Faculdade de Informática e Administração Paulista

LISTA DE EXERCÍCIOS

SQL

STRUCTURED QUERY LANGUAGE

PROFESSOR MESTRE ALEXANDRE BARCELOS

Índice

Exercícios

Índice

Soluções

I. Introdução

1. Como é chamada a linguagem utilizada para a manipulação de dados em um banco de dados relacional padronizada pela ANSI – American National Standards Institute

2. Responda (V) para a sentença Verdadeira e (F) para a sentença Falso

   ( ) Em um banco de dados relacional, você especifica a rota de acesso às tabelas e precisa saber como os dados estão organizados fisicamente.

   ( ) A linguagem SQL contém um amplo conjunto de operadores para particionar e combinar relações.

   ( ) É possível usar instruções SQL para modificar o banco de dados.

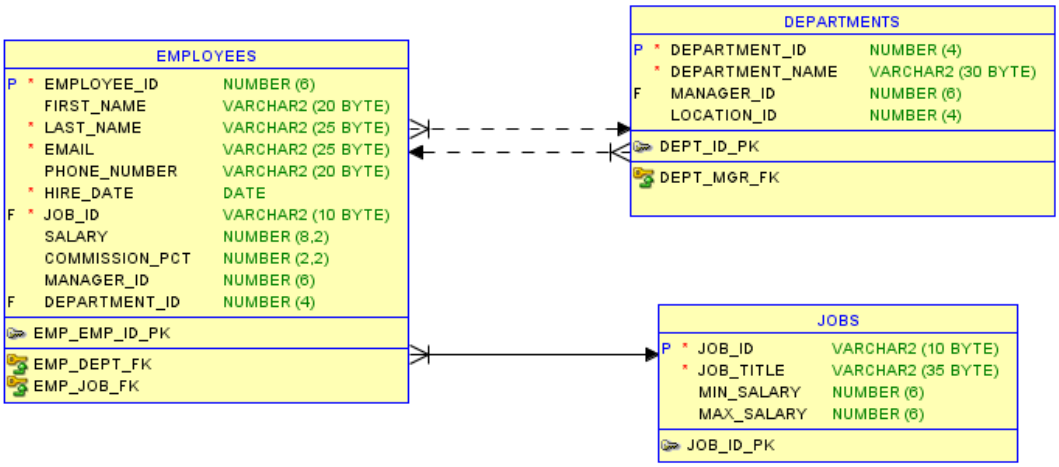
3. A SQL pode ser dividida em 5 sub-linguagens a saber:
   DML – Data Manipulation Language
   DDL – Data Definition Language
   TCL – Trasaction Control Language
   DCL – Data Control Language
   DQL – Data Query Language

   Informe a que grupo as instruções SQL a seguir pertencem

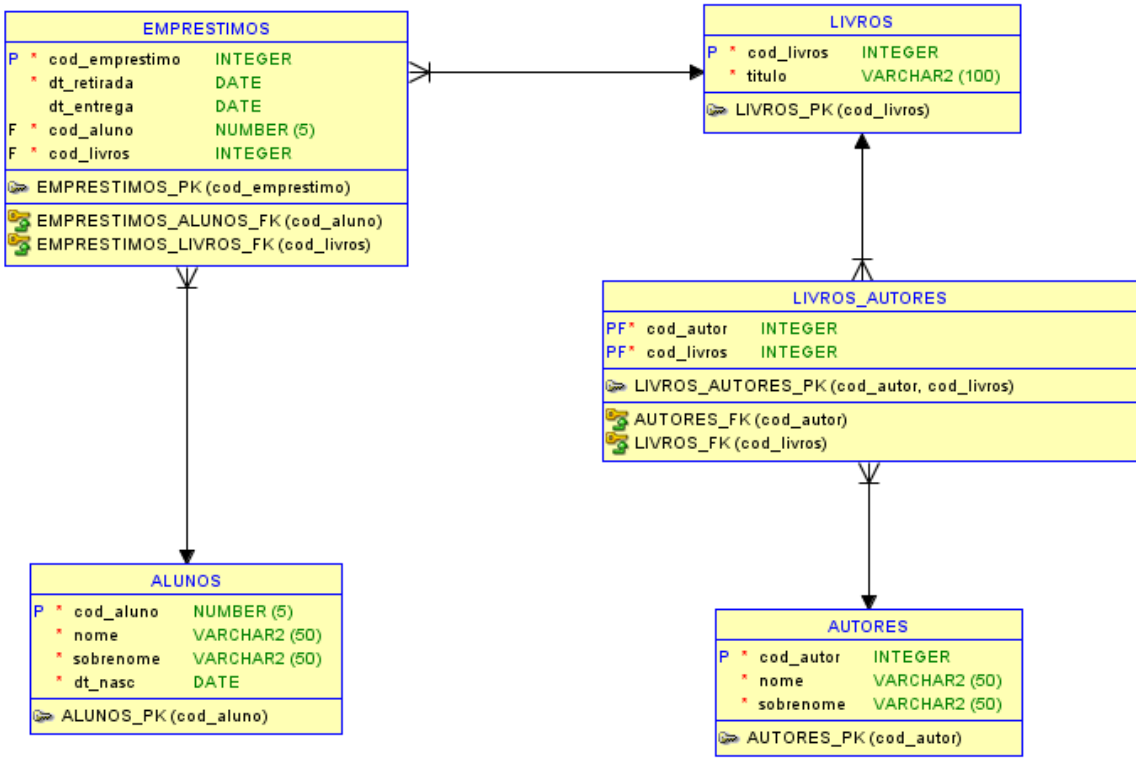
   ( ) GRANT
   ( ) ALTER
   ( ) INSERT
   ( ) DROP
   ( ) SELECT
   ( ) TRUNCATE
   ( ) UPDATE
   ( ) SAVEPOINT
   ( ) DELETE
   ( ) ROLLBACK
   ( ) MERGE
   ( ) CREATE
   ( ) COMMIT
   ( ) COMMENT
   ( ) REVOKE

II. Instruções DDL-Data Definition Language

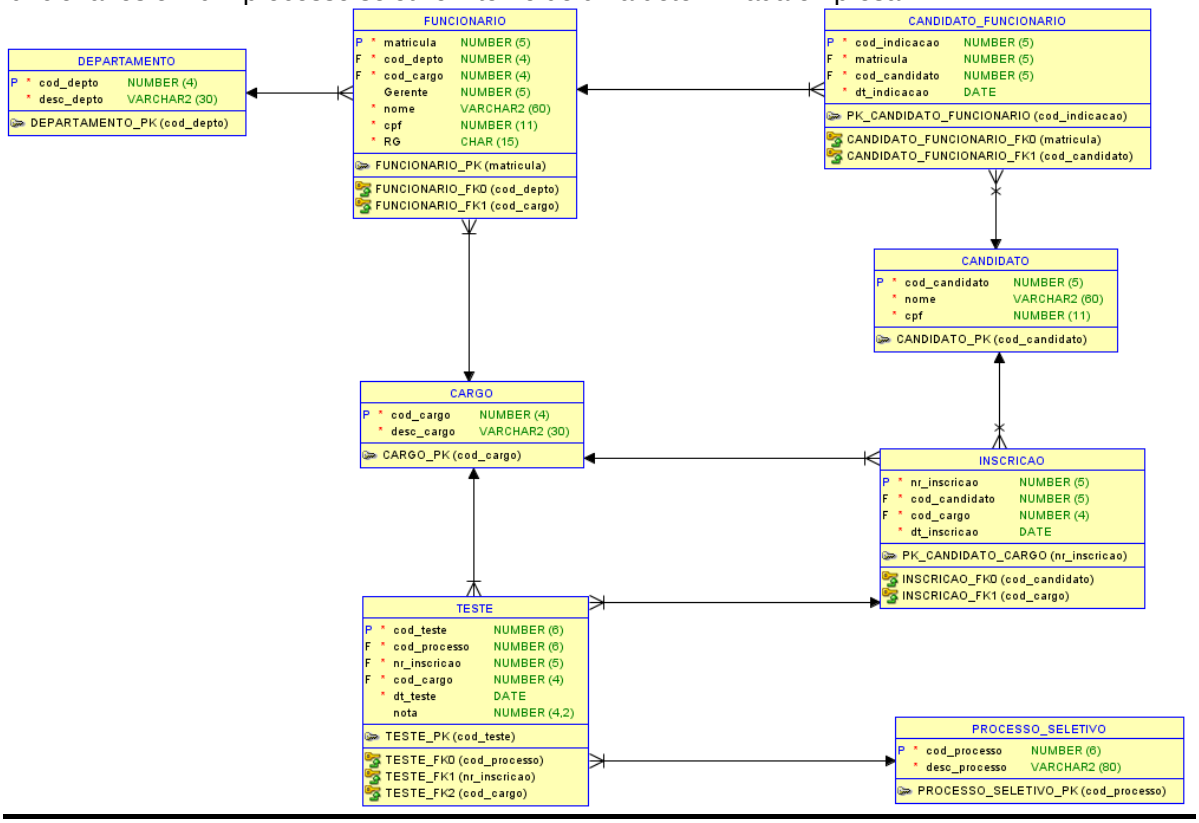
1. Dado o modelo relacional a seguir que representa a atribuição de um cargo para os funcionários e a alocação desses funcionários em um determinado departamento, implemente o modelo físico no Oracle RDBMS.
   Implemente a restrição chamada EMP_SALARY_MIN na coluna SALARY da tabela EMPLOYEES. Essa restrição deve garantir que somente salários positivos e diferentes de zero sejam inseridos. Opcionalmente nomeia as restrições NOT NULL.



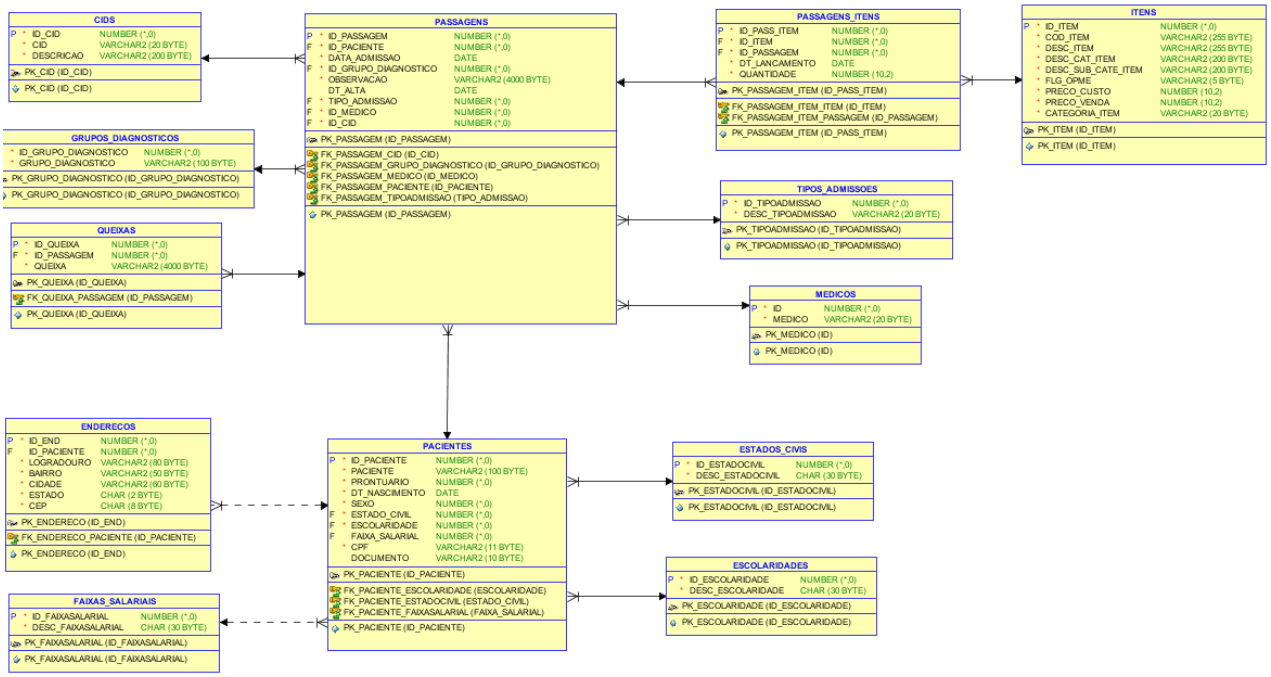
2. Dado o modelo relacional a seguir que representa, de forma simplificada, uma solução de empréstimos de livros para os alunos de uma determinada escola, implemente o modelo físico no Oracle RDBMS.



3. Implemente o modelo físico, conforme o modelo relacional a seguir, que permite a candidatura dos funcionários em um processo seletivo interno de uma determinada empresa.

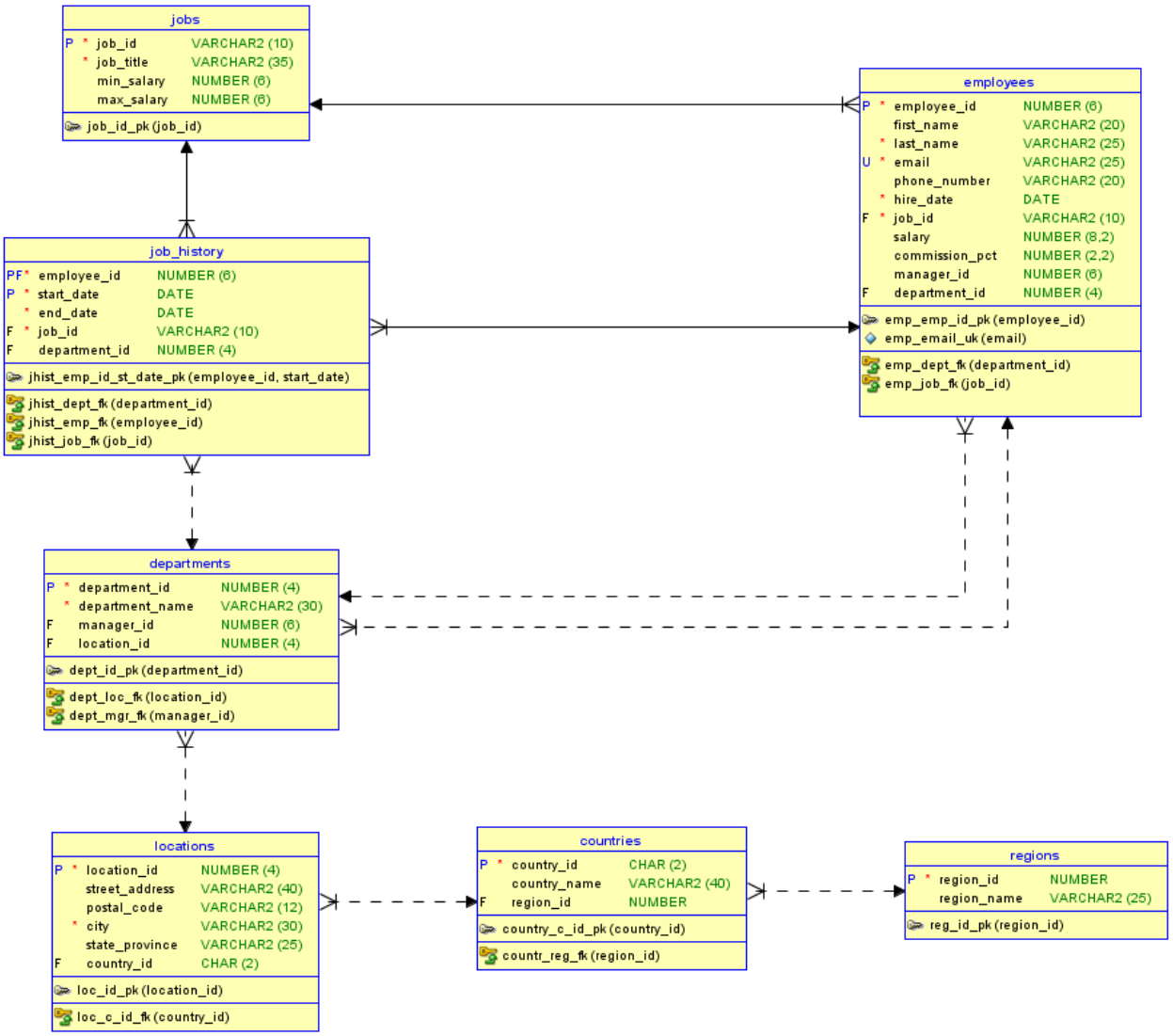


4. Implemente o modelo físico que representa, de forma sucinta, a admissão de pacientes de um determinado hospital.

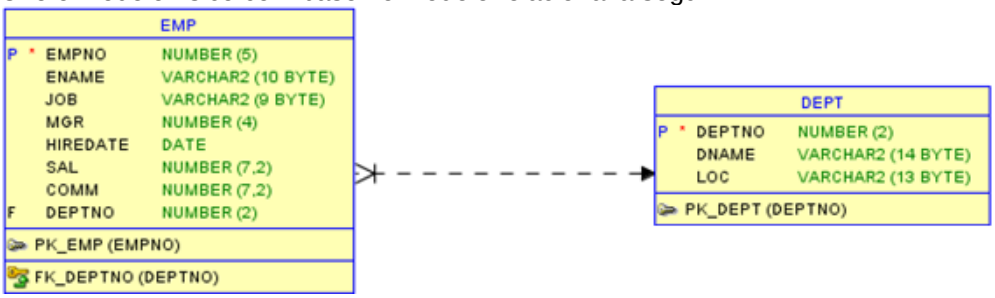


5. Crie o modelo físico de banco de dados conforme o modelo relacional a seguir. O modelo é baseados em uma empresa de amostra fictícia que vende produtos por meio de vários canais. A empresa opera em todo o mundo para atender pedidos de produtos e possui várias divisões. Esse diagrama representa a divisão de recursos humanos e rastreia informações sobre os funcionários e instalações da empresa.
Observações:

- Esse modelo será utilizado em todas as nossas aulas para as práticas de SQL.
- Esse modelo foi criado pela Oracle e é disponibilizado em todas as versões do Oracle RDBMS e tem caráter meramente educacional.



6. Execute as seguintes tarefas a seguir:
   a. Crie o modelo físico com base no modele relacional a seguir:



   b. Exiba a estrutura da tabela EMP.

   c. Exiba a estrutura da tabela DEPT.

   d. Renomeie a tabela EMP para EMPS.

   e. Renomeie a tabela DEPT para DEPTS.

   f. Renomeie a restrição PK_EMP da tabela EMPS para PK_EMPS.

   g. Renomeie a restrição FK_DEPTNO da tabela EMPS para FK_DEPTSNO.

   h. Renomeie a restrição PK_DEPT da tabela DETPS para PK_DEPTS.

   i. Renomeie a coluna SAL da tabela EMPS para SALARY.

   j. Remova a coluna LOC da tabela DEPTS.

   k. Adicione a coluna BIRTH_DATE na tela EMPS.

   l. Altere o tamanho da coluna SALARY da tabela EMPS para 8 de precisão e 3 de escala.

   m. Implemente a restrição necessária para que as colunas ENAME e SALARY da tabela EMPS sejam obrigatórias.

   n. Implemente a restrição necessária para que as colunas DNAME da tabela DEPTS seja obrigatória e passe a aceitar até 20 caracteres.

   o. Exiba a estrutura da tabela EMPS.

   p. Exiba a estrutura da tabela DEPTS.

III. Instruções DML-Data Manipulation Language

O departamento de recursos humanos deseja criar instruções SQL para inserir, atualizar e deletar dados de funcionários. Como protótipo, use a tabela MY_EMPLOYEE antes de fornecer as instruções ao departamento de recursos humanos.

1. Execute a instrução a seguir para criar a tabela MY_EMPLOYEE a ser usada no exercício:

```
CREATE TABLE my_employee
  (id  NUMBER(4) CONSTRAINT my_employee_id_nn NOT NULL,
   last_name  VARCHAR(25),
   first_name VARCHAR(25),
   userid     VARCHAR(8),
   salary     NUMBER(9,2));
```

Insira dados na tabela MY_EMPLOYEE.

2. Descreva a estrutura da tabela MY_EMPLOYEE para identificar os nomes de colunas.

3. Crie uma instrução INSERT para adicionar a ***primeira linha*** de dados à tabela MY_EMPLOYEE usando estes dados de amostra. Não liste as colunas na cláusula INSERT.
   Não informe todas as linhas ainda.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

4. Preencha a tabela MY_EMPLOYEE com a segunda linha de dados de amostra da lista anterior. Desta vez, liste as colunas explicitamente na cláusula INSERT.

5. Confirme a adição à tabela. Faça uma consulta que realize a projeção dos dados

6. Preencha a tabela MY_EMPLOYEE com a terceira linha de dados de amostra da lista anterior.

7. Preencha a tabela MY_EMPLOYEE com a quarta linha de dados de amostra da lista anterior.

8. Confirme as adições à tabela. Faça uma consulta que realize a projeção dos dados

9. Torne as adições de dados permanentes. Encerre a transação para que os dados sejam gravados no banco de dados.

Atualize e delete dados na tabela MY_EMPLOYEE.

10. Altere o sobrenome do funcionário 3 para Drexler.

11. Altere o salário de todos os funcionários com salário inferior a US$ 900 para US$ 1.000.

12. Verifique as alterações na tabela. Faça uma consulta que realize a projeção dos dados

13. Delete Betty Dancs da tabela MY_EMPLOYEE.

14. Confirme as alterações na tabela. Faça uma consulta que realize a projeção dos dados

15. Encerre a transação para que os dados sejam gravados no banco de dados.

IV. Instruções DQL-Data Query Language

Parte 1

Teste os seus conhecimentos.

1. Inicie uma sessão do SQL Developer com o ID de usuário e a senha.

2. Os comandos SQL Developer acessam o banco de dados. Essa sentença é Verdadeira ou Falsa?

3. A instrução SELECT é executada com êxito?

```
SELECT        last_name,        job_id,        salary        AS        Sal
FROM   employees;
```

Considere a estrutura da tabela EMPLOYEES como sendo:

```
Nome           Nulo?    Tipo
-------------- -------- ------------
EMPLOYEE_ID    NOT NULL NUMBER(6)
FIRST_NAME              VARCHAR2(20)
LAST_NAME      NOT NULL VARCHAR2(25)
EMAIL          NOT NULL VARCHAR2(25)
PHONE_NUMBER            VARCHAR2(20)
HIRE_DATE      NOT NULL DATE
JOB_ID         NOT NULL VARCHAR2(10)
SALARY                  NUMBER(8,2)
COMMISSION_PCT          NUMBER(2,2)
MANAGER_ID              NUMBER(6)
DEPARTMENT_ID           NUMBER(4)
```

4. A instrução a seguir é executada corretamente? Considere que a tabela JOB_GRADES existe:

```
SELECT * FROM job_grades;
```

5. Há quatro erros de codificação na instrução a seguir. Você consegue identificá-los? Reescreva a instrução corretamente. Utilize a estrutura apresentaa na questão 3.

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

Parte 2

Você foi admitido como programador SQL da Acme Corporation. Sua primeira tarefa é criar alguns relatórios com base nos dados das tabelas de recursos humanos.

6. Sua primeira tarefa é determinar a estrutura da tabela DEPARTMENTS e projetar seu conteúdo.

7. Você precisa determinar a estrutura da tabela EMPLOYEES.

   O departamento de recursos humanos deseja executar uma consulta para exibir o sobrenome, o código do cargo, a data de admissão e o telefone de cada funcionário, com o número do funcionário exibido primeiro. Forneça o apelido STARTDATE para a coluna HIRE_DATE.

8. O departamento de recursos humanos precisa de uma consulta para exibir todos os códigos de cargo exclusivos da tabela EMPLOYEES.

Parte 3

9. O departamento de recursos humanos deseja cabeçalhos de coluna mais descritivos em seu relatório sobre funcionários. Pegue a instrução executada no exercício 6 e faça as seguintes alterações. Nomeie os cabeçalhos de coluna como Emp #, Employee, Job e Hire Date, respectivamente.
   Execute a consulta novamente.

10. O departamento de recursos humanos solicitou um relatório de todos os funcionários e os respectivos IDs de cargo. Exiba o sobrenome concatenado com o ID do cargo (separado por uma vírgula e um espaço) e nomeie a coluna como Employee and Title.

V. Restringindo e Classificando Dados

O departamento de recursos humanos precisa da sua ajuda para criar algumas consultas.

1. Em função de questões orçamentárias, o departamento precisa de um relatório com o sobrenome e o salário dos funcionários que ganham mais de US$ 12.000.

2. Crie um relatório que exiba o sobrenome e o número do departamento do funcionário 176.

3. O departamento de recursos humanos precisa localizar funcionários com altos e baixos salários. Modifique o exercício 1 para exibir o sobrenome e o salário de todos os funcionários cuja faixa salarial não esteja entre US$ 5.000 e US$ 12.000.

4. Crie um relatório para exibir o sobrenome, o ID do cargo e a data de admissão dos funcionários cujos sobrenomes sejam Matos e Taylor. Organize a consulta em ordem crescente por data de admissão.

5. Exiba o sobrenome e o número do departamento de todos os funcionários nos departamentos 20 e 50 em ordem alfabética crescente por nome.

6. Modifique o exercício 3 para exibir o sobrenome e o salário dos funcionários que ganham entre US$ 5.000 e US$ 12.000 e estão no departamento 20 ou 50. Atribua às colunas os labels Employee e Monthly Salary, respectivamente.

7. O departamento de recursos humanos precisa de um relatório que exiba o sobrenome e a data de admissão de todos os funcionários admitidos em 1994.

8. Crie um relatório que exiba o sobrenome e o cargo de todos os funcionários não subordinados a um gerente.

9. Crie um relatório para exibir o sobrenome, o salário e a comissão de todos os funcionários que ganham comissão. Classifique os dados em ordem decrescente de salário e comissões.

10. Exiba todos os sobrenomes dos funcionários cuja terceira letra do nome seja *a*.

11. Exiba o sobrenome de todos os funcionários que contenha *a* e *e*.

12. Exiba o sobrenome, o cargo e o salário de todos os funcionários cujo cargo seja representante de vendas ou estoquista e cujo salário seja diferente de US$ 2.500, US$ 3.500 ou US$ 7.000.

13. Modifique o exercício 6 para exibir o sobrenome, o salário e a comissão de todos os funcionários cuja comissão seja de 20%.

VI. Exibindo Dados de Várias Tabelas

1. Crie uma consulta para o departamento de recursos humanos a fim de gerar os endereços de todos os departamentos. Use as tabelas LOCATIONS e COUNTRIES. Mostre o ID do local, o endereço, a cidade, o estado e o país na saída.

2. O departamento de recursos humanos precisa de um relatório de todos os funcionários. Crie uma consulta para exibir o sobrenome, o número do departamento e o nome do departamento de todos os funcionários.

3. O departamento de recursos humanos precisa de um relatório dos funcionários em Toronto. Exiba o sobrenome, o cargo, o número do departamento e o nome do departamento de todos os funcionários que trabalham em Toronto.

4. Crie um relatório para o departamento de recursos humanos que exiba os sobrenomes e os números de departamento dos funcionários, bem como todos os funcionários que trabalham no mesmo departamento como um funcionário específico. Atribua um label apropriado a cada coluna. Salve o script no arquivo lab_06_04.sql.

5. O departamento de recursos humanos deseja determinar os nomes de todos os funcionários admitidos após Davies. Crie uma consulta para exibir o nome e a data de admissão de todos os funcionários admitidos após Davies.

6. O departamento de recursos humanos precisa obter os nomes e as datas de admissão de todos os funcionários admitidos antes dos respectivos gerentes, além dos nomes e das datas de admissão desses gerentes. Salve o script no arquivo lab_06_06.sql.

VII. Funções de Grupo

Determine a validade das três instruções a seguir. Circule Verdadeiro ou Falso.

1. As functions de grupo trabalham com várias linhas para produzir um resultado por grupo.

   Verdadeiro/Falso

2. As functions de grupo incluem valores nulos em cálculos.

   Verdadeiro/Falso

3. A cláusula WHERE restringe as linhas em um cálculo de grupo.

   Verdadeiro/Falso

O departamento de RH necessita dos seguintes relatórios:

4. Obtenha o salário máximo, o salário-mínimo, a soma dos salários e o salário médio de todos os funcionários. Atribua os labels Maximum, Minimum, Sum e Average, respectivamente, às colunas. Arredonde os resultados para o número inteiro mais próximo. Inclua a instrução SQL no arquivo de texto lab_07_04.sql.

5. Modifique a consulta em lab_07_04.sql para exibir o salário-mínimo, o salário máximo, a soma dos salários e o salário médio de cada tipo de cargo. Salve novamente lab_04_04.sql como lab_07_05.sql. Execute a instrução em lab_07_05.sql.

6. Crie uma consulta para exibir o número de pessoas com o mesmo cargo.

   Generalize a consulta para que o usuário do departamento de RH seja solicitado a informar um cargo. Salve o script no arquivo lab_07_06.sql.

7. Determine o número de gerentes sem listá-los. Atribua o label Number of Managers à coluna. *Dica: Use a coluna MANAGER_ID para determiner o número de gerentes*.

8. Descubra a diferença entre o salário mais alto e o mais baixo. Atribua o label DIFFERENCE à coluna.

9. Crie um relatório para exibir o número do gerente e o salário do funcionário com menor remuneração desse gerente. Exclua todas as pessoas cujo gerente seja desconhecido. Exclua todos os grupos em que o salário-mínimo seja US$ 6.000 ou inferior. Classifique a saída em ordem decrescente de salário.

10. Crie uma consulta que exiba o número total de funcionários e, desse total, mostre o número de funcionários admitidos em 1995, 1996, 1997 e 1998. Crie cabeçalhos de colunas apropriados.

11. Crie uma consulta matriz que exiba o cargo, o salário relativo a esse cargo com base no número do departamento e o salário total desse cargo para os departamentos 20, 50, 80 e 90, atribuindo um cabeçalho apropriado a cada coluna

VIII. Subconsultas

1. Crie um relatório que exiba o número e o sobrenome de todos os funcionários cujo salário é maior que o salário médio. Classifique os resultados em ordem crescente de salário.

2. Crie uma consulta que exiba o número e o sobrenome de todos os funcionários que trabalham em um departamento com funcionários cujos sobrenomes contêm a letra *u*. Inclua a instrução SQL no arquivo de texto lab_08_02.sql. Execute a consulta.

3. O departamento de recursos humanos precisa de um relatório que exiba o sobrenome, o número do departamento e o ID do cargo de todos os funcionários cujo ID de local do departamento é 1700.

4. Crie um relatório para o departamento de recursos humanos que exiba o sobrenome e o salário de todos os funcionários subordinados a Steven King.

5. Crie um relatório para o departamento de recursos humanos que exiba o número do departamento, o sobrenome e o ID do cargo de todos os funcionários no departamento executivo.

6. Modifique a consulta em lab_08_02.sql para exibir o número, o sobrenome, bem como o salário de todos os funcionários que ganham mais que o salário médio e trabalham em um departamento com funcionários cujos sobrenomes contêm a letra *u*. Salve novamente lab_06_03.sql como lab_08_06.sql. Execute a instrução em lab_08_06.sql.

Lista de Respostas

I. Introdução

1. SQL – Structured Query Language

2. Responda (V) para a sentença Verdadeira e (F) para a sentença Falso

   (F)

   (V)

   (V)

3.

```
( DCL )     GRANT
( DDL )     ALTER
( DML )     INSERT
( DDL )     DROP
( DQL )     SELECT
( DDL )     TRUNCATE
( DML )     UPDATE
( TCL )     SAVEPOINT
( DML )     DELETE
( TCL )     ROLLBACK
( DML )     MERGE
( DDL )     CREATE
( TCL )     COMMIT
( DDL )     COMMENT
( DCL )     REVOKE
```

II. Instruções DDL-Data Definition Language

1. Dado o modelo relacional a seguir que representa a atribuição de um cargo para os funcionários e a alocação desses funcionários em um determinado departamento, implemente o modelo físico no Oracle RDBMS.
   Implemente a restrição chamada EMP_SALARY_MIN na coluna SALARY da tabela EMPLOYEES. Essa restrição deve garantir que somente salários positivos e diferentes de zero sejam inseridos. Opcionalmente nomeia as restrições NOT NULL.

```
CREATE TABLE departments (
    department_id    NUMBER(4)   CONSTRAINT dept_id_pk   PRIMARY KEY ,
    department_name  VARCHAR(30) CONSTRAINT dept_name_nn NOT NULL,
    manager_id       NUMBER(6),
    location_id      NUMBER(4)
);

CREATE TABLE employees (
    employee_id     NUMBER(6)    CONSTRAINT emp_emp_id_pk    PRIMARY KEY,
    first_name      VARCHAR(20),
    last_name       VARCHAR(25)  CONSTRAINT emp_last_name_nn NOT NULL,
    email           VARCHAR(25)  CONSTRAINT emp_email_nn     NOT NULL,
    phone_number    VARCHAR(20),
    hire_date       DATE         CONSTRAINT emp_hire_date_nn NOT NULL,
    job_id          VARCHAR(10)  CONSTRAINT emp_job_nn       NOT NULL,
    salary          NUMBER(8, 2) CONSTRAINT emp_salary_min   CHECK (
salary > 0 ),
    commission_pct  NUMBER(2, 2),
    manager_id      NUMBER(6),
    department_id   NUMBER(4)
);

CREATE TABLE jobs (
    job_id      VARCHAR(10) CONSTRAINT job_id_pk    PRIMARY KEY,
    job_title   VARCHAR(35) CONSTRAINT job_title_nn NOT NULL,
    min_salary  NUMBER(6),
    max_salary  NUMBER(6)
);


ALTER TABLE departments
    ADD CONSTRAINT dept_mgr_fk FOREIGN KEY ( manager_id )
        REFERENCES employees ( employee_id );

ALTER TABLE employees
    ADD CONSTRAINT emp_dept_fk FOREIGN KEY ( department_id )
        REFERENCES departments ( department_id );

ALTER TABLE employees
    ADD CONSTRAINT emp_job_fk FOREIGN KEY ( job_id )
        REFERENCES jobs ( job_id );
```

2. Dado o modelo relacional a seguir que representa, de forma simplificada, uma solução de empréstimos de livros para os alunos de uma determinada escola, implemente o modelo físico no Oracle RDBMS.

```
CREATE TABLE alunos (
    cod_aluno  NUMBER(5)    CONSTRAINT alunos_pk PRIMARY KEY ,
    nome       VARCHAR(50) NOT NULL,
    sobrenome  VARCHAR(50) NOT NULL,
    dt_nasc    DATE NOT NULL
);

CREATE TABLE autores (
    cod_autor  INTEGER      CONSTRAINT autores_pk PRIMARY KEY,
    nome       VARCHAR(50) NOT NULL,
    sobrenome  VARCHAR(50) NOT NULL
);


CREATE TABLE emprestimos (
    cod_emprestimo  INTEGER   CONSTRAINT emprestimos_pk PRIMARY KEY ,
    dt_retirada     DATE      NOT NULL,
    dt_entrega      DATE,
    cod_aluno       NUMBER(5) NOT NULL,
    cod_livros      INTEGER   NOT NULL
);

ALTER TABLE emprestimos ADD ( cod_emprestimo );

CREATE TABLE livros (
    cod_livros  INTEGER      CONSTRAINT livros_pk PRIMARY KEY,
    titulo      VARCHAR(100) NOT NULL
);


CREATE TABLE livros_autores (
    cod_autor   INTEGER NOT NULL,
    cod_livros  INTEGER NOT NULL,
      CONSTRAINT livros_autores_pk PRIMARY KEY ( cod_autor,cod_livros )
);


ALTER TABLE livros_autores
    ADD CONSTRAINT autores_fk FOREIGN KEY ( cod_autor )
        REFERENCES autores ( cod_autor );

ALTER TABLE emprestimos
    ADD CONSTRAINT emprestimos_alunos_fk FOREIGN KEY ( cod_aluno )
        REFERENCES alunos ( cod_aluno );

ALTER TABLE emprestimos
    ADD CONSTRAINT emprestimos_livros_fk FOREIGN KEY ( cod_livros )
        REFERENCES livros ( cod_livros );

ALTER TABLE livros_autores
    ADD CONSTRAINT livros_fk FOREIGN KEY ( cod_livros )
        REFERENCES livros ( cod_livros );
```

3. Implemente o modelo físico, conforme o modelo relacional a seguir, que permite a candidatura dos funcionários em um processo seletivo interno de uma determinada empresa.

```
CREATE TABLE CANDIDATO
    (
     cod_candidato  NUMBER(5)   CONSTRAINT CANDIDATO_PK PRIMARY KEY,
     nome           VARCHAR(60) NOT NULL ,
     cpf            NUMBER (11) NOT NULL
    ) ;

CREATE TABLE CARGO
    (
     cod_cargo  NUMBER(4)    CONSTRAINT CARGO_PK PRIMARY KEY,
     desc_cargo VARCHAR(30) NOT NULL
    ) ;

CREATE TABLE FUNCIONARIO
    (
     matricula  NUMBER (5)    CONSTRAINT FUNCIONARIO_PK PRIMARY KEY ,
     cod_depto  NUMBER (4)    NOT NULL ,
     cod_cargo  NUMBER (4)    NOT NULL ,
     Gerente    NUMBER (5)        NULL ,
     nome       VARCHAR (60) NOT NULL ,
     cpf        NUMBER (11)   NOT NULL ,
     RG         CHAR (15)     NOT NULL
    ) ;

CREATE TABLE DEPARTAMENTO
    (
     cod_depto  NUMBER (4)    CONSTRAINT DEPARTAMENTO_PK PRIMARY KEY,
     desc_depto VARCHAR (30) NOT NULL
    ) ;

CREATE TABLE TESTE
    (
     cod_teste NUMBER (6)  CONSTRAINT TESTE_PK PRIMARY KEY,
     cod_processo NUMBER (6)  NOT NULL ,
     nr_inscricao NUMBER (5)  NOT NULL ,
     cod_cargo NUMBER (4)  NOT NULL ,
     dt_teste DATE  NOT NULL ,
     nota NUMBER (4,2)
    ) ;

CREATE TABLE PROCESSO_SELETIVO
    (
     cod_processo   NUMBER (6)  CONSTRAINT PROCESSO_SELETIVO_PK PRIMARY
KEY ,
     desc_processo  VARCHAR (80)  NOT NULL
    ) ;

CREATE TABLE INSCRICAO
    (
     nr_inscricao   NUMBER (5)  CONSTRAINT PK_CANDIDATO_CARGO PRIMARY KEY
,
     cod_candidato  NUMBER (5)  NOT NULL ,
     cod_cargo      NUMBER (4)  NOT NULL ,
     dt_inscricao   DATE        NOT NULL
    ) ;

CREATE TABLE CANDIDATO_FUNCIONARIO
```

```
    (
     cod_indicacao  NUMBER (5)  CONSTRAINT PK_CANDIDATO_FUNCIONARIO
PRIMARY KEY ,
     matricula      NUMBER (5)  NOT NULL ,
     cod_candidato  NUMBER (5)  NOT NULL ,
     dt_indicacao   DATE        NOT NULL
    ) ;

ALTER TABLE FUNCIONARIO
    ADD ( FOREIGN KEY  (cod_depto)
          REFERENCES DEPARTAMENTO (cod_depto)
        );

ALTER TABLE FUNCIONARIO
    ADD ( FOREIGN KEY (cod_cargo)
          REFERENCES CARGO (cod_cargo)
        );

ALTER TABLE CANDIDATO_FUNCIONARIO
    ADD ( FOREIGN KEY (matricula)
          REFERENCES FUNCIONARIO (matricula)
      );

ALTER TABLE CANDIDATO_FUNCIONARIO
    ADD ( FOREIGN KEY (cod_candidato)
          REFERENCES CANDIDATO (cod_candidato)
          ON DELETE CASCADE
        );

ALTER TABLE INSCRICAO
    ADD ( FOREIGN KEY (cod_candidato)
          REFERENCES CANDIDATO (cod_candidato)
          ON DELETE CASCADE
         );

ALTER TABLE INSCRICAO
    ADD ( FOREIGN KEY (cod_cargo)
          REFERENCES CARGO  (cod_cargo)
        );

ALTER TABLE TESTE
    ADD ( FOREIGN KEY (cod_processo)
          REFERENCES PROCESSO_SELETIVO (cod_processo)
        );

ALTER TABLE TESTE
    ADD ( FOREIGN KEY (nr_inscricao)
          REFERENCES INSCRICAO    (nr_inscricao)
        );

ALTER TABLE TESTE
    ADD ( FOREIGN KEY  (cod_cargo)
          REFERENCES CARGO   (cod_cargo)
        );
```

4. Implemente o modelo físico que representa, de forma sucinta, a admissão de pacientes de um determinado hospital.

```
CREATE TABLE cids (
    id_cid     INTEGER CONSTRAINT pk_cid PRIMARY KEY,
    cid        VARCHAR(20)  NOT NULL,
    descricao  VARCHAR(200) NOT NULL
);

CREATE TABLE enderecos (
    id_end       INTEGER CONSTRAINT pk_endereco PRIMARY KEY,
    id_paciente  INTEGER,
    logradouro   VARCHAR(80) NOT NULL,
    bairro       VARCHAR(50) NOT NULL,
    cidade       VARCHAR(60) NOT NULL,
    estado       CHAR(2)     NOT NULL,
    cep          CHAR(8)     NOT NULL
);

CREATE TABLE escolaridades (
    id_escolaridade    INTEGER CONSTRAINT pk_escolaridade PRIMARY KEY,
    desc_escolaridade  CHAR(30) NOT NULL
);

CREATE TABLE estados_civis (
    id_estadocivil    INTEGER CONSTRAINT pk_estadocivil PRIMARY KEY,
    desc_estadocivil  CHAR(30) NOT NULL
);

CREATE TABLE faixas_salariais (
    id_faixasalarial    INTEGER CONSTRAINT pk_faixasalarial PRIMARY KEY,
    desc_faixasalarial  CHAR(30) NOT NULL
);

CREATE TABLE grupos_diagnosticos (
    id_grupo_diagnostico  INTEGER pk_grupo_diagnostico PRIMARY KEY,
    grupo_diagnostico     VARCHAR(100) NOT NULL
);

CREATE TABLE itens (
    id_item             INTEGER CONSTRAINT pk_item PRIMARY KEY,
    cod_item            VARCHAR(255)  NOT NULL,
    desc_item           VARCHAR(255)  NOT NULL,
    desc_cat_item       VARCHAR(200)  NOT NULL,
    desc_sub_cate_item  VARCHAR(200)  NOT NULL,
    flg_opme            VARCHAR(5)    NOT NULL,
    preco_custo         NUMBER(10, 2) NOT NULL,
    preco_venda         NUMBER(10, 2) NOT NULL,
    categoria_item      VARCHAR(20)   NOT NULL
);

CREATE TABLE medicos (
    id      INTEGER CONSTRAINT pk_medico PRIMARY KEY ,
    medico  VARCHAR(20) NOT NULL
);

CREATE TABLE pacientes (
    id_paciente     INTEGER CONSTRAINT pk_paciente PRIMARY KEY,
    paciente        VARCHAR(100) NOT NULL,
    prontuario      INTEGER      NOT NULL,
    dt_nascimento   DATE NOT NULL,
    sexo            INTEGER      NOT NULL,
```

```
    estado_civil    INTEGER      NOT NULL,
    escolaridade    INTEGER      NOT NULL,
    faixa_salarial  INTEGER,
    cpf             VARCHAR(11)  NOT NULL,
    documento       VARCHAR(10)
);

CREATE TABLE passagens (
    id_passagem           INTEGER CONSTRAINT pk_passagem PRIMARY KEY ,
    id_paciente           INTEGER       NOT NULL,
    data_admissao         DATE          NOT NULL,
    id_grupo_diagnostico  INTEGER       NOT NULL,
    observacao            VARCHAR(4000) NOT NULL,
    dt_alta               DATE,
    tipo_admissao         INTEGER       NOT NULL,
    id_medico             INTEGER       NOT NULL,
    id_cid                INTEGER       NOT NULL
);

CREATE TABLE passagens_itens (
    id_pass_item   INTEGER CONSTRAINT pk_passagem_item PRIMARY KEY,
    id_item        INTEGER       NOT NULL,
    id_passagem    INTEGER       NOT NULL,
    dt_lancamento  DATE          NOT NULL,
    quantidade     NUMBER(10, 2) NOT NULL
);

CREATE TABLE queixas (
    id_queixa    INTEGER CONSTRAINT pk_queixa PRIMARY KEY ,
    id_passagem  INTEGER       NOT NULL,
    queixa       VARCHAR(4000) NOT NULL
);

CREATE TABLE tipos_admissoes (
    id_tipoadmissao    INTEGER CONSTRAINT pk_tipoadmissao PRIMARY KEY,
    desc_tipoadmissao  VARCHAR(20) NOT NULL
ALTER TABLE enderecos
    ADD CONSTRAINT fk_endereco_paciente FOREIGN KEY ( id_paciente )
        REFERENCES pacientes ( id_paciente );

ALTER TABLE pacientes
    ADD CONSTRAINT fk_paciente_escolaridade FOREIGN KEY ( escolaridade )
        REFERENCES escolaridades ( id_escolaridade );

ALTER TABLE pacientes
    ADD CONSTRAINT fk_paciente_estadocivil FOREIGN KEY ( estado_civil )
        REFERENCES estados_civis ( id_estadocivil );

ALTER TABLE pacientes
    ADD CONSTRAINT fk_paciente_faixasalarial FOREIGN KEY ( faixa_salarial
)
        REFERENCES faixas_salariais ( id_faixasalarial );

ALTER TABLE passagens
    ADD CONSTRAINT fk_passagem_cid FOREIGN KEY ( id_cid )
        REFERENCES cids ( id_cid );

ALTER TABLE passagens
    ADD CONSTRAINT fk_passagem_grupo_diagnostico FOREIGN KEY (
id_grupo_diagnostico )
        REFERENCES grupos_diagnosticos ( id_grupo_diagnostico );
```

```
ALTER TABLE passagens_itens
    ADD CONSTRAINT fk_passagem_item_item FOREIGN KEY ( id_item )
        REFERENCES itens ( id_item );

ALTER TABLE passagens_itens
    ADD CONSTRAINT fk_passagem_item_passagem FOREIGN KEY ( id_passagem )
        REFERENCES passagens ( id_passagem );

ALTER TABLE passagens
    ADD CONSTRAINT fk_passagem_medico FOREIGN KEY ( id_medico )
        REFERENCES medicos ( id );

ALTER TABLE passagens
    ADD CONSTRAINT fk_passagem_paciente FOREIGN KEY ( id_paciente )
        REFERENCES pacientes ( id_paciente );

ALTER TABLE passagens
    ADD CONSTRAINT fk_passagem_tipoadmissao FOREIGN KEY ( tipo_admissao )
        REFERENCES tipos_admissoes ( id_tipoadmissao );

ALTER TABLE queixas
    ADD CONSTRAINT fk_queixa_passagem FOREIGN KEY ( id_passagem )
        REFERENCES passagens ( id_passagem );
```

5. Crie o modelo físico de banco de dados conforme o modelo relacional a seguir. O modelo é baseados em uma empresa de amostra fictícia que vende produtos por meio de vários canais. A empresa opera em todo o mundo para atender pedidos de produtos e possui várias divisões. Esse diagrama representa a divisão de recursos humanos e rastreia informações sobre os funcionários e instalações da empresa.

```
CREATE TABLE countries (
    country_id    CHAR(2) CONSTRAINT country_c_id_pk PRIMARY KEY,
    country_name  VARCHAR2(40),
    region_id     NUMBER
);

CREATE TABLE departments (
    department_id    NUMBER(4)    CONSTRAINT dept_id_pk   PRIMARY KEY,
    department_name  VARCHAR2(30) CONSTRAINT dept_name_nn NOT NULL,
    manager_id       NUMBER(6),
    location_id      NUMBER(4)
);

CREATE TABLE employees (
    employee_id     NUMBER(6)    CONSTRAINT emp_emp_id_pk    PRIMARY KEY
,
    first_name      VARCHAR2(20),
    last_name       VARCHAR2(25) CONSTRAINT emp_last_name_nn NOT NULL,
    email           VARCHAR2(25) CONSTRAINT emp_email_nn     NOT NULL
                                 CONSTRAINT emp_email_uk     UNIQUE,
    phone_number    VARCHAR2(20),
    hire_date       DATE         CONSTRAINT emp_hire_date_nn NOT NULL,
    job_id          VARCHAR2(10) CONSTRAINT emp_job_nn       NOT NULL,
    salary          NUMBER(8, 2) CONSTRAINT emp_salary_min   CHECK (
salary > 0 ),
    commission_pct  NUMBER(2, 2),
    manager_id      NUMBER(6),
    department_id   NUMBER(4)
);

CREATE TABLE job_history (
    employee_id    NUMBER(6)    CONSTRAINT jhist_employee_nn   NOT NULL,
    start_date     DATE         CONSTRAINT jhist_start_date_nn NOT NULL,
    end_date       DATE         CONSTRAINT jhist_end_date_nn   NOT NULL,
    job_id         VARCHAR2(10) CONSTRAINT jhist_job_nn        NOT NULL,
    department_id  NUMBER(4),
    CONSTRAINT jhist_emp_id_st_date_pk PRIMARY KEY (
employee_id,start_date )
);

ALTER TABLE job_history
ADD CONSTRAINT jhist_date_interval
CHECK ( end_date > start_date );

CREATE TABLE jobs (
    job_id      VARCHAR2(10) CONSTRAINT job_id_pk PRIMARY KEY,
    job_title   VARCHAR2(35) CONSTRAINT job_title_nn NOT NULL,
    min_salary  NUMBER(6),
    max_salary  NUMBER(6)
);

CREATE TABLE locations (
    location_id     NUMBER(4)    CONSTRAINT loc_id_pk PRIMARY KEY,
    street_address  VARCHAR2(40),
    postal_code     VARCHAR2(12),
```

```
    city            VARCHAR2(30) CONSTRAINT loc_city_nn NOT NULL,
    state_province  VARCHAR2(25),
    country_id      CHAR(2)
);

CREATE TABLE regions (
    region_id    NUMBER   CONSTRAINT reg_id_pk PRIMARY KEY,
    region_name  VARCHAR2(25)
);

ALTER TABLE countries
    ADD CONSTRAINT countr_reg_fk FOREIGN KEY ( region_id )
        REFERENCES regions ( region_id );

ALTER TABLE departments
    ADD CONSTRAINT dept_loc_fk FOREIGN KEY ( location_id )
        REFERENCES locations ( location_id );

ALTER TABLE departments
    ADD CONSTRAINT dept_mgr_fk FOREIGN KEY ( manager_id )
        REFERENCES employees ( employee_id );

ALTER TABLE employees
    ADD CONSTRAINT emp_dept_fk FOREIGN KEY ( department_id )
        REFERENCES departments ( department_id );

ALTER TABLE employees
    ADD CONSTRAINT emp_job_fk FOREIGN KEY ( job_id )
        REFERENCES jobs ( job_id );

ALTER TABLE job_history
    ADD CONSTRAINT jhist_dept_fk FOREIGN KEY ( department_id )
        REFERENCES departments ( department_id );

ALTER TABLE job_history
    ADD CONSTRAINT jhist_emp_fk FOREIGN KEY ( employee_id )
        REFERENCES employees ( employee_id );

ALTER TABLE job_history
    ADD CONSTRAINT jhist_job_fk FOREIGN KEY ( job_id )
        REFERENCES jobs ( job_id );

ALTER TABLE locations
    ADD CONSTRAINT loc_c_id_fk FOREIGN KEY ( country_id )
        REFERENCES countries ( country_id );
```

6. Execute as seguintes tarefas a seguir:
   a. Crie o modelo físico com base no modelo relacional a seguir:

```
CREATE TABLE dept (
    deptno  NUMBER(2) CONSTRAINT pk_dept PRIMARY KEY ,
    dname   VARCHAR2(14 BYTE),
    loc     VARCHAR2(13 BYTE)
);

CREATE TABLE emp (
    empno     NUMBER(5) CONSTRAINT pk_emp PRIMARY KEY ,
    ename     VARCHAR2(10 BYTE),
    job       VARCHAR2(9 BYTE),
    mgr       NUMBER(4),
    hiredate  DATE,
```

```
    sal       NUMBER(7, 2),
    comm      NUMBER(7, 2),
    deptno    NUMBER(2)
);

ALTER TABLE emp
    ADD CONSTRAINT fk_deptno
    FOREIGN KEY ( deptno )
    REFERENCES dept ( deptno );
```

b.

```
DESC EMP;
```

c.

```
DESC DEPT;
```

d.

```
ALTER TABLE EMP RENAME TO EMPS;
```

e.

```
ALTER TABLE DEPT RENAME TO DEPTS;
```

```
ALTER TABLE EMPS RENAME CONSTRAINT PK_EMP TO PK_EMPS;
```

f.

```
ALTER TABLE EMPS RENAME CONSTRAINT FK_DEPTNO TO FK_DEPSTNO;
```

g.

```
ALTER TABLE DEPTS RENAME CONSTRAINT PK_DEPT TO PK_DEPTS;
```

h.

```
ALTER TABLE EMPS RENAME COLUMN SAL TO SALARY;
```

i.

```
ALTER TABLE DEPTS DROP COLUMN LOC;
```

j.

```
ALTER TABLE EMPS ADD BIRTH_DATE DATE;
```

k.

```
ALTER TABLE EMPS MODIFY SALARY NUMBER(8,3);
```

l.

```
ALTER TABLE EMPS MODIFY ENAME NOT NULL;
ALTER TABLE EMPS MODIFY SALARY NOT NULL;
```

m.

```
ALTER TABLE DEPTS MODIFY DNAME VARCHAR(20) NOT NULL;
```

n.

```
DESC EMPS;
```

o.

```
DESC DETPS;
```

III. Instruções DML-Data Manipulation Language

1.
```
CREATE TABLE my_employee
  (id  NUMBER(4) CONSTRAINT my_employee_id_nn NOT NULL,
   last_name VARCHAR2(25),
   first_name VARCHAR2(25),
   userid  VARCHAR2(8),
   salary  NUMBER(9,2));
```

2.
```
DESCRIBE my_employee;
```

3.
```
INSERT INTO my_employee
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

4.
```
INSERT INTO my_employee (id, last_name, first_name, userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5.
```
SELECT   *
FROM     my_employee;
```

6.
```
INSERT INTO my_employee (id, last_name, first_name, userid, salary)
VALUES (3, 'Biri', 'Ben', 'bbiri', 1100);
```

7.
```
INSERT INTO my_employee (id, last_name, first_name, userid, salary)
VALUES (4, 'Newman', 'Chad', 'cnewman', 750);
```

8.
```
SELECT   *
FROM my_employee;


COMMIT;
```

9.
```
UPDATE  my_employee
SET     last_name = 'Drexler'
WHERE   id = 3;
```

10.
```
UPDATE  my_employee
SET     salary = 1000
WHERE   salary < 900;
```

11.
```
SELECT  last_name, salary
FROM    my_employee;
```

12.
```
DELETE
FROM  my_employee
WHERE last_name = 'Dancs';
```

13.
```
SELECT  *
```

```
FROM    my_employee;
```

14.
```
COMMIT;
```

IV. Instruções DQL-Data Query Language

Parte 1

Teste os seus conhecimentos.

1. Inicie uma sessão do SQL Developer com o ID de usuário e a senha.
2. Falso pois os comandos que acessam o banco de dados são comandos da SQL.
3. VERDADEIRO
4. VERDADEIRO
5.
```
1- Falta vírugla após a coluna last_name.
2- Não existe a coluna sal na tabela EMPLOYEES. A coluna é salary.
3- O operador de multiplicação é o * e não o X.
4- O apelido deve estar entre aspas "ANNUAL SALARY"

SELECT      employee_id, last_name, salary * 12 "ANNUAL SALARY"
FROM        employees;
```

Parte 2

6.
```
DESCRIBE departments

SELECT *
FROM   departments;
```
7.
```
DESCRIBE employees


SELECT employee_id, last_name, job_id, hire_date StartDate

FROM   employees;
```

Parte 3

8.
```
SELECT DISTINCT job_id

FROM   employees;
```
9.
```
SELECT employee_id "Emp #",

      last_name "Employee",

      job_id "Job",

      hire_date "Hire Date"

FROM   employees;
```
10.
```
SELECT last_name||', ' || job_id "Employee and Title"
FROM   employees;
```

V. Restringindo e Classificando Dados

O departamento de recursos humanos precisa da sua ajuda para criar algumas consultas.

1.
```
SELECT  last_name, salary
FROM    employees
WHERE   salary > 12000;
```

2.
```
SELECT  last_name, department_id
FROM    employees
WHERE   employee_id = 176;
```

3.
```
SELECT  last_name, salary
FROM    employees
WHERE   salary NOT BETWEEN 5000 AND 12000;
```

4.
```
SELECT   last_name, job_id, hire_date
FROM     employees
WHERE    last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

5.
```
SELECT   last_name, department_id
FROM     employees
WHERE    department_id IN (20, 50)
ORDER BY last_name ASC;
```

6.
```
SELECT   last_name "Employee", salary "Monthly Salary"
FROM     employees
WHERE    salary  BETWEEN 5000 AND 12000
AND      department_id IN (20, 50);
```

7.
```
SELECT   last_name, hire_date
FROM     employees
WHERE    hire_date LIKE '%94';
```

8.
```
SELECT   last_name, job_id
FROM     employees
WHERE    manager_id IS NULL;
```

9.
```
SELECT   last_name, salary, commission_pct
FROM     employees
WHERE    commission_pct IS NOT NULL
ORDER BY salary DESC, commission_pct DESC;
```

10.
```
SELECT   last_name
FROM     employees
WHERE    last_name LIKE '__a%';
```

11.

```
SELECT   last_name
FROM     employees
WHERE    last_name LIKE '%a%'
AND      last_name LIKE '%e%';
```

12.

```
SELECT   last_name, job_id, salary
FROM     employees
WHERE    job_id IN ('SA_REP', 'ST_CLERK')
AND      salary NOT IN (2500, 3500, 7000);
```

13.

```
SELECT   last_name "Employee", salary "Monthly Salary",  commission_pct
FROM     employees
WHERE    commission_pct = .20;
```

VI. Exibindo Dados de Várias Tabelas

1.
```
SELECT location_id, street_address, city, state_province, country_name
FROM   locations l JOIN  countries c
on (l.country_id=c.country_id);
```

2.
```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees  e JOIN   departments d
on (e.department_id=d.department_id);
```

3.
```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
JOIN   locations l
ON     (d.location_id = l.location_id)
WHERE LOWER(l.city) = 'toronto';
```

4.
```
SELECT e.department_id department, e.last_name employee,c.last_name
colleague
FROM   employees e JOIN employees c
ON     (e.department_id = c.department_id)
WHERE  e.employee_id <> c.employee_id
ORDER BY e.department_id, e.last_name, c.last_name;
```

5.
```
SELECT e.last_name, e.hire_date
FROM   employees e JOIN employees davies
ON     (davies.last_name = 'Davies')
WHERE  davies.hire_date < e.hire_date;
```

6.
```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w JOIN employees m
ON    (w.manager_id = m.employee_id)
WHERE    w.hire_date <  m.hire_date;
```

VII. Funções de Grupo

1. VERDADEIRO

2. FALSO

3. FALSO

4.
```
SELECT ROUND(MAX(salary),0) "Maximum",
ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),0) "Sum",
ROUND(AVG(salary),0) "Average"
FROM   employees;
```

5.
```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",
ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),0) "Sum",
ROUND(AVG(salary),0) "Average"
FROM   employees
GROUP BY job_id;
```

6.
```
SELECT job_id, COUNT(*)
FROM   employees
GROUP BY job_id;


SELECT job_id, COUNT(*)
FROM   employees
WHERE  job_id = '&job_title'
GROUP BY job_id;
```

7.
```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM   employees;
```

8.
```
SELECT   MAX(salary) - MIN(salary) DIFFERENCE
FROM     employees;
```

9.
```
SELECT   manager_id, MIN(salary)
FROM     employees
WHERE    manager_id IS NOT NULL
GROUP BY manager_id
HAVING   MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

10.
```
SELECT  COUNT(*) total,
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1995,1,0))"1995",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1996,1,0))"1996",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1997,1,0))"1997",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1998,1,0))"1998"
FROM    employees;
```

11.
```
SELECT   job_id "Job",
```

```
SUM(DECODE(department_id , 20, salary)) "Dept 20",
SUM(DECODE(department_id , 50, salary)) "Dept 50",
SUM(DECODE(department_id , 80, salary)) "Dept 80",
SUM(DECODE(department_id , 90, salary)) "Dept 90",
SUM(salary) "Total"
FROM     employees
GROUP BY job_id;
```

VIII. Subconsultas

1.
```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  salary > (SELECT AVG(salary)
                 FROM   employees)
ORDER BY salary;
```

2.
```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id IN (SELECT department_id
                         FROM   employees
                         WHERE  last_name like '%u%');
```

3.
```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                         FROM   departments
                         WHERE  location_id = 1700);
```

4.
```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                     FROM   employees
                    WHERE  last_name = 'King');
```

5.
```
SELECT department_id, last_name, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                         FROM   departments
                         WHERE  department_name = 'Executive');
```

6.
```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  department_id IN (SELECT department_id
                         FROM   employees
                         WHERE  last_name like '%u%')
AND    salary > (SELECT AVG(salary)
                 FROM   employees);
```