2019 SQL Workshop
Bruin Actuarial Society

**Goal:** To calculate 2018 loss ratio. We will assume that claims information includes loss adjustment expenses. The following formulas will be useful:

\[
\text{Loss Ratio} = \frac{\text{Total Losses Paid}}{\text{Earned Premiums}}
\]

\[
\text{Earned Premium} = \text{Written Premium} \cdot (\text{Proportion of policy period elapsed})
\]

Proportion of policy period elapsed, in red, is the hard part. You are given the tables policy.csv and claim.csv. Your task is to familiarize yourself with SQL and to solve the problem above. The annual base rate is $2,200, and rate relativities for driving frequency and county are below. For your convenience, driving frequency is given in discrete bins instead of as an annual number of miles driven. Additionally, you may assume that for the entire duration a policy is in force, driving frequency and county do not change. (This can be easily verified.)

<table>
<thead>
<tr>
<th>Driving Frequency</th>
<th>Relativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1.15</td>
</tr>
<tr>
<td>Medium</td>
<td>1.00</td>
</tr>
<tr>
<td>Low</td>
<td>0.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>County</th>
<th>Relativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ventura</td>
<td>0.80</td>
</tr>
<tr>
<td>San Bernardino</td>
<td>0.85</td>
</tr>
<tr>
<td>Riverside</td>
<td>1.00</td>
</tr>
<tr>
<td>Orange</td>
<td>1.10</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>1.20</td>
</tr>
</tbody>
</table>

**Contents**

**Part I**

0. Introduction to databases and SQL  
   a. What is a database?  
   b. What is a relational database?  
   c. What is the relational model?  
   d. Why do I care?  
   e. Introduction to the Access environment  
1. Table Creation: Importing from CSVs
2. Selecting Data  
   a. Selecting columns  
   b. Only selecting certain rows  
   c. Selecting distinct rows  
   d. Ordering the data  
3. Variables  
4. Aggregating the data  
   a. AVG, COUNT, and SUM  
   b. Where: What doesn't work  
   c. Having: What works  

**Part II**

5. Table Creation and Inserting Data: Using SQL  
   a. Creation  
   b. Insertion  
   c. Updating  
   d. Deleting  
   e. Dropping  
   f. Creating a table from a query  
6. VBA  
   a. Basic Syntax: Creating Table  
   b. Running multiple lines at once: Populating Table  
7. Joins  
   a. Inner  
   b. Left  
   c. Multiple Joins  
8. Subqueries  
9. A difficult example with SWITCH: Calculating Earned Premiums and Loss Ratio  
   a. Calculating Exposures  
   b. Joining on Written Premium  
   c. Calculating Written Premium  
   d. Calculating Loss Ratio: A different type of subquery
1. **Table Creation: Importing from CSVs**
   a. Import policy.csv through **External Data → New Data Source → From File → Text File**. Since PolicyNumber contains duplicates let Access create the primary key.
   b. Import claim.csv through the same process. However, this time, we will choose ClaimNumber as our primary key since there are no duplicates.

2. **Selecting Data: Create → Query Design.** Play with the interface a bit and see what you can do. Then, go to View → SQL View.
   a. Selecting a column
      - `SELECT PolicyNumber FROM Policy;
   b. Selecting more than one column
      - `SELECT PolicyNumber, PolicyStartDate FROM Policy;
   c. Selecting all columns
      - `SELECT * FROM Policy;
   d. Only selecting certain rows
      - `SELECT * FROM Policy WHERE DrivingFrequency = "Low"
      - `SELECT * FROM Policy WHERE DrivingFrequency = "High" AND County = "Los Angeles"
      - `SELECT * FROM Policy WHERE DrivingFrequency = "High" OR County = "Los Angeles"
      - `SELECT PolicyNumber, PolicyStartDate, DrivingFrequency FROM Policy WHERE County IN ("Los Angeles", "Riverside")
   e. Selecting distinct rows
      - `SELECT DISTINCT PolicyNumber, DrivingFrequency, County FROM Policy WHERE PolicyNumber = "P100000"
      - `SELECT DISTINCT PolicyNumber, County FROM Policy WHERE County NOT IN ("Ventura", "Orange")
   f. Ordering the data
      - `SELECT * FROM Claim ORDER BY PolicyNumber;
      - `SELECT * FROM Claim ORDER BY LossDate DESC;
   g. **Exercises (Presented without solution)**
      - Select all policies with low driving frequency in LA or Orange county. How many are there? Note that each policy gets a new entry upon renewal—be sure to account for this.
      - Select all policies with that have low or medium driving frequency or are not located in LA, Orange, or San Bernardino county.

3. **Variables**
   - `SELECT * FROM policy WHERE YEAR(PolicyStartDate) = myyear;

4. **Aggregating the data**
   a. **SUM**
      - `SELECT SUM(ClaimAmount) FROM Claim;
      - `SELECT SUM(ClaimAmount) AS TotalClaims FROM Claim;
   b. **COUNT**
      - `SELECT PolicyNumber, COUNT(ClaimNumber) AS ClaimCount FROM Claim GROUP BY PolicyNumber;
   c. **AVG**
• SELECT YEAR(LossDate) AS LossYear, \text{\texttt{AVG}}(\text{ClaimAmount}) AS Severity FROM Claim GROUP BY YEAR(LossDate) ORDER BY YEAR(LossDate);

d. **Exercises (Presented without solution)**
   - What is the total amount of claims reported in each year?
   - What is the total amount of claims occurring in each year?
   - Consider only claims that occurred in 2014. How much was reported in each year?

e. **Where: What doesn't work**
   - SELECT PolicyNumber, SUM(ClaimAmount) AS TotalClaims FROM Claim WHERE \text{\texttt{TotalClaims}} < 1000 GROUP BY PolicyNumber;
   - SELECT PolicyNumber, SUM(ClaimAmount) AS TotalClaims FROM Claim WHERE SUM(ClaimAmount) < 1000 GROUP BY PolicyNumber;

f. **Having: What works**
   - SELECT PolicyNumber, SUM(ClaimAmount) AS TotalClaims FROM Claim GROUP BY PolicyNumber HAVING SUM(ClaimAmount) < 1000;

g. **Exercises (Presented without solution)**
   - Use the previous example to determine how many claims each policy would have if we implemented a $1,000 policy deductible.
     - Hint: A $1,000 eliminates all policies with total claims under $1,000. For policies with claims under $1,000, their total claim amount will be reduced by $1,000.
   - How many policies had total claims over $60,000? What are their policy numbers?
   - Use the results from the previous part to determine how many claims each of these policies had. Which policy has the highest severity? What is it?
5. Table Creation and Inserting Data: Using SQL
   a. Creation
      • `CREATE TABLE` DrivingFrequency (DrivingFrequency CHAR, Relativity DOUBLE);
      • CREATE TABLE CreditScore(CreditScore INT, Relativity DOUBLE);
   b. Insertion
      • `INSERT INTO` DrivingFrequency (DrivingFrequency, Relativity)
        VALUES ("Low", 0.85);
      • INSERT INTO DrivingFrequency (DrivingFrequency, Relativity)
        VALUES ("Medium", 1);
      • INSERT INTO DrivingFrequency (DrivingFrequency, Relativity)
        VALUES ("High", 1.1);
      • INSERT INTO DrivingFrequency (DrivingFrequency, Relativity)
        VALUES ("Really High", 1.3);
   c. Updating
      • `UPDATE` DrivingFrequency SET Relativity = 1.15 WHERE
        DrivingFrequency = "High";
   d. Deleting
      • `DELETE FROM` DrivingFrequency WHERE DrivingFrequency NOT IN
        ("Low", "Medium", "High")
   e. Dropping
      • `DROP TABLE` CreditScore;
   f. Creating a table from a query
      • SELECT DISTINCT PolicyNumber, DrivingFrequency, County INTO
        UniquePolicies FROM Policy;

6. VBA: Database tools → Visual Basic. Insert Macro and write a Subroutine.
   a. Basic Syntax: Creating Table
      • `Sub` CreateCountyTable()
        Dim sql As String
        sql = "CREATE TABLE County (County CHAR, Relativity DOUBLE);"
        DoCmd.RunSQL sql
      End Sub
   b. Running multiple lines at once: Populating Table
      • `Sub` PopulateCountyTable()
        DoCmd.RunSQL "INSERT INTO County (County, Relativity)
        VALUES ("Ventura", 0.8);"
        DoCmd.RunSQL "INSERT INTO County (County, Relativity)
        VALUES ("San Bernardino", 0.85);"
        DoCmd.RunSQL "INSERT INTO County (County, Relativity)
        VALUES ("Riverside", 1);"
        DoCmd.RunSQL "INSERT INTO County (County, Relativity)
        VALUES ("Orange", 1.1);"
        DoCmd.RunSQL "INSERT INTO County (County, Relativity)
        VALUES ("Los Angeles", 1.2);"
7. Joins

a. Inner
   - SELECT Claim.*, UniquePolicies.County
     FROM Claim INNER JOIN UniquePolicies
     ON Claim.PolicyNumber = UniquePolicies.PolicyNumber;
   - SELECT p.County, SUM(c.ClaimAmount) AS TotalClaims
     FROM Claim AS c INNER JOIN UniquePolicies AS p
     ON c.PolicyNumber = p.PolicyNumber
     GROUP BY p.County;

b. Left
   - SELECT p.PolicyNumber, p.DrivingFrequency, p.County,
     SUM(c.ClaimAmount) AS TotalClaims
     FROM Policy as p LEFT JOIN Claim as c
     ON p.PolicyNumber = c.PolicyNumber
     GROUP BY p.PolicyNumber, p.DrivingFrequency, p.County;
   - SELECT p.PolicyNumber, p.DrivingFrequency, p.County,
     SUM(NZ(c.ClaimAmount, 0)) AS TotalClaims
     FROM Policy as p LEFT JOIN Claim as c
     ON p.PolicyNumber = c.PolicyNumber
     GROUP BY p.PolicyNumber, p.DrivingFrequency, p.County;

c. Multiple Joins
   - SELECT DISTINCT p.PolicyNumber, 2200 AS BaseRate,
     df.Relativity AS DrivingFreqRel, cty.Relativity AS CountyRel
     FROM ((Policy AS p
     LEFT JOIN DrivingFrequency AS df ON p.DrivingFrequency = df.DrivingFrequency)
     LEFT JOIN County as cty ON p.County = cty.County);

d. Exercises (Presented without solution)
   - Select the initial policy start date (renewals do not count as an initial policy start date) and
date of first reported claim for all policies with claims.
     • Hint: you may find the aggregate function MIN useful.
   - For accident year 2016, what is the severity in each county? For each driving frequency?
     • Accident year 2016 refers to claims that occurred between 1/1/2016 and 12/31/2016.
   - For report year 2016, what is the severity in each county? For each driving frequency?
     • Report year 2016 refers to claims that were reported between 1/1/2016 and
       12/31/2016.
   - Modify a previous example to select all policies that had no claims. How many are there?

8. Subqueries

a. How they work
   - SELECT PolicyNumber, BaseRate * DrivingFreqRel * CountyRel AS WrittenPrem FROM
     (SELECT DISTINCT p.PolicyNumber, 2200 AS BaseRate,
      df.Relativity AS DrivingFreqRel, cty.Relativity AS CountyRel
     FROM ((Policy AS p

LEFT JOIN DrivingFrequency AS df ON p.DrivingFrequency = df.DrivingFrequency
LEFT JOIN County as cty ON p.County = cty.County)

• SELECT PolicyNumber, BaseRate * DrivingFreqRel * CountyRel AS WrittenPrem INTO WrittenPremium FROM
  (SELECT DISTINCT p.PolicyNumber, 2200 AS BaseRate, df.Relativity AS DrivingFreqRel, cty.Relativity as CountyRel
  FROM ((Policy AS p
  LEFT JOIN DrivingFrequency AS df ON p.DrivingFrequency = df.DrivingFrequency)
  LEFT JOIN County as cty ON p.County = cty.County));

b. Exercises (Presented without solution)

• Refer to the previous exercise where we selected all policies that had no claims. How many are there in each county? In each driving frequency?

• Note that a subquery can return a single result. Use this fact to obtain the claim data for the policy with the largest single loss.
  o Hint: You may find the aggregate function MAX useful.
  o Hint: If a subquery returns a single value, you can use that value in WHERE statements to compare.

• Modify the previous exercise to also obtain the county and driving frequency for this policy. Note that this could be done with joins or with subqueries.
9. A difficult example with SWITCH: Calculating Earned Premiums and Loss Ratio
   a. Calculating Exposures
      • SELECT year, PolicyNumber, PolicyStartDate,
        SWITCH(PolicyStartDate <= DateSerial(year - 1, 1, 1), 0,
               PolicyStartDate <= DateSerial(year - 1, 12, 31),
               DateDiff("d", DateSerial(year - 1, 1, 1), PolicyStartDate),
               PolicyStartDate <= DateSerial(year, 12, 31),
               DateDiff("d", PolicyStartDate, DateSerial(year + 1, 1, 1)),
               PolicyStartDate > DateSerial(year, 12, 31), 0) AS exposures
        FROM policy
   b. Joining on Written Premium
      • SELECT p.PolicyNumber, p.PolicyStartDate, p.year AS RunYear,
        p.exposures/365 AS exposures, wp.WrittenPrem
        FROM (SELECT year, PolicyNumber, PolicyStartDate,
               SWITCH(PolicyStartDate <= DateSerial(year - 1, 1, 1), 0,
                      PolicyStartDate <= DateSerial(year - 1, 12, 31),
                      DateDiff("d", DateSerial(year - 1, 1, 1), PolicyStartDate),
                      PolicyStartDate <= DateSerial(year, 12, 31),
                      DateDiff("d", PolicyStartDate, DateSerial(year + 1, 1, 1)),
                      PolicyStartDate > DateSerial(year, 12, 31), 0) AS exposures
               FROM policy) AS p
        LEFT JOIN WrittenPremium AS wp ON p.PolicyNumber = wp.PolicyNumber
        WHERE p.exposures > 0;
   c. Calculating Written Premium
      • SELECT "12-31-" & year AS RunDate, SUM(exposures * WrittenPrem) AS EarnedPrem
        FROM (SELECT p.PolicyNumber, p.PolicyStartDate, p.year AS RunYear,
               p.exposures/365 AS exposures, wp.WrittenPrem
               FROM (SELECT year, PolicyNumber, PolicyStartDate,
                      SWITCH(PolicyStartDate <= DateSerial(year - 1, 1, 1), 0,
                             PolicyStartDate <= DateSerial(year - 1, 12, 31),
                             DateDiff("d", DateSerial(year - 1, 1, 1), PolicyStartDate),
                             PolicyStartDate <= DateSerial(year, 12, 31),
                             DateDiff("d", PolicyStartDate, DateSerial(year + 1, 1, 1)),
                             PolicyStartDate > DateSerial(year, 12, 31), 0) AS exposures
                      FROM policy) AS p
               LEFT JOIN WrittenPremium AS wp ON p.PolicyNumber = wp.PolicyNumber
               WHERE p.exposures > 0);
   d. Calculating Loss Ratio: A different type of subquery
      • SELECT "12-31-" & year AS RunDate, (SELECT Sum(ClaimAmount) FROM Claim
        WHERE Year(ReportDate) = year)/SUM(exposures * WrittenPrem) AS LossRatio
        FROM (SELECT p.PolicyNumber, p.PolicyStartDate, p.year AS RunYear,
               p.exposures/365 AS exposures, wp.WrittenPrem
               FROM (SELECT year, PolicyNumber, PolicyStartDate,
                      SWITCH(PolicyStartDate <= DateSerial(year - 1, 1, 1), 0,
                             PolicyStartDate <= DateSerial(year - 1, 12, 31),
                             DateDiff("d", DateSerial(year - 1, 1, 1), PolicyStartDate),
                             PolicyStartDate <= DateSerial(year, 12, 31),
                             DateDiff("d", PolicyStartDate, DateSerial(year + 1, 1, 1)),
                             PolicyStartDate > DateSerial(year, 12, 31), 0) AS exposures
                      FROM policy) AS p
               LEFT JOIN WrittenPremium AS wp ON p.PolicyNumber = wp.PolicyNumber
               WHERE p.exposures > 0);