Item: 1 (Ref:Cert-1Z0-071.1.5.3)

The `teacher` table contains these columns:

```
ID NUMBER(9) Primary Key
LAST_NAME VARCHAR2(25)
FIRST_NAME VARCHAR2(25)
SUBJECT_ID NUMBER(9)
```

Which query should you use to display only the full name of each teacher along with the identification number of the subject each teacher is responsible for teaching?

- ○
```
SELECT *
FROM teacher;
```
- ○
```
SELECT last_name, subject_id
FROM teacher;
```
- ○
```
SELECT last_name, first_name, id
FROM teacher;
```
- ○
```
SELECT last_name, first_name, subject_id
FROM teacher;
```

Answer:

```
SELECT last_name, first_name, subject_id
FROM teacher;
```

Explanation:

You should use the following query to display only the full name of each teacher along with the identification number of the subject each teacher is responsible for teaching:

```
SELECT last_name, first_name, subject_id
FROM teacher;
```

To restrict the columns displayed to `last_name`, `first_name`, and `subject_id`, you should place only these columns in the `SELECT` list. You should place the columns you want to display in the order you want them displayed and separate them with commas.

You should not use the statement that uses an asterisk (*) in the `SELECT` list because this statement returns all of the columns in the `teacher` table. It returns all of the information requested, the full name as well as the subject identification, but it also includes the identification number of the teacher, which is not a part of the desired results.

You should not use the statement that includes only the `last_name` and `subject_id` columns in the `SELECT` list because this statement does not return the full name, as required in this scenario. The full name consists of the `last_name` and the `first_name` columns.

You should not use the statement that includes the `last_name`, `first_name`, and `id` columns in the `SELECT` list because this statement does not include the subject identification. Instead, it includes the teacher identification number, `id`, which is not a part of the desired results.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database Concepts12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 2 (Ref:Cert-1Z0-071.1.5.1)

The `account` table contains these columns:

```
ACCOUNT_ID NUMBER(12)
NEW_PURCHASES NUMBER(7,2)
PREV_BALANCE NUMBER(7,2)
FINANCE_CHARGE NUMBER(7,2)
PAYMENTS NUMBER(7,2)
```

You must print a report that contains the account number and the current balance for a particular customer. The current balance consists of the sum of an account's previous balance, new purchases, and finance charge. You must calculate the finance charge based on a rate of 1.5 percent of the previous balance. Payments must be deducted from this amount. The customer's account number is 543842.

Which `SELECT` statement should you use?

○
```
SELECT new_balance + finance_charge - payments

FROM account
WHERE account_id = 543842;
```

○
```
SELECT account_id, new_purchases + prev_balance * 1.015 - payments

FROM account
WHERE account_id = 543842;
```

○
```
SELECT account_id, new_purchases + (prev_balance * .015) - payments

FROM account
WHERE account_id = 543842;
```

○
```
SELECT account_id, new_purchases + (prev_balance * 1.015) + finance_charge -
payments
FROM account
WHERE account_id = 543842;
```

Answer:

```
SELECT account_id, new_purchases + prev_balance * 1.015 - payments
```

```
FROM account
WHERE account_id = 543842;
```

Explanation:

You should use the following SELECT statement:

```
SELECT account_id, new_purchases + prev_balance * 1.015 - payments
FROM account
WHERE account_id = 543842;
```

To calculate the new balance on an account, the finance charge is calculated by multiplying the previous balance by .015. To include the previous balance amount, 1.015 is used instead of .015 (`prev_balance * 1.015`). The result equals the previous balance plus the finance charge, which is .015 percent of the previous balance. After adding new purchases and subtracting the payments, the current balance calculation is complete.

Although parentheses indicate a higher precedence in arithmetic calculations in a SQL statement, they are not needed in this scenario. Because multiplication has precedence over addition and subtraction and the parentheses surround the multiplication calculation, the parentheses have no impact.

The choice that starts with `SELECT account_id, new_purchases + (prev_balance * 1.015) + finance_charge - payments` executes, but does not return the desired results. Instead of using the `finance_charge` column, the finance charge value must be calculated based on a rate of .015.

The choice that starts with `SELECT new_balance + finance_charge - payments` returns undesired results because the required current balance calculation is not used.

The choice that starts with SELECT clause `SELECT account_id, new_purchases + (prev_balance * .015) - payments` calculates the new balance improperly by neglecting to add the previous balance into the calculation.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Build a SELECT statement to retrieve data from an Oracle Database table

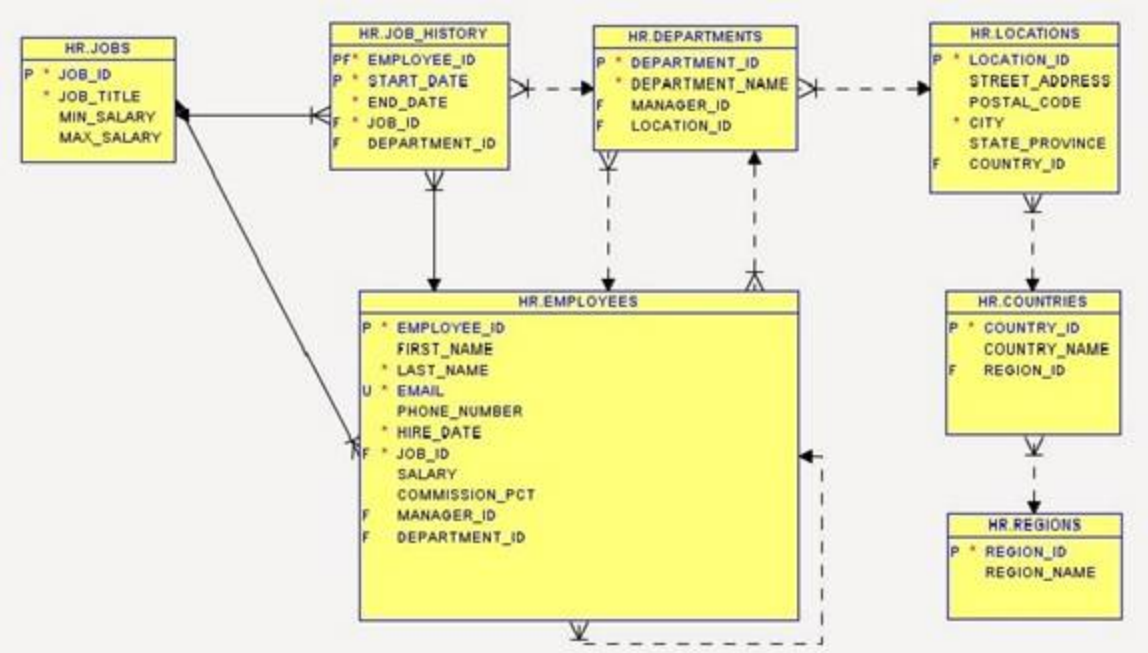
References:

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About SQL Conditions
http://docs.oracle.com/database/121/SQLRF/conditions001.htm#SQLRF52101

Item: 3 (Ref:Cert-1Z0-071.1.1.2)

You want to create an entity relationship diagram (ERD) for your database tables as follows:



Which tool should you use?

- ◯ Oracle Forms
- ◯ Oracle BI Publisher
- ◯ Oracle SQL Developer Data Modeler
- ◯ Oracle Reports

Answer:

Oracle SQL Developer Data Modeler

Explanation:

You can use the Oracle SQL Developer Data Modeler to create an ERD by importing the schema information from your existing database. You can use this tool to create logical, relational, physical, multi-dimensional, and data type models.

You do not use Oracle Forms to create an ERD. Oracle Forms is part of Oracle Fusion Middleware. Oracle Forms can be used to build data-entry forms. It cannot create an ERD.

You do not use Oracle BI Publisher to create an ERD. Oracle BI Publisher can create reports and dashboard layouts in a web browser, but does not support ERDs.

You should not use Oracle Reports. Oracle Reports is part of Oracle Fusion Middleware. It can be used to create declarative WYSIWYG reports

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Identify the connection between an ERD and a Relational Database

References:

Oracle SQL Developer Data Modeler User's Guide,
Release 4.1
E57984-04
1. Data Modeler Concepts and Usage
https://docs.oracle.com/cd/E57998_01/doc.41/e57984/data_modeling.htm#DMDUG25000

Oracle > TECHNOLOGY: SQL 101 > Modeling and Accessing Relational Data
http://www.oracle.com/technetwork/issue-archive/2014/14-may/o34sqldev-2193423.html

Item: 4 (Ref:Cert-1Z0-071.1.5.4)

Which `SELECT` statement should you use if you want to display unique combinations of the `position` and `manager` values from the `employee` table?

- ◯
```
SELECT position, manager DISTINCT
FROM employee;
```
- ◯
```
SELECT position, manager
FROM employee;
```
- ◯
```
SELECT DISTINCT position, manager
FROM employee;
```
- ◯
```
SELECT position, DISTINCT manager
FROM employee;
```

Answer:

```
SELECT DISTINCT position, manager
FROM employee;
```

Explanation:

You should use the following `SELECT` statement:

```
SELECT DISTINCT position, manager
FROM employee;
```

The `DISTINCT` keyword is used in a `SELECT` clause to return only the distinct (unique) values or combination of values for the columns(s) following the `DISTINCT` keyword. In this scenario, only unique combinations of `position` and `manager` will be returned by the query.

The `SELECT` statements that include the `DISTINCT` keyword after the column list and before only the `manager` column in the column list will fail when executed because the `DISTINCT` keyword does not precede both columns in the `SELECT` list.

The statement that does not include the `DISTINCT` keyword will execute successfully, but does not return the desired results. Because this statement does not contain the `DISTINCT` keyword, duplicate combinations of `position` and `manager` will be returned, if they exist.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 5 (Ref:Cert-1Z0-071.1.5.8)

The STUDENT table contains the following columns:

```
LAST_NAME VARCHAR2(25)
FIRST_NAME VARCHAR2(25)
EMAIL VARCHAR2(50)
```

You are writing a SELECT statement to retrieve the names of students that do NOT have an e-mail address.

```
SELECT last_name||', '||first_name "Student Name"
FROM student
```

Which WHERE clause should you use to complete this statement?

- ◯ `WHERE email = NULL;`
- ◯ `WHERE email != NULL;`
- ◯ `WHERE email IS NULL;`
- ◯ `WHERE email IS NOT NULL;`

Answer:

`WHERE email IS NULL;`

Explanation:
You should use the WHERE clause that uses the `IS NULL` comparison operator. When testing for null values in SQL, the `IS NULL` comparison operator should be used. This operator returns a Boolean value of TRUE when a null value is found, and returns FALSE when a value exists. The rows returned by this query will consist of students with no e-mail address.

You should not use the WHERE clause that uses the NOT operator with the `IS NULL` operator. When using the

NOT operator in combination with the IS NULL operator, a Boolean value of TRUE is returned when the condition is false. This statement will execute successfully, but will not return the desired results. The rows returned using this WHERE clause would be students with an e-mail address.

You should not use the WHERE clauses that use the = or != operators. An equality operator (=) or other comparison operator, such as !=, should not be used when testing for null values because a null value cannot be equal or unequal to another value. Comparisons between nulls and other values do not return a TRUE or FALSE value, but instead return a value of UNKNOWN.

If EMAIL contains:	Clause condition	Evaluates to:
value	email IS NULL	FALSE
value	email IS NOT NULL	TRUE
null	email IS NULL	TRUE
null	email IS NOT NULL	FALSE
value or null	email = NULL	UNKNOWN
value or null	email != NULL	UNKNOWN

Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About SQL Conditions
http://docs.oracle.com/database/121/SQLRF/conditions001.htm#SQLRF52101

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 6 (Ref:Cert-1Z0-071.1.2.2)

You are designing a database that your company will use to record its daily business activities. A fragment of the database is shown in the following image:

The database must comply with the following business rules:

- Each sales transaction must appear in exactly one invoice.
- Sales transactions are allowed only with registered customers.
- Only registered products can be sold.
- Each product must belong to a single registered category.

You are planning to enforce these rules by using foreign keys. Where should you define the foreign keys? (Select all that apply.)

- ☐ On the `Categories` table that references the `Inventory` table
- ☐ On the `Inventory` table that references the `Categories` table
- ☐ On the `Sales` table that references the `Inventory` table
- ☐ On the `Invoices` table that references the `Sales` table
- ☐ On the `Customers` table that references the `Invoices` table
- ☐ On the `Sales` table that references the `Invoices` table
- ☐ On the `Invoices` table that references the `Customers` table

Answer:

On the `Inventory` table that references the `Categories` table
On the `Sales` table that references the `Inventory` table
On the `Sales` table that references the `Invoices` table
On the `Invoices` table that references the `Customers` table

Explanation:

Each row in the `Sales` table represents a sales transaction and must reference an invoice that is registered in the `Invoices` table. Therefore, you should define a foreign key on the `InvoiceNumber` column in the `Sales` table to reference the `InvoiceNumber` column in the `Invoices` table.

Each row in the `Invoices` table represents an invoice that lists all sales transactions performed at one time with a specific customer. The customer must be registered in the `Customers` table, and, therefore, you should define a foreign key constraint on the `CustomerID` column in the `Invoices` table to reference the `CustomerID` column in the `Customers` table.

To enforce the requirement that only products that are registered in the `Inventory` table can be sold, you should define a foreign key on the `ProductID` column in the `Sales` table to reference the `ProductID` column in the `Inventory` table.

Each row in the `Inventory` table represents a product that must belong to a single category represented by a row in the `Categories` table. Therefore, you should define a foreign key on the `CategoryID` column in the `Inventory` table to reference the `CategoryID` column in the `Categories` table.

A foreign key is a set of one or more columns that references a matching set of columns in the same table or another table. Only those values that exist in the referenced column are allowed in a foreign key column. A foreign key is formally defined with a `FOREIGN KEY` constraint in the definition of the table where the foreign key column should exist. Foreign keys can reference only primary key columns or columns with a `UNIQUE` constraint. Foreign keys define relationships between tables or between different columns in the same table.

Foreign key constraints enforce referential integrity of data. Values in a referenced column cannot be changed and rows in the referenced table cannot be deleted if any of these actions would cause even one row in the foreign key table to become orphaned. A row in a foreign key table becomes orphaned if the value in the foreign key column in that row does not exist in the referenced column.

You should not define a foreign key on the `Categories` table that references the `Inventory` table, because `CategoryID` is primary key of the `Categories` table and referenced by the `Inventory` table. The foreign key should be defined on the `Inventory` table.

You should not define a foreign key the `Invoices` table that references the `Sales` table, because `InvoiceNumber` is primary key of the `Invoices` table and referenced by the `Sales` table. The foreign key should be defined on the `Sales` table.

You should not define a foreign key on the `Customers` table that references the `Invoices` table, because `CustomerID` is primary key of the `Customers` table and referenced by the `Invoices` table. The foreign key should be defined on the `Invoices` table.

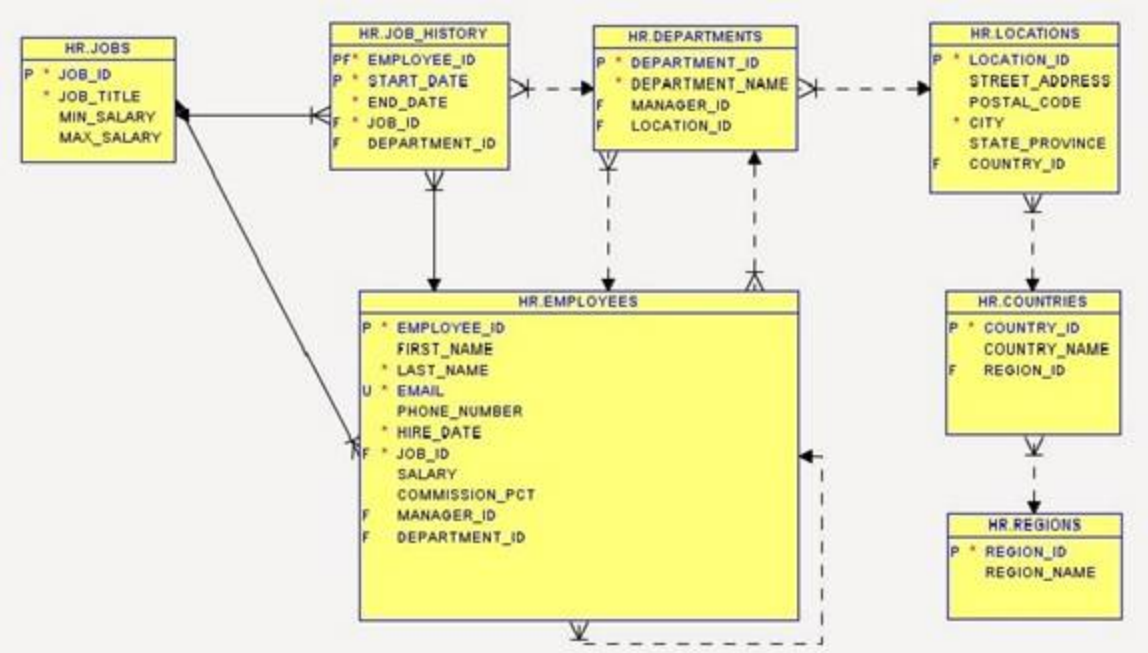
Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Explain the relationship between a database and SQL

References:
Oracle Database Concepts 12c Release 1 (12.1)
E41396-13
5. Data Integrity
http://docs.oracle.com/database/121/CNCPT/datainte.htm#CNCPT021

Item: 7 (Ref:Cert-1Z0-071.1.1.1)

You create an entity relationship diagram (ERD) based on an existing database:



Which of the following statements are true? (Choose all that apply.)

- [ ] Every region has at least one and at most one country.
- [ ] Every location can have one or many countries.
- [ ] Every job history can have one or many employees.
- [ ] Every manager can have one or many employees.
- [ ] Every department can have one or many employees.
- [ ] Every manager can have one or many departments.

Answer:

Every manager can have one or many employees.
Every department can have one or many employees.
Every manager can have one or many departments.

Explanation:

The following statements are correct:

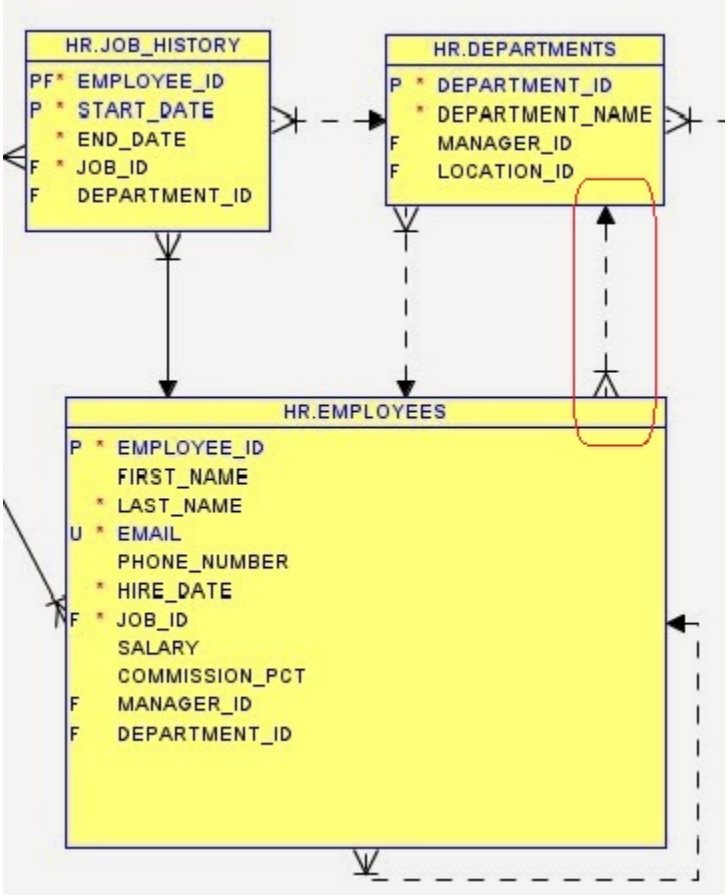
- Every manager can have one or many employees.
- Every department can have one or many employees.
- Every manager can have one or many departments.

The Oracle SQL Developer Data Modeler uses a crow's foot or fork to indicate the many side of a relationship. The arrowhead indicates the one side of the relationship.

According to the diagram, the `HR_EMPLOYEES` relation can have multiple employee entities for a single manager entity. This is indicated by the `MANAGER_ID` attribute. The crow's foot or fork is on the `HR_EMPLOYEES` relation, and the straight line with an arrow is pointing back to the `HR_EMPLOYEES` relation. Each employee entity has a

MANAGER_ID, making the referenced entity the manager of that employee.

The HR_DEPARTMENTS relation has a many-to-many relationship with the HR_EMPLOYEES relation, because there are two relationships between the tables, with the crow's foot or fork on both the HR_DEPARTMENTS and HR_EMPLOYEES relation. This means that a department can have one or many employees and a manager can have one or many departments.



It is NOT correct that every region has at least one and at most one country. According to the crow's feet on the HR_COUNTRIES relation, every region can have one or many countries.

Every location can NOT have one or many countries. Because the crow's feet is on the HR_LOCATIONS relation, every country can have one or more locations.

Every job history can NOT have one or many employees. Because the crow's feet is on the HR_JOB_HISTORY relation, every employee can have one or more job histories.

Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Identify the connection between an ERD and a Relational Database

References:
Oracle > TECHNOLOGY: SQL 101 > Modeling and Accessing Relational Data
http://www.oracle.com/technetwork/issue-archive/2014/14-may/o34sqldev-2193423.html

Oracle > TECHNOLOGY: SQL Developer > Document Entities

http://www.oracle.com/technetwork/issue-archive/2011/11-nov/o61sql-512018.html

Oracle SQL Developer Data Modeler User's Guide, Release 4.1
E57984-04
1. Data Modeler Concepts and Usage
1.3.5 Relational Models
https://docs.oracle.com/cd/E57998_01/doc.41/e57984/data_modeling.htm#DMDUG25000

Item: 8 (Ref:Cert-1Z0-071.1.5.7)

Examine the structure of the LINE_ITEM table shown in the exhibit:

LINE_ITEM

LINE_ITEM_ID	NUMBER(9)	NOT NULL, Primary Key
ORDER_ID	NUMBER(9)	NOT NULL, Primary Key, Foreign Key to ORDER_ID column of the CURR_ORDER table
PRODUCT_ID	NUMBER(9)	NOT NULL, Foreign Key to PRODUCT_ID column of the PRODUCT table
QUANTITY	NUMBER(9)	

You query the database with this SQL statement:

```
SELECT order_id||'-'||line_item_id||' '||product_id||' '||quantity "Purchase"
FROM line_item;
```

Which component of the SELECT statement is a literal?

- ○ '-'
- ○ ||
- ○ quantity
- ○ "Purchase"

Answer:
'-'

Explanation:

The literal component in this statement is '-'. A literal value is a value that is specified explicitly. When using the concatenation operator (||) in a SELECT list, any literal date, expression, number, or character value must be enclosed in single quotes. This statement contains three literal values ('-', ' ', and ' '), which is one hyphen and two individual spaces.

The concatenation operator (||) is not a literal. The concatenation operator is used to concatenate, or combine, data. By placing the concatenation operator between columns, expressions, spaces, or literal values, the items are combined and displayed as one concatenated value in the query results. The syntax of the concatenation operator is:

```
(column1|expression1)||(column2|expression2)[|| (column3|expression3)]...
```

"Purchase" is not a literal, but is a column alias for the concatenated string.

The quantity, order_id, line_item_id, and product_id components are columns, not literals.

Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Text Literals
http://docs.oracle.com/database/121/SQLRF/sql_elements003.htm#SQLRF00217

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 9 (Ref:Cert-1Z0-071.1.5.6)

Which statement, when executed, displays a zero if the prev_balance value is null and the new_balance value is zero?

- ◯
```
SELECT NVL(.009 * prev_balance, 0) + new_balance "Current Balance"
FROM account;
```
- ◯
```
SELECT NULL(.009 * prev_balance, 0) + new_balance "Current Balance"
FROM account;
```
- ◯
```
SELECT IS NULL(.009 * prev_balance, 0) + new_balance "Current Balance"
FROM account;
```
- ◯
```
SELECT TO_NUMBER(.009 * prev_balance) + new_balance "Current Balance"
FROM account;
```

Answer:

```
SELECT NVL(.009 * prev_balance, 0) + new_balance "Current Balance"
FROM account;
```

Explanation:

The following statement displays a zero if the prev_balance value is null and the new_balance value is zero:

```
SELECT NVL(.009 * prev_balance, 0) + new_balance "Current Balance"
FROM account;
```

If a column value in an arithmetic expression is null, the expression evaluates to null. If the value of the prev_balance column is null, the expression .009 * prev_balance returns a null value. When the null value is then used in an arithmetic expression, the value returned is UNKNOWN. When the value is UNKNOWN, no value is displayed.

The first expression, .009 * prev_balance, is evaluated first because multiplication takes precedence over addition. If the prev_balance value is null, it will be replaced with a 0, which is then used as part of an arithmetic calculation. The NVL function is used to replace a null value with a value. The first expression represents the column queried, and the second expression represents the string you want displayed if a null value is retrieved. The correct syntax of the NVL function is:

```
NVL(expression1, expression2)
```

The statement using NULL(.009 * prev_balance, 0) fails because NULL cannot be used by itself. It is not a function.

The statement using IS NULL(.009 * prev_balance, 0) fails because of the invalid use of the IS NULL comparison operator. The IS NULL operator should be used in the WHERE clause of a SQL statement to test for null values.

The statement containing TO_NUMBER(.009 * prev_balance, 0) fails because of the invalid use of the TO_NUMBER function. The TO_NUMBER function is used to convert a character data type value to a value of NUMBER data type or a specific format. This functionality is not needed in this example.

Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Nulls in SQL
http://docs.oracle.com/database/121/SQLRF/functions001.htm#SQLRF51173

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 10 (Ref:Cert-1Z0-071.1.3.1)

Which type of statement allows you to modify attributes of a schema object without affecting database users accessing that object?

- ◯ DML
- ◯ DDL
- ◯ System control
- ◯ Session control

Answer:

DDL

Explanation:

Data definition language (DDL) statements allow you to modify attributes of a schema object without affecting database users accessing that object. DDL statements create, define, and drop schema objects. A DDL statement starts with the keywords `ALTER`, `CREATE`, or `DROP`.

You should not use a data manipulation language (DML) statement. These types of statements modify or query existing schema objects. DML statements do not change the structure of the database like DDL statements do.

You should not use a session control statement. This type of statement controls the properties of a user's session. An example of a session control statement would be the `ALTER SESSION` statement or the `SET ROLE` statement.

You should not use a system control statement. This type of statement modifies the properties of the database instance. An example of a system control statement would be the `ALTER SYSTEM` statement.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Describe the purpose of DDL

References:

Oracle Database Concepts 12c Release 1 (12.1)
E41396-13
7. SQL
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 11 (Ref:Cert-1Z0-071.1.5.5)

The `account` table contains these columns:

```
ACCOUNT_ID NUMBER(12)
NEW_BALANCE NUMBER(7,2)
PREV_BALANCE NUMBER(7,2)
FINANCE_CHARGE NUMBER(7,2)
```

You must create statements to be mailed to all account holders. Each customer's statement must include the account holder's previous balance and finance charge in this format:

```
Previous Balance: 5000 Finance Charge: 45
```

Which SELECT statement will produce these results?

- ```
  SELECT Previous Balance: ||prev_balance|| Finance Charge: ||prev_balance * .009
  FROM account;
  ```
- ```
  SELECT 'Previous Balance:' ||prev_balance|| 'Finance Charge:' ||prev_balance
  * .009
  FROM account;
  ```
- ```
  SELECT 'Previous Balance: '||prev_balance||' Finance Charge: '||prev_balance
  * .009
  FROM account;
  ```
- ```
  SELECT "Previous Balance: "||prev_balance||" Finance Charge: "||prev_balance
  * .009
  FROM account;
  ```

Answer:

```
SELECT 'Previous Balance: '||prev_balance||' Finance Charge: '||prev_balance
* .009
FROM account;
```

Explanation:

The following SELECT statement will produce the desired results:

```
SELECT 'Previous Balance: '||prev_balance||' Finance Charge: '||prev_balance * .009
FROM account;
```

The concatenation operator (||) is used to concatenate, or combine, data. By placing the concatenation operator between columns, expressions, spaces, or literal values, the items are combined and displayed as one concatenated value in the query results. The syntax of the concatenation operator is:

```
(column1|expression1)||(column2|expression2)[||(column3|expression3)]...
```

Literal values must be enclosed in single quotes. To add space between a column value and text, place the desired number of blank spaces inside the single quotation marks, such as with `prev_balance||' Finance Charge '`. In this example a column is combined, or concatenated, to the text `'Finance Charge:'` with spaces separating them and a single space added to the end of the text. This returns the data in the desired format.

Because text cannot be combined with a column value using the concatenation operator unless single quotes surround the text, these SELECT statements will result in an error when executed:

```
SELECT Previous Balance ||prev_balance|| Finance Charge ||prev_balance * .009
FROM account;
```

```
SELECT "Previous Balance "||prev_balance||" Finance Charge "||prev_balance * .009
FROM account;
```

The following SELECT statement will not result in an error, but will not return the desired results because the spaces that are required between the text and column values displayed do not exist in the query results:

```
SELECT 'Previous Balance' ||prev_balance|| 'Finance Charge' ||prev_balance * .009
FROM account;
```

Objective:
Oracle and Structured Query Language (SQL)

Sub-Objective:
Build a SELECT statement to retrieve data from an Oracle Database table

References:
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries
http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

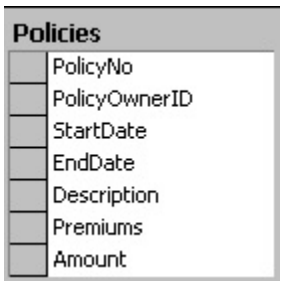
Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Text Literals
http://docs.oracle.com/database/121/SQLRF/sql_elements003.htm#SQLRF00217

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
CONCAT
http://docs.oracle.com/database/121/SQLRF/functions039.htm#SQLRF56641

Item: 12 (Ref:Cert-1Z0-071.1.2.1)

You are designing a database for an insurance company. Insurance policies will be registered in a table named `Policies` that is shown in the following image:



You must ensure that owners can have more than one policy, but each policy is uniquely identified and attributed to a single owner.

Which of the following should you do? (Choose all that apply.)

- ☐ Create a `FOREIGN KEY` constraint on the `PolicyOwnerID` column.

- [ ] Create a `FOREIGN KEY` constraint on the `PolicyNo` column.
- [ ] Create a `PRIMARY KEY` constraint on the `PolicyOwnerID` column.
- [ ] Create a `PRIMARY KEY` constraint on the `PolicyNo` column.

Answer:

Create a `FOREIGN KEY` constraint on the `PolicyOwnerID` column.
Create a `PRIMARY KEY` constraint on the `PolicyNo` column.

Explanation:

Each row in a table can be uniquely identified if the values in at least one of the columns are unique within the table. If no single column contains unique values, then a subset of columns can be chosen if each combination of values in those columns is unique. Such columns or subsets of columns are referred to as candidate keys.

A table can contain several candidate keys, one of which can be chosen as a primary key. To enforce uniqueness of the data in a primary key, you can define a `PRIMARY KEY` constraint on the primary key column or set of columns. In this scenario, each row in the `Policies` table represents an insurance policy. A natural choice for a primary key in the `Policies` table is the `PolicyNo` column.

A value in the `PolicyOwnerID` column identifies the policy owner. Because no other information about the policy owner is contained within the `Policies` table, information about all policy owners is probably stored in another table. In this scenario, the database should probably include a separate table (that may be named `Customers`) in which each row represents a customer. Each value in the `PolicyOwnerID` column in the `Policies` table should reference the row for a specific customer in the `Customers` table; this defines a relationship between the `Policies` and `Customers` tables.

Columns that define relationships between two tables are referred to as foreign keys. A foreign key relationship can be enforced by a `FOREIGN KEY` constraint defined on a foreign key column or set of columns. For a `FOREIGN KEY` constraint to be defined on a column or set of columns, the values in the referenced column or set of columns in the referenced table must be unique. The uniqueness of the values in the referenced column or set of columns can be enforced either by a `PRIMARY KEY` constraint or by a `UNIQUE` constraint. In this scenario, the `Customers` table should contain a primary key, for example, a `CustomerID` column. For each value in the `PolicyOwnerID` column, exactly one matching value must exist in the `CustomerID` column in the `Customers` table. The `FOREIGN KEY` constraint will enforce such a relationship.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Explain the relationship between a database and SQL

References:

Difference between Primary / Candidate Key
http://sql-plsql.blogspot.com/2012/09/difference-primary-candidate-key.html

Oracle Database Concepts 12c Release 1 (12.1)
E41396-13
5. Data Integrity
http://docs.oracle.com/database/121/CNCPT/datainte.htm#CNCPT021

Item: 13 (Ref:Cert-1Z0-071.1.4.1)

Which of the following correctly describes a Data Manipulation Language (DML) statement? (Choose all that apply.)

- ☐ The SELECT statement fetches data from one or more tables or views.
- ☐ The INSERT statement updates or adds rows conditionally into a table.
- ☐ The ALTER TABLE statement modifies the structure of a table.
- ☐ The MERGE statement deletes or inserts rows conditionally into a table or view.
- ☐ The DELETE statement removes rows from a table or views.

Answer:

The SELECT statement fetches data from one or more tables or views.

The DELETE statement removes rows from a table or views.

Explanation:

The following are true regarding Data Manipulation Language (DML) statements:

- The SELECT statement fetches data from one or more tables or views.
- The DELETE statement removes rows from a table or views

The ALTER TABLE modifies the structure of a table. However, the ALTER TABLE statement is a DDL statements that can be use to create, define, and drop schema objects. A DDL statement starts with the keywords ALTER, CREATE, or DROP. This type of statement modifies or queries existing schema objects. DML statements do not change the structure of the database like DDL statements do.

The INSERT statement does NOT update rows conditionally into a table. The MERGE statement does that. The INSERT statement adds rows conditionally into a table, but does not update them.

The MERGE statement does NOT delete rows conditionally into a table or view. The MERGE statement does update or insert rows conditionally into a table or view.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Describe the purpose of DML

References:

Oracle Database Concepts 12c Release 1 (12.1)
E41396-13
7. SQL
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015

Item: 14 (Ref:Cert-1Z0-071.1.5.2)

The account table contains these columns:

```
ACCOUNT_ID NUMBER(12)
NEW_BALANCE NUMBER(7,2)
PREV_BALANCE NUMBER(7,2)
FINANCE_CHARGE NUMBER(7,2)
```

With the least amount of effort, you want to display all of the rows in the account table.
Which query should you use?

- ○
```
SELECT *
FROM account;
```
- ○
```
SELECT all
FROM account;
```
- ○
```
SELECT any
FROM account;
```
- ○
```
SELECT account_id, new_balance, prev_balance, finance_charge
FROM account;
```

Answer:

```
SELECT *
FROM account;
```

Explanation:

You should use the following query to display all of the account table rows:

```
SELECT *
FROM account;
```

The asterisk (*) in the SELECT clause selects all columns from a table, view, materialized view, or snapshot. Using the asterisk (*) in the SELECT list simplifies the writing of such a query because the column names do not have to be individually included.

You should not use the query that includes each column name in the SELECT list. Although this query will achieve the desired results, it requires more effort than using an asterisk (*) in the SELECT list.

Both of the other options are incorrect because these statements fail. ALL is a keyword used in a FORUPDATE clause or in a WHERE clause comparison condition. When used in a SELECT clause, the statement fails. ANY is an operator used in a WHERE clause comparison condition. When used in a SELECT clause, the statement fails.

Objective:

Oracle and Structured Query Language (SQL)

Sub-Objective:

Build a SELECT statement to retrieve data from an Oracle Database table

References:

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
About Queries and Subqueries

http://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327

Oracle Database SQL Language Reference 12c Release 1 (12.1)
Part Number E41329-20
Creating Simple Queries
http://docs.oracle.com/database/121/SQLRF/queries002.htm#SQLRF52331

Oracle Database Concepts 12c Release 1 (12.1)
Part Number E41396-13
SELECT
http://docs.oracle.com/database/121/CNCPT/sqllangu.htm#CNCPT015