ORIE 3120
Industrial Data and Systems Analysis
Recitation 2
Database Schemas and GROUP BY
P. Jackson, L. Seligman, P. Frazier

Educational Objectives

1. Understand primary and foreign keys
2. Understand how to group data by a specific condition
3. Understand aggregation queries and aggregate operators
4. Understand terminology from manufacturing, and how data might be used in a performance improvement process to improve throughput

Basic Concepts

You must have completed Recitation 1 before attempting this recitation

1. Your teaching assistant will explain or review the following concepts
    a. Throughput
    b. Yield loss
    c. Speed loss
    d. Line transitions

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” from blackboard.

Relational Database Design

In a relational database, you should store information in the most efficient way possible by minimizing information duplication. For example, if you are logging phone conversations with a customer named James Bond, you might log his name as “James Bond”, “JBond”, or “Bond, James”. When it comes time to analyze the log, you will have difficulty identifying all entries corresponding to your communication with that customer. It would be better to have a separate table of customers with unique identifiers for each customer. This way, the customer would be in the table of customers exactly once, say as CustomerID:

007, Name: Bond, James. This way, in the log file, every time you talk to the customer, you log it as a conversation with Customer ID 007.

The database created today uses this approach. There are separate tables to describe clock codes, problem types, shift names, product names, and work center types. Each row in most of these tables has a primary key, which is a unique identifier [If you double click a table, you will be brought to the "Structure" tab. If a field has a key icon in the "Primary Key" column, this field is a primary key]. Some tables also have fields that link to fields in other tables, which are foreign keys [If you look in the "Structure" tab for a table, these fields will have a "linked tables" icon under the "Foreign Key" column.]

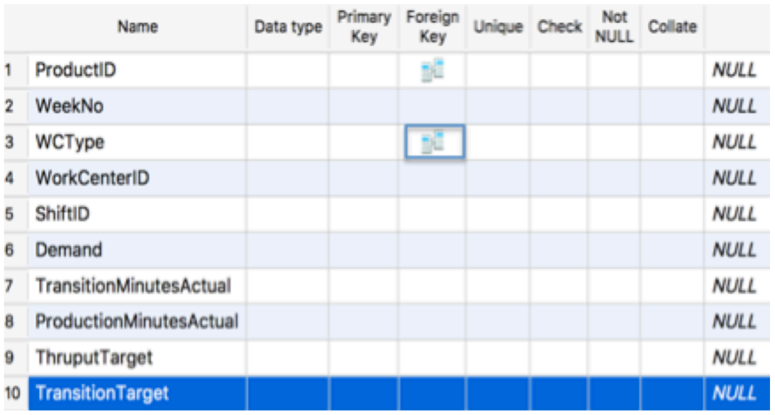
Primary and Foreign Keys

Exercise 1:

1. Connect to the "ThruputHistory" database by double clicking on the title or clicking on the "Connect" icon. Click on "Tables" to view the tables in this database. You can double click on a table title to view more details about the contents of this table.
2. What is the primary key for the table "ShiftNames"?

   ____
3. What is the primary key for the table "ProblemType"?

   ____
4. Double click the table "ProductionHistory" table to view its structure. Make sure you are viewing the table in the "Structure" tab, as shown in the screen shot below.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	ProductID			●					NULL
2	WeekNo								NULL
3	WCType			●					NULL
4	WorkCenterID								NULL
5	ShiftID								NULL
6	Demand								NULL
7	TransitionMinutesActual								NULL
8	ProductionMinutesActual								NULL
9	ThruputTarget								NULL
10	TransitionTarget								NULL

   Fields with an icon under the Foreign Key column are foreign keys, i.e., refer to data in other tables. Double click on a field to view which table it is accessing information from. (Hint: Click "Configure" to view where this data is being pulled from.)
5. List the fields in "ProductionHistory" that are foreign keys, and list the table and field each links to. (For example, the field "ProductID" refers to the field "Name" in the table "ProductNames").

   ____

   ____

6. In the table “LostThruputHistory”, list the fields that are foreign keys, and list the table and field each links to.

    ____

    ____

Review:

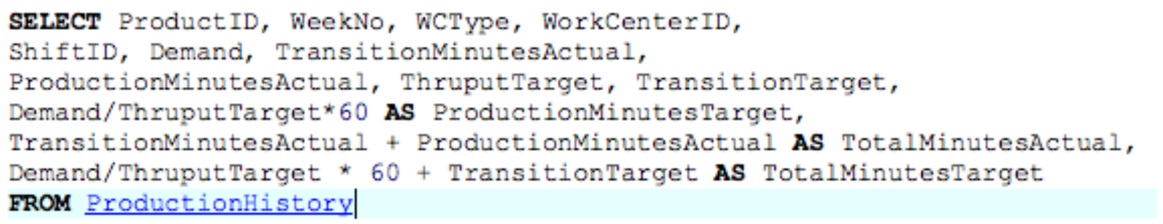
1. As a general principle in database design, you will assign a data item to exactly one table. If you need to store a customer’s phone number, make a table “Customers” where each customer corresponds to a row in the table, and include a field in that table to store the phone number. Don’t put the phone number for this customer in any other table, and this way you can access it from this table.
2. You might think it would be nice to have the customer’s phone number in the order file so that when you look at an order you can view this data, but this way you would store the number in multiple places. If the customer changes their phone number, you would have to correct it in multiple places in your database. Instead, allow other tables to have fields that refer to the unique identifier in the “Customers” table to acquire a customer’s telephone number.
3. If you are doing analysis work and your queries are taking a long time to run, you might save your query results in a table as intermediate results and this would result in saving data in multiple places. This would create a duplication of data, meaning you could delete this intermediate tables without losing data, so when you correct information in these tables you can delete them and reconstruct them anytime from the underlying database.

The GROUP BY Clause

Exercise 1:

The “ProductionHistory” table is a log of each product’s production by work center, week, and shift. We wish to analyze the data in this table to see where we should concentrate our throughput improvement efforts. The table contains five fields related to throughput. Using these fields, we can compute the total time spent on the production and the target time that should have been spent by each product, work center, week, and shift.

1. Create the following view and title it “Q100TotalMinutesActualAgainstTargetDetail”:

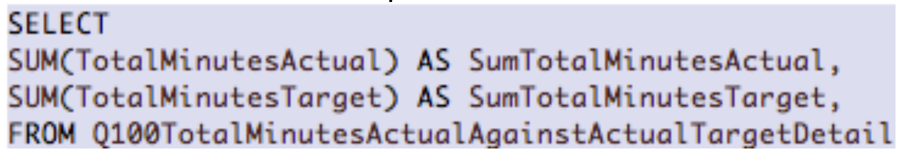


```
SELECT ProductID, WeekNo, WCType, WorkCenterID,
ShiftID, Demand, TransitionMinutesActual,
ProductionMinutesActual, ThruputTarget, TransitionTarget,
Demand/ThruputTarget*60 AS ProductionMinutesTarget,
TransitionMinutesActual + ProductionMinutesActual AS TotalMinutesActual,
Demand/ThruputTarget * 60 + TransitionTarget AS TotalMinutesTarget
FROM ProductionHistory
```

To understand this query, note the following about ProductionHistory:

- Demand describes the total number of pieces of the given product that were processed by the machine in the given workcenter during the given shift.
- ProductionMinutesActual describes time that the machine was actually performing useful work.
- TransitionMinutesActual describes time that the machine was transitioning between two different jobs (e.g., to change dies in a press).
- ThruputTarget describes how many pieces of this product we think we should be able to process in an hour of production time (not including transition time) in an ideal world. So Demand/ThruputTarget is how many hours of production time we would need in an ideal world to process the pieces in demand.
- TransitionTarget is how many minutes of transition time we think we need in an ideal world during the time period described by this data.
- Thus, Demand/ThruputTarget*60 + TransitionTarget is how many total minutes we think we need (including production and transitions) in an ideal world to process the number of pieces given by Demand.

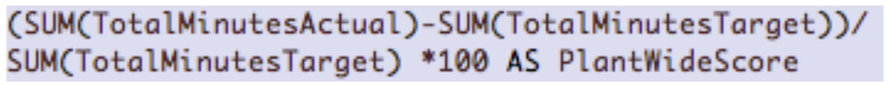
2. View the resulting dataset and verify that there are 840 rows.
3. Create another query which uses the results of the following query and save it as “Q110PlantWideThruputPerformance”:



```
SELECT
SUM(TotalMinutesActual) AS SumTotalMinutesActual,
SUM(TotalMinutesTarget) AS SumTotalMinutesTarget,
FROM Q100TotalMinutesActualAgainstActualTargetDetail
```

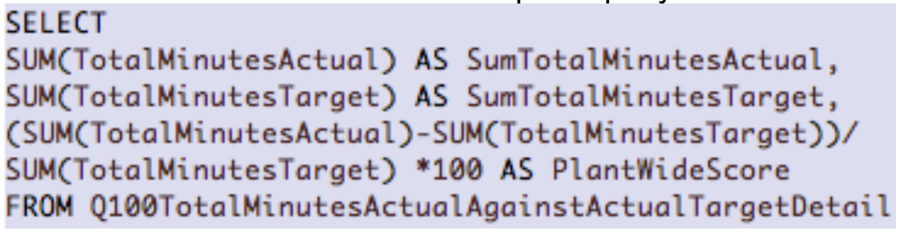
Note that this query is an aggregation query. View the resulting dataset.

4. Observe that it took 1.6 million minutes to perform all the work over the 20-week history, which is greater than our target of 1.1 million minutes.
5. Edit this query. Add the following calculated field to express this problem as a score:



```
(SUM(TotalMinutesActual)-SUM(TotalMinutesTarget))/
SUM(TotalMinutesTarget) *100 AS PlantWideScore
```

6. The screen shot below contains the complete query:



```
SELECT
SUM(TotalMinutesActual) AS SumTotalMinutesActual,
SUM(TotalMinutesTarget) AS SumTotalMinutesTarget,
(SUM(TotalMinutesActual)-SUM(TotalMinutesTarget))/
SUM(TotalMinutesTarget) *100 AS PlantWideScore
FROM Q100TotalMinutesActualAgainstActualTargetDetail
```

View the resulting dataset. The score should be calculated as 40.341743. An interpretation of this score is that we could reduce the total time required to process the demand by 40% of the ideal time.

7. Next, we want to compute this score for each of the three main work center areas (Blank and Shear, Pressing, and Welding and Assembly). We want to use summation, but we want to group by work center type.
8. Create a new query and title it “Q120ThruputPerformanceByWCType”. Select the fields “WCType”, “TotalMinutesActual” and “TotalMinutesTarget” from our query “Q100TotalMinutesActualAgainstTargetDetail”. This will be an aggregation query so sum TotalMinutesActual and TotalMinutesTarget as in our previous query.
9. Select one more field:
   ```
   (SUM(TotalMinutesActual)-SUM(TotalMinutesTarget))/
   SUM(TotalMinutesTarget)*100 AS WCTypeScore
   ```
10. After the **FROM** statement, add a **GROUP BY** clause in order to group the results by “WCType”.

    View the resulting dataset. Make sure there is a dedicated score for each WCType. Record each WCTypeScore:

    ____
    ____
    ____

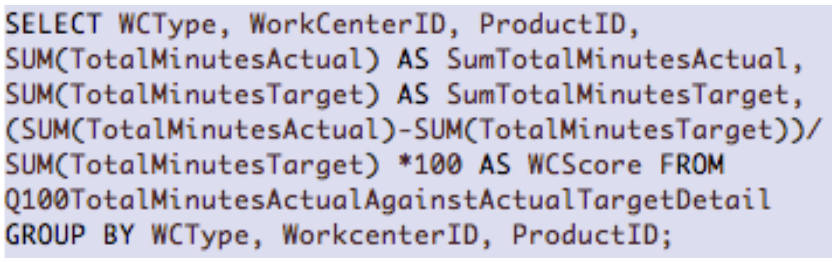
    Which area of the plant has the greatest opportunity for reducing the total time it uses? ____

Exercise 2:

1. Now, create a query that computes a score by work center. You will need a **GROUP BY** clause that groups by both WCType and WorkCenterID (in that order) to compute a “WCScore”. Save this new query as “Q130ThruputPerformanceByWC” and view the resulting dataset.
2. Record each of the scores:

   ____
   ____
   ____

   On which work center should we focus our improvement efforts?
   ____
3. Choose a product within this work center on which we should focus our efforts. Create a query that records a score by work center and product. You will need a **GROUP BY** clause that groups by WCType, WorkCenterID, ProductID to compute a “WCProductScore” and save this query as “Q140ThruputPerformanceByWCProduct”.
4. Here is the complete code for “Q140ThruputPerformanceByWCProduct”:



```
SELECT WCType, WorkCenterID, ProductID,
SUM(TotalMinutesActual) AS SumTotalMinutesActual,
SUM(TotalMinutesTarget) AS SumTotalMinutesTarget,
(SUM(TotalMinutesActual)-SUM(TotalMinutesTarget))/
SUM(TotalMinutesTarget) *100 AS WCScore FROM
Q100TotalMinutesActualAgainstActualTargetDetail
GROUP BY WCType, WorkcenterID, ProductID;
```

5. Within this work center, on which product should we focus our improvement efforts? ____
6. Run similar queries as these previous two, but first grouping by product alone, identifying the product with the largest score (call it product P), then grouping by product and work center together and looking for the work center & product combination that has the largest score among those for product P. Do you get the same results for the product/work center on which we need to focus our efforts?

   ____

   ____
7. It typically makes sense to look first for a "bottleneck" machine rather than an "inefficient" product. This is because a machine that is a bottleneck (either because it is processing an inefficient product or for other reasons) can slow things down on other machines that depend on what it makes. By fixing bottlenecks, we can often make the most improvement in a manufacturing process.

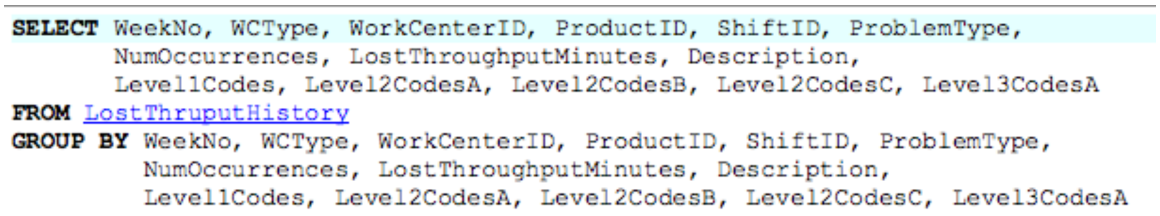
Selective Aggregation

We have made the following conclusion as a result of our analysis of the "ProductionHistory" table: the presses are the bottleneck work center area. Now that we know which product has the greatest opportunity for throughput improvement, we can now study the history of problems recorded in the factory and further focus our analysis.

The table "LostThruputHistory" is a partially aggregated log file. A system was in place to record the problems that occur in the factory and log certain details. A set of codes is associated with each problem event to classify the event, estimate the lost throughput time associated with the problem, and record information on which work center, product, and/or work shift was involved. Estimating lost throughput time involved working with more basic data such as machine speeds and scrap loss. The table includes the number of occurrences of the problem as well as the total estimated lost throughput time in the week.

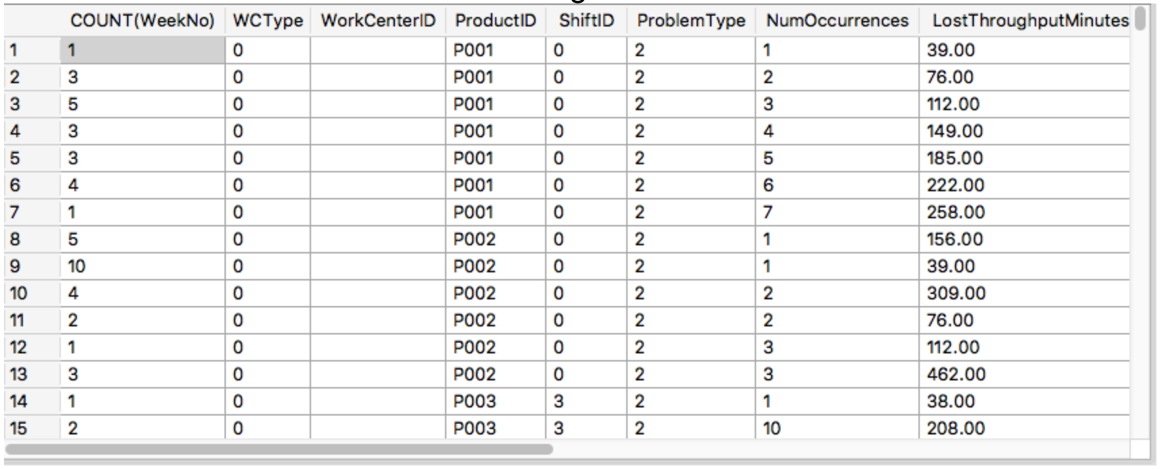
Exercise 1:

1. Create a new view and title it "Q150ProblemSummary". This will be a minimum aggregation query, and we will **GROUP BY** each field. The screen shot below contains the text of this query.



```
SELECT WeekNo, WCType, WorkCenterID, ProductID, ShiftID, ProblemType,
       NumOccurrences, LostThroughputMinutes, Description,
       Level1Codes, Level2CodesA, Level2CodesB, Level2CodesC, Level3CodesA
FROM LostThruputHistory
GROUP BY WeekNo, WCType, WorkCenterID, ProductID, ShiftID, ProblemType,
         NumOccurrences, LostThroughputMinutes, Description,
         Level1Codes, Level2CodesA, Level2CodesB, Level2CodesC, Level3CodesA
```

2. View the resulting dataset and verify that there are 3,188 rows. Note that we grouped by every single field; the only way we would have fewer than 3,188 is if some rows were exact duplicates of others.
3. Edit the view to add the COUNT function around the field WeekNo in the SELECT statement, and remove "WeekNo" from the **GROUP BY** clause. View the resulting dataset and verify that there are 1035 rows. Below is a screen shot of some rows of the resulting dataset:



	COUNT(WeekNo)	WCType	WorkCenterID	ProductID	ShiftID	ProblemType	NumOccurrences	LostThroughputMinutes
1	1	0		P001	0	2	1	39.00
2	3	0		P001	0	2	2	76.00
3	5	0		P001	0	2	3	112.00
4	3	0		P001	0	2	4	149.00
5	3	0		P001	0	2	5	185.00
6	4	0		P001	0	2	6	222.00
7	1	0		P001	0	2	7	258.00
8	5	0		P002	0	2	1	156.00
9	10	0		P002	0	2	1	39.00
10	4	0		P002	0	2	2	309.00
11	2	0		P002	0	2	2	76.00
12	1	0		P002	0	2	3	112.00
13	3	0		P002	0	2	3	462.00
14	1	0		P003	3	2	1	38.00
15	2	0		P003	3	2	10	208.00

4. Each of these rows refers to the same problem since they have the same level codes, product, and shift. We have several groupings for this problem since we are grouping by both "NumOccurrences" and "LostThroughputMinutes". This is not a useful view, so we should edit this query
5. Edit the view to add a SUM function around the fields "NumOccurrences" and "LostThroughputMinutes" in the **SELECT** clause and remove them from the **GROUP BY** clause. Rename these summed fields to SumNumOccurrences and SumLostThroughputMinutes respectively. View the resulting dataset. How many rows are there? ________________
6. Once again, edit the query. Remove "Description" from the **GROUP BY**, since people may have different descriptions for the same problem. It is not usually safe to group by a text field. Remove "Description" from the **SELECT** clause and replace it with:

```
MIN(Description) AS Description,
```

Usually we think of MIN as being applied to numbers or dates, but it can also be applied to text fields. In this case it selects the text that would appear first in an ORDER BY ASC. We just want to report one of the

Description fields from the records being aggregated, and ignore the others. To accomplish this we could have also used MAX.

View the resulting dataset and verify that the change we made does not change the number of rows in the aggregation, even though it could have. This implies that each set of problem codes had a unique description.

7. Add another field to be calculated from the data in "LostThroughputHistory" matching the following code:
```
SUM(LostThroughputMinutes)/SUM(NumOccurrences)
AS AvgLostThroughputMinutesPerOccurrence
```
Save the query and open the resulting dataset.

Exercise 2:

1. Create a new query titled "Q160ProblemSummaryForL1P011" and select all fields from our previous query "Q150ProblemSummary".
2. Add the following **WHERE** criterion:
```
WHERE WorkCenterID='L1' OR (WCType=1 AND ProductID='P011')
```
Note that this means that we are only interested in problems with this specific work center (L1) **OR** with this specific product (P011) in the pressing work center area.
3. Add the following **ORDER BY** clause to the end of your query:
```
ORDER BY SumLostThroughputMinutes DESC
```
View the resulting dataset and look at the first row.
4. Comment on the most pressing throughput problem in the factory based on its classification codes, description, average problem duration, and other factors. You can look up what shift the ShiftID corresponds to in the ShiftNames table, the problem type that the ProblemType corresponds to in the ProblemType table, and what the level codes mean in the ClockCodes table. To understand these codes, search for a record with the corresponding code, and the level indicated by the name of the field. For example, a Level2CodesA value of S4 corresponds to the record in the ClockTypes table with Level=2 and Code='S4'.

The LIMIT Clause

1. Create another query and title it "Q170TopFiveProblemsForL01P011" and select all fields from our previous query "Q160ProblemSummaryForL01P011".
2. At the end of this query, use the phrase "**Limit 5**" in order to select the five first records only. The query reads as follows:
```
SELECT * FROM Q160ProblemSummaryForL1P011 LIMIT 5
```

3. View the resulting dataset and note the top problems associated with press L1 **OR** product P011 in the pressing work center area. Record the number of rows in the resulting dataset: ___________________________

Review

We have used aggregation queries to analyze the data we are given. We identified the area, work center, and product which offers the greatest opportunity for throughput improvement as well as the top five problems which should be addressed immediately to improve throughput.

We have used the aggregate operators **SUM()**, **COUNT()**, and **MIN()** with calculations to capture quantities of interest. These tools support useful data analyses.