Database Class – Prof. Brodsky
Project Assignment:

The project has two required parts - Part 1 and Part 2 - and an optional Part 3.

Part1: Database design
Consider the following Supply Chain information system description. The system should support a collaborative supply chain composed of suppliers, manufacturers, shippers and end-customers.
Items of different kinds are being moved in the the supply chain. Manufacturers use Items of materials to manufacture Items of products for customers. Suppliers supply Items of materials to manufacturers; they also supply Items directly to Customers. Shippers (e.g., UPS, Fedex etc) move items from one business entity (supplier, manufacturer, customer etc) to another.

Items have a unique id and weight. Every business entity (suppliers, manufacturers, customers etc.) is identified by its id, and has a shipping location (to be used by Shippers for shipping orders),
address, phone, web location, and contact information.

Every product item (e.g., a table) has a number of associated material/part items in certain quantities necessary to produce 1 unit of the product item. For example,
a table product item, requires 1 table top item, 4 leg items and 8 screw items. Suppliers supply Items, using price per unit, which may vary among different Suppliers for the same Item.
Suppliers have volume discount applied on the dollar amount computed based on price per unit.
Volume discount is described by a percentage of deduction for

amount above a predetermined bound.
Manufacturers produce product Items; this production has an associated setUpCost and product cost per unit.
Manufacturers may offer volume discounts to customers applied the same way suppliers apply volume discounts.
Shippers price shipping services per pairs of (source, destination) pairs, where sources and destinations are shipping locations of business entities.
The pricing of each shipper is based on the total weight of shipment from source to destination, using price per lb, and a volume discount applied on the total dollar amount..
Customers have demand quantity for certain Items.
The orders are recorded separately for shipping, manufacturing and supply.
Shipping orders capture information about a shipper, sender, and receipient (who are business entities) and the Item being shipped, and record the quantity of the Item shipped.
Manufacturing orders capture information about a manufacturer, a manufactured Item and the ordered quantity; and
Supply orders capture information about a supplier, Item and the quantity supplied.

1. Create an ER diagram, and specify all integrity constraints. If some information is missing, suggest additional assumptions and briefly explain their rationale.
2. Translate the ER design into CREATE TABLE definitions using SQL, and add CHECK constraints as necessary

Important note: In Part 1, DO NOT try to reverse-engineer tables in Part2 - this won't give the correct answers!

Part 2: Database implementation (NOTE: you need to submit the script file **"queries.sql"** ONLY for Project Part 2 = HA3. See (B) below.

A. Consider the following tables (see file **create_empty_tables.sql** and **sample_db.sql**). For foreign key constraints definition, note that suppliers, manufacturers, shippers, customers, senders and recipients are business entities. Note also that product items, and material items are items.

1. **items(item,unitWeight)**: item has unitWeight. Assume that all items, including supplied and manufactured must appear in this table.
2. **busEntities(entity, shipLoc, address, phone, web, contact)**: every business entity has shipLoc (to be used by shipping companies), address, phone, web link, and contact info. Assume that all business entities, including suppliers, manufacturers and customers must appear in this table.
3. **billOfMaterials(prodItem, matItem, QtyMatPerItem):** to produce 1 unit of prodItem, manufacturers need QtyPerItem units of matItems (e.g., if a table (prodItem) needs 4 Legs (matItems), QtyPerItem = 4)
4. **supplierDiscounts(supplier, amt1, disc1, amt2, disc2):** supplier gives discount disc1 for purchase $ amount between amt1 and amt2, and disc2 for purchase amount above amt2. Note that discounts will be expressed as fractions rather then percentage, e.g., 0.15 rather than 15 (%).
5. **supplyUnitPricing(supplier, item, ppu):** item supplied by sup has ppu (price per unit)

6. **manufDiscounts(manuf, amt1, disc1):** manufacturer manuf gives discount disc1 for manufacturing cost in excess of amt1 of the base cost, which is computed according to manufUnitPricing table - see below.
7. **manufUnitPricing(manuf, prodItem, setUpCost, prodCostPerUnit):** For manufacturing of prodItem by manuf, the manufacturer base cost is computed as setUpCost plus the prodPricePerUnit times the qty of the produced prodItem
8. **shippingPricing(shipper, fromLoc, toLoc, minPackagePrice, pricePerLb, amt1, disc1, amt2, disc2):** The shipping cost for a shipper from fromLoc to toLoc is computed as follows:
    a. determine the total weight of all items shipped from fromLoc to toLoc (by all senders at fromLoc to all recipients at toLoc)
    b. base cost: is computed based on total weight of shipment and pricePerLb
    c. discounted cost: then for amount between amt1 and amt 2, disc 1 is applied; and to the amount above amt2, disc2 is applied
    d. total cost: the maximum of minPackagePrice and the discounted price
9. **customerDemand(customer, item, qty):** The demand by customer is qty units of item; note that items may come from any combination of manufacturers and/or suppliers.
10. **supplyOrders(item, supplier, qty):** qty units of item were ordered from supplier

11. **manufOrders(<u>item, manuf,</u> qty):** qty units of item were ordered to be produced by manuf

12. **shipOrders(<u>item, shipper, sender, recipient</u>, qty):**

- qty units of item were requested to be shipped by shipper from sender to recipient. Note senders and recipients are business entities such as suppliers, manufacturers and customers.

B. Implement the following SQL views by populating the template "queries.sql". This SQL file must have the view definitions only and nothing else. For Part 2 of the project (=HA3) submit **"queries.sql"** ONLY to Blackboard. Make sure that "queries.sql" compiles/runs w/out errors.

1. **shippedVsCustDemand:** For every (customer, item) pair in customerDemand, compute the total qty of this item shipped to this customer, along with the demand qty. Note that the items may come from manufacturers and/or suppliers. The resulting schema should be (customer, item, suppliedQty, demandQty). Order the result by customer, item.
2. **totalManufItems:** For every (product) item in manufOrders, compute the total qty of this item ordered from all manufacturers. The resulting schema should be (item, totalManufQty). Order the result by item.
3. **matsUsedVsShipped:** For every manuf in manufOrders, and matItem used by this manuf (i.e., manuf is ordered a prodItem that requires a matItem according to billOfMaterials) compute:
    - the total qty of this matItem requiered to produce all (product) items ordered from this manuf,
    - the total qty of this matItem shipped by all shippers to this manufacturer

    The resulting schema should be (manuf, matItem, requiredQty, shippedQty). Order the result by manuf, matItem.
4. **producedVsShipped:** For every (item, manuf) in manufOrders compute the total qty of this item shipped out from this manuf (by all shippers to any recipient), along with the total qty of this item ordered from this manufacturer (in manufOrders). The resulting schema should be (item, manuf, shippedOutQty, orderedQty). Order the result by item, manuf.

5. **suppliedVsShipped:** For every (item, supplier) in supplyOrders compute the total qty of this item shipped from this supplier (by all shippers to any recipient), along with the ordered qty of this item. The resulting schema should be (item,supplier, suppliedQty, shippedQty). Order the result by item, supplier.
6. **perSupplierCost:** For each supplier in supplierDiscounts, compute the total cost of items supplied by this supplier (according to supplyOrders).
   Clarification: to compute the perSupplierCost: (1) express the cost before discount (summation of ppu*qty over all items ordered from that supplier); (2) applying discount. For example, assume that CostBeforeDiscount for supplier 17 is $2500; and amt1=1000, disc1=0.1, amt2 = 2000, disc2 =0.2. The cost after discount will be (1000 + (2000- 1000)*(1-0.1) + (2500-2000)*(1-0.2) = 2300.
   The resulting schema should be (supplier, cost). Order the result by supplier.
7. **perManufCost:** For each manufacturer in manufDiscounts, compute the total manufacturing cost of all items produced by this manufacturer (according to manufOrders).
   The resulting schema should be (manuf, cost). Order the result by manuf.
8. **perShipperCost:** For each shipper in shippingPricing, compute the total shipping cost of this shipper. The resulting schema should be (shipper, cost). Order the result by shipper.
9. **totalCostBreakdown:** Compute the total supply cost, manufacturing cost, and shipping cost. The resulting schema should be (supplyCost, manufCost, shippingCost, totalCost).
10. **customersWithUnsatisfiedDemand:** Find customers, whose demand is NOT satisfied, i.e., are not shipped all the quantities of items. The resulting schema should be

(customer). Order the result by customer.

11. **suppliersWithUnsentOrders:** Find suppliers, whose orders are not fully shipped out. The resulting schema should be (supplier). Order the result by supplier.
12. **manufsWoutEnoughMats:** Find manufacturers who do NOT have enough materials to produce ordered product quantities, i.e., not enough materials were shipped to them. The resulting schema should be (manuf). Order the result by manuf.
13. **manufsWithUnsentOrders:** Find manufacturers whose orders are not fully shipped out. The resulting schema should be (manuf). Order the result by manuf.

C. How to test and debug your views (and know your score when you're done)?

1. Make sure that you have Python and MySQL installed and working. For details see **installation_instructions.md** (in the folder below.).
2. Upload the folder **cs_450_550_db_project_mysql_template_2022_template**
3. For **SQL**:
    a. In the folder **solution_sql**, duplicate the file **queries_template.sql** to create the file **queries.sql** Also, duplicate the file **credentials_template.py** to create the file **credentials.py** and fill it with your MySQL credentials (user and password).
    b. Fill in your queries in **queries.sql** by replacing the placeholder queries in the file. Make sure that the names of the attributes in the SELECT clause are exactly as they appear in the template.
    c. In terminal, make **solution_sql** your current folder
    d. In terminal, run **python produce_answers_sql.py** which will generate (or re-write) the file **answers.json**
    e. In terminal, run **python report.py** which will generate (or re-write) the file **report.json**
    f. You can redo Steps (d) and (e) to generate updated **answers.json** and **report.json** If **report.json** shows that some of your queries are incorrect, you can compare **answers.json** with **testDBs/correct_answers.json** to help debug your queries.
4. You must upload **queries.sql** as a solution to Project Part 2.

Part 3: Bonus Option (do not attempt before all non-bonus parts are implemented): Implement a web-based UI, that would allow users to update the DB and ask selection/projection queries from the defined views. You may want to use Oracle tools to do that.