ACCT 6321 Database Applications for Business Analytics in Accounting

Fall 2022

Instructor: Dr. James Scott, PhD

Assignment #3 – MSSQL and Advanced SQL

20 Questions (5 Points Each) - 100 points

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 7 Problems – Introduction to Structured Query Language (SQL)

1. SELECT * FROM EMPLOYEE WHERE lower(EMP_LNAME) LIKE 'smith%'
   ORDER BY EMP_NUM;
2. SELECT P.PROJ_NAME, P.PROJ_VALUE, P.PROJ_BALANCE,
   E.EMP_LNAME, E.EMP_FNAME, E.EMP_INITIAL, E.JOB_CODE,
   J.JOB_DESCRIPTION, J.JOB_CHG_HOUR
   FROM PROJECT P
   LEFT JOIN EMPLOYEE E
   ON E.EMP_NUM = P.EMP_NUM
   LEFT JOIN JOB J
   ON P.JOB_CODE = E.JOB_CODE
   ORDER BY P.PROJ_VALUE;
3. SELECT P.PROJ_NAME, P.PROJ_VALUE, P.PROJ_BALANCE,
   E.EMP_LNAME, E.EMP_FNAME, E.EMP_INITIAL, E.JOB_CODE,
   J.JOB_DESCRIPTION, J.JOB_CHG_HOUR
   FROM PROJECT P
   LEFT JOIN EMPLOYEE E
   ON E.EMP_NUM = P.EMP_NUM
   LEFT JOIN JOB J
   ON P.JOB_CODE = E.JOB_CODE
   ORDER BY E.EMP_LNAME;
4. SELECT DISTINCT PROJ_NUM
   FROM ASSIGNMENT
   ORDER BY PROJ_NUM;
5. SELECT ASSIGN_NUM, EMP_NUM, PROJ_NUM, ASSIGN_CHARGE,
   ROUND(ASSIGN_CHG_HR * ASSIGN_HOURS,2) AS
   CACLUATED_ASSIGN_CHARGE,
   CASE WHEN ROUND(ASSIGN_CHARGE,2) <> ROUND(ASSIGN_CHG_HR *
   ASSIGN_HOURS,2) THEN ‘Not equal’ else ‘Equal’ END AS QUALITY_FLAG
   FROM ASSIGNMENT
   ORDER BY ASSIGN_NUM;
6. SELECT A.EMP_NUM, E.EMP_LNAME,
   SUM(A.ASSIGN_HOURS) AS SumOfASSIGN_HOURS,
   SUM(A.ASSIGN_CHARGE) AS SumOfASSIGN_CHARGE
   FROM EMPLOYEE E
   left join ASSIGNMENT A
   WHERE E.EMP_NUM = A.EMP_NUM
   GROUP BY A.EMP_NUM, A.EMP_LNAME
   SORT BY A.EMP_NUM;

7. SELECT PROJ_NUM, SUM(ASSIGN_HOURS) AS SumOfASSIGN_HOURS,
   SUM(ASSIGN_CHARGE) AS SumOfASSIGN_CHARGE
   FROM ASSIGNMENT
   GROUP BY PROJ_NUM;

Chapter 8 Problems – Advanced Structured Query Language (SQL)

1. CREATE TABLE EMP_1 (EMP_NUM CHAR(3) PRIMARY KEY,
   EMP_LNAME VARCHAR(15) NOT NULL,
   EMP_FNAME VARCHAR(15) NOT NULL,
   EMP_INITIAL CHAR(1),
   EMP_HIREDATE DATE,
   JOB_CODE CHAR(3),
   FOREIGN KEY (JOB_CODE) REFERENCES JOB);
2. INSERT INTO EMP_1 VALUES (101, 'News', 'John', 'G', '2000-11-08', 502);
   INSERT INTO EMP_1 VALUES (102, 'Senior', 'David', 'H', '1989-07-12', 501);
3. INSERT INTO emp_1 (EMP_NUM, EMP_LNAME, EMP_FNAME,
   EMP_INITIAL, EMP_HIREDATE,JOB_CODE)
   SELECT EMP_NUM, EMP_LNAME, EMP_FNAME,
   EMP_INITIAL, EMP_HIREDATE,JOB_CODE
   FROM EMPLOYEE
   WHERE EMP_NUM NOT IN (101,102);
4. BEGIN TRANSACTION;
   SELECT * FROM EMP_1;
   COMMIT;
5. UPDATE EMP_1 SET JOB_CODE='501' WHERE EMP_NUM='107';
6. DELETE FROM EMP_1
   WHERE EMP_LNAME = 'Smithfield' AND
   EMP_FNAME = 'William' AND
   EMP_HIREDATE = '22-Jun-04' AND JOB_CODE = '500';
7. CREATE TABLE EMP_2 AS
   (SELECT * FROM EMP_1);

Using MSSQL to answer the following seven (14-20) practical SQL questions using the same university data from Assignment 1 and 2.

Problem #14 – Aggregates that are grouped and subsetted (using a GROUP BY clause and a HAVING clause)

Retrieve the class name, minimum GPA, maximum GPA, average GPA, and average GPA plus 10% for each class but only for classes with an average GPA less than 3.5.

```
select STDCLASS,
min(STDGPA) as MinGPA,
max(STDGPA) as MaxGPA,
avg(STDGPA) as AvgGPA,
avg(STDGPA) *1.1 as AdjGPA
from STUDENT
group by STDCLASS
having avg(STDGPA) < 3.5
```

Problem #15 – Aggregates of a subset of rows that are grouped and subsetted (using a WHERE clause, a GROUP BY clause, and a HAVING clause)

Retrieve the class name, minimum GPA, maximum GPA, average GPA, and average GPA plus 10% for each class but only for non-IS majors and only for classes with an average GPA greater than 3 for non-IS majors.

```
select STDCLASS,
min(STDGPA) as MinGPA,
max(STDGPA) as MaxGPA,
avg(STDGPA) as AvgGPA,
avg(STDGPA) *1.1 as AdjGPA
from STUDENT
where STDMAJOR <> 'IS'
group by STDCLASS
having avg(STDGPA) < 3
```

Problem #16 – Cartesian Products, how many rows expected

Perform a Cartesian Product between tables Student, Offering, Enrollment, Course, and Faculty
How many columns are expected?
How many rows are expected?

```
SELECT S.*,O.*,E.*,C.*,F.*
FROM STUDENT S, OFFERING O, ENROLLMENT E, COURSE C, FACULTY F;
```

Column 34
Row 222222

Problem #17 – Cartesian Products, figuring out which rows match

Perform a Cartesian Product between tables Student, Offering, Enrollment, Course, and Faculty
Retrieve only the columns which are needed to show matching based on the relationship between the five tables and order in such a way as to tell the matching records.

```
SELECT  S.StdNo, E.StdNo, E.OfferNo , O.OfferNo,
        O.FacNo, F.FacNo, O.CourseNo, C.CourseNo
FROM Student S, Offering O, Enrollment E, Course C, Faculty F
ORDER BY S.StdNo , E.StdNo, E.OfferNo , O.OfferNo,
        F.FacNo DESC,O.FacNo DESC, O.CourseNo, C.CourseNo;
```

Problem #18 – Turning a Cartesian Product into an Inner Join by adding a WHERE clause to the Cross Product Syntax

Start with a Cartesian Product between tables Student, Offering, Enrollment, Course, and Faculty
Retrieve only the columns which are needed to show matching based on the relationship between the five tables and order in such a way as to tell the matching records
Add a WHERE clause to turn the Cartesian Product into an Inner Join.

```
SELECT  S.StdNo, E.StdNo, E.OfferNo , O.OfferNo,
            O.FacNo, F.FacNo, O.CourseNo, C.CourseNo
FROM Student S ,  Enrollment E , Offering O , Course C, Faculty F
WHERE S.StdNo = E.StdNo
AND
E.OfferNo = O.OfferNo
AND
O.CourseNo = C.CourseNo
AND
O.FacNo = F.FacNo
```

Problem #19 – Converting an Inner Join from Cross Product Syntax to Join Operator Syntax

Start with the Inner Join using Cross Product Syntax for the tables: Student, Offering, Enrollment, Course, and Faculty Convert to Join Operator Syntax.

```
SELECT  S.StdNo, E.StdNo, E.OfferNo , O.OfferNo,
            O.FacNo, F.FacNo, O.CourseNo, C.CourseNo
FROM Student S
INNER JOIN Enrollment E ON S.StdNo = E.StdNo
INNER JOIN Offering O ON E.OfferNo = O.OfferNo
INNER JOIN Faculty F ON O.FacNo = F.FacNo
INNER JOIN Course C ON O.CourseNo = C.CourseNo
```

Problem #20 – Combining Inner Join and WHERE, GROUP BY, and HAVING clauses

List the course number, offer number, and average grade of students enrolled in fall 2010 IS course offerings in which more than one student is enrolled.

```
SELECT  C.CourseNo, O.OfferNo, AVG(EnrGrade) AS AVG_GRADE
FROM Student S ,  Enrollment E , Offering O , Course C, Faculty F
WHERE
      S.StdNo = E.StdNo
   AND
      E.OfferNo = O.OfferNo
   AND
      O.CourseNo = C.CourseNo
   AND
      O.FacNo = F.FacNo
         AND
            O.OffTerm = 'FALL'
         AND
            OffYear = 2009
GROUP BY C.CourseNo, O.OfferNo
HAVING COUNT(S.StdNo) > 1
```