Código de criação das Triggers’s

1) Preenche automaticamente a tabela Historico_cliente

```
CREATE OR REPLACE FUNCTION registraHistCli()
RETURNS trigger AS
$$
BEGIN
	if (TG_OP = 'INSERT') THEN
	INSERT INTO public.historico_cliente(
	id_cliente, nome, cpf, email, senha, cli_datainse, alteracao, data_alt)
	VALUES (new.id_cliente, new.nome, new.cpf, new.email, new.senha,
new.cli_datainse,'INSERT', now());

	elsif (TG_OP = 'DELETE') THEN
	INSERT INTO public.historico_cliente(
	id_cliente, nome, cpf, email, senha, cli_datainse, alteracao, data_alt)
	VALUES (old.id_cliente, old.nome, old.cpf, old.email, old.senha,
old.cli_datainse,'DELETE', now());

	elsif (TG_OP = 'UPDATE') THEN
	INSERT INTO public.historico_cliente(
	id_cliente, nome, cpf, email, senha, cli_datainse, alteracao, data_alt)
	VALUES (old.id_cliente, old.nome, old.cpf, old.email, old.senha,
old.cli_datainse,'UPDATE', now());
	end if;
	return NULL;
	 COMMIT;

	END; $$ language plpgsql;

	CREATE TRIGGER trigger_registraHistCli
	AFTER INSERT OR UPDATE OR DELETE ON cliente
	FOR EACH ROW
	EXECUTE PROCEDURE registraHistCli();
```

2) Preenche automaticamente a tabela Historico_produto

```
CREATE OR REPLACE FUNCTION registraHistProd()
RETURNS trigger AS
$$
BEGIN
```

```
if (TG_OP = 'INSERT') THEN
INSERT INTO public.historico_produto(
id_produto, id_categoria, nome, quantidade, preco, id_fornecedor, alteracao, data_alt)
VALUES (new.id_produto, new.id_categoria, new.nome, new.quantidade,
new.quantidade, new.preco, 'INSERT', now());


elsif (TG_OP = 'DELETE') THEN
INSERT INTO public.historico_produto(
id_produto, id_categoria, nome, quantidade, preco, id_fornecedor, alteracao, data_alt)
VALUES (old.id_produto, old.id_categoria, old.nome, old.quantidade, old.quantidade,
old.preco, 'DELETE', now());


elsif (TG_OP = 'UPDATE') THEN
INSERT INTO public.historico_produto(
id_produto, id_categoria, nome, quantidade, preco, id_fornecedor, alteracao, data_alt)
VALUES (old.id_produto, old.id_categoria, old.nome, old.quantidade, old.quantidade,
old.preco, 'UPDATE', now());

end if;
return NULL;
 COMMIT;

END; $$ language plpgsql;

CREATE TRIGGER trigger_registraHistProd
AFTER INSERT OR UPDATE OR DELETE ON produto
FOR EACH ROW
EXECUTE PROCEDURE registraHistProd();
```

3) Deleta em cascada dos os dados sobre o cliente

```
CREATE OR REPLACE FUNCTION casc_cliente()
RETURNS trigger AS
$$
BEGIN
                DELETE FROM telefone as t
                where t.id_cliente=old.id_cliente;

                DELETE FROM card as c
                where c.id_cliente=old.id_cliente;

                DELETE FROM item_pedido as ip
                where ip.id_pedido in (select id_pedido from pedido where id_cli =
old.id_cliente);
```

```
                DELETE FROM pedido as p
                where p.id_cli=old.id_cliente;

                DELETE FROM endereco as e
                where e.id_cliente=old.id_cliente;

return old;
COMMIT;
END; $$ language plpgsql;

CREATE TRIGGER trigger_casc_cliente
BEFORE DELETE ON cliente
FOR EACH ROW
EXECUTE PROCEDURE casc_cliente();
```