**J2EE/TCLD-01M/R12/01**
Online Test
Instructor and Examiner:
MD. ABDUL BARI

## Test Results

| surname | name | user | points |
|---|---|---|---|
| siraj | Sirajuddin Ahmed | siraj | 16.277 ( 48%) |

**test: R-14 Oracle mock test 2**

| | |
|---|---|
| start time: | 2011-11-17 09:27:25 |
| end time: | 2011-11-17 10:06:54 |
| time: | 00:39:29 |
| test time [min]: | 40 |
| basic points: | 1.000 |
| points for wrong answer: | 0.000 |
| points for no answer: | 0.000 |
| max score: | 34.000 |
| correct: | 16 ( 47%) |

R-14 Oracle mock test 2

| # | points | IP | start [hh:mm:ss] | end [hh:mm:ss] | time [mm:ss] | reaction [sec] |
|---|---|---|---|---|---|---|
| 1 S | 0.000 | 281473913978936 | 09:25:46 | 09:27:31 | 01:45 | 94.485 |

Examine the description of the EMPLOYEES table:
EMP_ID NUMBER(4) NOT NULL
LAST_NAME VARCHAR2(30) NOT NULL
FIRST_NAME VARCHAR2(30)
DEPT_ID NUMBER(2)
JOB_CAT VARCHARD2(30)
SALARY NUMBER(8,2)
Which statement shows the maximum salary paid in each job category of each department?

| | | |
|---|---|---|
| - | 1 | SELECT dept_id, job_cat, MAX(salary)<br>FROM employees<br>WHERE salary > MAX(salary); |
| | 2 | SELECT dept_id, job_cat, MAX(salary)<br>FROM employees<br>GROUP BY dept_id, job_cat, salary; |
| | 3 | SELECT dept_id, job_cat, MAX(salary)<br>FROM employees<br>GROUP BY dept_id, job_cat; |
| | 4 | SELECT dept_id, job_cat, MAX(salary)<br>FROM employees<br>GROUP BY dept_id; |
| | 5 | SELECT dept_id, job_cat, MAX(salary)<br>FROM employees; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 S | 1.000 | 281473913978936 | 10:03:00 | 10:04:05 | 01:05 | 51.609 |

Management has asked you to calculate the value 12*salary* comossion_pct for all the employees in the EMP table. The EMP table contains these columns:

LAST NAME VARCNAR2(35) NOT NULL
SALARY NUMBER(9,2) NOT NULL
COMMISION_PCT NUMBER(4,2)

Which statement ensures that a value is displayed in the calculated columns for all employees?

***explanation***
This SELECT statement provides correct usage of NVL function to calculate columns for all employees. Oracle give you possibility to substitute a value in place of NULL. The basic syntax for NVL() is NVL(column_name, value_if_null). Notice that the column specified in NVL() contains an actual value. That value is what Oracle returns; when the column is NULL, the special string is returned. The value specified to be returned if the column value is NULL must be the same datatype as the column specified.

| | | |
|---|---|---|
| + | 1 | SELECT last_name, 12*salary*(nvl(commission_pct,0))<br>FROM emp; |
| | 2 | SELECT last_name, 12*salary* (commission_pct,0)<br>FROM emp; |
| | 3 | SELECT last_name, 12*salary*(decode(commission_pct,0))<br>FROM emp; |
| | 4 | SELECT last_name, 12*salary*commison_pct<br>FROM emp; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 M | 1.000 | 281473913978936 | 09:32:05 | 09:33:50 | 01:45 | 104.11 |

Examine the description of the STUDENTS table:
STD_ID NUMBER(4)
COURSE_ID VARCHARD2(10)
START_DATE DATE
END_DATE DATE.

| | | |
|---|---|---|
| | | Which two aggregate functions are valid on the START_DATE column? (Choose two) |
| + | 1 | COUNT(start_date) |
| + | 2 | MIN(start_date) |
| + | 3 | AVG(start_date, end_date) |
| + | 4 | AVG(start_date) |
| + | 5 | MAXIMUM(start_date) |
| + | 6 | SUM(start_date) |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 S | 1.000 | 281473913978936 | 09:31:02 | 09:32:05 | 01:03 | 57.703 |

The EMPLOYEE tables has these columns:
LAST_NAME VARCNAR2(35)
SALARY NUMBER(8,2)
COMMISSION_PCT NUMBER(5,2)
You want to display the name and annual salary multiplied by the commission_pct for all employees. For records that have a NULL commission_pct, a zero must be displayed against the calculated column. Which SQL statement displays the desired results?

***explanation***

This SELECT statement provides correct usage of NVL function to calculate columns for all employees. Oracle give you possibility to substitute a value in place of NULL. The basic syntax for NVL() is NVL(column_name, value_if_null). Notice that the column specified in NVL() contains an actual value. That value is what Oracle returns; when the column is NULL, the special string is returned. The value specified to be returned if the column value is NULL must be the same datatype as the column specified.

| | | |
|---|---|---|
| | 1 | SELECT last_name, (salary * 12) * commission_pct<br>FROM EMPLOYEES; |
| + | 2 | SELECT last_name, (salary * 12) * NVL(commission_pct, 0)<br>FROM EMPLOYEES; |
| | 3 | SELECT last_name, (salary * 12) * NVL2(commission_pct, 0)<br>FROM EMPLOYEES; |
| | 4 | SELECT last_name, (salary * 12) * IFNULL(commission_pct,0)<br>FROM EMPLOYEES; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 S | 0.000 | 281473913978936 | 10:05:08 | 10:06:30 | 01:22 | 64.032 |

You would like to display the system date in the format "Monday, 01 June, 2001".
Which SELECT statement should you use?

***explanation***

This answer is correct: “Day” shows the day spelled out, “DD” shows the two-digit date, “Month” provides the month spelled out, “YYYY” shows the four-digit year. “FMDay” is special format mask to suppresses the extra spaces between the name of the day and the number of the date.

| | | |
|---|---|---|
| | 1 | SELECT TO_DATE(SYSDATE, 'FMDY, DDD Month, YYYY')<br>FROM dual; |
| | 2 | SELECT TO_DATE(SYSDATE, 'FMDAY, DD Month, YYYY')<br>FROM dual; |
| | 3 | SELECT TO_CHAR(SYSDATE, 'FMDay, DD Month, YYYY')<br>FROM dual; |
| | 4 | SELECT TO_CHAR(SYSDATE, 'FMDD, DY Month, 'YYY')<br>FROM dual; |
| - | 5 | SELECT TO_CHAR(SYSDATE, 'FMDY, DDD Month, YYYY')<br>FROM dual; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6 S | 0.000 | 281473913978936 | 09:46:31 | 09:47:20 | 00:49 | 45.985 |

Evaluate the SQL statement:
SELECT ROUND(TRUNC(MOD(1600,10),-1),2)
FROM dual;
What will be displayed?

***explanation***

Result will be 0. MOD(x,y) function calculates the modulus of x, defined in long division as the integer remainder when x is divided by y until no further whole number can be produced.
TRUNC() function truncates x to the decimal precision of y. ROUND(x,y) rounds x to the decimal precision of y.

| | | |
|---|---|---|
| | 1 | 1 |
| | 2 | 0 |
| - | 3 | An error statement |
| | 4 | 0.00 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 M | 0.667 | 281473913978936 | 09:29:23 | 09:29:58 | 00:35 | 28.891 |

Which three SELECT statements displays 2000 in the format “$2,000.00”? (Choose three)

| | | |
|---|---|---|
| - | 1 | SELECT TO_CHAR (2000, ‘$0,000.00’)<br>FROM dual; |
| + | 2 | SELECT TO_CHAR (2000, ‘$N,NNN.NN’)<br>FROM dual; |
| + | 3 | SELECT TO_CHAR (2000, ‘$9,999.99’)<br>FROM dual; |
| + | 4 | SELECT TO_CHAR (2000, ‘$#,###.##’)<br>FROM dual; |
| + | 5 | SELECT TO_CHAR (2000, ‘$9,999.00’)<br>FROM dual; |

| | | |
|---|---|---|
| - | 6 | SELECT TO_CHAR (2000, '$2,000.00')<br>FROM dual; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 M | 0.333 | 281473913978936 | 10:04:05 | 10:05:08 | 01:03 | 45.703 |

Which two are true about aggregate functions? (Choose two.)

***explanation***

It is possible to mix single row columns with aggregate functions in the column list of a SELECT statement by grouping on the single row columns. Also it is acceptable to pass column names, expressions, constraints, or other functions as parameters to an aggregate function.

| | | |
|---|---|---|
| + | 1 | You can use aggregate functions in any clause of a SELECT statement. |
| - | 2 | You cannot group the rows of a table by more than one column while using aggregate functions. |
| + | 3 | You can use aggregate functions only in the column list of the SELECT clause and in the WHERE clause of a SELECT statement. |
| - | 4 | You can pass column names, expressions, constants, or functions as parameters to an aggregate function. |
| - | 5 | You can use aggregate functions on a table, only by grouping the whole table as one single group. |
| - | 6 | You can mix single row columns with aggregate functions in the column list of a SELECT statement by grouping on the single row columns. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 M | 0.429 | 281473913978936 | 09:58:29 | 09:59:32 | 01:03 | 59.437 |

Which four statements correctly describe functions that are available in SQL? (Choose four)

***explanation***

INSTR returns the numeric position of a named character. DECODE translates an expression after comparing it to each search value. TRIM trims the heading of trailing characters (or both)

| | | |
|---|---|---|
| + | 1 | DECODE translates an expression after comparing it to each search value. |
| - | 2 | TRUNCATE rounds the column, expression, or value to n decimal places. |
| - | 3 | TRIM trims the heading of trailing characters (or both) from a character string. |
| - | 4 | NVL2 returns the first non-null expression in the expression list. |
| + | 5 | INSTR returns the numeric position of a named character. |
| - | 6 | NULLIF compares two expressions and returns null if they are equal, or the first expression if they are not equal. |
| + | 7 | NVL compares two expressions and returns null if they are equal, or the first expression of they are not equal. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 10 S | 0.000 | 281473913978936 | 09:23:57 | 10:06:54 | 42:57 | 21.969 |

Which clause should you use to exclude group results?

| | | |
|---|---|---|
| | 1 | ORDER BY |
| | 2 | WHERE |
| - | 3 | GROUP BY |
| | 4 | RESTRICT |
| | 5 | HAVING |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11 S | 1.000 | 281473913978936 | 10:00:35 | 10:01:32 | 00:57 | 45.719 |

In a SELECT statement that includes a WHERE clause, where is the GROUP BY clause placed in the SELECT statement?

***explanation***

The GROUP BY clause can be place only after the WHERE clause, or after FROM clause if there is no the WHERE clause in the statement.

| | | |
|---|---|---|
| + | 1 | After the WHERE clause |
| | 2 | Before the WHERE clause |
| | 3 | Immediately after the SELECT clause |
| | 4 | Before the FROM clause |
| | 5 | After the ORDER BY clause |

| | | | | | | |
|---|---|---|---|---|---|---|
| 12 M | 0.500 | 281473913978936 | 09:38:24 | 09:40:30 | 02:06 | 116.454 |

Which two are character manipulation functions? (Choose two.)

| | | |
|---|---|---|
| - | 1 | TRUNC |
| + | 2 | TO_DATE |
| - | 3 | CASE |
| - | 4 | TRIM |
| + | 5 | MOD |
| + | 6 | REPLACE |

| | | | | | | |
|---|---|---|---|---|---|---|
| 13 S | 1.000 | 281473913978936 | 09:54:59 | 09:55:34 | 00:35 | 31.906 |

Which SELECT statement should you use to extract the year from the system date and display it in the format "1998"?

***explanation***

Function TO_CHAR(x, y) converts the value x to a character or converts a date to a character string using formatting conventions.

| | | |
|---|---|---|
| | 1 | SELECT DECODE(SUBSTR(SYSDATE, 8), 'year')<br>FROM dual; |

| | | |
|---|---|---|
| | 2 | SELECT TO_CHAR(SUBSTR(SYSDATE, 8,2),'yyyy')<br>FROM dual; |
| | 3 | SELECT DECODE(SUBSTR(SYSDATE, 8), 'YYYY')<br>FROM dual; |
| + | 4 | SELECT TO_CHAR(SYSDATE,'yyyy')<br>FROM dual; |
| | 5 | SELECT TO_DATE(SYSDATE,'yyyy')<br>FROM dual;. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 M | 1.000 | 281473913978936 | 09:55:34 | 09:58:29 | 02:55 | 163.75 |

The CUSTOMERS table has these columns:
CUSTOMER_ID NUMBER(4) NOT NULL
CUSTOMER_NAME VARCHAR2(100) NOT NULL
STREET_ADDRESS VARCHAR2(150)
CITY_ADDRESS VARCHAR2(50)
STATE_ADDRESS VARCHAR2(50)
PROVINCE_ADDRESS VARCHAR2(50)
COUNTRY_ADDRESS VARCHAR2(50)
POSTAL_CODE VARCHAR2(12)
CUSTOMER_PHONE VARCHAR2(20)
The CUSTOMER_ID column is the primary key for the table.
Which two statements find the number of customers? (Choose two.)

***explanation***
These statements provide correct syntax and semantics to show the number of customers.
Function COUNT() can be used with substitution symbol of all columns "*" or just with one column name. Last query will be processed a little bit faster.

| | | |
|---|---|---|
| + | 1 | SELECT COUNT(customer_id)<br>FROM customers; |
| + | 2 | SELECT COUNT(*)<br>FROM customers; |
| + | 3 | SELECT TOTAL(customer_id)<br>FROM customers; |
| + | 4 | SELECT TOTAL(*)<br>FROM customers; |
| + | 5 | SELECT TOTAL(customer_name)<br>FROM customers; |
| + | 6 | SELECT COUNT(customers)<br>FROM customers; |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15 M | 0.600 | 281473913978936 | 09:27:31 | 09:29:23 | 01:52 | 108.328 |

Which two statements about the evaluation of clauses in a SELECT statement are true? (Choose two.)

***explanation***
In SELECT statement WHERE clause comes after the FROM clause and before GROUP By clause while HAVING clause use to restrict the data group wise and it comes after the GROUP BY clause and ORDER BY clause comes after at end of SELECT statement.

| | | |
|---|---|---|
| + | 1 | The Oracle Server will evaluate an ORDER BY clause before a WHERE clause. |
| - | 2 | The Oracle Server will evaluate a WHERE clause before a GROUP BY clause. |
| + | 3 | The Oracle Server will evaluate an ORDER BY clause before a HAVING clause. |
| - | 4 | The Oracle Server will evaluate a GROUP BY clause before a HAVING clause. |
| + | 5 | The Oracle Server will evaluate a HAVING clause before a WHERE clause. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 16 M | 0.667 | 281473913978936 | 09:36:24 | 09:38:24 | 02:00 | 118.672 |

Which two tasks can you perform by using the TO_CHAR function? (Choose two)

***explanation***
TO_CHAR(x) function is used to convert the value x to a character or converts a date to a character string using formatting conventions.

| | | |
|---|---|---|
| + | 1 | Convert '10' to 10 |
| - | 2 | Convert a character expression to a date |
| + | 3 | Convert 'TEN' to 10 |
| - | 4 | Convert '10' to '10' |
| + | 5 | Convert 10 to 'TEN' |
| + | 6 | Convert a date to a character expression |

| | | | | | | |
|---|---|---|---|---|---|---|
| 17 S | 0.000 | 281473913978936 | 09:47:20 | 09:49:19 | 01:59 | 110.516 |

Which SQL statement generates the alias Annual Salary for the calculated column SALARY*12?

***explanation***
This SQL statement provides correct syntax to generate the alias Annual Salary for the calculated column SALARY*12.

| | | |
|---|---|---|
| | 1 | SELECT ename, salary*12 AS INITCAP("ANNUAL SALARY")<br>FROM employees |
| | 2 | SELECT ename, salary*12 'Annual Salary'<br>FROM employees; |
| - | 3 | SELECT ename, salary*12 AS Annual Salary |

| | | |
|---|---|---|
| | | FROM employees; |
| | 4 | SELECT ename, salary*12 “Annual Salary”<br>FROM employees; |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 18 S | 1.000 | 281473913978936 | 09:54:38 | 09:54:59 | 00:21 | 20.782 | |
| | Which clause would you use in a SELECT statement to limit the display to those employees whose salary is greater then 5000? | | | | | | |
| | 1 | GROUP BY SALARY > 5000 | | | | | |
| + | 2 | WHERE SALARY > 5000 | | | | | |
| | 3 | ORDER BY SALARY > 5000. | | | | | |
| | 4 | HAVING SALARY > 5000 | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 19 S | 1.000 | 281473913978936 | 09:29:59 | 09:30:26 | 00:27 | 22.813 | |
| | You need to calculate the total of all salaries in the accounting department. Which group function should you use? | | | | | | |
| | 1 | LARGEST | | | | | |
| | 2 | COUNT | | | | | |
| | 3 | MIN | | | | | |
| | 4 | MAX | | | | | |
| + | 5 | SUM | | | | | |
| | 6 | TOTAL | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 20 S | 0.000 | 281473913978936 | 09:24:50 | 09:25:11 | 00:21 | 18.359 | |
| | Which SELECT statement will the result ‘elloworld’ from the string ‘HelloWorld’? | | | | | | |
| | ***explanation***<br>TRIM function accept a string describing the data you would like to trim from a column value. It can trim from both side of column value i.e. left and right. In the following statement this function will trim as<br>SELECT LOWER(TRIM (‘H’ FROM ‘HelloWorld’)) FROM dual;<br>From the above statement trim function will remove the character ‘H’ from ‘HelloWorld’ and LOWER function will convert the remaining character to lower case. | | | | | | |
| | 1 | SELECT LOWER(TRIM (‘H’ FROM ‘HelloWorld’)) FROM dual; | | | | | |
| | 2 | SELECT LOWER(SUBSTR(‘HelloWorld’, 2, 1) FROM dual; | | | | | |
| | 3 | SELECT INITCAP(TRIM (‘HelloWorld’, 1,1)) FROM dual; | | | | | |
| | 4 | SELECT LOWER(SUBSTR(‘HellowWorld’, 1, 1) FROM dual; | | | | | |
| - | 5 | SELECT SUBSTR( ‘HelloWorld’,1) FROM dual; | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 21 S | 0.000 | 281473913978936 | 10:01:32 | 10:02:18 | 00:46 | 34.796 | |
| | Evaluate this SQL statement:<br>SELECT e.employee_id, (.15* e.salary) + (.5 * e.commission_pct)<br>+ (s.sales amount * (.35 * e.bonus)) AS CALC_VALUE<br>FROM employees e, sales s<br>WHERE e.employee_id = s.emp_id;<br><br>What will happen if you remove all the parentheses from the calculation? | | | | | | |
| | 1 | There will be no difference in the value displayed in the CALC_VALUE column. | | | | | |
| | 2 | The value displayed in the CALC_VALUE column will be higher. | | | | | |
| - | 3 | An error will be reported. | | | | | |
| | 4 | The value displayed in the CALC_VALUE column will be lower. | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 22 S | 0.000 | 281473913978936 | 09:44:21 | 09:46:31 | 02:10 | 122.204 | |
| | Which script displays '01-JAN-02' when the ENROLL_DATE value is '01-JUL-01'? | | | | | | |
| | 1 | SELECT ROUND(enroll_date, 'DAY')<br>FROM student; | | | | | |
| | 2 | SELECT ROUND(enroll_date, 'MONTH')<br>FROM student; | | | | | |
| | 3 | SELECT ROUND(enroll_date, 'YEAR')<br>FROM student; | | | | | |
| - | 4 | SELECT ROUND(TO_CHAR(enroll_date, 'YYYY'))<br>FROM student; | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 23 M | 0.200 | 281473913978936 | 09:33:50 | 09:35:14 | 01:24 | 74.578 | |
| | Which three functions can be used to manipulate character, number, or date column values? (Choose three.) | | | | | | |
| | ***explanation***<br>CONCAT , ROUND and INSTR are three single row function that can be used to manipulate character, number and data column values. INSTR finds numeric position of named character<br>CONCAT function it concatenate the first character value to the second value and ROUND fucnion will round the value to n decimal. | | | | | | |
| - | 1 | RPAD | | | | | |
| + | 2 | INSTR | | | | | |
| - | 3 | CONCAT | | | | | |
| - | 4 | ROUND | | | | | |
| - | 5 | TRUNC | | | | | |

| 24 S | 0.000 | 281473913978936 | 09:49:19 | 09:50:22 | 01:03 | 57.375 |
|---|---|---|---|---|---|---|

Which statement concerning SQL functions is true?

***explanation***

Single row character functions accept character data as input and can return both character and number values. Character functions are case conversion function and character manipulation functions

| | | |
|---|---|---|
| | 1 | All date functions return DATE data type values. |
| - | 2 | Single-row functions can only be used in SELECT and WHERE clauses. |
| | 3 | Conversion functions convert a column definition from one data type to another data type. |
| | 4 | Character functions can return character or number values. |

| 25 S | 1.000 | 281473913978936 | 09:40:50 | 09:41:54 | 01:04 | 57.844 |
|---|---|---|---|---|---|---|

Evaluate the SQL statement:
SELECT ROUND(45.953, -1), TRUNC(45.936, 2)
FROM dual;
Which values are displayed?

***explanation***

ROUND (45.953,-1) will round value to 1 decimal places to the left.
TRUNC (45.936,2) will truncate value to 2 decimal
The answer will be 50 and 45.93

| | | |
|---|---|---|
| | 1 | 46 and 45.93 |
| | 2 | 45 and 45.93 |
| | 3 | 46 and 45 |
| | 4 | 45.95 and 45.93 |
| + | 5 | 50 and 45.93 |
| | 6 | 50 and 45.9 |

| 26 M | 0.714 | 281473913978936 | 09:22:25 | 09:23:57 | 01:32 | 83.062 |
|---|---|---|---|---|---|---|

Which of the following functions are available in SQL? (Choose four)

| | | |
|---|---|---|
| + | 1 | DECODE. |
| + | 2 | NULLIF. |
| + | 3 | NGL. |
| + | 4 | TRAM. |
| - | 5 | CASCADE. |
| - | 6 | TRIM. |
| + | 7 | INSTR. |

| 27 S | 0.000 | 0 | 09:59:32 | --:--:-- | --:-- | 0 |
|---|---|---|---|---|---|---|

What is true of using group functions on columns that contain NULL values?

| | | |
|---|---|---|
| | 1 | Group functions on columns include NULL values in calculations if you use the keyword INC_NULLS. |
| | 2 | Group functions on columns ignore NULL values. |
| | 3 | Group functions on columns returning dates include NULL values. |
| | 4 | Group functions on columns returning numbers include NULL values. |
| | 5 | Group functions on columns cannot be accurately used on columns that contain NULL values. |

| 28 S | 0.000 | 281473913978936 | 09:35:14 | 09:36:24 | 01:10 | 65.672 |
|---|---|---|---|---|---|---|

Which SQL statement returns a numeric value?

***explanation***

DATE value subtract DATE value will return numeric value.

| | | |
|---|---|---|
| | 1 | SELECT TO_NUMBER(hire_date + 7)FROM EMP; |
| | 2 | SELECT sysdate-hire_date FROM EMP; |
| | 3 | SELECT ADD_MONTHS(MAX(hire_Date), 6) FROM EMP; |
| - | 4 | SELECT ROUND(hire_date)FROM EMP; |

| 29 S | 0.000 | 281473913978936 | 10:02:18 | 10:03:00 | 00:42 | 38.5 |
|---|---|---|---|---|---|---|

In which case would you use a FULL OUTER JOIN?

***explanation***

Oracle9i also makes it possible for you to easily execute a full outer join, including all records from the tables that would have been displayed if you had used both LEFT OUTER JOIN or RIGTH OUTER JOIN clauses.

| | | |
|---|---|---|
| - | 1 | You want all matched and unmatched data from only one table. |
| | 2 | You want all unmatched data from one table. |
| | 3 | One of the tables has more data than the other. |
| | 4 | Both tables have NULL values. |
| | 5 | You want all unmatched data from both tables. |
| | 6 | You want all matched data from both tables. |

| 30 M | 0.500 | 281473913978936 | 09:25:11 | 09:25:46 | 00:35 | 32.687 |
|---|---|---|---|---|---|---|

Which three are true regarding the use of outer joins? (Choose three.)

***explanation***

You can use an outerjoin to see only the rows that do not meet the join condition. In the WHERE

condition, you use (+) following the name of the column in the table without matching rows, to perform an outerjoin. You cannot link a condition that is involved in an outerjoin to another condition by using the OR operator.

| | | |
|---|---|---|
| - | 1 | You cannot link a condition that is involved in an outerjoin to another condition by using the OR operator. |
| + | 2 | You use (*) on both sides of the WHERE condition to perform an outerjoin. |
| + | 3 | You cannot use IN operator in a condition that involves an outerjoin. |
| - | 4 | You use an outerjoin to see only the rows that do not meet the join condition. |
| + | 5 | You use (+) on both sides of the WHERE condition to perform an outerjoin. |
| - | 6 | In the WHERE condition, you use (+) following the name of the column in the table without matching rows, to perform an outerjoin. |

| 31 S | 0.000 | 281473913978936 | 09:30:26 | 09:31:02 | 00:36 | 30.328 |
|---|---|---|---|---|---|---|

What is true about joining tables through an equijoin?

***explanation***

For N joined tables using Oracle or ANSI/ISO syntax for table joins, you need at least N-1 equijoin conditions in the WHERE clause of your SELECT statement or N-1 JOIN table_name ON join_condition clauses in order to avoid a Cartesian product, respectively.

| | | |
|---|---|---|
| | 1 | You can join a maximum of two tables through an equijoin. |
| | 2 | You can join a maximum of two columns through an equijoin. |
| | 3 | You specify an equijoin condition in the SELECT or FROM clauses of a SELECT statement. |
| - | 4 | To join two tables through an equijoin, the columns in the join condition must be primary key and foreign key columns. |
| | 5 | You can join n tables (all having single column primary keys) in a SQL statement by specifying a minimum of n-1 join conditions. |

| 32 M | 1.000 | 281473913978936 | 09:51:07 | 09:54:38 | 03:31 | 202.515 |
|---|---|---|---|---|---|---|

Which two operators can be used in an outer join condition? (Choose two.)

***explanation***

Outer Join can be used to also see rows that do not meet the join condition.
For an outer join you can use the equality (=) operator with AND operator in the join condition.

| | | |
|---|---|---|
| + | 1 | IN |
| + | 2 | = |
| + | 3 | AND |
| + | 4 | OR |

| 33 S | 1.000 | 281473913978936 | 09:41:54 | 09:44:20 | 02:26 | 134.75 |
|---|---|---|---|---|---|---|

Examine the structure of the EMPLOYEES and DEPARTMENTS tables:
EMPLOYEES
EMPLOYEE_ID NUMBER
DEPARTMENT_ID NUMBER
MANAGER_ID NUMBER
LAST_NAME VARCHAR2(25)
DEPARTMENTS
DEPARTMENT_ID NUMBER
MANAGER_ID NUMBER
DEPARTMENT_NAME VARCHAR2(35)
LOCATION_ID NUMBER
You want to create a report displaying employee last names, department names, and locations.
Which query should you use to create an equi-join?

***explanation***

Equijoins are also called simple joins or inner joins. Equijoin involve primary key and foreign key.

| | | |
|---|---|---|
| | 1 | SELECT last_name, department_name, location_id FROM employees , departments ; |
| + | 2 | SELECT e.last_name, d.department_name, d.location_id FROM employees e, departments D WHERE e.department_id =d.department_id; |
| | 3 | SELECT employees.last_name, departments.department_name, departments.location_id FROM employees e, departments D WHERE e.department_id =d.department_id; |
| | 4 | SELECT e.last_name, d.department_name, d.location_id FROM employees e, departments D WHERE manager_id =manager_id; |

| 34 M | 0.667 | 281473913978936 | 09:50:22 | 09:51:07 | 00:45 | 44.515 |
|---|---|---|---|---|---|---|

In which two cases would you use an outer join? (Choose two.)

***explanation***

You use an outer join to also see rows that do not meet the join condition.

| | | |
|---|---|---|
| - | 1 | The columns being joined have NULL values. |
| + | 2 | The tables being joined have only matched data. |
| + | 3 | The tables being joined have only unmatched data. |
| + | 4 | The tables being joined have both matched and unmatched data. |
| + | 5 | The tables being joined have NOT NULL columns. |
| - | 6 | Only when the tables have a primary key/foreign key relationship. |

**topics**

| points | correct | module | |
|---|---|---|---|
| | points | correct | topic |
| 16.277 / 34 ( 48%) | 16 / 34 ( 47%) | Oracle9i | |
| | 2 / 7 ( 29%) | 2 / 7 ( 29%) | Single-Row Functions & Group Functions 5-1 |
| | 3 / 9 ( 33%) | 3 / 9 ( 33%) | Single-Row Functions & Group Functions 4-1 |
| | 3.5 / 5 ( 70%) | 3 / 5 ( 60%) | Single-Row Functions & Group Functions 6-2 |
| | 0.667 / 1 ( 67%) | 1 / 1 (100%) | Single-Row Functions & Group Functions 6-3 |
| | 1.143 / 2 ( 57%) | 1 / 2 ( 50%) | Single-Row Functions & Group Functions 7-4 |
| | 0.6 / 1 ( 60%) | 1 / 1 (100%) | Single-Row Functions & Group Functions 5-2 |
| | 2 / 2 (100%) | 2 / 2 (100%) | Single-Row Functions & Group Functions 6-1 |
| | 0.2 / 1 ( 20%) | 0 / 1 ( 0%) | Single-Row Functions & Group Functions 5-3 |
| | 0 / 1 ( 0%) | 0 / 1 ( 0%) | Displaying Data from Multiple Tables 6-1 |
| | 0.5 / 1 ( 50%) | 0 / 1 ( 0%) | Displaying Data from Multiple Tables 6-3 |
| | 0 / 1 ( 0%) | 0 / 1 ( 0%) | Displaying Data from Multiple Tables 5-1 |
| | 1 / 1 (100%) | 1 / 1 (100%) | Displaying Data from Multiple Tables 4-2 |
| | 1 / 1 (100%) | 1 / 1 (100%) | Displaying Data from Multiple Tables 4-1 |
| | 0.667 / 1 ( 67%) | 1 / 1 (100%) | Displaying Data from Multiple Tables 6-2 |