**DBAS 1100 Assignment 2 - Outer Joins \*Solutions at the End**

**Value:** 10% of your overall mark.

**Due Dates: Week of April 6th, Second class of week, before start of class**

**Required Database(s):**

* Chinook
* Bookstore
* Numbers

**Instructions:**

Create a single script file containing SQL statements to meet the query requirements listed below.

For this assignment, you will be using three separate databases, **Chinook, Bookstore** and **Numbers**. For the latter two, you will need to run the DDL scripts listed below to create the two databases required for this assignment.

**IMPORTANT NOTE:** Run each of these DDL scripts using a new connection that uses a new user account, named **bookstore** and **numbers** respectively. For the numbers.sql script, you will need to grant a new database permission to the numbers user. Up until now, we’ve always granted two permissions, *connect* and *resource* to the user. For this script to run correctly, you will need to grant *dba* privileges. Use the following syntax to do so:

**Grant dba to numbers;**

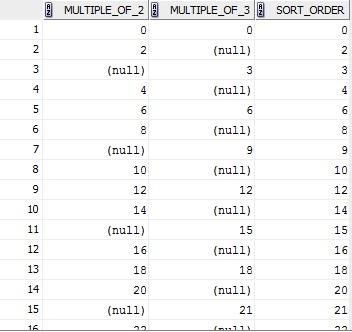
**BookStoreScript.sql** – Creates the “b\_” tables needed for this assignment.

**Numbers.sql** – Creates the “SEC” tables needed for this assignment.

When you are finished, upload your script to D2L as your submission for Assignment 2, using a file name similar to: **[YourName]\_DBAS\_Assignment2.sql**.

**Query Requirements:**

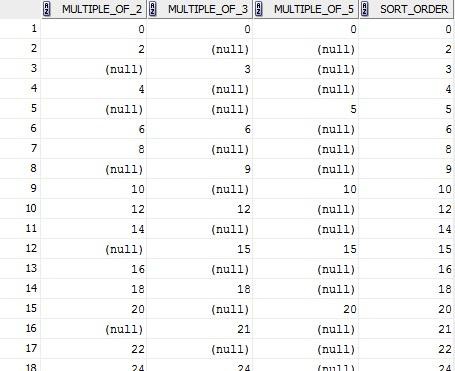
1. **(Chinook db)** Display the Name of every available track. In addition, display the number of times that the track has been included as part of an order.
2. **(Chinook db)** Display the TrackID, Track Name, of any tracks in Chinook DB that have not yet been purchased.
3. **(Bookstore db)** Using the “b\_” tables, display the first and last names of all authors who currently do not have any books listed.
4. **(Bookstore db)** Using the “b\_” tables, display the Title of any book that has yet to have been purchased.
5. **(Bookstore db)** Expanding on Question 4, display the name of any publisher who has published a book that has yet to have been purchased.
6. **(Bookstore db)** Using the “b\_” tables, display the Customer number, First name, and Last name of any customers who have yet to place an order.
7. **(Numbers db)** Create an **Inner Join** between the tables **NUMBERS\_twos** and **NUMBERS\_threes.** Look at the results and note what is returned between the two tables.
8. Change the Inner Join to a **Left Outer Join** and note the differences. Make sure you are clear as to what the differences are before you continue.
9. Change the Left Outer Join to a **Right Outer Join** and note the differences. Make sure you are clear as to what the differences are before you continue.
10. Change the Right Outer Join to a **Full Join** and note the differences**.** Make sure you are clear as to what the differences are before you continue.
11. Using what you’ve observed, create a SQL statement to return the results shown below:



(HINT: It’s a sorting issue and involves the NVL function)

**BONUS (1 point):**

**(Numbers db)** Add the table **NUMBERS\_fives** to the mix and modify the SQL in Question 7 to return the following:



SOLUTIONS:

/\* DBAS 1100 - 700 ASSIGNMENT 2

CHRIS ARBUTHNOTT, W0276812

APRIL 8TH, 2015 \*/

--1. (CHINOOK) Display the name of each track along with the number of times

-- it's been included in an order.

SELECT TRK.TRACKID, TRK.NAME, COUNT(INVLIN.INVOICEID) AS "Number of Purchases"

FROM TRACK TRK

LEFT JOIN INVOICELINE INVLIN ON INVLIN.TRACKID = TRK.TRACKID

GROUP BY TRK.TRACKID, TRK.NAME

ORDER BY TRK.NAME;

--2. (CHINOOK) Display the id and name of any tracks that have not been purchased.

SELECT TRK.TRACKID, TRK.NAME

FROM TRACK TRK

LEFT JOIN INVOICELINE INVLIN ON INVLIN.TRACKID = TRK.TRACKID

WHERE INVLIN.INVOICEID IS NULL

ORDER BY TRK.NAME;

--3. (BOOKSTORE) Display first and last names of authors with no books listed.

SELECT AUTH.FNAME, AUTH.LNAME

FROM B\_AUTHOR AUTH

LEFT JOIN B\_BOOKAUTHOR BKAUTH ON BKAUTH.AUTHORID = AUTH.AUTHORID

WHERE BKAUTH.ISBN IS NULL

ORDER BY AUTH.LNAME, AUTH.FNAME;

--4. (BOOKSTORE) Display title of books with no purchases.

SELECT BKS.TITLE

FROM B\_BOOKS BKS LEFT JOIN B\_ORDERITEMS ORDITM ON ORDITM.ISBN = BKS.ISBN

WHERE ORDITM.ITEM# IS NULL

ORDER BY BKS.TITLE;

--5. (BOOKSTORE) Display all publishers that have published any non-purchased books.

SELECT DISTINCT PUB.NAME AS "Publisher"

FROM B\_PUBLISHER PUB

INNER JOIN B\_BOOKS BKS ON BKS.PUBID = PUB.PUBID

LEFT JOIN B\_ORDERITEMS ORDITM ON ORDITM.ISBN = BKS.ISBN

WHERE ORDITM.ITEM# IS NULL

ORDER BY PUB.NAME;

-- 5.a) just to test question 5... make sure the books from q4 appear in results

-- of this query.

SELECT BKS.TITLE FROM B\_BOOKS BKS

INNER JOIN B\_PUBLISHER PUB ON BKS.PUBID = PUB.PUBID

WHERE PUB.NAME = 'READING MATERIALS INC.' OR PUB.NAME = 'PUBLISH OUR WAY';

--6. (BOOKSTORE) Display the id and names of customers who have made no orders.

SELECT CUS.CUSTOMER#, CUS.FIRSTNAME, CUS.LASTNAME

FROM B\_CUSTOMERS CUS

LEFT JOIN B\_ORDERS ORD ON ORD.CUSTOMER# = CUS.CUSTOMER#

WHERE ORD.ORDER# IS NULL

ORDER BY CUS.LASTNAME, CUS.FIRSTNAME;

--7. (NUMBERS) Create the inner join between number\_twos and number\_threes.

SELECT \* FROM NUMBERS\_TWOS TWOS

INNER JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3;

-- THE ONLY POSSIBLE INNER JOIN RETURNS THE MULTIPLES OF 6

--7.a) (NUMBERS) Observe the results of changing to a left join.

SELECT \* FROM NUMBERS\_TWOS TWOS

LEFT JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3;

-- THE LEFT JOIN ALSO INCLUDES THE MULTIPLES OF 2 THAT ARE NOT MULTIPLES OF 3

-- (WITH NULLS IN THE MULTIPLE OF 3 COLUMN)

--7.b) (NUMBERS) Observe the results of changing to a right join.

SELECT \* FROM NUMBERS\_TWOS TWOS

RIGHT JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3;

-- THE RIGHT JOIN INCLUDES MULTIPLES OF 6 AND THE MULTIPLES OF 3 THAT ARE NOT

-- MULTIPLES OF 2 (WITH NULLS IN THE MULTIPLE OF 2 COLUMN)

--7.c) (NUMBERS) Observe the results of changing to a full join.

SELECT \* FROM NUMBERS\_TWOS TWOS

FULL OUTER JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3;

-- THE FULL OUTER JOIN INCLUDES ALL MULTIPLES OF BOTH 2 AND 3.

--7.d) (NUMBERS) Produce the result table displayed in assignment - basically

-- the multiples of 2 and/or 3 together with their numerical sort order.

SELECT TWOS.MULTIPLE\_OF\_2, THREES.MULTIPLE\_OF\_3,

(NVL(TWOS.MULTIPLE\_OF\_2, THREES.MULTIPLE\_OF\_3) +

NVL(THREES.MULTIPLE\_OF\_3, TWOS.MULTIPLE\_OF\_2))/2 AS "SORT\_ORDER"

FROM NUMBERS\_TWOS TWOS

FULL OUTER JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3

ORDER BY SORT\_ORDER;

-- STRATEGY: REPLACE NULLS BY THE OTHER VALUE, THEN ADD THE VALUES AND

-- DIVIDE BY TWO.

--BONUS (NUMBERS) Add multiples of five to the above pattern.

SELECT TWOS.MULTIPLE\_OF\_2, THREES.MULTIPLE\_OF\_3, FIVES.MULTIPLE\_OF\_5,

GREATEST(NVL(TWOS.MULTIPLE\_OF\_2, 0),

NVL(THREES.MULTIPLE\_OF\_3, 0),

NVL(FIVES.MULTIPLE\_OF\_5, 0)) AS "SORT\_ORDER"

FROM NUMBERS\_TWOS TWOS

FULL OUTER JOIN NUMBERS\_THREES THREES ON TWOS.MULTIPLE\_OF\_2 = THREES.MULTIPLE\_OF\_3

FULL OUTER JOIN NUMBERS\_FIVES FIVES ON TWOS.MULTIPLE\_OF\_2 = FIVES.MULTIPLE\_OF\_5

ORDER BY SORT\_ORDER;

--STRATEGY: SET ALL NULLS TO ZERO AND USE THE 'GREATEST' FUNCTION.

-- END!