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 số lượng record của 1 column có thể kết hợp với where và group by, nên sử dụng khi cần đếm số record của 1 column nào đó |
| 2 | Sum() | Tính tổng giá trị trong 1 column có thể kết hợp với where và group by, nên sử dụng khi cần tính tổng giá trị |
| 3 | MAX() | Tìm giá trị lớn nhất trong một column có thể kết hợp với where và group by, nên sử dụng khi cần tìm giá trị max của 1 column nào đó |
| 4 | MIN() | Tìm giá trị nhỏ nhất trong một column có thể kết hợp với where và group by, nên sử dụng khi cần tìm giá trị min của 1 column nào đó |
| 5 | NVL() | Dùng để thay thế giá trị null trong 1 column thành 1 giá trị truyền vào param thứ 2, dùng khi ta cần chọn 1 cột có chứa giá trị null |
| 6 | TO\_CHAR() | Ép kiểu dữ liệu sang string, dùng khi ta cần dữ kiệu bắt buộc phải là dạng string |
| 7 | T0\_DATE() | Ép kiểu dữ liệu sang dạng ngày tháng, dùng khi ta cần dữ kiệu bắt buộc phải là dạng ngày tháng |
| 8 | TO\_NUMBER() | Ép kiểu dữ liệu sang number, dùng khi ta cần dữ kiệu bắt buộc phải là dạng number |
| 9 | SUBSTR() | Cắt chuỗi, dùng khi cần lấy 1 chuỗi con bên trong 1 chuỗi cha |
| 10 | REPLACE() | Thay thế 1 chuỗi ký tự trong 1 chuỗi bằng 1 nhóm ký tự khác, sử dụng khi cần thay thế các ký tự trong chuỗi |
| 11 | REVERSE() | Đảo ngược chuỗi, dùng khi cần 1 chuỗi đảo ngược |
| 12 | DECODE() | Thực hiện logic của if-then-else trong câu query, dùng khi cần kiểm tra điều kiện trong câu query |
| 13 | TRUNC() | Dùng với number thì sẽ cắt ngắn 1 số đến 1 vị trí thập phân được chỉ định.  Dùng với date thì sẽ cắt ngắn ngày thành 1 format được chỉ định.  Dùng khi cần lấy ra 1 số hoặc ngày cụ thể nào đó |
| 14 | LENGTH() | Trả về độ dài cụ thể của 1 chuỗi, dùng khi cần lấy chiều dài của chuỗi |
| 15 | lPAD() | Đệm vào bên trái của chuỗi với tập ký tự 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() | Đệm vào bên phải của chuỗi với tập ký tự 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() | Loại bỏ tất cả các ký tự được chỉ định từ đầu hoặc cuối của một chuỗi. Mặc định sẽ xóa khoảng trắng ở 2 đầu của chuỗi khi chỉ truyền 1 tham số chuỗi |
| 18 | LTRIM() | Loại bỏ tất cả các ký tự được chỉ định từ phía bên trái của một chuỗi. Mặc định xóa khoảng trắng |
| 19 | RTRIM() | Loại bỏ tất cả các ký tự được chỉ định từ phía bên phải của một chuỗi. Mặc định xóa khoảng trắng |
| 20 | ROUND() | Dùng với number thì sẽ làm tròn số đến 1 số thập phân được chỉ định.  Dùng với date thì sẽ làm tròn đến ngày theo 1 format được chỉ định. |
| 21 | ADD\_MONTHS() | Thêm số tháng được chỉ định vào 1 ngày cụ thể nào đó |

**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

**Lấy tất cả data từ bảng MDM\_CUSTOMER có cột CUST\_LGL\_ENG\_NM có chứa ‘\_LOGISTICS’ sắp xếp kết quả theo cột CUST\_LOCL\_LANG\_NM nếu có giá trị null thì đưa lên đầu tiên**

B) ý nghĩa của việc dùng ESCAPSE

**Để xác định kí tự đặc biệt phía sau ESCAPSE như 1 kí tự thường**

C) Ý nghĩa của việc dung Nulls First.

**Trong quá trình sắp xếp thì những giá trị null sẽ được đưa lên đầu tiên**

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

**Dùng alias để đổi tên nhằm mục đích hiển thị tên khác (với thuộc tính), gán tên cho kết quả trả về của hàm hoặc để tên ngắn đi, và đỡ rườm rà dài dòng hơn. 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?

**Hợp lý vì dữ liệu được so sánh là dạng chuỗi nên ta phải ép kiểu của dữ liệu đưa vào thành dạng chuỗi cho cùng kiểu dữ liệu thì kết quả mới chính xác**

**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(\*) là đếm tất cả các hàng theo điều kiện where**

**Count(1) cũng tương tự count(\*) nhưng các giá trị bên trong bảng sẽ được thay thế bằng 1**

**Count(**CUST\_LOCL\_LANG\_NM**) là đếm các hàng không phải null của cột** CUST\_LOCL\_LANG\_NM

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

**Vì không có record nào trong cột** CUST\_LOCL\_LANG\_NM **tương ứng với điều kiện trong where**

**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 hàm sum sẽ bỏ qua các cột null và tính tổng giá trị sau đó NVL chỉ thực hiện 1 lần, còn Cách 2 thì hàm NVL sẽ thực hiện n lần (n giá trị null) sau đó sum sẽ phải cộng cả giá trị 0 vào nên sẽ chậm hơn cách 1**

**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 1 tốt hơn vì tốn ít thời gian thực hiện hơn**

**Dùng IN khi subquery là 1 list nhỏ, dùng EXISTS khi subquery là 1 list lớn**

**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 phải join 2 bảng lại với nhau, trong khi cách 2 thì không**

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

**TO\_CHAR(8988.80, ‘$9,999.000’)**

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

**TO\_CHAR(**TRUNC(8988.80, -3), **, ‘$9,999.000’)**

**TO\_CHAR(**TRUNC(820988.80, -3), **, ‘$999,999.000’)**

**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.

**Do tiền số ‘fm’ sẽ loại bỏ khoảng trắng và số 0 ở đầu string nên A bắt đầu từ 9 còn B bắt đầu từ khoảng trắng**

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

**SYSDATE, SYSDATE-1, SYSDATE+1**

**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**

**DECODE(COUNT(\*), 0, CONCAT(TO\_CHAR(SYSDATE, 'YYYYMMDD'), '0001'), MAX(ORD\_DTTM)+1)**

**AS ORD\_NO**

**FROM TB\_ORD WHERE ord\_dttm LIKE CONCAT(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**

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

**FROM MDM\_CUSTOMER**

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

**AND CUST\_GRP\_ID NOT IN (SELECT CUST\_GRP\_ID FROM MDM\_CUSTOMER WHERE CUST\_GRP\_HRCHY\_CD = 'G')**

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

**WITH TEMP AS (**

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

**FROM MDM\_CUSTOMER**

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

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

**(SELECT CUST\_GRP\_ID FROM MDM\_CUSTOMER WHERE CUST\_GRP\_HRCHY\_CD = 'C')**

**GROUP BY CUST\_GRP\_ID, CUST\_GRP\_HRCHY\_CD)**

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

**FROM MDM\_CUSTOMER**

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

**AND CUST\_GRP\_ID IN**

**(SELECT CUST\_GRP\_ID FROM TEMP**

**GROUP BY CUST\_GRP\_ID**

**HAVING COUNT(CUST\_GRP\_ID) = 2)**

**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 A.\*,**

**B.PROD\_NM AS MAX\_NAME**

**FROM (SELECT MAX(PROD\_UNIT\_AMT) AS MAX\_AMT,**

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

**AVG(PROD\_UNIT\_AMT) AS AVG**

**FROM TB\_PROD) A, TB\_PROD B**

**WHERE B.PROD\_UNIT\_AMT = A.MAX\_AMT**

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



A) viết cấu SQL lấy ra top3 sản phẩm đc bán nhiều nhất**.**

**SELECT PRO\_CD, COUNT(PRO\_CD) AS SL**

**FROM TB\_ORD**

**GROUP BY PRO\_CD**

**ORDER BY SL DESC**

**FETCH FIRST 3 ROWS ONLY;**

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

Kết quả mong đợi như sau:



**WITH TEMP AS (**

**SELECT CUST\_NO, MAX(ORD\_DTTM) AS ORD\_DTTM**

**FROM TB\_ORD**

**GROUP BY CUST\_NO)**

**SELECT A.CUST\_NO, A.ORD\_NO, A.ORD\_DTTM, TO\_NUMBER(A.PRO\_CD) AS PRO\_CD**

**FROM TB\_ORD A, TEMP B**

**WHERE A.CUST\_NO = B.CUST\_NO**

**AND A.ORD\_DTTM = B.ORD\_DTTM**

**ORDER BY A.CUST\_NO;**

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

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



**WITH TEMP1 AS (SELECT SUBSTR(ORD\_DTTM,1,6) AS MON, PRO\_CD, COUNT(PRO\_CD) AS SL FROM TB\_ORD**

**WHERE REGEXP\_LIKE(ORD\_DTTM, '20190[6789](\*)')**

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

**TEMP2 AS (SELECT DISTINCT SUBSTR(A.ORD\_DTTM, 1, 6) AS MON, B.PRO\_CD**

**FROM TB\_ORD A, TB\_ORD B**

**WHERE REGEXP\_LIKE(A.ORD\_DTTM, '20190[6789](\*)'))**

**SELECT B.MON, B.PRO\_CD, NVL(A.SL,0)**

**FROM TEMP1 A RIGHT JOIN TEMP2 B ON A.MON = B.MON AND A.PRO\_CD = B.PRO\_CD**

**ORDER BY B.PRO\_CD, B.MON**

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



**WITH TEMP1 AS (SELECT SUBSTR(ORD\_DTTM,1,6) AS MON, PRO\_CD, COUNT(PRO\_CD) AS SL FROM TB\_ORD**

**WHERE REGEXP\_LIKE(ORD\_DTTM, '20190[6789](\*)')**

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

**TEMP2 AS (SELECT DISTINCT SUBSTR(A.ORD\_DTTM, 1, 6) AS MON, B.PRO\_CD**

**FROM TB\_ORD A, TB\_ORD B**

**WHERE REGEXP\_LIKE(A.ORD\_DTTM, '20190[6789](\*)')**

**AND B.PRO\_CD = '00001')**

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

**FROM TEMP1 A RIGHT JOIN TEMP2 B ON A.MON = B.MON AND A.PRO\_CD = B.PRO\_CD**

**ORDER BY B.PRO\_CD, B.MON**