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 trong bảng, bỏ qua null  COUNT( [ALL | DISTINCT | \* ] expression) |
| 2 | Sum() | Trả về giá trị tổng của biểu thức, dùng cho dữ liệu kiểu số, Không tính giá trị NULL, biểu thức là NULL thì trả về NULL  SUM( [ALL | DISTINCT] expression) |
| 3 | MAX() | Dùng để lấy giá trị lớn nhất của một trường trong bảng  MAX( expression ); |
| 4 | MIN() | Dùng để lấy giá trị nhỏ nhất của một trường trong bảng  MIN( expression ); |
| 5 | NVL() | Dùng để thay thế giá trị NULL bằng giá trị mong muốn  NVL(e1, e2) |
| 6 | TO\_CHAR() | Dùng để chuyển đổi thành 1 chuỗi ký tự  TO\_CHAR(expr [, date\_format] [, nslparam]); |
| 7 | T0\_DATE() | Dùng để chuyển CHAR, VARCHAR2, NCHAR hoặc NVARCHAR2 sang định dạng ngày tháng  TO\_DATE (string, format, nls\_language) |
| 8 | TO\_NUMBER() | Dùng để chuyển CHAR, VARCHAR2, NCHAR hoặc NVARCHAR2 sang định dạng ngày số  TO\_NUMBER (string, format, nls\_language) |
| 9 | SUBSTR() | Trích xuất 1 chuỗi con từ chuỗi lớn với các tùy chọn khác nhau  SUBSTR( str, start\_position [, substring\_length, [, occurrence ]] );   * Nếu start\_position là 0, phần đầu của chuỗi con sẽ là ký tự đầu tiên của chuỗi. * Trong trường hợp start\_position > 0, đếm từ đầu của chuỗi đến để xác định ký tự đầu tiên của chuỗi con. * Nếu start\_position < 0, đếm ngược từ cuối chuỗi để tìm ký tự đầu tiên của chuỗi con. * Substring\_length * *substring\_length* xác định số ký tự trong chuỗi con.  Nếu *substring\_length* bị bỏ qua,  hàm sẽ trả về tất cả các ký tự bắt đầu từ *start\_position*. * Trong trường hợp *substring\_length* nhỏ hơn 1,  hàm trả về null. |
| 10 | REPLACE() | Thay thế tất cả các lần xuất hiện của 1 chuỗi con trong chuỗi bằng chuỗi khác  REPLACE(string\_expression, string\_pattern [,string\_replacement]) |
| 11 | REVERSE() | Đảo ngược chuỗi ký tự |
| 12 | DECODE() | Hoạt động như một câu lệnh IF – ELSE hoặc SWITCH CASE  So sánh *expr* với từng *search* giá trị một. Nếu *expr* bằng *search*, thì trả về giá trị tương ứng *result*. Nếu không tìm thấy kết quả phù hợp nào, thì trả về *default*. Nếu *default* bị bỏ qua, thì trả về null.  **DECODE**(expr, search, result  [, search, result ]...  [, default ] ) |
| 13 | TRUNC() | Rút gọn số đến 1 số thập phân nhất định (được chỉ định bởi tham số truyền vào)  TRUNC(date, format) |
| 14 | LENGTH() | Trả về số ký tự của một chuỗi  LENGTH(string\_expression); |
| 15 | lPAD() | Trả về chuỗi mới được thêm các ký tự bên trái với độ dài nhất định  LPAD(source\_string, target\_length [,pad\_string]);  Ví dụ: LPAD(‘ABCD’, 6, ’\*’) 🡪 Result: ‘\*\*ABCD’  Nếu giá trị target\_length < source\_string hàm LPAD () sẽ rút ngắn giá trị chuỗi source\_string thành giá trị target\_length và cắt bớt chuỗi gốc |
| 16 | RPAD() | Trả về chuỗi mới được thêm các ký tự bên phải với độ dài nhất định  RPAD(source\_string, target\_length [,pad\_string]);  Ví dụ: RPAD(‘ABCD’, 6, ’\*’) 🡪 Result: ‘ABCD\*\*’  Nếu giá trị target\_length < source\_string hàm RPAD () sẽ rút ngắn giá trị chuỗi source\_string thành giá trị target\_length và cắt bớt chuỗi gốc |
| 17 | TRIM() | Loại bỏ ký tự ở đầu cuối hoặc cả hai của chuỗi  TRIM( [ [ LEADING | TRAILING | BOTH ] trim\_character FROM ] trim\_source) |
| 18 | LTRIM() | Loại bỏ các ký tự được chỉ định từ bên trái của chuỗi  LTRIM(trim\_source, [set])  LTRIM(‘ ABC’, ‘ ‘ ) -> result: ‘ABC’ |
| 19 | RTRIM() | Loại bỏ các ký tự được chỉ định từ bên phải của chuỗi  RTRIM(trim\_source, [set])  LTRIM(‘ABC ’, ‘ ‘ ) -> result: ‘ABC’ |
| 20 | ROUND() | Trả về một số được làm tròn đến một số chữ số thập phân nhất định.  ROUND( number [, decimal\_places] )   * number * Số để làm tròn. * decimal\_places * Không bắt buộc. Số chữ số thập phân được làm tròn đến. Giá trị này phải là một số nguyên. Nếu tham số này bị bỏ qua, hàm ROUND sẽ làm tròn số đến 0 chữ số thập phân.   ROUND(125.315, 1) -> result: 125.3 |
| 21 | ADD\_MONTHS() | Trả về một ngày với một số tháng cụ thể được thêm vào.  ADD\_MONTHS(date\_expression, month)  ADD\_MONTHS('01-Aug-03', 3) 🡪 result: '01-Nov-03' |

**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

1. Vui lòng giải thích ý nghĩa của câu SQL trên

* Lấy tất cả data từ bảng MDM\_CUSTOMER với điều kiện cột ‘CUST\_LGL\_ENG\_NM’ chứa chuỗi ‘\_LOGISTICS’ và sắp xếp kết quả theo CUST\_LOCL\_LANG\_NM đưa giá trị null lên đầu

1. ý nghĩa của việc dùng ESCAPSE

* Dùng ESCAPE để xác định ‘\_LOGISTICS’ là 1 chuỗi
* Trong ví dụ ‘\_LOGISTICS%’ nếu không dùng ESCAPE thì dấu \_ có nghĩa là cần ít nhất 1 ký tự trước LOGISTICS, nhưng ở đây ý cần tìm là dấu ‘\_‘. nên cần sử dụng ESCAPE

1. Ý nghĩa của việc dung Nulls First.

* Đưa những data có giá trị null lên đầu tiên

1. Ý nghĩa của việc dung alias, có nên dung alias trong mọi trường hợp không?

* Dùng để đặt tên cho cột và bảng , dùng để rút ngắn SQL, không nên vì trong trường hợp ít bảng thì không cẩn sử dụng tới alias

**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 dùng TO\_CHAR(CUST\_SEQ) = '201791' có hợp lý không, tại sao?

* Việc sử dụng TO\_CHAR(CUST\_SEQ) kiểu dữ liệu của cust\_seq thanh chuỗi kí tự việc so sánh 2 chuỗi kí tự sẽ chính xác hơn

**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 (\*): đếm tổng số hàng trong bảng, bao gồm các giá trị null
* COUNT (1): giống COUNT (\*) chỉ khác là gán giá trị 1
* COUNT (CUST\_LOCL\_LANG\_NM): Đếm tất cả các hàng trong cột được chỉ định không bao gồm giá trị null

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

* Vì tất cả các hàng đều 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: cost = 3, time: 0,024
* Cách 2: cost = 3, time: 0,039
* Cách 1 tốt hơn vì cách 1 chạy nhanh hơn khi test trong SQL tool

**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 2 sẽ sai khi gặp trường hợp có record mà Col1 hoăc Col2 là NULL và cột còn lại là số đẫn đến kết quả 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 dùng khi nào và EXISTS đc dung khi nào.

* Cách 1 tốt hơn vì cost và time thấp hơn khi test bằng sql tool
* IN dùng khi kết quả truy vấn phụ nhỏ, EXISTS dùng khi kết quả truy vấn phụ 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. Tại vì cách 1 thực hiện phép kết bảng sẽ tốn tài nguyên hơn so với cách 2 viết 1 subquery trong select

**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') AS r1,  to\_char(trunc(820988.80, - 3), '$999,999.999') AS r2  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.

* Vì độ dài của to\_char (98765, ‘fm00000’) là 5 còn độ dài của to\_char (98765,’00000’) là 6. Do fm loại bỏ khoảng trắng của dấu +, - trong chuỗi số trả về

**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 |

**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') || to\_char(COUNT(\*) + 1, 'fm0000') seq  FROM tb\_ord  WHERE ord\_dttm 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**

|  |
| --- |
| SELECT cust\_grp\_id  FROM  (SELECT cust\_grp\_id,  cust\_grp\_hrchy\_cd  FROM mdm\_customer  WHERE cust\_grp\_id NOT IN  (SELECT cust\_grp\_id  FROM mdm\_customer  WHERE cust\_grp\_hrchy\_cd LIKE 'G' )  GROUP BY cust\_grp\_id,  cust\_grp\_hrchy\_cd)  GROUP BY cust\_grp\_id  HAVING COUNT(\*) = 1 |

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  WHERE cust\_grp\_id NOT IN (  SELECT  cust\_grp\_id  FROM  mdm\_customer  WHERE  cust\_grp\_hrchy\_cd like 'C'  )  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

|  |
| --- |
| SELECT  MAX(prod\_unit\_amt) AS max\_amt,  MIN(prod\_unit\_amt) AS min\_amt,  AVG(prod\_unit\_amt) AS avg,  MAX(prod\_nm) KEEP(DENSE\_RANK FIRST ORDER BY prod\_unit\_amt DESC NULLS LAST) AS max\_name  FROM  tb\_prod  WHERE  prod\_unit\_amt IS NOT NULL; |

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 top 3 sản phẩm đc bán nhiều nhất**.**

|  |
| --- |
| select \*  from ( select pro\_cd, count(pro\_cd) as num  from tb\_ord  group by pro\_cd)  where rownum <= 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 tb.cust\_no, tb.ord\_dttm, tb.ord\_no, tb.pro\_cd  from tb\_ord tb, (  select cust\_no, max(ord\_dttm) max\_date  from tb\_ord  group by cust\_no  ) tmp  where tb.cust\_no = tmp.cust\_no  and tb.ord\_dttm = tmp.max\_date |

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, NVL(B.pro\_cd, '00001') as pro\_cd, NVL(B.total, 0)  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 join (  select pro\_cd, substr(ord\_dttm,1,6) as ord\_dttm, count(\*) as total  from tb\_ord  where pro\_cd = '00001'  group by pro\_cd, substr(ord\_dttm,1,6) ) B  on A.dt = B.ord\_dttm |

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



|  |
| --- |
| select A.dt, NVL(B.pro\_cd, '00001') as pro\_cd, 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 join (  select pro\_cd, substr(ord\_dttm,1,6) as ord\_dttm, count(\*) as total  from tb\_ord  where pro\_cd = '00001'  group by pro\_cd, substr(ord\_dttm,1,6) ) B  on A.dt = B.ord\_dttm |