Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Lab 5

Database Systems Oct. 3, 2018

Background: AirSupplyData-2018.xlsx contains transactional data from supply orders placed with multiple vendors. While items may share product names, they do not share item numbers or descriptions. A/P refers to the accounts / payable, or the number of days before a payment is due. The default arrival date should be set to NULL for new orders. Cost per order can be calculated by multiplying item cost with quantity. Product price is both transactional (order-level data) and static data (product-level data).

**Part I. Get Set (15pts)**

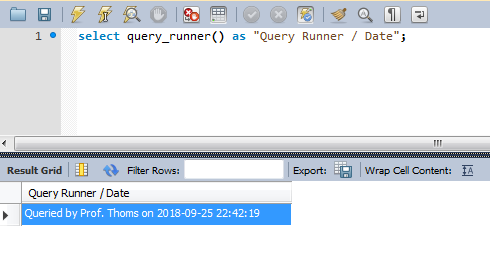
1. Create a normalized database to 3NF for the transactional data found in AirSupplyData-2018.xlsx. Reverse engineer your database in MySQL Workbench and paste the result below.

**Part II. Go (5pts)**

1. Construct *n* bulk insert statements (1 per entity) to insert the data found in AirSupplyData-2018.xlsx.
   1. Note: Data cleansing may be required. Do not include data for derived attributes.

**Part III. Go (20pts)**

For all views and stored procedures, make a call to query\_runner() to show your name and date as shown in the screenshot below.



1. Create the view `all\_orders` that recreates the output found in AirSupplyData-2018.xlsx.
2. Create the view `vendor\_product\_count` that returns a listing of vendors and the number of products provided by each vendor. Sort your results high to low.
3. Create the view `vendor\_purchases` that returns a listing of vendors and the total amount of purchases made through that vendor. Sort your results from largest to smallest overall dollar amount.
4. Create the view `payments` that outputs the future dates for when payments are due. Use the accounts / payable terms to determine the last date for payment. Capture all relevant information from order, item and vendor tables.
5. Create the view `vendor\_shipping\_stats` that produces a listing of the average ship times for each vendor. Sort the results by longest to shortest duration.
6. Create the stored procedure `find\_vendor\_orders` that accepts vendor data as input and produces a listing of orders placed with that vendor.
7. Create the stored procedure `show\_vendor\_purchases` that accepts a vendor number as input and returns the total cost of goods purchased through that vendor. Sort by the total cost highest to lowest and capture all relevant information from order, item and vendor tables.
8. Create the stored procedure `find\_products` that accepts a vendor number as input and outputs all relevant product information for products manufactured by that vendor, including item number, description and cost. Sort by vendor lowest to highest.
9. Create the stored procedure `show\_product\_pricing` that accepts an item description as input and returns a listing of vendors and their prices for that specific product. Sort results by item cost low to high.
10. Create the stored procedure `show\_orders\_by\_date` that accepts a start date and end date as parameters and returns all order, product and vendor information for orders placed on or between those dates sorted by order date. Capture all relevant information from order, item and vendor tables.