# COSC6340: Project List

## 1 Introduction

The goal is to work on an important theory topic of database systems or data mining, including AI applications. Requirements are as follows:

- Main requirement: working prototype with correct results with simple to complex inputs.
- Secondary requirement: efficient solution with good $\Theta(g(n))$ with C++, or quasi linear speedup if parallel, or low I/O cost if based on queries.
- Project to be developed in 2 phases: Phase 1 working prototype with simple inputs; Phase 2 optimized version.
- Team project: 2 students.

## 2 Project List

1. Tensor storage and querying for tensors rank-3 up to rank-5, for AQI prediction. Input: AQI measurements by hour/day. Train a simple RNN with up to 2 intermediate layers storing input and intermediate layers as tuples with SQL. Language: SQL+Py.
2. Join reordering using smart overlapping recursion (a.k.a. dynamic programming). Input: synthetic queries with 10+ joins and table cardinalities. Language: C++.
3. Stochastic gradient descent on sparse matrices programmed with recursive queries and C++. Language: C++, SQL.
4. Updating a deep neural network for binary classification with 3 intermediate layers with concurrent insertions with multiple threads. Input: a data set with d real features. Goal: maintainaing data set with transactions and propagating the changes to the NN model. Explore deferred model computation vs immediate updates on the NN. Language: C++.
5. Querying paths of a precomputed transitive closure of a graph in SQL. Input: a binary undirected graph OR a directed graph with distances. Use a parallel columnar DBMS. Language: SQL.
6. Building a traffic prediction graph data set with SQL, to be consumed by a GNN. Input: traffic data by hour/day with GPS locations, road information. Vertices=intersections, Edges=roads, Edge weight=traffic measurement by hour from sensors. Vertices labeled with road names. Language: Python.
7. JSON: store and query a history of divrse neural network training runs, including MLPs, CNNs, RNNs, LSTM, transformers. Input: main functions, pytorch parameters, data set metadata, related library functions, user/machine. Goal: comparing different experiments, isolating critical parameter values, merging models, finding out who is doing what. System: MongoDB recommended, but Py libraries feasible. Language: Python and MongoDB API.