with no employees | SELECT d.DEPT\_NAME, e.EMP\_NAME FROM DEPARTMENTS d LEFT JOIN EMPLOYEES e ON d.DEPT\_ID = e.DEPT\_ID; | LEFT JOIN to include unmatched departments. |
| 30 | Total salary by department and position | SELECT DEPT\_ID, JOB\_TITLE, SUM(SALARY) AS TOTAL\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID, JOB\_TITLE; | Group by multiple columns. |
| 31 | Cumulative total salary (window) | SELECT EMP\_NAME, DEPT\_ID, SALARY, SUM(SALARY) OVER (PARTITION BY DEPT\_ID ORDER BY SALARY) AS RUNNING\_TOTAL FROM EMPLOYEES; | Running total per dept. |
| 32 | Average salary by department (window) | SELECT EMP\_NAME, DEPT\_ID, SALARY, AVG(SALARY) OVER (PARTITION BY DEPT\_ID) AS DEPT\_AVG FROM EMPLOYEES; | Window average per dept. |
| 33 | Find employees hired before their department’s average hire date | SELECT EMP\_NAME, HIRE\_DATE, DEPT\_ID FROM EMPLOYEES e WHERE HIRE\_DATE < ( SELECT AVG(HIRE\_DATE) KEEP (DENSE\_RANK FIRST ORDER BY HIRE\_DATE) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID ); | Compare hire date to dept average (approx) using KEEP. |
| 34 | Calculate total sales by region and rank | SELECT REGION, SUM(AMOUNT) AS TOTAL, RANK() OVER (ORDER BY SUM(AMOUNT) DESC) AS SALES\_RANK FROM SALES GROUP BY REGION; | Aggregate + analytic rank. |
| 35 | Use CTE to find employees above company average | WITH AVG\_SAL AS ( SELECT AVG(SALARY) AS AVG\_SALARY FROM EMPLOYEES ) SELECT EMP\_NAME, SALARY FROM EMPLOYEES, AVG\_SAL WHERE EMPLOYEES.SALARY > AVG\_SAL.AVG\_SALARY; | CTE to compute company average. |
| 36 | Monthly sales growth (LAG) | SELECT REGION, TRUNC(SALE\_DATE, 'MM') AS MONTH, SUM(AMOUNT) AS MONTH\_SALES, SUM(AMOUNT) - LAG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY TRUNC(SALE\_DATE, 'MM')) AS SALES\_DIFF FROM SALES GROUP BY REGION, TRUNC(SALE\_DATE, 'MM'); | Monthly aggregation and LAG comparison. |
| 37 | Employees hired in last 90 days | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE HIRE\_DATE >= SYSDATE - 90; | Date arithmetic using SYSDATE. |
| 38 | Add 3 months to hire date | SELECT EMP\_NAME, ADD\_MONTHS(HIRE\_DATE, 3) AS REVIEW\_DATE FROM EMPLOYEES; | ADD\_MONTHS usage. |
| 39 | Employees with hire date in next 15 days (future schedule) | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE HIRE\_DATE BETWEEN SYSDATE AND SYSDATE + 15; | BETWEEN for date range. |
| 40 | Find most recent hire per department | SELECT \* FROM ( SELECT EMP\_NAME, DEPT\_ID, HIRE\_DATE, ROW\_NUMBER() OVER (PARTITION BY DEPT\_ID ORDER BY HIRE\_DATE DESC) AS RN FROM EMPLOYEES ) WHERE RN = 1; | ROW\_NUMBER to get latest per group. |

## Subquery-Focused

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No | Question | Solution | Explanation |
| 121 | Find employees earning above the company average salary | SELECT EMP\_NAME, SALARY FROM EMPLOYEES WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES); | Scalar subquery to get company average. |
| 122 | Find employees earning more than their department average | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES e WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | Correlated subquery. |
| 123 | Get the department(s) with the maximum number of employees | SELECT DEPT\_ID, COUNT(\*) AS EMP\_COUNT FROM EMPLOYEES GROUP BY DEPT\_ID HAVING COUNT(\*) = (SELECT MAX(COUNT(\*)) FROM EMPLOYEES GROUP BY DEPT\_ID); | Aggregate subquery in HAVING. |
| 124 | Find the employee(s) with the second highest salary | SELECT EMP\_NAME, SALARY FROM EMPLOYEES WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEES WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEES)); | Nested aggregated subqueries. |
| 125 | List employees who joined before their manager | SELECT e.EMP\_NAME, e.HIRE\_DATE, m.EMP\_NAME AS MANAGER, m.HIRE\_DATE AS MANAGER\_HIRE FROM EMPLOYEES e JOIN EMPLOYEES m ON e.MANAGER\_ID = m.EMP\_ID WHERE e.HIRE\_DATE < m.HIRE\_DATE; | Self-join to compare dates. |
| 126 | Find departments where total salary > company average | SELECT DEPT\_ID, SUM(SALARY) AS TOTAL\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID HAVING SUM(SALARY) > (SELECT AVG(SALARY)\*COUNT(\*) FROM EMPLOYEES); | Compare group sum to company metric. |
| 127 | Employees who earn the maximum in their department | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES e WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | Correlated MAX per dept. |
| 128 | Find departments that have no employees | SELECT DEPT\_NAME FROM DEPARTMENTS d WHERE NOT EXISTS (SELECT 1 FROM EMPLOYEES e WHERE e.DEPT\_ID = d.DEPT\_ID); | NOT EXISTS to find empty depts. |
| 129 | Display employees who joined in the same year as the CEO | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE EXTRACT(YEAR FROM HIRE\_DATE) = (SELECT EXTRACT(YEAR FROM HIRE\_DATE) FROM EMPLOYEES WHERE JOB\_TITLE = 'CEO'); | Scalar subquery referencing CEO row. |
| 130 | Find employees whose salary is higher than their manager’s | SELECT EMP\_NAME, SALARY FROM EMPLOYEES e WHERE SALARY > (SELECT SALARY FROM EMPLOYEES WHERE EMP\_ID = e.MANAGER\_ID); | Correlated subquery comparing to manager. |
| 131 | Get employees working in the department with minimum average salary | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES WHERE DEPT\_ID IN (SELECT DEPT\_ID FROM EMPLOYEES GROUP BY DEPT\_ID HAVING AVG(SALARY) = (SELECT MIN(AVG(SALARY)) FROM EMPLOYEES GROUP BY DEPT\_ID)); | Nested grouping aggregates. |
| 132 | Find employees whose department total salary > 1.5× overall average | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES e WHERE (SELECT SUM(SALARY) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID) > 1.5 \* (SELECT AVG(SALARY) FROM EMPLOYEES); | Compare dept sum with company average. |
| 133 | Find top 3 salaries using subquery with ROWNUM | SELECT EMP\_NAME, SALARY FROM (SELECT EMP\_NAME, SALARY FROM EMPLOYEES ORDER BY SALARY DESC) WHERE ROWNUM <= 3; | Inline view with ROWNUM limiting. |
| 134 | Find employees whose salary is within ±10% of their department average | SELECT EMP\_NAME, SALARY, DEPT\_ID FROM EMPLOYEES e WHERE SALARY BETWEEN (SELECT AVG(SALARY)\*0.9 FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID) AND (SELECT AVG(SALARY)\*1.1 FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | Correlated subqueries for bounds. |
| 135 | Employees from departments with more than 5 people | SELECT EMP\_NAME, DEPT\_ID FROM EMPLOYEES WHERE DEPT\_ID IN (SELECT DEPT\_ID FROM EMPLOYEES GROUP BY DEPT\_ID HAVING COUNT(\*) > 5); | HAVING in subquery. |
| 136 | Employees earning more than average of earlier hires | SELECT EMP\_NAME, SALARY, HIRE\_DATE FROM EMPLOYEES e1 WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES e2 WHERE e2.HIRE\_DATE < e1.HIRE\_DATE); | Correlated subquery referencing earlier hires. |
| 137 | Employees who joined in same month as the earliest hire | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE TO\_CHAR(HIRE\_DATE, 'MM-YYYY') = (SELECT TO\_CHAR(MIN(HIRE\_DATE), 'MM-YYYY') FROM EMPLOYEES); | Match month-year to earliest hire. |
| 138 | Display departments where total commission > total bonus | SELECT d.DEPT\_ID, d.DEPT\_NAME FROM DEPARTMENTS d WHERE (SELECT SUM(COMMISSION) FROM EMPLOYEES e WHERE e.DEPT\_ID = d.DEPT\_ID) > (SELECT SUM(BONUS) FROM BONUS\_TABLE b WHERE b.DEPT\_ID = d.DEPT\_ID); | Compare sums across related tables. |
| 139 | Find employees whose job title is unique within the company | SELECT EMP\_NAME, JOB\_TITLE FROM EMPLOYEES e WHERE 1 = (SELECT COUNT(\*) FROM EMPLOYEES WHERE JOB\_TITLE = e.JOB\_TITLE); | COUNT(\*) correlated to verify uniqueness. |
| 140 | Employees hired first in their department | SELECT EMP\_NAME, DEPT\_ID, HIRE\_DATE FROM EMPLOYEES e WHERE HIRE\_DATE = (SELECT MIN(HIRE\_DATE) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | MIN per department via correlated subquery. |

## Tricky/Advanced SQL

|  |  |  |  |
| --- | --- | --- | --- |
| Serial No | Question | Solution | Explanation |
| 61 | Extract domain name from email address | SELECT EMAIL, SUBSTR(EMAIL, INSTR(EMAIL, '@') + 1) AS DOMAIN FROM USERS; | Use INSTR to find @ and SUBSTR to extract domain. |
| 62 | Reverse each employee’s name | SELECT EMP\_NAME, REVERSE(EMP\_NAME) AS REVERSED FROM EMPLOYEES; | REVERSE string function (note: some Oracle versions require custom function). |
| 63 | Find employees with palindrome names | SELECT EMP\_NAME FROM EMPLOYEES WHERE EMP\_NAME = REVERSE(EMP\_NAME); | Compare name to its reversed form. |
| 64 | Replace all digits with X in phone numbers | SELECT PHONE\_NUMBER, REGEXP\_REPLACE(PHONE\_NUMBER, '[0-9]', 'X') AS MASKED FROM CONTACTS; | REGEXP\_REPLACE for masking digits. |
| 65 | Extract initials from full name | SELECT FULL\_NAME, REGEXP\_REPLACE(FULL\_NAME, '(\b[a-zA-Z])[a-zA-Z]\* ?', '\1.') AS INITIALS FROM EMPLOYEES; | Use regex to capture initials. |
| 66 | Get length excluding spaces | SELECT EMP\_NAME, LENGTH(REPLACE(EMP\_NAME, ' ', '')) AS LEN\_NO\_SPACE FROM EMPLOYEES; | Remove spaces then measure length. |
| 67 | Compare strings ignoring case | SELECT \* FROM EMPLOYEES WHERE UPPER(CITY) = UPPER('chennai'); | Use UPPER for case-insensitive compare. |
| 68 | Replace multiple spaces with a single space | SELECT REGEXP\_REPLACE(DESCRIPTION, '\s+', ' ') AS CLEAN\_TEXT FROM DOCUMENTS; | Regex to collapse whitespace. |
| 69 | Extract only digits from a text | SELECT REGEXP\_REPLACE('Order#A123B45', '[^0-9]', '') AS DIGITS\_ONLY FROM DUAL; | Strip non-digits. |
| 70 | Find last word in full name | SELECT FULL\_NAME, REGEXP\_SUBSTR(FULL\_NAME, '[^ ]+$') AS LAST\_NAME FROM EMPLOYEES; | Regex to fetch last token. |
| 71 | Find last Friday of the current month | SELECT NEXT\_DAY(LAST\_DAY(SYSDATE) - 7, 'FRIDAY') AS LAST\_FRIDAY FROM DUAL; | Compute last Friday by combining LAST\_DAY and NEXT\_DAY. |
| 72 | Calculate employee age precisely | SELECT EMP\_NAME, TRUNC(MONTHS\_BETWEEN(SYSDATE, DOB)/12, 1) AS AGE\_YEARS FROM EMPLOYEES; | MONTHS\_BETWEEN and division for years. |
| 73 | Display hire date and next appraisal date (after 6 months) | SELECT EMP\_NAME, HIRE\_DATE, ADD\_MONTHS(HIRE\_DATE, 6) AS APPRAISAL\_DATE FROM EMPLOYEES; | ADD\_MONTHS to compute future date. |
| 74 | Get employees hired on weekends | SELECT EMP\_NAME, HIRE\_DATE FROM EMPLOYEES WHERE TO\_CHAR(HIRE\_DATE, 'DY', 'NLS\_DATE\_LANGUAGE=ENGLISH') IN ('SAT','SUN'); | Use TO\_CHAR weekday name. |
| 75 | Find number of working days between two dates | SELECT COUNT(\*) FROM ( SELECT TO\_DATE('2025-10-01','YYYY-MM-DD') + LEVEL - 1 AS DT FROM DUAL CONNECT BY LEVEL <= (TO\_DATE('2025-10-31','YYYY-MM-DD') - TO\_DATE('2025-10-01','YYYY-MM-DD')) + 1 ) WHERE TO\_CHAR(DT, 'DY', 'NLS\_DATE\_LANGUAGE=ENGLISH') NOT IN ('SAT','SUN'); | Generate date range with CONNECT BY and filter weekends. |
| 76 | Round to nearest month | SELECT ROUND(SYSDATE, 'MM') AS NEAREST\_MONTH FROM DUAL; | ROUND with 'MM'. |
| 77 | Get quarter name | SELECT TO\_CHAR(SYSDATE, '"Q"Q') AS QUARTER FROM DUAL; | TO\_CHAR format for quarter. |
| 78 | Find first Monday of next month | SELECT NEXT\_DAY(TRUNC(ADD\_MONTHS(SYSDATE,1),'MM') - 1, 'MONDAY') AS FIRST\_MON FROM DUAL; | NEXT\_DAY over start of month. |
| 79 | Calculate experience in years and months | SELECT EMP\_NAME, TRUNC(MONTHS\_BETWEEN(SYSDATE, HIRE\_DATE)/12) || ' Years ' || MOD(TRUNC(MONTHS\_BETWEEN(SYSDATE, HIRE\_DATE)),12) || ' Months' AS EXPERIENCE FROM EMPLOYEES; | Combine MONTHS\_BETWEEN with integer math. |
| 80 | Add 10 working days to current date | SELECT NEXT\_DAY(SYSDATE + 10, 'MONDAY') AS NEXT\_WORK\_DAY FROM DUAL; | Approximate business-day addition. |
| 81 | Round and truncate salary | SELECT SALARY, ROUND(SALARY, -3) AS ROUNDED, TRUNC(SALARY, -3) AS TRUNCATED FROM EMPLOYEES; | Numeric rounding and truncation. |
| 82 | Find highest, lowest, average salary per department | SELECT DEPT\_ID, MAX(SALARY), MIN(SALARY), ROUND(AVG(SALARY),2) FROM EMPLOYEES GROUP BY DEPT\_ID; | Basic aggregates per group. |
| 83 | Calculate total commission including NULLs as zero | SELECT SUM(NVL(COMMISSION\_PCT,0)) AS TOTAL\_COMM FROM EMPLOYEES; | NVL to treat NULL as 0. |
| 84 | Find % contribution of each department to total salary | SELECT DEPT\_ID, ROUND(SUM(SALARY)/SUM(SUM(SALARY)) OVER ()\*100,2) AS PCT\_TOTAL FROM EMPLOYEES GROUP BY DEPT\_ID; | Window total to compute percentage. |
| 85 | Rank departments by total payroll | SELECT DEPT\_ID, SUM(SALARY) AS TOTAL, RANK() OVER (ORDER BY SUM(SALARY) DESC) AS RANKING FROM EMPLOYEES GROUP BY DEPT\_ID; | RANK on grouped totals. |
| 86 | Find department with max average salary | SELECT DEPT\_ID FROM ( SELECT DEPT\_ID, AVG(SALARY) AS AVG\_SAL, RANK() OVER (ORDER BY AVG(SALARY) DESC) AS RN FROM EMPLOYEES GROUP BY DEPT\_ID ) WHERE RN=1; | Use analytic RANK over grouped averages. |
| 87 | Compute difference between current and previous record | SELECT EMP\_ID, SALARY, SALARY - LAG(SALARY) OVER (ORDER BY HIRE\_DATE) AS SAL\_DIFF FROM EMPLOYEES; | LAG for previous value. |
| 88 | Running total of salary per department | SELECT DEPT\_ID, EMP\_NAME, SALARY, SUM(SALARY) OVER (PARTITION BY DEPT\_ID ORDER BY EMP\_NAME) AS RUNNING\_TOTAL FROM EMPLOYEES; | SUM as window function. |
| 89 | Average of top 3 salaries | SELECT AVG(SALARY) AS TOP3\_AVG FROM ( SELECT SALARY FROM EMPLOYEES ORDER BY SALARY DESC ) WHERE ROWNUM <= 3; | Inline view with ROWNUM. |
| 90 | Find 90th percentile salary | SELECT PERCENTILE\_CONT(0.9) WITHIN GROUP (ORDER BY SALARY) AS P90\_SAL FROM EMPLOYEES; | Percentile function. |
| 91 | Employees with above-department average salary (subquery) | SELECT EMP\_NAME, DEPT\_ID, SALARY FROM EMPLOYEES e WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES WHERE DEPT\_ID = e.DEPT\_ID); | Correlated subquery. |
| 92 | Second highest salary using CTE | WITH SALS AS ( SELECT SALARY, DENSE\_RANK() OVER (ORDER BY SALARY DESC) AS RN FROM EMPLOYEES ) SELECT SALARY FROM SALS WHERE RN = 2; | CTE + DENSE\_RANK. |
| 93 | Count employees per salary range | SELECT CASE WHEN SALARY < 50000 THEN 'Low' WHEN SALARY BETWEEN 50000 AND 90000 THEN 'Mid' ELSE 'High' END AS SAL\_RANGE, COUNT(\*) AS EMP\_COUNT FROM EMPLOYEES GROUP BY CASE WHEN SALARY < 50000 THEN 'Low' WHEN SALARY BETWEEN 50000 AND 90000 THEN 'Mid' ELSE 'High' END; | CASE in GROUP BY. |
| 94 | Find employees in departments where total salary > 1M | SELECT EMP\_NAME, DEPT\_ID FROM EMPLOYEES WHERE DEPT\_ID IN (SELECT DEPT\_ID FROM EMPLOYEES GROUP BY DEPT\_ID HAVING SUM(SALARY) > 1000000); | HAVING in subquery. |
| 95 | Get top 2 paid employees per department using CTE | WITH DEPT\_TOP AS ( SELECT EMP\_NAME, DEPT\_ID, SALARY, ROW\_NUMBER() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY DESC) AS RN FROM EMPLOYEES ) SELECT \* FROM DEPT\_TOP WHERE RN <= 2; | CTE with ROW\_NUMBER. |
| 96 | Calculate % difference from department average | SELECT EMP\_NAME, DEPT\_ID, SALARY, ROUND((SALARY - AVG(SALARY) OVER (PARTITION BY DEPT\_ID)) / AVG(SALARY) OVER (PARTITION BY DEPT\_ID) \* 100, 2) AS PCT\_DIFF FROM EMPLOYEES; | Window avg and percentage. |
| 97 | Employees who joined before their manager | SELECT e.EMP\_NAME, m.EMP\_NAME AS MANAGER FROM EMPLOYEES e JOIN EMPLOYEES m ON e.MANAGER\_ID = m.EMP\_ID WHERE e.HIRE\_DATE < m.HIRE\_DATE; | Self-join. |
| 98 | Use CASE to group by appraisal year | SELECT EMP\_NAME, CASE WHEN HIRE\_DATE < ADD\_MONTHS(SYSDATE,-24) THEN 'Eligible' ELSE 'Not Eligible' END AS STATUS FROM EMPLOYEES; | CASE with date arithmetic. |
| 99 | Find departments with same total salary | SELECT d1.DEPT\_ID, d2.DEPT\_ID AS MATCH\_DEPT, d1.TOTAL FROM ( SELECT DEPT\_ID, SUM(SALARY) AS TOTAL FROM EMPLOYEES GROUP BY DEPT\_ID ) d1 JOIN ( SELECT DEPT\_ID, SUM(SALARY) AS TOTAL FROM EMPLOYEES GROUP BY DEPT\_ID ) d2 ON d1.TOTAL = d2.TOTAL AND d1.DEPT\_ID < d2.DEPT\_ID; | Self-join on aggregates to find equal totals. |
| 100 | Identify most senior employee per region | SELECT REGION, EMP\_NAME, HIRE\_DATE FROM ( SELECT REGION, EMP\_NAME, HIRE\_DATE, ROW\_NUMBER() OVER (PARTITION BY REGION ORDER BY HIRE\_DATE ASC) AS RN FROM EMPLOYEES ) WHERE RN = 1; | ROW\_NUMBER to get earliest hire per region. |
| 101 | Running 3-month moving total | SELECT REGION, TRUNC(SALE\_DATE, 'MM') AS MONTH, SUM(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY TRUNC(SALE\_DATE, 'MM') ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS MOV\_TOTAL FROM SALES GROUP BY REGION, TRUNC(SALE\_DATE, 'MM'); | Moving total over months. |
| 102 | Cumulative count of employees by hire date | SELECT HIRE\_DATE, COUNT(\*) OVER (ORDER BY HIRE\_DATE) AS TOTAL\_JOINED FROM EMPLOYEES; | Cumulative count analytic. |
| 103 | Find previous department’s average salary | SELECT DEPT\_ID, AVG\_SAL, LAG(AVG\_SAL) OVER (ORDER BY DEPT\_ID) AS PREV\_DEPT\_AVG FROM ( SELECT DEPT\_ID, AVG(SALARY) AS AVG\_SAL FROM EMPLOYEES GROUP BY DEPT\_ID ); | LAG over grouped avg. |
| 104 | Calculate salary percentile per department | SELECT EMP\_NAME, DEPT\_ID, SALARY, PERCENT\_RANK() OVER (PARTITION BY DEPT\_ID ORDER BY SALARY) AS SAL\_PERCENT FROM EMPLOYEES; | PERCENT\_RANK analytic. |
| 105 | Running product of sales (LOG trick) | SELECT REGION, EXP(SUM(LOG(AMOUNT)) OVER (PARTITION BY REGION ORDER BY SALE\_DATE)) AS RUNNING\_PRODUCT FROM SALES; | Use LOG+EXP to compute running product. |
| 106 | Identify salary gaps | SELECT SALARY, LEAD(SALARY) OVER (ORDER BY SALARY) - SALARY AS GAP FROM EMPLOYEES; | LEAD to compute gap to next salary. |
| 107 | Group customers into quartiles by spending | SELECT CUSTOMER\_ID, TOTAL\_SPEND, NTILE(4) OVER (ORDER BY TOTAL\_SPEND DESC) AS SPENDING\_QUARTILE FROM CUSTOMERS; | NTILE to bucket customers. |
| 108 | Compare current month vs previous month sales | SELECT REGION, TRUNC(SALE\_DATE, 'MM') AS MONTH, SUM(AMOUNT) AS MONTH\_SALES, SUM(AMOUNT) - LAG(SUM(AMOUNT)) OVER (PARTITION BY REGION ORDER BY TRUNC(SALE\_DATE, 'MM')) AS DELTA FROM SALES GROUP BY REGION, TRUNC(SALE\_DATE, 'MM'); | Month over month comparison. |
| 109 | Find top 1 sale per product | SELECT \* FROM ( SELECT PRODUCT, SALE\_ID, AMOUNT, ROW\_NUMBER() OVER (PARTITION BY PRODUCT ORDER BY AMOUNT DESC) AS RN FROM SALES ) WHERE RN = 1; | Top per product with ROW\_NUMBER. |
| 110 | Calculate cumulative distinct customer count | SELECT ORDER\_DATE, COUNT(DISTINCT CUSTOMER\_ID) OVER (ORDER BY ORDER\_DATE) AS CUM\_CUSTOMERS FROM ORDERS; | Cumulative distinct count (note: in Oracle 19c+, COUNT(DISTINCT) as analytic has restrictions). |
| 111 | Coalesce vs NVL demo | SELECT NVL(MOBILE, 'No Number') AS NVL\_OUT, COALESCE(MOBILE, PHONE, 'No Contact') AS COALESCE\_OUT FROM CONTACTS; | NVL and COALESCE differences. |
| 112 | Pivot total sales by region | SELECT \* FROM ( SELECT REGION, PRODUCT, AMOUNT FROM SALES ) PIVOT ( SUM(AMOUNT) FOR REGION IN ('NORTH' AS NORTH, 'SOUTH' AS SOUTH, 'EAST' AS EAST, 'WEST' AS WEST) ); | PIVOT clause for cross-tab. |
| 113 | Unpivot product columns | SELECT REGION, PRODUCT, SALES\_AMOUNT FROM SALES\_DATA UNPIVOT (SALES\_AMOUNT FOR PRODUCT IN (LAPTOP, MOBILE, TABLET)); | UNPIVOT to normalize columns. |
| 114 | JSON extraction | SELECT JSON\_VALUE(DATA, '$.customer.name') AS CUSTOMER\_NAME FROM ORDERS\_JSON; | JSON\_VALUE to fetch data from JSON column. |
| 115 | Generate series of dates (recursive CTE) | WITH DATES (DT) AS ( SELECT TRUNC(SYSDATE) FROM DUAL UNION ALL SELECT DT + 1 FROM DATES WHERE DT + 1 < TRUNC(SYSDATE) + 10 ) SELECT \* FROM DATES; | Recursive CTE to generate dates. |
| 116 | Identify gaps in employee IDs | SELECT EMP\_ID + 1 AS MISSING\_ID FROM EMPLOYEES e WHERE NOT EXISTS (SELECT 1 FROM EMPLOYEES WHERE EMP\_ID = e.EMP\_ID + 1); | Detect missing sequential IDs. |
| 117 | Find employees with duplicate emails | SELECT EMAIL, COUNT(\*) AS CNT FROM EMPLOYEES GROUP BY EMAIL HAVING COUNT(\*) > 1; | Find duplicate emails. |
| 118 | Create comma-separated list of employees per department | SELECT DEPT\_ID, LISTAGG(EMP\_NAME, ', ') WITHIN GROUP (ORDER BY EMP\_NAME) AS EMP\_LIST FROM EMPLOYEES GROUP BY DEPT\_ID; | LISTAGG to aggregate strings. |
| 119 | Generate hash of employee info | SELECT EMP\_ID, STANDARD\_HASH(EMP\_NAME || SALARY || DEPT\_ID, 'SHA256') AS HASH\_VAL FROM EMPLOYEES; | STANDARD\_HASH for deterministic hashing. |
| 120 | Find the longest-serving employee using analytic min | SELECT EMP\_NAME, HIRE\_DATE FROM ( SELECT EMP\_NAME, HIRE\_DATE, MIN(HIRE\_DATE) OVER () AS EARLIEST FROM EMPLOYEES ) WHERE HIRE\_DATE = EARLIEST; | MIN as analytic to find earliest hire. |