## Denormalize ABC\_Retail tables

See the code file **denormalize\_abc\_retail.sql**

**Check the result:**

Table **raw** is the table that we imported the original ABC\_Retail.csv 2 weeks ago.

Table **ABC\_Retail** is the table which we de-normalize all data from other tables and insert into it.

-- create table

CREATE TABLE abc\_retail.ABC\_Retail(

OrderID int,

OrderDate date default null,

Order\_ShippedDate date default null,

Order\_Freight numeric(20,2),

Order\_ShipCity varchar(255),

Order\_ShipCountry varchar(255),

Order\_UnitPrice numeric(20,2),

Order\_Quantity numeric(20,2),

Order\_Amount numeric(20,2),

ProductName varchar(255),

Employee\_LastName varchar(255),

Employee\_FirstName varchar(255),

Employee\_Title varchar(255),

CompanyName varchar(255),

Customer\_ContactName varchar(255),

Customer\_City varchar(255),

Customer\_Country varchar(255),

Customer\_Phone varchar(255)

);

-- denormalize tables into table ABC\_Retail

INSERT INTO abc\_retail.ABC\_Retail

(

    OrderID, OrderDate, Order\_ShippedDate, Order\_Freight, Order\_ShipCity,

    Order\_ShipCountry, Order\_UnitPrice, Order\_Quantity, Order\_Amount,

    ProductName, Employee\_LastName, Employee\_FirstName, Employee\_Title,

    CompanyName, Customer\_ContactName, Customer\_City, Customer\_Country,

    Customer\_Phone

)

(

    SELECT DISTINCT

        o.order\_id, o.order\_date, o.order\_shippeddate, o.order\_freight,

        o.order\_shipcity, o.order\_shipcountry,

        opp.order\_unitprice, opp.order\_quantity, opp.order\_amount,

        opp.product\_name,

        e.employee\_lastname, e.employee\_firstname, e.employee\_title,

        cocu.company\_name,

        cocu.customer\_contactname, cocu.customer\_city, cocu.customer\_country,

        cocu.customer\_phone

    FROM abc\_retail.orders AS o

    JOIN abc\_retail.employees as e

    ON o.employee\_id = e.employee\_id

    JOIN

    (

        SELECT DISTINCT

            cu.customer\_id,

            co.company\_name,

            cu.customer\_contactname, cu.customer\_city, cu.customer\_country,

            cu.customer\_phone

        FROM abc\_retail.companys AS co

        JOIN abc\_retail.customers AS cu

        ON co.company\_id = cu.company\_id

    ) AS cocu

    ON o.customer\_id = cocu.customer\_id

    JOIN

    (

        SELECT DISTINCT

            op.order\_id, op.order\_unitprice, op.order\_quantity, op.order\_amount,

            p.product\_name

        FROM abc\_retail.order\_products AS op

        JOIN abc\_retail.products AS p

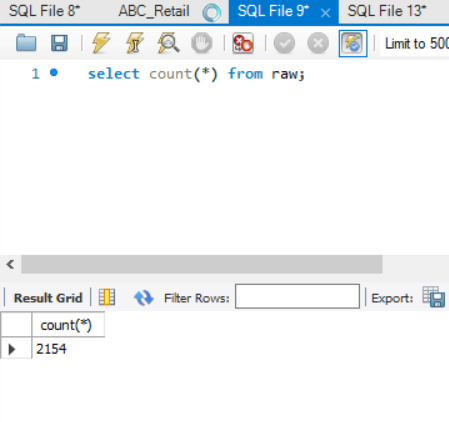
        ON op.product\_id = p.product\_id

    ) AS opp

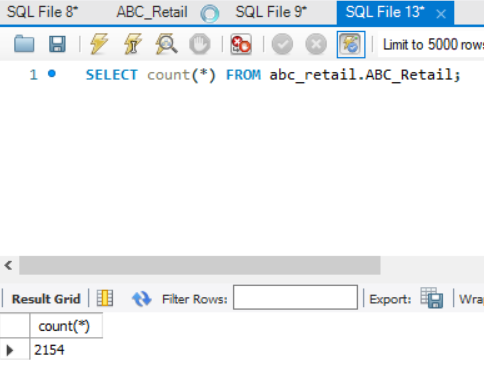
    ON o.order\_id = opp.order\_id

);

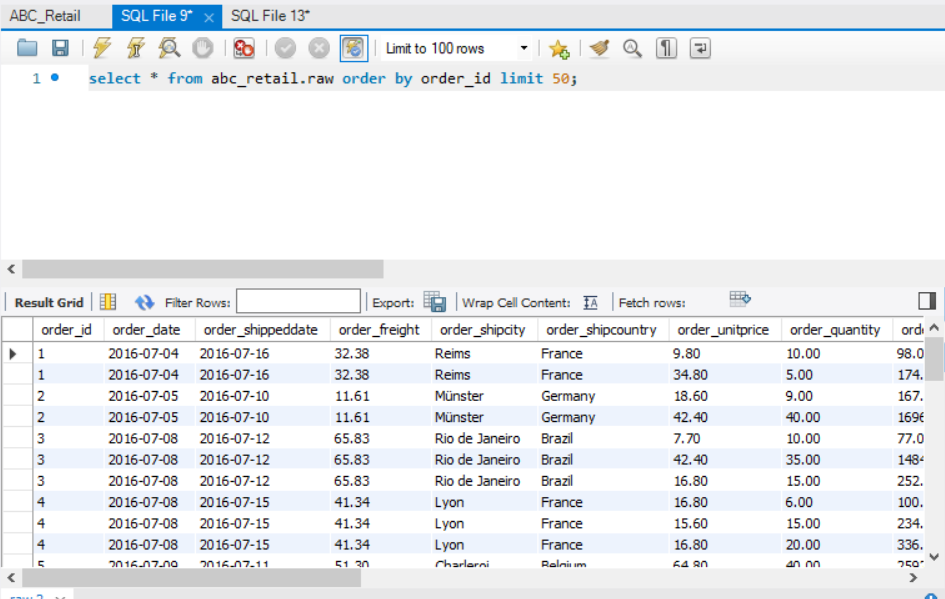
Check number of data in table **raw**.



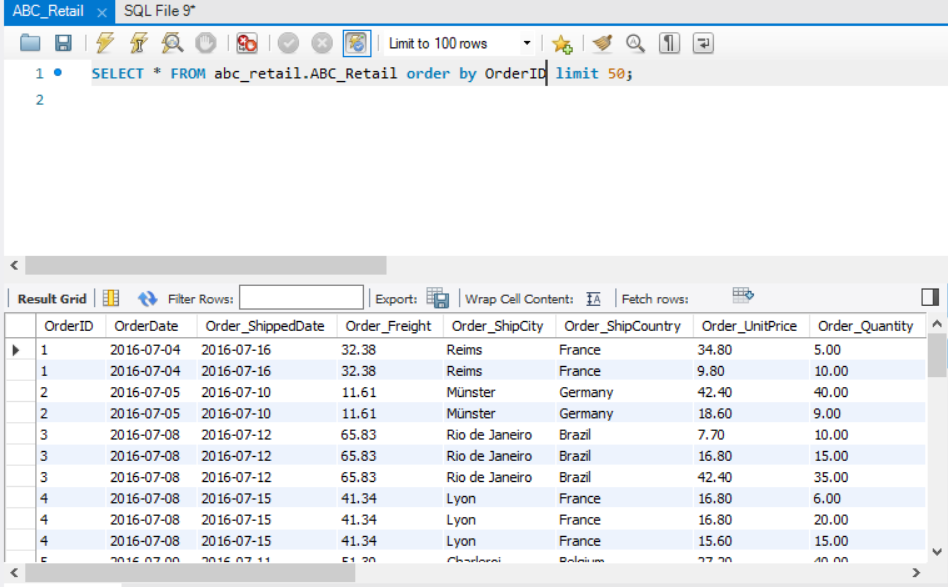
Check number of data in table **ABC\_Retail**.



Data in table **raw**.



Data in table **ABC\_Retail**.



## Create MyCube table from the denormalized table

Since **select … into** is not valid in our MySQL version, detail reasons could be found at <https://stackoverflow.com/questions/2949653/select-into-and-undeclared-variable-error>

We just use **create** then **insert into … select …**

-- create table

CREATE TABLE abc\_retail.MyCube(

    ThisYear year default null,

    Region varchar(255),

    Product varchar(255),

    Sales numeric(20,2)

);

-- Create MyCube table from the denormalized table

INSERT INTO abc\_retail.MyCube

(

    ThisYear, Region, Product, Sales

)

SELECT

    year(OrderDate) as ThisYear

    ,Order\_ShipCountry as Region

    ,ProductName as Product

    ,Order\_Amount as Sales

FROM

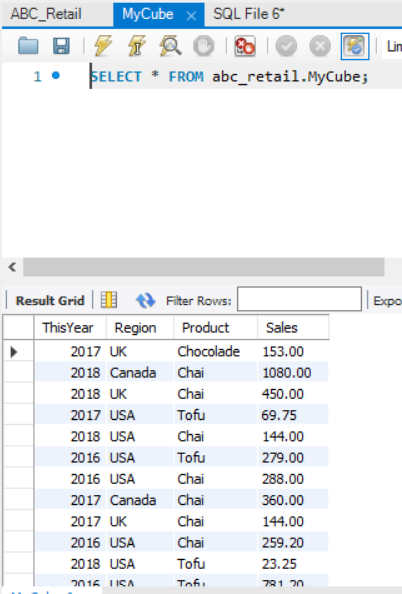
    abc\_retail.ABC\_Retail

WHERE

    Order\_ShipCountry in ('USA','Canada','UK')

    and ProductName in ('Chai','Tofu','Chocolade');

Check the result:



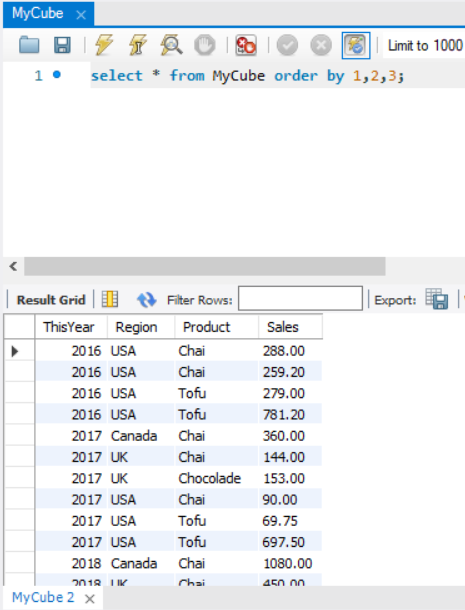
## Run all the attached OLAP Operators against the newly created MyCube table

The following OLAP operators could be run in our MySQL database.

And other operations like the **PIVOT** operators could not be run on MySQL.

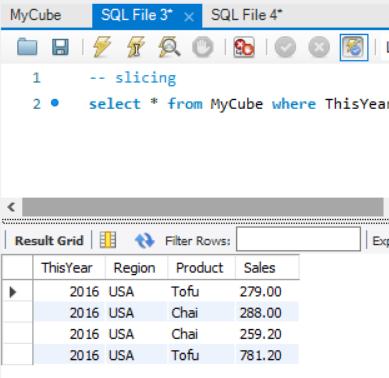
-- table

select \* from MyCube order by 1,2,3



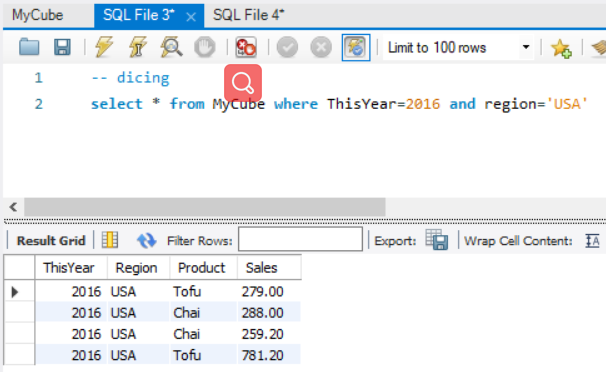
-- slicing

select \* from MyCube where ThisYear=2016



-- dicing

select \* from MyCube where ThisYear=2016 and region='USA'



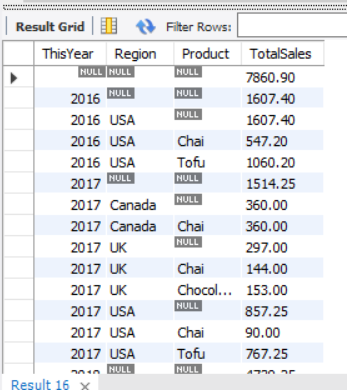
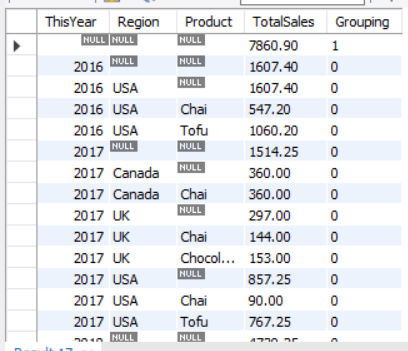
-- group by with rollup

SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

FROM MyCube

GROUP BY ThisYear, Region, Product with rollup

ORDER BY 1,2,3

-- group by with cube

SELECT ThisQuarter, Region, Product, SUM(Sales)  
as TotalSales--, GROUPING(ThisQuarter) AS 'Grouping'

FROM MyCube

GROUP BY ThisQuarter, Region, Product with cube

ORDER BY 1,2,3

Could be write as:

(

    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY ThisYear, Region, Product with rollup

    UNION

    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY ThisYear, Product, Region with rollup

    UNION

    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY Region, ThisYear, Product with rollup

    UNION

    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY Region, Product, ThisYear with rollup

    UNION

    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY Product, ThisYear, Region with rollup

    UNION

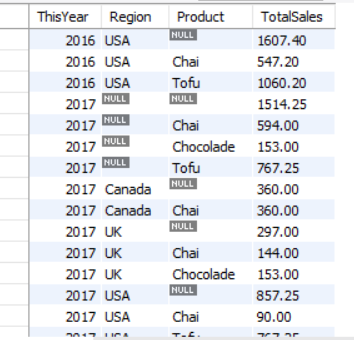
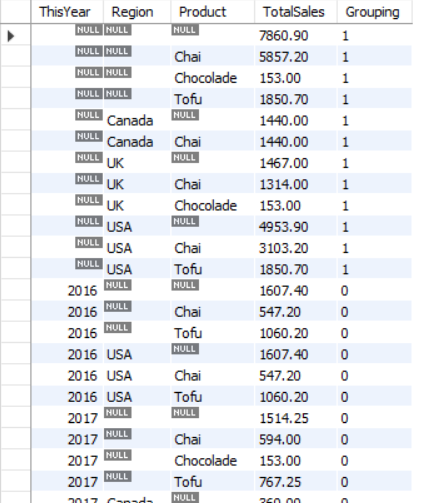
    SELECT ThisYear, Region, Product, SUM(Sales)as TotalSales--, GROUPING(ThisYear) AS 'Grouping'

    FROM MyCube

    GROUP BY Product, Region, ThisYear with rollup

)

ORDER BY 1,2,3;

-- group by grouping sets

SELECT ThisQuarter, Region, SUM(Sales) as TotalSales

FROM MyCube

GROUP BY GROUPING SETS ((ThisQuarter), (Region))

ORDER BY 1,2

Is the same as:

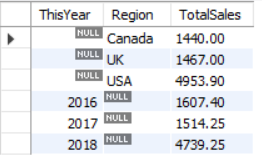
--

SELECT ThisYear, NULL as Region, SUM(Sales) as TotalSales FROM MyCube GROUP BY ThisYear

UNION ALL

SELECT NULL, Region, SUM(Sales) as TotalSales FROM MyCube GROUP BY Region

ORDER BY 1,2



-- Ranking

SELECT

    Product, Sales

    , RANK() OVER (ORDER BY Sales ASC) as RANK\_SALES

    , DENSE\_RANK() OVER (ORDER BY Sales ASC) as DENSE\_RANK\_SALES

    , PERCENT\_RANK() OVER (ORDER BY Sales ASC) as PERC\_RANK\_SALES

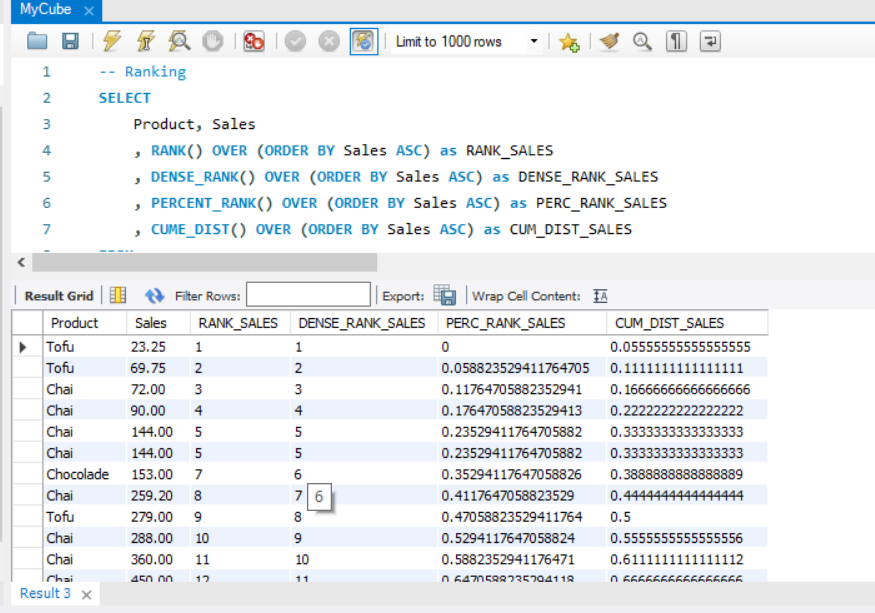
    , CUME\_DIST() OVER (ORDER BY Sales ASC) as CUM\_DIST\_SALES

FROM

    MyCube

ORDER BY

    RANK\_SALES ASC



-- Windowing

SELECT

    ThisYear, Region, Sales

    , AVG(Sales) OVER (PARTITION BY Region ORDER BY ThisYear) AS Sales\_Avg

FROM

    MyCube

ORDER BY

    Region, ThisYear, Sales\_Avg



-- Windowing

SELECT

    ThisYear, Region, Sales

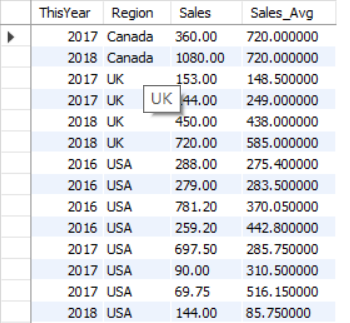
    , AVG(Sales) OVER (PARTITION BY Region ORDER BY ThisYear ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS Sales\_Avg

FROM

    MyCube

ORDER BY

    Region, ThisYear, Sales\_Avg



## If your DBMS does not support PIVOT, write a SQL script to produce the same result

Our DBMS is MySQL server.

-- pivot query

select

    Product, Q1, Q2, Q3, Q4

from

    MyCube PIVOT(SUM(Sales) FOR ThisQuarter IN (Q1,Q2,Q3,Q4)) AS P

Could be write as:

-- pivot query

select

    Product,

    SUM( IF(ThisYear=2016,Sales,0) ) As `2016`,

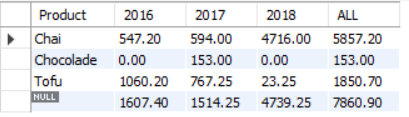
    SUM( IF(ThisYear=2017,Sales,0) ) As `2017`,

    SUM( IF(ThisYear=2018,Sales,0) ) As `2018`,

    SUM(Sales) As 'ALL'

from MyCube

group by Product with rollup;



-- pivot query

SELECT Product, Region, Q1, Q2, Q3, Q4

FROM

(SELECT Product, Region, ThisQuarter, Sales FROM MyCube) AS p

PIVOT

(sum(Sales) FOR ThisQuarter IN (Q1,Q2,Q3,Q4)) AS pvt

Could be write as:

SELECT

    Product,

    Region,

    sum( IF(ThisYear=2016,Sales,0) ) As `2016`,

    sum( IF(ThisYear=2017,Sales,0) ) As `2017`,

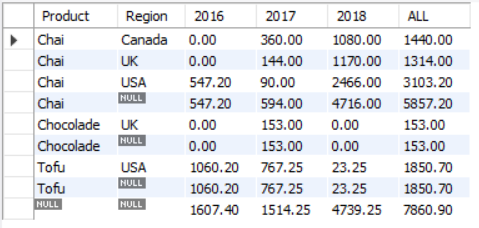
    sum( IF(ThisYear=2018,Sales,0) ) As `2018` ,

    sum(Sales) As 'ALL'

FROM

(SELECT Product, Region, ThisYear, Sales FROM MyCube) AS p

GROUP BY Product, Region WITH ROLLUP;

****