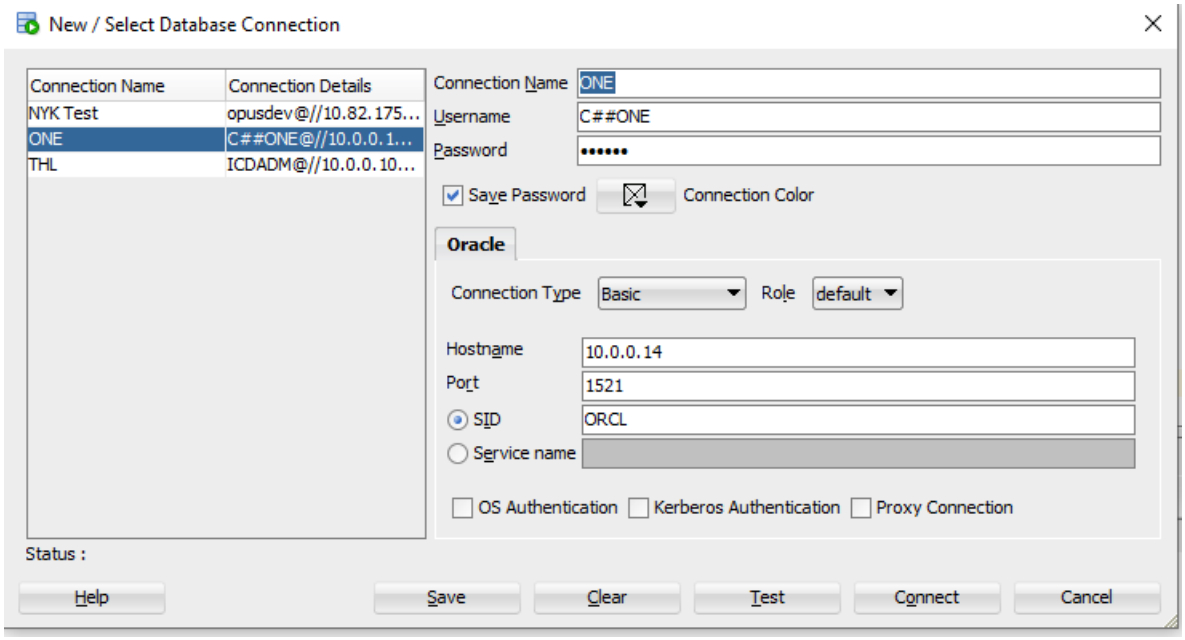
Databse test



Câu 1: Giải thích các hàm thông dụng sau

Stt	Tên Hàm	Mục đích sử dụng & nên sử dụng khi nào
1	Count()	COUNT([ALL | DISTINCT | *] expression): Là hàm tổng hợp trả về số dòng trong một nhóm. Hàm COUNT() nhận vào một mệnh đề All, DISTINCT hoặc * + COUNT(*): Bao gồm duplicate values và NULL values + COUNT(DISTINCT expression): Bao gồm unique values và non-null values + COUNT(ALL expression): Bao gồm duplicate values và non-null values Mặc định là ALL nếu để trống
2	Sum()	SUM([ALL | DISTINCT] expression): Là hàm trả về tổng của một tập các giá trị (không tính NULL value). Hàm SUM() nhận vào một mệnh đề DISTINCT hoặc ALL + DISTINCT: Chỉ unique values + ALL: Bao gồm cả duplicate values Mặc định là ALL nếu để trống
3	MAX()	MAX(expression): Là hàm trả về giá trị lớn nhất của một tập các giá trị (không tính NULL value).
4	MIN()	MIN(expression): Là hàm trả về giá trị nhỏ nhất của một tập các giá trị (không tính NULL value).
5	NVL()	NVL(e1, e2): Cho phép ta thay thế một giá trị NULL thành một giá trị khác Hàm NVL này nhận vào 2 tham số e1, e2. Nếu e1 là NULL, thì hàm này sẽ trả về e2, ngược lại thì trả về e1 Hai tham số e1 và e2 có thể có cùng kiểu hoặc khác kiểu dữ liệu. Nếu khác kiểu thì sẽ được chuyển đổi dựa trên các quy tắc sau: + Nếu e1 là kiểu ký tự thì e2 sẽ được chuyển đổi thành kiểu dữ liệu của e1 trước khi so sánh với NULL

		+ Nếu e1 là kiểu số thì Oracle sẽ kiểm trả xem kiểu dữ liệu nào sẽ có độ ưu tiên cao hơn để chuyển đổi ngầm định tham số còn lại về kiểu có độ ưu tiên cao hơn + Nếu không thể chuyển đổi các kiểu dữ liệu thì sẽ phát sinh lỗi
6	TO_CHAR()	TO_CHAR(expr [, date_format] [, nslparam]): Là hàm chuyển đổi dữ liệu có kiểu DATE hoặc INTERVAL thành dạng string có format được chỉ định Hàm TO_CHAR() nhận vào 3 tham số: + expr: Giá trị kiểu DATE hoặc INTERVAL cần được chuyển đổi + date_format: Dạng string chỉ định format cần được chuyển đổi thành + date_format là tham số không bắt buộc. Nếu không được truyền vào thì mặc định hàm TO_CHAR sẽ sử dụng các format mặc định của từng kiểu dữ liệu. + nlsparam: Chỉ định ngôn ngữ, tên cho ngày và tháng. Ví dụ Monday hay Mon, January hay Jan,... Tham số này không bắt buộc
7	T0_DATE()	TO_DATE (string, format, nls_language): Hàm TO_DATE có tác dụng chuyển định dạng của một chuỗi hoặc một dãy số sang định dạng ngày tháng Hàm To_DATE() nhận vào 3 tham số: + string: giá trị dạng chuỗi cần được chuyển đổi + format: format cần được chuyển đổi thành. Nếu không được chỉ định thì sẽ có dạng format mặc định là DD-MON-YY (31-DEC-2000) + nls_language: Chỉ định ngôn ngữ cho ngày và tháng. Tham số này không bắt buộc, nếu để trống sẽ sử dụng ngôn ngôn của session hiện tại
8	TO_NUMBER()	TO_NUMBER(string1 [, format_mask] [, nls_language]): Hàm TO_NUMBER có tác dụng chuyển đổi string thành số Hàm To_DATE() nhận vào 3 tham số: + string: giá trị dạng chuỗi cần được chuyển đổi + format: format cần được chuyển đổi thành thành số. Tham số này không bắt buộc + nls_language: Chỉ định ngôn ngữ cho số. Tham số này không bắt buộc, nếu để trống sẽ sử dụng ngôn ngôn của session hiện tại
9	SUBSTR()	SUBSTR(str, start_position [, substring_length]): Hàm SUBSTR lấy một chuỗi con từ chuỗi ban đầu Hàm SUBSTR() nhận vào 3 tham số: + str: giá trị dạng chuỗi + start_position: Vị trí bắt đầu lấy Nếu >=0: Vị trí bắt đầu từ đầu chuỗi Nếu <0: Vị trí bắt đầu từ cuối chuỗi + substring_length: Độ dài chuỗi con cần lấy. Nếu không được truyền vào mặc định sẽ lấy hết tính từ vị trí bắt đầu. Trong trường hợp substring_length < 1 thì trả về chuỗi rỗng
10	REPLACE()	REPLACE(string_expression, string_pattern [,string_replacement]): Hàm REPLACE dùng để thay thế tất cả các lần xuất hiện thỏa string_pattern thành string_replacement Hàm REPLACE() nhận vào 3 tham số: + string_expression: giá trị dạng chuỗi + string_pattern: chuỗi con cần được thay thế + string_replacement: chuỗi thay thế

11	REVERSE()	REVERSE(string_expression): Hàm REVERSE dùng để đảo ngược một chuỗi + string_expression: giá trị dạng chuỗi cần đảo ngược
12	DECODE()	DECODE(expression , search , result [, search , result]... [, default]): Hàm DECODE dùng để hiện thực cấu trúc rẽ nhánh (If Else, Case when) + expression: Biểu thức để so sánh + search: Giá trị để so sánh với biểu thức. + result: Giá trị sẽ trả về nếu biểu thức khớp với giá trị so sánh. + default Giá trị mà có thể trả về mặc định khi biểu thức và giá trị so sánh không khớp nhau. Vời default là tham số tùy chọn vì vậy hàm DECODE sẽ trả về giá trị null nếu biểu thức không khớp với giá trị so sánh. Tham số không bắt buộc
13	TRUNC()	TRUNC(number [, decimal_places]): Hàm TRUNC được dùng để thu gọn một số đến vị trí decimal_places Hàm này nhận vào 2 tham số: + number: giá trị số + decimal_places: vị trí muốn lấy sau dấu phẩy, đây là một số nguyên âm hoặc dương, mặc định là số 0.
14	LENGTH()	LENGTH(string_expression): Hàm LENGTH trả về độ dài của một chuỗi Hàm này nhận vào 1 tham số: + string_expression: giá trị dạng chuỗi cần được biết độ dài
15	lPAD()	LPAD(source_string, target_length [,pad_string]): Hàm LPAD được sử dụng để đệm vào bên trái của chuỗi với tập ký tự được xác định trong tham số của nó. Hàm này nhận vào 3 tham số: + source_string: chuỗi bạn muốn áp dụng + target_length: độ dài ký tự của kết quả trả về. Nếu bạn nhập vào số nhỏ hơn chuỗi gốc thì nó sẽ cắt ngắn chuỗi gốc, ngược lại nó sẽ bổ sung ký tự pad_string + pad_string là chuỗi cần chèn vào phía bên trái của chuỗi.
16	RPAD()	RPAD(source_string, target_length [,pad_string]): Hàm RPAD được sử dụng để đệm vào bên phải của chuỗi với tập ký tự được xác định trong tham số (pad_string) của nó. Hàm này nhận vào 3 tham số: + source_string: chuỗi bạn muốn áp dụng + target_length: độ dài ký tự của kết quả trả về. Nếu bạn nhập vào số nhỏ hơn chuỗi gốc thì nó sẽ cắt ngắn chuỗi gốc, ngược lại nó sẽ bổ sung ký tự pad_string + pad_string là chuỗi cần chèn vào phía bên phải của chuỗi.
17	TRIM()	TRIM([[LEADING | TRAILING | BOTH] trim_character FROM] trim_source): Hàm TRIM được sử dụng để xóa ký tự khoảng trắng hoặc ký tự bất kì tại vị trí đầu hoặc cuối hoặc cả hai Hàm này nhận vào 3 tham số: + LEADING, TRAILING, BOTH: Tham số 1 cho phép chỉ định vị trí sẽ được xóa: LEADING: đầu TRAILING: cuối BOTH: cả đầu và cuối

		+ trim_character: các ký tự sẽ được remove + trim_source: string cần được loại bỏ các ký tự không mong muốn
18	LTRIM()	LTRIM(trim_source,[set]): Hàm LTRIM được sử dụng để xóa các ký tự từ bên trái nằm trong tập các ký tự được chỉ định Hàm này nhận vào 2 tham số: + trim_source: string cần được loại bỏ các ký tự không mong muốn + set: các ký tự cần được loại bỏ khỏi trim_source
19	RTRIM()	RTRIM(trim_source,[set]): Hàm RTRIM được sử dụng để xóa các ký tự từ bên phải nằm trong tập các ký tự được chỉ định Hàm này nhận vào 2 tham số: + trim_source: string cần được loại bỏ các ký tự không mong muốn + set: các ký tự cần được loại bỏ khỏi trim_source
20	ROUND()	ROUND(number [, decimal_places]): Hàm ROUND được sử dụng để làm tròn số Hàm này nhận vào 2 tham số: + number: số cần được làm tròn + decimal_places: số thập phân cần được làm tròn. Nếu để trống sẽ mặc định loại bỏ phần thập phân
21	ADD_MONTHS()	ADD_MONTHS(date_expression, month): Hàm ADD_MONTHS được sử dụng để thêm tháng vào date value Hàm này nhận vào 2 tham số: + date_expression: giá trị Date + month: số tháng cần được thêm vào, nếu ta truyền vào số âm thì sẽ có nghĩa là trừ đi bấy nhiêu 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

SELECT * : Hiển thị tất cả các cột

FROM MDM_CUSTOMER A: Truy xuất từ bằng MDM_CUSTOMER và đổi tên thành A

WHERE A.CUST_LGL_ENG_NM LIKE '%_LOGISTICS%' ESCAPE '\' : Giá trị CUST_LGL_ENG_NM chứa string có dạng _LOGISTICS

ORDER BY A.CUST_LOCL_LANG_NM NULLS FIRST: Sắp xếp theo tăng dần theo CUST_LOCL_LANG_NM và giá trị NULL sẽ đứng trước giá trị non-null

Hiển thị tất cả các cột trong bảng mdm_customer theo điều kiện và sắp theo cột locl_lang_nm với các giá trị NULL đứng trước giá trị non-null.

B) ý nghĩa của việc dùng ESCAPSE

Để chỉ định ký tự sau ESCAPSE là ký tự thông thường.

Để rõ nghĩa hơn ta xét ví điều kiện của ví dụ trên'%_LOGISTICS%', nếu không dùng ESCAPSE thì _ ở trên sẽ có nghĩa là cần ít nhất 1 ký tự trước chữ LOGISTICS, nhưng ở đây ý ta là dấu gạch dưới (_) chứ không phải là placeholder cho ký tự nào đó. Chính vì thế ta cần sử dụng ESCAPSE

C) Ý nghĩa của việc dung Nulls First.

Sắp xếp các giá trị NULL trước các giá trị non-null

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

Ta không nhất thiết phải sử dụng alias ở mọi trường hợp. Thông thường sẽ sử dụng theo 2 trường hợp sau:

+ Tính dễ đọc, dễ viết. Vì tên cột thường có dạng viết tắt nên đôi khi sẽ gây khó hiểu cho người đọc nên ta cần alias thành tên dễ hiểu hơn, nhiều ý nghĩa hơn

+ Bắt buộc phải dùng. Ví dụ khi sử dụng self-join. Giả sử ta có bảng Employee có cột managerId, mà managerId lại tham chiếu đến id của bảng Employee, lúc này thì self-join ta cần phải alias tên của 1 trong 2 bảng đi, để khi ta gọi tên cột như id hay managerId thì Oracle mới hiểu đang nói đến bảng nào

Câu 3:

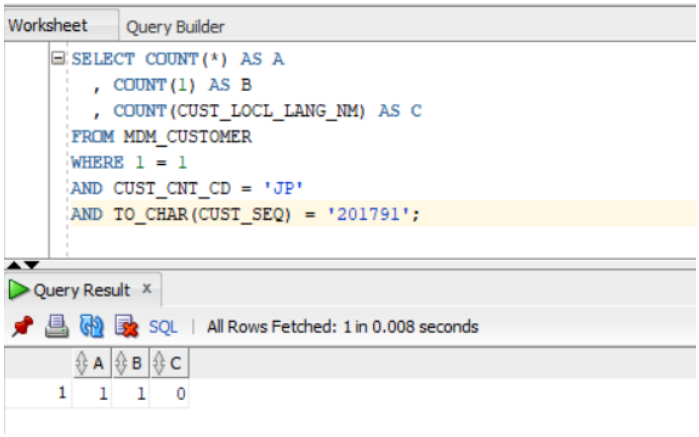
```
SELECT *
FROM MDM_CUSTOMER
WHERE 1 = 1
AND CUST_CNT_CD = 'JP'
AND TO_CHAR(CUST_SEQ) = '201791'
```

Theo bạn câu trên cách dùng TO_CHAR(CUST_SEQ) = '201791' có hợp lý không, tại sao?

Không hợp lý vì không tận dụng được index của database. Tức là trong trường hợp này truy vấn của ta sẽ phải lặp qua các cột để tìm cột CUST_SEQ sau đó dùng hàm TO_CHAR lên cột CUST_SEQ.

Thông thường khi nhận vào tên của một cột thì truy vấn của ta sẽ thông qua index để trỏ đến cột cần tìm. Index như mục lục của sách thay vì tìm từng trang ta có thể biết chính xác số trang (index) cần đến

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 1 cho mọi hàng trong bảng rồi đếm tổng số hàng trong bảng, bao gồm các giá trị null

⇒ COUNT(any_string): gán any_string cho mọi hàng trong bảng rồi đếm tổng (COUNT(*) thì gán * và COUNT(1) thì gán 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 cột CUST_LOCL_LANG_NM chỉ chứa 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: SUM sẽ bỏ qua giá trị NULL nên không cần phải dùng NVL ở mỗi bước rồi mới SUM ⇒ SUM chạy 1 lần và NVL chạy 1 lần

Cách 2: Trước khi SUM ta thực hiện NVL ⇒ SUM chạy 1 lần, NVL chạy n lần (với n là số dòng)

⇒ Cách 1 tốt hơn cách 2

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 nhanh hơn vì ta chỉ thực hiện 1 lần duyệt từ đầu bảng đến cuối bảng thay vì như cách 1 thực hiện 2 lần duyệt

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 1: Dùng IN khi kết quả truy vấn phụ nhỏ. IN thì sẽ so sánh từng phần tử trong danh sách (nhiều phép OR gộp lại)

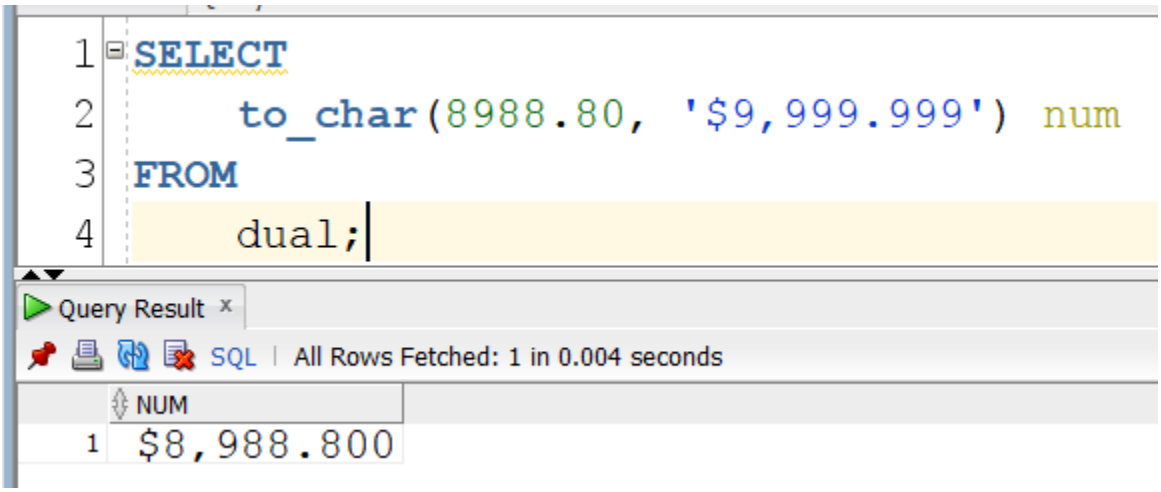
Cách 2: Dùng EXISTS khi kết quả truy vấn phụ lớn. EXISTS thì kiểm ra xem truy vấn con có trả về kết quả hay không (số dòng có lớn hơn 1 hay không)

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 bảng TB_ORD đã được bỏ bớt các dòng dữ liệu vi phạm và khi đó ta thực hiện subquery ở Select sẽ ít lần. Nếu ta JOIN từ 2 bảng như cách 1 thì trường hợp 2 bảng lớn sẽ khiến cho chi phí khi JOIN vô cùng lớn mà bảng sau khi JOIN lại còn tồn tại vi phạm, ta lại cần where để loại bỏ

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



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')      num1,
    to_char(trunc(820988.80, - 3), '$999,999.999') num2
FROM
    dual;
```

Query Result | SQL | All Rows Fetched: 1 in 0.004 seconds

	NUM1	NUM2
1	$8,000.000	$820,000.000

Câu 11: Cho cấu SQL và kết quả như sau:

```
SELECT substr(TO_CHAR(98765, 'fm00000'),1,3) as A
, substr(TO_CHAR(98765, '00000'),1,3)  as B from dual
```

Query Result | SQL | All Rows Fetched: 1 in 0.004 seconds

	A	B
1	987	98

Như hình trên cả 2 A và B điêu substr từ 1, đến 3 tại sao kết quả lại khác nhau.

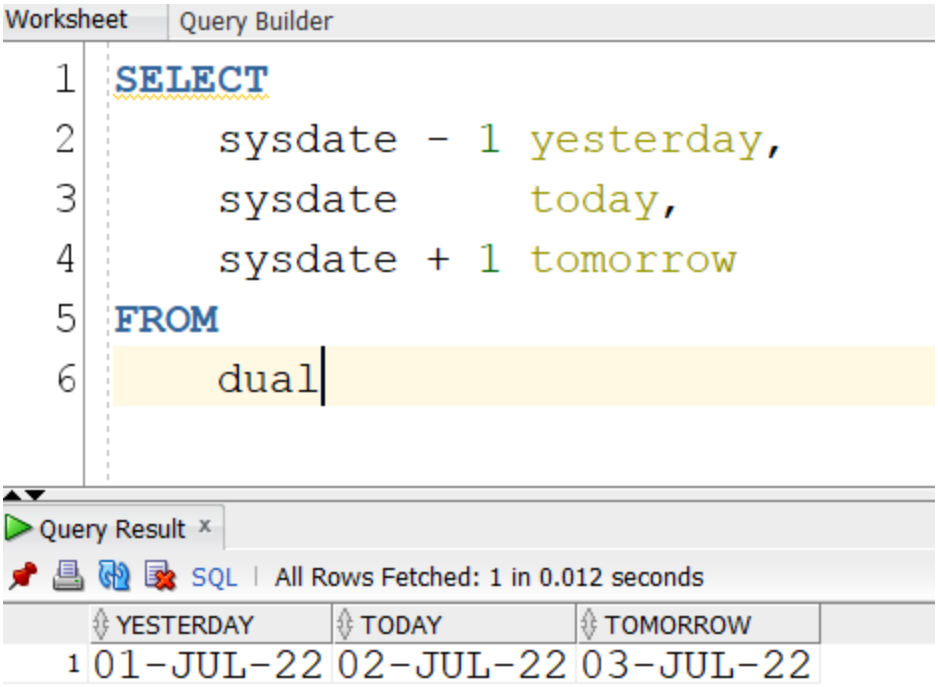
Nếu không có FM thì sẽ xuất hiện khoảng trắng (khoảng trắng dùng để thể hiện dấu +/-) ở kết quả trả về. Bằng chứng là là length(to_char(98765, 'fm00000')) là 5 và length(to_char(98765, '00000')) là 6. Kết quả như hình bên dưới. Do đó khi dùng hàm substr() ta thu được 2 kết quả khác nhau.

```
SELECT
    length(to_char(98765, 'fm00000')) has_FM,
    length(to_char(98765, '00000')) without_FM
FROM
    dual
```

Query Result | SQL | All Rows Fetched: 1 in 0.012 seconds

	HAS_FM	WITHOUT_FM
1	5	6

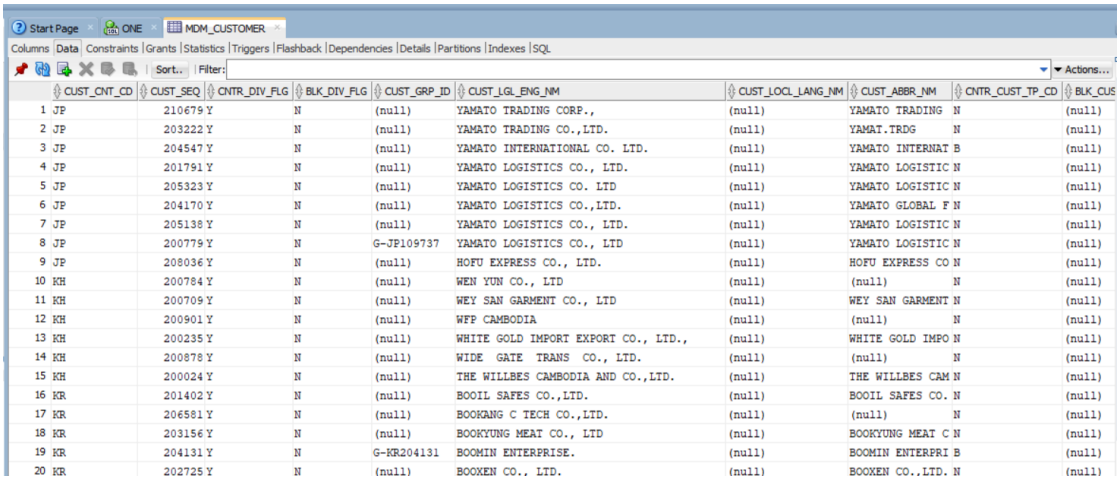
Câu 12: Viết Câu SQL xuất ra, Ngày hiện tại, này hôm qua, ngày mai



Câu 13: Ta có table (TB_ORD), yêu cầu viết câu SQL để generate ORD_NO có độ 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_dttm), 9, 4) + 1, 4, '0'), '0001') AS ord_dttm
FROM
    tb_ord
WHERE
    ord_dttm LIKE to_char(sysdate, 'YYYYMMDD') || '%'
```

Câu 14: Ta có table (MDM_CUSTOMER) và dữ liệu như bên dưới



Start Page | ONE | MDM_CUSTOMER

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter: | Actions...

	CUST_CNT_CD	CUST_SEQ	CNTR_DIV_FLG	BLK_DIV_FLG	CUST_GRP_ID	CUST_LGL_ENG_NM	CUST_LOCL_LANG_NM	CUST_ABBR_NM	CNTR_CUST_TP_CD	BLK_CUS
1	JP	210679	Y	N	(null)	YAMATO TRADING CORP.,	(null)	YAMATO TRADING	N	(null)
2	JP	203222	Y	N	(null)	YAMATO TRADING CO.,LTD.	(null)	YAMAT.TRDG	N	(null)
3	JP	204547	Y	N	(null)	YAMATO INTERNATIONAL CO. LTD.	(null)	YAMATO INTERNAT	B	(null)
4	JP	201791	Y	N	(null)	YAMATO LOGISTICS CO., LTD.	(null)	YAMATO LOGISTIC	N	(null)
5	JP	205323	Y	N	(null)	YAMATO LOGISTICS CO. LTD	(null)	YAMATO LOGISTIC	N	(null)
6	JP	204170	Y	N	(null)	YAMATO LOGISTICS CO.,LTD.	(null)	YAMATO GLOBAL F	N	(null)
7	JP	205138	Y	N	(null)	YAMATO LOGISTICS CO., LTD.	(null)	YAMATO LOGISTIC	N	(null)
8	JP	200779	Y	N	G-JP109737	YAMATO LOGISTICS CO., LTD	(null)	YAMATO LOGISTIC	N	(null)
9	JP	208036	Y	N	(null)	HOFU EXPRESS CO., LTD.	(null)	HOFU EXPRESS CO	N	(null)
10	KH	200784	Y	N	(null)	WEN YUN CO., LTD	(null)	(null)	N	(null)
11	KH	200709	Y	N	(null)	WEY SAN GARMENT CO., LTD	(null)	WEY SAN GARMENT	N	(null)
12	KH	200901	Y	N	(null)	WFP CAMBODIA	(null)	(null)	N	(null)
13	KH	200235	Y	N	(null)	WHITE GOLD IMPORT EXPORT CO., LTD.,	(null)	WHITE GOLD IMPO	N	(null)
14	KH	200878	Y	N	(null)	WIDE GATE TRANS CO., LTD.	(null)	(null)	N	(null)
15	KH	200024	Y	N	(null)	THE WILLBES CAMBODIA AND CO.,LTD.	(null)	THE WILLBES CAM	N	(null)
16	KR	201402	Y	N	(null)	BOOIL SAFES CO.,LTD.	(null)	BOOIL SAFES CO.	N	(null)
17	KR	206581	Y	N	(null)	BOOKANG C TECH CO.,LTD.	(null)	(null)	N	(null)
18	KR	203156	Y	N	(null)	BOOKYUNG MEAT CO., LTD	(null)	BOOKYUNG MEAT C	N	(null)
19	KR	204131	Y	N	G-KR204131	BOOMIN ENTERPRISE.	(null)	BOOMIN ENTERPRI	B	(null)
20	KR	202725	Y	N	(null)	BOOXEN CO.. LTD.	(null)	BOOXEN CO..LTD.	N	(null)

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

```sql
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

```sql
WITH temp AS (
    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
    ORDER BY
        cust_grp_id,
        cust_grp_hrchy_cd
)
```

```sql
SELECT cust_grp_id
FROM
    (
        SELECT
            cust_grp_id,
            cust_grp_hrchy_cd,
            LEAD(cust_grp_id)
            OVER(
                ORDER BY
                    cust_grp_id
            ) next_cust_grp_id,
            LEAD(cust_grp_hrchy_cd)
            OVER(
                ORDER BY
                    cust_grp_id
            ) next_cust_grp_hrchy_cd
        FROM
            temp
        WHERE
            volumn = 2
    )
WHERE cust_grp_id = next_cust_grp_id
    AND cust_grp_hrchy_cd != 'C'
    AND next_cust_grp_id != 'C'
```

Câu 15: Ta có table (TB_PROD) và dữ liệu như bên dưới

	PROD_CD	PROD_NM	PROD_UNIT_AMT
1	00001	Prod 01	100
2	00002	Prod 02	300
3	00003	Prod 03	500
4	00004	Prod 04	300
5	00005	Prod 05	500
6	00006	Prod 06	700
7	00007	Prod 07	800
8	00008	Prod 08	100
9	00009	Prod 09	100
10	00010	Prod 10	100

Viết cấu SQL để suất ra kết 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:

	MAX_AMT	MAX_NAME	MIN_AMT	AVG
1	800	Prod 07	100	350

```sql
SELECT
    MAX(prod_unit_amt) max_amt,
    MIN(prod_unit_amt) min_amt,
    AVG(prod_unit_amt) avg_amt,
    MIN(prod_nm)
        KEEP(DENSE_RANK LAST ORDER BY prod_unit_amt) AS max_name
FROM
    tb_prod
WHERE
    prod_unit_amt IS NOT NULL
```

Câu 16: Ta có table (TB_ORD) và dữ liệu như bên dưới

	CUST_NO	ORD_NO	ORD_DTTM	PRO_CD
1	CUS01	ORD01	201910041500	00001
2	CUS01	ORD02	201909041230	00002
3	CUS01	ORD03	201909041540	00004
4	CUS01	ORD04	201910041520	00001
5	CUS01	ORD05	201908041500	00001
6	CUS02	ORD01	201910041500	00001
7	CUS02	ORD02	201909041230	00003
8	CUS02	ORD03	201909041540	00002
9	CUS02	ORD04	201910041520	00005
10	CUS02	ORD05	201908041500	00006
11	CUS03	ORD01	201910041500	00001
12	CUS03	ORD02	201909041230	00001
13	CUS03	ORD03	201909041540	00001
14	CUS03	ORD04	201910041520	00001
15	CUS03	ORD05	201908041500	00001
16	CUS04	ORD01	201910041500	00003
17	CUS04	ORD02	201909041230	00002
18	CUS04	ORD03	201909041540	00004
19	CUS04	ORD04	201910041520	00002
20	CUS04	ORD05	201908041500	00002

A) viết cấu SQL lấy ra top 3 sản phẩm đc bán nhiều nhất.

```sql
SELECT
    *
FROM
    (
        SELECT
            pro_cd,
            DENSE_RANK()
            OVER(
                ORDER BY
                    COUNT(*) DESC
            ) AS rank
        FROM
            tb_ord
        GROUP BY
            pro_cd
    ) temp
WHERE
    temp.rank <= 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:

CUS01	ORD02	201911130002	2
CUS02	ORD05	201908041500	6
CUS03	ORD05	201908041500	1
CUS04	ORD06	201911190001	4

```sql
SELECT
    cust_no,
    ord_dttm,
    ord_no,
    pro_cd
FROM
    (
        SELECT
            cust_no,
            ord_dttm,
            ord_no,
            pro_cd,
            ROW_NUMBER()
            OVER(PARTITION BY cust_no
                 ORDER BY
                     ord_dttm DESC
            ) rank
        FROM
            tb_ord
    ) temp
WHERE
    temp.rank = 1;
```

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:

	MON	PRO_CD	NVL(B.TOTAL,0)
1	201906	00001	0
2	201907	00001	0
3	201908	00001	2
4	201909	00001	2
5	201906	00002	0
6	201907	00002	0
7	201908	00002	1
8	201909	00002	2
9	201906	00003	0
10	201907	00003	0
11	201908	00003	0
12	201909	00003	1
13	201906	00004	0
14	201907	00004	0
15	201908	00004	0
16	201909	00004	2
17	201906	00005	0
18	201907	00005	0
19	201908	00005	0
20	201909	00005	0
21	201906	00006	0

```
WITH report AS (
    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
)
```

```sql
SELECT
    report.dt,
    nvl(pro_cd,'00001') pro_cd,
    nvl(ord.total, 0) total
FROM
         report left
    JOIN (
        SELECT
            pro_cd,
            substr(ord_dttm, 1, 6) AS ord_dttm,
            COUNT(*)                total
        FROM
            tb_ord
        WHERE
            pro_cd = '00001'
        GROUP BY
            pro_cd,
            substr(ord_dttm, 1, 6)
    ) ord ON report.dt = ord.ord_dttm;
```

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

	PRO_CD	MONTH	TOTAL	REMAIN
1	00001	201906	0	100
2	00001	201907	0	100
3	00001	201908	2	98
4	00001	201909	2	96

```
WITH report AS (
    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
)
```

```sql
SELECT
    report.dt,
    nvl(ord.total, 0) total,
    100 - nvl(SUM(ord.total)
              OVER(PARTITION BY ord.pro_cd
                   ORDER BY
                       report.dt
              ), 0)              AS remain
FROM
    report
    LEFT JOIN (
        SELECT
            pro_cd,
            substr(ord_dttm, 1, 6) AS ord_dttm,
            COUNT(*)               total
        FROM
            tb_ord
        WHERE
            pro_cd = '00001'
        GROUP BY
            pro_cd,
            substr(ord_dttm, 1, 6)
    ) ord ON report.dt = ord.ord_dttm
```