**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;

1. 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;

1. SELECT DISTINCT PROJ\_NUM

FROM ASSIGNMENT

ORDER BY PROJ\_NUM;

1. 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;

1. 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;

1. 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);

1. 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);

1. 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);

1. BEGIN TRANSACTION;

SELECT \* FROM EMP\_1;

COMMIT;

1. UPDATE EMP\_1 SET JOB\_CODE='501' WHERE EMP\_NUM='107';
2. DELETE FROM EMP\_1

WHERE EMP\_LNAME = 'Smithfield' AND

EMP\_FNAME = 'William' AND

EMP\_HIREDATE = '22-Jun-04' AND JOB\_CODE = '500';

1. 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 |