ORIE 3120
Industrial Data and Systems Analysis
Recitation 3
Joins

P. Jackson, L. Seligman, P. Frazier

Educational Objectives

1. Understand how to join the contents of two tables with both "Where" clauses and "Joins"
2. Understand when you need to refer to a table name when selecting a field
3. Understand table name aliases

Basic Concepts

You must have completed Recitations 1 and 2 before attempting this recitation

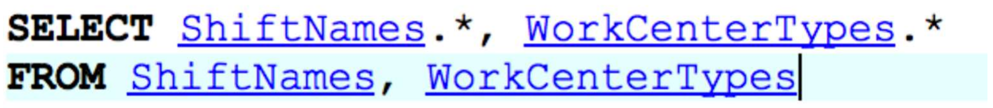
Getting Started

1. To receive credit for attendance, copy this answer sheet onto your computer and fill in the answers to the questions asked as you go through the recitation. Then, show it to your TA when you are done and ask her or him to mark down your netid. You can also submit the sheet (saved as a pdf) electronically before the start of your recitation. You may work by yourself, or with one other person, but please do not work in groups larger than two.
2. Open SQLite Studio, version 3.1.1 or higher.
3. Open an existing database by selecting the option "Add database" under the "Databases" tab. Download and select the file "ThruputHistory.sqlite".

Multiple Table Queries and Joins

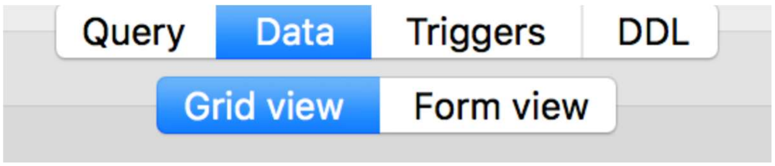
Exercise 1:

1. Create a new view named "Rec3Query1" and include the following code to be executed.



```
SELECT ShiftNames.*, WorkCenterTypes.*
FROM ShiftNames, WorkCenterTypes
```

   Note: the asterisk ("*") adds all the fields from the selected table to the query.
2. We have created a multiple table query simply by listing different tables separated by a comma after the **FROM** keyword. We can refer to any field from each of these tables after the **SELECT** keyword by specifying from which table you are requesting a field.
3. View your results by double clicking this query. It should be listed under "Views" within the ThruputHistory database. Make sure you are viewing the "Results" in "Grid View" as shown in the screen shot below.



4. How many rows are there in the resulting dataset? ___________________
5. Explain why there are this many rows:

   ____

   ____

6. Create another query titled "Rec3Query2" which executes the following SQL code:

```
SELECT ProductionHistory.*, ShiftNames.*
FROM ProductionHistory, ShiftNames
```

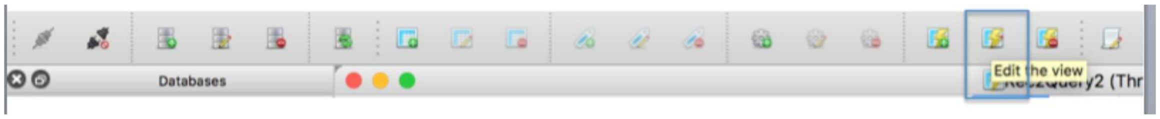
7. The table "ProductionHistory" has 840 rows and the table "ShiftNames" has 4 rows. How many rows are in the resulting dataset? ______________
8. Explain why there are this many:

   ____

   ____

   We are given more rows than are relevant. We are only interested in rows where the fields are related to each other in terms of "ShiftID". We could be more selective by adding a **WHERE** criterion.

9. Edit Rec3Query2 by clicking on the query and clicking the icon "Edit view". This icon is next to the "New view" icon, and it has a box around it in the screen shot below:



   Add the following **WHERE** criterion to your query and click "Commit" (the green check mark):

```
WHERE ProductionHistory.ShiftID=ShiftNames.ShiftID
```

10. Create another query based on table "ProductionHistory" that only shows the following fields: ProductID, WeekNo, WCType, WorkcenterID. Also gather the data from the field "Shift" from table "ShiftNames" where "Shift" is the name associated with the proper "ShiftID" as specified in the "ProductionHistory" table. (Hint: You will need a **WHERE** criterion.)
11. Write your query here, and name it "Rec3Query3". Make sure that the resulting dataset has exactly 840 rows.

    ____

    ____

    ____

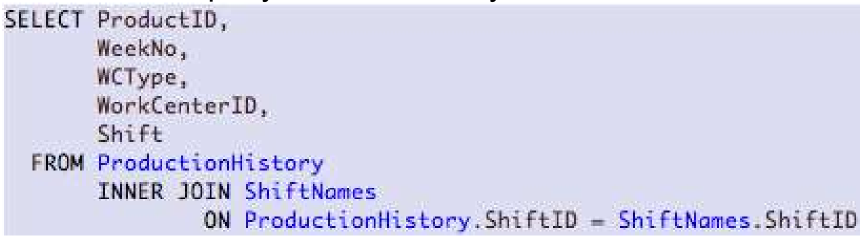
12. The field "ShiftID" occurs in both tables of your multiple table query, so this field name would be ambiguous without the prefix specifying where to access this data. Note that you do not always need to specify which table you are accessing fields from when the field names are not ambiguous.

Exercise 2:

1. There can be more than one way to execute a query in SQL. Joining tables together based on a common field (or fields) is very common, and as a result SQL has a special clause to handle this operation, the **JOIN** operator.

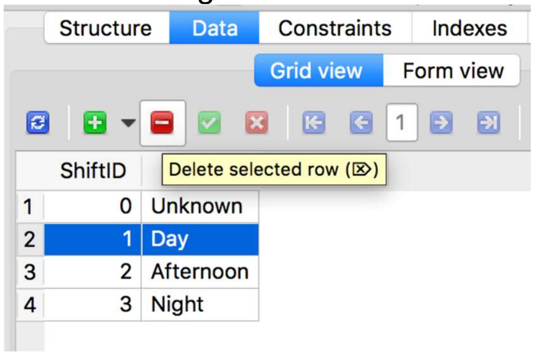
   A **JOIN** clause is of the following format: "*tablename1* **INNER JOIN** *tablename1* **ON** *tablename1.field1 = tablename2.field2*"

   Steps 2-4 will outline the steps to perform this operation.
2. Create a new query titled "Rec3Query4" referring to tables "ProductionHistory" and "ShiftNames". This query will use the **JOIN** operator to join these two tables on the unique key "ShiftID". We will want to access the following fields from "ProductionHistory": ProductID, WeekNo, WCType, WorkCenterID; we will want to access the field "Shift" from table "ShiftNames".
3. The resulting dataset should be identical to the results from the updated "Rec3Query2" using the **WHERE** clause. Verify that this dataset is indeed identical and has 840 rows.
4. Here is the correct query for "Rec3Query4":



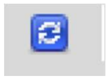
```
SELECT ProductID,
       WeekNo,
       WCType,
       WorkCenterID,
       Shift
  FROM ProductionHistory
       INNER JOIN ShiftNames
               ON ProductionHistory.ShiftID = ShiftNames.ShiftID
```

   Note that the table names were omitted wherever the field name was unambiguous.
5. View the contents of the table "ShiftNames" by double clicking on the table name and viewing the "Data" tab in "Grid View".
6. Delete the row with "ShiftID=1" and "Shift=Day" by selecting the appropriate row, clicking the red minus sign to delete, and then the green checkmark to commit the change.



	ShiftID	
1	0	Unknown
2	1	Day
3	2	Afternoon
4	3	Night

7. Now reopen "Rec3Query4" (it should be listed under "Views" within the ThruputHistory database). Hit the refresh button (see this screenshot:)



8. How many rows does the resulting dataset have after the deletion?
   ____
   Explain why this dataset has fewer rows:
   ____
   ____

Exercise 3:
1. We now want to display all rows of the “ProductionHistory” table with appropriate shift names, even when it contains a “ShiftID” that isn’t in the table “ShiftNames”. The following steps outline this action.
2. Open view “Rec3Query4” for editing. Copy the SQL code and create a new view “Rec3Query5”. Paste the SQL code from Rec3Query4, since we will be editing the **JOIN** clause.
3. We will use a **LEFT JOIN** which will include all records from “ProductionHistory” and only the records from “ShiftNames” where the join condition is met. Change the join in this query to a **LEFT JOIN**. Open the resulting dataset. Verify that the resulting dataset again has 840 rows. What is displayed under “Shift” when “ProductionHistory.ShiftID=1”?
   ____
4. Write down the query you ran in step 3:
   ____
   ____
   Note that you do not need to specify the table name “ProductionHistory” in front of a field name anywhere where it is not ambiguous.
5. You can now close this query. Re-open the table “ShiftNames” and add the row we deleted earlier (“ShiftID=1” and “Shift=Day”) by clicking the red plus icon and filling the data in the appropriate fields.

Exercise 4:
1. Create a new view titled “Rec3Query6LostThruputHistoryWithNames” containing the following fields: “LostThruputHistory.WeekNo, WorkCenterTypes.WCDescription, LostThruputHistory.WorkCenterID, ProductNames.LongName, ShiftNames.Shift, ProblemType.ProblemTypeDescription, LostThruputHistory.Description, LostThruputHistory.NumOccurrences, LostThruputHistory.LostThroughputMinutes, LostThruputHistory.Level1Codes, LostThruputHistory.Level2CodesA, LostThruputHistory.Level2CodesB, LostThruputHistory.Level2CodesC, LostThruputHistory.Level3CodesA” in such a way that every row of “LostThruputHistory” is visible (**LEFT JOIN**) exactly once (exactly 3188 rows).
2. Here is the appropriate from clause:

```
FROM LostThruputHistory
    LEFT JOIN ProblemType
        ON LostThruputHistory.ProblemType=ProblemType.ProblemType
    LEFT JOIN ShiftNames
        ON LostThruputHistory.ShiftID=ShiftNames.ShiftID
    LEFT JOIN WorkCenterTypes
        ON LostThruputHistory.WCType=WorkCenterTypes.WCType
    LEFT JOIN ProductNames
        ON LostThruputHistory.ProductID = ProductNames.Name
```

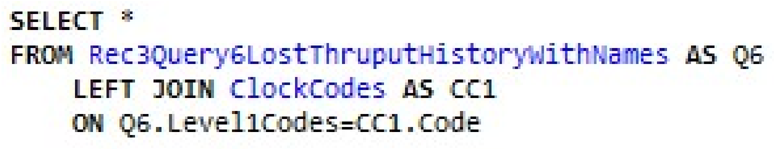
3. Run this query to see a more readable version of the “LostThruputHistory” table.

Table Name Aliases

Using long table names when creating tables, queries, and fields make it easier to make sense of your database when you look at it weeks. These long names can add time to the query-writing process though, and one way around this is using short names as aliases.

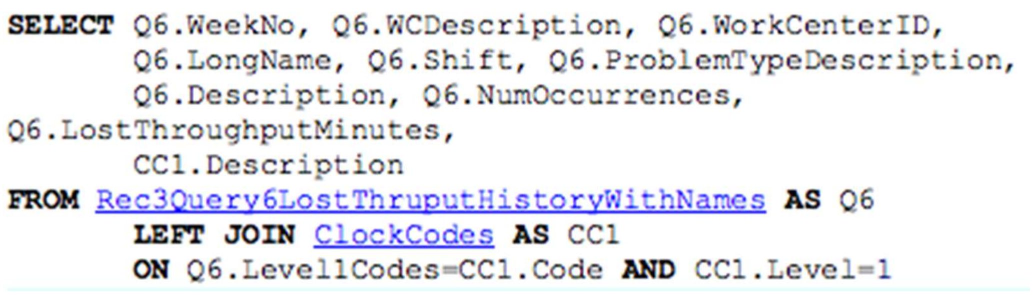
Exercise 1:

1. We now want to take the query “Rec3Query6” and replace the problem codes “Level1Codes”, “Level2CodesA, …” with their respective descriptions from the table “ClockCodes”. Create a new view titled “Rec3Query7LostThruputHistoryWithNamesAndLevel1Descriptions” and start off this view as:



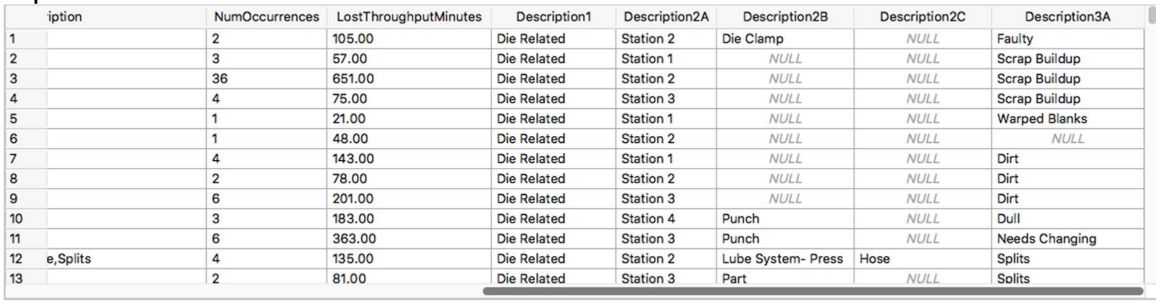
```
SELECT *
FROM Rec3Query6LostThruputHistoryWithNames AS Q6
    LEFT JOIN ClockCodes AS CC1
    ON Q6.Level1Codes=CC1.Code
```

   Note the use of the SQL keyword **AS** used to create table aliases.
2. Now edit the query to select only the following fields: Q6.WeekNo, Q6.WCDescription, Q6.WorkCenterID, Q6.LongName, Q6.Shift, Q6.ProblemTypeDescription, Q6.Description, Q6.NumOccurrences, Q6.LostThroughputMinutes, CC1.Description
3. Open the resulting dataset. How many rows are there? _______________
4. There are only 3,188 rows in the resulting dataset from Q6. This new query has more rows in the dataset because there are be multiple records in “ClockCodes” with the same value in the field “Code”, since “Code” is not a unique identifier. Code is only unique for records with the same value for Level.
5. Note that the field “Level1Codes” refers to codes with “Level=1”. Add an additional condition to the left join to ensure that CC1.Level is 1. Here is the complete query:



```
SELECT Q6.WeekNo, Q6.WCDescription, Q6.WorkCenterID,
       Q6.LongName, Q6.Shift, Q6.ProblemTypeDescription,
       Q6.Description, Q6.NumOccurrences,
Q6.LostThroughputMinutes,
       CC1.Description
FROM Rec3Query6LostThruputHistoryWithNames AS Q6
       LEFT JOIN ClockCodes AS CC1
       ON Q6.Level1Codes=CC1.Code AND CC1.Level=1
```

6. Open the resulting dataset and verify that it has exactly 3,188 rows.
7. In the homework we will repeat the process of replacing the other codes in this view result with the corresponding descriptions from ClockCodes.
8. Create a new view called "Rec3Query8LostThruputHistoryWithNamesAndAllLevelDescriptions". In this view, write a new query that is similar to the one in Q7 but that substitutes the entry from the Description field from ClockCodes for *all* of the fields Level1Codes, Level2CodesA, Level2CodesB, Level2CodesC, and Level3CodesA. In your view output, rename these fields Description1, Description2A, Description2B, Description2C, and Description3A respectively. Keep in mind that Level2CodesA, Level2CodesB, and Level2CodesC all correspond to a record in ClockCodes where Level=2. Similarly, Level3CodesA corresponds to a record in ClockCodes where Level=3. Also keep in mind that some records have NULL values for Level2CodesA, Level2CodesB, Level2CodesC, or Level3CodesA. These records should still appear in your result, with a NULL value in the corresponding description.
9. Run your query, and check that it has the same number of records as Rec2Query7LostThruputHistoryWithNamesAndLevel1Descriptions. Here is a screenshot of some of the fields and records that your query will report:



	iption	NumOccurrences	LostThroughputMinutes	Description1	Description2A	Description2B	Description2C	Description3A
1		2	105.00	Die Related	Station 2	Die Clamp	NULL	Faulty
2		3	57.00	Die Related	Station 1	NULL	NULL	Scrap Buildup
3		36	651.00	Die Related	Station 2	NULL	NULL	Scrap Buildup
4		4	75.00	Die Related	Station 3	NULL	NULL	Scrap Buildup
5		1	21.00	Die Related	Station 1	NULL	NULL	Warped Blanks
6		1	48.00	Die Related	Station 2	NULL	NULL	NULL
7		4	143.00	Die Related	Station 1	NULL	NULL	Dirt
8		2	78.00	Die Related	Station 2	NULL	NULL	Dirt
9		6	201.00	Die Related	Station 3	NULL	NULL	Dirt
10		3	183.00	Die Related	Station 4	Punch	NULL	Dull
11		6	363.00	Die Related	Station 3	Punch	NULL	Needs Changing
12	e,Splits	4	135.00	Die Related	Station 2	Lube System- Press	Hose	Splits
13		2	81.00	Die Related	Station 3	Part	NULL	Splits

10. Write the query that you created for Rec3Query8LostThruputHistoryWithNamesAndAllLevelDescriptions here.

____

____

____

____

____

____

Review

We have discussed database design and discovered the need to join the content of two tables together based on key fields. By nesting queries and using a combination of **JOIN** and **WHERE** clauses, we are able to create queries that replace all the connecting fields with description fields. We used this to make the resulting dataset more readable. In the following sections we will begin to analyze this data and we will be less concerned with readability. You can utilize the tools from the beginning of this recitation to make the following datasets more readable if you desire.