# A3-20163-DBS301 --- DUE week 12 Friday before MIDNIGHT

**Extended deadline:** Monday Week 13 December 2016 before midnight. After midnight the value of assignment is zero.

# Immediately save this as a3-20163-yourname

# OR lose 20% …thank you (maybe this will work)

CREATE TABLES Question

DIVISION

|  |  |  |
| --- | --- | --- |
| **Column Name** | DIVISION\_ID | DIVISION\_NAME |
| **Key Type** | **PK** |  |
| **Null/Unique** |  | **NN, U** |
| **FK Table** |  |  |
| **FK Column** |  |  |
| **Validation** |  |  |
| **Datatype** | **NUMBER** | **VARCHAR** |
| **Length** | **3** | **25** |
| **Sample data** |  |  |
|  | **10** | **East Coast** |
|  | **20** | **Quebec** |
|  | **30** | **Ontario** |

WAREHOUSE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | WAREHOUSE\_ID | CITY | RATING | FOUND\_DATE | DIVISION\_ID |
| **Key Type** | **PK** |  | **CK** |  | **FK** |
| **Null/Unique** |  | **NN, U** |  | **NN** | **NN** |
| **FK Table** |  |  |  |  | **DIVISION** |
| **FK Column** |  |  |  |  | **DIVISION\_ID** |
| **Validation** |  |  | **A, B, C, D** |  |  |
| **Datatype** | **NUMBER** | **VARCHAR** | **CHAR** | **DATE** | **NUMBER** |
| **Length** | **3** | **15** | **1** |  | **3** |
| **Sample Data** | **1** | **Montreal** | **A** | **SYSDATE()** | **10** |
|  | **7** | **Fredericton** | **B** |  | **10** |
|  | **10** | **Toronto** | **A** |  | **30** |

SECTION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | WAREHOUSE\_ID | SECTION\_ID | DESCRIPTION | CAPACITY |
| **Key Type** | **PK, FK** | **PK** |  |  |
| **Null/Unique** |  |  | **NN** |  |
| **FK Table** | **WAREHOUSE** |  |  |  |
| **FK Column** | **WAREHOUSE\_ID** |  |  |  |
| **Datatype** | **NUMBER** | **NUMBER** | **VARCHAR** | **NUMBER** |
| **Length** | **3** | **2** | **50** | **8** |

1 (10 marks) Write the required SQL statements to create tables WAREHOUSE, DIVISION and SECTION.

Follow these general rules in the process:

A. Create all CHECK (incl. NOT NULL) and UNIQUE as column level constraints

Constraint names needed for CHECK constraints. The other constraints (NN and UK) do not need a name.

B. Create all PK and FK constraints at the table level and give them proper names.

PUT ANSWERS starting here

**CREATE TABLE DIVISION(**

**DIVISION\_ID NUMBER(3),**

**DIVISION\_NAME VARCHAR(25) NOT NULL UNIQUE,**

**CONSTRAINT DIV\_DIVISION\_ID\_PK**

**PRIMARY KEY(DIVISION\_ID)**

**);**

**CREATE TABLE WAREHOUSE(**

**WAREHOUSE\_ID NUMBER(3),**

**CITY VARCHAR(15) NOT NULL UNIQUE,**

**RATING CHAR(1)**

**CONSTRAINT WHR\_RATING\_CHK CHECK(RATING IN ('A','B','C','D')),**

**FOUND\_DATE DATE NOT NULL,**

**DIVISION\_ID NUMBER(3) NOT NULL,**

**CONSTRAINT WHR\_WAREHOUSE\_ID\_PK**

**PRIMARY KEY(WAREHOUSE\_ID),**

**CONSTRAINT WHR\_DIVISION\_ID\_FK**

**FOREIGN KEY(DIVISION\_ID)**

**REFERENCES DIVISION(DIVISION\_ID)**

**);**

**CREATE TABLE SECTION(**

**WAREHOUSE\_ID NUMBER(3),**

**SECTION\_ID NUMBER(2),**

**DESCRIPTION VARCHAR(50) NOT NULL,**

**CAPACITY NUMBER(8),**

**CONSTRAINT SECT\_WAREHOUSE\_ID\_FK**

**FOREIGN KEY(WAREHOUSE\_ID)**

**REFERENCES WAREHOUSE(WAREHOUSE\_ID),**

**CONSTRAINT SECT\_COMPOSITE\_PK**

**PRIMARY KEY(WAREHOUSE\_ID, SECTION\_ID)**

**);**

**2** (3 marks) After creating all tables add column MGR\_ID to table SECTION as a FK column, that is related to the PK column EMPLOYEE\_ID in table EMPLOYEE

**ALTER TABLE SECTION**

**ADD MGR\_ID NUMBER(6)**

**CONSTRAINT SECTION\_MGR\_ID\_FK**

**REFERENCES EMPLOYEES(EMPLOYEE\_ID);**

3 (3 marks) Modify the CHECK constraint on column RATING in table WAREHOUSE, so that it also may accept a new value F.

**ALTER TABLE WAREHOUSE**

**DROP CONSTRAINT WHR\_RATING\_CHECK;**

**ALTER TABLE WAREHOUSE**

**ADD CONSTRAINT WHR\_RATING\_CHECK**

**CHECK(RATING IN ('A','B','C','D','F'));**

4 (3 marks) Create a new **Sequence** called **Whse\_id\_seq** that will generate unique numbers for PK values in table WAREHOUSE, so that the numbers start at 410 with the step of 10 and upper limit is 700 and will have NO values stored in the memory.

**CREATE SEQUENCE WHSE\_ID\_SEQ**

**INCREMENT BY 10**

**START WITH 410**

**MAXVALUE 700**

**NOCYCLE**

**NOCACHE;**

5 (3 marks) Add new row to table WAREHOUSE by using this sequence for a city in Atlanta with unknown rating. You will assume today’s date as a foundation date. You cannot enter a specific date.

\*\*Insert record to DIVISION first as DIVISION\_ID is FK of WAREHOUSE.

**INSERT INTO DIVISION VALUES(10,'Europe');**

\*\*Rating is with CHECK constraint. It does not accept unknown rating.

**INSERT INTO WAREHOUSE (WAREHOUSE\_ID, CITY, RATING, FOUND\_DATE,DIVISION\_ID)**

**VALUES(WHSE\_ID\_SEQ.NEXTVAL,'Atlanta','A',SYSDATE,10);**

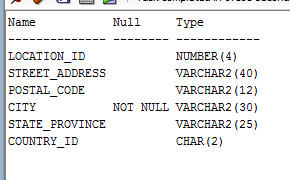
6 (5 marks) Create table CITIES **from table LOCATIONS,** but only for location numbers less than 2000 (do NOT create this table from scratch). 🡪 You will have 5 to 18 rows

**CREATE TABLE CITIES AS**

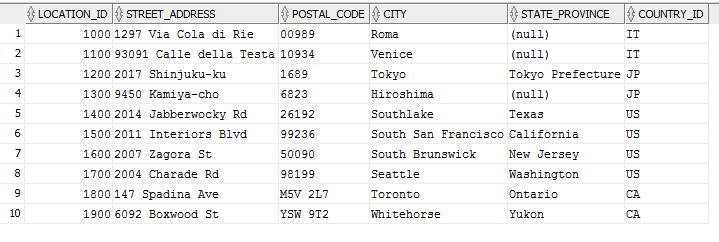
**(SELECT \* FROM LOCATIONS WHERE LOCATION\_ID < 2000);**

7 (2 marks) Issue command to show the structure of the table CITIES

**DESCRIBE CITIES;**



8 (1 mark) Issue the SELECT command on cities and show result here.



9 (5 marks) Create a View called **WhsSec\_Man\_vu** that will display for each Warehouse\_id and Section\_id, the City, Division and manager’s Last\_name. Alias for Last\_name should be LName and for Division should be Group.

**CREATE VIEW WHSSEC\_MAN\_VU AS(**

**SELECT WAREHOUSE\_ID,**

**SECTION\_ID,**

**CITY,**

**DIVISION\_NAME AS "GROUP",**

**LAST\_NAME AS "LName"**

**FROM WAREHOUSE JOIN SECTION USING(WAREHOUSE\_ID)**

**JOIN DIVISION USING(DIVISION\_ID)**

**JOIN EMPLOYEES**

**ON MGR\_ID = EMPLOYEE\_ID**

**);**

10 (1 mark) What is the SELECT command to issue if in 2 months I want to test if a view was actually was created

**SELECT VIEW\_NAME FROM USER\_VIEWS;**

11 (1 mark) If you want to modify the view what is the first line of the command

**CREATE OR REPLACE VIEW WHSSEC\_MAN\_VU AS (**

12 Issue a SET operator to show the rows that were in LOCATIONS but not in CITIES

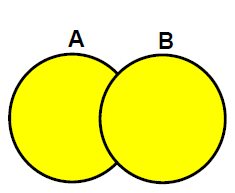
**SELECT \* FROM LOCATIONS**

**MINUS**

**SELECT \* FROM CITIES;**

Using the following diagram as a hint and not a perfect representation.

Answer 13, 13, 15 and 16



13 All the rows in A and all the rows in B with no duplicates is the set operator called [**UNION**]

14 All the rows in A and all the rows in B with duplicates [**UNION ALL**]

15 The rows in common to BOTH A and B tables [**INTERSECT**]

16 Rows that are in A but not in B would use the word [**MINUS**]