1. **Para este trabalho e preciso gerar uma base de testes para o sistema de HOSPEDAGEM. Portanto, devemos escrever procedimentos para realizar o seguinte:**

   a) **Escrever um procedimento para inserir registros na tabela de HÓSPEDES:**
      - Esse procedimento deve receber por parâmetro a quantidade de hóspedes que deverão ser inseridos e dois outros parâmetros indicando a idade mínima e máxima de cada hóspede;
      - A idade mínima deverá ser menor que a máxima. Sendo que a idade mínima deverá ser superior a 18 e a máxima inferior a 65;
      - As cidades deverão ser obtidas aleatoriamente da tabela CIDADE.

   b) **Escrever procedimento para inserir registros na tabela ATENDENTE:**
      - Receber por parâmetro a quantidade de atendentes que deverão ser gerados;
      - Fixar que o atendente 1 é superior de todos os demais.

   c) **Escrever procedimento para inserir registros na tabela HOSPEDAGEM:**
      - Receber por parâmetro a quantidade de hospedagens que deverão ser geradas e o intervalo de tempo para o qual serão geradas as diárias (duas datas);
      - As hospedagens serão aleatoriamente vinculadas a hóspedes e atendentes
      - A data de entrada da hospedagem deverá ser gerada de forma que esteja dentro do intervalo passado por parâmetro;
      - O sistema deverá considerar que as datas de saída de algumas hospedagens deverão ser preenchidas (vamos imaginar que o hotel tem um número de quartos que vai do 1 ao 100. Logo, somente uma hospedagem poderá estar aberta para esses quartos ao mesmo tempo – sempre a mais recente);

      Para facilitar imagine que a estadia de uma pessoa não ultrapassa 3 dias.

   d) **Escreva um procedimento para atualizar dados na tabela de hospedagem da seguinte forma:**
      - Receber por parâmetro o código da hospedagem;
      - Somente podem ser atualizados os campos datasaida, codatendente e valorDiaria;
      - Nem todos os campos serão atualizados ao mesmo tempo, ou seja, haverá situações em que apenas um, dois ou os três serão atualizados;
      - Utilizar um único UPDATE de forma dinâmica;
      - Ao final retornar a quantidade de linhas atualizadas.

1.

a)

```
CREATE OR REPLACE FUNCTION
fn_insere_hospede (qtd int, min int, max int) RETURNS VOID AS $$

DECLARE

_MIN int;
_MAX int;
_DATA date;
_ID_NOME int;
_ID_CIDADE int;
_NOME varchar(50);
_CIDADE varchar(50);
_SIM int;
_NAO int;

BEGIN

_SIM = 0;
_NAO = 0;

FOR i IN 1..qtd LOOP

        _MIN = trunc(random() * max + min);
        _MAX = trunc(random() * max + min);

        -- seleciona randomicamente uma data no intervalo e, um nome e uma cidade de suas tabelas:

        _ID_NOME = trunc(random() * 30 + 1);

        _ID_CIDADE = trunc(random() * 9706 + 1);

        _DATA =         (       select  date (('1996-01-01':: date - '45 years'::interval)  +
                                trunc(random() * 365)  *  '1 day':: interval +
                                trunc(random() * 45)  *  '1 year':: interval)           );

        _NOME =         (       select  nome
                                from    nome
                                where   codNome = _ID_NOME                      );

        _CIDADE =       (       select  nome
                                from    cidade
                                where   codCidade = _ID_CIDADE                  );

        IF _MIN > 18 AND _MAX  < 65 THEN

                insert into hospede (nome, cidade, dataNasc) values (_NOME, _CIDADE, _DATA);
                RAISE NOTICE 'NOME: % | CIDADE: % | DATA: %', _NOME, _CIDADE, _DATA;
                _SIM = _SIM + 1;

        ELSE
                RAISE WARNING '     Idade não está entre 18 e 65 anos';
                _NAO = _NAO + 1;

        END IF;

END LOOP;

RAISE INFO '  ----  INSERIDOS: % | NÃO INSERIDOS: %  ----', _SIM, _NAO;

END;
$$ LANGUAGE PLpgSQL;

select fn_insere_hospede (30,10,90);
```

b)

```
CREATE OR REPLACE FUNCTION
fn_insere_atendente(qtd int) RETURNS VOID AS $$

DECLARE

_ID_NOME int;
_NOME varchar(50);

BEGIN

FOR i IN 1..qtd LOOP

        -- seleciona randomicamente um nome da tabela nomes:
        _ID_NOME = trunc(random() * 30 + 1);
        _NOME = (       select  nome
                        from    nome
                        where   codNome = _ID_NOME      );

        insert into atendente (codSuperior, nome) values (1, _NOME);

        RAISE NOTICE 'ATENDENTE: %, incluído com sucesso!', _NOME;

END LOOP;

END;
$$ LANGUAGE PLpgSQL;

select fn_insere_atendente (5);
```

c)

```
CREATE OR REPLACE FUNCTION
fn_insere_hospedagem (qtd int, inicio date, fim date) RETURNS VOID AS $$

DECLARE
_ID_ATENDENTE int;
_ID_HOSPEDE int;
_DATA_INTERVALO int;
_ESTADIA int;
_ENTRADA date;
_SAIDA date;
_QUARTO int;
_VALOR decimal (9,2);
_OCUPADO_1 int;
_OCUPADO_2 int;
_OCUPADO_3 int;

BEGIN

IF fim > inicio THEN

FOR i IN 1..qtd LOOP

        -- existem valores com minimo de 15 atendentes e 50 hospedes em suas respectivas tabelas;
        _ID_ATENDENTE = trunc(random() * 15 + 1);
        _ID_HOSPEDE = trunc(random() * 50 + 1);
        _DATA_INTERVALO = ( select ( fim - inicio ) );
        _ESTADIA = trunc(random() * 14 + 1);
        _ENTRADA = ( select date ( (inicio:: date) + trunc(random()  * _DATA_INTERVALO) * '1 day'::interval) );
        _SAIDA =  _ENTRADA + _ESTADIA;
        _VALOR = random() * 100 + 130;
        _QUARTO = trunc(random() * 10 + 10);

        _OCUPADO_1 = (          select    codHospedagem
                                from      hospedagem
                                where     numQuarto = _QUARTO
                                and       _ENTRADA between dataEntrada and dataSaida limit 1          );
        _OCUPADO_2 = (          select    codHospedagem
                                from      hospedagem
                                where     numQuarto = _QUARTO
                                and       _SAIDA between dataEntrada and dataSaida limit 1            );
        _OCUPADO_3 = (          select    codHospedagem
                                from      hospedagem
                                where     numQuarto = _QUARTO
                                and       dataEntrada between _ENTRADA and _SAIDA limit 1             );

        IF (_OCUPADO_1) IS NULL AND (_OCUPADO_2) IS NULL AND (_OCUPADO_3) IS NULL THEN

                insert into hospedagem (codAtendente, codHospede, dataEntrada, dataSaida, numQuarto, valorDiaria)
                values (_ID_ATENDENTE, _ID_HOSPEDE, _ENTRADA, _SAIDA, _QUARTO, _VALOR);

                RAISE NOTICE 'QUARTO: %, HOSPEDE %, ATENDENTE: %, ENTRADA: %, SAÍDA: %, VALOR: %',
                _QUARTO, _ID_HOSPEDE, _ID_ATENDENTE, _ENTRADA, _SAIDA, _VALOR;

        ELSE
                RAISE WARNING 'QUARTO % OCUPADO ENTRE: % e %', _QUARTO, _ENTRADA, _SAIDA;

        END IF;

END LOOP;

ELSE
        RAISE EXCEPTION 'DATA DE SAÍDA: % ANTERIOR A DATA DE ENTRADA: %', fim, inicio;

END IF;

END;
$$ LANGUAGE PLpgSQL;

select fn_insere_hospedagem (10, '2016-01-01', '2016-01-31');
```

d)

```
CREATE OR REPLACE FUNCTION
fn_atualiza_hospedagem (_COD int, _ID_ATENDENTE int, _SAIDA date, _VALOR decimal (9,2)) RETURNS INT AS $$

DECLARE
_TOTAL int;

BEGIN

        EXECUTE 'UPDATE hospedagem SET '
        || 'codAtendente = '
        || quote_literal(_ID_ATENDENTE)
        || ','
        || 'dataSaida = '
        || quote_literal(_SAIDA)
        || ','
        || 'valorDiaria = '
        || quote_literal(_VALOR)
        || ' WHERE codHospedagem = '
        || quote_literal(_COD);

        -- retorna a quantidade de linhas atualizadas
        GET DIAGNOSTICS _TOTAL = ROW_COUNT;

RETURN _TOTAL;

END;
$$ LANGUAGE PLpgSQL;

select fn_atualiza_hospedagem (750, 11, '2016-02-20', 165.50);
```

2. Descreve-se abaixo as consultas a serem feitas:

CONSULTA 1

Listar:

- nome do hóspede;
- nome do atendente;
- número do quarto onde esse hóspede esteve hospedado;
- valor da hospedagem (quantidade de diárias X valor da diária).

Condições:

- Somente hospedagens já encerradas;
- Hóspedes com 21 anos de idade (no período da hospedagem);
- Cujo data de entrada de hospedagem esteja entre uma das datas de hospedagem de hóspedes que tenham entre 40 e 45 anos de idade.

Ordenar:

- Por valor (descendente) e nome (ascendente).

Limite:

- Retornar somente as primeiras 10 linhas.

```
select    hospede.nome as "Hospede",
          atendente.nome as "Atendente",
          numQuarto as "Quarto",
          ((dataSaida - dataEntrada) * valorDiaria) as "Valor Total"
from      hospedagem, hospede, atendente
where     hospedagem.codHospede = hospede.codHospede
and       hospedagem.codAtendente = atendente.codAtendente
and       now() > dataSaida
and       (extract (year from age (dataEntrada, dataNasc)) = 21)
and       dataEntrada

between

(         select    dataEntrada
          from      hospedagem, hospede
          where     hospedagem.codHospede = hospede.codHospede
          and       extract (year from age (dataEntrada, dataNasc)) between 40 and 45)

and

(         select    dataSaida
          from      hospedagem, hospede
          where     hospedagem.codHospede = hospede.codHospede
          and       extract (year from age (dataEntrada, dataNasc)) between 40 and 45)

order by  "Valor Total" desc, "Hospede" asc

limit 10;
```

CONSULTA 2

Listar:

- Soma dos valores obtidos em diárias (quantidade de dias x valor diária);
- Mês e Ano obtido a partir da data de saída das hospedagens (formato: YYYY/MM);
- Nome do superior dos atendentes relacionados às hospedagens (em maiúsculas).

Condições:

- Somente considerar, para soma, linhas em que a data de saída da hospedagem não tenha ocorrido entre junho e julho de 2011;
- Somente considerar linhas em que a soma dos valores obtidos em diárias seja superior a média dos valores de hospedagens com data de saída nos últimos 10 dias.

Ordenar:

- Mês e ano da data de saída ascendente.

```
*


select    sum(valorDiaria * (dataSaida - dataEntrada)) as "Valor Total",
          to_char(dataSaida, 'MM/YYYY') as "MES/ANO",
          upper(nome) as "Superior"
from      hospedagem, atendente
where     codSuperior = atendente.codAtendente
and       codSuperior = (
                              select distinct     codSuperior
                              from                hospedagem, atendente
                              where               hospedagem.codAtendente = atendente.codAtendente
                         )

group by  dataSaida, nome

except

select              sum(valorDiaria * (dataSaida - dataEntrada)) as "Valor Total",
                    to_char(dataSaida, 'MM/YYYY') as "MES/ANO",
                    upper(nome) as "Superior"
from                hospedagem, atendente
where               codSuperior = atendente.codAtendente
and                 dataSaida between '2011-06-01' and '2011-07-31'

group by  dataSaida, nome

order by  "MES/ANO" asc;
```

CONSULTA 3

Listar:

- Nome do cliente;
- Soma do valor das hospedagens para o cliente (valor diária x qtd dias);
- Com base na soma do valor obtido, liste uma coluna, chamada "Classe" com a seguinte regra:
  - Se a soma estiver entre 0 e 1000 Então 'E'
  - Se a soma estiver entre 1000.01 e 3000.00 Então 'D'
  - Se a soma estiver entre 3000.01 e 7000.00 Então 'C'
  - Se a soma estiver entre 7001.00 e 10000 Então 'B'
  - Se a soma for superir a 10000 Então 10

Condições:

- Somente hospedagens ocorridas no ano de 2010;
- Cuja cidade de origem do hóspede inicie entre A e M;

ou

- Cujo nome do hospede seja igual a FABIO e cuja última hospedagem tenha ocorrido nos últimos 30 dias.

Ordenar:

- Classe e nome

```
select    nome as "Nome",
          CASE
          WHEN "Valor Total" > 0      and  "Valor Total" <= 1000    THEN 'E'
          WHEN "Valor Total" > 1000   and  "Valor Total" <= 3000    THEN 'D'
          WHEN "Valor Total" > 3000   and  "Valor Total" <= 7000    THEN 'C'
          WHEN "Valor Total" > 7000   and  "Valor Total" <= 10000   THEN 'B'
          WHEN "Valor Total" > 10000                                THEN '10'
          END as "Classe"

from
          (
                    select    nome, codHospedagem, hospede.codHospede,
                              sum (valorDiaria * (dataSaida - dataEntrada)) as "Valor Total"
                    from      hospedagem, hospede
                    where     hospedagem.codHospede = hospede.codHospede

                    and       (
                                        extract   (year from dataEntrada ) = 2010
                                        and       (left (upper(cidade),  1) between 'A' and 'M')
                              )
                    or
                              (
                                        upper(nome) = 'FABIO'
                                        and       current_date - dataSaida < 30
                              )

                    group by nome, codHospedagem, hospede.codHospede

          )         as "Resultado"

order by  "Classe", "Nome";
```

CONSULTA 4:

Listar:

- Nome das três cidades mais rentáveis;
- Soma do valor em diárias obtido para essa cidade.

Condições:

- Somente hospedagens ocorridas nos últimos três meses (encerradas ou não) (para hospedagens encerradas considerar a data do dia da consulta para o cálculo do valor das diárias);
- Em caso de empate (quatro ou mais cidades com o mesmo valor), listar o nome de todas. Exemplo:
  - Porto Alegre, 1000.00
  - Canoas, 9000.00
  - São Leopoldo, 8500.00
  - Novo Hamburgo, 8500.00

Ordenar:

- Cidade mais rentável para a menos rentável.

```
select    "Cidade", "Valor Total"
from      (

          select    cidade "Cidade",
                    sum(valorDiaria * (dataSaida - dataEntrada)) "Valor Total",

                    dense_rank()
                    over      ( order by sum(valorDiaria * (dataSaida - dataEntrada)) desc ) "Top 3"

          from      hospedagem h1, hospede h2
          where     h1.codHospede = h2.codHospede
          and       CURRENT_DATE - interval '3 months' < dataEntrada
          group by  "Cidade"

          ) as resultado

where "Top 3" <= 3;
```

CONSULTA 5:

Listar:

- nome do atendente;
- nome do superior do atendente;
- quantidade de atendimentos realizados pelo atendente.

Condições:

- devem ser listados todos os atendentes, mesmo aqueles sem atendimento. Para esses a quantidade de atendimentos deve ser igual a zero;
- somente considerar atendimentos ocorridos nos últimos 30 dias.

```
select      a1.nome "Nome", a2.nome "Superior", count(h1.codAtendente) "QTD"
from        atendente a1

            left join     hospedagem h1
            on            a1.codAtendente = h1.codAtendente
            and           h1.dataEntrada < ( now() - interval '30 DAY' )

                  inner join    atendente a2
                  on            a2.codAtendente = a1.codSuperior
                  group by      "Nome", "Superior"
                  order by      "QTD" desc
```