PrácticaS DML

**1ºRel (DML Basic SQL sentences)**

1.Full name of those customers belonging to the country identified by the id 52778. 2039 rows

select count (\*) from (select cust\_first\_name,cust\_last\_name from customers where country\_id = 52778);

2.Full name and descriptions of the products whose category is not 205 (the largest category). 53 rows. Notice that both attributes have the same content.

select count (\*) from (select prod\_name, prod\_desc from products where prod\_category\_id != 205);

3.Full name of the customers having an email address in the server "apple.com". The mail can be written in uppercase or lowercase.1897 rows

select count (\*) from (select cust\_first\_name, cust\_last\_name from customers where lower (cust\_email) like '%apple.com');

4.Full name of those customers that have not provided their marital status. 17428 rows. Write the query using the IS NULL operator and using = NULL. Do they have the same behaviour?

select count (\*) from (select cust\_first\_name, cust\_last\_name from customers where cust\_marital\_status IS NULL);

5.Show the sales of the product 45. Show the list chronologically ordered by date and also by the customer code (ascending). 10742

select \* from sales where prod\_id = 45 order by time\_id, cust\_id;

6.Products in the category 205. Provide their names and the proportion between the catalog price and the minimum price (provide the data in percentage, with the % at the end to indicate it). 28 rows

select prod\_name, Round((prod\_list\_price/prod\_min\_price)\*100,2) || '%' "Proportion" from products where prod\_category\_id=205;

**2ºRel (DML more SQL sentences)**

1.List the information of all the sales made in the last century (we consider that the current century began on Jan 1st 2000).

select \* from sales where time\_id < to\_date ('01/01/2000');

2.Information of the costs that was stored in the database at most 22 years ago. SH.V\_FUN\_2 (the solution is wrong. It returns 0 results. You may get 33700 lines.

select \* from costs where ((months\_between(sysdate,time\_id))/12)<22;

3. Get a list with the id of the customer and the id of the product sold and the date of the sale. Show the amount of three-years period spent from that day up to now. Name this column TRIENNIUM. SH.V\_FUN\_3

select cust\_id, prod\_id, time\_id, floor((months\_between(sysdate,time\_id)/36)) from sales;

4. Show the name of all the products containing the (verbatim) tag "Phone". In the list, substitute such tag by "Smartphone". SH.V\_FUN\_4

select replace(prod\_name, ‘Phone’, ‘Smartphone’) from products;

5. Get a list of the customers that didn't submit their marital status. You have to inform about this situation and the name of the customer, following the schema: "The customer ..........(full name of the customer) didn't provide .......(his or her) marital status. You can use the concat function (||) and also the DECODE function. The scape code for the symbol ' in Oracle is ''. SH.V\_FUN\_5

select ‘The customers’ || cust\_first\_name || ‘ ‘ || cust\_last\_name || ‘ didn’t provide ‘ || decode(cust\_gender, ‘M’, ‘his’, ‘her’) || ‘ marital status’ “Non submitted” from customers where cust\_marital\_status is null;

6. Show the information of the customers (get the first and last name, ID and year of birth) living in the cities Aachen, Aalborg or Aalen. Those customers with no year of birth provided will appear with the label "NON SUBMITTED". Notice that the NVL function must have all its parameters of the same data\_type.

Can you simplify your query by using the property that these three cities begin with "Aa" (they are no more cities fulfilling this property). SH.V\_FUN\_6

select cust\_first\_name, cust\_last\_name, cust\_id, nvl(to\_char(cust\_year\_of\_birth), 'non submitted') "cust\_year\_of\_birth" from customers where cust\_city like ‘Aa%’;

7. List the information of the sales made on the second Monday of each month. Provide the ID of the customer, ID of the product and date. Take into account that the DAY token in the TO\_CHAR function for the dates can provides some extra spaces because it renders a fixed size output. SH.V\_FUN\_7

Select cust\_id, prod\_id, time\_id, from sales where to\_char(time\_id,’W’)=2 and to\_char(time\_id,’D’)=1;

**3ºRel (Set Operators)**

1.List all the tags that appears in the columns EDUCATION, OCCUPATION or MARITAL\_STATUS (they belongs to two different tables). Provide the output in two columns, where the first one is the tag and the second one is the type (EDUCATION, OCCUPATION or MARITAL\_STATUS) [SH.V\_SETOP\_1]

Select education tag, ‘EDUCATION’ type from supplementary\_demographics union

Select occupation, ‘OCCUPATION’ from supplementary\_demographics union

Select cust\_marital\_status, ‘MARITAL STATUS’ from customers;

2.List the PROD\_ID of the products that haven't ever been purchased by any customer [SH.V\_SETOP\_2]

Select prod\_id from products minus select prod\_id from sales;

3.List the CUST\_ID of the customers having extra information in the SUPPLEMENTARY\_DEMOGRAPHICS table [SH.V\_SETOP\_3]

select distinct cust\_id from customers

minus

select distinct cust\_id from supplementary\_demographics;

4.List the descriptions of the products appearing exactly in the same way as a PROD\_DESC, a PROD\_CAT\_DESC and a PROD\_SUBCAT\_DESC [SH.V\_SETOP\_4]

select prod\_name from products

where prod\_name=prod\_subcategory\_desc and prod\_category\_desc=prod\_subcategory\_desc;

//OTRA MANERA

select prod\_desc from products

intersect

select prod\_subcategory\_desc from products

intersect

select prod\_category\_desc from products;

5.List the PROD\_ID and PROD\_NAME of the products having the same NAME and DESC [SH.V\_SETOP\_5]

select prod\_id,prod\_name from products

intersect

select prod\_id,prod\_desc from products;

**4ºRel (DML Join (I))**

1.Provide a list of all the sales from customers born on 1990. Provide the quantity, the product id and the customer id. SH.V\_JOIN\_1

select distinct quantity\_sold, cust\_id, prod\_id from sales join customers using (cust\_id)

where cust\_year\_of\_birth=1990;

2.List the sales made on Monday. Do not use the TO\_CHAR function to get the day. Use the TIMES table where all date are categorized. Provide the the same attributes that appears in the previous query. Order it by the amount of the sale. SH.V\_JOIN\_2

select distinct quantity\_sold, cust\_id, prod\_id from sales join customers using (cust\_id) join times using (time\_id)

where day\_number\_in\_week=1;

3.List the CUST\_ID and the full name of the persons who spend more than 1500€ in a single sale. Do you believe that the DISTINCT clause is needed in this query? SH.V\_JOIN\_3

select distinct cust\_id,cust\_first\_name,cust\_last\_name from customers join sales using (cust\_id)

where amount\_sold>1500;

4.At what dates we have sale something using the Digital Marketing channel? (described with the Internet key word) Note that repeat information may appear. Avoid redundant information in the output. SH.V\_JOIN\_4

select time\_id from sales join channels using (channel\_id)

where channel\_desc = 'Internet' order by time\_id;

5.List the customer living in Singapore. Provide first name and last name. SH.V\_JOIN\_5

select distinct cust\_first\_name,cust\_last\_name,cust\_id from customers join countries using (country\_id)

where country\_name='Singapore';

6.List the full name of the persons born in 1990 and spending more than 500€ in one single sale. Do not remove repeated customers names.SH.V\_JOIN\_6

select cust\_first\_name,cust\_last\_name from customers join sales using (cust\_id)

where cust\_year\_of\_birth=1990 and amount\_sold>500;

7.List the name of the successful promotions, i.e. those ones that get at least one sale. Provide promotion name and id. SH.V\_JOIN\_7

select distinct promo\_id,promo\_name from promotions join sales using (promo\_id)

where quantity\_sold>=1;

**5ºRel (DML Join (II))**

1.List the name of the products sold by using a Tele-sale channel. [SH.V\_MULJOIN\_1]

select distinct prod\_name

from sales

join products using (prod\_id)

join channels using (channel\_id)

where channel\_desc='Tele Sales';

2.List the name of the products bought by customers living in Middle East. [SH.V\_MULJOIN\_2]

select distinct prod\_name

from sales join customers using (cust\_id) join products using (prod\_id) join countries using (country\_id)

where country\_subregion = 'Middle East';

3.Provide the full name of the customers with a credit limit of 11000€ that have bought by using a digital marketing channel some product costing a big deal (more than 1000€). The product cost has to be checked with the PROD\_LIST\_PRICE attribute of the PRODUCTS table. [SH.V\_MULJOIN\_3]

select distinct cust\_first\_name, cust\_last\_name

from customers join sales using (cust\_id) join channels using (channel\_id) join products using (prod\_id)

where cust\_credit\_limit=11000 and channel\_desc = 'Internet' and prod\_list\_price>1000;

4.List the name of the products bought on a leap year by some customer with a 15000€ credit limit. The product cannot have a list-price greater than 30. [SH.V\_MULJOIN\_4]

select distinct prod\_name

from products join sales using (prod\_id) join customers using (cust\_id) join times using (time\_id)

where cust\_credit\_limit=15000 and prod\_list\_price<=30 and days\_in\_cal\_year=366;

5.A non-foresight customer is the one who has to go out on Christmas eve to buy in a direct sale a camera battery. Search the year of birth of the non-foresight customers. [SH.V\_MULJOIN\_5]

select distinct cust\_year\_of\_birth

from customers join sales using (cust\_id) join channels using (channel\_id) join times using (time\_id) join products using (prod\_id)

where prod\_subcategory='Camera Batteries' and channel\_desc='Direct Sales' and day\_number\_in\_month=24 and calendar\_month\_number=12;

**6ºRel (DML Self-joins.)**

1.List the full name and ID of the customers having another soul-mate in the customer table; i.e. there exists another customer living in the same city, who also was born in their year of birth, has its gender, the same income level, marital status and credit limit. Even more, both are categorized as Active or Inactive in the CUST\_VALID attribute. [V\_SELF\_1]

select distinct c1.cust\_first\_name, c1.cust\_last\_name, c1.cust\_id

from customers c1 join customers c2 using (cust\_year\_of\_birth,cust\_city,cust\_gender,cust\_income\_level,cust\_marital\_status,cust\_credit\_limit,cust\_valid)

WHERE c1.cust\_id < c2.cust\_id;

2.Make the same query but listing pairs of customers. [V\_SELF\_2]

select distinct c1.cust\_first\_name, c1.cust\_last\_name, c1.cust\_id, c2.cust\_first\_name, c2.cust\_last\_name, c2.cust\_id

from customers c1 join customers c2 using (cust\_year\_of\_birth,cust\_city,cust\_gender,cust\_income\_level,cust\_marital\_status,cust\_credit\_limit,cust\_valid)

where c1.cust\_id < c2.cust\_id;

3.List the name and ID of the products having another one in its subcategory and so that both of them have the same list price and min price. [V\_SELF\_3]

select distinct p1.prod\_name, p1.prod\_ID

from products p1 join products p2 using (prod\_subcategory,prod\_list\_price,prod\_min\_price)

where p1.prod\_id<p2.prod\_id;

4.Make the same report but listing pairs of products. [V\_SELF\_4].

select distinct p1.prod\_name, p1.prod\_ID, p2.prod\_name, p2.prod\_ID

from products p1 join products p2 using (prod\_subcategory,prod\_list\_price,prod\_min\_price)

where p1.prod\_id<p2.prod\_id;

5.List the full name and ID of the customers who has bought one product that have also been bought by another soul-mate customer. [V\_SELF\_5]

select distinct c1.cust\_first\_name, c1.cust\_last\_name, c1.cust\_id

from customers c1 join customers c2 using (cust\_year\_of\_birth,cust\_city,cust\_gender,cust\_income\_level,cust\_marital\_status,cust\_credit\_limit,cust\_valid)

join sales s1 on (c1.cust\_id=s1.cust\_id) join sales s2 on (c2.cust\_id=s2.cust\_id)

where c1.cust\_id<>c2.cust\_id and s1.prod\_id=s2.prod\_id;

6.List the name and ID of the products successfully sold although there are other product in its same subcategory with minor price. [V\_SELF\_6]

select distinct p1.prod\_name, p1.prod\_id

from products p1 join products p2 using (prod\_subcategory\_id) join sales s1 on (s1.prod\_id=p1.prod\_id)

where p1.prod\_id<>p2.prod\_id and p1.prod\_list\_price>p2.prod\_list\_price;

**7ºRel (DML Subqueries)**

1.Name of the products where the difference between their cost and their price is more than 600 and such that it have been sold in a sale where the amount\_sold value is greater than 1780. [V\_SUBQ\_1]

select distinct prod\_name from products

where prod\_id in

(select prod\_id from products join costs using (prod\_id) join sales using (prod\_id)

where (unit\_price-unit\_cost)>600 and amount\_sold>1780);

2.Full name and customer id of the customers from USA who bought some product listed in the above query and their salary is greater than 200.000. [V\_SUBQ\_2]

select distinct cust\_id,cust\_first\_name,cust\_last\_name from customers

join sales using (cust\_id)

where prod\_id in

(select distinct prod\_id from sales natural join costs join products using (prod\_id)

where (unit\_price-unit\_cost)>600 and amount\_sold>1780)

and country\_id=52790

and substr(cust\_income\_level,0,1)='J' or substr(cust\_income\_level,0,1)='K' or substr(cust\_income\_level,0,1)='L';

3.List the name of the products bought in USA on 1998 Christmas holidays. Notice that Christmas in USA coincides with the last two weeks of the year. [V\_SUBQ\_3]

select distinct prod\_name

from products

where prod\_id in

(select prod\_id from sales join customers using (cust\_id)

where to\_char(time\_id,'YYYY')='1998' and to\_char(time\_id,'WW') in ('52','53') and country\_id=52790);

4.List the same information of the above query but translating it to Spain. In Spain, as you know, Christmas holidays also includes the first week of the next year. [V\_SUBQ\_4]

select distinct prod\_name

from products

where prod\_id in

(select prod\_id from sales natural join times join customers using (cust\_id)

where (calendar\_year='1999' and calendar\_week\_number=1 and country\_id=52778)

or (calendar\_year='1998' and calendar\_week\_number in (52,53) and country\_id=52778));

5.How can you get all these sales in a single query? [V\_SUBQ\_5]

select distinct prod\_name

from products

where prod\_id in

(select prod\_id from sales join customers using (cust\_id) where to\_char(time\_id,'YYYY')='1998'

and to\_char(time\_id,'WW') in ('52','53') and country\_id=52790)

union

select distinct prod\_name

from products

where prod\_id in

(select prod\_id from sales natural join times join customers using (cust\_id)

where (calendar\_year='1999' and calendar\_week\_number=1 and country\_id=52778)

or (calendar\_year='1998' and calendar\_week\_number in (52,53) and country\_id=52778));

6.List the customer full name and year of birth of the customers who have bought a product of the Hardware category [V\_SUBQ\_6]

select distinct cust\_first\_name,cust\_last\_name,cust\_year\_of\_birth

from customers

where cust\_id in

(select cust\_id from sales join products using (prod\_id) where prod\_category\_id=202);

7.List the time ID of the sales where the customer was some of the previous customers. [V\_SUBQ\_7]

select distinct time\_id

from sales

where cust\_id in

(select cust\_id from sales join products using (prod\_id) where prod\_category\_id=202);

8.Observe that in the previous query we are not searching the date of the product of the Hardware category sales, but the TIME ID of any sale of any of the customers in the previous exercise. Is there any difference? [V\_SUBQ\_8]

**8ºRel (DML Negations)**

1.List the name of the products which are not in the PHOTO category. [V\_NEG\_1]

select distinct prod\_name from products where prod\_category!='Photo';

2.List the full name of all customers not living in USA. [V\_NEG\_2]

select cust\_first\_name,cust\_last\_name from customers

join countries using (country\_id)

where country\_name!='United States of America';

3.List the name of the customers who have bought nothing belonging to the PHOTO category. Notice that if a customer bought a Camera and a printer, he has actually bought a Photo-product. [V\_NEG\_3]

select distinct cust\_first\_name, cust\_last\_name from customers

where cust\_id not in

(select distinct cust\_id from customers

join sales using (cust\_id) join products using (prod\_id)

where prod\_category='Photo');

4.List the cust\_id and the full name of the customers who never use an internet marketing channel. [V\_NEG\_4]

select distinct cust\_id, cust\_first\_name, cust\_last\_name from customers

where cust\_id not in

(select distinct cust\_id from customers

join sales using (cust\_id) join channels using (channel\_id)

where channel\_desc='Internet')

5.A thrifty customer is the one who never spend more than 1780 in the sale amount. List the name of the countries where none of their customers are thrifty. Notice that this query hide a double negation. [V\_NEG\_5]

select distinct country\_name from countries

where country\_id not in

(select distinct country\_id from countries

join customers using (country\_id)

where cust\_id not in

(select distinct cust\_id from customers join sales using (cust\_id)

where amount\_sold<1780))

**9ºRel (DML Aggregate functions)**

1.Calculate the number of customers stored in the database.V\_AGGR\_1

select count(cust\_id) from customers;

2.What is the total number of products?. Does it correspond with the highest ID\_PRODUCTS or there exist some gap in the sequence?V\_AGGR\_2

select count(prod\_id) "NUM\_PRODUCTS" from products;

3.What is the number of different values for the marital status attribute?V\_AGGR\_3

select count(distinct cust\_marital\_status) from customers;

4.Calculate the average of money that the customers spend.V\_AGGR\_4

select avg(amount\_sold) from sales;

5.What is the average of the list price and the minimum price all the products?V\_AGGR\_5

select min(prod\_min\_price), avg(prod\_list\_price) from products;

6.What is the average, the maximum and the minimum of the list price of all the products?V\_AGGR\_6

select avg(prod\_list\_price), max(prod\_list\_price), min(prod\_list\_price) from products;

7.Compute the global amount of money earn by the company and the total number of single sales made during its lifetime.V\_AGGR\_7

select sum(amount\_sold), count(\*) from sales;

**10ºRel (DML Groups)**

1.For each product, list its ID and the total amount of money earn with it. V\_GR\_1

select prod\_id, sum(amount\_sold) from sales group by prod\_id;

2.For each country, list the number of its citizens and the ID of the country. V\_GR\_2

select country\_id, count (\*) from customers group by country\_id;

3.For each promotion, list its ID, the total number of sales corresponding with it, the number of different channels and the number of different customers that use the advantages of such promotion. V\_GR\_3

select promo\_id, count (\*) "TOTAL NUMBER OF SALES",

count (distinct channel\_id) "NUMBER OF CHANNELS",

count (distinct cust\_id) "NUMBER OF DIFFERENT CUSTOMERS"

from sales group by promo\_id;

4.For each promotion and channel, compute the total amount of money received with its sales. V\_GR\_4

select promo\_id, channel\_id ,

sum(amount\_sold) "GLOBAL MONEY SPEND"

from sales group by promo\_id,channel\_id;

5.For each country and gender, calculate the number of customers in such category. Provide also the range of ages (minimum and maximum) for each category. V\_GR\_5

select country\_id, cust\_gender, count (\*),

min(cust\_year\_of\_birth) "Oldest", max(cust\_year\_of\_birth) "Youngest"

from customers group by country\_id,cust\_gender;

6.For each continent, list the number of countries and the number of subcontinents. V\_GR\_6

select country\_region\_id, count (\*),

count(distinct country\_subregion\_id) from countries

group by country\_region\_id;

**11ºRel (DML More about Groups.)**

1.For each product, list the money earn with it. List the name of the product and the global amount of money.SH.V\_MOREGR\_1

select prod\_name, sum(nvl(amount\_sold,0)) money\_earned

from products outer left join sales using (prod\_id)

group by prod\_name,prod\_id;

2.For each country, list the number of its citizens. Show the name of the customer and this number of people.SH.V\_MOREGR\_2

select country\_name,count(\*)citizens from customers

join countries using (country\_id)

group by country\_name;

3.For each month, make a report with the total number of sales (show the month name stored in the TIMES table).SH.V\_MOREGR\_3

select to\_char(time\_id,'Month') ,count(\*) solds from sales

group by to\_char(time\_id,'Month');

4.In the above query, add the global amount of money sold in the month.SH.V\_MOREGR\_4

select to\_char(time\_id,'Month') ,count(\*) solds, sum(amount\_sold)

amount from sales

group by to\_char(time\_id,'Month');

5.Change the above query to structure the information according to the day of the weeks instead of the month.SH.V\_MOREGR\_5

select to\_char(time\_id,'Day') ,count(\*) solds, sum(amount\_sold)

amount from sales

group by to\_char(time\_id,'Day');

6.For each promotion and product, list the number of sales. Show the name of the product and the name of the promotion.SH.V\_MOREGR\_6

select promo\_name, prod\_name, count(\*) from sales

join promotions using (promo\_id) join products using (prod\_id)

group by promo\_name, prod\_name;

**12ºRel (Even more about groups.)**

1.We would like to compute the number of women older than 92 which brought each product. The age must be computed with the sysdate as a reference (do not add manually the current year) to preserve its validity in the future. We also want to show the name of the product in this report. [V\_EVENMABGR\_1]

select prod\_name, count(distinct cust\_id)

from sales join products using (prod\_id)

join customers using (cust\_id)

where cust\_gender='F' and (to\_number(to\_char(sysdate,'YYYY'))-cust\_year\_of\_birth)>92

group by prod\_name;

2.In the database we would like to check the behavior of the products with low price. Particularly we will show for each product with a list price less than 8$ the total amount of money we earn with its sales, i.e. the net benefit. This net benefit is computed by subtracting to the amount sold value the unit cost price. Thus, for each product we compute the total of the money we get as a benefit in all the sales of such product. Show this global net benefit, the total amount of money received for this product, the number of sales for the product, the last time when such product has been sold and, finally, the product name. [V\_EVENMABGR\_2].

From this report, do you deduce that all the cheap products worth to be in the catalogue?

select prod\_name, sum(amount\_sold) quantity, count(\*) number\_sales, max(time\_id),

sum(amount\_sold-unit\_cost) benefit

from sales natural join costs join products using (prod\_id)

where prod\_list\_price<8

group by prod\_name;

3.A receipt is the information regarding those products that a customer has brought at one particular day. It includes the full name of the customer, its ID and address together with the total amount sold and the date. Besides that, we include the tax information. In Spain the IVA is the 21% of the amount we have paid. [V\_EVENMABGR\_3] .

select cust\_first\_name,cust\_last\_name,cust\_id,cust\_street\_address,cust\_postal\_code,cust\_city,

sum(amount\_sold) paided,time\_id,

sum(amount\_sold)\*0.21 taxes

from customers join sales using (cust\_id)

group by cust\_first\_name, cust\_last\_name,cust\_id,cust\_street\_address,cust\_postal\_code,cust\_city,time\_id;

4.We would like to analyze the behavior of the sales by regions. Thus, for each subcontinent, we will show its name, the number of countries, the number of customers, the average of the sales in such region, the total amount sold and the range of sales (the interval where all amounts are included in this region). We do not take into account the customers who were born before 1950. [V\_EVENMABGR\_4]

select country\_subregion, count(distinct country\_name) countries, count(distinct cust\_id) customers,

avg(amount\_sold) average\_sales, sum(amount\_sold), min(amount\_sold) min, max(amount\_sold) max

from sales join customers using (cust\_id) join countries using (country\_id)

where to\_number(to\_char(time\_id,'YYYY'))>=1950

group by country\_subregion;

5.Show the first and the last day where we received money by using channel number 3. Indicate also the number of sales and the average of all these sales. [V\_EVENMABGR\_5]

select min(time\_id),max(time\_id), count(\*), avg(amount\_sold)

from sales where channel\_id=3;

**13ºRel (DML Having.)**

1.For each promotion (show the promotion ID), list the number of sales made. In this list, only the significant promotions, i.e. those which have at least 10.000 sales. V\_HAVING\_1

select promo\_id, count(\*) num\_sales

from promotions join sales using (promo\_id)

group by promo\_id

having count(\*) >=10000;

2.For each product (show the product ID), list the average of its amount in the sales and the total units sold. Extract only the products having less than 1200 sales. V\_HAVING\_2

select prod\_id, avg(amount\_sold), sum(quantity\_sold)

from sales join products using (prod\_id)

group by prod\_id

having sum(quantity\_sold)<1200;

3.For each country, show its ID, its name, the number of its customer being a female and the average of its credit limit. Show only the significant countries, i.e. the ones having at least 300 women. V\_HAVING\_3

select country\_id, country\_name, count(distinct cust\_id), avg(cust\_credit\_limit)

from customers join countries using (country\_id)

where cust\_gender='F'

group by country\_id,country\_name

having count(distinct cust\_id)>=300;

4.For each product, show its ID and name, the average of amounts in the sales but only for those products sold in four different channels. V\_HAVING\_4

select prod\_id, prod\_name, avg(amount\_sold)

from products join sales using (prod\_id) join channels using (channel\_id)

group by prod\_id, prod\_name

having count(distinct channel\_id)=4;

5.For each products, list its name together with the total amount of money earned with it (amount sold minus its cost). Pick out only those products that have been brought by 5000 different customers (at least) and whose global amount in the sales is greater than 300.000 V\_HAVING\_5

select prod\_name, sum(amount\_sold-unit\_cost)

from products join sales using (prod\_id) join costs using (promo\_id,prod\_id,channel\_id,time\_id)

group by prod\_name

having count(distinct cust\_id)>=5000 and sum(amount\_sold)>300000;

6.For each country and gender, show the country ID and the number of customers in such groups. List the information only for those groups where all its customers was born on 1925 or later. V\_HAVING\_6

select country\_id, cust\_gender, count(\*)

from customers

group by country\_id, cust\_gender

having min(cust\_year\_of\_birth)>=1925;

7.List the full name of the persons who spend more than 16.000€ in a daily sale. A daily sale is consists of all purchases belonging to the same customer in the same day. V\_HAVING\_7

select time\_id,cust\_id, sum(amount\_sold)

from customers join sales using (cust\_id) join times using (time\_id)

group by cust\_id,time\_id

having sum(amount\_sold)>16000;

**14ºRel (DML nested subqueries)**

1.List the customers who had spent the most amount of money in a single sale. Do not show repeated rows. . SH.V\_NEST\_1

select distinct cust\_id, cust\_first\_name,cust\_last\_name

from customers join sales using (cust\_id)

where amount\_sold=(select max(amount\_sold) from sales);

2.List the same information of customers who had spent in a single sale more amount than the average of single sales in all the history. Do not show repeated rows. . SH.V\_NEST\_2

select distinct cust\_id, cust\_first\_name,cust\_last\_name

from customers join sales using (cust\_id)

where amount\_sold>(select avg(amount\_sold) from sales);

3.List the same info for those customers who had spent in a single sale the maximum amount of money of all sales made in the same day on which he brought such item. Thus, the customer is the best customer in some specific day, since he spent the maximum on one day. Do not show repeated rows. . SH.V\_NEST\_3

select distinct cust\_id, cust\_first\_name,cust\_last\_name

from customers join sales using (cust\_id) join

(select time\_id, max(amount\_sold) max\_amount from sales group by time\_id) using (time\_id)

where amount\_sold=max\_amount;

4.List a similar query to the first one but, instead of "in a single sale", we look for the amount in all the sales that the customer did. . SH.V\_NEST\_4

select distinct cust\_id, cust\_first\_name,cust\_last\_name

from customers join sales using (cust\_id)

group by cust\_id,cust\_first\_name,cust\_last\_name

having sum(amount\_sold)=

(select max(sum(amount\_sold)) from sales group by cust\_id);

5.List the PROD\_ID of the products that have a better chance of making a sale to the men. Hint: these products seem to be those ones that have been sold in a smaller number to the men (do not consider the equality). . SH.V\_NEST\_5

select prod\_id from

(select prod\_id,sum(quantity\_sold) fem\_sales

from products join sales using (prod\_id) join customers using (cust\_id)

where cust\_gender='F'

group by prod\_id,cust\_gender)

join

(select prod\_id,sum(quantity\_sold) masc\_sales

from products join sales using (prod\_id) join customers using (cust\_id)

where cust\_gender='M'

group by prod\_id,cust\_gender)

using (prod\_id)

where masc\_sales<fem\_sales

6.List the ID and the list price of the top-ten products which have a highest list price. To get the position of a row in a given table you can use the function ROWNUM. If you put this function is a SELECT the rows are labelled with its position. But observe that if you ORDER the result by the ROWNUM values, it seems to be unordered. The top-ten cannot be listed in a flat query, but you can first order the result by using the PRICE and use this table as a nested query in a from and then, in the outer query, you can only show the first ten rows by filtering the ROWNUM <= 10 in the second (outer) query . SH.V\_NEST\_6

select prod\_id, prod\_list\_price

from

(select prod\_id,prod\_list\_price

from products order by prod\_list\_price desc)

where rownum<=10;

7.Read again the exercise 5 of the "Even More About Groups" list of exercise. Can you include in this query the information of the day of the week corresponding to these first and last day? [SH.V\_NEST\_7]

select number\_sales,average\_sales, to\_char(first\_day,'DAY') first\_day\_name,

first\_day,to\_char(last\_day,'DAY') last\_day\_name, last\_day

**15ºRel (Views)**

1.Create a new table named TOUR with the same schema and content of the granted table DOCENCIA.TOUR2017. Use the order CREATE TABLE TOUR AS SELECT \* FROM DOCENCIA.TOUR2017.

2.Now, divide the table into INFO\_PERSONAL, containing the cyclist attributes and INFO\_PROFESIONAL with the team attributes for each cyclist. The attribute ID will be in both tables and serves as a linker between them. Define a PK on ID in both tables.

3.Create a view TOUR\_VIEW so that older applications can access the information as it was originally conceived. Thus, the queries SELECT \* FROM TOUR\_VIEW and SELECT \* FROM TOUR will produce the same result. This exercise will allows to fulfill the data independence property.

4.Create two views: one named TOUR\_SPAIN showing the Spanish cyclists and another, TOUR\_ITALY, with the Italian ones.

5.Insert a new Spanish cyclist throughout the view TOUR\_SPAIN.

6.If the new cyclist is not from Spain, can you also insert his information by using the view TOUR\_SPAIN. Check this situation. Can you see the new row by querying the TOUR\_SPAIN? But, it is in the TOUR table?

7.Redefine the view including the CHECK OPTION to fix this wire situation.

8.Redefine the view TOUR\_SPAIN to be a READ ONLY. Check if you can now INSERT a new row.

9.Create a view named V\_TOUR\_FRANCE showing the French cyclists. Grant the DELETE option on this view V\_TOUR\_FRANCE to the user ALUMNOBD1 (whose password is bd).

10.Create another connection for this user and try to delete a French cyclist from your schema. Can you delete a non-French cyclist? How can you fix this situation?

11.Since the ALUMNOBD1 can be used for any student in this class, remaining the DELETING privilege does not seems to be a good idea. Revoke this grant from this user.

12.Create a view named CYCLIST\_PER\_TEAM where the name of each team is listed together with the number of cyclists in the team. Can you make a successful insertion by using this view? Why not?

EXAMENES DE WUOLAH

**2018D**

4.Obtener la nota media de todos los alumnos que no nacieron en lunes ni martes, redondeando a 2 decimales. Para calcular la nota media use DECODE y considere AP=1, NT=2, SB=3 y MH=4. No sume los suspensos ni los no presentados

select NOMBRE, APELLIDO1, APELLIDO2, ROUND (avg (decode (calificacion, 'AP',1,'NT',2,'SB',3,'MH',4)),2) NOTA from matricular JOIN ALUMNOS ON ALUMNO=DNI WHERE CALIFICACION IN ('AP','NT','SB','MH') and to\_char(fecha\_nacimiento,'day') not like 'lunes%' and to\_char(fecha\_nacimiento,'day') not like 'martes%' group by NOMBRE, APELLIDO1, APELLIDO2 ;

**2018F**

Pregunta 4: Nombre completo de los profesores que han impartido la asignatura que más alumnos ha tenido matriculados, pero teniendo en cuenta solamente aquellas asignaturas con menos de 25 alumnos matriculados. Si un alumno se ha matriculado varias veces en la asignatura debe contarse sólo una vez.

Select distinct p.nombre, p.apellido1, p.apellido2 from profesores p join impartir i on i.profesor = p.id where exists (select \* from( select distinct asignatura,count(alumno) from matricular group by asignatura having count(alumno) < 25 order by count(alumno) desc) where rownum = 1);

**2019F**

1.Obtenga el nombre y 2 apellidos de los alumnos que no sea de Badajoz junto con su edad (en años sin decimales). Muestre también el nombre de la provincia

Select a.nombre,apellido1,apellido2,

trunc(months\_between(sysdate,fecha\_nacimiento)/12), p.nombre

From alumnos a join provincia p on a.cpro=p.codigo

Where p.nombre!=’Badajoz’

2.Obtenga apellidos de alumnus (puede ser el primero o el segundo) que contengan una ‘Z’. Sustituyala por ‘S’. Utilice la función REPLACE. Pase todo a mayúsculas para hacer la búsqueda y para mostrar el resultado.

Select replace(upper(apellido1),’Z’,’S’) from alumnus where upper(apllido1) like ‘%Z%’

Union

Select replace(upper(apellido2),’Z’,’S’) from alumnus where upper(apllido2) like ‘%Z%’

3.Muestre el nombre y 2 apellidos de los alumnos a los que no les imparta clase ningún profesor que ingresara en los 9 primeros meses del año (el mes de antigüedad está entre 1 y 9). Tenga en cuenta asignatura, curso y grupo en las tablas impartir y matricular.

Select a.nombre,apellido1,aplleido2

From alumnos a

Where dni not in

(select alumno from matriuclar m natural join impartir i join profesores p on i.profesor=p.id where to\_char(antiguedad,’mm’) between 1 and 9)

4.Muestre por cada professor el total de la carga de créditos del curso 15/16. En el listado deben aparecer todos los profesores. Si no existe información de ese año para el curso 15/16 muestre 15/16 en Curso y 0 en la carga de créditos.

Select nvl(i.curso,’15/16’), p.nombre, p.apellido1, p.apellido2, sum(nvl(carga\_creditos,0))

From profesores p left join impartir i on i.profesor=p.id

Where i.curso=’15/16’ or i.curso is null

Group by i.curso, p.id, p.nombre, p.apellido1, p.apellido2

**2020E**

3.Hemos detectado que hay alumnos que están evaluados en una asignatura en un curso anterior a su fecha de primera matrícula. Saque un listado con el código de la asignatura, el nombre y dos apellidos del alumno, el curso de la matrícula y la fecha de primera matríucla del alumno. Use SUBSTR con MATRICULAR.CURSO para obtener el año en que cursó la asignatura. OJO: No hay que usar la tabla asignaturas, solo matricular y alumnos

Select m.asignatura, a.nombre, a.apellido1,a.apellido2, m.curso, a.fecha\_prim\_matricula

From matricular m join alumnos a on (alumno=dni)

Where a.fecha\_prim\_matricula>to\_date(’01-01-20’||SUBSTR(CURSO,1,2),’DD-MM-YYYY’)

4.Queremos saber cuantos créditos imparte cada profesor con respecto al total de la asignatura. Para ello sume los créditos que tiene cada asignatura que imparte (campo créditos de ASIGNATURAS) y muestre también la suma de la carga de créditos de la tabla IMPARTIR. Deben salir todos los profesores aunque no impartan asignaturas y sustituya los NULL por 0.

Select a.nombre, a.apellido, a.apellido2, sum(nvl(créditos,0)),sum(nvl(carga\_creditos,0))

From profesores a left outer join impartir i on i.profesor=a.id

Left outer join asignaturas asi on i.asignatura=asi.codigo

Group by a.nombre,a.apellido1,a.apellido2

**2020F**

2. Mostrar por cada asignatura y provincia, cuántos alumnos distintos hay matriculados en el curso 15/16 (atributo curso de matricula). Pero muestre solo aquellas filas en las que haya más de 3 alumnos matriculados

select asi.nombre, pp.nombre, count (distinct alumno) from asignaturas asi join matricular m on m.asignatura = asi.codigo join alumnos a on m.alumno = a.dni join provincia pp on a.cpro = pp.codigo where m.curso='15/16' group by asi.nombre, pp.nombre having count (distinct alumno) >3

3. Mostrar el listado de municipios cuyos nombres se repiten. En el listado deberá aparecer únicamente una columna: el nombre del municipio. Un mismo nombre de municipio no debería aparecer más de una vez en este listado. Muestre el nombre del municipio en mayúsculas y elimine aquellos cuyo nombre contenga una coma.

SELECT upper(NOMBRE) FROM MUNICIPIO where nombre not like '%,%' GROUP BY upper(NOMBRE) HAVING COUNT(\*) >1

4. Obtener por cada provincia, el municipio con más habitantes, pero muestre sólo aquellos en los cuales el nombre de la provincia y el del municipio no coincidan

SELECT p.nombre, M.NOMBRE, M.HOMBRES + M.MUJERES FROM MUNICIPIO M join provincia p on m.cpro = p.codigo WHERE (M.HOMBRES + M.MUJERES) = (SELECT MAX(HOMBRES + MUJERES) FROM MUNICIPIO m1 WHERE m1.CPRO = M.CPRO) and p.nombre <> m.nombre

5. Obtenga el ID, el nombre y el número de asignaturas impartidas para todos los profesores adscritos al departamento de ‘Arquitectura de Computadores’. Deben aparecer todos los profesores aunque no impartan ninguna asignatura

select p.id, p.nombre, count(i.asignatura) from profesores p left join impartir i on (p.id=i.profesor) left join departamentos d on (d.codigo=p.departamento) where d.nombre = 'Arquitectura de Computadores' group by p.id, p.nombre

6. Obtener alumnos con una nota media mayor que la nota media de aquellos alumnos que tienen al menos tres asignaturas aprobadas (Para calcular la nota media use DECODE y considere AP=1, NT=2, SB=3 y MH=4. No sume los suspensos ni los no presentados. Redondee la nota a 2 decimales)

SELECT NOMBRE, APELLIDO1, APELLIDO2, ROUND (AVG (DECODE (CALIFICACION, 'AP',1,'NT',2,'SB',3,'MH',4)),2) NOTA FROM MATRICULAR JOIN ALUMNOS ON ALUMNO=DNI WHERE CALIFICACION IN ('AP','NT','SB','MH') GROUP BY NOMBRE, APELLIDO1, APELLIDO2 HAVING AVG(DECODE (CALIFICACION, 'AP',1,'NT',2,'SB',3,'MH',4)) > (SELECT AVG (ROUND (AVG (DECODE (CALIFICACION, 'AP',1,'NT',2,'SB',3,'MH',4)),2) ) NOTA\_MEDIA FROM MATRICULAR WHERE CALIFICACION IN ('AP','NT','SB','MH') GROUP BY alumno HAVING COUNT (\*) >=3)

**2021E**

1.Realizar una consulta de los alumnos que muestre su año de nacimiento. La consulta devolverá una única columna por cada alumno con el texto “nombre apellido1 nacido en año”. Así si Juan García hubiese nacido en 1995 debería indicar “Juan García nacido en 1995”. No evite repeticiones

Select nombre ||’ ‘ || apellido1 || ‘ nacido en ‘ || to\_char(fecha\_nacimiento,’yyyy’) from alumnos;

3.Realizar una consulta que devuelva el dni, nombre, apellido1 y apellido2 de aquellos alumnos que nunca han suspendido ninguna asignatura. Se deberán evitar repeticiones. Únicamente las matriulaciones que tienen explícitamente un suspenso serán consideradas como tales. Las matriculaciones sin calificación o calidicados como no presentados no serñan considerados como suspensos. Igualmente, los alumnos no matriculados en ninguna asignatura deberán aparecer en el listado, al no haber suspendido ninguna asignatura.

Select a.dni, a.nombre, a.apellido1, a.apellido2

From alumnos a

Minus

Select a.dni, a.nombre, a.apellido1, a.apellido2

From alumnos a join matricular m on (a.dni=m.alumno)

Where m.calificacion=’SP’;

5.Indicar el dni, nombre, apellido1 y apellido2 del alumno que más veces ha suspendido. Si varios alumnos empatan en la primera posición de alumnos con más suspensos deberán salir todos ellos.

Select dni, nombre, apellido1, apellido2

From alumnos join matricular on (alumno=dni)

Where calificacion=’SP’

Group by dni, nombre, apellido1, apellido2

Having count(\*)=(select max(count(\*)) from matricular where calificacion=’SP’ group by alumno);

6.Obtener un listado con parejas de alumnos distintos que han compartido clase en 5 o más grupos (deben coincidir en asignatura, curso y grupo). Se mostratán únicamente los dni de los alumnos y se deberán evitar repeticiones (si el alumno 1 aparece emparejado con el alumno 2 en el listado, entonces no debería aparecer una fila adicional indicando que el alumno 2 se empareja con el 1).

Select a1.dni, a2.dni

From alumnos a1 join matricular m1 on (a1.dni=m1.alumno)

Join matricular m2 on (m1.asignatura=m2.asignatura and m1.curso=m2.curso and m1.grupo=m2.grupo)

Join alumnus a2 on (m2.alumno=a2.dni)

Where a1.dni<a2.dni

Group by a1.dni,a2.dni

Having count(\*)>=5;

2021E2

6.Queremos saber (solo para el curso 18/19) qué alumnos obtuvieron una nota media mejor que el promedio de las notas medias. Para ello obtén la media de cada alumno (para el curso 18/19) y halla la media de esos valores. La nota media de un alumno se calcula utilizando solo las aprobadas y aplicando el valor 4 a MH, 3 a SB, 2 a NT y 1 a AP. El resto de valores (null o SP) no se computan. Muestra también la nota media de esos alumnos redondeando a 2 decimales y el número de asignaturas aprobadas.

Select nombre, apellido1,apellido2,round(avg(decode(calificación,’MH’,4,’SB’,3,’NT’,2,’AP’,1)),2), count(\*)

From matricular join alumnos on alumno=dni

Where calificacion in (‘MH’,’SB’,’NT’,’AP’) and curso=’18/19’

Group by alumno, nombre, apellido1, apellido2

Having avg(decode(calificación,’MH’,4,’SB’,3,’NT’,2,’AP’,1))>

(select avg(avg(decode(calificaion,’MH’,4,’SB’,3,’NT’,2,’AP’,1)))

From matricular

Where calificacion in (‘MH’,’SB’,’NT’,’AP’) and curso = ‘18/19’

Group by alumno)