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;

```
CREATE OR REPLACE FUNCTION FN_INSERE_HOSPEDE(PQUANTIDADEHOSPEDES
integer, PIDADEMINIMAHOSPEDES integer[], PIDADEMAXIMAHOSPEDES integer[])
RETURNS VOID AS $$
DECLARE
   _nomeHospedes char(10)[] := '{"Ana", "Antônia", "Amilton", "Antônio", "Armando"}';
   _sobrenomeHospedes char(10)[] := '{"Mourao", "Vargas", "Silva", "Pereira", "Abreu"}';
   _cidades char(25)[] := '{"Porto Alegre", "Sapiranga", "Sao Paulo", "Cidreira", "Torres"}';
   _nomeEscolhido char(10) := '';
   _sobrenomeEscolhido char(10) := '';
   _cidadeEscolhida char(25) := '';
   _idadeEscolhida int := 0;
   _anoNascimento int := 0;
   _dataNascimento date;
   _nomeCompleto char(20) := '';
BEGIN
        FOR i IN 1..pQuantidadeHospedes LOOP
                IF pIdadeMinimaHospedes[i] < 18 THEN
        RAISE EXCEPTION 'Idade mínima precisa ser maior que 17 anos.';
      END IF;

                IF pIdadeMinimaHospedes[i] > pIdadeMaximaHospedes[i] THEN
                        RAISE EXCEPTION 'Idade minima nao pode ser maior que idade
maxima';
                END IF;

                IF pIdadeMinimaHospedes[i] > 65 THEN
                        RAISE EXCEPTION 'Idade minima nao pode ser maior que 65 anos';
                END IF;

      IF pIdadeMaximaHospedes[i] > 65 THEN
        RAISE EXCEPTION 'Idade máxima precisa ser menor que 65 anos.';
      END IF;

                IF pIdadeMaximaHospedes[i] < pIdadeMinimaHospedes[i] THEN
                        RAISE EXCEPTION 'Idade máxima nao pode ser maior que idade
minima';
                END IF;

                SELECT round(random() * (pIdadeMaximaHospedes[i] -
pIdadeMinimaHospedes[i]) + pIdadeMinimaHospedes[i]) INTO _idadeEscolhida;
```

```
                _anoNascimento := 2023 - _idadeEscolhida;

                _dataNascimento := MAKE_DATE(_anoNascimento, 1, 1) +
                        (RANDOM() * (MAKE_DATE(_anoNascimento, 12, 31) -
MAKE_DATE(_anoNascimento, 1, 1)))::int;

                SELECT _nomeHospedes[ceil(random() * array_length(_nomeHospedes, 1))]
INTO _nomeEscolhido;
                SELECT _sobrenomeHospedes[ceil(random() *
array_length(_sobrenomeHospedes, 1))] INTO _sobrenomeEscolhido;
                SELECT _cidades[ceil(random() * array_length(_cidades, 1))] INTO
_cidadeEscolhida;
                _nomeCompleto := _nomeEscolhido || ' ' || _sobrenomeEscolhido;

                INSERT INTO hospede (nome, cidade, datanascimento) VALUES
(_nomeCompleto, _cidadeEscolhida, _dataNascimento);
                raise notice 'Hospede %, que reside na cidade %, com a data de nascimento
%, foi inserido com sucesso!',
                                        rtrim(_nomeCompleto), rtrim(_cidadeEscolhida),
_dataNascimento;
        END LOOP;
END;

$$ LANGUAGE PLPGSQL;
```

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

```
CREATE OR REPLACE FUNCTION
FN_INSERE_ATENDENTE(PQUANTIDADEATENDENTES integer) RETURNS VOID AS $$

declare

_nomeAtendentes char(10)[] := '{"Bruna", "Gabriel", "Eduardo", "Marcos", "Rafael"}';
_sobrenomeAtendentes char(10)[] := '{"Santos", "Medeiros", "Cunha", "Weber", "Borges"}';
_nomeEscolhido char(10) := '';
_sobrenomeEscolhido char(10) := '';
_nomeCompleto char(20) := '';
_idSuperior integer := 1;

begin
        IF pQuantidadeAtendentes IS NOT NULL THEN
                FOR i IN 1..pQuantidadeAtendentes LOOP
                        SELECT _nomeAtendentes[ceil(random() *
array_length(_nomeAtendentes, 1))] INTO _nomeEscolhido;
                        SELECT _sobrenomeAtendentes[ceil(random() *
array_length(_sobrenomeAtendentes, 1))] INTO _sobrenomeEscolhido;
```

```
                    _nomeCompleto := _nomeEscolhido || ' ' || _sobrenomeEscolhido;

                    insert into atendente(codsuperior, nome) values (_idSuperior,
_nomeCompleto);
                END LOOP;
        ELSE
                RAISE EXCEPTION 'Quantidade de atendentes nao pode ser vazio';
        END IF;

end;

$$ LANGUAGE PLPGSQL;
```

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

```
CREATE OR REPLACE FUNCTION
FN_INSERE_HOSPEDAGEM(PQUANTIDADEHOSPEDAGEM integer, PDATAENTRADA
date, PDATASAIDA date) RETURNS VOID AS $$

declare
_atendente atendente%rowtype;
_hospede hospede%rowtype;
_diferencaDias integer := 0;
_dataAleatoriaEntrada date;
_dataAleatoriaSaida date;
_numeroQuarto integer := 0;
_quartoResultado integer;

begin
        loop
                if pDataEntrada = pDataSaida then
                        raise exception 'Data entrada nao pode ser igual a data saida';
                end if;

                select pDataSaida - pDataEntrada into _diferencaDias;
```

```
		if _diferencaDias < 0 then
			raise exception 'Data saida nao pode ser menor que data de entrada';
		end if;

		if _diferencaDias = 1 then
			_dataAleatoriaEntrada := pDataEntrada;
			_dataAleatoriaSaida := pDataSaida;
		else
			loop
				SELECT
					(DATE_TRUNC('day', start_date) + (random() *
(DATE_TRUNC('day', end_date) - DATE_TRUNC('day', start_date))))::date into
_dataAleatoriaEntrada
				FROM
					(SELECT
						pDataEntrada AS start_date,
						pDataSaida AS end_date
					) AS dates;

				SELECT
					(DATE_TRUNC('day', start_date) + (random() *
(DATE_TRUNC('day', end_date) - DATE_TRUNC('day', start_date))))::date into
_dataAleatoriaSaida
				FROM
					(SELECT
						_dataAleatoriaEntrada AS start_date,
						pDataSaida AS end_date
					) AS dates;

				if _dataAleatoriaEntrada != _dataAleatoriaSaida then
					exit;
				end if;

			end loop;
		end if;

		select * into _atendente from atendente order by random() limit 1;
		select * into _hospede from hospede order by random() limit 1;
		select floor(random() * 100) + 1 into _numeroQuarto;

		select COALESCE(numquarto, 0) into _quartoResultado from hospedagem
		where numquarto = _numeroQuarto
		and dataentrada between _dataAleatoriaEntrada and _dataAleatoriaSaida
		and datasaida = between _dataAleatoriaEntrada and _dataAleatoriaSaida

		if not found _quartoResultado then
```

```
                        insert into hospedagem (codatendente, codhospede, dataentrada,
datasaida, numquarto, valordiaria)
                        values (_atendente.codatendente, _hospede.codhospede,
_dataAleatoriaEntrada, _dataAleatoriaSaida, _numeroQuarto, 150);

                        raise notice 'Hospedagem inserida com sucesso!';

                        exit;
                end if;
        end loop;
end;

$$ LANGUAGE PLPGSQL;
```

[Consulta 1]
Escreva uma consulta que atenda ao abaixo solicitado:
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 (com data saída preenchida, portanto)
E
hóspedes com 21 anos de idade (no período da hospedagem)
E
cujo data de entrada de hospedagem esteja dentre uma das datas de hospedagem de hóspedes que tenham entre 40 e 45 anos de idade.
Ordem:
Ordenar por valor (descendente) e nome (ascendente)
Limite de dados retornados:
Retornar somente as primeiras 10 linhas.

```
SELECT HSP.NOME as "Nome hospede",
        ATD.NOME as "Nome atendente",
        HPDG.NUMQUARTO as "Numero do quarto",
        (HPDG.DATASAIDA - HPDG.DATAENTRADA) * HPDG.VALORDIARIA AS "Valor da
hospedagem"
FROM HOSPEDAGEM HPDG
INNER JOIN HOSPEDE HSP ON HSP.CODHOSPEDE = HPDG.CODHOSPEDE
INNER JOIN ATENDENTE ATD ON ATD.CODATENDENTE = HPDG.CODATENDENTE
WHERE HPDG.DATASAIDA IS NOT NULL
        AND HSP.DATANASCIMENTO <= (CURRENT_DATE - interval '21 years')
        AND EXISTS
                (SELECT *
                FROM HOSPEDAGEM H
```

```
            INNER JOIN HOSPEDE HP ON H.CODHOSPEDE = HP.CODHOSPEDE
            WHERE HP.DATANASCIMENTO BETWEEN (CURRENT_DATE - INTERVAL
'45 years') AND (CURRENT_DATE - INTERVAL '40 years')
                    AND H.DATAENTRADA = HPDG.DATAENTRADA )
ORDER BY
    "Valor da hospedagem" desc,
    "Nome hospede" asc
LIMIT 10;
```

[Consulta 2]
Escreva uma consulta que atenda ao abaixo solicitado:
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: YYYYMM)
- 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
E
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 por Mês e ano da data de saída ascendente

```
SELECT SUM((H.DATASAIDA - H.DATAENTRADA) * H.VALORDIARIA) AS "Soma dos
valores obtidos em diárias",
    TO_CHAR(H.DATASAIDA,'MM-YYYY') AS "Mes e ano",
    UPPER(A2.NOME) AS "Nome do superior dos atendentes"
FROM HOSPEDAGEM H
JOIN ATENDENTE A ON H.CODATENDENTE = A.CODATENDENTE
JOIN ATENDENTE A2 ON A.CODSUPERIOR = A2.CODATENDENTE
WHERE H.DATASAIDA NOT BETWEEN '2011-06-01' AND '2011-07-31'
GROUP BY TO_CHAR(H.DATASAIDA, 'MM-YYYY'), A2.NOME
HAVING SUM((H.DATASAIDA - H.DATAENTRADA) * H.VALORDIARIA) >
    (SELECT AVG((H2.DATASAIDA - H2.DATAENTRADA) * H2.VALORDIARIA)
        FROM HOSPEDAGEM H2
    WHERE H2.DATASAIDA BETWEEN (CURRENT_DATE - INTERVAL '10 days') AND
CURRENT_DATE )
ORDER BY "Mes e ano" ASC;
```

[Consulta 5]
Escreva uma consulta que liste:
- nome do atendente.
- nome do superior do atendente.
- quantidade de atendimentos realizados pelo atendente.
Critérios:
- 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 A.NOME AS "Nome do atendente",
        COALESCE(S.NOME, 'Nenhum') AS "Nome do superior do atendente",
        COUNT(H.CODHOSPEDAGEM) AS "Quantidade de atendimentos realizados"
FROM ATENDENTE A
LEFT JOIN ATENDENTE S ON A.CODSUPERIOR = S.CODATENDENTE
LEFT JOIN HOSPEDAGEM H ON A.CODATENDENTE = H.CODATENDENTE
AND H.DATAENTRADA BETWEEN (CURRENT_DATE - INTERVAL '30 days') AND
CURRENT_DATE
GROUP BY A.NOME,
        S.NOME
ORDER BY "Nome do atendente" ASC;
```