ACCT 6321 Database Applications for Business Analytics in Accounting

Fall 2022

Instructor: Dr. James Scott

Assignment #4 – SQL & NoSQL Problems

Each question is worth 5 points – Total 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 and NoSQL code
- 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

THIS HOMEWORK IS BROKEN INTO TWO SECTIONS: SQL and NoSQL
This assignment is not due until December 8th at COD.

Problem #1 – 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 Faculty f
INNER JOIN Student s
ON f.FacNo = s.StdNo;
```

Problem #2 – 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 F1.FACNO AS FACNO,
F1.FACFIRSTNAME AS FACFIRSTNAME,
F1.FACLASTNAME AS FACLASTNAME,
F1.FACSALARY AS FAC_SALARY,
F2.FACSALARY AS SUP_SALARY
FROM FACULTY F1, FACULTY F2
WHERE F1.FACSUPERVISOR = F2.FACNO AND F1.FACSALARY < F2.FACSALARY;
```

Problem #3 – 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 , o.CourseNo
from Faculty f
left join Offering o
```

```
on f.FacNo = o.FacNo
LEFT join Faculty f2
on f2.facNo = f.FacSupervisor
left join Offering o2
on o2.FacNo = f2.FacNo and o2.CourseNo = o.CourseNo
where o2.OffYear = 2013;
```

Problem #4 – 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 join Offering o
on c.CourseNo = o.CourseNo;
```

Problem #5 – 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 as OffYear , o.OffTerm as OffTerm ,
o.OfferNo as OfferNo ,
f.FacLastName  as FacLastName,
f.FacFirstName as FacFirstName
```

```
from Offering o

right join Course c

on o.CourseNo = c.CourseNo

left JOIN Faculty f

on o.FacNo = f.FacNo ;
```

Problem #6 – 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 offerNo, CourseNo, offTerm ,Offterm, CrsDesc, facNo, facLastName,
FacFirstName
from
(select o.OfferNo , o.CourseNo , o.OffTerm , c.CrsDesc , f.FacNo ,
f.FacLastName,f.FacFirstName , count(distinct e.StdNo) as student_enrolled
from Offering o
left join Course c
on o.CourseNo = c.CourseNo
left join Faculty f
on o.FacNo = f.FacNo
INNER join Enrollment e
on o.CourseNo = e.OfferNo
where f.FacDept = 'IS' and o.OffYear = 2013
group by o.OfferNo , o.CourseNo , o.OffTerm , c.CrsDesc , f.FacNo , f.FacLastName,
f.FacFirstName
having count(distinct e.StdNo) > 1
)a;
```

Problem #7 – Examining the difference between UNION and UNION ALL

Retrieve all faculty and students
Only show common columns in the result
Remove duplicates
Repeat query allowing duplicates

```
 select s.StdNo ,s.StdFirstName , s.StdLastName, s.StdCity , s.StdState , s.StdZip
from Student s
union
select f.FacNo , f.FacFirstName , f.FacLastName , f.FacCity , f.FacState , f.FacZipCode
from Faculty f ;

select s.StdNo ,s.StdFirstName , s.StdLastName, s.StdCity , s.StdState , s.StdZip
from Student s
union all
select f.FacNo , f.FacFirstName , f.FacLastName , f.FacCity , f.FacState , f.FacZipCode
from Faculty f ;
```

Problem #8 – 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
where s.StdNo in
(SELECT e.StdNo
from Student s
left join Enrollment e
on s.StdNo = e.StdNo
left join Offering o
on o.OfferNo  = e.OfferNo
where o.OffYear = 2012 and e.EnrGrade  >= 3.5
group by e.StdNo
having count(*) > 1);
```

Problem #9 – 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'));
```

Problem #10 – 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);
```

Problem #11 – 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.FacNo , f.FacFirstName , f.FacLastName
FROM Faculty f, Offering o
WHERE f.FacNo = o.FacNo
AND OffTerm = ' FALL ' AND OffYear = 2012
AND CourseNo LIKE 'IS%'
GROUP BY f.FacNo , f.FacFirstName , f.FacLastName
HAVING COUNT ( * ) = ( SELECT COUNT ( * ) FROM Offering
WHERE OffTerm = ' FALL '
AND OffYear = 2012 AND
CourseNo LIKE 'IS%' )
```

Problem #12 – 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  , CrsDesc ,
o.OfferNo , COUNT ( * ) AS EnrollCount
FROM Offering o, Enrollment e, Course c
WHERE o.OfferNo = e.OfferNo AND c.CourseNo = o.CourseNo GROUP BY
c.CourseNo , CrsDesc , o.OfferNo ) AS T
GROUP BY T.CourseNo , T.CrsDesc
```

Second portion of this assignment is to work on NoSQL queries. Using the Restaurants database write the following queries using NoSQL syntax.

1. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

```
db.restaurants.find( {$and:
[ {"cuisine" : {$ne :"American"}},
{"grades.score":{$gt:70}},
{"address.coord":{$lt: -65.754168}} ]
        } );
```

2. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168. Note : Do this query without using $and operator.

```
db.getCollection("restaurants").find(
   {
       "cuisine" : {
           "$ne" : "American"
       },
       "grades.score" : {
           "$gt" : NumberLong(70)
       },
       "address.coord.0" : {
           "$lt" : -65.754168
       }
```

```
  }
```

```
);
```

3. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

```
db.getCollection("restaurants").find(
    {
        "grades.grade" : "A",
        "cuisine" : {
            "$ne" : "American"
        },
        "borough" : {
            "$ne" : "Brooklyn"
        }
    },
    {
        "restaurant_id" : "$restaurant_id",
        "name" : "$name",
        "cuisine" : "$cuisine",
        "grades.grade" : "$grades.grade",
        "_id" : NumberInt(0)
    }
).sort(
    {
        "cuisine" : NumberInt(-1)
    }
);
```

4. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
db.getCollection("restaurants").find(
  {
    "name" : /^Wil.*$/i
  },
  {
    "restaurant_id" : "$restaurant_id",
    "name" : "$name",
    "borough" : "$borough",
    "cuisine" : "$cuisine",
    "_id" : NumberInt(0)
  }
);
```

5. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

```
db.getCollection("restaurants").find(
  {
    "name" : /^.*ces$/i
  },
  {
    "restaurant_id" : "$restaurant_id",
    "name" : "$name",
    "borough" : "$borough",
        "cuisine" : "$cuisine",
        "_id" : NumberInt(0)
    }
);
```

6. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

```
db.getCollection("restaurants").find(
   {
      "name" : /^.*reg.*$/i
   },
   {
      "restaurant_id" : "$restaurant_id",
      "name" : "$name",
      "borough" : "$borough",
      "cuisine" : "$cuisine",
      "_id" : NumberInt(0)
   }
);
```

7. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

```
db.getCollection("restaurants").find(
   {
      "borough" : "Bronx",
      "cuisine" : {
         "$in" : [
            "American",
            "Chinese"
         ]
```

```
        }
    },
    {
        "restaurant_id" : "$restaurant_id",
        "_id" : NumberInt(0)
    }
);
```

8. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.

```
db.getCollection("restaurants").find(
    {
        "borough" : {
            "$in" : [
                "Staten Island",
                "Queens",
                "Bronx",
                "Brooklyn"
            ]
        }
    },
    {
        "restaurant_id" : "$restaurant_id",
        "name" : "$name",
        "borough" : "$borough",
        "cuisine" : "$cuisine",
        "_id" : NumberInt(0)
    }
          );
```