# COMP 3380 Project Overview

This project is an opportunity for you to design and implement your own database based on publicly available data. You will model the database, apply techniques, and use SQL to formulate and run queries based on concepts learned in class.

## Project Teams

The project is to be completed in **teams of two or three people**. You are free to select your own teams by assigning yourself to a group in UMLearn. Anyone who is not in a team by September 30th will be assigned a team by the course instructor.

A note about teams: some teams will have communication problems, or work distribution problems. ALL team members are responsible for effective team functioning. In extreme cases only contact your instructor to mediate a group meeting. Waiting just before a deadline to solve the problem is too late.

## Project Deliverables

The project is divided into three parts. Parts of the project build on each other. When starting the project, read through every part’s description to have an idea of what’s coming. For example, in Part 3, you will have to explain decisions you made for Part 1, taking good notes while doing Part 1 will help with Part 3. Each part will be submitted to UMLearn.

* Part 1 (30%): Finding data and designing a database. Due October 16th at 4:30pm.
* Part 2 (55%): Creating and populating a database, and implementing an interface. Due November 20th at 4:30pm.
* Part 3 (15%): Writing a report. Due November 27th at 4:30pm.

## Project Marking

Please see the marking rubrics in the individual part descriptions to help prioritize your work. In general, we will be looking for ambitious and creative projects that are well executed. **Details listed in the descriptions are but minimum requirements, an excellent project deserving of an A+ will seek to surpass what is described in this document**.

All rubrics use a non-linear scale as follows to represent the quality of the work submitted. Each item will be graded /5. Furthermore, some items might carry more weight and are denoted by multipliers.

* 5 – top of the class, amazing
* 4 – great work, all required components covered and done well
* 3 – OK work, some components missing, some things done well
* 2 – poor work, some components missing, few components done well
* 1 – very poor work, many components missing, components not done well
* 0 – not submitted or does not demonstrate understanding the deliverable

# Part 1: Designing a Database

## A) Data Discovery

The first part of the project involves finding some data to analyze. One viable source is the Winnipeg Open Data portal (<https://data.winnipeg.ca>), though you are free to use other sources (e.g., wikis). Data from any public source is acceptable but remember to acknowledge your source(s)! (Be aware of any copyright or licensing issues if you want to use this data beyond this course.)

You can aggregate data from multiple sources, but all data must ultimately be connected. In other words, the ER diagram you draw with this data for Part 1.B must be a connected graph (even better, try to have a tightly connected graph, i.e., the graph remains connected if you start removing tables). Aim to find and/or create a dataset that will ultimately break down to more than 10 tables and 1000 rows with relatively few support tables compared to main tables after completing Part 1.B. A support table is a table that is usually small in both arity and cardinality, and is mostly used for lookup purposes (e.g., a table that just has Rank and Salary, where you can lookup a Salary based on Rank).

## B) Database Design

You will draw an ER model (including EER components, if appropriate) which represents the database you have chosen to create. The model must include participation and cardinality constraints, as well as a brief justification for each. Justifications should explain the “why” of constraints, not merely putting them into words (e.g., “not every Song is written by an Artist” = bad, “some Songs are written by unknown Artists, and so aren’t in the Wrote table” = better).

You will then convert your ER model to a relational model and normalize it as much as possible using the rules and standards discussed in class and in the lectures.

## Part 1: Deliverable

A PDF containing a brief summary of your dataset (e.g., a short description of what it is, how much data), your ER model, and the final relational model (post-merge and post-normalization). Include your justifications for participation/cardinality constraints and the steps you took for translating your ER model to your final relational model (including steps for merging and normalizing).

## Part 1: Marking

* Quality of dataset (e.g., size, connectedness)
* Ratio of support tables to main tables
* ER/EER diagram
* Justification of participation and cardinality constraints
* Translating and merging ER/EER diagram to relational model
* Normalization

# Part 2: The Database and its Interface

## A) Database Creation and Population

You will implement the database you designed in Part 1. If you were given feedback from Part 1, it should be incorporated with You will be given a Microsoft SQL Server schema you must create your database on.

Once your database is created, you will populate it with your data. You are must to use a code-based method to add records, which is handed in.

## B) Implementing an Interface

You will create a front-end interface which allows a person (say, an analyst) to access and use your database. It can be as simple (e.g., command line interface) or as feature-rich (e.g., complete GUI) as you want, and can be implemented using any programming language you want (e.g., Java, Python, as a website). However, you must consider the following requirements:

* When using the interface, the database should be relatively secure according to what was discussed in class (e.g., can’t allow freely entering SQL commands to prevent SQL injection).
* Your interface should support an analyst trying to answer interesting questions they might have of the data, which are not easily answerable. You might consider taking some time to come up with interesting questions an analyst might have and allow your interface to execute the relevant queries. You should support at least one query which includes GROUP BY, one that includes ORDER BY, and one that includes an aggregate function. Note that your interface does not necessarily have to make these components explicit, at a minimum, it should simply allow someone to run those queries. Some tips:
  + GROUP BY and aggregate functions are hard for humans to do on the fly, so are an easy way to create interesting queries.
  + Queries should be relevant and potentially useful to an analyst. Complex queries that are hundreds of words long, nested four layers deep could be interesting for you to implement, but might be too convoluted for an analyst to ever reasonably run.
  + Don’t worry too much about optimizing your queries, as long as they run in a reasonable amount of time. However, consider informing the user with a message of some kind if a query is currently being executed (so that they don’t think your interface has crashed).
* Content from all tables should be accessible one way or another.

## Part 2: Deliverable

Everything required to create database tables, relationships, populate the tables, and finally the program you wrote for interacting with your database. Include a readme.md file with instructions on how to create and populate the database, and run your program.

## Part 2: Marking

* Database created and populated
* Queries – correctness
* Queries – complexity and interestingness
* Interface – functionality
* Interface – level of abstraction and ease-of-use

# Part 3: Project Report

For this last part, you will write a short report (3 - 5 pages) detailing your progress through the project. If you took good notes when working through Parts 1 and 2, this should be relatively quick and easy to put together. The report should at least include the following information, but you should consider adding anything else that you find interesting or that might help someone reading your report understand what you did.

* A summary of the data: Why was it chosen? What does if consist of? How large is it (file size, number of records)? Don’t forget to acknowledge the sources of your data! Include an ER diagram.
* A discussion of the data model
  + Why was it broken down into those tables?
  + Did you face any difficult choices when designing the model (e.g., tricky participation/cardinality ratio decisions)?
  + Did the data model cleanly fit into the relational database?
  + Do you regret any decisions you made in your model? Did you change your model from part 1 when implementing it in part 2? What changes, and why?
* Could the data be modelled in a different way, why or why not? Given the work completed, would you choose this model?
* A list of interesting queries you can run using the interface: Explain what the queries return, you don’t have to include the SQL code.
* Does this dataset require a relational database? Would other database system be a better choice in modelling this data? Why or why not? Would the “interesting queries” you wrote be easier or harder to re-create if you were using an alternative database?
* Would this database be a good teaching tool for COMP 3380? Are there good problems for future students to solve in this database? (Optional)

## Part 3: Deliverable

A PDF of your report.

## Part 3: Marking

* Summary of data
* Discussion of data model
* Summary of the database
* Summary of the interface
* List of interesting queries
* Other interesting discussions or summaries
* Writing quality