Item: 1 (Ref:Cert-1Z0-071.14.2.2)

You are working as a DBA for a national electricity board. Due to shortage of electricity and heavy consumption, the board decides to levy an additional of 25% on the billing amount of those customers whose billing amount is more than $2,000. The records of all the customers are stored in the `billed_customers` table, while the records of the customers who will have to pay the extra 25% are stored in the `surcharged_customers` table.

The `billed_customers` table is as follows:

cust_id	cust_name	bill_amount
C001	Anthony Miller	1500
C002	Raymond Jones	750
C003	Grace Hall	3000
C004	Jack Robinson	2250
C005	Paula Lewis	900

The `surcharged_customers` table is as follows:

cust_id	cust_name	bill_amount	surcharged_amount
C002	Raymond Jones	750	0
C004	Jack Robinson	2250	1000

You execute the following statement:

```
MERGE INTO surcharged_customers sc
USING billed_customers bc
ON (sc.cust_id = bc.cust_id)
WHEN NOT MATCHED THEN
INSERT (sc.cust_id, sc.cust_name, sc.bil_amount, sc.surcharged_amount)
VALUES (bc.cust_id, bc.cust_name, bc.bill_amount, bc.bill_amount*1.25)
WHERE (bc.bill_amount>2000);
```

How many rows are inserted in the `surcharged_customers` table?

- ◯ One row
- ◯ Two rows
- ◯ Three rows
- ◯ Five rows

Answer:

One row

Explanation:

In this scenario, only one row is inserted in the `surcharged_customers` table as a result of the `MERGE` statement. The `surcharged_customers` table already has two rows with `C002` and `C004` as values of `cust_id`. For these two rows, the `MERGE` statement checks whether the value of `cust_id` matches with the value of `cust_id` of any row in the `billed_customers` table. There are two rows with the same `cust_id` values, which implies that the join condition (`sc.cust_id = bc.cust_id`) is satisfied. However, since there is no `WHEN MATCHED THEN` clause in the `MERGE` statement, nothing happens. For those rows in the target table that do not satisfy the join condition, new rows are inserted in the `surcharged_customers` table only when the

bill_amount in the billed_customers table is more than $2,000.

The option stating that two rows are inserted in the surcharged_customers table is incorrect because only one row in the billed_customers table satisfies the greater than $2,000 condition.

The option stating that three rows are inserted in the surcharged_customers table is incorrect. This option would be correct if there was no WHERE condition for the INSERT clause of the MERGE statement.

The option stating that five rows are inserted in the surcharged_customers table is incorrect. This option would be correct if there were no rows in the surcharged_customers table, in which case, the join condition would be false for all the rows in the billed_customers table.

Item: 2 (Ref:Cert-1Z0-071.14.2.1)

Which of the following two statements are TRUE about the MERGE statement? (Choose two.)

- ☐ Rows in the target table can be deleted using the MERGE statement.
- ☐ Rows in the target table are updated if the join condition is not met.
- ☐ Rows are either inserted or updated in the target table during a single pass of the source table.
- ☐ Rows can be deleted even if they do not match the join condition for merging.

Answer:

Rows in the target table can be deleted using the MERGE statement.

Rows are either inserted or updated in the target table during a single pass of the source table.

Explanation:

The options stating that rows in the target table can be deleted through the MERGE statement and the option that rows can be simultaneously updated and inserted in the target table are both true. You can use the MERGE statement to delete rows that are updated through the MERGE statement. This is helpful for situations where you want to remove rows with inconsistent data after they are updated. You can use the DELETE clause of the MERGE statement to delete rows. You can delete rows either unconditionally or conditionally (using the WHERE clause).

With the MERGE statement, you can simultaneously update and insert rows in a particular table. This statement combines the update, insert, and delete operations that can be performed on a table based on the data in one or more source tables. The update and insertion of rows is based on a join condition. If the join condition is satisfied, then rows are updated and optionally deleted; otherwise, rows are inserted.

It is false that rows in the target table are updated when the join condition for merging is not met. The join condition in the MERGE statement specifies a condition for the target table to satisfy. The join condition uses the target table as well as the source table. If the join condition is satisfied, then rows in the target table are updated; however, if the join condition is not met, then new rows are inserted in the target table.

It is false that rows can be deleted even if they do not match the join condition for merging. The join condition is required for the merging of data from the source table with the data in the target table. If it is not satisfied, then rows are inserted instead of being updated. Only those rows that were updated by the MERGE statement can be deleted. You can specify a condition for which the rows can be deleted. However, if the delete condition is satisfied but the join condition for merging is not, then rows are deleted.

Item: 3 (Ref:Cert-1Z0-071.14.1.2)

You are the DBA for an insurance company, and are required to maintain data about the agents hired, insurance policies offered, and the policies sold by the company. As recognition to the agents who have sold 100 or more policies total to date, the company decides to increase their commission by 50% of their current commission.

A table called `agents` contains one row for each agent in the company.

```
SQL> desc agents
Name                Type
------------------------------
agent_id           NUMBER(3)
agent_name           VARCHAR2(25)
commission           NUMBER(5,1)
```

A table called `sold_policies` contains one row for each policy sold.

```
SQL> desc sold_policies
Name Type
---------------------------------
policy_id           NUMBER(3)
agent_id           NUMBER(3)
policy_holder      VARCHAR2(25)
```

Which of the following statements should you use to update the appropriate commissions?

- ○
```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies WHERE SUM(policy_id) >100;
```
- ○
```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies
GROUP BY agent_id);
```
- ○
```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies
GROUP BY agent_id
HAVING SUM(policy_id)>=100);
```
- ○
```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies GROUP BY agent_id
HAVING COUNT (agent_id ) >= 100;
```

Answer:

```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies GROUP BY agent_id
HAVING COUNT (agent_id ) >= 100;
```

Explanation:

In this scenario, you should use the following statement to return the desired results:

```
UPDATE agents SET commission=1.5*commission
WHERE agent_id IN (SELECT agent_id AS PolicySold
FROM sold_policies GROUP BY agent_id
HAVING COUNT (agent_id ) >= 100);
```

This statement is an UPDATE statement in which the WHERE clause has a subquery. This subquery returns the agent_id values from the sold_policies table grouped by agent_id, but only for the groups where the count of the number of agent_id's is greater than or equal to 100. The UPDATE statement then updates the value of the commission column in the agents table, but only for those agents whose agent_id values appeared more than 100 times in the agent_id column of the policies_sold table.

Item: 4 (Ref:Cert-1Z0-071.14.1.1)

In which two of the following clauses of the SELECT statement can you use a subquery? (Choose two.)

- [ ] ORDER BY
- [ ] FROM
- [ ] WHERE
- [ ] GROUP BY

Answer:

FROM
WHERE

Explanation:

You can use a subquery in the FROM and the WHERE clauses of a SELECT statement. A subquery is a SELECT statement within another DDL or DML statement. When you use a subquery in the FROM clause of the SELECT statement, it is known as an inline view. You can have an unlimited number of inline views in the FROM clause. When you use a subquery in the WHERE clause of the SELECT statement, it is known as nested subquery. There can be a maximum of 255 nested subqueries in the WHERE clause.

The ORDER BY and GROUP BY clauses of a SELECT statement are used to sort and group the rows in the result set based on a particular column. The SELECT statement, however, returns a result set of a single or multiple rows, which cannot be used to sort or group the rows.