Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

OLAP та сховища даних

**ЗВІТ**

з лабораторної роботи № 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *Гр. ІП-71 Кувiчка Максим Євгенович* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *Ст. викладач Олійник Юрій Олександрович* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2019

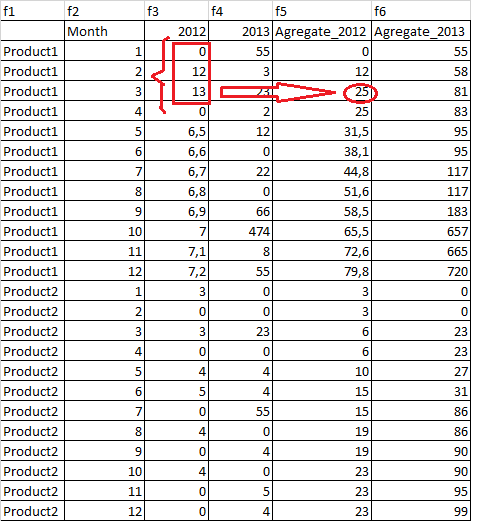
**Завдання**

Виконати запуск скриптів KP2\script(create\_scheme).sql та КР2\tables\_data.sql

Для ознайомлення з структурою БД використовуйте модель Sales-normalized.pdm.

Необхідно сформувати наступні аналітичні запити:

1. Показати ранг кожного товару у групі/категорії (відповідно до зменшення ціни в групі). Запит реалізувати через функції rank() та dense\_rank(). Порівняти результати виконання. **Крім того сформуйте запит без використання аналітичних функцій.**
2. За допомогою аналітичного SQL сформуйте запит для виведення списку із трьох найдешевших товарів у кожній групі. Крім того сформуйте запит без використання аналітичних функцій.
3. F1 – продукти по яким здійснюються продажі, f2 – номер місяця, f3 – продажі по товару за певний місяць 2012 року, f4 – продажі по товару за певний місяць 2013 року, f5 – наростаючий підсумок продажів по товару за певний місяць 2012 року, f5 – наростаючий підсумок продажів по товару за певний місяць 2013 року. Крім того сформуйте запит без використання аналітичних функцій.



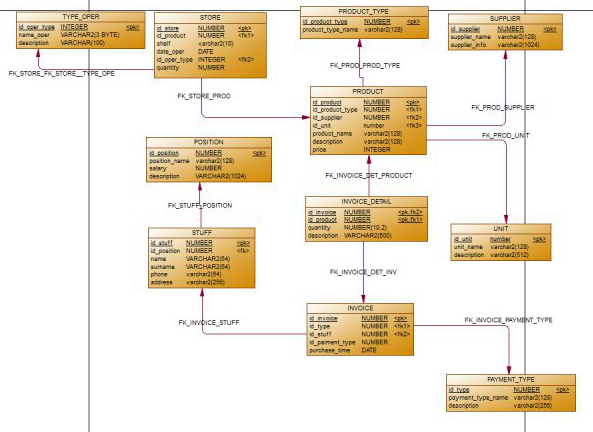
1. Показати, які товари по кожній групі мають найбільші та найменші продажі. Крім того сформуйте запит без використання аналітичних функцій.
2. По кожному товару підрахувати кількість товарів, у яких вартість вища від даного товару у діапазоні від 5 до 10 включно. Наприклад якщо прайс товару 6, то знайти кількість товарів, у яких діапазон прайсів від 11 до 16.
3. По кожному товару вивести першу(f3) та останню(f4) дату продажів по кожному місяцю, cуму продажів (f5), відсоток від річної суми(f6).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| f1 | f2 | f3 | f4 | f5 | f6 |
| Product | Month | Date\_First | Date\_Last | S2013 | P2013 |
| Product1 | 1 | 01.01.2013 | 13.01.2013 | 24,00 | 12% |
| Product1 | 2 | 03.02.2013 | 15.02.2013 | 26,00 | 13% |

1. За допомогою ROLLUP та CUBE виведіть проміжні суми продажу по групам товарів.

**2. Звіт**

1. Діаграма предметної області



1. SQL – запити.

SELECT id\_product\_type,

product\_name,

price,

RANK() OVER(

PARTITION BY id\_product\_type

ORDER BY price DESC

)price\_rank

FROM product;

SELECT id\_product\_type, --RANK without analytical functions

product\_name,

price,

(SELECT

COUNT(\*) + 1

FROM

product p2MIN

WHERE p2.id\_product\_type = p.id\_product\_type AND

p2.price > p.price

) AS price\_rank

FROM product p

ORDER BY p.id\_product\_type, price\_rank;

SELECT id\_product\_type,

product\_name,

price,

DENSE\_RANK() OVER(

PARTITION BY id\_product\_type

ORDER BY price DESC

)price\_rank

FROM product;

SELECT id\_product\_type, --DENSE\_RANK without analytical functions

product\_name,

price,

(SELECT

COUNT(DISTINCT price) + 1

FROM

product p2

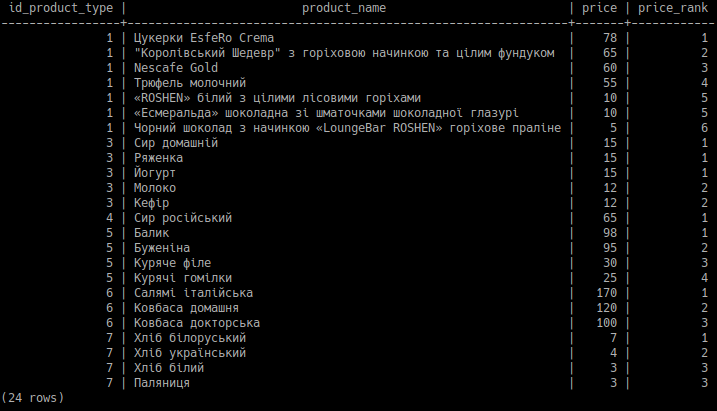
WHERE p2.id\_product\_type = p.id\_product\_type AND

p2.price > p.price

) AS price\_rank

FROM product p

ORDER BY p.id\_product\_type, price\_rank;



SELECT \* FROM(

SELECT id\_product\_type,

product\_name,

price,

DENSE\_RANK() OVER(

PARTITION BY id\_product\_type

ORDER BY price

)price\_rank

FROM product) AS RANK

WHERE RANK.price\_rank <= 3;

-- without analytical functions

SELECT \* FROM(

SELECT id\_product\_type,

product\_name,

price,

(SELECT

COUNT(DISTINCT price) + 1

FROM

product p2

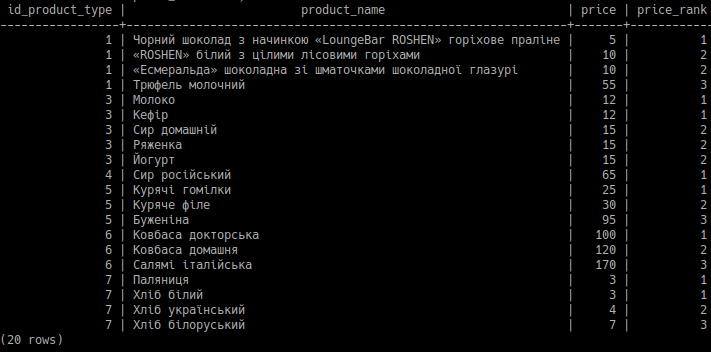
WHERE p2.id\_product\_type = p.id\_product\_type AND

p2.price < p.price

) AS price\_rank

FROM product p) AS RANK

WHERE RANK.price\_rank <= 3;



WITH invoices\_2012 AS (

SELECT id\_product, SUM(quantity) AS quant, EXTRACT(MONTH FROM i.purchase\_time) AS month

FROM invoice i

JOIN invoice\_detail idet

ON i.id\_invoice = idet.id\_invoice

WHERE EXTRACT(YEAR FROM i.purchase\_time) = 2012

GROUP BY id\_product, month

ORDER BY id\_product, month

),

invoices\_2013 AS (

SELECT id\_product, SUM(quantity) AS quant, EXTRACT(MONTH FROM i.purchase\_time) AS month

FROM invoice i

JOIN invoice\_detail idet

ON i.id\_invoice = idet.id\_invoice

WHERE EXTRACT(YEAR FROM i.purchase\_time) = 2013

GROUP BY id\_product, month

ORDER BY id\_product, month

)

SELECT p.product\_name,

curr\_m,

COALESCE(i2012.quant \* p.price, 0) AS "2012",

COALESCE(i2013.quant \* p.price, 0) AS "2013",

SUM(COALESCE(i2012.quant \* p.price, 0)) OVER(

PARTITION BY p.product\_name

ORDER BY curr\_m

) Analytic\_2012,

SUM(COALESCE(i2013.quant \* p.price, 0)) OVER(

PARTITION BY p.product\_name

ORDER BY curr\_m

) Analytic\_2013

FROM product p

cross JOIN generate\_series(1, 12) curr\_m

LEFT JOIN invoices\_2012 i2012 ON i2012.month = curr\_m AND i2012.id\_product = p.id\_product

LEFT JOIN invoices\_2013 i2013 ON i2013.month = curr\_m AND i2013.id\_product = p.id\_product

ORDER BY p.product\_name, curr\_m;

-- without analytical functions

SELECT p.product\_name,

curr\_m,

COALESCE(i2012.quant \* p.price, 0) AS "2012",

COALESCE(i2013.quant \* p.price, 0) AS "2013",

(SELECT SUM(COALESCE(i2012\_2.quant \* p2.price, 0))

FROM product p2

cross JOIN generate\_series(1, 12) curr\_m\_2

LEFT JOIN invoices\_2012 i2012\_2 ON i2012\_2.month = curr\_m\_2 AND i2012\_2.id\_product = p2.id\_product

WHERE p2.product\_name = p.product\_name AND

curr\_m\_2 <= curr\_m

) AS Analytic\_2012,

(SELECT SUM(COALESCE(i2013\_2.quant \* p2.price, 0))

FROM product p2

cross JOIN generate\_series(1, 12) curr\_m\_2

LEFT JOIN invoices\_2013 i2013\_2 ON i2013\_2.month = curr\_m\_2 AND i2013\_2.id\_product = p2.id\_product

WHERE p2.product\_name = p.product\_name AND

curr\_m\_2 <= curr\_m

) AS Analytic\_2013

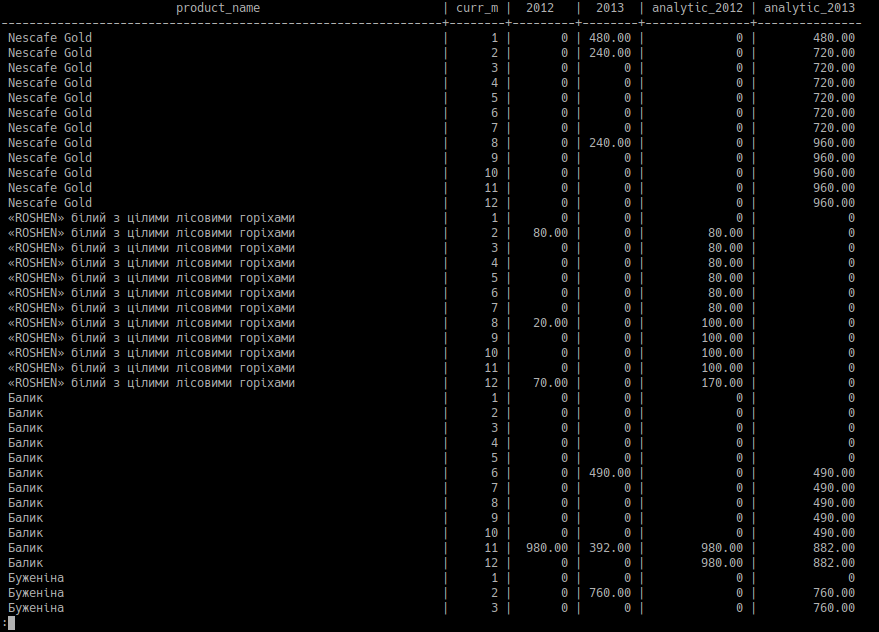
FROM product p

cross JOIN generate\_series(1, 12) curr\_m

LEFT JOIN invoices\_2012 i2012 ON i2012.month = curr\_m AND i2012.id\_product = p.id\_product

LEFT JOIN invoices\_2013 i2013 ON i2013.month = curr\_m AND i2013.id\_product = p.id\_product

ORDER BY p.product\_name, curr\_m;



SELECT DISTINCT RANK.product\_type\_name,

RANK.product\_name,

(CASE

WHEN RANK.category\_max = 1 then 'biggest sales'

else 'smallest sales'

end)

FROM (

SELECT pn.product\_type\_name,

p.product\_name,

SUM(p.price \* i.quantity),

RANK() OVER(

PARTITION BY pn.product\_type\_name

ORDER BY SUM(p.price \* i.quantity) DESC

) category\_max,

RANK() OVER(

PARTITION BY pn.product\_type\_name

ORDER BY SUM(p.price \* i.quantity)

) category\_min

FROM product p

JOIN invoice\_detail i

ON p.id\_product = i.id\_product

JOIN product\_type pn

ON p.id\_product\_type = pn.id\_product\_type

GROUP BY pn.product\_type\_name, p.product\_name

ORDER BY pn.product\_type\_name, p.product\_name

) AS RANK

WHERE RANK.category\_max = 1 OR RANK.category\_min = 1;

-- without analytical functions

SELECT DISTINCT RANK.product\_type\_name,

RANK.product\_name,

(CASE

WHEN RANK.category\_max = 1 THEN 'biggest sales'

ELSE 'smallest sales'

END)

FROM (

SELECT pn.product\_type\_name,

p.product\_name,

SUM(p.price \* i.quantity),

(SELECT

COUNT(\*) + 1

FROM

(SELECT p2.product\_name FROM

product p2

JOIN invoice\_detail i2

ON p2.id\_product = i2.id\_product

JOIN product\_type pn2

ON p2.id\_product\_type = pn2.id\_product\_type

WHERE pn2.product\_type\_name = pn.product\_type\_name

GROUP BY p2.product\_name

HAVING SUM(p2.price \* i2.quantity) > SUM(p.price \* i.quantity)) a

) AS category\_max,

(SELECT

COUNT(\*) + 1

FROM

(SELECT p2.product\_name FROM

product p2

JOIN invoice\_detail i2

ON p2.id\_product = i2.id\_product

JOIN product\_type pn2

ON p2.id\_product\_type = pn2.id\_product\_type

WHERE pn2.product\_type\_name = pn.product\_type\_name

GROUP BY p2.product\_name

HAVING SUM(p2.price \* i2.quantity) < SUM(p.price \* i.quantity)) a

) AS category\_min

FROM product p

JOIN invoice\_detail iuct\_name,

price,

COUNT(\*) OVER(

ORDER BY price

RANGE BETWEEN 5 FOLLOWING AND 10 FOLLOWING

)

FROM product

ORDER BY price;

ON p.id\_product = i.id\_product

JOIN product\_type pn

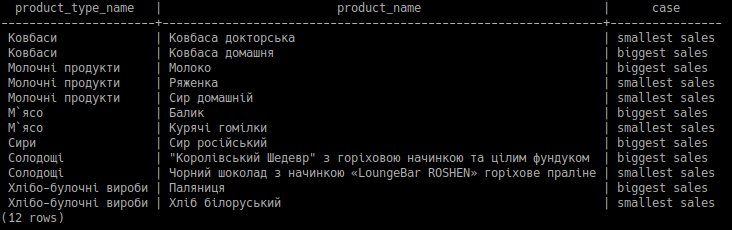
ON p.id\_product\_type = pn.id\_product\_type

GROUP BY pn.product\_type\_name, p.product\_name

ORDER BY pn.product\_type\_name, p.product\_name

) AS RANK

WHERE RANK.category\_max = 1 OR RANK.category\_min = 1;



SELECT product\_name,

price,

COUNT(\*) OVER(

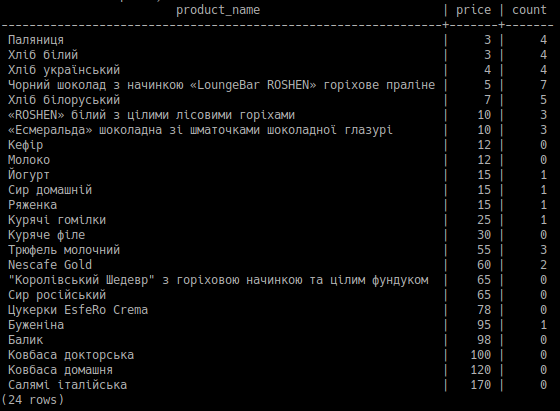
ORDER BY price

RANGE BETWEEN 5 FOLLOWING AND 10 FOLLOWING

)

FROM product

ORDER BY price;



WITH invoices\_2013 AS (

SELECT id\_product,

SUM(quantity) AS quant,

EXTRACT(MONTH FROM i.purchase\_time) AS month,

MIN(i.purchase\_time) AS first,

MAX(i.purchase\_time) AS last

FROM invoice i

JOIN invoice\_detail idet

ON i.id\_invoice = idet.id\_invoice

WHERE EXTRACT(YEAR FROM i.purchase\_time) = 2013

GROUP BY id\_product, month

ORDER BY id\_product, month

)

SELECT p.product\_name,

curr\_m,

i.first,

i.last,

COALESCE(SUM(i.quant \* p.price) OVER(

PARTITION BY p.id\_product

ORDER BY curr\_m

), 0) s2013,

concat(ROUND(COALESCE(SUM(i.quant \* p.price) OVER(

PARTITION BY p.id\_product

ORDER BY curr\_m

), 0) /

COALESCE(SUM(i.quant \* p.price) OVER (

PARTITION BY p.id\_product

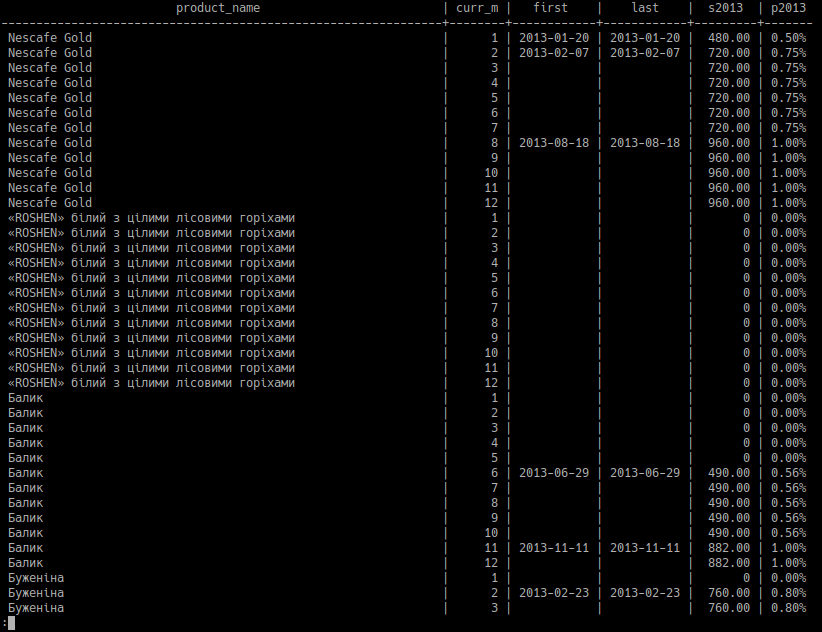
), 1), 2), '%') p2013

FROM product p

cross JOIN generate\_series(1, 12) curr\_m

LEFT JOIN invoices\_2013 i ON i.month = curr\_m AND i.id\_product = p.id\_product

ORDER BY p.product\_name, curr\_m;



SELECT pt.product\_type\_name, p.product\_name, SUM(i.quantity \* p.price)

FROM product p

JOIN product\_type pt

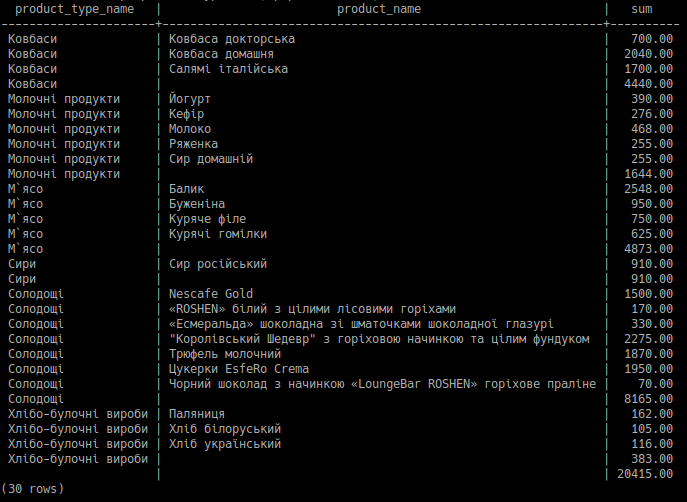
ON p.id\_product\_type = pt.id\_product\_type

JOIN invoice\_detail i

ON p.id\_product = i.id\_product

GROUP BY ROLLUP (pt.product\_type\_name, p.product\_name)

ORDER BY pt.product\_type\_name, p.product\_name;



SELECT p.product\_name, EXTRACT(YEAR FROM i.purchase\_time) AS year, SUM(idet.quantity \* p.price)

FROM product p

JOIN invoice\_detail idet

ON p.id\_product = idet.id\_product

JOIN invoice i

ON i.id\_invoice = idet.id\_invoice

GROUP BY CUBE (p.product\_name, EXTRACT(YEAR FROM i.purchase\_time))

ORDER BY p.product\_name, year;

