# 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 (https://en.wikipedia.org/wiki/Star_schema) using the data from your Classic Models database.
   - The fact in the fact table is of the form `(productCode, quantityPrders, priceEach, orderedData, 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 [6]:

```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

Out[6]:

```
'Connected: root@None'
```

In [7]:

```
country_region = pd.read_csv('./country_region.csv')
```

In [8]:

```
country_region
```

Out[8]:

| | 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 [9]:

```
from sqlalchemy import create_engine
```

In [11]:

```
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 [14]:

```
%%sql

create schema sm4940_S22_classic_models_star
```

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

Out[14]:

```
[]
```

# Added the constraints after data loading

# 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.

# Note: I have considered the region for the countries which are not in the csv file as null and this adds 5 extra rows compared to filtering out these countries

In [15]:

```
country_region.to_sql('country_region', schema = "sm4940_S22_classic_models_star", con=engi
                      index = False,
                      if_exists = 'replace')
```

In [18]:

```
%%sql

use sm4940_S22_classic_models_star;

update country_region set Region = 'NA' where Region is NULL; # north america correction as

update classicmodels.customers set country = Replace(country," ","") # correcting some coun
```

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

Out[18]:

```
[]
```

In [19]:

```
%%sql

create table sales_facts as
select concat(orderNumber, "-", orderLineNumber) as sales_fact_id,
       productCode, quantityOrdered, priceEach, orderDate, customerNumber
from classicmodels.orders
join classicmodels.orderdetails using(orderNumber)
```

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

Out[19]:

```
[]
```

In [20]:

```
%%sql

create table date_dimension as
select distinct(orderDate), year(orderDate) as year, quarter(orderDate) as quarter,
       month(orderDate) as month, day(orderDate) as DayOfMonth
from sales_facts;
```

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

Out[20]:

```
[]
```

In [21]:

```
%%sql

create table location_dimension as
select a.customerNumber, a.country, a.city, b.region
from classicmodels.customers a
left join sm4940_S22_classic_models_star.country_region b using(country);
```

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

Out[21]:

```
[]
```

In [22]:

```
%%sql

create table product_dimension as
select productCode, productLine, productScale, productVendor
from classicmodels.products;
```

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

Out[22]:

```
[]
```

In [23]:

```
%%sql

alter table sales_facts add primary key(sales_fact_id);

alter table date_dimension add primary key(orderDate);

alter table sales_facts add constraint sales_facts_ibfk_1
    foreign key(orderDate) references date_dimension(orderDate);

alter table location_dimension add primary key(customerNumber);

alter table sales_facts add constraint sales_facts_ibfk_2
    foreign key(customerNumber) references location_dimension(customerNumber);

alter table product_dimension add primary key(productCode);

alter table sales_facts add constraint sales_facts_ibfk_3
    foreign key(productCode) references product_dimension(productCode);
```

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

Out[23]:

```
[]
```

# Queries

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

In [24]:

```
%%sql

#creating a flat table as view for queries

create view flat_table_view as
select sf.productCode, sf.quantityOrdered, sf.priceEach, sf.orderDate, sf.customerNumber,
       dd.year, dd.quarter, dd.month, dd.DayOfMonth,
       ld.country, ld.city, ld.region,
       pd.productScale, pd.productVendor, pd.productLine
from sales_facts sf
left join date_dimension dd on sf.orderDate = dd.orderDate
left join location_dimension ld on sf.customerNumber = ld.customerNumber
left join product_dimension pd on sf.productCode = pd.productCode
```

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

Out[24]:

```
[]
```

# Slice

Explanation: If our data is a cube then selecting one specific value of an attribute from one of the dimension tables will become a slice of that cude, In the query below, we are slicing the data set with Germany as the value of the country in the location_dimension to slice the cube

In [33]:

```
%%sql

select productVendor, month, country, count(*) as total_sales
from flat_table_view
group by productVendor, month, country
having country = 'Germany'
```

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

Out[33]:

| productVendor | month | country | total_sales |
|---|---|---|---|
| Welly Diecast Productions | 11 | Germany | 2 |
| Autoart Studio Design | 11 | Germany | 1 |
| Studio M Art Models | 11 | Germany | 2 |
| Gearbox Collectibles | 11 | Germany | 1 |
| Min Lin Diecast | 11 | Germany | 1 |
| Classic Metal Creations | 11 | Germany | 2 |
| Welly Diecast Productions | 3 | Germany | 1 |
| Classic Metal Creations | 3 | Germany | 1 |
| Min Lin Diecast | 3 | Germany | 1 |
| Gearbox Collectibles | 3 | Germany | 2 |
| Exoto Designs | 3 | Germany | 1 |
| Highway 66 Mini Classics | 3 | Germany | 2 |
| Studio M Art Models | 9 | Germany | 1 |
| Motor City Art Classics | 9 | Germany | 3 |
| Unimax Art Galleries | 9 | Germany | 1 |
| Exoto Designs | 9 | Germany | 1 |
| Min Lin Diecast | 9 | Germany | 2 |
| Carousel DieCast Legends | 9 | Germany | 1 |
| Second Gear Diecast | 9 | Germany | 2 |
| Red Start Diecast | 9 | Germany | 1 |
| Autoart Studio Design | 9 | Germany | 2 |
| Gearbox Collectibles | 10 | Germany | 5 |
| Welly Diecast Productions | 10 | Germany | 6 |
| Autoart Studio Design | 10 | Germany | 2 |
| Studio M Art Models | 10 | Germany | 3 |
| Min Lin Diecast | 10 | Germany | 3 |
| Classic Metal Creations | 10 | Germany | 3 |
| Highway 66 Mini Classics | 10 | Germany | 1 |
| Unimax Art Galleries | 10 | Germany | 1 |
| Second Gear Diecast | 10 | Germany | 1 |

| productVendor | month | country | total_sales |
|---|---|---|---|
| Motor City Art Classics | 11 | Germany | 1 |
| Exoto Designs | 11 | Germany | 1 |

# Dice

Explanation: Dice is a sub-cube of our cube of data where the data is filtered using a range of values from the dimension tables like in the query below where a sub-cube is selected with country values in Germany and USA and month values in Jan,Feb and Mar.

In [32]:

```
%%sql

select productVendor, month, country, count(*) as total_sales
from flat_table_view
group by productVendor, month, country
having country in ('Germany','USA') and
       month in (1,2,3)
```

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

Out[32]:

| productVendor | month | country | total_sales |
|---|---|---|---|
| Red Start Diecast | 1 | USA | 5 |
| Motor City Art Classics | 1 | USA | 10 |
| Welly Diecast Productions | 1 | USA | 3 |
| Exoto Designs | 1 | USA | 9 |
| Gearbox Collectibles | 1 | USA | 6 |
| Classic Metal Creations | 1 | USA | 7 |
| Autoart Studio Design | 1 | USA | 7 |
| Studio M Art Models | 1 | USA | 6 |
| Unimax Art Galleries | 2 | USA | 4 |
| Min Lin Diecast | 2 | USA | 6 |
| Autoart Studio Design | 2 | USA | 6 |
| Red Start Diecast | 2 | USA | 6 |
| Highway 66 Mini Classics | 2 | USA | 2 |
| Welly Diecast Productions | 2 | USA | 5 |
| Exoto Designs | 2 | USA | 5 |
| Red Start Diecast | 3 | USA | 7 |
| Gearbox Collectibles | 3 | USA | 4 |
| Min Lin Diecast | 3 | USA | 9 |
| Highway 66 Mini Classics | 3 | USA | 6 |
| Unimax Art Galleries | 3 | USA | 5 |
| Second Gear Diecast | 3 | USA | 7 |
| Exoto Designs | 3 | USA | 3 |
| Studio M Art Models | 3 | USA | 7 |
| Motor City Art Classics | 3 | USA | 3 |
| Welly Diecast Productions | 3 | USA | 3 |
| Unimax Art Galleries | 1 | USA | 3 |
| Min Lin Diecast | 1 | USA | 3 |
| Carousel DieCast Legends | 1 | USA | 6 |
| Second Gear Diecast | 1 | USA | 5 |

| productVendor | month | country | total_sales |
|---|---|---|---|
| Gearbox Collectibles | 2 | USA | 4 |
| Carousel DieCast Legends | 2 | USA | 9 |
| Classic Metal Creations | 2 | USA | 3 |
| Motor City Art Classics | 2 | USA | 6 |
| Second Gear Diecast | 2 | USA | 6 |
| Classic Metal Creations | 3 | USA | 5 |
| Carousel DieCast Legends | 3 | USA | 7 |
| Autoart Studio Design | 3 | USA | 5 |
| Welly Diecast Productions | 3 | Germany | 1 |
| Classic Metal Creations | 3 | Germany | 1 |
| Min Lin Diecast | 3 | Germany | 1 |
| Gearbox Collectibles | 3 | Germany | 2 |
| Exoto Designs | 3 | Germany | 1 |
| Highway 66 Mini Classics | 3 | Germany | 2 |
| Highway 66 Mini Classics | 1 | USA | 3 |
| Studio M Art Models | 2 | USA | 4 |

# Roll Up

Explanation: rollup is a relative transformation with respect to the previous selection like our flat table is the most drilldown version of our data, so in the query below when we are grouping the data by productVendor, year and region we are rolling up(aggregating) the data into slices which are specified by the attributes in the dimension tables.

In [27]:

```
%%sql

select productVendor, year, region, count(*) as total_sales
from flat_table_view
where region is not null
group by productVendor, year, region
```

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

# Drilldown

Explanation: Drilldown is also a relative transformation with respect to the previous state of the data, so in the previous query when we rolled up the data with year and when in this query when we are seleting by month, we are drilling down into year as month is a subset of year and similar analogy for country which is a subset of region.

In [28]:

```
%%sql

select productVendor, month, country, count(*) as total_sales
from flat_table_view
where region is not null
group by productVendor, month, country
```

```
 * mysql+pymysql://root:***@localhost
1044 rows affected.
```

In [ ]:

```

```