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() | Để 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() | Để 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() | 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() | 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 các data trong table MDM\_CUSTOMER sao cho CUST\_LGL\_ENG\_NM có 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**

C) Ý nghĩa của việc dung 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 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?

**Cách sử dụng TO\_CHAR(CUST\_SEQ) = '201791' vì mặc định oracle có hỗ trợ so sánh số khi nó là chuỗi hoặc khi nó là số. Việc sử dụng TO\_CHAR sẽ làm tăng cost vì phải chuyển đổi dữ liệu thuộc tính cust\_seq ở mỗi dòng thành chuỗi.**

**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.

* **Mặc dù hiệu suất của cả 2 cách là như nhau nhưng cách 1 tốt hơn vì nó thể hiện rõ mục đích câu truy vấn là tìm kiểm trong bảng D giá trị PROD\_CD giống trong A và B.**
* **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, 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 cấu 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**
* **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**

**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

**INSERT INTO TB\_ORD(CUST\_NO, ORD\_NO, PRO\_CD, ORD\_DTTM)**

**SELECT :CUSTNO, :ORDNO, :PROCD,**

**DECODE(COUNT(\*), 0, RPAD(TO\_CHAR(SYSDATE, 'YYYYMMDD'), 12, '0') + 1, MAX(ORD\_DTTM) + 1)**

**FROM TB\_ORD**

**WHERE ORD\_DTTM LIKE TO\_CHAR(SYSDATE, 'YYYYMMDD')||'%'**

**AND 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, CUST\_GRP\_HRCHY\_CD**

**FROM MDM\_CUSTOMER**

**WHERE CUST\_GRP\_ID NOT IN (SELECT CUST\_GRP\_ID**

**FROM MDM\_CUSTOMER B**

**WHERE CUST\_GRP\_HRCHY\_CD = 'G')**

**AND 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 A**

**WHERE CUST\_GRP\_ID NOT IN (SELECT CUST\_GRP\_ID**

**FROM MDM\_CUSTOMER**

**WHERE CUST\_GRP\_HRCHY\_CD = 'C')**

**AND EXISTS (SELECT 1**

**FROM MDM\_CUSTOMER B**

**WHERE A.CUST\_GRP\_HRCHY\_CD = 'I' AND B.CUST\_GRP\_HRCHY\_CD = 'G'**

**AND A.CUST\_GRP\_ID = B.CUST\_GRP\_ID);**

**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 AND ROWNUM <= 1;**

**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)**

**FROM TB\_ORD**

**GROUP BY pro\_cd**

**ORDER BY COUNT(PRO\_CD) 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 ORD AS**

**(**

**SELECT**

**CUST\_NO,**

**ORD\_NO,**

**ORD\_DTTM,**

**TO\_NUMBER(PRO\_CD) AS PRO\_CD,**

**ROW\_NUMBER() OVER (PARTITION BY CUST\_NO ORDER BY ORD\_DTTM DESC) AS RN**

**FROM TB\_ORD**

**WHERE ORD\_NO != 'temp'**

**)**

**SELECT CUST\_NO, ORD\_NO, ORD\_DTTM, PRO\_CD**

**FROM ORD**

**WHERE RN = 1;**

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 DATEORDER AS(**

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

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

**WHERE A.ORD\_DTTM BETWEEN '201906000000' AND '201909302359'),**

**DATEORDERORIGIN AS (**

**SELECT SUBSTR(ORD\_DTTM, 1, 6) AS MON, PRO\_CD, COUNT(\*) AS TOTAL**

**FROM TB\_ORD**

**WHERE ORD\_DTTM BETWEEN '201906000000' AND '201909302359'**

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

**SELECT DO.MON, DO.PRO\_CD, NVL(DOO.TOTAL,0) AS TOTAL**

**FROM DATEORDER DO LEFT JOIN DATEORDERORIGIN DOO**

**ON DO.MON = DOO.MON AND DO.PRO\_CD = DOO.PRO\_CD**

**ORDER BY PRO\_CD, 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 DATEORDER AS(**

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

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

**WHERE SUBSTR(A.ORD\_DTTM, 1, 6) BETWEEN '201906' AND '201909'),**

**DATEORDERORIGIN AS (**

**SELECT SUBSTR(ORD\_DTTM, 1, 6) AS MON, PRO\_CD, COUNT(\*) AS TOTAL**

**FROM TB\_ORD**

**WHERE SUBSTR(ORD\_DTTM, 1, 6) BETWEEN '201906' AND '201909'**

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

**SELECT DO.MON, DO.PRO\_CD, NVL(DOO.TOTAL,0) AS TOTAL,**

**:REMAIN - NVL(SUM(DOO.TOTAL) OVER (PARTITION BY DO.PRO\_CD ORDER BY DO.MON),0) AS REMAIN**

**FROM DATEORDER DO LEFT JOIN DATEORDERORIGIN DOO**

**ON DO.MON = DOO.MON AND DO.PRO\_CD = DOO.PRO\_CD**

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

**ORDER BY PRO\_CD, MON;**

**với giá trị của param :REMAIN là 100**