Database test

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

|  |  |  |
| --- | --- | --- |
| Stt | Tên Hàm | Mục đích sử dụng & nên sử dụng khi nào |
| 1 | Count() | Dùng để đếm số lượng |
| 2 | Sum() | Dùng để tính tổng |
| 3 | MAX() | Dùng để tìm số lớn nhất |
| 4 | MIN() | Dùng để tìm số nhỏ nhất |
| 5 | NVL() | Cài đặt giá trị mặc định cho các của cột có giá trị null |
| 6 | TO\_CHAR() | Dùng để chuyển đổi thành 1 chuỗi ký tự |
| 7 | T0\_DATE() | Dùng để chuyển đổi kiểu dữ liệu CHAR, VARCHAR, NVARCHAR, NCHAR thành kiểu DATE |
| 8 | TO\_NUMBER() | Dùng để chuyển đổi kiểu dữ liệu CHAR, VARCHAR, NVARCHAR, NCHAR thành kiểu NUMBER |
| 9 | SUBSTR() | Cắt chuỗi nhỏ từ chuỗi lớn theo độ dài |
| 10 | REPLACE() | Dùng để thay thế chuỗi thành một chuỗi khác |
| 11 | REVERSE() | Dùng để đảo ngược chuỗi ký tự |
| 12 | DECODE() | Giúp để có thể sử dụng “CASE….WHEN….THEN….ELSE….END” |
| 13 | TRUNC() | Dung để thu gọn một số tới vị trị nào đó sau dấu phẩy. Vị trị này được quyết định theo tham số truyền vào. |
| 14 | LENGTH() | Tính độ dài của 1 chuỗi hoặc đếm số ký tự trong 1 chuỗi bao gồm khoảng trắng ‘space’ |
| 15 | LPAD() | Dùng để đệm vào bên trái của chuỗi với tập ký tự được xác định trong tham số của nó |
| 16 | RPAD() | Dùng để đệm vào bên phải của chuỗi với tập ký tự được xác định trong tham số của nó |
| 17 | TRIM() | Dùng để xoá khoảng trắng 2 bên của một chuỗi |
| 18 | LTRIM() | Dùng để xoá khoảng trắng bên trái của một chuỗi |
| 19 | RTRIM() | Dùng để xoá khoảng trắng bên phải của một chuỗi |
| 20 | ROUND() | Trả về một số được làm tròn đến một trị trí thập phân theo tham số đầu vào |
| 21 | ADD\_MONTHS() | Dùng để thêm hoặc bớt một hoặc nhiều tháng dựa vào tham số truyền vào (số dương sẽ thêm, số âm sẽ bớt) |

**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

\_ Hiển thị tất cả các cột có trong bảng MDM\_CUSTOMER và sắp xếp theo cột CUST\_LOCL\_LANG\_NM với các giá trị null được đưa lên dòng đầu tiên

B) ý nghĩa của việc dùng ESCAPSE

\_ Dùng để chuyển ký tự sau ESCAPSE thành ký tự bình thường

C) Ý nghĩa của việc dung Nulls First.

\_ Để những cell có giá trị bị null được đưa lên trên

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

\_ Dùng alias giúp cú pháp clean hơn, dễ đọc hơn. Tuy nhiên nên sử dụng trong trường hợp truy vấn nhiều hơn 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?

\_ Không hợp lý vì Oracle đã autoparse CUST\_SEQ thành dạng CHAR rồi.

**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)

\_ Đếm tổng số record có trong table bao gồm các giá trị null

\_ Tạo ra cột ảo mang giá trị của giống giá trị bên trong dấu ngoặc đơn sau đó thì bắt đầu đếm, bao gồm các giá trị null

\_ Đếm các giá trị trong cột CUST\_LOCL\_LANG\_NM, không bao gồm giá trị null

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

Vì cột CUST\_LOCL\_LANG\_NM trong record được trả về khi thực thi câu truy vấn mang kết quả null nhưng COUNT(col\_name) không nhận giá trị 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 vì cách 1 chỉ thực thi NVL 1 lần còn cách 2 thực thi NVL nhiều lần tuỳ theo số record trong bảng

**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 bên có giá trị null thì kết quả sẽ trả về là null cho bên kia có giá trị gì đi nữa. Điều này dẫn đến kết quả 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 sẽ tốt hơn vì trong trường hợp này subquery trả về kết quả chúng ta không biết

\_ Nếu Subquery trả về kết quả ngắn thì nên dùng IN, ngược lại subquery trả về kết quả lớn thì nên dùng EXISTS

**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 sẽ tốt hơn vì cách 1 sẽ phải full scan cả 2 bảng TB\_ORD và TB\_PROD sau đó mới thực thi câu A.PRO\_CD = B.PROD\_CD. Còn cách 2 sẽ thực thi câu subquery để tìm PROD\_NM mà không cần phải full scan thế bảng TB\_PROD

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

\_ SELECT TO\_CHAR (8988.80,'L9G999G999D000') 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), 'L9G999G999D000') A, TO\_CHAR(TRUNC(820988.80,-3), 'L9G999G999D000') B 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.

\_ Do length(to\_char(98765,’fm00000’)) là 5 còn length(to\_char(98765,’ 00000’)) là 6 vì vị trí [0] dùng để biểu diễn dấu + hoặc – khi thêm ‘fm’ thì số dương sẽ không còn dấu + nên length =5 nên khi substr từ 1 đến 3 A = 987, B = +98 nhưng khi display thì không hiện ‘+’

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

select to\_char(sysdate -1,'dd/mm/yyyy') YESTERDAY,

to\_char(sysdate,'dd/mm/yyyy') TODAY,

to\_char(sysdate +1 ,'dd/mm/yyyy') 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') || 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**

\_ select distinct(CUST\_GRP\_ID) from mdm\_customer where CUST\_GRP\_HRCHY\_CD IN ('I','C');

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

\_ select distinct(CUST\_GRP\_ID) from mdm\_customer where CUST\_GRP\_HRCHY\_CD like 'I' AND CUST\_GRP\_HRCHY\_CD like 'G'

**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) PROD\_UNIT\_AMT

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

, AVG(PROD\_UNIT\_AMT) PROD\_UNIT\_AMT

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

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 cấu SQL lấy ra top 3 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 <4;

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 \*

FROM (

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

FROM TB\_ORD

) A

WHERE A.RN = 1;

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

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

WHERE B.PRO\_CD ='00001'

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



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

FROM

(

SELECT '201906' AS MON FROM DUAL

UNION ALL

SELECT '201907' AS MON FROM DUAL

UNION ALL

SELECT '201908' AS MON FROM DUAL

UNION ALL

SELECT '201909' AS MON 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.MON = B.ORD\_DTTM

WHERE B.PRO\_CD ='00001';