Item: 1 (Ref:Cert-1Z0-071.10.2.1)

Evaluate this CREATE TABLE statement:

```
CREATE TABLE customer (
customer_id NUMBER,
company_id VARCHAR2(30),
contact_name VARCHAR2(30),
contact_title VARCHAR2(20),
address VARCHAR2(30),
city VARCHAR2(25),
region VARCHAR2(10),
postal_code VARCHAR2(20),
country_id NUMBER DEFAULT 25,
phone VARCHAR2(20),
fax VARCHAR2(20),
credit_limit NUMBER (7,2));
```

Which three business requirements will this statement accomplish? (Choose three.)

- [ ] Credit limit values can be up to $1,000,000.
- [ ] Company identification values could be either numbers or characters, or a combination of both.
- [ ] All company identification values are only 6 digits so the column should be fixed in length.
- [ ] Phone number values can range from 0 to 20 characters so the column should be variable in length.
- [ ] The value 25 should be used if no value is provided for the country identification when a record is inserted.

Answer:

Company identification values could be either numbers or characters, or a combination of both.

Phone number values can range from 0 to 20 characters so the column should be variable in length.

The value 25 should be used if no value is provided for the country identification when a record is inserted.

Explanation:

The given CREATE TABLE statement will accomplish the following three business requirements:

- Company identification values could be either numbers or characters, or a combination of both.
- Phone number values can range from 0 to 20 characters so the column should be variable in length.
- The value 25 should be used if no value is provided for the country identification when a record is inserted.

Defining the company_id column as a VARCHAR2 data type allows for character data, including A-Z, a-z, and 0-9. Defining the PHONE column in the customer table as a VARCHAR2 data type provides for variable-length character data. In this example, phone number values can be up to 20 characters in length, but the size of the column will vary depending on each row value. Defining the COUNTRY_ID column with the DEFAULT option with a value of 25 ensures that the value of 25 will be used in INSERT operations if no value is provided. The DEFAULT option will prevent a null value from being inserted into the COUNTRY_ID column if a row is inserted without a COUNTRY_ID value.

Defining the CREDIT_LIMIT column of data type NUMBER(7,2) allows values up to 99,999.99. The column is defined with a precision of 7 and a scale of 2. If the user attempts to insert a credit limit value with more than 5

digits to the left of the decimal, an `ORA-01438: value larger than specified precision allows for this column` error would be returned.

Defining the `company_id` value of data type `VARCHAR2(30)` creates a variable-length character column of 30 bytes. Therefore, the business rule requiring a six-byte fixed-length column is not met.

Item: 2 (Ref:Cert-1Z0-071.10.1.2)

Which statements about data types are true? (Choose all that apply.)

- ☐ The `BLOB` data type stores character data up to four gigabytes.
- ☐ The `TIMESTAMP` data type is an extension of the `VARCHAR2` data type.
- ☐ The `CHAR` data type should be used for fixed-length character data.
- ☐ The `INTERVAL YEAR TO MONTH` data type allows time to be stored as an interval of years and months.
- ☐ The `VARCHAR2` data type requires that a minimum size be specified when defining a column of this type.

Answer:

The `CHAR` data type should be used for fixed-length character data.

The `INTERVAL YEAR TO MONTH` data type allows time to be stored as an interval of years and months.

Explanation:

The `CHAR` data type should be used for fixed-length character data. The length of the column is specified in bytes. The minimum size is 1, and the maximum size is 2000.

The `INTERVAL YEAR TO MONTH` data type stores a period of time using the `YEAR` and `MONTH` datetime fields. For example, `INTERVAL '315-5' YEAR(3) TO MONTH` indicates an interval of 315 years and 5 months.

The `TIMESTAMP` data type is an extension of the `DATE` data type and is used to store time as a date with fractional seconds.

The `BLOB` data type stores binary data up to four gigabytes, and the CLOB data type stores character data up to four gigabytes.

The `VARCHAR2` data type does not require that a minimum size be specified, but it does require that a maximum size be specified.

Item: 3 (Ref:Cert-1Z0-071.10.2.2)

Which `CREATE TABLE` statements will fail? (Choose all that apply.)

- ☐ `CREATE TABLE time1 (time1 NUMBER(9));`
- ☐ `CREATE TABLE date (time_id NUMBER(9));`

- [ ] `CREATE TABLE time (time_id NUMBER(9));`
- [ ] `CREATE TABLE time* (time_id NUMBER(9));`
- [ ] `CREATE TABLE $time (time_id NUMBER(9));`
- [ ] `CREATE TABLE datetime (time_id NUMBER(9));`

Answer:

```
CREATE TABLE date (time_id NUMBER(9));
CREATE TABLE time* (time_id NUMBER(9));
CREATE TABLE $time (time_id NUMBER(9));
```

Explanation:

The following `CREATE TABLE` statements will fail:

```
CREATE TABLE date (time_id NUMBER(9));
CREATE TABLE time* (time_id NUMBER(9));
CREATE TABLE $time (time_id NUMBER(9));
```

The `CREATE TABLE` statement for the `date` table will return an `ORA-00903: invalid table name` error because `DATE` is an Oracle Server reserved word. The `CREATE TABLE` statement for the `time*` table will return an `ORA-00922: missing or invalid option` error because database tables and column names cannot contain the `*` character. The `CREATE TABLE` statement for the `$time` table will return an `ORA-00911: invalid character` error because table and column names must begin with a letter.

The `CREATE TABLE` statement for the `time1` table will succeed. Table names must not duplicate the name of another object owned by the same Oracle Server user, but this rule does not apply to column names.

The `CREATE TABLE` statements for the `time` and `datetime` tables will succeed because neither `time` nor `datetime` are considered to be Oracle Server reserved words, and each table name follows the standard database object-naming conventions.

Item: 4 (Ref:Cert-1Z0-071.10.2.3)

Evaluate this `CREATE TABLE` statement.

```
SQL> CREATE TABLE order*1 (
order# NUMBER(9),
cust_id NUMBER(9),
date_1 DATE DEFAULT SYSDATE);
```

Which line of this statement will cause an error?

- ◯ Line 1
- ◯ Line 2
- ◯ Line 3
- ◯ Line 4

Answer:

Line 1

Explanation:

The first line of this `CREATE TABLE` statement will cause an `ORA-00903: invalid table name` error because `order*1` is not a legal table name. Table and column names cannot contain the * character. Table and column names can only contain the characters A-Z, a-z, 0-9, _ (underscore), $, and #. Oracle discourages the use of the $ and # characters.

The column names `order#`, `cust_id`, and `date_1` are all legal column names. Each begins with a letter, is 1-30 characters long, contains only legal characters, and is not an Oracle Server reserved word.

Item: 5 (Ref:Cert-1Z0-071.10.3.3)

Which `ALTER TABLE` statement should you use to add a `PRIMARY KEY` constraint on the `manufacturer_id` column of the `inventory` table?

- ◯
```
ALTER TABLE inventory
ADD PRIMARY KEY (manufacturer_id);
```
- ◯
```
ALTER TABLE inventory
ADD CONSTRAINT manufacturer_id PRIMARY KEY;
```
- ◯
```
ALTER TABLE inventory
MODIFY manufacturer_id CONSTRAINT PRIMARY KEY;
```
- ◯
```
ALTER TABLE inventory
MODIFY CONSTRAINT PRIMARY KEY manufacturer_id;
```

Answer:

```
ALTER TABLE inventory
ADD PRIMARY KEY (manufacturer_id);
```

Explanation:

You should use the following `ALTER TABLE` statement to add a `PRIMARY KEY` constraint on the `manufacturer_id` column of the `inventory` table:

```
SQL> ALTER TABLE inventory
ADD PRIMARY KEY (manufacturer_id);
```

To add a `PRIMARY KEY` constraint to an existing column in a table, use the `ALTER TABLE` statement with the `ADD` clause. In this scenario, the `ALTER TABLE` statement adds a `PRIMARY KEY` constraint to the `manufacturer_id` column in the `inventory` table. Because a name is not provided for the `PRIMARY KEY` constraint, a system-generated name will be used. To specify a constraint name, you must use the CONSTRAINT keyword in the ADD clause. For this statement to execute, the table must be empty or all the values in the `manufacturer_id` column must be unique and non-null.

The `ALTER TABLE` statements using the `MODIFY` clause are incorrect because you cannot add a `PRIMARY KEY` constraint using this clause. These statements will return a syntax error.

The ALTER TABLE statement containing the clause ADD CONSTRAINT manufacturer_id PRIMARY KEY will return a syntax error because the column name for the PRIMARY KEY constraint is not specified.

Item: 6 (Ref:Cert-1Z0-071.10.1.5)

Which data type is a hexadecimal string representing the unique address of a row in its table?

- ◯ RAW
- ◯ ROWID
- ◯ BFILE
- ◯ VARCHAR2

Answer:

ROWID

Explanation:

The ROWID data type is a hexadecimal string that represents the unique address of a row in its table. ROWID values can be queried just like other values in a table. Because ROWID provides a unique identifier for each row in the table, it can be used to locate a row in a table.

The RAW data type allows for the storage of binary data of a specified size. When data is stored as type RAW or LONG RAW, the Oracle Server does not perform character set conversion when the data is transmitted across machines in a network or when data is moved from one database to another.

The BFILE data type stores unstructured data in operating system files.

The VARCHAR2 data type stores variable-length character data and uses only the number of bytes needed to store the actual column value.

Item: 7 (Ref:Cert-1Z0-071.10.3.2)

Examine the structure of the employee table:

EMPLOYEE

EMPLOYEE_ID	NUMBER	NOT NULL, Primary Key
EMP_LNAME	VARCHAR2(25)	
EMP_FNAME	VARCHAR2(25)	
DEPT_ID	NUMBER	Foreign key to DEPT_ID column of the DEPARTMENT table
JOB_ID	NUMBER	Foreign key to JOB_ID column of the JOB table
MGR_ID	NUMBER	References EMPLOYEE_ID column
SALARY	NUMBER(9,2)	
HIRE_DATE	DATE	

Which CREATE TABLE statement should you use to create the employee table?

○
```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id));
```

○
```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id),
CONSTRAINT mgr_id_fk FOREIGN KEY(mgr_id) REFERENCES employee(employee_id));
```

○
```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id),
CONSTRAINT dept_id_fk FOREIGN KEY(dept_id) REFERENCES department(dept_id),
CONSTRAINT job_id_fk FOREIGN KEY(job_id) REFERENCES job(job_id));
```

○
```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id),
CONSTRAINT dept_id_fk FOREIGN KEY(dept_id) REFERENCES department(dept_id),
CONSTRAINT job_id_fk FOREIGN KEY(job_id) REFERENCES job(job_id),
CONSTRAINT mgr_id_fk FOREIGN KEY(mgr_id) REFERENCES employee(employee_id));
```

Answer:

```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
```

```
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id),
CONSTRAINT dept_id_fk FOREIGN KEY(dept_id) REFERENCES department(dept_id),
CONSTRAINT job_id_fk FOREIGN KEY(job_id) REFERENCES job(job_id),
CONSTRAINT mgr_id_fk FOREIGN KEY(mgr_id) REFERENCES employee(employee_id));
```

Explanation:

You should use the following `CREATE TABLE` statement to create the `employee` table:

```
CREATE TABLE employee (
employee_id NUMBER,
emp_lname VARCHAR2(25),
emp_fname VARCHAR2(25),
dept_id NUMBER,
job_id NUMBER,
mgr_id NUMBER,
salary NUMBER(9,2),
hire_date DATE,
CONSTRAINT employee_id_pk PRIMARY KEY(employee_id),
CONSTRAINT dept_id_fk FOREIGN KEY(dept_id) REFERENCES department(dept_id),
CONSTRAINT job_id_fk FOREIGN KEY(job_id) REFERENCES job(job_id),
CONSTRAINT mgr_id_fk FOREIGN KEY(mgr_id) REFERENCES employee(employee_id));
```

The statement required to create the `employee` table contains four `CONSTRAINT` clauses. A `PRIMARY KEY` constraint and three `FOREIGN KEY` constraints are required. The clause `CONSTRAINT employee_id_pk PRIMARY KEY(employee_id)` creates a `PRIMARY KEY` constraint on the `employee_id` column. This `PRIMARY KEY` constraint enforces uniqueness of the values in the `employee_id` column. You do not need to specify a `NOT NULL` constraint on this column because a `PRIMARY KEY` constraint ensures that no part of the primary key can contain a null value.

The following clause creates a `FOREIGN KEY` constraint on the `dept_id` column in the `employee` table that references the `dept_id` column in the `department` table:

```
CONSTRAINT dept_id_fk FOREIGN KEY(dept_id) REFERENCES department(dept_id)
```

The following clause creates a `FOREIGN KEY` constraint on the `job_id` column in the `employee` table that references the `job_id` column in the `department` table:

```
CONSTRAINT job_id_fk FOREIGN KEY(job_id) REFERENCES job(job_id)
```

The following clause creates a `FOREIGN KEY` constraint on the `mgr_id` column in the `employee` table that references the `employee_id` column in the `employee` table:

```
CONSTRAINT mgr_id_fk FOREIGN KEY(mgr_id) REFERENCES employee(employee_id)
```

These `FOREIGN KEY` constraints ensure that all values in the `dept_id`, `job_id`, and `mgr_id` columns in the `employee` table match an existing value in the parent table or have a null value.

All of the other options are incorrect because they do not satisfy the `employee` table requirements. These statements do not create all of the necessary constraints.

Item: 8 (Ref:Cert-1Z0-071.10.1.4)

Evaluate this CREATE TABLE statement:

```
CREATE TABLE curr_order (
id NUMBER,
customer_id NUMBER,
emp_id NUMBER,
order_dt TIMESTAMP WITH LOCAL TIME ZONE,
order_amt NUMBER(7,2),
ship_method VARCHAR2(5));
```

Which statement about the order_dt column is true?

- ◯ Data will be normalized to the database time zone.
- ◯ Data will include a time zone displacement in its value.
- ◯ Data will be stored using a fractional seconds precision of 3.
- ◯ Data stored in the column will be returned in the server's local time zone.

Answer:

Data will be normalized to the database time zone.

Explanation:

The TIMESTAMP WITH LOCAL TIME ZONE data type includes a time zone displacement value. Time zone displacement is the difference between local time and Coordinated Universal Time (UTC). When the TIMESTAMP WITH LOCAL TIME ZONE data type is used and data is stored in the database, the data is normalized to the database time zone.

The option that states data will include a time zone displacement in its value is incorrect. With the TIMESTAMP WITH LOCAL TIME ZONE data type, time zone displacement is not stored as part of the column data.

The option that states data will be stored using a fractional seconds precision of 3 is incorrect. The precision of the TIMESTAMP data type is not specified in the CREATE TABLE statement, so the precision of fractional seconds defaults to 6.

The option that states data stored in the column will be returned in the server's local time zone is incorrect. When the data is queried, the Oracle Server returns the data in the user's local session time zone.

Item: 9 (Ref:Cert-1Z0-071.10.4.1)

Your database contains the Customer and Order tables. The following graphic shows the fields of both tables:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	Sex
1	Aaron Anderons	Jeff Lebowski	2507 Henry St	Santa Monica	90401	USA	M
2	Amy Lawerence	Walter Sobchak	1199 Paddocks Way	Santa Monica	90401	USA	F
3	Anderson East	Jackie Treehorn	11 Main Street	El Segundo	90245	USA	M

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	20	7	2016-07-05	3
10309	370	3	2016-07-05	1
10310	770	8	2016-07-06	2

The database administrator has determined that the `Sex` column in the `Customer` table is not relevant and has not been used on any reports. Although this information is not needed today, it may be used in future with other Business Intelligence tools. You need to ensure the following:

- Applications that query the table do not display the sex of the customer in a report.
- The column is still physically stored in the table.

What should you run?

- ◯
```
ALTER TABLE Customers
DROP COLUMN Sex
```
- ◯
```
ALTER TABLE Customers
deallocate_unused_clause Sex
```
- ◯
```
ALTER TABLE Customers SET UNUSED Sex;
```
- ◯
```
ALTER TABLE Customers SET UNUSED Sex;
ALTER TABLE Customers DROP UNUSED COLUMNS;
```

Answer:

```
ALTER TABLE Customers SET UNUSED Sex;
```

Explanation:

You should run the following:

```
ALTER TABLE Customers SET UNUSED Sex;
```

The `SET UNUSED` clause with the `ALTER STATEMENT` table logically deletes a column from a table, but does not actually physically delete it from the table. The column will no longer be visible to a user or an application that queries the table. You may want to logically delete a column instead of physically deleting the with the `DROP COLUMN` clause. With large tables, it can take a lot of time and resources to physically delete a column. You could use the `SET UNUSED` clause to logically delete the column, and

Item: 10 (Ref:Cert-1Z0-071.10.1.3)

You need to create the `ELEMENT` table. The atomic weights of elements have varying decimal places. For example, values could be 4, 4.35, or 4.3567.

Which data type would be most appropriate for the atomic weight values?

- ◯ `RAW`
- ◯ `LONG`

- ◯ NUMBER
- ◯ NUMBER(p,s)

Answer:

NUMBER

Explanation:

The NUMBER data type without precision or scale would be most appropriate. When a fixed-point number is not known, as in this example, floating-point numbers should be used. Floating-point numbers are represented using the NUMBER data type.

All of the other options are incorrect. The NUMBER data type with precision and scale (p,s) stores fixed-point numbers. The LONG data type stores character data of variable length up to 2 gigabytes. The RAW data type stores raw binary data of a specified size.

Item: 11 (Ref:Cert-1Z0-071.10.5.2)

An external table named contract_employees provides data for the employees table in a database. The following statement is executed for the external table:

```
DROP TABLE contract_employees;
```

Which of the following statements is TRUE about the DROP TABLE statement?

- ◯ The DROP TABLE statement removes only the external table from the database.
- ◯ The DROP TABLE statement removes only the metadata of the external table from the database.
- ◯ The DROP TABLE statement removes the external table and its metadata from the database.
- ◯ The DROP TABLE statement removes neither the external table nor its metadata from the database.

Answer:

The DROP TABLE statement removes only the metadata of the external table from the database.

Explanation:

In this scenario, the DROP TABLE statement removes the metadata of the external table from the database. The DROP TABLE statement does not remove the external data in the external source because the table is an external table, and so does not exist in the database. Only the table metadata is stored in the database. Therefore, using the DROP TABLE statement does not affect the external data.

All the other options are incorrect because the DROP TABLE statement removes the external table metadata that is stored in the database. The actual data remains in a flat file stored in the operating system of the computer on which Oracle is installed.

Item: 12 (Ref:Cert-1Z0-071.10.1.1)

Which two statements about a column are true? (Choose two.)

- ☐ You cannot decrease the width of a CHAR column.
- ☐ You cannot increase the width of a VARCHAR2 column.
- ☐ You cannot convert a CHAR data type column to the VARCHAR2 data type.
- ☐ You cannot specify the column's position when adding a new column to a table.
- ☐ You cannot modify the data type of a column if the column contains non-null data.

Answer:

You cannot specify the column's position when adding a new column to a table.
You cannot modify the data type of a column if the column contains non-null data.

Explanation:

The following two statements about a column are true:

- You cannot specify the column's position when adding a new column to a table.
- You cannot modify the data type of a column if the column contains non-null data.

You can add or modify a column using the ALTER TABLE statement. When you add a column, you cannot specify the column's position. The new column automatically becomes the last column. You can modify a column's data type, size, or default value, but you can only change a column's data type if the column contains only null values or if the table is empty.

All of the other options are incorrect. You can decrease the width of a column if the column contains only null values or if the table is empty. You can increase the width of a column. You can convert a CHAR data type column to the VARCHAR2 data type if the column contains null values or if you do not change the column size.

Item: 13 (Ref:Cert-1Z0-071.10.5.1)

John works as a DBA for the railways. Data for all trains and passengers is centrally managed in the railway_records database. This database has train_info and passenger_info tables that contain data about various local trains and passengers for all the stations. The RailwayRecords database is unavailable for a few days after a server crash. Officers at the local stations store data in a Notepad file when the database in unavailable. When the server is functional again, John needs to collate the data from all the Notepad files and populate the railway_records database. He executes the following statements:

```
USE DATABASE railway_records;
CREATE TABLE missing_railway_info
(
train_no NUMBER(10),
origin_point VARCHAR2,
```

```
destination_point VARCHAR2
)
ORGANIZATION EXTERNAL
(
TYPE ORACLE_LOADER
ACCESS PARAMETER
(
RECORDS DELIMITED BY NEWLINE
LOGFILE 'train.log'
FIELDS TERMINATED BY ','
MISSING FIELD VALUES ARE NULL
( train_no,
origin_point,
destination_point )
)
LOCATION ('station1.txt', 'station2.txt', 'station3.txt')
PARALLEL 3;
```

Which of the following are TRUE about the given statements? (Choose all that apply.)

- ☐ Data can be loaded from the external table into the `railway_records database`.
- ☐ Data in the `missing_railway_info` table can be retrieved in parallel.
- ☐ Data is loaded from the `station1.txt` file only.
- ☐ Data is loaded from the `station1.txt` and the `station2.txt` files only.

Answer:

Data can be loaded from the external table into the `railway_records database`.

Data in the `missing_railway_info` table can be retrieved in parallel.

Explanation:

The correct answer is that data can be loaded from the external table into the `railway_records` database and data in the `missing_railway_info` table can be retrieved in parallel.

By default, the access driver of an external table is `ORACLE_LOADER`, which implies that data can be loaded from the external table into an existing table in the database. If you want to write data to the external table, you need to specify the access driver as `ORACLE_DATAPUMP`. In addition, the `PARALLEL` clause in the `CREATE TABLE` statement allows data in the external table to be retrieved in parallel.

Data is not loaded from the `station1.txt` file only, or from the `station1.txt` and the `station2.txt` files only. The `LOCATION` clause is used to specify the external source or files from where the data is actually loaded into the external table. In this scenario, the `LOCATION` clause specifies three files: `station1.txt`, `station2.txt`, and `station3.txt`. This means that data from all three files are extracted and loaded into the external table.

Item: 14 (Ref:Cert-1Z0-071.10.3.1)

You disabled the `PRIMARY KEY` constraint on the `id` column in the `inventory` table and updated all the values

in the `inventory` table. You need to enable the constraint and verify that the new `id` column values do not violate the constraint. If any of the `id` column values do not conform to the constraint, an error message should be returned.

Evaluate this statement:

```
ALTER TABLE inventory
ENABLE CONSTRAINT inventory_id_pk;
```

Which statement is true?

- ◯ The statement will achieve the desired results.
- ◯ The statement will execute, but will not enable the `PRIMARY KEY` constraint.
- ◯ The statement will execute, but will not verify that values in the ID column do not violate the constraint.
- ◯ The statement will return a syntax error.

Answer:

The statement will achieve the desired results.

Explanation:

The statement will achieve the desired results. In this scenario, the `ALTER TABLE ENABLE CONSTRAINT` statement will activate the currently disabled `PRIMARY KEY` constraint on the `inventory` table. When a `PRIMARY KEY` constraint is disabled, the corresponding unique indexes for the constraint are automatically dropped. When the constraint is enabled, the constraint will apply to all the data in the table. If existing data in the table does not adhere to the constraint, the `ALTER TABLE ENABLE CONSTRAINT` statement will fail and generate an error message. When you enable the `PRIMARY KEY` constraint, a primary key index is automatically created.

The remaining options are incorrect because the statement will execute and the desired results are achieved.