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 bản ghi có trong 1 bảng dữ liệu , những giá trị null bỏ qua không tính |
| 2 | Sum() | Trả về giá trị tổng của biểu thức , bỏ qua các giá trị null , nếu biểu thức là null thì trả về giá trị null |
| 3 | MAX() | Tìm giá trị lớn nhất trong 1 nhóm |
| 4 | MIN() | Tìm giá trị nhỏ nhất trong 1 nhóm |
| 5 | NVL() | Thay những giá trị null của cột này bằng giá trị đã điền |
| 6 | TO\_CHAR() | Chuyển đổi thành một chuỗi kí tự |
| 7 | T0\_DATE() | Chuyển đổi kiểu dữ liệu CHAR, VARCHAR2, NCHAR hoặc NVARCHAR2 thành kiểu dữ liệu DATE |
| 8 | TO\_NUMBER() | Chuyển đổi kiểu dữ liệu CHAR, VARCHAR2, NCHAR hoặc NVARCHAR2 thành kiểu dữ liệu NUMBER |
| 9 | SUBSTR() | Lấy ra chuỗi nhỏ từ chuỗi lớn |
| 10 | REPLACE() | Thay thế chuỗi bằng một chuỗi khác |
| 11 | REVERSE() | Đảo ngược chuỗi ký tự |
| 12 | DECODE() | Hỗ trợ tính năng câu lệnh IF-THEN-ELSE hay CASE |
| 13 | TRUNC() | Thu gọn một số đến một vị trí nào đó sau dấu phẩy (vị trí được chỉ định bởi tham số truyền vào) |
| 14 | LENGTH() | Trả về độ dài của chuỗi truyền vào |
| 15 | LPAD() | Trả về một chuỗi mới. (ví dụ: LPAD(‘ABC’,6) -> result: ‘ ABC’) |
| 16 | RPAD() | Trả về một chuỗi mới. (vi dụ: RPAD(‘ABC’,6) -> result: ‘ABC ‘) |
| 17 | TRIM() | Loại bỏ tất cả các ký tự được chỉ định từ đầu hoặc cuối của một chuỗi. (TRIM ([LEADING|TRAILING|BOTH] trim\_character FROM string)) |
| 18 | LTRIM() | Loại bỏ tất cả các ký tự được chỉ định từ phía bên trái của một chuỗi. (LTRIM(‘ ABC’, ‘ ‘ ) -> result: ‘ABC’) |
| 19 | RTRIM() | Loại bỏ tất cả các ký tự được chỉ định từ phía bên phải của một chuỗi. (LTRIM(‘ABC ’, ‘ ‘ ) -> result: ‘ABC’) |
| 20 | ROUND() | Trả về một số được làm tròn đến một số chữ số thập phân nhất định.( 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('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

A) Vui lòng giải thích ý nghĩa của câu SQL trênLấy tất cả data từ bảng MDM\_CUSTOMER có cột CUST\_LGL\_ENG\_NM có chứa chuỗi ‘\_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: Biến ký tự sau escape trở thành ký tự bình thường

C) Ý nghĩa của việc dung Nulls First: để những ô null lên đầu tiên

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

Alias dung để đặt tên tạm thời cho cột hoặc bảng. alias sử dụng khi cái query của chúng ta có nhiều 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 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 chuổi 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): gán giá trị từ dấu ngoặc đơn cho mọi hàng trong bảng rồi đếm đếm tổng số hàng trong bảng, bao gồm các giá trị null

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: tại 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 tốt hơn. Tại cách 1 chạy nhanh hơn vì ờ cách 1 hàm sum thực hiện trước nó sẽ tính tổng theo COL1 và bỏ qua các cột null sau đó NVL sẽ chạy 1 lần để kiểm tra nếu kq của hàm sum là null thì sẽ đc gán lại thành 0 , còn với cách 2 thì NVL sẽ chạy trước , scan 1 lần trên COL1 và thay thế các giá trị null về 0 sau đó mới dung hàm sum tính tổng dẫn đến sẽ tốn hiệu xuất hơn so với 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 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); |

* Cách 2 tốt hơn trong trường hợp này vì IN chỉ phù hợp cho dữ liệu nhỏ , dữ liệu cố định , nhưng mà bên trong IN và EXISTS thì lại là 1 subquery , mà subquey thì dữ liệu có thể lớn dần theo thời gian nên cách 2 vẫn tốt hơn trong trường hợp này

**IN** đc dung khi nào và EXISTS đc dung khi nào.

* Dùng IN khi kết quả truy vấn phụ nhỏ, dùng EXISTS 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 nó sẽ tốn tài nguyên hơn so với cách 2 chỉ viết 1 subquery trong select

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

to\_char(8988.80, 'L9G999G999D000')

**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),'$9,999.999')**

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.

* Tại độ dài của to\_char(98765,’fm00000’) là 5 số và độ dài của to\_char(98765,’00000’) là 6 số. Do fm loại 1 khoảng trắng dành để biểu diễn dấu + - trong 1 chuỗi số trả về.

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

select sysdate from dual

select sysdate -1 from dual

select sysdate +1 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,

COUNT(\*) OVER(PARTITION BY cust\_grp\_id) volumn,

cust\_grp\_hrchy\_cd

FROM

mdm\_customer

GROUP BY

cust\_grp\_id,

cust\_grp\_hrchy\_cd

)

WHERE

cust\_grp\_hrchy\_cd != 'G'

AND volumn = 1;

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

**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)

select max(PROD\_UNIT\_AMT) from tb\_prod;

1. Lấy giá trị min(PROD\_UNIT\_AMT)

select min(PROD\_UNIT\_AMT) from tb\_prod;

1. Lấy giá trị trung bình PROD\_UNIT\_AMT

select avg(PROD\_UNIT\_AMT) from tb\_prod;

1. Lấy tên của sản phẩm có PROD\_UNIT\_AMT lớn nhất

SELECT

prod\_nm

FROM

(

SELECT

prod\_nm,

RANK()

OVER(

ORDER BY

prod\_unit\_amt DESC NULLS LAST

) AS rankq

FROM

tb\_prod

)

WHERE

rankq = 1

Kết quả phải ra đc như sau:



**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 \*

from (select pro\_cd, count(pro\_cd) as countp from tb\_ord group by pro\_cd order by countp desc) A

where ROWNUM <= 3;

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

Kết quả mong đợi như sau:



SELECT

p.cust\_no,

p. maxdate,

ord.ord\_no,

ord.ord\_dttm,

ord.pro\_cd

FROM

(

SELECT

cust\_no,

MAX(ord\_dttm) maxdate

FROM

tb\_ord

GROUP BY

cust\_no

) p,

tb\_ord ord

WHERE

p.cust\_no = ord.cust\_no

AND p.maxdate = ord.ord\_dttm

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



SELECT

a.mon,

nvl(b.pro\_cd, '00001') AS pro\_cd,

nvl(b.total, 0)

FROM

(

SELECT DISTINCT

( substr(ord\_dttm, 1, 6) ) AS mon

FROM

tb\_ord

WHERE

substr(ord\_dttm, 1, 5) LIKE '20190'

AND substr(ord\_dttm, 6, 1) IN ( '6', '7', '8',

'9' )

) a

LEFT OUTER JOIN (

SELECT

a.mon,

pro\_cd,

COUNT(pro\_cd) AS total

FROM

(

SELECT

substr(ord\_dttm, 1, 6) AS mon,

pro\_cd

FROM

tb\_ord

WHERE

substr(ord\_dttm, 1, 5) LIKE '20190'

AND substr(ord\_dttm, 6, 1) IN ( '6', '7', '8',

'9' )

AND pro\_cd LIKE '00001'

) a

GROUP BY

mon,

pro\_cd

) b ON a.mon = 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



SELECT

doo.pro\_cd,

doo.total,

100 - nvl(SUM(doo.total)

OVER(PARTITION BY doo.pro\_cd

ORDER BY

doo.mon

), 0) AS remain

FROM

(

SELECT

a.mon AS mon,

nvl(b.pro\_cd, '00001') AS pro\_cd,

nvl(b.total, 0) AS total

FROM

(

SELECT DISTINCT

( substr(ord\_dttm, 1, 6) ) AS mon

FROM

tb\_ord

WHERE

substr(ord\_dttm, 1, 5) LIKE '20190'

AND substr(ord\_dttm, 6, 1) IN ( '6', '7', '8',

'9' )

) a

LEFT OUTER JOIN (

SELECT

a.mon,

pro\_cd,

COUNT(pro\_cd) AS total

FROM

(

SELECT

substr(ord\_dttm, 1, 6) AS mon,

pro\_cd

FROM

tb\_ord

WHERE

substr(ord\_dttm, 1, 5) LIKE '20190'

AND substr(ord\_dttm, 6, 1) IN ( '6', '7', '8',

'9' )

AND pro\_cd LIKE '00001'

) a

GROUP BY

mon,

pro\_cd

) b ON a.mon = b.mon

) doo