Oracle
1Z0-071
Oracle Database SQL Exam

For More Information – Visit link below:
https://www.examsempire.com/

Product Version

1. **Up to Date products, reliable and verified.**
2. **Questions and Answers in PDF Format.**

Latest Version: 21.1

Question: 1

Choose the best answer.
Examine the description of the EMPLOYEES table:

Name	Null	Type
EMP_ID	NOT NUL	NUMBER
EMP_NAME		VARCHAR2 (40)
DEPT_ID		NUMBER(2)
SALARY		NUMBER(8,2)
JOIN_DATE		DATE

Which query is valid?

A. SELECT dept_id, join_date,SUM(salary) FROM employees GROUP BY dept_id, join_date;
B. SELECT depe_id,join_date,SUM(salary) FROM employees GROUP BY dept_id:
C. SELECT dept_id,MAX(AVG(salary)) FROM employees GROUP BY dept_id;
D. SELECT dept_id,AVG(MAX(salary)) FROM employees GROUP BY dapt_id;

Answer: A

Explanation:
In Oracle 12c SQL, the GROUP BY clause is used to arrange identical data into groups with the GROUP BY expression followed by the SELECT statement. The SUM() function is then used to calculate the sum for each grouped record on a specific column, which in this case is the salary column.
Option A is valid because it correctly applies the GROUP BY clause. Both dept_id and join_date are included in the SELECT statement, which is a requirement when using these columns in conjunction with the GROUP BY clause. This means that the query will calculate the sum of salaries for each combination of dept_id and join_date. It adheres to the SQL rule that every item in the SELECT list must be either an aggregate function or appear in the GROUP BY clause.
Option B is invalid due to a typo in SELECT depe_id and also because it ends with a colon rather than a semicolon.
Option C is invalid because you cannot nest aggregate functions like MAX(AVG(salary)) without a subquery.
Option D is invalid for the same reason as option C, where it tries to nest aggregate functions AVG(MAX(salary)), which is not allowed directly in SQL without a subquery.
For further reference, you can consult the Oracle 12c documentation, which provides comprehensive guidelines on how to use the GROUP BY clause and aggregate functions like SUM():
Oracle Database SQL Language Reference, 12c Release 1 (12.1): GROUP BY Clause
Oracle Database SQL Language Reference, 12c Release 1 (12.1): Aggregate Functions

Question: 2

Which three are true about the CREATE TABLE command?

A. It can include the CREATE...INDEX statement for creating an index to enforce the primary key constraint.
B. The owner of the table should have space quota available on the tablespace where the table is defined.
C. It implicitly executes a commit.
D. It implicitly rolls back any pending transactions.
E. A user must have the CREATE ANY TABLE privilege to create tables.
F. The owner of the table must have the UNLIMITED TABLESPACE system privilege.

Answer: B, C, E

Explanation:
A . False - The CREATE TABLE command cannot include a CREATE INDEX statement within it. Indexes to enforce constraints like primary keys are generally created automatically when the constraint is defined, or they must be created separately using the CREATE INDEX command.
B . True - The owner of the table needs to have enough space quota on the tablespace where the table is going to be created, unless they have the UNLIMITED TABLESPACE privilege. This ensures that the database can allocate the necessary space for the table. Reference: Oracle Database SQL Language Reference, 12c Release 1 (12.1).
C . True - The CREATE TABLE command implicitly commits the current transaction before it executes. This behavior ensures that table creation does not interfere with transactional consistency. Reference: Oracle Database SQL Language Reference, 12c Release 1 (12.1).
D . False - It does not implicitly roll back any pending transactions; rather, it commits them.
E . True - A user must have the CREATE ANY TABLE privilege to create tables in any schema other than their own. To create tables in their own schema, they need the CREATE TABLE privilege. Reference: Oracle Database Security Guide, 12c Release 1 (12.1).
F . False - While the UNLIMITED TABLESPACE privilege allows storing data without quota restrictions on any tablespace, it is not a mandatory requirement for a table owner. Owners can create tables as long as they have sufficient quotas on the specific tablespaces.

Question: 3

The CUSTOMERS table has a CUST_CREDT_LIMIT column of data type number.
Which two queries execute successfully?

A. SELECT TO_CHAR(NVL(cust_credit_limit * .15,'Not Available')) FROM customers;
B. SELECT NVL2(cust_credit_limit * .15,'Not Available') FROM customers;
C. SELECT NVL(cust_credit_limit * .15,'Not Available') FROM customers;
D. SLECT NVL(TO_CHAR(cust_credit_limit * .15),'Not available') from customers;
E. SELECT NVL2(cust_credit_limit,TO_CHAR(cust_credit_limit * .15),'NOT Available') FROM customers;

Answer: A, E

Explanation:
A . True - The TO_CHAR function is used correctly here to convert the numeric value to a string, and NVL handles the case where cust_credit_limit might be NULL. The expression inside NVL computes 15% of the credit limit or displays 'Not Available' if the credit limit is NULL. The syntax is correct.
B . False - The NVL2 function requires three parameters: the expression to check for NULL, the value to return if it's not NULL, and the value to return if it is NULL. The given usage lacks the required parameters and syntax.
C . False - The NVL function expects both parameters to be of the same data type. Since the second parameter 'Not Available' is a string, it causes a data type conflict with the numerical result of the first parameter.
D . False - The keyword SELECT is misspelled as SLECT, making the syntax incorrect.
E. True - This query uses NVL2 correctly by checking if cust_credit_limit is not NULL, then applying TO_CHAR to compute 15% of it and converting it to string, or returning 'NOT Available' if it is NULL. The syntax and function usage are correct.

Question: 4

Choose two
Examine the description of the PRODUCT DETALS table:

NAME	NULL	TYPE
PRODUCT_ID	NOT NULL	NUMBER(2)
PRODUCT_NAME	NOT NULL	VARCHAR2(25)
PRODUCT_PRICE		NUMBER(8,2)
EXPIRY_DATE		DATE

A. PRODUCT_ID can be assigned the PEIMARY KEY constraint.
B. EXPIRY_DATE cannot be used in arithmetic expressions.
C. EXPIRY_DATE contains the SYSDATE by default if no date is assigned to it
D. PRODUCT_PRICE can be used in an arithmetic expression even if it has no value stored in it
E. PRODUCT_PRICE contains the value zero by default if no value is assigned to it.
F. PRODUCT_NAME cannot contain duplicate values.

Answer: A, D

Explanation:
A . PRODUCT_ID can be assigned the PRIMARY KEY constraint.
In Oracle Database 12c, a PRIMARY KEY constraint is a combination of a NOT NULL constraint and a unique constraint. It ensures that the data contained in a column, or a group of columns, is unique among all the rows in the table and not null. Given the PRODUCT_ID is marked as NOT NULL, it is a

candidate for being a primary key because we can assume that it is intended to uniquely identify each product in the table.
Reference: Oracle 12c documentation states that a column defined with the NOT NULL constraint can be used as a primary key, provided the values in the column are also unique (Oracle Database SQL Language
Reference, 12c Release 1 (12.1)).
B . EXPIRY_DATE cannot be used in arithmetic expressions. (Incorrect)
This statement is not necessarily true. Dates in Oracle can be used in arithmetic expressions, typically to add or subtract days from a date.
C . EXPIRY_DATE contains the SYSDATE by default if no date is assigned to it. (Incorrect)
Unless explicitly specified, a date column does not default to SYSDATE. A default value must be set using the DEFAULT clause during the table creation or altered later.
D . PRODUCT_PRICE can be used in an arithmetic expression even if it has no value stored in it.
This is correct. In Oracle, if a numeric column like PRODUCT_PRICE has a NULL value (meaning no value stored in it), it can still be used in an arithmetic expression. In such expressions, NULL is typically treated as a zero, but the result of any arithmetic with NULL is also NULL.
Reference: Oracle 12c SQL Language Reference indicates that if you include a numeric column with NULL in an arithmetic expression, the outcome will be NULL, meaning that the operation considers the NULL but does not necessarily treat it as zero (Oracle Database SQL Language Reference, 12c Release 1 (12.1)).
E . PRODUCT_PRICE contains the value zero by default if no value is assigned to it. (Incorrect)
Unless a default value is explicitly specified during the table creation or altered later, a numeric column like PRODUCT_PRICE does not automatically have a default value of zero.
F . PRODUCT_NAME cannot contain duplicate values. (Incorrect)
There is no constraint indicated that would prevent PRODUCT_NAME from containing duplicate values. Without a UNIQUE or PRIMARY KEY constraint, a column can contain duplicates.
The correct answers are A and D. PRODUCT_ID can be the primary key because it's specified as NOT NULL, thus it can uniquely identify each row in the table. PRODUCT_PRICE can be used in an arithmetic expression with the understanding that if it's NULL, the result of the expression would be NULL as well.

Question: 5

The CUSTOMERS table has a CUST_LAST_NAME column of data type VARCHAR2.
The table has two rows whose COST_LAST_MANE values are Anderson and Ausson.
Which query produces output for CUST_LAST_SAME containing Oder for the first row and Aus for the second?

A. SELECT REPLACE (REPLACE(cust_last_name,'son',''),'An','O') FROM customers;
B. SELECT REPLACE (TRIM(TRALING'son' FROM cust_last_name),'An','O') FROM customers;
C. SELECT INITCAP (REPLACE(TRIM('son' FROM cust_last_name),'An','O')) FROM customers;
D. SELECT REPLACE (SUBSTR(cust_last_name,-3),'An','O') FROM customers;

Answer: A

Explanation:
The REPLACE function in Oracle SQL is used to replace occurrences of a specified substring with another

substring. In this query, the inner REPLACE function call REPLACE(cust_last_name, 'son', '') removes the substring 'son' from cust_last_name. The outer REPLACE function call then replaces the substring 'An' with 'O'. For the given data, 'Anderson' would first be transformed to 'Ander' by the inner REPLACE, and then 'Ander' would be transformed to 'Oder' by the outer REPLACE. Similarly, 'Ausson' would first change to 'Aus' by the inner REPLACE, which is unaffected by the outer REPLACE.
Reference can be found in the Oracle Database SQL Language Reference documentation, which details the functionality of string functions, including REPLACE.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/