DELIMITER $$

-- Trigger 1: Registrar data de criação em 'usuario'

CREATE TRIGGER before\_insert\_usuario

BEFORE INSERT ON `usuario`

FOR EACH ROW

BEGIN

SET NEW.dataNasc = IFNULL(NEW.dataNasc, CURDATE());

END$$

INSERT INTO usuario (nome, email) VALUES ('João Silva', 'joao@email.com');

DELIMITER $$

-- Trigger 2: Garantir telefone único por usuário

CREATE TRIGGER before\_insert\_telefone

BEFORE INSERT ON `telefone`

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM `telefone` WHERE numero = NEW.numero AND usuario\_idUsuario = NEW.usuario\_idUsuario

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'O número de telefone já está registrado para este usuário.';

END IF;

END$$

DELIMITER $$

-- Trigger 3: Garantir relação válida entre cliente e usuário

CREATE TRIGGER before\_insert\_cliente

BEFORE INSERT ON `cliente`

FOR EACH ROW

BEGIN

IF NOT EXISTS (

SELECT 1 FROM `usuario` WHERE idUsuario = NEW.usuario\_idUsuario

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Usuário associado ao cliente não existe.';

END IF;

END$$

DELIMITER $$

-- Trigger 4: Atualizar log após alteração em 'alteracao'

CREATE TRIGGER after\_insert\_alteracao

AFTER INSERT ON `alteracao`

FOR EACH ROW

BEGIN

INSERT INTO `log\_alteracoes` (`idalteracao`, `dataAlteracao`, `descricao`)

VALUES (NEW.idalteracao, NEW.dataAlteracao, CONCAT('Alteração de serviço: ', NEW.novoServico));

END$$

DELIMITER $$

-- Trigger 5: Impedir agendamentos duplicados

CREATE TRIGGER before\_insert\_agendamento

BEFORE INSERT ON `agendamento`

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM `agendamento` WHERE data = NEW.data AND hora = NEW.hora

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Já existe um agendamento para esta data e hora.';

END IF;

END$$

DELIMITER $$

-- Trigger 6: Calcular preço total em 'servicos\_has\_agendamento'

CREATE TRIGGER after\_insert\_servicos\_has\_agendamento

AFTER INSERT ON `servicos\_has\_agendamento`

FOR EACH ROW

BEGIN

DECLARE total DECIMAL(10,2);

SELECT SUM(preco) INTO total

FROM `servicos`

WHERE idservicos = NEW.servicos\_idservicos;

UPDATE `agendamento`

SET totalPreco = total

WHERE idagendamento = NEW.agendamento\_idagendamento;

END$$

DELIMITER $$

-- Trigger 7: Verificar se especialista está disponível

CREATE TRIGGER before\_insert\_Disponibilidade

BEFORE INSERT ON `Disponibilidade`

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM `Disponibilidade`

WHERE data = NEW.data AND hora = NEW.hora AND especialista\_idespecialista = NEW.especialista\_idespecialista

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Especialista já está ocupado nesse horário.';

END IF;

END$$

DELIMITER $$

-- Trigger 8: Atualizar tabela de serviços após remoção de um agendamento

CREATE TRIGGER after\_delete\_agendamento

AFTER DELETE ON `agendamento`

FOR EACH ROW

BEGIN

DELETE FROM `servicos\_has\_agendamento`

WHERE agendamento\_idagendamento = OLD.idagendamento;

END$$

DELIMITER $$

-- Trigger 9: Atualizar total de agendamentos por administrador

CREATE TRIGGER after\_insert\_agendamento\_count

AFTER INSERT ON `agendamento`

FOR EACH ROW

BEGIN

UPDATE `administrador`

SET totalAgendamentos = (SELECT COUNT(\*) FROM `agendamento` WHERE administrador\_idadministrador = NEW.administrador\_idadministrador)

WHERE idadministrador = NEW.administrador\_idadministrador;

END$$

DELIMITER $$

-- Trigger 10: Bloquear exclusão de usuários com agendamentos ativos

CREATE TRIGGER before\_delete\_usuario

BEFORE DELETE ON `usuario`

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM `agendamento` WHERE administrador\_usuario\_idUsuario = OLD.idUsuario

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Usuário não pode ser excluído enquanto possuir agendamentos ativos.';

END IF;

END$$

CREATE TRIGGER after\_insert\_agendamento\_log

AFTER INSERT ON `agendamento`

FOR EACH ROW

BEGIN

INSERT INTO `log\_agendamentos` (`mensagem`, `dataRegistro`)

VALUES (CONCAT('Agendamento criado: Data - ', NEW.data, ', Hora - ', NEW.hora), NOW());

END;

DELIMITER ;