CSE 462 : SQL

Name: _____ Date: February 16, 2011

***** Solved in class, on February 16, 2011. *****

Consider the **Products Schema** below, where keys are underlined. Refer to this schema in order to formulate the SQL queries described in the problems. Note: the *gross value* of an order is the sum of `qty` × `unitPrice` over all entries in the order.

`Customer(cId, name, city, phone)`

`Product(sku, pName, unitPrice)`

`Order(oId, cId, date)`

`OrderEntry(oId, sku, qty)`

1. Create a `GrossRevenues` view as follows. For every order, list the `oId`, the `date` of the order, the `name` of the customer who placed the order, the number of entries in the order (name it `TotalEntries`), the quantity of items sold in the order (name it `TotalItems`), and the gross value of the order (name it `GrossValue`).

```
CREATE VIEW GrossReveniews AS
  SELECT
    O.oId,
    O.date,
    C.name,
    COUNT(OE.sku) AS TotalEntries,
    SUM(OE.qty) AS TotalItems,
    SUM(OE.qty*P.unitPrice) AS GrossValue
  FROM
    Customer AS C
    NATURAL JOIN Order AS O
    NATURAL JOIN OrderEntry AS OE
    NATURAL JOIN Product AS P
  GROUP BY
    O.oId,
    O.date,
    C.name;
```

2. Using the `GrossRevenues` view, list the average gross value of the orders with gross value higher than the average gross value for all orders (name the result `HigherAverage`).

```
  SELECT
    AVG(GrossValue) AS HigherAverage
  FROM
    GrossRevenues
  WHERE
    GrossValue > (SELECT AVG(GrossValue) FROM GrossRevenues);
```

3. For every customer from Amherst, list the `name` and sum of the gross value of all orders that he/she placed (name it `TotalValue`). If the customer did not place any orders, a total value of 0 should be displayed. Order the results by decreasing total value.

```
  SELECT
    C.name,
    COALESCE(SUM(OE.qty*P.unitPrice), 0) AS TotalValue
  FROM
    Customer AS C
    NATURAL LEFT OUTER JOIN Order AS O     -- keeps customers with no orders
    NATURAL LEFT OUTER JOIN OrderEntry AS OE -- keeps dangling orders
    NATURAL LEFT OUTER JOIN Product AS P     -- keeps dangling order entries
  WHERE
    C.city = 'Amherst'
  GROUP BY
    C.name
  ORDER BY 2 DESC;  -- 2 refers to the second attribute
```

4. List the unique `name` of all customers who placed some order containing all products.

```
SELECT DISTINCT
  C.name
FROM
  Customer AS C
  NATURAL JOIN Order AS O
  NATURAL JOIN OrderEntry AS OE
GROUP BY
  C.cId, C.name, O.OrderNo
HAVING COUNT(sku) = (SELECT COUNT(*) FROM Product);
```