Q3-v1 solution

Mcq:
1.B
2.C
3.C
4.B
5.b

Theory:

1.Use a SEQUENCE object starting at 100000 with INCREMENT BY 10, and call NEXT VALUE FOR across all tables to ensure unique, non-overlapping invoice numbers.

2.After SELECT INTO, you must manually add constraints (PK, FK, CHECK, DEFAULT, UNIQUE, indexes) to match the original table.

3.Return values: Single integer status code.
Output parameters: Can return multiple values to the calling program.

Q4. How do you add constraints to a table created with SELECT INTO?
Use ALTER TABLE to add constraints like PRIMARY KEY or UNIQUE after the table is created.

Q5. How do you retrieve the next value from a sequence?
Use SELECT NEXT VALUE FOR SequenceName.

Q6. Suppose the sales manager wants to reduce prices by 5% for all products in category 10. Write an SQL UPDATE query to achieve this.

```
UPDATE SalesLT.Product
SET ListPrice = ListPrice * 0.95
WHERE ProductCategoryID = 10;
```

Q7. What are @@ functions in SQL Server and how are they typically used in stored procedures?
@@ functions are global variables that return system-level information (e.g., @@ROWCOUNT, @@ERROR).
They help control flow and monitor execution results.

Q3-v2 solution

Mcq:
1. C
2.b
3.c
4.b
5.b

Theory:

Q1. How do you add constraints to a table created with SELECT INTO?
Use ALTER TABLE to add constraints like PRIMARY KEY or UNIQUE after the table is created.

Q2. Why can't views with aggregate functions or computed columns be updated?
Because SQL Server cannot map the update to specific rows in the base table due to data transformation/summarization.

Q3. How can INSTEAD OF triggers enable DML operations on views that are otherwise non-updatable?
They intercept the DML operation and redirect it to appropriate base tables, allowing controlled inserts, updates, or deletes.

Q4. How do stored procedures improve security in SQL Server?
Users can execute procedures without direct access to underlying tables, allowing controlled access and reduced permission complexity.

Q5. What is the purpose of SET IDENTITY_INSERT in SQL Server? When would you use it?
It allows explicit insertion into an identity column. Used when migrating data or reseeding identity values.

Q6. What is the difference between DELETE and TRUNCATE TABLE?
DELETE: Removes specific rows, supports WHERE, fully logged, fires triggers.
TRUNCATE: Removes all rows, no WHERE, minimally logged, does not fire triggers, resets identity.

Q7. What does the MERGE statement do? What are its 3 clauses?
It performs insert, update, or delete on a target table based on matching source data.

WHEN MATCHED → Update existing rows

WHEN NOT MATCHED BY TARGET → Insert new rows

WHEN NOT MATCHED BY SOURCE → Delete rows not present in source

Q3-v3 solution

Mcq:
1.b
2.c
3.c
4.c
5.a

Theory:

Q1. How does SCOPE_IDENTITY() differ from IDENT_CURRENT('table')?
 SCOPE_IDENTITY() returns the last identity value in the current session & scope.
 IDENT_CURRENT('table') returns the last identity value for the table across all sessions and scopes.

Q2. Why might the OUTPUT clause be useful in an auditing or logging system? Provide an example scenario.
 It captures affected rows during INSERT/UPDATE/DELETE — e.g., logging old and new values of salaries into an audit table automatically.

Q3. What does the MERGE statement do? What are its 3 clauses?
 It performs insert, update, or delete on a target table based on matching source data.

WHEN MATCHED → UPDATE

WHEN NOT MATCHED BY TARGET → INSERT

WHEN NOT MATCHED BY SOURCE → DELETE

Q4. How do you add constraints to a table created with SELECT INTO?
Use ALTER TABLE to add constraints like PRIMARY KEY, UNIQUE, DEFAULT, or FOREIGN KEY after creation.

Q5. How do stored procedures improve security in SQL Server?
They let users run procedures without direct table access, ensuring controlled access and reduced permission complexity.

Q6. Why would you update data using a JOIN? Provide a brief example.
To sync rows using data from another table.

```
UPDATE s

SET s.Email = t.NewEmail

FROM hr.students s

JOIN temp_students t ON s.StudentID = t.StudentID;
```

Q7. What is the difference between return values and output parameters in stored procedures?
Return values: a single integer status code.
Output parameters: can return multiple values to the calling program.

Q3-v4 solution

Mcq:
1.a
2.b
3.b
4.b
5.b

Theory:

Q1.
Use a SEQUENCE object.
Create a sequence starting at 100000, increment by 10. Each table can call NEXT VALUE FOR <sequence_name> to ensure non-overlapping invoice numbers.

Q2.
SCOPE_IDENTITY() returns the last identity value generated in the same scope and session.
IDENT_CURRENT('<table_name>') returns the last identity value generated for the specified table across all sessions and scopes.

Q3.
After SELECT INTO, you must manually add constraints (primary keys, foreign keys, unique, check) and indexes to make the backup behave like the original.

Q4.
SET NOCOUNT ON stops SQL Server from sending the "(X rows affected)" messages.
It improves performance in stored procedures/triggers by reducing network traffic.
Configure it with:

```
SET NOCOUNT ON;   -- inside procedure or session
```

Q5.
INSTEAD OF triggers intercept DML (INSERT/UPDATE/DELETE) on views.
They redirect operations to the underlying base tables, enabling modifications on views that would otherwise be non-updatable.

Q6.
Types of triggers:

AFTER triggers – Fire after DML, enforce business rules.

INSTEAD OF triggers – Replace the DML action, often used on views.

DDL triggers – Fire on schema changes (e.g., CREATE, DROP).

Q7.
The MERGE statement performs INSERT, UPDATE, DELETE in one go by comparing

a target and source.
Clauses:

WHEN MATCHED → update or delete.

WHEN NOT MATCHED BY TARGET → insert new rows.

WHEN NOT MATCHED BY SOURCE → delete rows from target.

Q3-v5 solution

Mcq:
1.d
2.d
3.b
4.c
5.b

Theory:

Q1. Why can't you directly update the value of an IDENTITY column in SQL Server? What is required to manually insert the value in such a column?
You can't directly update IDENTITY columns because SQL Server auto-generates their values. To manually insert, you must use SET IDENTITY_INSERT <table> ON.

Q2. Differentiate between DELETE and TRUNCATE statements
DELETE: Removes selected rows, supports WHERE, fully logged, fires triggers, identity not reset.
TRUNCATE: Removes all rows, no WHERE, minimally logged, does not fire triggers, resets identity.

Q3. How can INSTEAD OF triggers enable DML operations on views that are otherwise non-updatable?
They intercept the DML operation and redirect it to the base tables, allowing controlled INSERT, UPDATE, or DELETE actions.

Q4. What is the purpose of SET NOCOUNT ON in SQL Server? And how is it used to improve performance when client applications are dealing with databases?
It stops "x rows affected" messages from being returned. This reduces network traffic and improves performance in stored procedures and triggers.

Q5. Compare DELETE and TRUNCATE TABLE. How do they differ in terms of performance and usage?
DELETE: Slower, logs each row, used for selective deletion with WHERE.
TRUNCATE: Faster, minimally logged, used for quickly clearing entire tables and resetting identity.

Q6. Why is omitting a WHERE clause in an UPDATE or DELETE statement dangerous? Explain with an example.
Without WHERE, all rows are affected.
Example: DELETE FROM Employees; → removes all employee records instead of one.

Q7. How can INSTEAD OF triggers enable DML operations on views that are otherwise non-updatable?
By intercepting the DML statement on the view and rewriting it to perform the corresponding operation on the underlying base tables.