COMS W4111-002/V002 (Spring 2023) Introduction to Databases

Homework 4: All Tracks

Overview

- There are two parts to HW 4:
  - `4a` : Written questions
  - `4b:` A common set of practical tasks for both the programming and non-programming tracks.
- HW 4 **does not** have separate assignments for the programming and non-programming tracks.

Homework 4b has the following tasks:

1. Create a new schema `<uni>_S22_classic_models_star.` Replace `<uni>` with your UNI.

2. You will create a star schema using the data from your Classic Models database.
   - The fact in the fact table is of the form `(productCode, quantityPrders, priceEach, orderDate, customerNumber).`
   - The dimensions are:
     - `date_dimension: year, quarter, month, day of the month.`
     - `location_dimension: region, country, city.` The zip file contain a file `country_region.csv` that provides the mapping of countries to regions.
     - `product_dimension: product_scale, product_line, product_vendor.`

3. You will write queries that demonstrate:
   - A slice of the data.
   - A dice of the data.
   - A drill-down.
   - A roll-up.

- The homework is due on `2022-MAY-01` at 11:59 PM. We will post detailed submission instructions on Ed and Gradescope. Your submission format will be PDF and zip copies of this notebook. You must name your files following the instructions we publish.

Setup

In [1]:
```
import pandas as pd
```

In [2]:
```
%load_ext sql
```

In [3]:
```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

Out[3]:
```
'Connected: root@None'
```

In [4]:
```
country_region = pd.read_csv('./country_region.csv')
```

In [5]:
```
country_region
```

Out[5]:

	Country	Region
0	France	EMEA
1	USA	NaN
2	Australia	APAC
3	Norway	EMEA
4	Poland	EMEA
5	Germany	EMEA
6	Spain	EMEA
7	Sweden	EMEA
8	Denmark	EMEA
9	Singapore	APAC
10	Portugal	EMEA
11	Japan	APAC
12	Finland	EMEA
13	UK	EMEA
14	Ireland	EMEA
15	Canada	NaN
16	Hong Kong	APAC
17	Italy	EMEA
18	Switzerland	EMEA
19	Netherlands	EMEA
20	Belgium	EMEA
21	New Zealand	APAC
22	South Africa	EMEA
23	Austria	APAC
24	Philippines	APAC
25	Russia	EMEA
26	Israel	EMEA

In [6]:
```
import pandas as pd
```

In [7]:
```
from sqlalchemy import create_engine
```

In [8]:
```
engine = \
    create_engine("mysql+pymysql://root:dbuserdbuser@localhost")
```

Schema

- Execute your SQL statements for creating the schema, table and constraints for the fact and dimension tables in the following cells.

In [9]:
```
%sql drop schema if exists ad3706_S22_classic_models_star;
```

```
 * mysql+pymysql://root:***@localhost
5 rows affected.
```

Out[9]:
```
[]
```

In [10]:
```
%sql create schema if not exists ad3706_S22_classic_models_star;
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[10]:
```
[]
```

In [11]:
```
country_region.to_sql('country_region',
                      schema='ad3706_s22_classic_models_star',
                     con=engine,
                     index=False,
                     if_exists="replace")
```

In [12]:
```
%sql use ad3706_S22_classic_models_star;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[12]:
```
[]
```

In [13]:
```
%%sql

update ad3706_S22_classic_models_star.country_region set region='NA' where regi
```

```
 * mysql+pymysql://root:***@localhost
2 rows affected.
```

Out[13]:
```
[]
```

In [14]:
```
%%sql

update classicmodels.customers set country='Norway' where country='Norway';
update classicmodels.offices set country='Norway' where country='Norway';
update ad3706_S22_classic_models_star.country_region set country='Norway' where
```

```
 * mysql+pymysql://root:***@localhost
3 rows affected.
0 rows affected.
1 rows affected.
```

Out[14]:
```
[]
```

In [15]:
```
%sql drop table if exists date_dimension;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[15]:
```
[]
```

In [16]:
```
%%sql
```

```
#date dimension: year, quarter, month, day of the month

use ad3706_S22_classic_models_star;

create table if not exists date_dimension
(
    orderDate    date not null,
    year         int  null,
    quarter      int  null,
    month        int  null,
    day_of_month int  null,
    constraint date_dimension_pk
        primary key (orderDate)
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[16]:
```
[]
```

In [17]:
```
%sql drop table if exists location_dimension;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[17]:
```
[]
```

In [17]:
```
%%sql

#location dimension: region, country, city

use ad3706_S22_classic_models_star;

create table if not exists location_dimension
(
    customerNumber int         not null,
    region         text        null,
    country        varchar(50) null,
    city           varchar(50) null,
    constraint location_dimension_pk
        primary key (customerNumber)
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[17]:
```
[]
```

In [18]:
```
%sql drop table if exists product_dimension;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[18]:
```
[]
```

In [19]:
```
%%sql

#product dimension: product_scale, product_line, product_vendor.

use ad3706_S22_classic_models_star;
```

```
create table if not exists product_dimension
(
    productCode    varchar(15) not null,
    product_scale  varchar(10) null,
    product_line   varchar(50) null,
    product_vendor varchar(50) null,
    constraint product_dimension_pk
        primary key (productCode)
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[19]: []

In [27]:
```
%sql drop table if exists sales_facts;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[27]: []

In [28]:
```
%%sql

#sales_facts:(productCode, quantityOrdered, priceEach, orderDate, customerNumbe

use ad3706_S22_classic_models_star;

create table if not exists sales_facts
(
    sales_facts_id  varchar(255)   not null,
    productCode     varchar(15)    null,
    quantityOrdered int            null,
    priceEach       decimal(10, 2) null,
    orderDate       date           null,
    customerNumber  int            null,
    orderNumber     int            null,
    constraint sales_facts_pk
        primary key (sales_facts_id),
    constraint sales_facts_date_dimension_orderDate_fk
        foreign key (orderDate) references date_dimension (orderDate),
    constraint sales_facts_location_dimension_customerNumber_fk
        foreign key (customerNumber) references location_dimension (customerNum
    constraint sales_facts_product_dimension_productCode_fk
        foreign key (productCode) references product_dimension (productCode)
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[28]: []

Data Loading

- Enter and execute your SQL for loading the data into the facts and dimensions table.

The source of the information is the Classic Models data.

In [22]:
```
%%sql

#date_dimension: year, quarter, month, day_of_month

use ad3706_S22_classic_models_star;

insert into date_dimension (orderDate, year, quarter, month, day_of_month)
select
    distinct(orderDate) as orderDate,
    year(orderDate) as year,
    quarter(orderDate) as quarter,
    month(orderDate) as month,
    day(orderDate) as day_of_month
from classicmodels.orders;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
267 rows affected.
```

Out[22]:
```
[]
```

In [23]:
```
%%sql

#location dimension: region, country, city

use ad3706_S22_classic_models_star;

insert into location_dimension (customerNumber, region, country, city)
select
    customerNumber,
    region,
    country,
    city
from classicmodels.customers
join ad3706_S22_classic_models_star.country_region using (country);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
122 rows affected.
```

Out[23]:
```
[]
```

In [24]:
```
%%sql

#product dimension: product_scale, product_line, product_vendor.

use ad3706_S22_classic_models_star;

insert into product_dimension (productCode, product_scale, product_line, produc
select
    productCode,
    productScale,
    productLine,
    productVendor
from classicmodels.products;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
110 rows affected.
```

Out[24]: []

In [29]:
```
%%sql

#sales_facts: productCode, quantityPrders, priceEach, orderDate, customerNumber

set FOREIGN_KEY_CHECKS=0;

use ad3706_S22_classic_models_star;

insert into sales_facts (sales_facts_id, productCode, quantityOrdered,
                         priceEach, orderDate, customerNumber, orderNumber)
select
    concat(orderNumber, '-', orderLineNumber) as sales_facts_id,
    productCode,
    quantityOrdered,
    priceEach,
    orderDate,
    customerNumber,
    orderNumber
from classicmodels.orders
    join classicmodels.orderDetails using (orderNumber);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
2996 rows affected.
```

Out[29]: []

Queries

- In each of the sections below, define what your query is producing, provide the query and execute to produce the results.

In [30]:
```
%sql use ad3706_S22_classic_models_star;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[30]: []

Slice

Explanation: Slice is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with fewer dimensions. Here, the location_dimension is fixed, selecting only rows where the country is Germany. The date and product dimensions are not fixed. The information of all countries other than Germany are sliced out of the data cube.

In [31]:
```
%%sql

use ad3706_S22_classic_models_star;

with
    d1 as (
        select * from sales_facts left join location_dimension using (customerN
    d2 as (
        select * from d1 join product_dimension using (productCode)),
    d3 as (
        select * from d2 join date_dimension using (orderDate))
select * from d3
where country='Germany'
limit 10;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
10 rows affected.
```

Out[31]:

orderDate	productCode	customerNumber	sales_facts_id	quantityOrdered	priceEach	orderNu
2003-01-09	S18_2795	128	10101-1	26	167.06	
2003-01-09	S24_2022	128	10101-2	46	44.35	
2003-01-09	S24_1937	128	10101-3	45	32.53	
2003-01-09	S18_2325	128	10101-4	25	108.06	
2004-03-15	S12_3148	128	10230-1	43	128.42	
2004-03-15	S50_1514	128	10230-2	43	57.41	
2004-03-15	S18_4027	128	10230-3	42	142.18	
2004-03-15	S32_3207	128	10230-4	46	59.03	
2004-03-15	S24_4048	128	10230-5	45	99.36	
2004-03-15	S24_1444	128	10230-6	36	47.40	

Dice

Explanation: Dice is the act of producing a subcube by allowing the analyst to pick specific values of multiple dimensions. Here, the date and location dimensions are fixed, producing a sub-cude with information on entires where country is USA or Germany and month is Jan, Feb, or March. Rows which don't fulfill these requirements are cut out of the new cube. The product dimension is not fixed.

In [32]:
```
%%sql
```

```
use ad3706_S22_classic_models_star;

with
    d1 as (
        select * from sales_facts left join location_dimension using (customerN
    d2 as (
        select * from d1 join product_dimension using (productCode)),
    d3 as (
        select * from d2 join date_dimension using (orderDate))
select * from d3
where country in ('USA','Germany') and month in (1,2,3)
limit 10;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
10 rows affected.
```

Out[32]:

orderDate	productCode	customerNumber	sales_facts_id	quantityOrdered	priceEach	orderNu
2003-03-26	S32_3522	124	10113-1	23	58.82	
2003-03-26	S12_1666	124	10113-2	21	121.64	
2003-03-26	S18_4668	124	10113-3	50	43.27	
2003-03-26	S18_1097	124	10113-4	49	101.50	
2004-03-11	S12_4473	124	10229-1	36	95.99	
2004-03-11	S18_4600	124	10229-10	41	119.87	
2004-03-11	S700_2824	124	10229-11	50	91.04	
2004-03-11	S32_3522	124	10229-12	30	52.36	
2004-03-11	S12_1666	124	10229-13	25	110.70	
2004-03-11	S18_4668	124	10229-14	39	43.77	

Roll Up

Explanation: Roll up performs an aggregation on the cube by climbing up in the concept hierarchy and reducing the dimensions. Here, the query performs an aggregation on the cube, collecting information on the no_of_sales related to a specific products. The sub-cube produced by this roll up, compared to that of the drill down below, has less detail.

In [33]:

```
%%sql

use ad3706_S22_classic_models_star;

with
```

```
    d1 as (
        select * from sales_facts left join location_dimension using (customerN
    d2 as (
        select * from d1 join product_dimension using (productCode)),
    d3 as (
        select * from d2 join date_dimension using (orderDate))
select product_vendor, year, region, count(*) as no_of_sales
from d3
group by product_vendor, year, region
limit 10;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
10 rows affected.
```

Out[33]:

product_vendor	year	region	no_of_sales
Min Lin Diecast	2003	NA	29
Min Lin Diecast	2003	EMEA	33
Min Lin Diecast	2004	EMEA	47
Min Lin Diecast	2004	APAC	19
Min Lin Diecast	2004	NA	40
Min Lin Diecast	2005	EMEA	18
Min Lin Diecast	2005	NA	13
Min Lin Diecast	2005	APAC	7
Classic Metal Creations	2003	EMEA	39
Classic Metal Creations	2003	NA	35

Drilldown

Explanation: Here, the query drills down in two dimensions: [1] in the date dimension from year to year and month and [2] in the location dimension from region to region and country. From a cube point of view, the date dimension is formed by a combination of year and month (ex. 2003.January) and the location dimension is formed by a combination of region and country (ex. EMEA.Germany). The product dimension is not drilled down on. This drill down creates a cube with more detail than that which was created in the roll up above.

In [34]:

```
%%sql

use ad3706_S22_classic_models_star;

with
    d1 as (
        select * from sales_facts left join location_dimension using (customerN
    d2 as (
        select * from d1 join product_dimension using (productCode)),
    d3 as (
        select * from d2 join date_dimension using (orderDate))
select product_vendor, year, month, region, country, count(*) as no_of_sales
from d3
```

```
group by product_vendor, year, month, region, country
limit 10;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
10 rows affected.
```

Out[34]:

product_vendor	year	month	region	country	no_of_sales
Min Lin Diecast	2003	2	NA	USA	1
Min Lin Diecast	2003	5	EMEA	France	3
Min Lin Diecast	2003	7	EMEA	France	2
Min Lin Diecast	2003	8	NA	USA	5
Min Lin Diecast	2003	10	NA	USA	6
Min Lin Diecast	2003	11	EMEA	France	2
Min Lin Diecast	2003	11	EMEA	Norway	3
Min Lin Diecast	2003	12	NA	USA	4
Min Lin Diecast	2004	1	EMEA	France	3
Min Lin Diecast	2004	2	APAC	Australia	2

In [ ]: