A8: Base de dados povoada com dados

Neste artefacto é apresentado o *script* completo de criação da base de dados, englobando:

- todos os índices e *clusters* utilizados.
- todas as restrições de integridade (*constraints* e *checks*).
- todos os *triggers* e os predicados que estes utilizam.
- todos os *inserts* utilizados para povoar a base de dados.

Estes *inserts* demonstram valores credíveis e possíveis no contexto do nosso projeto e que demonstram as regras de negócio. Assim, por exemplo, um *Feedback* que seja entre os Membros 1 e 2 tem que existir, no máximo, duas vezes por leilão. Outro exemplo possível de verificação de integridade é que o valor da licitação mais alta, para um dado leilão, tem que ser necessariamente o tuplo da tabela Licitacao com o valor mais alto (para aquele leilão).

```
-- TABELAS

-- Função necessária para o constraint confirmaCliente funcionar:

CREATE OR REPLACE FUNCTION confirmaCliente(
        idCliente INTEGER
    )
  RETURNS VARCHAR(5) AS
$confirmaCliente$
BEGIN
 IF EXISTS (SELECT * FROM Membro WHERE idCliente = Membro.idUtilizador AND
Membro.tipoMembro LIKE 'Cliente')
 THEN
    RETURN 'True';
 END IF;
 RETURN 'False';
END;
$confirmaCliente$ LANGUAGE plpgsql;

-- Tabelas

DROP TABLE IF EXISTS Membro CASCADE
;

CREATE TABLE Membro
(
    idUtilizador SERIAL,
    nomeUtilizador VARCHAR(20) NOT NULL,
    email TEXT NOT NULL,
    password TEXT NOT NULL,
    dataNascimento DATE NOT NULL,
    nomeCivil TEXT,
        tipoMembro VARCHAR(7),
```

```
    PRIMARY KEY (idUtilizador),
    UNIQUE (nomeUtilizador),
    UNIQUE (email),
    CONSTRAINT dataCorreta CHECK (EXTRACT(YEAR FROM CURRENT_DATE)-
EXTRACT(YEAR FROM DataNascimento) >= 18),
        CONSTRAINT tipoMembroCorreto CHECK (tipoMembro LIKE 'Admin' OR
tipoMembro LIKE 'Cliente')
)
;

DROP TABLE IF EXISTS MembroBanido CASCADE
;

CREATE TABLE MembroBanido
(

    idMembroBanido INTEGER,
    dataBanido TIMESTAMP NOT NULL DEFAULT NOW(),
    duracao INTEGER,
    motivo TEXT,

    PRIMARY KEY (idMembroBanido),
    FOREIGN KEY (idMembroBanido)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Mensagem CASCADE
;

CREATE TABLE Mensagem
(
    idMensagem SERIAL,
    idEmissor INTEGER NOT NULL,
    idRecetor INTEGER NOT NULL,
    texto VARCHAR(5000) NOT NULL,
    dataMensagem TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idMensagem),
    FOREIGN KEY (idRecetor)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idEmissor)
        REFERENCES Membro(idUtilizador),
    CONSTRAINT comprimentoMensagem CHECK (CHAR_LENGTH(texto) >  AND
CHAR_LENGTH(texto) < 5000)
)
;

DROP TABLE IF EXISTS Notificacao CASCADE
;
```

```
CREATE TABLE Notificacao
(
    idNotificacao SERIAL,
    idUtilizador INTEGER NOT NULL,
    texto TEXT NOT NULL,
    dataNotificacao TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idNotificacao),
    FOREIGN KEY (idUtilizador)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Marca CASCADE
;

CREATE TABLE Marca
(
    idMarca SERIAL,
    nome VARCHAR(20) NOT NULL,

    PRIMARY KEY (idMarca),
    UNIQUE(nome)
)
;

DROP TABLE IF EXISTS Feedback CASCADE
;

CREATE TABLE Feedback
(
    idFeedback SERIAL,
    texto VARCHAR(30) NOT NULL,
    dataFeedback TIMESTAMP NOT NULL DEFAULT NOW(),
    valor INTEGER DEFAULT 5,

    PRIMARY KEY (idFeedback),
    CONSTRAINT tamanhoFeedback CHECK (CHAR_LENGTH(texto) >= 1 AND
CHAR_LENGTH(texto) <= 30),
    CONSTRAINT valorFeedback CHECK (valor >=  AND valor <= 5)
)
;

DROP TABLE IF EXISTS FeedbackComprador CASCADE
;

CREATE TABLE FeedbackComprador
(
    idFeedback INTEGER,
    idComprador INTEGER,
```

```
    PRIMARY KEY (idFeedback, idComprador),
    FOREIGN KEY (idComprador)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idFeedback)
        REFERENCES Feedback(idFeedback)
)
;

DROP TABLE IF EXISTS FeedbackLeiloeiro CASCADE
;

CREATE TABLE FeedbackLeiloeiro
(
    idFeedback INTEGER,
    idLeiloeiro INTEGER,

    PRIMARY KEY (idFeedback, idLeiloeiro),
    FOREIGN KEY (idLeiloeiro)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idFeedback)
        REFERENCES Feedback(idFeedback)
)
;

DROP TABLE IF EXISTS Leilao CASCADE
;

CREATE TABLE Leilao
(
    idLeilao  SERIAL,
    nome VARCHAR(50) NOT NULL,
    descricao VARCHAR(5000) NOT NULL,
    licitacaoBase FLOAT NOT NULL,
    licitacaoAtual FLOAT DEFAULT NULL,
    dataColocacao TIMESTAMP NOT NULL DEFAULT NOW(),
    duracao FLOAT NOT NULL,
    idLeiloeiro INTEGER NOT NULL,
    idMarca INTEGER,
    idFeedbackLeiloeiro INTEGER DEFAULT NULL,
    idFeedbackCliente INTEGER DEFAULT NULL,

    PRIMARY KEY (idLeilao),
    FOREIGN KEY (idLeiloeiro)
        REFERENCES Membro(idUtilizador),
    FOREIGN KEY (idMarca)
        REFERENCES Marca(idMarca),
    FOREIGN KEY (idFeedbackLeiloeiro)
        REFERENCES Feedback(idFeedback),
    FOREIGN KEY (idFeedbackCliente)
        REFERENCES Feedback(idFeedback),
    CONSTRAINT tamanhoDescricao CHECK (CHAR_LENGTH(descricao) > 10 AND
```

```
CHAR_LENGTH(descricao) < 2500),
    CONSTRAINT valorLicitacaoBase CHECK (licitacaoBase > 0.01),
    CONSTRAINT dataCorreta CHECK (dataColocacao <= NOW()),
    CONSTRAINT duracaoLeilao CHECK (duracao >= 1.0 AND duracao <= 14.0),
        CONSTRAINT confirmaCliente CHECK (confirmaCliente(idLeiloeiro) =
'True')
)
;

DROP TABLE IF EXISTS Imagem CASCADE
;

CREATE TABLE Imagem
(
    idImagem SERIAL,
    idLeilao INTEGER NOT NULL,
    link TEXT NOT NULL,

    PRIMARY KEY (idImagem),
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao),
    UNIQUE(link)
)
;

DROP TABLE IF EXISTS Licitacao CASCADE
;

CREATE TABLE Licitacao
(
    idLicitacao SERIAL,
    idLeilao INTEGER NOT NULL,
    idCliente INTEGER NOT NULL,
    valor FLOAT NOT NULL,
    dataLicitacao TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idLicitacao),
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao)
        ON DELETE CASCADE,

        CONSTRAINT confirmaCliente CHECK ( confirmaCliente(idCliente) =
'True' )
)
;

DROP TABLE IF EXISTS Preferencia CASCADE
;
```

```
CREATE TABLE Preferencia
(
    idCliente INTEGER,
    idMarca INTEGER,

    PRIMARY KEY (idCliente , idMarca),
    FOREIGN KEY (idMarca)
        REFERENCES Marca(idMarca) ,
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Registo CASCADE
;

CREATE TABLE Registo
(
    idCliente INTEGER,
    idLeilao INTEGER,

    PRIMARY KEY (idCliente , idLeilao),
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao) ,
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador)
)
;



-- GATILHOS


CREATE OR REPLACE FUNCTION atualizarLeilao ()
  RETURNS TRIGGER AS
$$
    BEGIN
        INSERT INTO Notificacao (idUtilizador, texto) VALUES
        ((SELECT idLeiloeiro FROM Leilao WHERE idLeilao = NEW.idLeilao),
            'O cliente ' || (SELECT nomeUtilizador FROM Membro WHERE
idUtilizador = NEW.idCliente) || ' licitou no seu leilao ' || (SELECT nome
FROM Leilao WHERE idLeilao = NEW.idLeilao) || '.');
        IF (NEW.valor > (SELECT valor FROM Licitacao WHERE idLeilao =
NEW.idLeilao ORDER BY valor LIMIT 1))
        THEN
            INSERT INTO Notificacao (idUtilizador, texto) VALUES
            ((SELECT idCliente FROM Licitacao WHERE idLeilao = NEW.idLeilao
ORDER BY valor LIMIT 1),
                'A sua licitacao no leilao ' || (SELECT nome FROM Leilao
WHERE idLeilao = NEW.idLeilao) || ' foi ultrapassada.');
```

```
        END IF;
        UPDATE Leilao SET licitacaoAtual = NEW.valor WHERE idLeilao =
NEW.idLeilao AND
            (NEW.valor > licitacaoAtual OR licitacaoAtual IS NULL);
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER maiorLicitacao
BEFORE INSERT
ON Licitacao
FOR EACH ROW EXECUTE PROCEDURE atualizarLeilao();



CREATE OR REPLACE FUNCTION notificaLicitacoesRemovidas ()
    RETURNS TRIGGER AS
$$
  DECLARE
    _leiloeiro INTEGER;
    _licitacao INTEGER;
    _nomeLeilao VARCHAR(50);
  BEGIN
        FOR _licitacao IN (SELECT idLicitacao FROM Licitacao WHERE idLeilao
= OLD.idLeilao)
        LOOP
            SELECT nome FROM Leilao WHERE idLeilao = OLD.idLeilao INTO
_nomeLeilao;
            SELECT idCliente FROM Licitacao WHERE idLicitacao = _licitacao
INTO _leiloeiro;
            IF _leiloeiro IS NOT NULL THEN
                INSERT INTO Notificacao (idUtilizador, texto) VALUES
(_leiloeiro, 'O leilao ' || _nomeLeilao || ' foi eliminado, como tal as tuas
licitacoes nesse leilao tambem foram eliminadas.');
            END IF;
        END LOOP;
        RETURN OLD;
  END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER leilaoApagado
BEFORE DELETE
ON Leilao
FOR EACH ROW EXECUTE PROCEDURE notificaLicitacoesRemovidas ();



CREATE OR REPLACE FUNCTION atualizaMaiorLicitacao ()
    RETURNS void AS
$$
    DECLARE
        _leilao INTEGER;
```

```
        _cliente INTEGER;
        _nomeLeilao VARCHAR(50);
    BEGIN
        FOR _leilao IN (SELECT idLeilao FROM Leilao)
        LOOP
            UPDATE Leilao SET licitacaoAtual = (SELECT valor FROM Licitacao
WHERE idLeilao = _leilao ORDER BY valor DESC LIMIT 1)
            WHERE idLeilao = _leilao RETURNING nome INTO _nomeLeilao;
            SELECT idCliente INTO _cliente FROM Licitacao WHERE idLeilao =
_leilao ORDER BY valor DESC LIMIT 1;
            IF (_cliente) IS NOT NULL THEN
                INSERT INTO Notificacao (idUtilizador, texto) VALUES (
_cliente, 'A sua licitacao no leilao ' || _nomeLeilao || ' voltou a ser a
mais alta.');
            END IF;
        END LOOP;
    END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION membroBanido ()
    RETURNS TRIGGER AS
$$
    BEGIN
        -- Leiloeiro banido
        DELETE FROM Leilao
            WHERE idLeiloeiro = NEW.idMembroBanido;
        -- Cliente banido
        DELETE FROM Licitacao
            WHERE idCliente = NEW.idMembroBanido;
        EXECUTE atualizaMaiorLicitacao ();
        INSERT INTO Notificacao (idUtilizador, texto) VALUES (
NEW.idMembroBanido, 'Foi banido durante ' ||  NEW.duracao ||
         ' dias, como tal todos os seu leiloes/licitacoes foram apagadas.
Mensagem do administrador: ' || NEW.motivo );
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER membroBanido
AFTER INSERT
ON MembroBanido
FOR EACH ROW EXECUTE PROCEDURE membroBanido ();


CREATE OR REPLACE FUNCTION notificaMensagem ()
    RETURNS TRIGGER AS
$$
    BEGIN
        INSERT INTO Notificacao (idUtilizador, texto) VALUES (
NEW.idRecetor, 'Recebeu uma nova mensagem de ' || (SELECT nomeUtilizador
FROM Membro WHERE idUtilizador = NEW.idEmissor) || '.' );
```

```
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER mensagem
AFTER INSERT
ON Mensagem
FOR EACH ROW EXECUTE PROCEDURE notificaMensagem ();

-- ÍNDICES E CLUSTERS

CREATE INDEX nomeUtilizadorIndex ON Membro USING hash(nomeUtilizador);

CREATE INDEX emailUtilizadorIndex ON Membro USING hash(email);

CREATE INDEX linkImagemIndex ON Imagem USING hash(linkImagem);

CREATE INDEX tipoMembroIndex ON Membro USING hash(tipoMembro);

CREATE INDEX dataColocacaoIndex ON Leilao USING btree(dataColocacao);

CREATE INDEX nomeLeilaoIndex ON Leilao USING btree(nome);

CREATE INDEX motivoIndex ON MembroBanido USING btree(motivo);

CREATE INDEX dataFeedbackIndex ON Feedback USING btree(dataFeedback);

 -- encontra leilões rapidamente pelo seu nome.
CLUSTER leilao USING nomeLeilaoIndex;

 -- encontra membros banidos rapidamente pelo motivo pelo quais foram
banidos.
CLUSTER membroBanido USING motivoIndex;

-- encontra membros rapidamente pelo seu nome.
CLUSTER membro USING nomeUtilizadorIndex;

-- encontra mensagens rapidamente pelo índice das descrições.
CLUSTER mensagem USING msgidx;

-- encontra feedbacks rapidamente pela sua data de colocação.
CLUSTER feedback USING dataFeedbackIndex;

-- POVOAR BASE DE DADOS

INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('joaoferreira', 'jaoo@up.pt', '123abc',
'1990-03-03', 'Joao Ferreira', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('anamendes', 'amendes@up.pt', 'letmein',
'1993-10-20', 'Ana Mendes', 'Cliente');
```

```
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('ruiguimaraes', 'ruiguima@up.pt', 'awsd',
'1982-04-10', 'Rui Guimaraes', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('bertopassos', 'a.passos@up.pt', 'plot25',
'1988-01-06', 'Alberto Passos', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('joaoestrada', 'jestrada@up.pt', 'kek',
'1995-02-15', 'Joao Estrada', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('diogoreis', 'diogoreis@up.pt', '85ykbd',
'1987-06-10', 'Diogo Reis', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('joanagomes', 'jogomes@up.pt', 'bottomsup',
'1990-12-21', 'Joana Gomes', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('davidpereira', 'davsper@up.pt', 'fgbuf',
'1991-07-12', 'David Pereira', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('tiagomatos', 'tiagomatos@up.pt', 'wfwfoihn',
'1993-05-01', 'Tiago Matos', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('sergiomad', 'sergio.madeira@up.pt',
'jino44', '1989-02-28', 'Sergio Madeira', 'Cliente');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('pedrofraga', 'fraga@up.pt', 'letmein',
'1995-08-03', 'Pedro Fraga', 'Admin');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('luisoliveira', 'luis@up.pt', '123abc',
'1995-12-06', 'Luis Oliveira', 'Admin');
INSERT INTO Membro (nomeUtilizador, email, password, dataNascimento,
nomeCivil, tipoMembro) VALUES ('pedromartins', 'pvlmartins@up.pt',
'canttouchthis', '1992-03-25', 'Pedro Martins', 'Admin');

INSERT INTO Marca (nome) VALUES ('Ford');
INSERT INTO Marca (nome) VALUES ('Renault');
INSERT INTO Marca (nome) VALUES ('Seat');
INSERT INTO Marca (nome) VALUES ('Citroen');
INSERT INTO Marca (nome) VALUES ('Toyota');
INSERT INTO Marca (nome) VALUES ('Mercedes');
INSERT INTO Marca (nome) VALUES ('Nissan');

INSERT INTO Feedback (texto) VALUES ('Muito prestavel');
INSERT INTO Feedback (texto, valor) VALUES ('Correu bem', 4);
INSERT INTO Feedback (texto, valor) VALUES ('Pagamento demorado', 2);
INSERT INTO Feedback (texto, valor) VALUES ('Filtro de ar entupido', 3);
INSERT INTO Feedback (texto, valor) VALUES ('Horrivel! Não recomendo', 1);

INSERT INTO FeedbackComprador (idFeedback, idComprador) VALUES (1, 3);
INSERT INTO FeedbackComprador (idFeedback, idComprador) VALUES (4, 8);
INSERT INTO FeedbackComprador (idFeedback, idComprador) VALUES (5, 10);
```

```
INSERT INTO FeedbackLeiloeiro (idFeedback, idLeiloeiro) VALUES (2, 1);
INSERT INTO FeedbackLeiloeiro (idFeedback, idLeiloeiro) VALUES (3, 2);

INSERT INTO Leilao (nome, descricao, licitacaoBase, licitacaoAtual,
dataColocacao, duracao, idLeiloeiro, idMarca, idFeedbackLeiloeiro,
idFeedbackCliente) VALUES ('Toyota Prius - Bom estado', 'Carro de 2006 em
bom estado, na zona do Porto', 5000, 10500,'2016-03-15 14:03:52', 10, 1, 5,
1, 1);
INSERT INTO Leilao (nome, descricao, licitacaoBase, licitacaoAtual,
dataColocacao, duracao, idLeiloeiro, idMarca, idFeedbackLeiloeiro,
idFeedbackCliente) VALUES ('Ford Focus 1600 Turbo Diesel', 'Carro de 2009
com 150000Km, bem conservado', 6000, 8600,'2016-01-10 21:41:12', 13, 2, 1,
2, 2);
INSERT INTO Leilao (nome, descricao, licitacaoBase, licitacaoAtual,
dataColocacao, duracao, idLeiloeiro, idMarca, idFeedbackCliente) VALUES
('Seat Ibiza 140CV', 'Veiculo a gasolina de 1995, com 170000km, em bom
estado. Contacte pra saber mais', 800, 1100,'2016-02-22 10:20:30', 7, 3, 3,
3);
INSERT INTO Leilao (nome, descricao, licitacaoBase, licitacaoAtual,
dataColocacao, duracao, idLeiloeiro, idMarca) VALUES ('Renault Clio 2008',
'Em optimo estado, 100000km, revisão e inspecção em dia', 5500, 8000,
'2016-04-02 17:32:11', 14, 4, 2);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Citroen DS3 Turbo Diesel', 'Carro de 2013, atual,
citadino, 3 portas, bem estimado', 10000, 14, 5, 4);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Citroen Saxo', 'Veiculo de 2004, com 140000km, bem
conservado', 3000, 10, 6, 4);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Seat Leon FR', 'Veiculo de 2011, 5 portas, como novo',
9000, 7, 5, 3);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Ford Fiesta', 'Carro de 1999, com inspecção em dia e em
muito bom estado.', 1200, 5, 7, 1);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Toyota Auris', 'Carro de 2009, em muito bom estado, pouco
uso.', 7500, 12, 8, 5);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Renault Megane', 'Modelo de 2007, muito espaçoso. Inclui
uma mala de tejadilho', 5000, 8, 9, 2);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Mercedes Classe A', 'Modelo de 2012, em optimo estado,
apenas 17000km', 16000, 7, 10, 6);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Mercedes Classe C', 'Modelo de 2010, da zona do Porto, bem
estimado', 10000, 10, 1, 6);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
idMarca) VALUES ('Renault Captur', 'Modelo de 2013, na zona do Porto, em
muito boas condições, como novo!', 7000, 5, 6, 2);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro,
```

```
idMarca) VALUES ('Nissan Rogue', 'Carro de 2011, bem cuidado, com inspecção em dia.', 8000, 8, 10, 7);
INSERT INTO Leilao (nome, descricao, licitacaoBase, duracao, idLeiloeiro, idMarca) VALUES ('Nissan Qashqai', 'Carro de 2012, na zona de Lisboa, muito bem estimado', 8700, 5, 9, 7);

INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (1, 2, 6000, '2016-03-18 23:05:40');
INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (1, 4, 8500, '2016-03-20 09:12:33');
INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (1, 3, 10500, '2016-03-24 13:28:00');
INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (2, 9, 7000, '2016-01-12 05:03:21');
INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (2, 8, 8600, '2016-01-20 21:14:50');
INSERT INTO Licitacao (idLeilao, idCliente, valor, dataLicitacao) VALUES (3, 10, 1100, '2016-02-25 05:03:21');
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (4, 5, 8300);
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (4, 6, 8400);
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (5, 1, 10500);
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (6, 7, 3400);
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (7, 3, 9350);
INSERT INTO Licitacao (idLeilao, idCliente, valor) VALUES (7, 5, 9500);


INSERT INTO Mensagem (idEmissor, idRecetor, texto, dataMensagem) VALUES (5, 7, 'Reparei no que fazes com as laranjas. Se eu te trouxer uma vaca tiras-lhe o leite?', '2016-02-13 16:23:59');
INSERT INTO Mensagem (idEmissor, idRecetor, texto) VALUES (2, 12, 'Ola. Eu tenho uma duvida que gostava de ver esclarecida. Agradecia que me contactasse de volta. Cumprimentos');
INSERT INTO Mensagem (idEmissor, idRecetor, texto) VALUES (12, 13, 'Hey, precisamos de reunir para discutir sobre umas alteracoes que gostava de fazer no site.');
INSERT INTO Mensagem (idEmissor, idRecetor, texto) VALUES (3, 7, 'Ola. Gostava de obter mais informacoes sobre o carro. Cumprimentos');
INSERT INTO Mensagem (idEmissor, idRecetor, texto) VALUES (7, 3, 'Sim, claro. Tal como diz na descricao o carro e de 1999, tem 132450km e esta em muito bom estado. Obrigado, disponha!');

INSERT INTO MembroBanido (idMembroBanido, duracao, motivo) VALUES (3, 365, 'Foi rude');

INSERT INTO Registo (idCliente, idLeilao) VALUES (3, 5);
INSERT INTO Registo (idCliente, idLeilao) VALUES (3, 8);
INSERT INTO Registo (idCliente, idLeilao) VALUES (4, 5);
INSERT INTO Registo (idCliente, idLeilao) VALUES (2, 2);
INSERT INTO Registo (idCliente, idLeilao) VALUES (2, 5);

INSERT INTO Preferencia (idCliente, idMarca) VALUES (6, 4);
```

```
INSERT INTO Preferencia (idCliente, idMarca) VALUES (6, 3);
INSERT INTO Preferencia (idCliente, idMarca) VALUES (6, 5);
INSERT INTO Preferencia (idCliente, idMarca) VALUES (6, 2);
INSERT INTO Preferencia (idCliente, idMarca) VALUES (6, 1);
INSERT INTO Preferencia (idCliente, idMarca) VALUES (7, 3);
INSERT INTO Preferencia (idCliente, idMarca) VALUES (7, 4);
```