Scalable Data Engineering, Exercise 2

Henry Haustein

Task 1

(a) False, for some reporting functions we need `OVER`

(b) True

(c) False, this does `CUBE`

(d) True

(e) True

(f) False, SQL integer division returns an integer

(g) False, `ORDER BY` in an `OVER` clause has no effect on the sorting of the result. And an `ORDER BY` outside an `OVER` only sorts the result.

(h) False, MDX operates on data cubes. This can sometimes work on relational databases but it's not always the case.

Task 2

(a) SQL:

```sql
SELECT ORDERS.O_ORDERKEY, AVG(ORDERS.O_TOTALPRICE)
FROM ORDERS
GROUP BY ORDERS.O_ORDERKEY
```

(b) SQL:

```sql
SELECT REGION.R_NAME, NATION.N_NAME,  SUM(ORDERS.O_TOTALPRICE)
FROM REGION, ORDERS, CUSTOMER, NATION
WHERE
  ORDERS.O_CUSTKEY = CUSTOMER.C_CUSTKEY AND
  CUSTOMER.C_NATIONKEY = NATION.N_NATIONKEY AND
  NATION.N_REGIONKEY = REGION.R_REGIONKEY
GROUP BY ROLLUP(REGION.R_NAME, NATION.N_NAME)
```

(c) SQL:

```sql
SELECT
  year, quarter, sales, sales/SUM(sales)
FROM (
  SELECT
```

```
    EXTRACT(year FROM ORDERS.O_ORDERDATE) AS year,
    EXTRACT(quarter FROM ORDERS.O_ORDERDATE) AS quarter,
    COUNT(*) AS sales
  FROM ORDERS
  GROUP BY year, quarter
) AS x
```

(d) SQL:

```
SELECT
  PART.P_NAME,
  qty,
  RANK() OVER(ORDER BY qty DESC)
FROM (
  SELECT
    PART.P_NAME,
    SUM(LINEITEM) AS qty
  FROM PART, LINEITEM
  WHERE PART.P_NAME = LINEITEM.L_PARTKEY
  GROUP BY PART.P_NAME
) AS x
```

(e) SQL:

```
SELECT
  PART.P_NAME,
  NATION.N_NAME,
  SUM(LINEITEM.L_QUANTITY),
  RANK() OVER(
    PARTITION BY PART.P_NAME
    ORDER BY SUM(LINEITEM.L_QUANTITY) DESC
    )
FROM PART, NATION, LINEITEM, SUPPLIER
WHERE
  PART.P_PARTKEY = LINEITEM.L_PARTKEY AND
  LINEITEM.L_SUPPKEY = SUPPLIER.S_SUPPKEY AND
  SUPPLIER.S_NATIONKEY = NATION.N_NATIONKEY
GROUP BY PART.P_NAME, NATION.N_NAME
```