1. Transações são funções criadas e personalizadas. Utilizada para otimizar consultas avançadas. Garante que as transações sejam feitas da maneira correta, controla as operações.

2. Atomicidade: Garante referencia entre tabelas (?)

   Consistência: Consistência e integridade entre as tabelas, após passar por alteração continua consistente.

   Isolamento: Dados protegidos, permissões para execuções negadas. Isolar transações do código fonte do banco.

   Durabilidade: As restrições possuem durabilidade de acordo com o banco de dados.

3.
```
--3
BEGIN TRANSACTION atualizacao;
   DECLARE @numero_aluno INT;

   SELECT @numero_aluno = Numero_aluno
   FROM ALUNO
   WHERE Numero_aluno= 4343;

   IF EXISTS (SELECT 1 FROM ALUNO WHERE @numero_aluno = Numero_aluno)
   BEGIN
          UPDATE ALUNO
          SET Curso = 'ENG'
          WHERE ALUNO.Numero_aluno = 1;
          COMMIT TRANSACTION;
   END

   ELSE
   BEGIN
          PRINT 'ALUNO NAO EXISTENTE';
          ROLLBACK TRANSACTION;
   END
```

4. Triggers são gatilhos disparados ao sistema a partir de uma determinada condição ou ao fazer alguma ação. São utilizados antes ou após operações de inserir, atualização e excluir.

5.
```
--5
CREATE TRIGGER alunos_menores_idade
ON ALUNO
INSTEAD OF INSERT
AS
BEGIN
   DECLARE @datanasc DATE,
                @idade INT

   SELECT @datanasc = I.Data_Nascimento
   FROM inserted AS I;

   SELECT @idade = YEAR(GETDATE())-YEAR(I.Data_Nascimento)
   FROM inserted AS I;
   IF (@idade < 16)
   BEGIN
          RAISERROR ('NAO PODE ADICIONAR GENTE TAO NOVA',16,1);
```

```
            ROLLBACK TRANSACTION;
    END

    ELSE
    BEGIN
            INSERT INTO ALUNO
            SELECT *
            FROM INSERTED;
    END
END

INSERT INTO ALUNO (Nome, Numero_aluno, Tipo_aluno, Curso, Data_Nascimento)
VALUES
('Joãozão', 22, 1, 'SI', '2004-07-17');
```

6.

```
--6

CREATE TABLE Log_Aluno (
    logId INT IDENTITY(1,1) PRIMARY KEY,
    Numero_aluno INT,
    Operacao VARCHAR (10),
    Data_hora DATETIME DEFAULT GETDATE()
);


CREATE TRIGGER log_aluno_insert
ON ALUNO
AFTER INSERT
AS
BEGIN
    DECLARE @idAluno INT;

    SELECT @idAluno = I.Numero_aluno
    FROM inserted AS I;

    INSERT INTO Log_Aluno (Numero_aluno, operacao)
    VALUES (@idAluno,'INSERT');
END

SELECT * FROM LOG_aluno;

INSERT INTO ALUNO (Nome, Numero_aluno, Tipo_aluno, Curso, Data_Nascimento)
VALUES
('Marcelo', 22, 1, 'SI', '2004-07-17');

-----------------------------------------------------------------
CREATE TRIGGER log_aluno_update
ON ALUNO
AFTER UPDATE
AS
BEGIN
    DECLARE @idAluno INT;

    SELECT @idAluno = I.Numero_aluno
    FROM inserted AS I;

    INSERT INTO Log_Aluno (Numero_aluno, operacao)
    VALUES (@idAluno,'UPDATE');

END
```

```sql
UPDATE ALUNO
SET Nome = 'Joãozão Bala'
WHERE Numero_aluno = 1;

------------------------------------------------------------
ALTER TRIGGER log_aluno_delete
ON ALUNO
AFTER DELETE
AS
BEGIN
    DECLARE @idAluno INT;

    SELECT @idAluno = I.Numero_aluno
    FROM inserted AS I;

    INSERT INTO Log_Aluno (Numero_aluno, operacao)
    VALUES (@idAluno,'DELETE');

END

DELETE
FROM ALUNO
WHERE Numero_aluno = 22;

SELECT * FROM ALUNO;

SELECT * FROM Log_Aluno;
```

7. Garantem integridade dos dados. São importantes pois, podem ou não, permitir exclusão de informações utilizadas em outras tabelas.

8.

```sql
----8
ALTER TABLE HISTORICO_ESCOLAR
WITH CHECK ADD CONSTRAINT fk_identificacao_turma
FOREIGN KEY (identificacao_turma) REFERENCES TURMA (identificacao_turma)
ON UPDATE CASCADE;

UPDATE TURMA
SET Identificacao_turma = 777
WHERE Identificacao_turma = 102;

SELECT * FROM TURMA;

SELECT * FROM HISTORICO_ESCOLAR;
```

9. Views são consultas de pesquisa facilitadas. Permitem criação de pesquisas específicas e fácil reutilização. Permitem também, restrições de pesquisas. Ideais para pesquisas recorrentes.

10. Sdadadadada

```
--10
ALTER VIEW V_Desempenho_Alunos
AS

SELECT A.Numero_aluno,
       A.Nome,
       A.Curso,
       D.Nome_disciplina,
       T.Semestre,
       T.Ano,
       H.Nota

FROM ALUNO AS A
JOIN HISTORICO_ESCOLAR AS H ON A.Numero_aluno = H.Numero_aluno
JOIN TURMA AS T ON H.Identificacao_turma = T.Identificacao_turma
JOIN DISCIPLINA AS D ON T.Numero_disciplina = D.Numero_disciplina;

SELECT *
FROM V_Desempenho_Alunos;
/* Organizando para os Joins
HISTORICO_ESCOLAR AS H
   H.NOTA
   H.ANO
   H.SEMESTRE
DISCIPLINA AS D
   D.NOME_DISCIPLINA

*/
```

11.

```
--11

ALTER VIEW V_Alunos_Curso
AS

SELECT A.Numero_aluno,
       A.Nome,
       A.Curso
FROM ALUNO AS A
ORDER BY A.Curso;

SELECT *
FROM V_Alunos_Curso;
```