Databse test



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

|  |  |  |
| --- | --- | --- |
| Stt | Tên Hàm | Mục đích sử dụng & nên sử dụng khi nào |
| 1 | Count() | Đếm các dòng dữ liệu khác nhau của một thuộc tính (hoặc nguyên dòng) |
| 2 | Sum() | Tổng giá trị của 1 thuộc tính, thường sử dụng với Group By |
| 3 | MAX() | Tìm giá trị lớn nhất của 1 thuộc tính, thường sử dụng với group by |
| 4 | MIN() | Tìm giá trị nhỏ nhất của 1 thuộc tính, thường sử dụng với group by |
| 5 | NVL() | Thay thế giá trị null thành giá trị của param truyền vào |
| 6 | TO\_CHAR() | Chuyển một đối tượng bất kỳ sang chuỗi |
| 7 | T0\_DATE() | Chuyển chuỗi sang date |
| 8 | TO\_NUMBER() | Để chuyển chuỗi sang number |
| 9 | SUBSTR() | Lấy chuỗi con từ chuỗi cha |
| 10 | REPLACE() | Thay thế chuỗi con trong chuỗi ban đầu thành chuỗi tương ứng |
| 11 | REVERSE() | Nghịch đảo chuỗi |
| 12 | DECODE() | Thực hiện logic if-then-else trong câu query |
| 13 | TRUNC() | Thu gọn một số đến vị trí nào đó sau dấu phẩy, vị trí này sẽ được chỉ định bới tham số truyền vào. |
| 14 | LENGTH() | Trả về độ dài chuỗi |
| 15 | lPAD() | Đem vào bên trái của chuỗi với tập hợp các ký tự cụ thể được xác định trong tham số của nó. Nếu bạn nhập vào số nhỏ hơn chuỗi gốc thì nó sẽ cắt ngắn chuỗi gốc, ngược lại nó sẽ bổ sung ký tự được xác định vào chuỗi gốc. |
| 16 | RPAD() | Đem vào bên phải của chuỗi với tập hợp các ký tự cụ thể được xác định trong tham số của nó. Nếu bạn nhập vào số nhỏ hơn chuỗi gốc thì nó sẽ cắt ngắn chuỗi gốc, ngược lại nó sẽ bổ sung ký tự được xác định vào chuỗi gốc. |
| 17 | TRIM() | Xóa các khoảng trắng hai bên của chuỗi, ngoài ra còn dùng để xóa một ký tự chỉ định hai bên của chuỗi. |
| 18 | LTRIM() | Xóa các khoảng trắng bên trái của chuỗi, ngoài ra còn dùng để xóa các ký tự chỉ định phía bên trái. |
| 19 | RTRIM() | Xóa các khoảng trắng bên trái của chuỗi, ngoài ra còn dùng để xóa các ký tự chỉ định phía bên trái. |
| 20 | ROUND() | Làm tròn một số đến một số vị trí thập phân nhất định |
| 21 | ADD\_MONTHS() | Thêm hoặc bớt đi một hoặc nhiều tháng dựa vào tham số truyền vào, nếu bạn truyền số dương thì sẽ thêm tháng và số âm thì sẽ trừ tháng |

**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

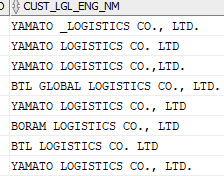
**A) Vui lòng giải thích ý nghĩa của câu SQL trên**

**Chọn tất cả các trường trong table MDM\_CUSTOMER có CUST\_LGL\_ENG\_NM chứa ‘\_LOGISTICS’**

**B) ý nghĩa của việc dùng ESCAPSE**

**Để sử dụng ký tự trong nó là một ký tự escape. Rồi sử dụng ký tự đó để nhận những ký tự đặc biệt trong chuỗi**

**Trong câu trên ta sử dụng dấu backslash để escape ký hiệu và ký tự đặc biệt “\_” , rồi dùng ESCAPE để nhận kí tự đó vào để tìm kiếm chuỗi cần tìm, nếu câu trên không dùng dấu backslash trước kí tự đặc biệt và bỏ luôn ESCAPE thì kết quả sẽ ra như sau:**



**Ta nhận thấy chuỗi được tìm kiếm sẽ gồm cả trường hợp có kí tự đặc biệt và không có trước chuỗi “LOGISTICS%”**

**C) Ý nghĩa của việc dùng NULLS FIRST.**

**Để đưa cái giá trị null của cột được sắp xếp lên trước các giá trị thông thường**

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

**Alias là sử dụng bí danh cho một đối tượng cụ thể nào đó. Ta có thể sử dụng alias cho column, table hay một tên bất kì (ngắn gọn hơn), và kí hiệu của nó trong SQL là AS. Lệnh này sẽ tạo ra một tên tạm và chỉ tồn tại trong quá trình chạy câu truy vấn, sau đó nó sẽ được giải phóng khỏi bộ nhớ.**

**Như trong phân tích thiết kế database thì trong một bảng tên các columns phải là duy nhất, nhưng ở 2 bảng khác nhau thì tên column có thể sẽ bị trùng tên nên việc này sẽ gây ra lỗi nếu bạn không chỉ định rõ column của bảng nào vì vậy cần sử dụng Alias trong trường hợp này.**

**Alias không cần thiết được sử dụng khi chỉ có 1 table hoặc khi không muốn đổi tên thuộc tính.**

**Câu 3:**

SELECT \*

FROM MDM\_CUSTOMER

WHERE 1 = 1

AND CUST\_CNT\_CD = 'JP'

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

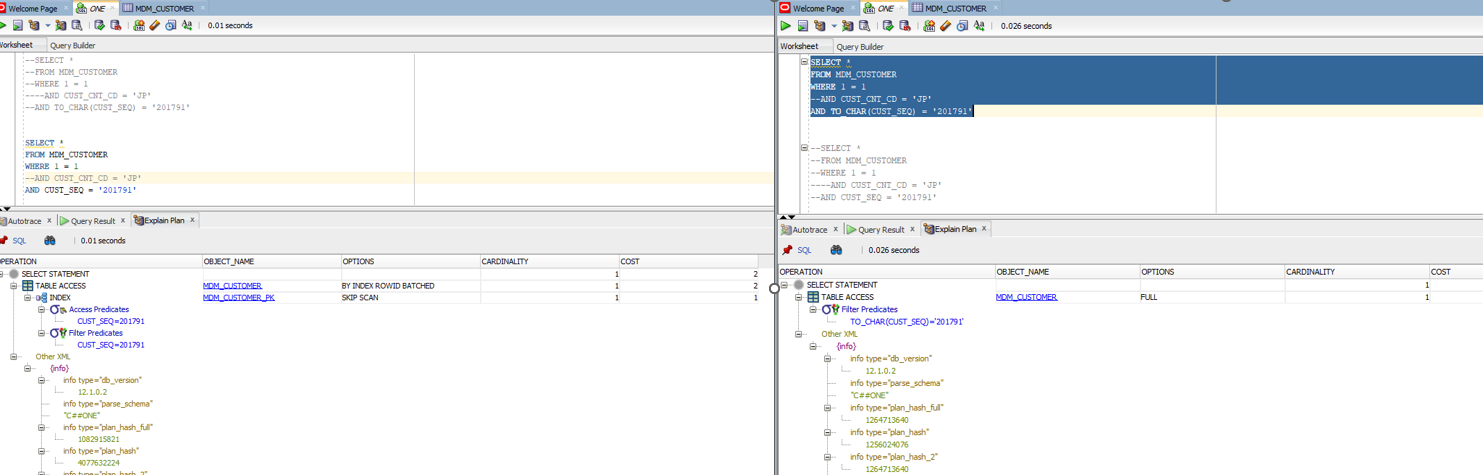
**Theo bạn câu trên cách dung TO\_CHAR(CUST\_SEQ) = '201791' có hợp lý không, tại sao?**

**TO\_CHAR sử dụng trong tường hợp này không hợp lý.**

**Lý do: vì nó bị mất index nên câu truy vấn sẽ phải chạy qua tất cả các Row của bảng** MDM\_CUSTOMER **để so sánh và lấy ra những Row thỏa mãn điều kiện CUST\_SEQ = '201791'**

* **Kết quả là Performance sẽ rất tệ khi data lớn, dù kết quả câu query ra đúng.**

**Khi so sánh 2 kết quả dùng TO\_CHAR và không dung TO\_CHAR ta có kết quả như sau:**



**=> Kết quả chạy câu truy vấn khi dùng TO\_CHAR thì có cost lớn hơn khi không dung TO\_CHAR.**

**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)

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

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

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

**Vì giá trị CUST\_LOCL\_LANG\_NM trong các dòng kết quả trả về chỉ chứa 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 về performance vì chỉ thực hiện nvl 1 lần trong khi cách 2 thực hiện hàm nvl n lần (với n là số dòng dữ liệu trong db).**

**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 trong cách 2 nếu 1 trong 2 cột có giá trị Null thì kết quả sẽ ra null dù bên còn lại có giá trị đi nữa điều này dẫn đến giá trị nhận được bị sai.**

**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?

* **Cách 2 tốt hơn vì nó có cost nhỏ hơn**
* **Từ khóa Exists đánh giá true hoặc false, nhưng từ khóa IN sẽ so sánh tất cả các giá trị trong truy vấn con tương ứng. Mặt khác, nếu chúng ta đang sử dụng EXISTS, SQL engine sẽ dừng quá trình quét ngay sau khi tìm thấy kết quả trùng khớp nên sẽ đỡ tốn performance hơn IN.**

**Refer to :**

[**http://www.dba-oracle.com/t\_exists\_clause\_vs\_in\_clause.htm**](http://www.dba-oracle.com/t_exists_clause_vs_in_clause.htm)

* **IN thường được dùng trong trường hợp chỉ có một vài giá trị trong một list hoặc các giá trị không thể lấy được từ các bảng.**
* **EXISTS thường được dùng trong trường hợp kiểm tra sự tồn tại của một thuộc tính trong bảng khác(khi subquery là 1 list lớn), ngoài ra nó còn thường được dùng trong trường hợp kiểm tra sự tồn tại của nhiều thuộc tính cái mà IN không thể làm được.**

**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 2 tốt hơn. Vì cách 1 full scan 2 bảng rồi mới thực hiện tìm kiếm PRO\_CD. Trong khi cách 2, full scan bảng A thực hiện tìm kiếm PRO\_CD rồi mới full scan bảng B với PRO\_CD của bảng A. Điều này sẽ giúp phép kết giữa bảng B và bảng A ít hơn nên dẫn đến câu truy vấn sẽ nhanh hơn.**

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 câu SQL và kết quả như sau:**



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

* **Với format ‘00000’ sẽ có kiểu string với độ dài là 6 (5 cho ký tự số 0-9 và 1 ký tự đầu cho dấu) tuy nhiên do đang là số dương nên ký tự đầu tiên là khoảng trắng => nên khi lấy substr từ 1->3 thì sẽ bắt đầu từ vị trí của khoảng trắng nên kết quả sẽ là 98**
* **Với format ‘fm00000’ sẽ có kiểu string với độ dài là 6 (5 cho ký tự số 0-9 và 1 ký tự đầu cho dấu) tuy nhiên do đang là số dương nên ký tự đầu tiên là khoảng trắng tương tự như format ’00000’. Không những vậy hàm này còn trim khoảng trống hai bên vì vậy trong trường hợp số dương số ký tự của chuỗi chỉ là 5 thay vì 6 như ban đầu => nên khi lấy substr từ 1->3 thì sẽ bắt đầu từ vị trí của số 987**

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

**SELECT**

**SYSDATE + INTERVAL '-1' DAY AS YESTERDAY,**

**SYSDATE as NOW,**

**SYSDATE + INTERVAL '1' DAY AS TOMORROW**

**FROM dual**

**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**

**CONCAT(TO\_CHAR(SYSDATE,'yyyyMMdd'), (**

**SELECT(TO\_CHAR(count(\*) + 1, 'fm0000'))**

**FROM TB\_ORD**

**WHERE ord\_dttm LIKE CONCAT(TO\_CHAR(SYSDATE,'yyyyMMdd'),'\_\_\_\_'))**

**)**

**FROM TB\_ORD**

**WHERE ROWNUM =1**

**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**

**SELECT CUST\_GRP\_ID**

**FROM MDM\_CUSTOMER A**

**WHERE CUST\_GRP\_ID IS NOT NULL**

**AND NOT EXISTS (SELECT CUST\_GRP\_ID FROM MDM\_CUSTOMER B WHERE CUST\_GRP\_HRCHY\_CD = 'G' AND A.CUST\_GRP\_ID=B.CUST\_GRP\_ID)**

**GROUP BY CUST\_GRP\_ID;**

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

**SELECT CUST\_GRP\_ID**

**FROM MDM\_CUSTOMER A**

**WHERE CUST\_GRP\_ID IS NOT NULL**

**AND NOT EXISTS (SELECT CUST\_GRP\_ID FROM MDM\_CUSTOMER B WHERE CUST\_GRP\_HRCHY\_CD = 'C' AND A.CUST\_GRP\_ID=B.CUST\_GRP\_ID)**

**GROUP BY CUST\_GRP\_ID;**

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



Viết câu 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

**SELECT prod\_unit\_amt AS MAX\_AMT, prod\_nm AS MAX\_NAME, B.minp AS MIN\_AMT, B.avgp AS AVG**

**from tb\_prod A,(SELECT MAX(prod\_unit\_amt) AS MAXP, MIN(prod\_unit\_amt) AS MINP, AVG(prod\_unit\_amt) AS AVGP from tb\_prod) B**

**WHERE prod\_unit\_amt = (SELECT MAX(PROD\_UNIT\_AMT) FROM tb\_prod) AND A.prod\_unit\_amt = B.maxp;**

Kết quả phải ra đc như sau:



**Câu 16: ta có table (TB\_ORD) và dữ liệu như bên dưới**



**A) Viết câu SQL lấy ra top3 sản phẩm đc bán nhiều nhất.**

**SELECT \***

**FROM (**

**SELECT PRO\_CD, DENSE\_RANK() OVER(ORDER BY COUNT(\*) DESC) AS TOP**

**FROM TB\_ORD**

**GROUP BY PRO\_CD**

**) A**

**WHERE A.TOP <= 3;**

**B) Viết câu SQL lấy ra cái ORD\_DT, ORD\_TM, PROD\_CD gần nhất theo CUST\_NO**

Kết quả mong đợi như sau:



**SELECT \***

**FROM (**

**SELECT CUST\_NO,ORD\_DTTM, ROW\_NUMBER () OVER (PARTITION BY CUST\_NO ORDER BY ORD\_DTTM DESC) rn**

**FROM TB\_ORD**

**GROUP BY CUST\_NO, ORD\_DTTM**

**) A**

**WHERE A.RN = 1;**

**C) viết câu SQL report xem trong tháng 06, 07, 08, 09 của 2019 sản phẩm có mã code là 00001 bán đc bao nhiêu cái.**

Kết quả mong đợi như sau: left outer join partition - 130



**SELECT A.DT, B.PRO\_CD, NVL(B.TOTAL,0) AS TOTAL**

**FROM**

**(**

**SELECT '201906' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201907' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201908' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201909' AS DT FROM DUAL**

**) A**

**LEFT OUTER JOIN**

**(SELECT B.PRO\_CD, SUBSTR(B.ORD\_DTTM, 1, 6) AS ORD\_DTTM, COUNT(\*)**

**AS TOTAL**

**FROM TB\_ORD B**

**GROUP BY B.PRO\_CD, SUBSTR(B.ORD\_DTTM, 1, 6)**

**) B PARTITION BY (B.PRO\_CD)**

**ON A.DT = B.ORD\_DTTM;**

**D) Giả sử lúc đầu sản phẩn 00001 có 100 cái, viết report để tính số lương remain theo tháng 06, 07, 08, 09**



**SELECT B.PRO\_CD,A.DT, NVL(B.TOTAL,0) AS TOTAL, 100 - NVL(SUM(B.TOTAL) OVER (PARTITION BY B.PRO\_CD ORDER BY A.DT),0) AS REMAIN**

**FROM**

**(**

**SELECT '201906' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201907' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201908' AS DT FROM DUAL**

**UNION ALL**

**SELECT '201909' AS DT FROM DUAL**

**) A**

**LEFT OUTER JOIN**

**(SELECT B.PRO\_CD, SUBSTR(B.ORD\_DTTM, 1, 6) AS ORD\_DTTM, COUNT(\*)**

**AS TOTAL**

**FROM TB\_ORD B**

**WHERE B.PRO\_CD = '00001'**

**GROUP BY B.PRO\_CD, SUBSTR(B.ORD\_DTTM, 1, 6)**

**) B PARTITION BY (B.PRO\_CD)**

**ON A.DT = B.ORD\_DTTM;**