SQL Programming Tutorial

Agenda (10/3/2018)

* Union, Intersect, and Minus
* Joins (Inner, Outer, Full)
* Case Structures
* Working With Dates

# UNION, INTERSECT, and MINUS

# Business Requirement: Find the top 2 most profitable and losing products in terms of profits for 2013.

Select \* from

(SELECT Prodname, extract(year from factdate)||' max' as SalesYear, sum(actprofit)

FROM factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

ORDER BY sum(actsales) DESC)

WHERE ROWNUM <= 3

**UNION**

Select \* from

(SELECT Prodname, extract(year from factdate)||' min' as SalesYear, sum(actprofit)

FROM factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

ORDER BY sum(actsales))

WHERE ROWNUM <= 3;

PIVOT (

SUM(sumprofit)

for salesyear IN ('2013 min', '2013 max')

);

# Business Requirement: Find the products that made over 10000 either in 2012 or 2013.

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000

UNION

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2012

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000;

Business Requirement: Find the products that made over 10000 in BOTH 2012 and 2013.

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000

Intersect

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2012

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000;

Business Requirement: Find the products that made over 10000 only in 2013 but not in 2012.

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000

MINUS

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2012

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000;

Business Requirement: Find the products that made over 10000 only in 2012 but not in 2013.

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2012

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000

MINUS

SELECT prodname

from factcoffee, prodcoffee

WHERE factcoffee.productid = prodcoffee.productid AND extract(year from factdate) = 2013

GROUP BY Prodname, extract(year from factdate)

HAVING sum(actprofit) > 10000;

Joins

Provides an easy way to combine data from two tables. This action can also be done using traditional operators like = and (+), but Join makes it easier to read the code, because it is more human readable code.

There are three types of Joins

Inner Join: Joins records that are equally matched between two tables

Outer Join: Joins one records in one table that are also present in the second table. Two types of outer joins, the left outer join and the right outer join. Better illustrated below with an example.

To explain different types of Join statements, let us create two views based on the Coffee tables that we have imported. These two views contain products that have total profits greater than $10,000 for each 2013 and 2012, respectively.

CREATE OR REPLACE View firstprod AS

(SELECT productid proda, sum(actprofit) Sum2013

FROM factcoffee

WHERE extract(year from factdate) = 2013

GROUP BY productid

HAVING sum(actprofit) > 10000);

CREATE OR REPLACE View Secprod AS

(SELECT productid prodb, sum(actprofit) Sum2012

FROM factcoffee

WHERE extract(year from factdate) = 2012

GROUP BY productid

HAVING sum(actprofit) > 10000);

Business Requirement: Find products that have total profits greater than $10,000 in both 2012 and 2013.

To achieve this goal, we would do an outer join. Remember from our earlier definition, an outer join combines records that are present in both tables. Without using the Join keyword, you would perform this query as

SELECT Proda, Prodb

FROM Firstprod, Secprod

WHERE firstprod.proda = secprod.prodb;

The same can be written with JOIN like

SELECT proda, prodb

FROM firstprod INNER JOIN Secprod ON firstprod.proda = secprod.prodb;

The above matches records based on proda and prodb.

Business Requirement: Find products that have total profits greater than $10,000 in 2012 and 2013.

The one below displays all Proda from firstprod and those that matches in secprod:

SELECT proda, prodb

FROM firstprod LEFT OUTER JOIN Secprod ON firstprod.proda = secprod.prodb;

The above is equivalent to:

SELECT Proda, Prodb

FROM Firstprod, Secprod

WHERE firstprod.proda = secprod.prodb (+);

The one below displays all Prodb from Secprod and those that matches in firstprod:

SELECT proda, prodb

FROM firstprod RIGHT OUTER JOIN Secprod ON firstprod.proda = secprod.prodb;

The above is equivalent to:

SELECT Proda, Prodb

FROM Firstprod, Secprod

WHERE firstprod.proda (+) = secprod.prodb;

The one below displays all records from firstprod and all records from secprod:

SELECT proda, prodb

FROM firstprod FULL JOIN Secprod ON firstprod.proda = secprod.prodb;

Business Requirement: Classify different products based on total sales as High, Medium, Low

SELECT prodname, Sum(actsales) as SumSales, (

CASE WHEN Sum(actsales) > 40000 THEN 'High'

     WHEN Sum(actsales) BETWEEN 20000 AND 40000 then 'Medium'

     ELSE 'Low'

     END) SalesLevel

FROM prodcoffee, factcoffee

WHERE prodcoffee.productid = factcoffee.productid AND extract(year from

factdate) = 2012

GROUP BY prodname;

Business Requirement: Classify states with total sales and total profits based on profit margins and sales as Excellent, Good, OK and Horrible

SELECT statename, SUM(Actsales) as SumSales, SUM(ActProfit) as SumProfit,

ROund(100\*Sum(ActProfit)/Sum(ActSales),2) as ProfMargin,

    (CASE

    WHEN(Round(100\*Sum(ActProfit)/Sum(ActSales),2) > 25) AND (SUM(Actsales) >

25000) THEN 'Excellent'

    WHEN Round(100\*Sum(ActProfit)/Sum(ActSales),2) > 20 AND SUM(Actsales)

BETWEEN 10000 AND 25000 THEN 'Good'

    WHEN Round(100\*Sum(ActProfit)/Sum(ActSales),2) < 10 THEN  'Horrible'

    ELSE 'OK'

    END) PerGrade

FROM factcoffee, areacode, states

WHERE factcoffee.areaid = areacode.areaid AND states.stateid = Areacode.stateid

AND Extract(year from factdate) = 2012

GROUP BY statename

ORDER BY PerGrade;

Business Requirement: Display actual sales for all dates between June 1, 2012 and June 1, 2013

SELECT productid, sum(actsales)

from factcoffee

WHERE to\_char(factdate, 'mm/dd/yyyy') >= '06/01/2012' AND to\_char(factdate, 'mm/dd/yyyy') <='06/01/2013'

GROUP BY productid

ORDER BY productid;

Business Requirement: Display actual sales for the months of January and February

SELECT productid, sum(actsales)

from factcoffee

WHERE to\_char(factdate, 'mon') LIKE 'jan' OR to\_char(factdate, 'MON') ='FEB'

GROUP BY productid

ORDER BY productid;