BUAN 6320 Database Foundations for Business Analytics

Spring 2019

Instructor: Dr. James Scott

Assignment #4 – Ch 8 & Ch 10

General Instructions

- Students may study together for the assignment and review each other's completed work
- Students must each complete the assignment by their own hand
- Please use the provided word document template
- Please save the completed word document into PDF format before uploading
- Please submit the PDF file electronically through eLearning before the due date and time
- Do not worry about variations among database vendors – you may write SQL to any vendor's dialect
- Do not include output – only the SQL
- Use table aliases for all tables in all queries (unless otherwise specified)
- Column aliases are required for all derived columns including aggregate columns (unless otherwise specified)
- Do not use column aliases unless required as stated previously
- If a problem does not ask for a specific sort order, use your best judgement to add a sort order

Chapter 8 – Problems 1-7, page 401

1. Create the tables. (Use the MS Access example shown in Figure P8.1 to see what table names and attributes to use.)

```
CREATE TABLE CUSTOMER (
CUST_NUM NUMERIC PRIMARY KEY,
CUST_LNAME VARCHAR(20),
CUST_FNAME VARCHAR(20),
CUST_BALANCE NUMERIC);

CREATE TABLE CUSTOMER_2 (
CUST_NUM NUMERIC PRIMARY KEY,
CUST_LNAME VARCHAR(20),
CUST_FNAME VARCHAR(20));

CREATE TABLE INVOICE (
INV_NUM NUMERIC PRIMARY KEY,
CUST_NUM NUMERIC,
INV_DATE DATE,
INV_AMOUNT NUMERIC);
```

2. Insert the data into the tables you created in Problem 1.

```
INSERT INTO CUSTOMER VALUES (1000 ,'Smith','Jeanne',1050.11);
INSERT INTO CUSTOMER VALUES (1001 ,'Ortega','Juan',840.92);

INSERT INTO CUSTOMER_2 VALUES (2000 ,'McPherson','Anne');
INSERT INTO CUSTOMER_2 VALUES (2001 ,'Ortega','Juan');
INSERT INTO CUSTOMER_2 VALUES (2002 ,'Kowalski','Jan');
INSERT INTO CUSTOMER_2 VALUES (2003 ,'Chen','George');

INSERT INTO INVOICE VALUES(8000 ,1000 ,'23-APR-2008' ,235.89);
INSERT INTO INVOICE VALUES(8001 ,1001 ,'23-MAR-2008' ,312.82);
INSERT INTO INVOICE VALUES(8002 ,1001 ,'30-MAR-2008' ,528.1);
INSERT INTO INVOICE VALUES(8003 ,1000 ,'12-APR-2008' ,194.78);
INSERT INTO INVOICE VALUES(8004 ,1000 ,'23-APR-2008' ,619.44);
```

3. Write the query that will generate a combined list of customers from the tables CUSTOMER and CUSTOMER_2 that do not include the duplicate customer records. Only the customer named Juan Ortega shows up in both customer tables.
```
SELECT C1.CUST_LNAME, C1.CUST_FNAME FROM CUSTOMER C1
UNION
SELECT C2.CUST_LNAME, C2.CUST_FNAME FROM CUSTOMER_2 C2;
```

4. Write the query that will generate a combined list of customers to include the duplicate customer records.
```
SELECT C1.CUST_LNAME, C1.CUST_FNAME FROM CUSTOMER C1
UNION ALL
SELECT C2.CUST_LNAME, C2.CUST_FNAME FROM CUSTOMER_2 C2;
```

5. Write the query that will show only the duplicate customer records.
```
SELECT C1.CUST_LNAME, C1.CUST_FNAME FROM CUSTOMER C1
INTERSECT
SELECT C2.CUST_LNAME, C2.CUST_FNAME FROM CUSTOMER_2 C2;
```

6. Write the query that will generate only the records that are unique to the CUSTOMER_2 table
```
SELECT C2.CUST_LNAME, C2.CUST_FNAME FROM CUSTOMER_2 C2
EXCEPT
SELECT C1.CUST_LNAME, C1.CUST_FNAME FROM CUSTOMER C1;
```

7. Write the query to show the invoice number, the customer number, the customer name, the invoice date, and the invoice amount for all customers in the CUSTOMER table with a balance of $1,000 or more.

```
SELECT I.INV_NUM, C.CUST_NUM, C.CUST_LNAME, C.CUST_FNAME, I.INV_DATE,
I.INV_AMOUNT
FROM INVOICE I INNER JOIN CUSTOMER C ON I.CUST_NUM = C.CUST_NUM
WHERE C.CUST_BALANCE >= 1000 order by 1;
```

Chapter 10 – Problem 1, page 483

1. Suppose that you are a manufacturer of product ABC, which is composed of parts A, B, and C. Each time a new product ABC is created, it must be added to the product inventory, using the PROD_QOH in a table named PRODUCT. Also, each time the product is created, the parts inventory, using PART_QOH in a table named PART, must be reduced by one each of parts A, B, and C. The sample database contents are shown in the following tables.
   Given the preceding information, answer Questions a through e.

a. How many database requests can you identify for an inventory update for both PRODUCT and PART?
   We can update PRODUCT and PART tables using two queries. One each for both the tables as below.

b. Using SQL, write each database request you identified in Step a.

```
UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 1
WHERE PROD_CODE = 'ABC';

UPDATE PART SET PART_QOH = PART_QOH - 1
WHERE
PART_CODE = 'A' OR
PART_CODE = 'B' OR
PART_CODE = 'C';
```

c. Write the complete transaction(s).

```
BEGIN TRANSACTION

UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 1
WHERE PROD_CODE = 'ABC';

UPDATE PART SET PART_QOH = PART_QOH - 1
WHERE
PART_CODE = 'A' OR
PART_CODE = 'B' OR
PART_CODE = 'C';

COMMIT;
```

d. Write the transaction log, using Table 10.1 as your template.

TRL_ID	TRX_NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
1	101	NULL	2	START	** Start Transaction				
2	101	1	3	UPDATE	PRODUCT	'ABC'	PROD_QOH	1205	1206
3	101	2	4	UPDATE	PART	'A'	PART_QOH	567	566
4	101	3	5	UPDATE	PART	'B'	PART_QOH	498	497
5	101	4	6	UPDATE	PART	'C'	PART_QOH	549	548
6	101	5	NULL	COMMIT	** End Transaction				

e. Using the transaction log you created in Step d, trace its use in database recovery
Database recovery is used restore to the last committed state. When failure occurs database recovery is triggered.
For transaction recovery deferred-write and write-through techniques are the two types of techniques used.
When the recovery procedure uses a deferred-write technique (also called a deferred update), the transaction operations do not immediately update the physical database. Instead, only the transaction log is updated. The database is physically updated only with data from committed transactions, using information from the transaction log.

For example, let's consider the transaction log created in part d.
This log table can be used to retrieve the original data i.e., the data before updating the PRODUCT and PART table if any failure or mistake occurs.

Using MS-SQL create the following SQL queries using UniversityDDL-Bad.

Problem #24 – Join not involving a Primary Key to a Foreign Key

List faculty who are also students. Include all student columns in the result.

```
select s.*
from student s
where s.StdNo in (select FacNo from Faculty);
```

Problem #25 – Self Join

List faculty members who have a higher salary than their supervisor
List the faculty number, last and first names, and salary for both

```
SELECT  f.FacNo AS 'Faculty Num',s.FacNo AS 'Supervisor Num',
        f.FacLastName AS 'Faculty Last Name',
        s.FacLastName AS 'Supervisor Last Name',
        f.FacFirstName AS 'Faculty First Name', s.FacFirstName AS 'Supervisor First
Name',
        f.FacSalary AS 'Faculty salary', s.FacSalary AS 'Supervisor salary'
FROM Faculty f INNER JOIN Faculty s
ON s.FacNo = f.FacSupervisor
WHERE f.FacSalary > s.FacSalary
ORDER BY 1,2,3;
```

Problem #26 – Multiple Joins involving a Table more than once

List the last and first names of faculty members and the course number for which the faculty member taught the same course number as their supervisor in 2013
Select F.FacLastName, F.FacFirstName, o1.CourseNo

```
SELECT f.FacLastName, f.FacFirstName,o1.CourseNo
 FROM(Faculty f INNER JOIN Offering o1
                           ON f.FacNo = o1.FacNo)
       INNER JOIN Offering o2
                        ON f.FacSupervisor=o2.FacNo
                        AND o1.CourseNo=o2.CourseNo
                        AND o1.OffYear=o2.OffYear
   WHERE o1.OffYear = '2013'
   ORDER BY 1,2,3;

No data can be retrieved for the above given year so trying with other year


SELECT f.FacLastName, f.FacFirstName,o1.CourseNo
 FROM(Faculty f INNER JOIN Offering o1
                           ON f.FacNo = o1.FacNo)
       INNER JOIN Offering o2
                        ON f.FacSupervisor=o2.FacNo
                        AND o1.CourseNo=o2.CourseNo
                        AND o1.OffYear=o2.OffYear
   WHERE o1.OffYear = '2010'
   ORDER BY 1,2,3;
```

Problem #27 – Left Outer Join

List all courses and their offerings
Include courses without offerings
List all columns of courses and offerings
(use a Left Outer Join)

```
SELECT c.*, o.*
FROM course c LEFT OUTER JOIN Offering o
ON c.CourseNo = o.CourseNo
ORDER BY 1,2,3,4,5,6;
```

Problem #28 – Right Outer Join

List all offerings and the faculty assigned to teach them
Also include courses without a faculty assigned to them
List year, term, course number, offering number, faculty last and first name (use a Right Outer Join)

```
SELECT o.OffYear, o.OffTerm,o.CourseNo,o.OfferNo,f.FacLastName,f.FacFirstName
FROM Faculty f RIGHT OUTER JOIN Offering o
```

```
ON O.FacNo = F.FacNo
ORDER BY 1,2,3,4,5,6;
```

Problem #29 – Mixing Left Outer Join with Inner Joins

List information for all IS courses offered in 2013 with at least 1 student enrolled
Include offerings without a faculty assigned
List the offer number, course number, term, description, faculty
number, faculty last and first names
Suppress duplicates when more than 1 student is enrolled

```
SELECT DISTINCT o.OfferNo, c.CourseNo, o.OffTerm, c.CrsDesc, f.FacNo,
                f.FacLastName, f.FacFirstName
FROM ((Offering o LEFT JOIN faculty f
             ON o.FacNo = f.FacNo)
        INNER JOIN course c
             ON o.CourseNo = c.CourseNo)
        INNER JOIN enrollment e
             ON o.OfferNo = e.OfferNo
WHERE c.CourseNo LIKE 'IS%'
AND o.OffYear = '2013'
ORDER BY 1,2;

No data can be retrieved for the above given year so trying with other years

SELECT DISTINCT o.OfferNo, c.CourseNo, o.OffTerm, c.CrsDesc, f.FacNo,
                f.FacLastName, f.FacFirstName
FROM ((Offering o LEFT JOIN faculty f
             ON o.FacNo = f.FacNo)
        INNER JOIN course c
             ON o.CourseNo = c.CourseNo)
        INNER JOIN enrollment e
             ON o.OfferNo = e.OfferNo
WHERE c.CourseNo LIKE 'IS%'
AND o.OffYear = '2009'
ORDER BY 1,2;

SELECT DISTINCT o.OfferNo, c.CourseNo, o.OffTerm, c.CrsDesc, f.FacNo,
                f.FacLastName, f.FacFirstName
FROM ((Offering o LEFT JOIN faculty f
             ON o.FacNo = f.FacNo)
        INNER JOIN course c
             ON o.CourseNo = c.CourseNo)
        INNER JOIN enrollment e
             ON o.OfferNo = e.OfferNo
WHERE c.CourseNo LIKE 'IS%'
AND o.OffYear = '2010'
ORDER BY 1,2;
```

Problem #30 – Examining the difference between UNION and UNION ALL

Retrieve all faculty and students
Only show common columns in the result

Remove duplicates

```
SELECT f.FacNo AS PersonNo, f.FacFirstName AS FirstName, f.FacLastName AS LastName,
f.FacCity AS City
FROM Faculty f
UNION
SELECT s.StdNo AS PersonNo, s.StdFirstName AS FirstName, s.StdLastName AS LastName,
s.StdCity AS City
FROM Student s
ORDER BY 1,2,3;
```

Repeat query allowing duplicates

```
SELECT f.FacNo AS PersonNo, f.FacFirstName AS FirstName, f.FacLastName AS LastName,
f.FacCity AS City
FROM Faculty f
UNION ALL
SELECT s.StdNo AS PersonNo, s.StdFirstName AS FirstName, s.StdLastName AS LastName,
s.StdCity AS City
FROM Student s
ORDER BY 1,2,3;
```

Problem #31 – Type 1 Subquery (nested one level)

List student last and first names and majors for students who had at least one high grade (>= 3.5) in at least one course offered in fall of 2012
(use a Type 1 Subquery)

```
SELECT s.StdLastName, s.StdFirstName, s.StdMajor
FROM Student s INNER JOIN Enrollment e
                   ON s.StdNo = e.StdNo
WHERE e.EnrGrade >= 3.5
      AND e.OfferNo IN
      (SELECT o.OfferNo
       FROM Offering o
     WHERE o.OffTerm = 'FALL'
             AND o.OffYear = '2012')
Order by 1,2;
No data can be retrieved for the above given year so trying with other years
```

```
SELECT s.StdLastName, s.StdFirstName, s.StdMajor
FROM Student s INNER JOIN Enrollment e
                   ON s.StdNo = e.StdNo
WHERE e.EnrGrade >= 3.5
      AND e.OfferNo IN
      (SELECT o.OfferNo
       FROM Offering o
     WHERE o.OffTerm = 'FALL'
             AND o.OffYear = '2009')
Order by 1,2;
```

Problem #32 – Type 1 Subquery (nested multiple levels)

List student last and first names and majors for students who had at least one high grade (>= 3.5) in at least one course offered in winter of 2013 which was not taught by Leonard Vince (Use nested Type 1 Subqueries)

```
SELECT S.StdLastName, S.StdFirstName, S.StdMajor
FROM Student S INNER JOIN Enrollment E
ON S.StdNo= E.StdNo
WHERE E.EnrGrade >= 3.5
AND E.OfferNo in
       (SELECT O.OfferNo
                    FROM Offering O
                    WHERE O.OffTerm = 'WINTER'
                    AND O.OffYear = '2013'
                    AND O.FacNo not in
                                 (SELECT f.FacNo
                                        FROM Faculty F
                          WHERE F.FacFirstName = 'LEONARD'
             AND F.FacLastName = 'VINCE'));

No data can be retrieved for the above given year so trying with other years

SELECT S.StdLastName, S.StdFirstName, S.StdMajor
FROM Student S INNER JOIN Enrollment E
ON S.StdNo= E.StdNo
WHERE E.EnrGrade >= 3.5
AND E.OfferNo in
       (SELECT O.OfferNo
                    FROM Offering O
                    WHERE O.OffTerm = 'WINTER'
                    AND O.OffYear = '2010'
                    AND O.FacNo not in
                                 (SELECT f.FacNo
                                        FROM Faculty F
                          WHERE F.FacFirstName = 'LEONARD'
             AND F.FacLastName = 'VINCE'));
```

Problem #33 – Type 2 Subquery

Retrieve the faculty last and first names of faculty who are not students (use a Type 2 Subquery)

```
 SELECT f.FacLastName, f.FacFirstName
FROM Faculty f
WHERE NOT EXISTS( SELECT*
                   FROM Student s
                   WHERE s.StdNo=f.FacNo)
ORDER BY 1,2;
```

Problem #34 – Division Problem using Type 2 Subquery

List faculty last and first names of faculty who taught all of the fall of 2012 IS offerings

```
SELECT f.FacLastName, f.FacFirstName
 FROM Faculty f INNER JOIN Offering o
               ON f.FacNo=o.FacNo
 WHERE o.OffTerm='FALL'
      AND o.OffYear='2012'
      AND o.CourseNo like 'IS%'
GROUP BY f.FacLastName,f.FacFirstName
HAVING COUNT(*)= (SELECT COUNT(*)
                  FROM  Offering o1
                  WHERE o1.Offterm = 'FALL'
                  AND o1.OffYear = '2012'
       AND o1.CourseNo LIKE 'IS%')
       ORDER BY 1,2;
```

No data can be retrieved for the above given year so trying with other years

```
SELECT f.FacLastName, f.FacFirstName
 FROM Faculty f INNER JOIN Offering o
               ON f.FacNo=o.FacNo
 WHERE o.OffTerm='FALL'
      AND o.OffYear='2009'
      AND o.CourseNo like 'IS%'
GROUP BY f.FacLastName,f.FacFirstName
HAVING COUNT(*)= (SELECT COUNT(*)
                  FROM  Offering o1
                  WHERE o1.Offterm = 'FALL'
                  AND o1.OffYear = '2009'
       AND o1.CourseNo LIKE 'IS%')
       ORDER BY 1,2;
```

Problem #35 – Subquery in the FROM Clause aka "Table on the fly"

List the course number, course description, number of offerings, and the average enrollment across offerings

```
SELECT t.CourseNo,t.CrsDesc,
        COUNT(*) AS NumOfferings, AVG(T.EnrollCount) AS AvgEnroll
 FROM
     (SELECT c.CourseNo, c.CrsDesc,o.OfferNo, COUNT(*) AS EnrollCount
     FROM ( Course c INNER JOIN Offering o
                   ON c.CourseNo = o.CourseNo)
             INNER JOIN Enrollment e ON
                   o.OfferNo= e.OfferNo
                 GROUP BY c.CourseNo,c.CrsDesc,o.OfferNo
                 )t
                 GROUP BY t.CourseNo, t.CrsDesc
                 ORDER BY 1,2;
```