**Câu 1: Giải thích các hàm thông dụng sau**

|  |  |  |
| --- | --- | --- |
| Stt | Tên Hàm | Mục đích sử dụng & nên sử dụng khi nào |
| 1 | COUNT() | Trả về số lượng hàng được trả về bởi truy vấn. Có thể sử dụng nó như một chức năng tổng hợp hoặc phân tích |
| 2 | SUM() | Trả về tổng số lượng của 1 thuộc tính, thường sử dụng count() chung với group by |
| 3 | MAX() | Trả về dữ liệu lớn nhất của một thuộc tính, thường sử dụng chung với group by |
| 4 | MIN() | Trả về dữ liệu nhỏ nhất của 1 thuộc tính, thường sử dụng với group by |
| 5 | NVL() | Tránh giá trị null của 1 thuộc tính và nếu null thì sẽ gán bằng 1 giá trị truyền vào |
| 6 | TO\_CHAR() | Đổi 1 giá trị sang kiểu chuỗi |
| 7 | TO\_DATE() | Đổi 1 giá trị sang kiểu date |
| 8 | TO\_NUMBER() | Đổi 1 giá trị sang kiểu number |
| 9 | SUBSTR() | Dùng để lấy 1 chuỗi con từ 1 chuỗi cha |
| 10 | REPLACE() | Dùng để thay thế một chuỗi các ký tự trong một chuỗi với một bộ ký tự khác. |
| 11 | REVERSE() | Dùng để đảo chuỗi |
| 12 | DECODE() | Dùng như if-then-else trong Oracle |
| 13 | TRUNC() | Dùng để thu gọn một số đến 1 vị trí nào đó sau dấu phẩy, vị trị sẽ được chỉ định bởi tham sô truyền vào |
| 14 | LENGTH() | Dùng để trả về tổng số ký tự có trong chuỗi, kể cả khoảng trắng |
| 15 | lPAD() | Đệm vào bên trái của chuỗi với tập ký tự được xác định trong tham số của nó |
| 16 | RPAD() | Đệm vào bên phải của chuỗi với tập ký tự được xác định trong tham số của nó |
| 17 | TRIM() | Dùng để loại bỏ khoảng trắng đầu và cuối của chuỗi, cũng có thể xóa ký tự chỉ định ở đầu và cuối của chuỗi |
| 18 | LTRIM() | Dùng để loại bỏ khoảng trắng ở bên trái của chuỗi, cũng có thể xóa ký tự chỉ định ở bên trái của chuỗi |
| 19 | RTRIM() | Dùng để loại bỏ khoảng trắng ở bên phải của chuỗi, cũng có thể xóa ký tự chỉ định ở bên phải của chuỗi |
| 20 | ROUND() | Hàm trả về một số được làm tròn cho một số vị trí thập phân nhất định |
| 21 | ADD\_MONTHS() | Có tác dụng là thêm hoặc bớt đi một hoặc nhiều tháng dựa vào tham số truyền vào |

**Câu 2:**

**SELECT \***

**FROM MDM\_CUSTOMER A**

**WHERE A.CUST\_LGL\_ENG\_NM LIKE ‘%\\_LOGISTICS%’ ESCAPE ‘\’**

**ORDER BY A.CUST\_LOCL\_LANG\_NM NULLS FIRST**

**A) Vui lòng giải thích ý nghĩa của câu SQL trên**

**B) Ý nghĩa của việc dùng ESCAPSE**

**C) Ý nghĩa của việc dung Nulls First.**

**D) Ý nghĩa của việc dung alias, có nên dung alias trong mọi trường hợp không?**

**Câu A:**

* Trả về tất cả các cột của bảng MDM\_CUSTOMER với điều kiện là giá trị trong cột CUST\_LGL\_ENG\_NM phải có chứa “\_LOGISTICS” và được sắp xếp theo cột CUST\_LOCL\_LANG\_NM với giá trị null ở đầu.

**Câu B:**

* Sử dụng ESCAPE giúp mình chuỗi trong LIKE có thể chứa các ký tự đặc biệt.

**Câu C:**

* Sử dụng Nulls First có nghĩa là các giá trị null sẽ được sắp xếp lên đầu.

**Câu D:**

* Việc sử dụng ALIAS dùng để đổi tên 1 bảng, có thể giúp ngắn hơn so với tên bảng đỡ rờm rà. Và cũng giúp cho DEV đọc vào có thể dễ hiểu hơn. Chúng ta không nhất thiết phải sử dụng ALIAS mọi lúc như trong trường hợp có 1 bảng

**Câu 3**:

**SELECT \***

**FROM MDM\_CUSTOMER**

**WHERE 1 = 1**

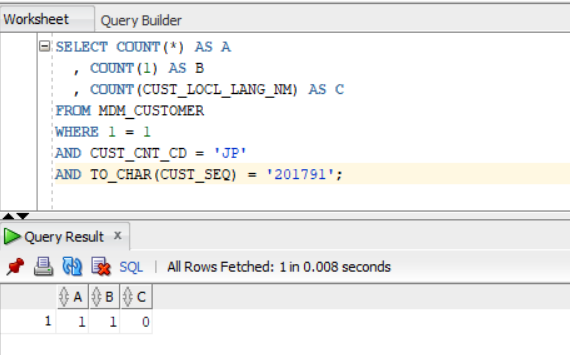
**AND CUST\_CNT\_CD = 'JP'**

**AND TO\_CHAR(CUST\_SEQ) = '201791'**

**Theo bạn câu trên cách dung TO\_CHAR(CUST\_SEQ) = '201791' có hợp lý không, tại sao?**

* Em nghĩ là không hợp lý. Sử dụng TO\_CHAR() trong trường hợp này dẫn đến tăng Cost và làm mất index của nó.

**Câu 4**: **Cho câu SQL và kết quả như hình bên dưới**



**A) Giải thích ý nghĩa COUNT(\*), COUNT(1), COUNT(CUST\_LOCL\_LANG\_NM)**

**B) Tại sao COUNT(CUST\_LOCL\_LANG\_NM) lại bằng 0**

**Câu A:**

* COUNT(\*) , COUNT(1) : trả về số lượng row trong table

COUNT(CUST\_LOCL\_LANG\_NM) : trả về số lượng row có các giá trị duy nhất của CUST\_LOCL\_LANG\_NM.

**Câu B:**

* Tại vì tất cả các row trong cột CUST\_LOCL\_LANG\_NM đều mang giá trị là null.

**Câu 5: Có 2 cách như bên dưới, cách nào tốt tại sao?**

|  |  |  |
| --- | --- | --- |
|  | **Cách 1** | **Cách 2** |
|  | NVL(SUM(COL1),0) | SUM(NVL(COL1,0)) |

* Cách 1 tốt hơn vì call NVL một lần, sau khi tổng số được tính toán, bất kể có bao nhiêu hàng được tìm thấy. Nó có khả năng hiệu quả hơn.
* Cách 2 mỗi 1 dòng thì call NVL 1 lần rất lãng phí.

**Câu 6**: **Có 2 cách như bên dưới, cách nào tốt tại sao?**

|  |  |  |
| --- | --- | --- |
|  | **Cách 1** | **Cách 2** |
|  | NVL(SUM(COL1),0) + NVL(SUM(COL2),0) | Ex.1] SUM(NVL(COL1 + COL2,0))  Ex.2] NVL(SUM(COL1 + COL2),0) |

* Cách 1 tốt hơn. Vì khi ta sử dụng cách 2 chỉ cần 1 trong 2 COL bằng null thì giá trị sẽ trả ra null bất kể COL còn lại có giá trị. Dẫn đến sai lệch giá trị.

**Câu 7**: **Có 2 cách như bên dưới, cách nào tốt tại sao?**

|  |  |  |
| --- | --- | --- |
|  | **Cách 1** | **Cách 2** |
|  | SELECT A.CUST\_NO, A.ORD\_NO, A.PRO\_CD, B.PROD\_NM  FROM TB\_ORD A,    TB\_PROD B  WHERE 1 = 1    AND A.PRO\_CD = B.PROD\_CD    AND B.PROD\_CD IN (SELECT PROD\_CD FROM TB\_PROD D WHERE D.PROD\_CD = A.PRO\_CD AND PROD\_UNIT\_AMT < 800); | SELECT A.CUST\_NO, A.ORD\_NO, A.PRO\_CD, B.PROD\_NM  FROM TB\_ORD A,    TB\_PROD B  WHERE 1 = 1    AND A.PRO\_CD = B.PROD\_CD    AND EXISTS (SELECT D.PROD\_CD FROM TB\_PROD D WHERE D.PROD\_CD = A.PRO\_CD AND D.PROD\_UNIT\_AMT < 800); |

* **IN** đc dung khi nào và EXISTS đc dung khi nào.
* Theo em cách 1 có vẻ tốt hơn. Vì lúc này truy vấn con có số bộ ít nên việc dùng EXISTS sẽ khá là phí. Và em có dựa thêm vào COST của Oracle để đưa đến quyết định.
* Câu truy vấn con quét ở một quan hệ có số bộ từ ít đến trung bình thì  
  nên dùng IN VÀ NOT IN.
* Ngược lại nếu câu truy vấn con đòi hỏi phải quét nhiều quan hệ có số  
  bộ lớn thì EXISTS và NOT EXITST lại tốt hơn.

**Câu 8: Có 2 cách như bên dưới, cách nào tốt tại sao?**

|  |  |  |
| --- | --- | --- |
|  | **Cách 1** | **Cách 2** |
|  | SELECT A.CUST\_NO, A.ORD\_NO, A.PRO\_CD, B.PROD\_NM  FROM TB\_ORD A,    TB\_PROD B  WHERE 1 = 1    AND A.PRO\_CD = B.PROD\_CD    AND B.PROD\_CD IN ('00001','00002'); | SELECT  A.CUST\_NO, A.ORD\_NO, A.PRO\_CD    , (SELECT B.PROD\_NM FROM TB\_PROD B WHERE B.PROD\_CD = A.PRO\_CD) AS PROD\_NM  FROM TB\_ORD A  WHERE 1 = 1    AND A.PRO\_CD IN ('00001','00002'); |

* Cách 1 tốt hơn vì xử lý data với số lượng ít thì dùng JOIN sẽ đỡ phí hơn là sử dụng SUBQUERY

**Câu 9**: **Cho số 8988.80 vui lòng xuất ra định dạng $8,988.800**

* SELECT TO\_CHAR(8988.80,’$9,999.999’) FROM DUAL;

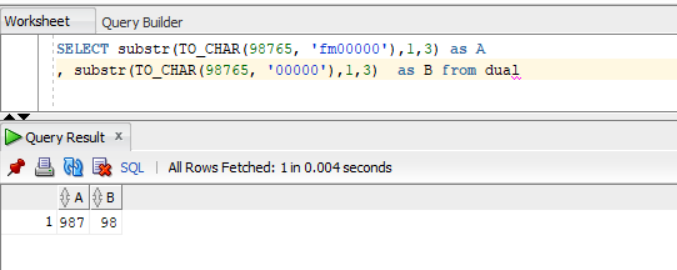
**Câu 10: Cho số 8988.80, 820988.80 vui lòng xuất ra định dạng $8,000.000, $820,000.000**

* SELECT TO\_CHAR(TRUNC(8988.80,-3), '$9,999.999'),

TO\_CHAR(TRUNC(820988.80,-3),'$999,999.999')

FROM DUAL;

**Câu 11: Cho cấu SQL và kết quả như sau:**



**Như hình trên cả 2 A và B điêu substr từ 1, đến 3 tại sao kết quả lại khác nhau.**

**Câu 12: Viết Câu SQL xuất ra, Ngày hiện tại, này hôm qua, ngày mai?**

* **SELECT** SYSDATE AS TODAY,

SYSDATE - 1 AS YESTERDAY,

SYSDATE +1 AS TOMORROW

**FROM** DUAL;

* With SYSDATE là lấy ngày hiện tại.

**Câu 13**: **Ta có table (TB\_ORD), yêu cầu viết câu SQL để generate ORD\_NO có đô dài 10 tự với format sau: yyyymmdd000Seq, ví dụ hnay là 20191028 và chưa có seq nào thì ORD\_NO sẽ là 201910280001, và nếu đã tồn tại ORD\_NO 201910280001 thì nó sẽ là 201910280002**

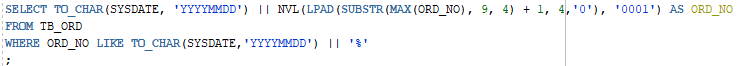
**SELECT** TO\_CHAR(SYSDATE, 'YYYYMMDD') || NVL(LPAD(SUBSTR(MAX(ORD\_NO), 9, 4) + 1, 4,'0'), '0001') AS ORD\_NO

**FROM** TB\_ORD

**WHERE** ORD\_NO LIKE TO\_CHAR(SYSDATE,'YYYYMMDD') || '%'

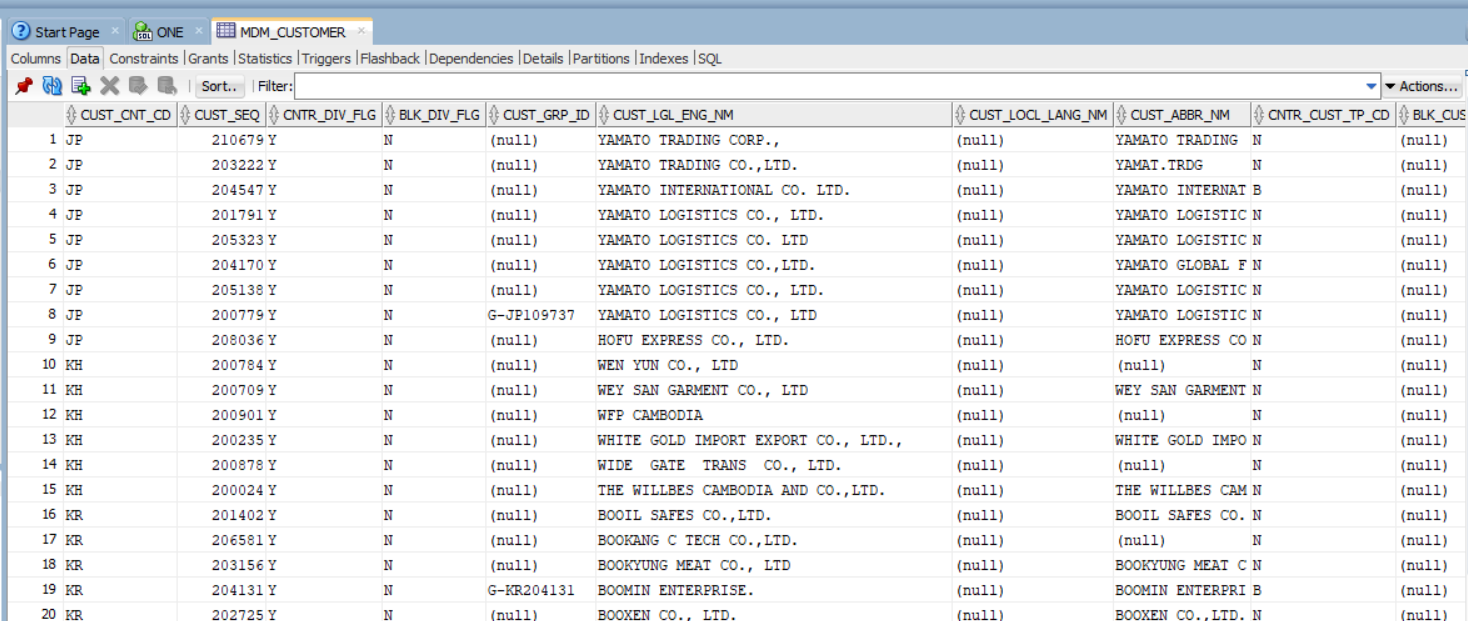
;

----------------------------------------------------------------------------------------------------------------



----------------------------------------------------------------------------------------------------------------

**Câu 14**: **Ta có table (MDM\_CUSTOMER) và dữ liệu như bên dưới**



**Các field liên quan: CUST\_CNT\_CD, CUST\_SEQ, CUST\_GRP\_HRCHY\_CD, CUST\_GRP\_ID**

**Dữ liệu cột CUST\_GRP\_HRCHY\_CD có thể có(I: Individual, C: Country, G: Global)**

**A) Viết câu SQL tìm CUST\_GRP\_ID sao cho: CUST\_GRP\_HRCHY\_CD có I hoặc C nhưng không có G**

**B) Viết câu SQL tìm CUST\_GRP\_ID sao cho: CUST\_GRP\_HRCHY\_CD có G và có I nhưng không có C**

Câu A:

**SELECT** CUST\_GRP\_ID,CUST\_GRP\_HRCHY\_CD

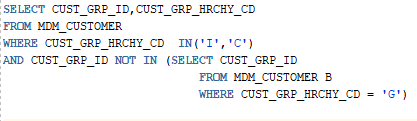
**FROM** MDM\_CUSTOMER

**WHERE** CUST\_GRP\_HRCHY\_CD IN('I','C')

**AND** CUST\_GRP\_ID NOT IN (**SELECT** CUST\_GRP\_ID

**FROM** MDM\_CUSTOMER B

**WHERE** CUST\_GRP\_HRCHY\_CD = 'G')

----------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------

**Câu B:**

**SELECT** **DISTINCT** CUST\_GRP\_ID,CUST\_GRP\_HRCHY\_CD

**FROM** MDM\_CUSTOMER A

**WHERE** CUST\_GRP\_ID NOT IN (**SELECT** CUST\_GRP\_ID

**FROM** MDM\_CUSTOMER

**WHERE** CUST\_GRP\_HRCHY\_CD = 'C')

**AND** CUST\_GRP\_ID IN (**SELECT** CUST\_GRP\_ID

**FROM** MDM\_CUSTOMER B

**WHERE** B.CUST\_GRP\_HRCHY\_CD = 'I'

**AND** A.CUST\_GRP\_ID = B.CUST\_GRP\_ID)

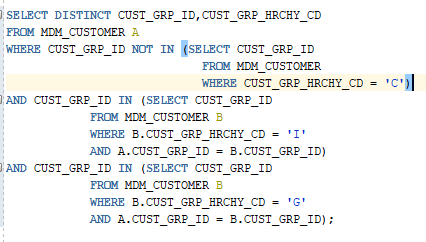
**AND** CUST\_GRP\_ID IN (**SELECT** CUST\_GRP\_ID

**FROM** MDM\_CUSTOMER B

**WHERE** B.CUST\_GRP\_HRCHY\_CD = 'G'

**AND** A.CUST\_GRP\_ID = B.CUST\_GRP\_ID);

----------------------------------------------------------------------------------------------------------------



----------------------------------------------------------------------------------------------------------------

**Câu 15**: Ta có table (**TB\_PROD**) và dữ liệu như bên dưới



Viets cấu SQL để suất ra kêt quả như sau:

1. Lấy max(PROD\_UNIT\_AMT)
2. Lấy giá trị min(PROD\_UNIT\_AMT)
3. Lấy giá trị trung bình PROD\_UNIT\_AMT
4. Lấy tên của sản phẩm có PROD\_UNIT\_AMT lớn nhất

Kết quả phải ra đc như sau:



SELECT MAX(PROD\_UNIT\_AMT) MAX\_AMT

, MIN(PROD\_NM) KEEP (DENSE\_RANK LAST ORDER BY PROD\_UNIT\_AMT) AS PROD\_NM

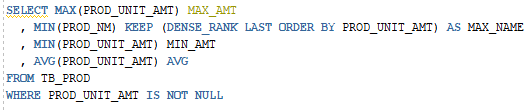
, MIN(PROD\_UNIT\_AMT) MIN\_AMT

, AVG(PROD\_UNIT\_AMT) AVG

FROM TB\_PROD

WHERE PROD\_UNIT\_AMT IS NOT NULL

----------------------------------------------------------------------------------------------------------------



* Mệnh đề KEEP (DENSE\_RANK LAST ORDER BY [column name] ) : cho phép ta giữ lại giá trị cuối cùng trong column