Prepare Smart for Success Free Oracle 1Z0-071 Exam Questions and Answers

Ready to pass faster? Grab free and updated Oracle Database SQL exam PDF questions now. Get authentic 1Z0-071 dumps packed with verified answers and secure your certification success with PrepBolt 1Z0-071 exam pdf questions and answers.

Thank you for Downloading 1Z0-071 exam PDF Demo

https://prepbolt.com/1Z0-071.html

Question 1

Question Type: MultipleChoice

Examine the description of the CUSTOMERS table:

Which two SELECT statements will return these results:

CUSTOMER_ NAME

Mandy

Mary

Options:

A- SELECT customer_ name FROM customers WHERE customer_ name LIKE ' % a % ' ;
B- SELECT customer_ name FROM customers WHERE customer name LIKE 'Ma%' ;
C- SELECT customer_ name FROM customers WHERE customer_ name='*Ma*';
D- SELECT customer_ name FROM customers WHERE UPPER (customer_ name) LIKE 'MA*. ;
E- SELECT customer_ name FROM customers WHERE customer name LIKE 'Ma*';
F- SELECT customer_ name FROM customers WHERE UPPER (customer name) LIKE 'MA&';
G- SELECT customer_ name FROM customers WHERE customer_ name KIKE .*Ma*';

Answer:

B, D

Explanation:

The SQL LIKE operator is used in a WHERE clause to search for a specified pattern in a column. Here are the evaluations of the options:

A: The pattern ' % a % ' will match any customer names that contain the letter 'a' anywhere in the name, not necessarily those starting with 'Ma'.

B: This is correct as 'Ma%' will match any customer names that start with 'Ma'.

C: In SQL, the wildcard character is '%' not '*', therefore, the pattern 'Ma' is incorrect.

D: This is the correct syntax. 'UPPER (customer_ name) LIKE 'MA%'' will match customer names starting with 'Ma' in a case-insensitive manner.

E: Again, '*' is not a valid wildcard character in SQL.

F: The character '&' is not a wildcard in SQL.

G: The operator 'KIKE' is a typo, and '.Ma' is not valid SQL pattern syntax.

Question 2

Question Type: MultipleChoice

You must find the number of employees whose salary is lower than employee 110.

Which statement fails to do this?

Options:

A- SELECT COUNT (*)
FROM employees
JOIN employees a
ON e. salary< a. salary
WHERE a. employee_ id= 110;
B- SELECT COUNT (*)
FROM employees
WHERE salary < (SELECT salary FROM employees WHERE employee id =
110) ;
C- SELECT COUNT (*)
FROM employees e
JOIN (SELECT salary FROM employees WHERE employee_ id= 110) a
ON e. salary< a. salary;
D- SELECT COUNT (*)
FROM employees e
WHERE e. salary < (SELECT a. salary FROM employees a WHERE e. employee_ id = 110);

Answer:

D

Explanation:

In this context, the correct answers will provide a count of employees with a salary less than that of employee with ID 110. Let's evaluate the provided statements:

A: This statement is incorrect due to a syntax error; it uses alias 'e' without defining it, and 'a' is also used incorrectly in the JOIN clause. It should have been written with proper aliases.

B: This statement is correct as it uses a subquery to find the salary of employee ID 110 and compares it to the salaries of all employees to get the count.

C: This statement is also correct as it creates a derived table with the salary of employee ID 110 and compares it with the salaries of the 'employees' table using a JOIN.

D: This is the statement that fails. It is incorrect because the subquery references 'e.employee_id = 110', which will compare the salary of each employee to themselves if their ID is 110, not to the salary of employee 110 for all employees.

Question 3

Question Type: MultipleChoice

Examine this description of the PRODUCTS table:

You successfully execute this command:

CREATE TALE new_prices(prod_id NUBER(2),price NUMBER(8,2));

Which two statements execute without errors?

Options:

A- MERGE INTO new_prices n
USING(SELECT*FROM products)p
WHEN MATECHED THEN
UPDATE SET n.price=p.cost*.01
WHEN NOT MATCHED THEN
INSERT(n.prod_id,n.price)VALUES (p.prod_id,cost*01)
WHERE(p.cost<200);
B- MERGE INTO new_prices n
USING(SELECT*FROM product WHERE cost>150) p
ON (n.prod_id=p.prod_id)
WHEN NATCHED THEN
DELETE WHERE(p.cost<200)
WHEN NOT MATCHED THEN
INSERT (n.prod_id,n.price)VALUES (p.prod_id,p.cost*.01);
C- MERGE INTO new_prices n
USING (SELECT * FROM products WHERE cost>150) p
ON (n.prod_id=p.prod_id)
WHEN NATCHED THEN
UPDATE SET n.price=p.cost*.01
DELETE WHERE (p.cost<200);

D- MERGE INTO new_prices n
USING products p
WHEN NOT NATCHED THEN
INSERT (n.prod_id, n.price)VALUES (p.prod_id,cost*.01)
WHERE (p.cost <200);

Answer:

B, D

Explanation:

In the context of the MERGE statement, certain syntax rules and logical sequences must be observed:

Option B:

Correct use of the MERGE statement includes specifying the source and target correctly, and handling cases for matched and unmatched rows. This option properly structures the MERGE command with correct conditions for both matching (on prod_id) and actions (insert for unmatched, and delete under certain conditions for matched rows).

Option D:

While this option has a typographical error with 'WHEN NOT NATCHED', assuming it's intended as 'WHEN NOT MATCHED', it correctly describes the action for inserting into new_prices when no match is found in the target table based on the condition specified (product cost less than 200).

Options A and C contain syntax errors and logical contradictions (such as attempting to both update and delete in the same matched clause without proper separation or conditions), which are not allowed in a single WHEN MATCHED clause in Oracle SQL.

Question 4

Question Type: MultipleChoice

Which two statements will return the names of the three employees with the lowest salaries?

Options:

A- SELECT last_name, salary
FROM employees
WHERE ROWNUM<=3
B- SELECT last_name,salary

FROM employees
ORDER BY salary
FETCH FIRST 3 ROWS ONLY;
C- SELECT last_name,salary
FROM employees
WHERE ROWNUM<=3
ORDER BY (SELECT salary FROM employees);
D- SELECT last_name,salary
FROM (SELECT * FROM employees ORDER BY salary)
E- SELECT last_name,salary
FROM employees
FETCH FIRST 3 ROWS ONLY
ORDER BY salary;

Answer:

B

Explanation:

To retrieve the names of the three employees with the lowest salaries, the correct SQL syntax and logic are crucial:

Option B:

SELECT last_name, salary FROM employees ORDER BY salary FETCH FIRST 3 ROWS ONLY;

This query correctly sorts employees by their salary in ascending order and fetches the first three rows only. The FETCH FIRST n ROWS ONLY syntax is a standard way to limit the result set in SQL.

Options A, C, D, and E do not correctly implement the logic for fetching the lowest three salaries due to misuse of ROWNUM or incorrect placement of ORDER BY and FETCH clauses.

Question 5

Question Type: MultipleChoice

Which two are true about scalar subquery expressions?

Options:

A- You cannot correlate them with a table in the parent statement
B- You can use them as a default value for a column.

C- .You must enclose them in parentheses.
D- They can return at most one row.
E- They can return two columns.

Answer:

C, D

Explanation:

Scalar subquery expressions in Oracle SQL have specific rules:

Option C: You must enclose them in parentheses.

Scalar subqueries must be enclosed in parentheses. This is a requirement for syntax clarity and to distinguish the subquery from the rest of the SQL statement.

Option D: They can return at most one row.

Scalar subqueries are designed to return exactly one row containing one column. If a scalar subquery returns more than one row, Oracle will throw an error, ensuring that the subquery either returns a single value or no value (NULL).

Options A, B, and E are incorrect based on Oracle SQL functionalities:

Option A is incorrect because scalar subqueries can indeed be correlated with the parent query.

Option B is true but not in the context of default constraints for table columns in the CREATE TABLE statement.

Option E is incorrect because scalar subqueries can only return a single column by definition.

Question 6

Question Type: MultipleChoice

Examine the description of the CUSTOMERS table:

Which three statements will do an implicit conversion?

Options:

A- SELECT * FROM customers WHERE insert_date=DATE'2019-01-01';
B- SELECT * FROM customers WHERE customer_id='0001';
C- SELECT * FROM customers WHERE TO_DATE(insert_date)=DATE'2019-01-01';

D- SELECT * FROM customers WHERE insert_date'01-JAN-19';
E- SELECT * FROM customers WHERE customer_id=0001;
F- SELECT * FROM customers WHERE TO_CHAR(customer_id)='0001';

Answer:

B, D, F

Explanation:

A: No implicit conversion; DATE is explicitly specified.

B: Implicit conversion happens here if customer_id is stored as a numeric type because '0001' is a string.

C: TO_DATE is explicitly used, so no implicit conversion occurs here.

D: Implicit conversion from string '01-JAN-19' to DATE occurs because it's being compared directly to insert_date which is of DATE type.

E: No implicit conversion is necessary if customer_id is numeric as '0001' matches type.

F: TO_CHAR function is used, which means an explicit conversion of numeric customer_id to string is performed, so this is not implicit. Hence, this is incorrect regarding implicit conversion.

Each answer has been verified with reference to the official Oracle Database 12c SQL documentation, ensuring accuracy and alignment with Oracle's specified functionalities.

Question 7

Question Type: MultipleChoice

Examine this list of requirements for a sequence:

1. Name:EMP_SEQ

2. First value returned:1

3. Duplicates are never permitted.

4. Provide values to be inserted into the EMPLOYEES.EMPLOYEE_ID COLUMN.

5. Reduce the chances of gaps in the values.

Which two statements will satisfy these requirements?

Options:

A- CREATE SEQUENCE emp_seq START WITH 1 INCRENENT BY 1 NOCACHE;
B- CREATE SEQUENCE emp_seq START WITH 1 INCREMENT BY 1 CYCLE;
C- CREATE SEQUENCE emp_seq NOCACHE;
D- CREATE SEQUENCE emp_seq START WITH 1 CACHE;
E- CREATE SEQUENCE emp_seq START WITH 1 INCREMENT BY 1 CACHE;
F- CREATE SEQUENCE emp_seq;

Answer:

A, E

Explanation:

A: 'CREATE SEQUENCE emp_seq START WITH 1 INCREMENT BY 1 NOCACHE;' is correct for ensuring unique values without gaps as much as possible by not caching sequence numbers, which might otherwise be lost in a system crash.

B: CYCLE allows the sequence to restart when its max/min value is reached, which does not help with requirement 3 (duplicates are never permitted). Therefore, B is incorrect.

C: This lacks the necessary attributes like START WITH and INCREMENT BY which are crucial to defining a sequence. Thus, statement C is incorrect.

D: 'CREATE SEQUENCE emp_seq START WITH 1 CACHE;' might introduce gaps due to the caching of sequence numbers. This statement is somewhat contrary to requirement 5.

E: 'CREATE SEQUENCE emp_seq START WITH 1 INCREMENT BY 1 CACHE;' will provide continuous unique values, but may include gaps when the cache is lost due to a restart, yet it is more efficient and still generally aligns with the requirements. Hence, statement E is considered correct.

F: This lacks detail and is too ambiguous, lacking the necessary parameters. Therefore, F is incorrect.

Thank You for trying 1Z0-071 PDF Demo

To try our 1Z0-071 practice exam software visit link below

https://prepbolt.com/1Z0-071.html

Start Your 1Z0-071 Preparation

Use Coupon “SAVE50” for extra 50% discount on the purchase of Practice Test Software. Test your 1Z0-071 preparation with actual exam questions.