CIS 260 Practice Exam – Lab #15

**INSTRUCTIONS:**

Company XYZ is currently in the process of converting its antiquated electronic HR database system into a modern relationship database system that was designed specifically for HR functions.

Following each of the defined steps and perform each of the requirements asked. Read the instructions carefully for each step. You will be graded on all aspects to include proper syntax, naming constraints, etc.

**STEP I – Using the HR database entity relationship diagram, create the following tables in your Oracle database schema (i.e., namespace):**

REGIONS

COUNTRIES

LOCATIONS

DEPARTMENTS – FK constraints not required for this table

\*\*\*IN ORDER TO MINIMIZE ANY ERRORS THAT MAY RESULT FROM FK CONSTRAINTS, IT IS ADVISED THAT YOU START BY CREATING THE TABLE THAT DOESN’T HAVE ANY FKs FIRST. IF YOU DO NOT FOLLOW THIS ADVICE, YOU WILL FIND YOURSELF HAVING INTEGRITY PROBLEMS. (HINT: REGIONS🡪 COUNTRIES🡪 LOCATIONS🡪 DEPARTMENTS)\*\*\*

You are to create the tables and columns exactly as they are named and defined on the ERD.

\*\*\*IGNORE THE USERNAME QUALIFIER ‘JGEORG32’ IN THE ENTITY NAME\*\*\*

You must properly create any required NOT NULL, PRIMARY KEY, FOREIGN KEY, and UNIQUE constraints. You ARE NOT required to create INDEXES.

COPY AND PASTE YOUR COUNTRIES CREATE TABLE SQL STATEMENT HERE

create table regions (

region\_id number,

region\_name varchar2(25),

CONSTRAINT reg\_id\_PK PRIMARY KEY(region\_id));

COPY AND PASTE YOUR LOCATIONS CREATE TABLE SQL STATEMENT HERE

CREATE TABLE LOCATIONS(

LOCATION\_ID NUMBER(4),

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),

CONSTRAINT LOC\_ID\_PK PRIMARY KEY(LOCATION\_ID),

CONSTRAINT LOC\_C\_ID\_FK FOREIGN KEY(COUNTRY\_ID)REFERENCES COUNTRIES(COUNTRY\_ID));

COPY AND PASTE YOUR COUNTRIES CREATE TABLE SQL STATEMENT HERE

create table countries (

country\_id char(2),

country\_name varchar2(40),

region\_id number,

CONSTRAINT country\_c\_id\_PK PRIMARY KEY(country\_id),

CONSTRAINT countr\_reg\_fk FOREIGN KEY(region\_id)REFERENCES REGIONS(region\_id));

**STEP II – Import/Insert Data**

COPY AND PASTE YOUR DEPARTMENTS CREATE TABLE SQL STATEMENT HERE

**\*NOTE – CREATE THE MANAGER\_ID COLUMN, BUT DO NOT CREATE THE FOREIGN KEY CONSTRAINT\***

CREATE TABLE DEPARTMENTS(

DEPARTMENT\_ID NUMBER(4),

DEPARTMENT\_NAME VARCHAR2(30) CONSTRAINT DEPT\_DEPT\_NAME\_NN NOT NULL,

MANAGER\_ID NUMBER(6),

LOCATION\_ID NUMBER(4),

CONSTRAINT DEPT\_ID\_PK PRIMARY KEY(DEPARTMENT\_ID),

CONSTRAINT DEPT\_LOC\_FK FOREIGN KEY(LOCATION\_ID)REFERENCES LOCATIONS(LOCATION\_ID))

**(PART A)**

For the REGIONS table, use the INSERT statement to manually insert the following records. You should have four(4) separate insert statements for this. (YOU MUST SHOW YOUR INSERT SQL STATEMENT BESIDES RUNNING IT IN SQL DEVELOPER)

|  |  |
| --- | --- |
| REGION ID | REGION\_NAME |
| 1 | Europe |
| 2 | Americas |
| 3 | Asia |
| 4 | Middle East and Africa |

COPY AND PASTE YOUR FOUR (4) SQL INSERT STATEMENTS IN THIS BOX:

INSERT INTO REGIONS VALUES(1,'EUROPE');

INSERT INTO REGIONS VALUES(2,'AMERICAS');

INSERT INTO REGIONS VALUES(3,'ASIA');

INSERT INTO REGIONS VALUES(4,'MIDDLE EAST AND AFRICA');

**(PART B)** Copy and paste and run the following SQL as a script in SQL Developer to INSERT the following records:

INSERT INTO countries VALUES

( 'IT'

, 'Italy'

, 1

);

INSERT INTO countries VALUES

( 'JP'

, 'Japan'

, 3

);

INSERT INTO countries VALUES

( 'US'

, 'United States of America'

, 2

);

INSERT INTO countries VALUES

( 'CA'

, 'Canada'

, 2

);

INSERT INTO countries VALUES

( 'CN'

, 'China'

, 3

);

INSERT INTO countries VALUES

( 'IN'

, 'India'

, 3

);

INSERT INTO countries VALUES

( 'AU'

, 'Australia'

, 3

);

INSERT INTO countries VALUES

( 'ZW'

, 'Zimbabwe'

, 4

);

INSERT INTO countries VALUES

( 'SG'

, 'Singapore'

, 3

);

INSERT INTO countries VALUES

( 'UK'

, 'United Kingdom'

, 1

);

INSERT INTO countries VALUES

( 'FR'

, 'France'

, 1

);

INSERT INTO countries VALUES

( 'DE'

, 'Germany'

, 1

);

INSERT INTO countries VALUES

( 'ZM'

, 'Zambia'

, 4

);

INSERT INTO countries VALUES

( 'EG'

, 'Egypt'

, 4

);

INSERT INTO countries VALUES

( 'BR'

, 'Brazil'

, 2

);

INSERT INTO countries VALUES

( 'CH'

, 'Switzerland'

, 1

);

INSERT INTO countries VALUES

( 'NL'

, 'Netherlands'

, 1

);

INSERT INTO countries VALUES

( 'MX'

, 'Mexico'

, 2

);

INSERT INTO countries VALUES

( 'KW'

, 'Kuwait'

, 4

);

INSERT INTO countries VALUES

( 'IL'

, 'Israel'

, 4

);

INSERT INTO countries VALUES

( 'DK'

, 'Denmark'

, 1

);

INSERT INTO countries VALUES

( 'HK'

, 'HongKong'

, 3

);

INSERT INTO countries VALUES

( 'NG'

, 'Nigeria'

, 4

);

INSERT INTO countries VALUES

( 'AR'

, 'Argentina'

, 2

);

INSERT INTO countries VALUES

( 'BE'

, 'Belgium'

, 1

);

**(PART C)**

Using the SQL Developer Import Wizard, import the two .csv files for the LOCATIONS and DEPARTMENTS tables. These have been tested and should import ‘cleanly’ and without errors.

**STEP III – CREATE VIEW and JOINS**

The HR Department would like access to a quick “view” of the location\_id, city, state, and country\_name a specific location is in. Create a VIEW that will display: location\_id, city, state, and country\_name. The column appearance for STATE\_PROVINCE should appear as “STATE”, and the column appearance COUNTRY\_NAME should appear as “COUNTRY”.

COPY AND PASTE YOUR SQL STATEMENT USED TO CREATE THE VIEW HERE:

SELECT l.location\_id, l.city, l.state\_province, c.country\_name

From locations l join countries c on l.country\_id=c.country\_id;

create view v\_locations as (SELECT l.location\_id, l.city, l.state\_province AS STATE, c.country\_name AS COUNTRY

From locations l join countries c on l.country\_id=c.country\_id);

**STEP IV – PL/SQL**

Senior management would like an “easy button” located on their front-end application dashboard that when clicked will display the following information:

“Location ID: **location\_id** is located in **city** and in the state of **state\_province**”

You must write a PL/SQL script that uses three declared variables. Using a CURSOR, the script will display all rows found in the result set.

COPY AND PASTE YOUR PL/SQL SCRIPT HERE:

SET SERVEROUTPUT ON;

DECLARE

v\_loc\_city locations.city%type;

v\_loc\_state locations.state\_province%type;

v\_loc\_id LOCATIONS.LOCATION\_ID%type;

CURSOR location\_cursor is

SELECT location\_id,city, state\_province from locations;

BEGIN

OPEN location\_cursor;

LOOP

FETCH location\_cursor into v\_loc\_id, v\_loc\_city, v\_loc\_state;

EXIT WHEN location\_cursor%notfound;

dbms\_output.put\_line('Location ID: ' || v\_loc\_id|| ' is located in ' || v\_loc\_city || ' and in the state of ' || v\_loc\_state);

END LOOP;

CLOSE location\_cursor;

END;