

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO MÔN HỌC

Cơ sở dữ liệu nâng cao

Sinh viên thực hiện: NGUYỄN ĐỨC MINH

Mã số sinh viên: 20152459

Lớp: Toán Tin 02 – K60

HÀ NỘI – 2019

MỤC LỤC

MỤC LỤC	1
PHẦN 1: CƠ SỞ DỮ LIỆU LỚN	4
1.1 Định nghĩa	4
1.2 Đặc trưng.....	4
1.3 Ứng dụng	5
PHẦN 2: CƠ SỞ DỮ PHÂN TÁN	6
2.1 Định nghĩa	6
2.2 Đánh giá ưu, nhược điểm của cơ sở dữ liệu phân tán.....	6
2.3 Kiến trúc của cơ sở dữ liệu phân tán.....	7
2.4 Thiết kế cơ sở dữ liệu phân tán.	8
PHẦN 3: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE	9
3.1 Lecture 1.....	9
3.2 Lecture 2.....	9
3.2.1 Practice 1.....	9
3.2.2 Practice 2.....	9
3.2.3 Practice 3.....	10
3.2.4 Practice 4.....	11
3.2.5 Practice 5.....	12
3.3 Lecture 3.....	12
3.3.1 Practice 1.....	12
3.3.2 Practice 2.....	13

3.3.3	Practice 3	13
3.3.4	Practice 4	14
3.4	Lecture 4	15
3.4.1	Practice 1	15
3.4.2	Practice 2	20
3.5	Lecture 5	23
3.5.1	Practice 1	23
3.5.2	Practice 2	29
3.6	Lecture 6	32
3.6.1	Practice 1	32
3.6.2	Practice 2	33
3.6.3	Practice 3	35
3.6.4	Practice 4	36
3.6.5	Practice 5	37
3.6.6	Practice 6	38
3.6.7	Practice 7	39
3.7	Lecture 7	40
3.8	Lecture 8	40
3.8.1	Practice 1	40
3.8.2	Practice 2	42
3.8.3	Practice 3	42
3.8.4	Practice 4	44

3.9	Lecture 9.....	45
PHẦN 4: BÀI TẬP KẾT THÚC MÔN HỌC		45
4.1	Bài 1	45
4.2	Bài 2	50
4.3	Bài 3	55
4.4	Bài 4	58
4.5	Bài 5	59
4.6	Bài 6	63
4.7	Bài 7	69
4.8	Bài 8	75
4.9	Bài 9	81
4.10	Bài 10	84

PHẦN 1: CƠ SỞ DỮ LIỆU LỚN

1.1 Định nghĩa

Big data nhìn chung liên quan đến các tập dữ liệu có khối lượng lớn và phức tạp đến mức các phần mềm xử lý dữ liệu truyền thống không có khả năng thu thập, quản lý và xử lý dữ liệu trong một khoảng thời gian hợp lý.

Những tập dữ liệu lớn này có thể bao gồm các dữ liệu có cấu trúc, không có cấu trúc và bán cấu trúc, mỗi tập có thể được khai thác để tìm hiểu insights.

Bao nhiêu dữ liệu để đủ gọi là "big" vẫn còn được tranh luận, nhưng nó có thể là các bội số của petabyte – và các dự án lớn nhất với phạm vi exabytes.

Big data thường đặc trưng với ba Vs:

- Khối lượng dữ liệu
- Nhiều loại dữ liệu đa dạng
- Vận tốc mà dữ liệu cần phải được xử lý và phân tích

Dữ liệu tạo thành các kho dữ liệu lớn có thể đến từ các nguồn bao gồm các trang web, phương tiện truyền thông xã hội, ứng dụng dành cho máy tính để bàn và ứng dụng trên thiết bị di động, các thí nghiệm khoa học, và các thiết bị cảm biến ngày càng tăng và các thiết bị khác trong internet (IoT).

Khái niệm big data đi kèm với các thành phần có liên quan cho phép các tổ chức đưa dữ liệu vào sử dụng thực tế và giải quyết một số vấn đề kinh doanh, bao gồm cơ sở hạ tầng IT cần thiết để hỗ trợ big data; các phân tích áp dụng với dữ liệu; công nghệ cần thiết cho các dự án big data; các bộ kỹ năng liên quan; và các trường hợp thực tế có ý nghĩa đối với big data.

1.2 Đặc trưng

Big Data được mô tả bởi những đặc trưng sau:

Volume (Dung lượng): Số lượng dữ liệu được tạo ra và lưu trữ. Kích thước của dữ liệu xác định giá trị và tiềm năng insight- và liệu nó có thể thực sự được coi là dữ liệu lớn hay không.

Variety (Tính đa dạng): Các dạng và kiểu của dữ liệu. Dữ liệu được thu thập từ nhiều nguồn khác nhau và các kiểu dữ liệu cũng có rất nhiều cấu trúc khác nhau.

Velocity (Vận tốc): Trong trường hợp này nghĩa là tốc độ các dữ liệu được tạo ra và xử lý để đáp ứng các nhu cầu và thách thức trên con đường tăng trưởng và phát triển.

Veracity (Tính xác thực): Chất lượng của dữ liệu thu được có thể khác nhau rất nhiều, ảnh hưởng đến sự phân tích chính xác.

Nhà máy và các hệ thống không thực-ảo có thể có một hệ thống 6C bao gồm:

- Kết nối (cảm biến và mạng)
- Đám mây (tính toán và dữ liệu theo yêu cầu)
- Nội dung ảo (mẫu và bộ nhớ)
- Nội dung / ngữ cảnh (ý nghĩa và tương quan)
- Cộng đồng (chia sẻ và cộng tác)
- Tùy chỉnh (cá nhân hoá và giá trị)

Dữ liệu phải được xử lý bằng các công cụ tiên tiến (phân tích và thuật toán) để cho ra các thông tin có ý nghĩa. Ví dụ, để quản lý một nhà máy phải xem xét cả hai vấn đề hữu hình và vô hình với các thành phần khác nhau. Các thuật toán tạo thông tin phải phát hiện và giải quyết các vấn đề không nhìn thấy được như sự xuống cấp của máy, mài mòn linh kiện, vv. trong nhà máy.

1.3 Ứng dụng

Cơ sở dữ liệu lớn được ứng dụng trong các lĩnh vực như:

- Phân tích URL.
- Phân tích mạng xã hội.
- An ninh mạng.
- Phân tích thời gian thực.

PHẦN 2: CƠ SỞ DỮ PHÂN TÁN

2.1 Định nghĩa

Cơ sở dữ liệu phân tán (Distributed Database - DDB): là một tuyến tập dữ liệu có quan hệ logic với nhau, được phân bố trên các máy tính của một mạng máy tính.

Hệ quản trị CSDL phân tán: Hệ thống phần mềm cho phép quản lý CSDL phân tán và cung cấp các cơ chế truy xuất đảm bảo tính trong suốt (transparent) về sự phân tán đối với người dùng. Tính trong suốt của hệ phân tán được thể hiện như sau:

- Tính trong suốt phân đoạn (fragmentation transparency): mức cao nhất của tính trong suốt, người dùng cuối và người lập trình không cần biết về sự phân tán của cơ sở dữ liệu (không biết tên đoạn và vị trí phân bố các đoạn).
- Tính trong suốt định vị (location transparency): người dùng cuối hoặc người lập trình biết cơ sở dữ liệu phân chia thành các đoạn, tên của các đoạn nhưng không biết vị trí phân bố của các đoạn.
- Tính trong suốt đối với ánh xạ địa phương (local mapping transparency): người dùng cuối hoặc người lập trình biết tên các đoạn và vị trí của các đoạn.

Hệ cơ sở dữ liệu phân tán (Distributed Database System - DDBS) bao gồm: DDB và D_DBMS (Distributed Database Management System).

Hệ cơ sở dữ liệu phân tán được chia ra làm 2 loại ứng dụng:

- Ứng dụng cục bộ: được yêu cầu và thực hiện trên máy tính ở một nút trong hệ CSDLPT và chỉ liên quan đến CSDL tại nút đó.
- Ứng dụng toàn cục: yêu cầu truy nhập dữ liệu ở nhiều nút thông qua hệ thống truyền thông.

2.2 Đánh giá ưu, nhược điểm của cơ sở dữ liệu phân tán.

Ưu điểm:

- Phù hợp với cấu trúc của tổ chức.
- Nâng cao khả năng chia sẻ và tính tự trị địa phương.
- Nâng cao tính sẵn sàng.
- Nâng cao tính tin cậy.
- Nâng cao hiệu năng.
- Dễ mở rộng.

Hạn chế:

- Thiết kế CSDL phức tạp hơn.
- Khó điều khiển tính nhất quán dữ liệu.
- Khó phát hiện và khử lỗi.
- Giá thành.
- Bảo mật.
- Thiếu chuẩn mực.
- Thiếu kinh nghiệm.

2.3 Kiến trúc của cơ sở dữ liệu phân tán

Chưa có kiến trúc nào được công nhận tương đương với kiến trúc 3 mức ANSI/ARC.

Kiến trúc tham khảo:

- Tập các lược đồ ngoài toàn cục.
- Lược đồ khái niệm toàn cục.
- Lược đồ phân đoạn.
- Tập các lược đồ cho mỗi Hệ Cơ sở dữ liệu cục bộ tuân theo tiêu chuẩn 3 mức ANSI/ARC

Có thể khuyết một số mức, phụ thuộc vào các mức trong suốt được cung cấp.

Có thể là Cơ sở dữ liệu phân tán thuần nhất hoặc Cơ sở dữ liệu phân tán hỗn tạp.

Hệ quản trị cơ sở dữ liệu phân tán phải đảm bảo tính nguyên tố của các giao tác con.

Hệ quản trị cơ sở dữ liệu phân tán phải đảm bảo:

- Đồng bộ giữa các thao tác con với các giao tác cục bộ khác được thi hành tương tranh tại một nút.
- Đồng bộ các giao tác con với giao tác toàn cục đang thi hành đồng thời tại cùng nút hoặc khác nút.

Bộ quản lý giao tác tại mọi nút phải điều phối các giao tác cục bộ và toàn cục được khởi tạo tại nút đó.

Ba mô đun chính trong Hệ quản trị cơ sở dữ liệu phân tán:

- Phần mềm xử lý dữ liệu.

- Phần mềm ứng dụng.
- Phần mềm truyền thông.

2.4 Thiết kế cơ sở dữ liệu phân tán.

Quan hệ: CSDL gồm một tập các quan hệ, trong đó các khung nhìn là các tập con của một quan hệ. Các quan hệ cần nhiều chi phí truyền thông.

Đoạn mảnh (fragment) của quan hệ (sub-relations): Thi hành tương tranh nhiều giao tác truy xuất trên các phần khác nhau của quan hệ, cùng với đó là điều khiển ngữ nghĩa dữ liệu (đặc biệt là các ràng buộc toàn vẹn) khó hơn. Đặc biệt với những khung nhìn không được định nghĩa trên một mảnh/đoạn riêng lẻ thì sẽ đòi hỏi nhiều xử lý hơn

Phân đoạn dữ liệu là phân hoạch cơ sở dữ liệu thành các đoạn (fragments) sự phân đoạn cho phép phân chia một đối tượng đơn lẻ thành hai hay nhiều khúc hay mảnh. Thông tin phân đoạn dữ liệu được lưu trữ trong catalog dữ liệu phân tán. Phần mềm xử lý giao tác sẽ truy nhập thông tin ở đây để xử lý các yêu cầu của người dùng.

Các loại phân đoạn:

- Phân đoạn ngang: Dùng phép chọn để phân đoạn.
- Phân đoạn dọc: Dùng phép chiếu để phân đoạn.
- Phân đoạn hỗn hợp: Dùng cả phép chọn và chiếu để phân đoạn.
- Phân đoạn ngang suy diễn: Dùng phép nửa nối để phân đoạn.

Trong mọi loại phân đoạn, một đoạn được xác định bởi một biểu thức quan hệ, trong đó toán hạng là các quan hệ tổng thể, kết quả là các đoạn.

Các quy tắc phân đoạn:

- Điều kiện phân đoạn đầy đủ (Completeness condition)
- Điều kiện khôi phục phân đoạn (Reconstruction condition).
- Điều kiện không giao nhau (Disjointness condition) với phân đoạn ngang.

Mục tiêu của phân đoạn:

- Tăng tính cục bộ.
- Nâng cao độ tin cậy và tính sẵn sàng.
- Nâng cao hiệu năng.
- Cân bằng khả năng lưu trữ và chi phí.
- Tối thiểu chi phí truyền thông.

PHẦN 3: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ORACLE

3.1 Lecture 1

Phần này không có bài tập.

3.2 Lecture 2

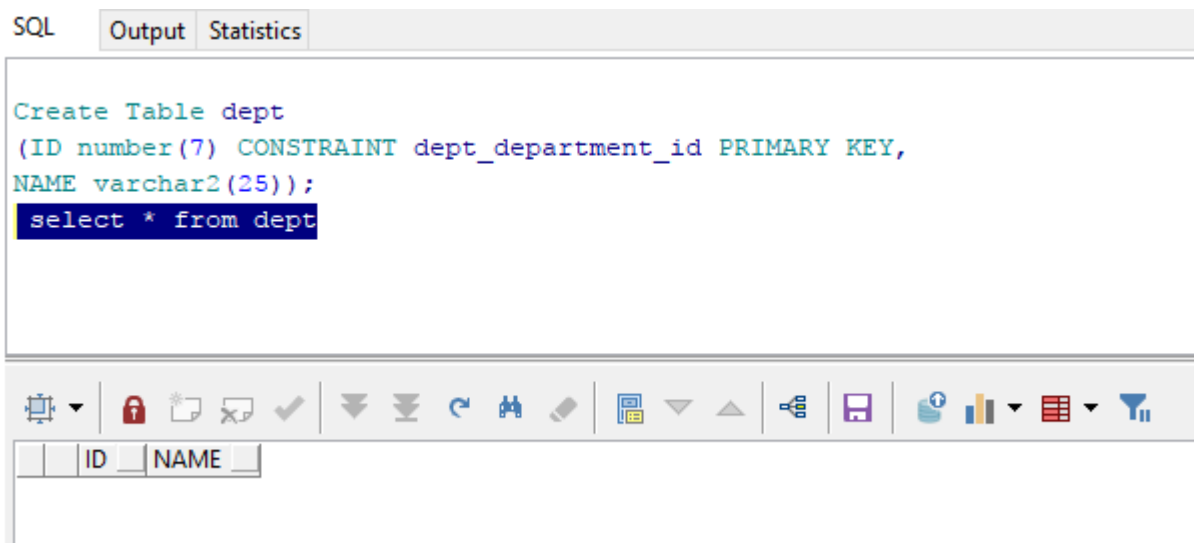
3.2.1 PRACTICE 1

Tạo bảng bảng DEPT trên sau theo biểu đồ thể hiện của bảng, thực hiện câu lệnh trong các script để tạo bảng, xác định bảng được tạo.

Câu lệnh SQL:

```
Create Table dept  
(ID number(7) CONSTRAINT dept_department_id PRIMARY KEY, NAME  
varchar2(25));
```

Kết quả:



SQL Output Statistics

```
Create Table dept  
(ID number(7) CONSTRAINT dept_department_id PRIMARY KEY,  
NAME varchar2(25));  
select * from dept
```

Toolbar icons: Undo, Redo, Cut, Copy, Paste, Find, Replace, Save, Print, Run, Stop, Refresh, Zoom In, Zoom Out, Full Screen, Help, etc.

ID	NAME
----	------

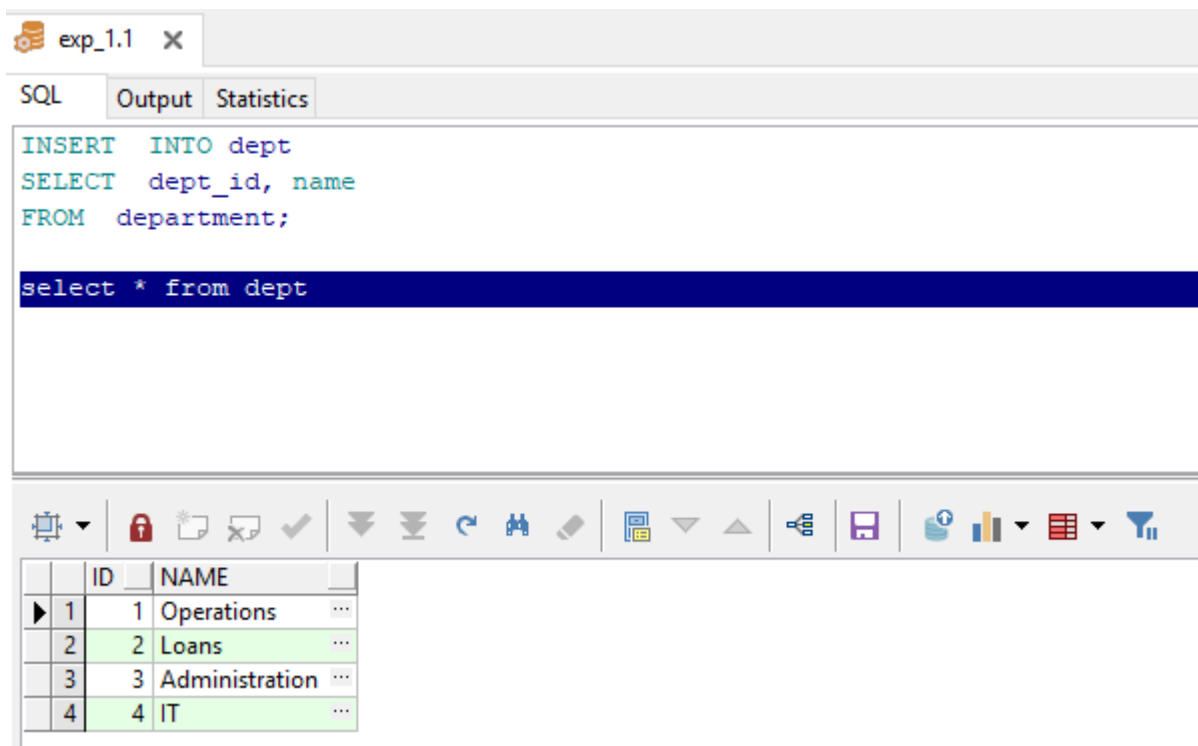
3.2.2 PRACTICE 2

Điền vào bảng DEPT với dữ liệu từ bảng DEPARTMENTS. Chỉ bao gồm các cột mà bạn cần.

Câu lệnh SQL:

```
INSERT INTO dept
SELECT dept_id, name
FROM department;
```

Kết quả:



The screenshot shows a SQL IDE window titled 'exp_1.1'. The 'SQL' tab is active, displaying the following SQL code:

```
INSERT INTO dept
SELECT dept_id, name
FROM department;
```

Below the code, a blue bar contains the text 'select * from dept'. The 'Output' tab is also visible. Below the tabs, a toolbar contains various icons for SQL operations. At the bottom, a table with 2 columns (ID, NAME) displays the data inserted into the 'dept' table:

ID	NAME
1	Operations
2	Loans
3	Administration
4	IT

3.2.3 PRACTICE 3

Tạo bảng EMP từ biểu đồ, xác nhận bảng đã được tạo.

Câu lệnh SQL:

```
CREATE TABLE emp
(id NUMBER(7) CONSTRAINT emp_employee_id PRIMARY KEY,
last_name VARCHAR2(25),
first_name VARCHAR2(25),
dept_id NUMBER(7) CONSTRAINT empdept_fk1 REFERENCES dept (id));
```

Kết quả:

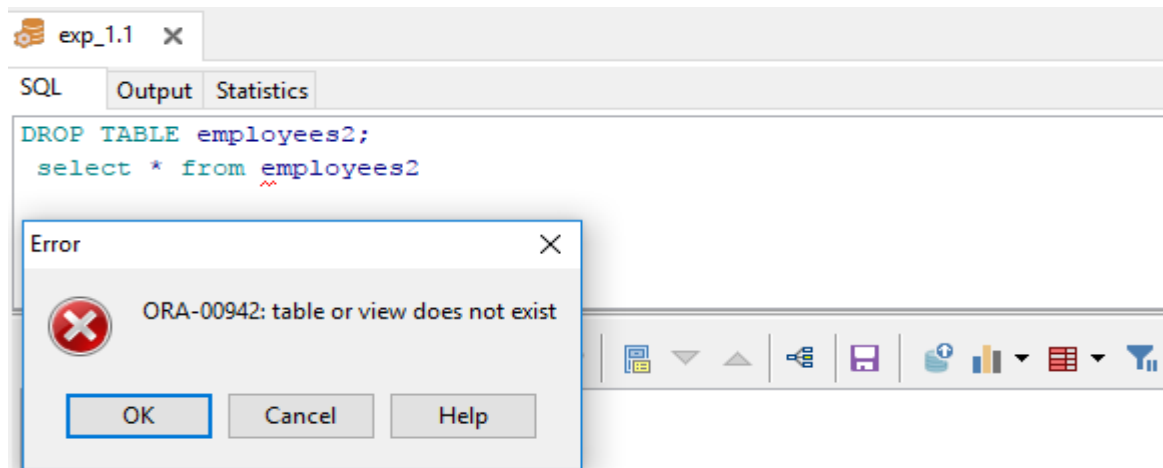
3.2.5 PRACTICE 5

Drop the EMP table.

Câu lệnh SQL:

```
DROP TABLE employees2;
```

Kết quả:



3.3 Lecture 3

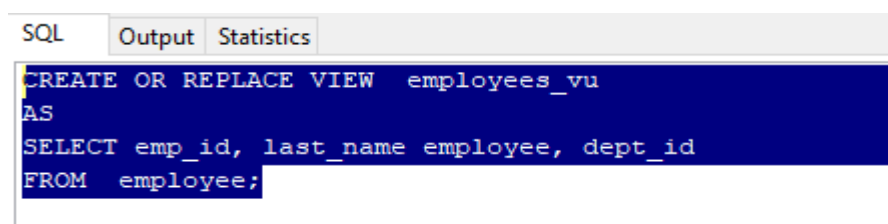
3.3.1 PRACTICE 1

Các nhân viên trong phòng nhân sự muốn ẩn một số dữ liệu trong bảng EMPLOYEES. Họ muốn có một View được gọi là EMPLOYEES_VU dựa trên employee numbers, employee names, and department numbers từ bảng EMPOLYEES. Họ muốn tiêu đề cho tên nhân viên là EMPLOYEE.

Câu lệnh SQL:

```
CREATE OR REPLACE VIEW employees_vu  
AS  
SELECT emp_id, last_name employee, dept_id  
FROM employee;
```

Kết quả: Tạo view thành công nên không thông báo lỗi.



3.3.2 PRACTICE 2

Kiểm tra view employees_vu

Câu lệnh SQL:

```
select * from employees_vu
```

Kết quả:

SQL			
Output			
Statistics			
<pre>CREATE OR REPLACE VIEW employees_vu AS SELECT emp_id, last_name employee, dept_id FROM employee; select * from employees_vu</pre>			
	EMP_ID	EMPLOYEE	DEPT_ID
▶ 1	1	Smith	3
2	2	Barker	3

3.3.3 PRACTICE 3

Sử dụng view EMPLOYEES_VU, viết truy vấn cho bộ phận nhân sự để hiển thị tất cả tên nhân viên và số phòng ban.

Câu lệnh SQL:

```
SELECT employee, dept_id
FROM employees_vu;
```

Kết quả:

SQL			
Output			
Statistics			
<pre>SELECT employee, dept_id FROM employees_vu;</pre>			
	EMPLOYEE	DEPT_ID	
▶ 1	Smith	3	
2	Barker	3	

3.3.4 PRACTICE 4

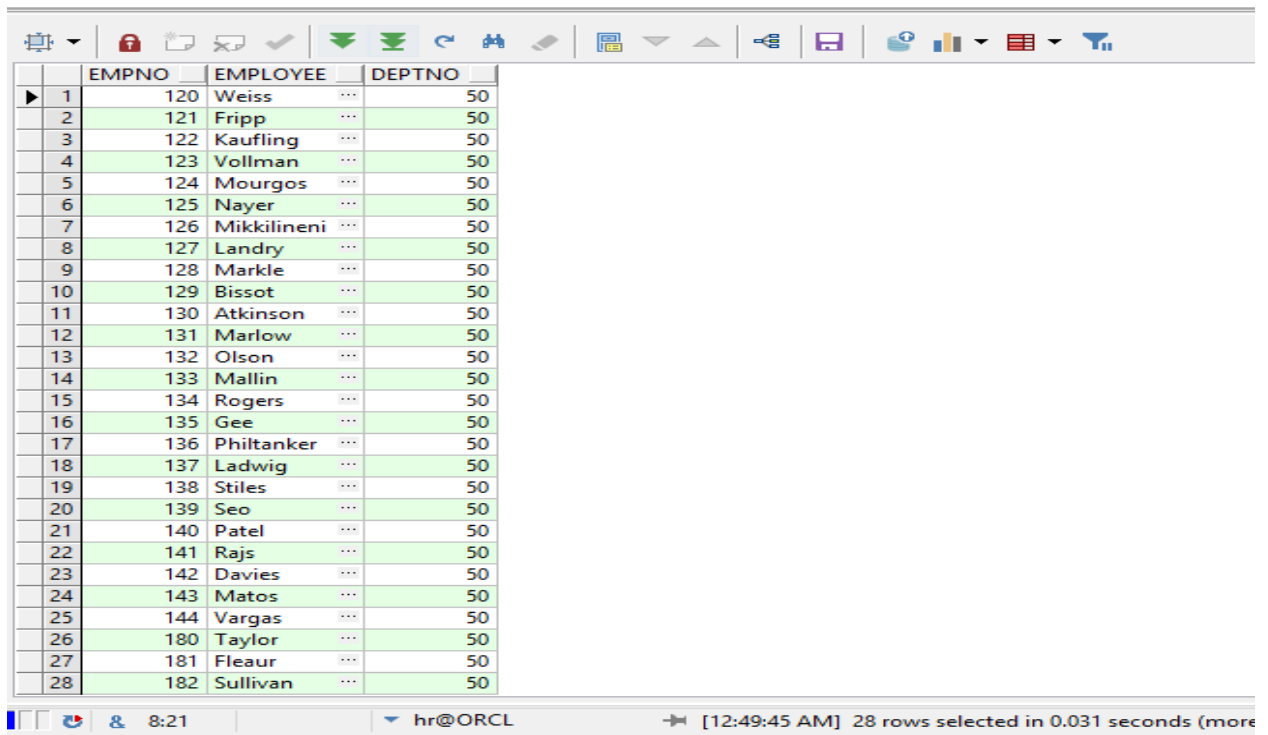
Department 50 cần truy cập vào dữ liệu nhân viên của mình. Tạo view có tên DEPT50 chứa số nhân viên, họ của nhân viên và số phòng ban cho tất cả nhân viên trong Department 50. Yêu cầu đặt tên các cột trong view là EMPNO, EMPLOYEE và DEPTNO. Vì mục đích bảo mật, không cho phép nhân viên được chỉ định lại cho bộ phận khác thông qua view. Kiểm tra view và hiển thị kết quả.

Câu lệnh SQL:

```
CREATE OR REPLACE VIEW dept50
AS
SELECT  employee_id  EMPNO, last_name  EMPLOYEE, department_id
DEPTNO
FROM employees
WHERE department_id = 50
WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

Kết quả:

```
select * from dept50|
```



	EMPNO	EMPLOYEE	DEPTNO
1	120	Weiss	50
2	121	Fripp	50
3	122	Kaufling	50
4	123	Vollman	50
5	124	Mourgos	50
6	125	Nayer	50
7	126	Mikkilineni	50
8	127	Landry	50
9	128	Markle	50
10	129	Bissot	50
11	130	Atkinson	50
12	131	Marlow	50
13	132	Olson	50
14	133	Mallin	50
15	134	Rogers	50
16	135	Gee	50
17	136	Philtanker	50
18	137	Ladwig	50
19	138	Stiles	50
20	139	Seo	50
21	140	Patel	50
22	141	Rajs	50
23	142	Davies	50
24	143	Matos	50
25	144	Vargas	50
26	180	Taylor	50
27	181	Fleaur	50
28	182	Sullivan	50

3.4 Lecture 4

3.4.1 PRACTICE 1

a.

Câu lệnh SQL:

```
SELECT last_name, job_id, salary AS Sal  
FROM employees;  
True/False
```

Kết quả:


```
SELECT last_name, job_id, salary AS Sal
FROM employees;
```

		LAST_NAME	JOB_ID	SAL
1	King	...	AD_PRES	24000.00
2	Kochhar	...	AD_VP	17000.00
3	De Haan	...	AD_VP	17000.00
4	Hunold	...	IT_PROG	9000.00
5	Ernst	...	IT_PROG	6000.00
6	Austin	...	IT_PROG	4800.00
7	Pataballa	...	IT_PROG	4800.00
8	Lorentz	...	IT_PROG	4200.00
9	Greenberg	...	FI_MGR	12008.00
10	Faviet	...	FI_ACCOUNT	9000.00
11	Chen	...	FI_ACCOUNT	8200.00
12	Sciarra	...	FI_ACCOUNT	7700.00

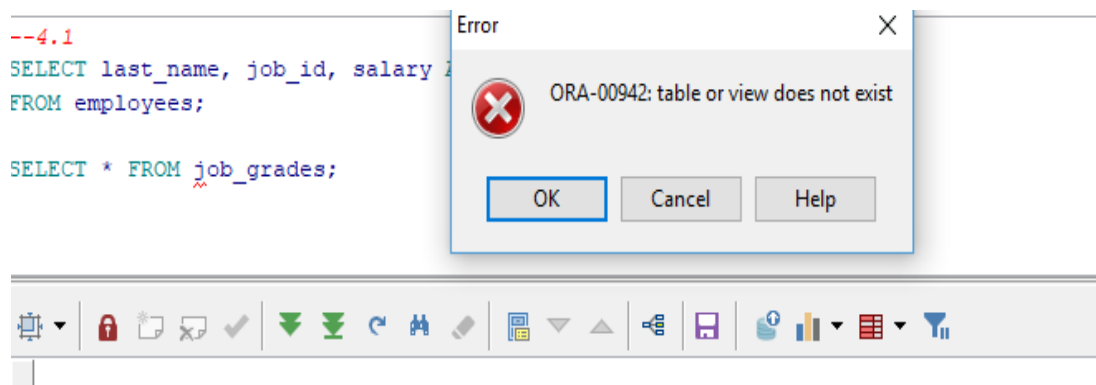
Câu truy vấn chính xác.

b. The following SELECT statement executes successfully:

```
SELECT * FROM job_grades;
```

True/False

Kết quả khi thực hiện truy vấn:



Câu truy vấn bị sai do chưa tồn tại bảng job_grades.

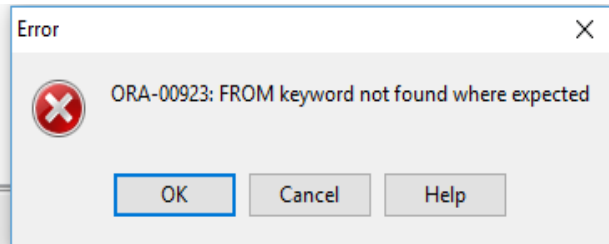
c. There are four coding errors in the following statement. Can you identify them?

```
SELECT employee_id, last_name
sal x 12 ANNUAL SALARY
```

FROM employees;

Kết quả thu được:

```
SELECT employee_id, last_name  
sal x 12 ANNUAL SALARY  
FROM employees;
```



Câu truy vấn bị sai

Sửa lại như sau

Câu lệnh SQL sửa:

```
SELECT employee_id, last_name,  
  
salary * 12 "ANNUAL SALARY"  
  
FROM employees;
```

Kết Quả:

```
SELECT employee_id, last_name,  
salary * 12 "ANNUAL SALARY"  
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	ANNUAL SALARY
1	100	King	288000
2	101	Kochhar	204000
3	102	De Haan	204000
4	103	Hunold	108000
5	104	Ernst	72000
6	105	Austin	57600
7	106	Pataballa	57600
8	107	Lorentz	50400
9	108	Greenberg	144096
10	109	Faviet	108000
11	110	Chen	98400
12	111	Sciarra	92400
13	112	Urman	93600
14	113	Popp	82800
15	114	Raphaely	132000
16	115	Khoo	37200
17	116	Baida	34800
18	117	Tobias	33600
19	118	Himuro	31200

Đã sửa chính xác.

- d. Bộ phận nhân sự cần một truy vấn để hiển thị tất cả các mã công việc không bị lặp từ bảng EMPLOYEES

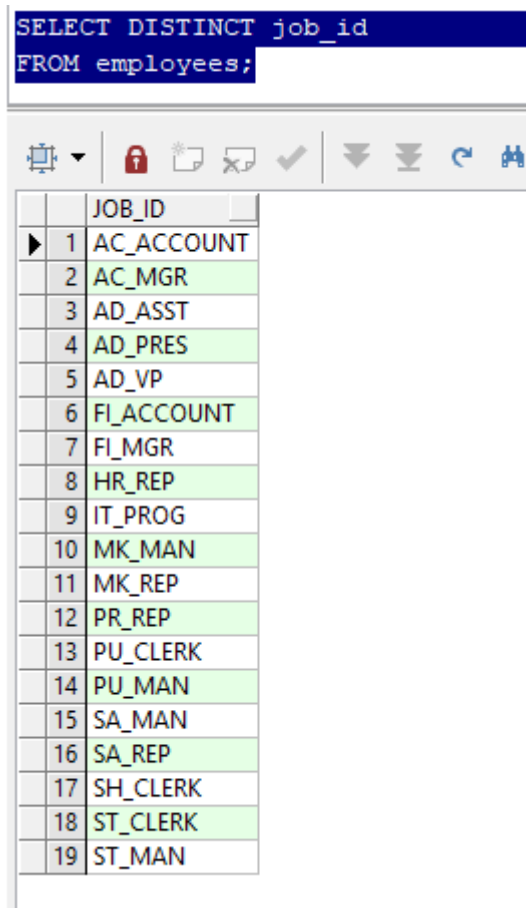
Câu lệnh SQL:

```
SELECT DISTINCT job_id
```

```
FROM employees;
```

Kết Quả:

```
SELECT DISTINCT job_id
FROM employees;
```



	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	HR_REP
9	IT_PROG
10	MK_MAN
11	MK_REP
12	PR_REP
13	PU_CLERK
14	PU_MAN
15	SA_MAN
16	SA_REP
17	SH_CLERK
18	ST_CLERK
19	ST_MAN

Nhận Xét: Sử dụng từ khóa Distinct trong câu truy vấn giúp loại bỏ các kết quả trùng nhau.

- e. Bộ phận nhân sự đã yêu cầu một báo cáo của tất cả nhân viên và jobs_ID của họ. Hiển thị last_name cùng với jobs_ID (được phân tách bằng dấu phẩy và dấu cách) và đặt tên cho cột Employee and Title

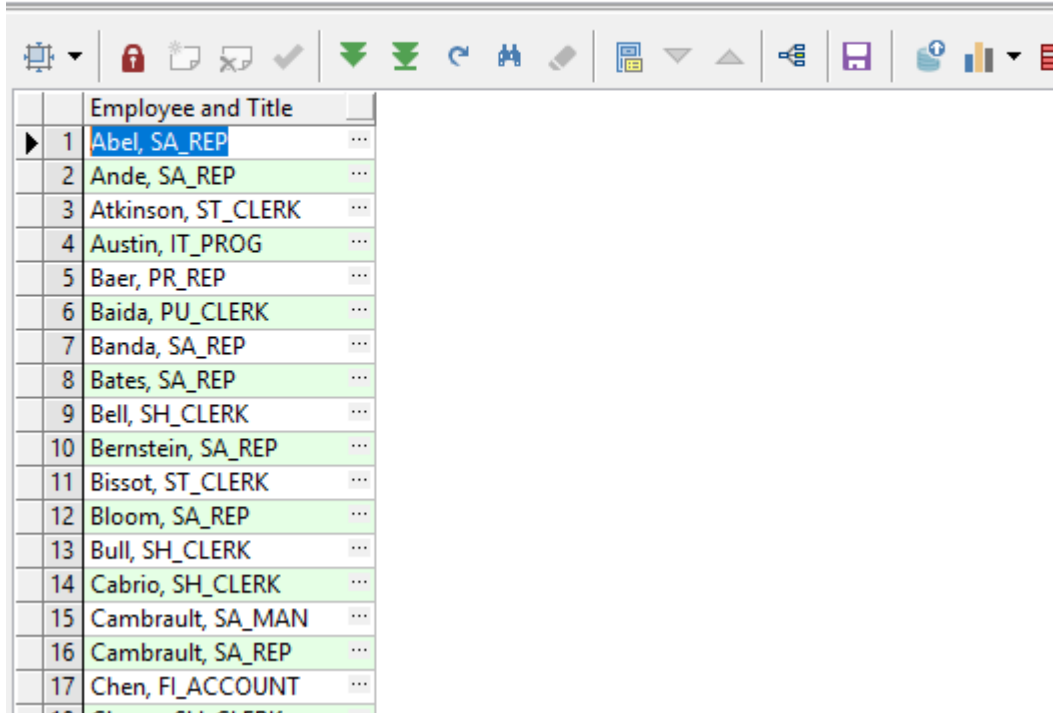
Câu lệnh SQL:

```
SELECT last_name || ', ' || job_id AS "Employee and Title"
```

```
FROM employees;
```

Kết Quả:

```
SELECT last_name || ', ' || job_id AS "Employee and Title"
FROM employees;
```



	Employee and Title
1	Abel, SA_REP
2	Ande, SA_REP
3	Atkinson, ST_CLERK
4	Austin, IT_PROG
5	Baer, PR_REP
6	Baida, PU_CLERK
7	Banda, SA_REP
8	Bates, SA_REP
9	Bell, SH_CLERK
10	Bernstein, SA_REP
11	Bissot, ST_CLERK
12	Bloom, SA_REP
13	Bull, SH_CLERK
14	Cabrio, SH_CLERK
15	Cambrault, SA_MAN
16	Cambrault, SA_REP
17	Chen, FI_ACCOUNT
18	Chen, SH_CLERK

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.4.2 PRACTICE 2

- a. Các phòng nhân sự cần các tìm nhân viên lương cao và lương thấp. Hiển thị tên last_name và salary của nhân viên kiếm được từ 5.000\$ đến 12.000\$ và ở bộ phận 20 hoặc 50. Gán nhãn tương ứng cho các cột tương ứng là Employee and Monthly Salary

Câu lệnh SQL:

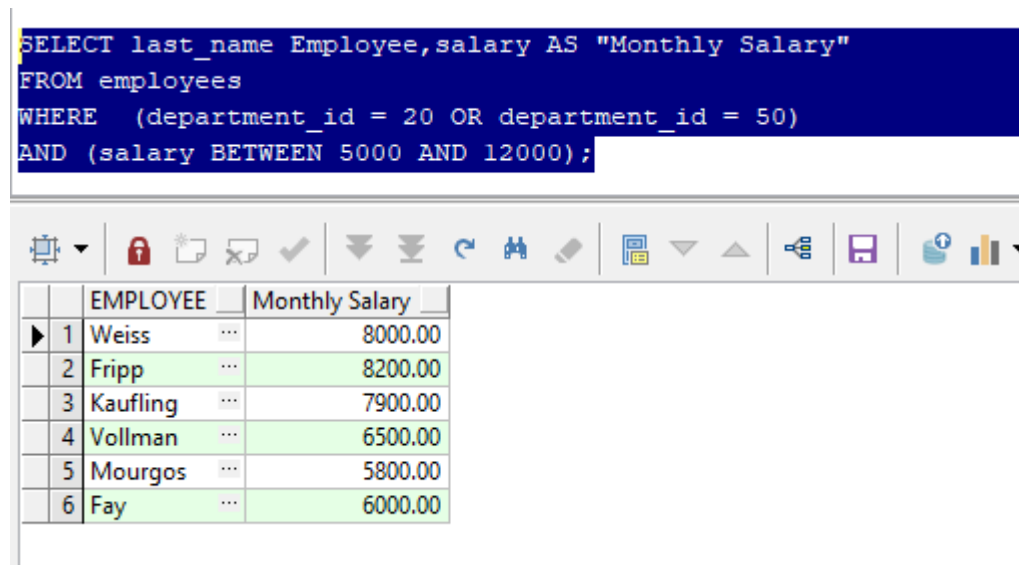
```
SELECT last_name Employee,salary AS "Monthly Salary"
```

```
FROM employees
```

```
WHERE (department_id = 20 OR department_id = 50)
```

```
AND (salary BETWEEN 5000 AND 12000);
```

Kết Quả:



```
SELECT last_name Employee,salary AS "Monthly Salary"
FROM employees
WHERE (department_id = 20 OR department_id = 50)
AND (salary BETWEEN 5000 AND 12000);
```

	EMPLOYEE	Monthly Salary
1	Weiss	8000.00
2	Fripp	8200.00
3	Kaufling	7900.00
4	Vollman	6500.00
5	Mourgos	5800.00
6	Fay	6000.00

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- b. Tạo một báo cáo để hiển thị last name, salary và commission của tất cả các nhân viên. Sắp xếp dữ liệu theo thứ tự giảm dần của salary và comission.

Câu lệnh SQL:

```
SELECT last_name, salary, commission_pct
```

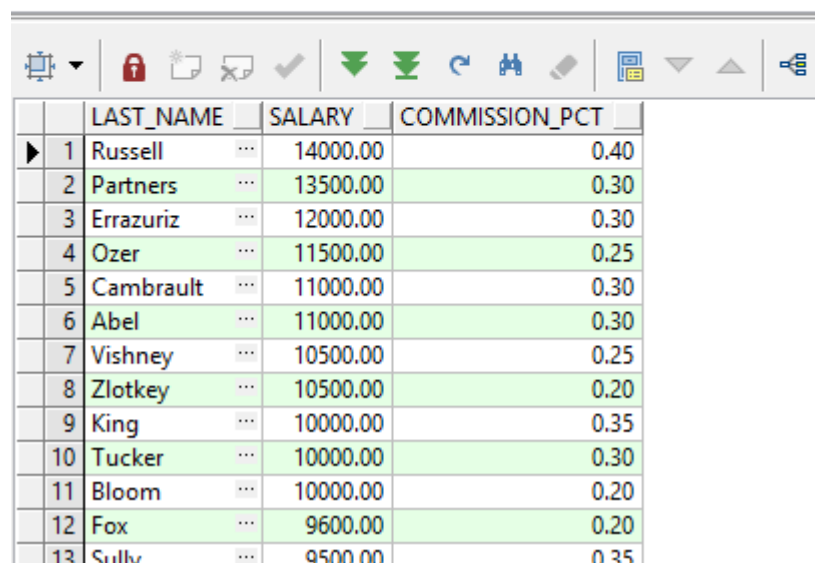
FROM employees

WHERE commission_pct IS NOT NULL

ORDER BY salary DESC, commission_pct DESC;

Kết Quả:

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY salary DESC, commission_pct DESC;
```



		LAST_NAME	SALARY	COMMISSION_PCT
▶	1	Russell	14000.00	0.40
	2	Partners	13500.00	0.30
	3	Errazuriz	12000.00	0.30
	4	Ozer	11500.00	0.25
	5	Cambrault	11000.00	0.30
	6	Abel	11000.00	0.30
	7	Vishney	10500.00	0.25
	8	Zlotkey	10500.00	0.20
	9	King	10000.00	0.35
	10	Tucker	10000.00	0.30
	11	Bloom	10000.00	0.20
	12	Fox	9600.00	0.20
	13	Sullivan	9500.00	0.35

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

c. Hiển thị last name của tất cả các nhân viên có cả chữ cái a và e trong họ của họ

Câu lệnh SQL:

SELECT last_name

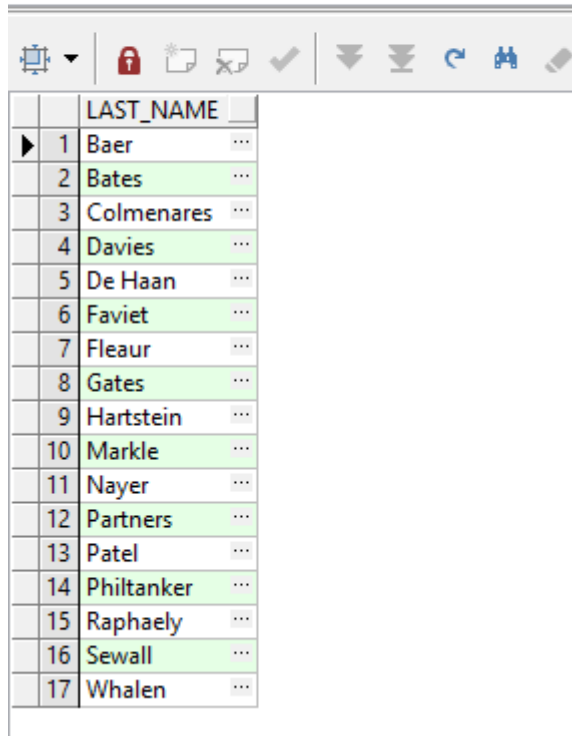
FROM employees

WHERE last_name LIKE '%a%'

AND last_name LIKE '%e%';

Kết Quả:

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%'
AND last_name LIKE '%e%';
```



The screenshot shows a database query result in a table. The table has a single column labeled 'LAST_NAME'. The results are listed in rows, numbered 1 through 17. The last names are: Baer, Bates, Colmenares, Davies, De Haan, Faviat, Fleaur, Gates, Hartstein, Markle, Nayer, Partners, Patel, Philtanker, Raphaely, Sewall, and Whalen. The rows for Bates, Davies, Faviat, Gates, Markle, Partners, Philtanker, and Sewall are highlighted in green.

	LAST_NAME
1	Baer
2	Bates
3	Colmenares
4	Davies
5	De Haan
6	Faviat
7	Fleaur
8	Gates
9	Hartstein
10	Markle
11	Nayer
12	Partners
13	Patel
14	Philtanker
15	Raphaely
16	Sewall
17	Whalen

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- d. Hiển thị last name, job, and salary cho tất cả nhân viên có công việc là SA_REP hoặc ST_CLERK và có mức lương không bằng 2.500 \$, 3.500 \$ hoặc 7.000 \$.

Câu lệnh SQL:

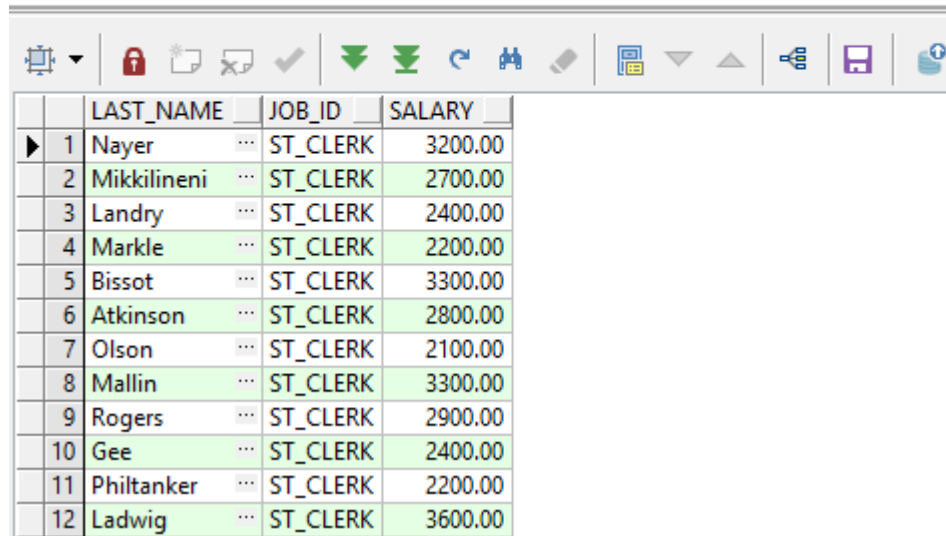
```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP' OR job_id = 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
```

Kết Quả:

```

SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP' OR job_id = 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
|

```



	LAST_NAME	JOB_ID	SALARY
1	Nayer	ST_CLERK	3200.00
2	Mikkilineni	ST_CLERK	2700.00
3	Landry	ST_CLERK	2400.00
4	Markle	ST_CLERK	2200.00
5	Bissot	ST_CLERK	3300.00
6	Atkinson	ST_CLERK	2800.00
7	Olson	ST_CLERK	2100.00
8	Mallin	ST_CLERK	3300.00
9	Rogers	ST_CLERK	2900.00
10	Gee	ST_CLERK	2400.00
11	Philtanker	ST_CLERK	2200.00
12	Ladwig	ST_CLERK	3600.00

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.5 Lecture 5

3.5.1 PRACTICE 1

- Viết một truy vấn hiển thị last name (với chữ hoa đầu tiên và tất cả chữ cái khác chữ thường) và độ dài của last name cho tất cả nhân viên có tên bắt đầu bằng các chữ cái J, A hoặc M. Đặt cho mỗi cột một nhãn thích hợp. Sắp xếp kết quả theo last_name.

Câu lệnh SQL:

```

SELECT INITCAP(last_name) last_name_initcap, LENGTH(last_name) length_name
FROM employees
WHERE last_name LIKE 'J%'
      OR last_name LIKE 'A%'
      OR last_name LIKE 'M%'
ORDER BY last_name DESC;

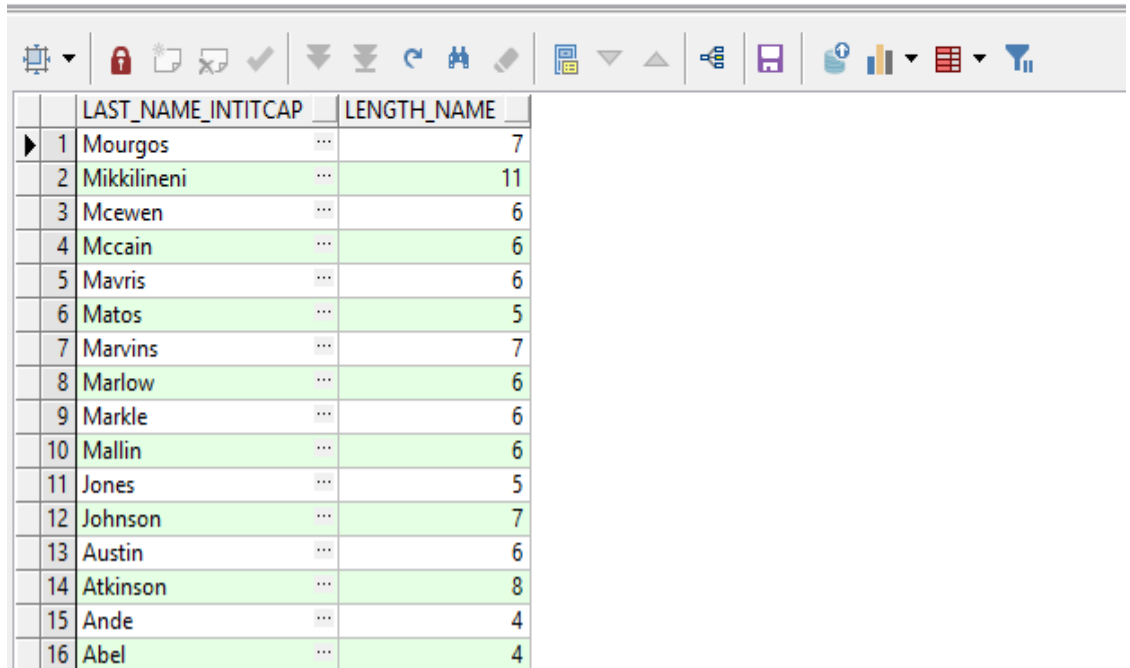
```

Kết Quả:


```

SELECT INITCAP(last_name) last_name_initcap, LENGTH(last_name) length_name
FROM employees
WHERE last_name LIKE 'J%'
      OR last_name LIKE 'A%'
      OR last_name LIKE 'M%'
ORDER BY last_name DESC;

```



	LAST_NAME_INITCAP	LENGTH_NAME
1	Mourgos	7
2	Mikkilineni	11
3	Mcewen	6
4	Mccain	6
5	Mavris	6
6	Matos	5
7	Marvins	7
8	Marlow	6
9	Markle	6
10	Mallin	6
11	Jones	5
12	Johnson	7
13	Austin	6
14	Atkinson	8
15	Ande	4
16	Abel	4

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- b. Bộ phận nhân sự muốn tìm thời gian làm việc của mỗi nhân viên. Đối với mỗi nhân viên, hiển thị last name và tính số tháng giữa ngày hôm nay và ngày mà nhân viên được thuê. Dán nhãn cột MONTHS_WORKED. Sắp xếp kết quả của bạn theo số tháng làm việc. Làm tròn số tháng cho đến số nguyên gần nhất

Câu lệnh SQL:

```

SELECT last_name, ROUND(MONTHS_BETWEEN(sysdate, hire_date)) "Months_Worked"
FROM employees
ORDER BY "Months_Worked";

```

Kết Quả:

```
SELECT last_name, ROUND(MONTHS_BETWEEN(sysdate, hire_date)) "Months_Worked"
FROM employees
ORDER BY "Months_Worked";
```

	LAST_NAME	Months_Worked
1	Banda	128
2	Kumar	128
3	Ande	129
4	Markle	130
5	Lee	130
6	Philtanker	131
7	Zlotkey	131
8	Marvins	131
9	Geoni	131
10	Johnson	132
11	Perkins	132
12	Grant	132
13	Popp	133
14	Gee	133
15	Tuvault	133
16	Mourgos	134
17	Cambrault	135

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- c. Hiển thị mỗi nhân viên tên, họ, ngày thuê và ngày xem xét lương, là ngày thứ Hai đầu tiên sau sáu tháng phục vụ. Dán nhãn cột REVIEW. Định dạng ngày xuất hiện theo định dạng tương tự như "Monday, the Thirty-First of July, 2000".

Câu lệnh SQL:

```
SELECT last_name, hire_date,
```

```
TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'), "Monday, the" fmddspth
"of" Month", " YYYY") "REVIEW"
```

```
FROM employees;
```

Kết Quả:

```
SELECT last_name, hire_date,
TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'), '"Monday, the" fmdspth "of" Month," YYYY') "REVIEW"
FROM employees;
```

	LAST_NAME	HIRE_DATE	REVIEW
1	King	6/17/2003	Monday, the twenty-second of December, 2003
2	Kochhar	9/21/2005	Monday, the twenty-seventh of March, 2006
3	De Haan	1/13/2001	Monday, the sixteenth of July, 2001
4	Hunold	1/3/2006	Monday, the tenth of July, 2006
5	Ernst	5/21/2007	Monday, the twenty-sixth of November, 2007
6	Austin	6/25/2005	Monday, the twenty-sixth of December, 2005
7	Pataballa	2/5/2006	Monday, the seventh of August, 2006
8	Lorentz	2/7/2007	Monday, the thirteenth of August, 2007
9	Greenberg	8/17/2002	Monday, the twenty-fourth of February, 2003
10	Faviet	8/16/2002	Monday, the seventeenth of February, 2003
11	Chen	9/28/2005	Monday, the third of April, 2006
12	Sciarra	9/30/2005	Monday, the third of April, 2006
13	Urman	3/7/2006	Monday, the eleventh of September, 2006
14	Popp	12/7/2007	Monday, the ninth of June, 2008
15	Raphaely	12/7/2002	Monday, the ninth of June, 2003
16	Khoo	5/18/2003	Monday, the twenty-fourth of November, 2003
17	Baida	12/24/2005	Monday, the twenty-sixth of June, 2006
18	Tobias	7/24/2005	Monday, the thirtieth of January, 2006
19	Himuro	11/15/2006	Monday, the twenty-first of May, 2007
20	Colmenares	8/10/2007	Monday, the eleventh of February, 2008
21	Weiss	7/18/2004	Monday, the twenty-fourth of January, 2005

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- d. Tạo một truy vấn hiển thị last name và commission của nhân viên. Nếu một nhân viên không kiếm được hoa hồng, hãy hiển thị "No commission". Đặt tên cột là COMM

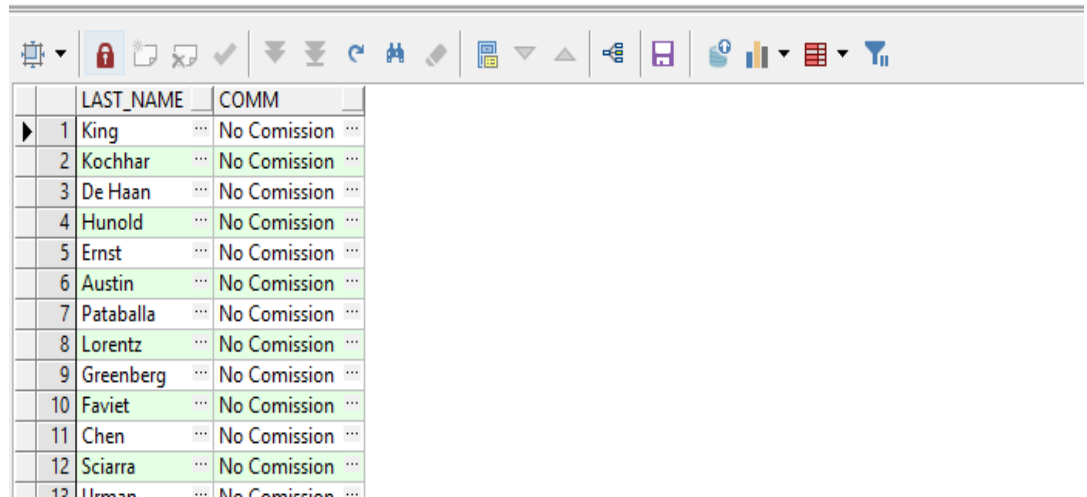
Câu lệnh SQL:

```
SELECT last_name, NVL2(commission_pct,TO_CHAR(commission_pct),'No Comission')
"COMM"
```

```
FROM employees;
```

Kết Quả:

```
SELECT last_name, NVL2(commission_pct,TO_CHAR(commission_pct),'No Comission') "COMM"
FROM employees;
```



	LAST_NAME	COMM
1	King	No Comission
2	Kochhar	No Comission
3	De Haan	No Comission
4	Hunold	No Comission
5	Ernst	No Comission
6	Austin	No Comission
7	Pataballa	No Comission
8	Lorentz	No Comission
9	Greenberg	No Comission
10	Faviet	No Comission
11	Chen	No Comission
12	Sciarra	No Comission
13	Ullman	No Comission

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- e. Sử dụng hàm DECODE, viết truy vấn hiển thị cấp bậc của tất cả nhân viên dựa trên giá trị của cột JOB_ID, sử dụng dữ liệu sau:

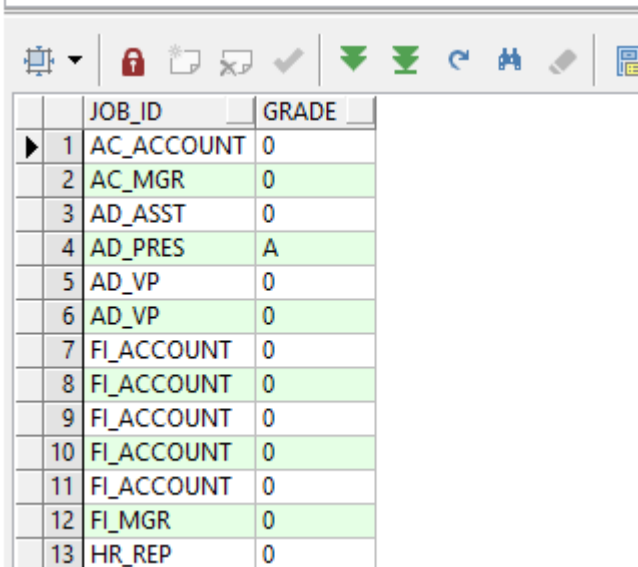
<i>Job Grade</i>	
AD_PRE	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

Câu lệnh SQL:

```
SELECT job_id,
DECODE (job_id, 'AD_PRE', 'A',
'ST_MAN', 'B',
'IT_PROG', 'C',
'SA_REP', 'D',
'ST_CLERK', 'E',
0 ) "GRADE"
FROM employees
```

Kết Quả:

```
SELECT job_id,  
       DECODE (job_id, 'AD_PRES', 'A',  
                 'ST_MAN', 'B',  
                 'IT_PROG', 'C',  
                 'SA_REP', 'D',  
                 'ST_CLERK', 'E',  
                 0 ) "GRADE"  
FROM employees
```



	JOB_ID	GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0
6	AD_VP	0
7	FI_ACCOUNT	0
8	FI_ACCOUNT	0
9	FI_ACCOUNT	0
10	FI_ACCOUNT	0
11	FI_ACCOUNT	0
12	FI_MGR	0
13	HR_REP	0

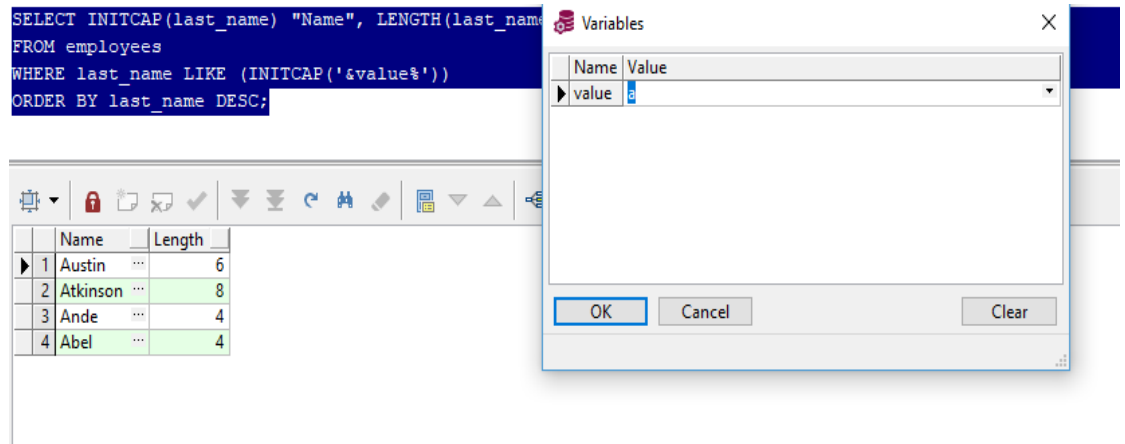
Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- f. Viết lại truy vấn để người dùng được nhắc nhập một chữ cái bắt đầu tên cuối cùng. Ví dụ: nếu người dùng nhập H khi được nhắc cho một chữ cái, thì đầu ra sẽ hiển thị tất cả các nhân viên có họ bắt đầu bằng chữ H

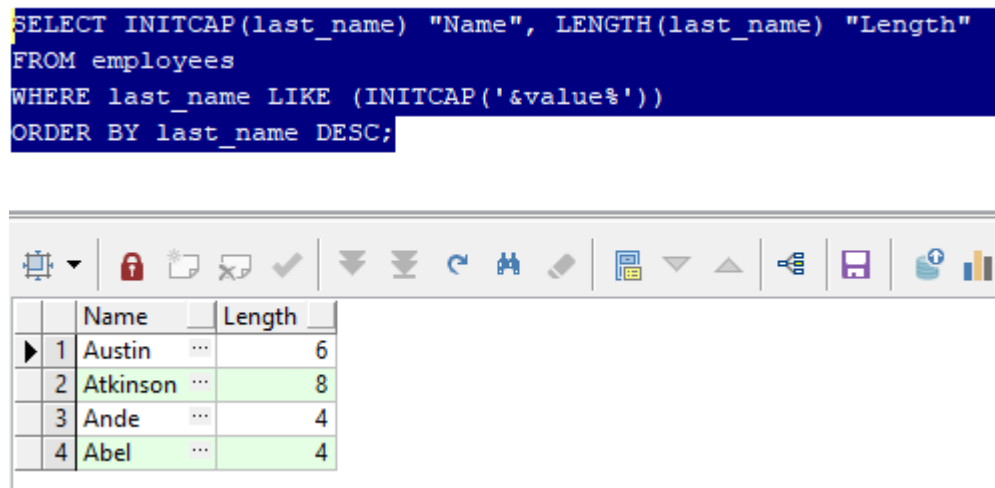
Câu lệnh SQL:

```
SELECT INITCAP(last_name) "Name", LENGTH(last_name) "Length"  
  
FROM employees  
  
WHERE last_name LIKE (INITCAP('&value%'))  
  
ORDER BY last_name DESC;
```

Kết Quả:



Sau khi nhập giá trị vào:



Nhận Xét: Kết quả thực hiện phù hợp với yêu cầu

3.5.2 PRACTICE 2

- Tìm mức lương cao nhất, thấp nhất, tổng và trung bình của tất cả nhân viên. Dán nhãn các cột tương ứng là MAXIMUM, MINIMUM, SUMMARY và AVERAGE. Làm tròn kết quả của bạn đến số nguyên gần nhất.

Câu lệnh SQL:

```
SELECT ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minumum", ROUND(SUM(salary)) "Sum",
ROUND(AVG(salary)) "Average"
```

```
FROM employees;
```

Kết Quả:

```
SELECT ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minumum", ROUND(SUM(salary)) "Sum", ROUND(AVG(salary)) "Average"
FROM employees;
```

	Maximum	Minumum	Sum	Average
1	24000	2100	691416	6462

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- b. Sửa đổi truy vấn trong bài tập trên để hiển thị mức lương tối thiểu, tối đa, tổng và trung bình cho từng loại công việc

Câu lệnh SQL:

```
SELECT job_id, ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minumum", ROUND(SUM(salary))
"Sum", ROUND(AVG(salary)) "Average"
```

```
FROM employees
```

```
GROUP BY job_id;
```

Kết Quả:

```
SELECT job_id, ROUND(MAX(salary)) "Maximum", ROUND(MIN(salary)) "Minumum", ROUND(SUM(salary)) "Sum", ROUND(AVG(salary)) "Average"
FROM employees
GROUP BY job_id;
```

	JOB_ID	Maximum	Minumum	Sum	Average
1	IT_PROG	9000	4200	28800	5760
2	AC_MGR	12008	12008	12008	12008
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	8200	5800	36400	7280
5	PU_MAN	11000	11000	11000	11000
6	AD_ASST	4400	4400	4400	4400
7	AD_VP	17000	17000	34000	17000
8	SH_CLERK	4200	2500	64300	3215
9	FI_ACCOUNT	9000	6900	39600	7920

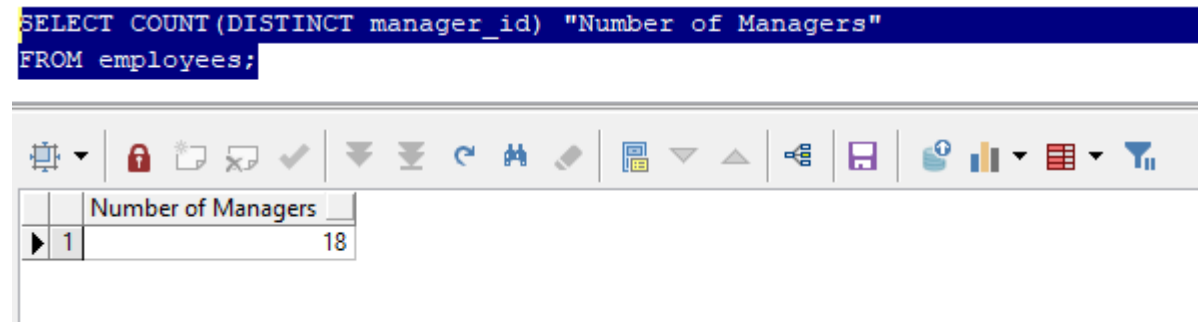
Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- c. Xác định số lượng người quản lý mà không liệt kê chúng. Dán nhãn cột Number of Managers. Gợi ý: Sử dụng cột Manager_ID để xác định số lượng người quản lý.

Câu lệnh SQL:

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM employees;
```

Kết Quả:



The screenshot shows a SQL query execution interface. At the top, the query is displayed in a dark blue box: `SELECT COUNT(DISTINCT manager_id) "Number of Managers" FROM employees;`. Below the query, there is a toolbar with various icons for editing and viewing. The result is shown in a table with one column titled "Number of Managers" and one row with the value 18.

	Number of Managers
1	18

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

- d. Tạo một báo cáo để hiển thị số người quản lý và mức lương của nhân viên được trả lương thấp nhất cho người quản lý đó. Loại trừ bất cứ ai mà người quản lý không được biết đến. Không bao gồm bất kỳ nhóm nào có mức lương tối thiểu là 6.000 đô la trở xuống. Sắp xếp đầu ra theo thứ tự giảm dần của tiền lương.

Câu lệnh SQL:

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

Kết Quả:


```

SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MIN(salary) DESC;

```

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	145	7000
4	146	7000
5	108	6900
6	147	6200
7	149	6200
8	148	6100

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.6 Lecture 6

3.6.1 PRACTICE 1

Nhân sự cần báo cáo của tất cả nhân viên. Viết một truy vấn để hiển thị họ, số phòng ban và tên bộ phận cho tất cả nhân viên

Câu lệnh SQL:

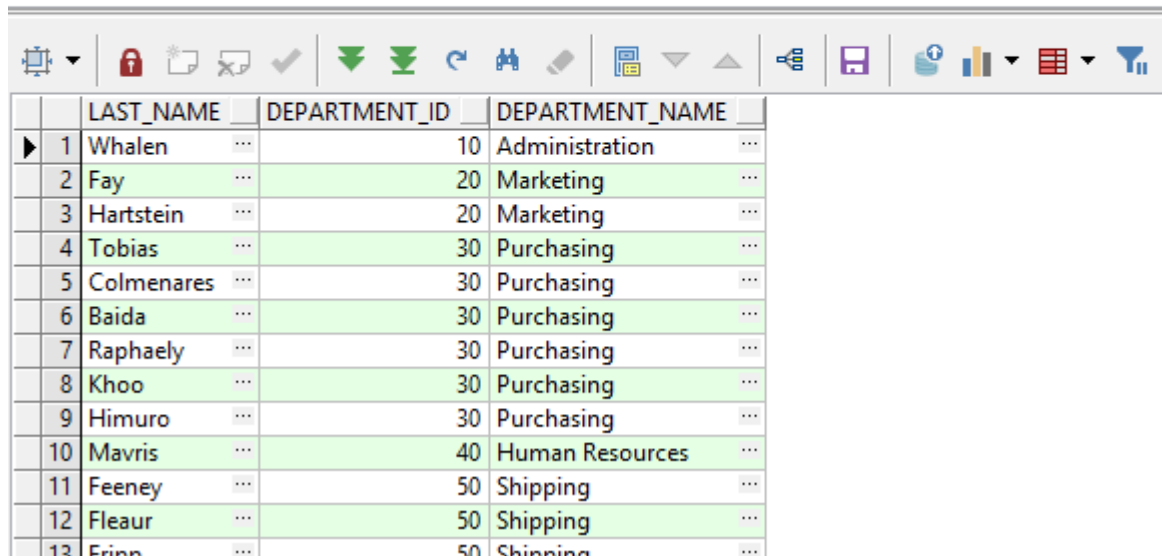
```
SELECT a.last_name, b.department_id, b.department_name
```

```
FROM employees a,departments b
```

```
WHERE a.department_id = b.department_id;
```

Kết Quả:

```
SELECT a.last_name, b.department_id, b.department_name
FROM employees a, departments b
WHERE a.department_id = b.department_id;
```



	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Tobias	30	Purchasing
5	Colmenares	30	Purchasing
6	Baida	30	Purchasing
7	Raphaely	30	Purchasing
8	Khoo	30	Purchasing
9	Himuro	30	Purchasing
10	Mavris	40	Human Resources
11	Feeney	50	Shipping
12	Fleaur	50	Shipping
13	Finn	50	Shipping

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.6.2 PRACTICE 2

Tạo một báo cáo để hiển thị họ tên nhân viên và số hiệu nhân viên cùng với người quản lý của họ. Dán nhãn các cột tương ứng Employee, Emp#, Manager, and Mgr#

Câu lệnh SQL:

```
SELECT e.last_name "Employee", e.employee_id "Emp#", m.last_name "Manager", m.employee_id "Mgr#"
```

```
FROM employees e
```

```
LEFT OUTER JOIN employees m
```

```
ON (e.manager_id = m.employee_id)
```

Kết Quả:

```

SELECT e.last_name "Employee", e.employee_id "Emp#", m.last_name "Manager", m.employee_id "Mgr#"
FROM employees e
LEFT OUTER JOIN employees m
ON (e.manager_id = m.employee_id)
--b
SELECT e.last_name "Employee", e.employee_id "Emp#", m.last_name "Manager", m.employee_id "Mgr#"
FROM employees e
LEFT OUTER JOIN employees m
ON (e.manager_id = m.employee_id)

```

	Employee	Emp#	Manager	Mgr#
1	Kumar	...	173	Cambrault ...
2	Bates	...	172	Cambrault ...
3	Smith	...	171	Cambrault ...
4	Fox	...	170	Cambrault ...
5	Bloom	...	169	Cambrault ...
6	Ozer	...	168	Cambrault ...
7	Hunold	...	103	De Haan ...
8	Banda	...	167	Errazuriz ...
9	Ande	...	166	Errazuriz ...
10	Lee	...	165	Errazuriz ...
11	Marvins	...	164	Errazuriz ...
12	Greene	...	163	Errazuriz ...
13	Vishney	...	162	Errazuriz ...
14	Cabrio	...	187	Frapp ...

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

(B)Modify Part A to display all employees including King, who has no manager. Order the results by the employee number.

-> Sửa lại A sau đó hiển thị tất cả nhân viên, kể cả King(người không có người quản lý). Sắp xếp kết quả theo số hiệu nhân viên.

Câu lệnh SQL:

```

SELECT e.last_name "Employee", e.employee_id "Emp#", m.last_name "Manager", m.employee_id
"Mgr#"

FROM employees e

LEFT OUTER JOIN employees m

ON (e.manager_id = m.employee_id)

ORDER BY e.employee_id;

```

Kết Quả:

```
SELECT e.last_name "Employee", e.employee_id "Emp#", m.last_name "Manager", m.employee_id "Mgr#"
FROM employees e
LEFT OUTER JOIN employees m
ON (e.manager_id = m.employee_id)
ORDER BY e.employee_id;
```

	Employee	Emp#	Manager	Mgr#
1	King	100		
2	Kochhar	101	King	100
3	De Haan	102	King	100
4	Hunold	103	De Haan	102
5	Ernst	104	Hunold	103
6	Austin	105	Hunold	103
7	Pataballa	106	Hunold	103
8	Lorentz	107	Hunold	103
9	Greenberg	108	Kochhar	101
10	Faviet	109	Greenberg	108
11	Chen	110	Greenberg	108
12	Sciarra	111	Greenberg	108
13	Urman	112	Greenberg	108

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.6.3 PRACTICE 3

Bộ phận nhân sự cần tìm tên và ngày thuê cho tất cả nhân viên được thuê trước người quản lý của họ, cùng với họ tên của người quản lý và ngày thuê.

Câu lệnh SQL:

```
SELECT e.last_name "Employee", e.hire_date "Emp's Hire Date", m.last_name "Manager", m.hire_date
"Mng's Hire Date"
```

```
FROM employees e
```

```
JOIN employees m
```

```
ON (e.manager_id = m.employee_id)
```

```
WHERE e.hire_date < m.hire_date;
```

Kết Quả:

```

SELECT e.last_name "Employee", e.hire_date "Emp's Hire Date", m.last_name "Manager", m.hire_date "Mng's Hire Date"
FROM employees e
JOIN employees m
ON (e.manager_id = m.employee_id)
WHERE e.hire_date < m.hire_date;

```

	Employee	Emp's Hire Date	Manager	Mng's Hire Date
1	Kaufling	5/1/2003	King	6/17/2003
2	Raphaely	12/7/2002	King	6/17/2003
3	De Haan	1/13/2001	King	6/17/2003
4	Higgins	6/7/2002	Kochhar	9/21/2005
5	Baer	6/7/2002	Kochhar	9/21/2005
6	Mavris	6/7/2002	Kochhar	9/21/2005
7	Whalen	9/17/2003	Kochhar	9/21/2005
8	Greenberg	8/17/2002	Kochhar	9/21/2005
9	Austin	6/25/2005	Hunold	1/3/2006
10	Faviet	8/16/2002	Greenberg	8/17/2002
11	Bull	2/20/2005	Fripp	4/10/2005
12	Sarchand	1/27/2004	Fripp	4/10/2005
13	Madow	2/16/2005	Fripp	4/10/2005

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.6.4 PRACTICE 4

Hiển thị số nhân viên, họ và tiền lương của tất cả các nhân viên kiếm được nhiều hơn mức lương trung bình và những người làm việc trong một bộ phận với bất kỳ nhân viên nào có họ của họ đều có chữ cái u

Câu lệnh SQL:

```

SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
AND department_id IN
  (SELECT department_id
   FROM employees
   WHERE last_name LIKE '%u%');

```

Kết Quả:

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
AND department_id IN
  (SELECT department_id
   FROM employees
   WHERE last_name LIKE '%u%');
```

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000.00
2	123	Vollman	6500.00
3	122	Kaufling	7900.00
4	121	Fripp	8200.00
5	120	Weiss	8000.00
6	177	Livingston	8400.00
7	176	Taylor	8600.00
8	175	Hutton	8800.00
9	174	Abel	11000.00
10	172	Bates	7300.00
11	171	Smith	7400.00
12	170	Fox	9600.00
13	169	Bloom	10000.00
14	168	Ozer	11500.00
15	165	Lee	6000.00

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.6.5 PRACTICE 5

Bộ phận nhân sự cần một báo cáo với các thông số sau:

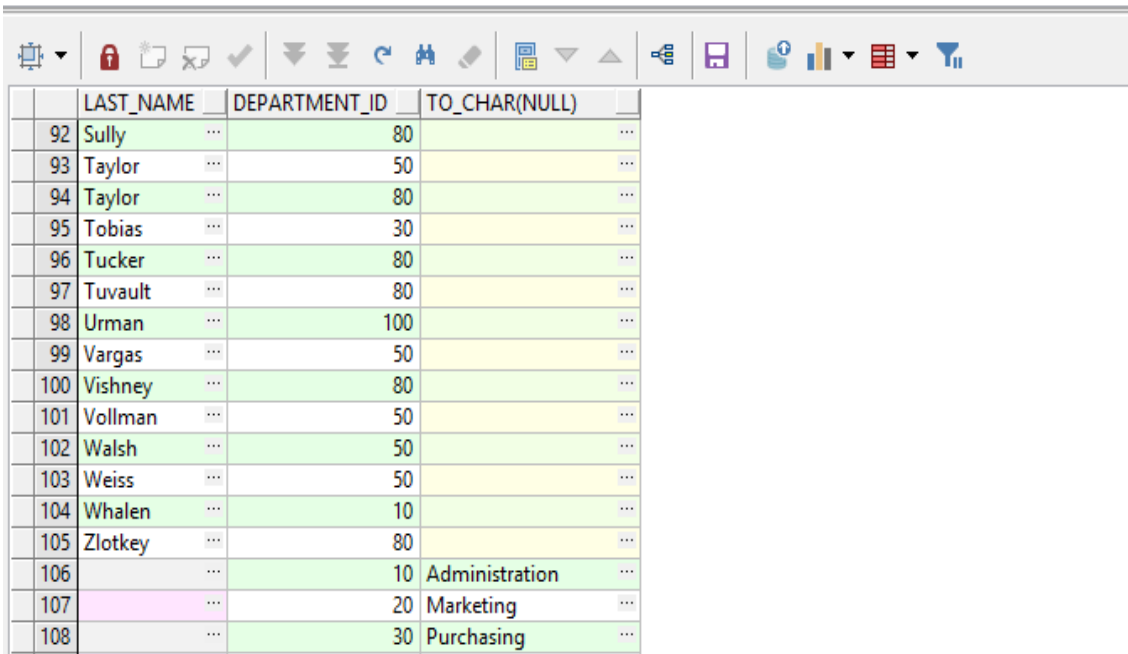
- Họ và tên bộ phận của tất cả các nhân viên trong bảng EMPLOYEES, bất kể dù cho họ không thuộc về một bộ phận nào
- ID phòng ban và tên bộ phận của tất cả các phòng ban từ bảng DEPARTMENTS, bất kể dù cho phòng ban đó không có nhân viên nào làm việc

Câu lệnh SQL:

```
SELECT last_name, department_id, TO_CHAR(NULL)
FROM employees
UNION
SELECT TO_CHAR(NULL), department_id, department_name
FROM departments;
```

Kết Quả:

```
SELECT last_name, department_id, TO_CHAR(NULL)
FROM employees
UNION
SELECT TO_CHAR(NULL), department_id, department_name
FROM departments;
```



	LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
92	Sully	80	
93	Taylor	50	
94	Taylor	80	
95	Tobias	30	
96	Tucker	80	
97	Tuvault	80	
98	Urman	100	
99	Vargas	50	
100	Vishney	80	
101	Vollman	50	
102	Walsh	50	
103	Weiss	50	
104	Whalen	10	
105	Zlotkey	80	
106		10	Administration
107		20	Marketing
108		30	Purchasing

Nhận Xét:

- Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.
- Kết hợp 2 bảng bằng cách xếp chồng chúng lên nhau

3.6.6 PRACTICE 6

Tạo một báo cáo liệt kê ID nhân viên và job ID của những nhân viên hiện đang có chức danh công việc giống như chức danh công việc của họ khi ban đầu được công ty thuê (nghĩa là họ đã thay đổi công việc nhưng giờ đã quay lại làm công việc ban đầu của họ).

Câu lệnh SQL:

```
SELECT employee_id, job_id
```

```
FROM employees
```

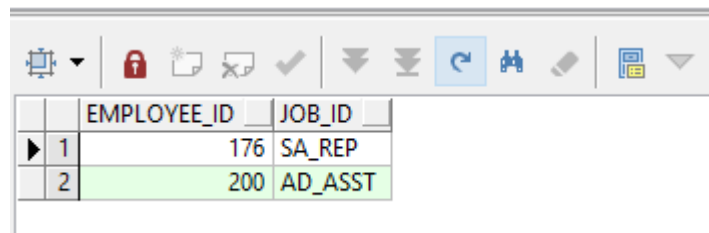
```
INTERSECT
```

```
SELECT employee_id, job_id
```

```
FROM job_history;
```

Kết Quả:

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```



	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

Nhận Xét:

- Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.
- INTERSECT chỉ trả về các hàng chung được trả về bởi hai lệnh SELECT.

3.6.7 PRACTICE 7

Bộ phận nhân sự cần một danh sách các vùng không có bộ phận nào trong đó. Hiển thị ID của vùng và tên của các vùng. Sử dụng bộ toán tử để tạo báo cáo này

Câu lệnh SQL:

```
SELECT country_id, country_name
```

```
FROM countries
```

```
MINUS
```

```
SELECT l.country_id, c.country_name
```

```
FROM locations l JOIN countries c
```

```
ON (l.country_id = c.country_id);
```

Kết Quả:


```

SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id);

```

		COUNTRY_ID	COUNTRY_NAME	
▶	1	AR	Argentina	...
	2	BE	Belgium	...
	3	DK	Denmark	...
	4	EG	Egypt	...
	5	FR	France	...
	6	IL	Israel	...
	7	KW	Kuwait	...
	8	ML	Malaysia	...
	9	NG	Nigeria	...
	10	ZM	Zambia	...
	11	ZW	Zimbabwe	...

Nhận Xét:

- Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.
- MINUS sẽ hợp lại và chọn ra những record chỉ có ở câu SELECT đầu tiên mà không có ở câu SELECT thứ hai.

3.7 Lecture 7

Không có bài tập.

3.8 Lecture 8

3.8.1 PRACTICE 1

Viết truy vấn để hiển thị thông tin sau cho những nhân viên mà người quản lý của họ có ID dưới 120:

+ Manager ID

+ Job_ID và tổng tiền lương cho mỗi job_ID cho nhân viên báo thuộc cùng quản lý

+ Tổng tiền lương của những người quản lý

+ Tổng tiền lương của những người quản lý, không phân biệt job_ID

Câu lệnh SQL:

```
select manager_id,job_id, sum(salary)
from employees
where manager_id <120
group by rollup(manager_id,job_id);
```

Kết Quả:

```
select manager_id,job_id, sum(salary)
from employees
where manager_id <120
group by rollup(manager_id,job_id);
```

	MANAGER_ID	JOB_ID	SUM(SALARY)
1	100	AD_VP	34000
2	100	MK_MAN	13000
3	100	PU_MAN	11000
4	100	SA_MAN	61000
5	100	ST_MAN	36400
6	100		155400
7	101	AC_MGR	12008
8	101	FI_MGR	12008
9	101	HR_REP	6500
10	101	PR_REP	10000
11	101	AD_ASST	4400
12	101		44916
13	102	IT_PROG	9000
14	102		9000
15	103	IT_PROG	19800
16	103		19800
17	108	FI_ACCOUNT	39600
18	108		39600
19	114	PU_CLERK	13900
20	114		13900
21			282616

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.8.2 PRACTICE 2

Quan sát đầu ra từ câu hỏi 1. Viết truy vấn bằng hàm GROUPING để xác định xem các giá trị NULL trong các cột tương ứng với các biểu thức GROUP BY có phải do hàm ROLLUP gây ra hay không

Câu lệnh SQL:

```
select manager_id MGR, job_id JOB, SUM(salary),
```

```
Grouping(manager_id), grouping(job_id)
```

```
from employees where manager_id < 120
```

```
group by rollup(manager_id,job_id);
```

Kết Quả:

```
select manager_id MGR, job_id JOB, SUM(salary),
Grouping(manager_id), grouping(job_id)
from employees where manager_id < 120
group by rollup(manager_id,job_id);
```

	MGR	JOB	SUM(SALARY)	GROUPING(MANAGER_ID)	GROUPING(JOB_ID)
1	100	AD_VP	34000	0	0
2	100	MK_MAN	13000	0	0
3	100	PU_MAN	11000	0	0
4	100	SA_MAN	61000	0	0
5	100	ST_MAN	36400	0	0
6	100		155400	0	1
7	101	AC_MGR	12008	0	0
8	101	FI_MGR	12008	0	0
9	101	HR_REP	6500	0	0
10	101	PR_REP	10000	0	0
11	101	AD_ASST	4400	0	0
12	101		44916	0	1
13	102	IT_PROG	9000	0	0
14	102		9000	0	1
15	103	IT_PROG	19800	0	0
16	103		19800	0	1
17	108	FI_ACCOUNT	39600	0	0
18	108		39600	0	1
19	114	PU_CLERK	13900	0	0
20	114		13900	0	1
21			282616	1	1

Nhận Xét: Kết quả hoàn toàn chứng tỏ các giá trị NULL trong các cột tương ứng với các biểu thức GROUP BY do hàm ROLLUP gây ra.

3.8.3 PRACTICE 3

Viết truy vấn để hiển thị thông tin sau cho những nhân viên mà người quản lý có ID dưới 120:

- Manager ID
- Công việc và tổng tiền lương cho mọi công việc cho nhân viên có cùng một người quản lý
- Tổng tiền lương của những người quản lý
- Các giá trị lập bảng chéo để hiển thị tổng tiền lương cho mọi công việc, không phân biệt quản lý
- Tổng tiền lương không phân biệt tất cả các chức danh công việc

Câu lệnh SQL:

```
select manager_id,job_id, sum(salary)
from employees
where manager_id <120
group by cube(manager_id,job_id);
```

Kết Quả:

```
select manager_id,job_id, sum(salary)
from employees
where manager_id <120
group by cube(manager_id,job_id);
```

	MANAGER_ID	JOB_ID	SUM(SALARY)
1			282616
2		AD_VP	34000
3		AC_MGR	12008
4		FI_MGR	12008
5		HR_REP	6500
6		MK_MAN	13000
7		PR_REP	10000
8		PU_MAN	11000
9		SA_MAN	61000
10		ST_MAN	36400
11		AD_ASST	4400
12		IT_PROG	28800
13		PU_CLERK	13900
14		FI_ACCOUNT	39600
15	100		155400
16	100	AD_VP	34000
17	100	MK_MAN	13000
18	100	PU_MAN	11000
19	100	SA_MAN	61000
20	100	ST_MAN	36400
21	101		44916
22	101	AC_MGR	12008
23	101	FI_MGR	12008

Nhận Xét: Kết quả xuất ra màn hình phù hợp với yêu cầu đề bài đưa ra.

3.8.4 PRACTICE 4

Sử dụng GROUPING SET, viết truy vấn để hiển thị các nhóm sau:

- department_id, manager_id, job_id
- department_id, job_id
- manager_id, job_id
- Truy vấn sẽ tính tổng tiền lương cho mỗi nhóm này.

Câu lệnh SQL:

```
select department_id, manager_id, job_id, Sum(salary)
from employees
group by grouping sets((department_id,manager_id,job_id),
(department_id, job_id), (manager_id,job_id));
```

Kết quả:

```
select department_id, manager_id, job_id, Sum(salary)
from employees
group by grouping sets((department_id,manager_id,job_id),
(department_id, job_id), (manager_id,job_id));
```

	DEPARTMENT_ID	MANAGER_ID	JOB_ID	SUM(SALARY)
1	90		AD_PRES	24000
2	90	100	AD_VP	34000
3	20	100	MK_MAN	13000
4	30	100	PU_MAN	11000
5	80	100	SA_MAN	61000
6	50	100	ST_MAN	36400
7	110	101	AC_MGR	12008
8	100	101	FI_MGR	12008
9	40	101	HR_REP	6500
10	70	101	PR_REP	10000
11	10	101	AD_ASST	4400
12	60	102	IT_PROG	9000
13	60	103	IT_PROG	19800
14	100	108	FI_ACCOUNT	39600
15	30	114	PU_CLERK	13900
16	50	120	SH_CLERK	11600
17	50	120	ST_CLERK	10500

Nhận xét: Câu truy vấn phù hợp với yêu cầu đề bài.

3.9 Lecture 9

PHẦN 4: BÀI TẬP KẾT THÚC MÔN HỌC

4.1 Bài 1

Đề bài: Kiểm tra 1 sinh viên đã đủ điều kiện tốt nghiệp chưa biết rằng các điều kiện để một sinh viên tốt nghiệp là:

- Tích lũy đủ số tín chỉ
- Điểm trung bình tốt nghiệp không nhỏ hơn 1.0, biết bảng đổi điểm như sau:

	Thang điểm 4	
	Điểm chữ	Điểm số
ĐẠT	A+	4.5
	A	4.0
	A-	3.5
	B+	3.0
	B	2.5
	B-	2.0
	C+	1.5
	C	1.0
KHÔNG ĐẠT	C-	0.5

Các bước làm:

Bước 1: Đầu tiên điểm được lưu ở bảng takes dưới dạng chữ ta cần phải tạo một view (`CHUYEN_HE_SO_4`) để chuyển điểm từ dạng chữ sang số.

Bước 2: Vì trường hợp học sinh có thể học đi học lại nhiều lần lên ta phải tạo 1 view (`LAY_DIEM CAO NHAT`) để lấy điểm học cao nhất của học sinh.

Bước 3: Theo chú ý “Sinh viên phải tính số tín chỉ tích lũy. Không sử dụng trường thông tin trong bảng Student” lên ta phải tạo một view (`TINH_TINH_LUY`) để lấy ra số tín chỉ của sinh viên với điều kiện điểm số lớn hơn hoặc bằng 1.

Bước 4: Tạo một thủ tục để kiểm tra điều kiện tốt nghiệp thông tin với dữ liệu số tín chỉ tích lũy từ view (`TINH_TINH_LUY`), điểm phải trung bình dựa vào bảng course (lấy số tín môn học) và view (`LAY_DIEM CAO NHAT`).

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1:

```
CREATE OR REPLACE VIEW CHUYEN_HE_SO_4 AS
SELECT id,
       course_id,
       sec_id,
       semester,
       year ,
       CASE
         WHEN grade = 'A+' THEN 4.5
         WHEN grade = 'A ' THEN 4
         WHEN grade = 'A-' THEN 3.5
         WHEN grade = 'B+' THEN 3
         WHEN grade = 'B' THEN 2.5
```

```

        WHEN grade = 'B-' THEN 2
        WHEN grade = 'C+' THEN 1.5
        WHEN grade = 'C' THEN 1
        WHEN grade = 'C-' THEN 0.5
        ELSE 0
    END AS point
FROM takes;

```

Bước 2:

```

CREATE OR REPLACE VIEW LAY_DIEM_CAO_NHAT AS
SELECT id,
       course_id,
       MAX(point) count
FROM CHUYEN_HE_SO_4
GROUP BY id,
       course_id;

```

Bước 3:

```

CREATE OR REPLACE VIEW TINH_TINH_LUY AS
SELECT s.ID,
       s.Name,
       p.TOTAL_CRED tich_luy_chi_thuc,
       s.TOT_CRED tich_luy_ao
FROM
    (SELECT SUM(c.CREDITS) total_cred,
     t.ID
     FROM LAY_DIEM_CAO_NHAT t
     JOIN COURSE c ON c.COURSE_ID = t.COURSE_ID
     where t.count >= 1
     GROUP BY t.ID) p
JOIN STUDENT s
ON p.ID = s.ID;

```

Bước 4:

```

CREATE OR REPLACE PROCEDURE SP_BAI1(student_id IN VARCHAR)
AS tot_cred NUMBER;
   cpa NUMBER;
BEGIN
    SELECT tich_luy_chi_thuc INTO tot_cred
    FROM tinh_tinh_luy
    WHERE ID = student_id;
    IF tot_cred > 128 THEN
        SELECT SUM(t.count * c.credits) / SUM(c.credits)
        INTO cpa
        FROM LAY_DIEM_CAO_NHAT t
        JOIN course c ON t.id = student_id
        AND t.course_id = c.course_id;
    END IF;

```

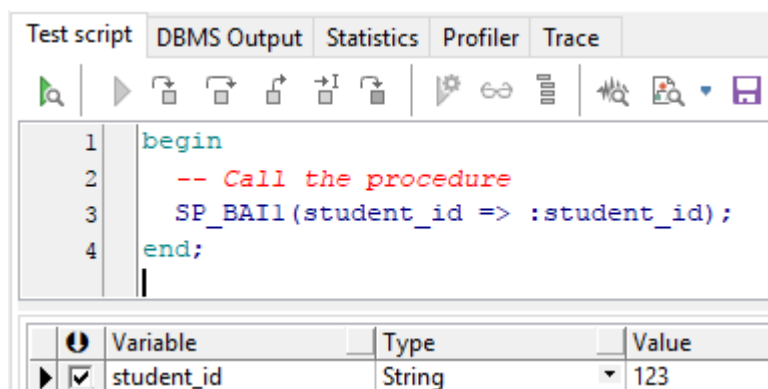


```

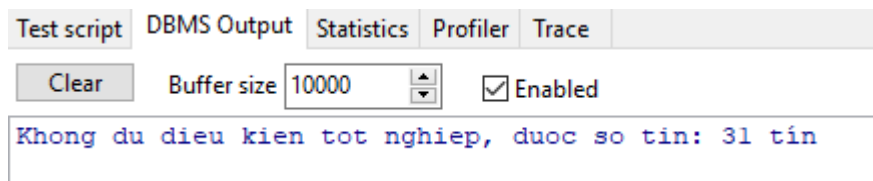
IF cpa < 1.0 THEN
    DBMS_OUTPUT.PUT_LINE('Khong du dieu kien tot nghiep, diem trung binh ' ||
cpa);
ELSE DBMS_OUTPUT.PUT_LINE('Da du dieu kien tot nghiep, diem trung binh ' ||
cpa);
END IF;
ELSE DBMS_OUTPUT.PUT_LINE('Khong du dieu kien tot nghiep, duoc so tin: ' ||
tot_cred || ' tín');
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Ma sinh vien khong dung: ' || student_id);
END;

```

Chạy kiểm tra với học sinh có id: 123.



Kết quả:



b. Các câu lệnh SQL trong SQL server

Bước 1: Chuyển sang hệ số 4

```

CREATE VIEW CHUYEN_HE_SO_4 AS
SELECT id,
course_id,
sec_id,
semester,
year,
CASE
    WHEN grade = 'A+' THEN 4.5
    WHEN grade = 'A' THEN 4
    WHEN grade = 'A-' THEN 3.5
    WHEN grade = 'B+' THEN 3

```

```

        WHEN grade = 'B' THEN 2.5
        WHEN grade = 'B-' THEN 2
        WHEN grade = 'C+' THEN 1.5
        WHEN grade = 'C' THEN 1
        WHEN grade = 'C-' THEN 0.5
        ELSE 0
    END AS point

```

```

FROM takes;

```

Bước 2: Lấy ra điểm cao nhất

```

CREATE VIEW LAY_DIEM_CAO_NHAT AS
SELECT id,
       course_id,
       MAX(point) count
FROM CHUYEN_HE_SO_4
GROUP BY id, course_id;

```

Bước 3:

```

CREATE VIEW TINH_TINH_LUY AS
SELECT s.ID,
       s.Name,
       p.TOTAL_CRED tich_luy_chi_thuc,
       s.TOT_CRED tich_luy_ao
FROM
    (SELECT SUM(c.CREDITS) total_cred,
     t.ID
    FROM LAY_DIEM_CAO_NHAT t
    JOIN COURSE c ON c.COURSE_ID = t.COURSE_ID
    where t.count >= 1
    GROUP BY t.ID) p
JOIN STUDENT s
ON p.ID = s.ID;

```

Bước 4:

```

create procedure SP_BAI1 @student_id VARCHAR(20)
AS
BEGIN
    declare @tot_cred tinyint
    declare @cpa tinyint
    declare @KQ tinyint
    SELECT @tot_cred = tich_luy_chi_thuc
    FROM tinh_tinh_luy
    WHERE ID = @student_id;
    SELECT @cpa = SUM(LAY_DIEM_CAO_NHAT.count * c.credits) / SUM(c.credits)
    FROM LAY_DIEM_CAO_NHAT
    JOIN course c ON LAY_DIEM_CAO_NHAT.id = @student_id

```

```

AND LAY_DIEM_CAO_NHAT.course_id = c.course_id;
    if @cpa >= 1
        set @KQ = 1
    else
        set @KQ = 0

    if @tot_cred > 128
        set @KQ = 1
    else
        set @KQ = 0

    if @KQ = 1
        print 'Du Dieu Kien Tot Nghiep'
    else
        print 'Chua Du Dieu Kien Tot Nghiep so tin: ' +
convert(varchar,@tot_cred) + 'diem trung binh: ' + convert(varchar,@cpa)

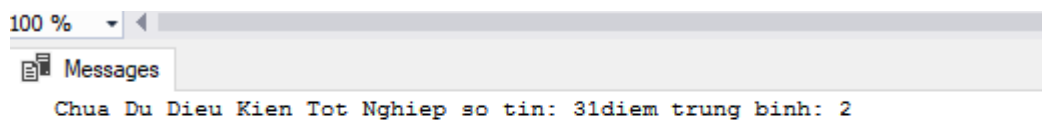
end

```

Chạy kiểm tra với id: 123

```
exec SP_BAI1 @student_id = 123
```

Kết quả:



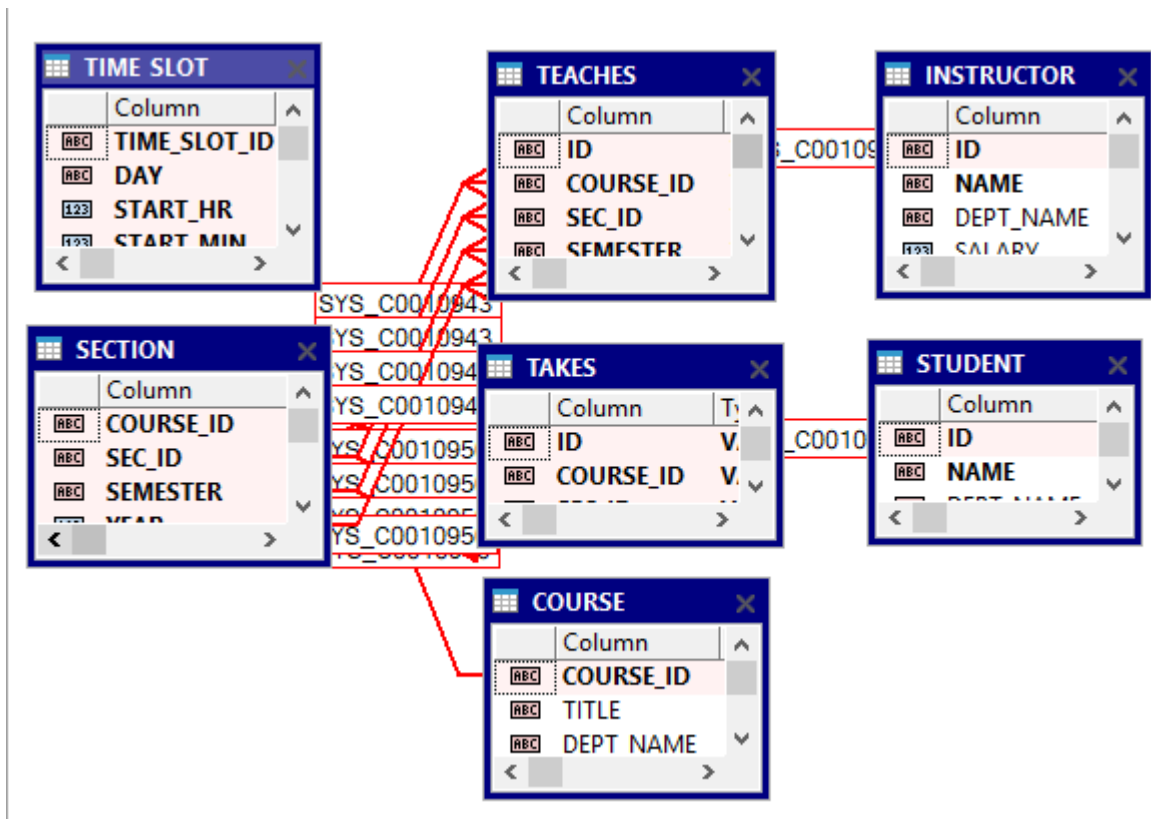
4.2 Bài 2

Đề bài: Viết thủ tục SP_LOC_DU_LIEU cho phép nhập vào tên trường bất kỳ và một giá trị của trường (Ví dụ: SP_LOC_DU_LIEU 'dept_name', 'Physics'). Kết quả trả về là dữ liệu sau khi lọc theo giá trị của trường dữ liệu đó.

Bảng kết quả trả về gồm các trường: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện.

Các bước làm:

Bước 1: Để có dữ liệu của mã sinh viên, họ và tên sinh viên, năm học, kỳ học, khóa học, thời gian học, phòng học, giảng viên, khoa viên ta phải lấy dữ liệu từ các bảng: section, course, teaches, instructor, section, takes, student, course.



Bước 2: Tạo một thủ tục với input đầu vào là vào tên trường bất kỳ và một giá trị của trường sau đó sẽ select trong view mới tạo ra và in ra kết quả bao gồm: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện.

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo view TONG_HOP_CAU2 lấy dữ liệu theo yêu cầu qua các bảng.

```
CREATE OR REPLACE VIEW TONG_HOP_CAU2 AS
SELECT st.id,
       st.name studnet_name,
       se.year,
       se.semester,
       c.title,
       ts.day,
```

```

ts.start_hr,
ts.start_min,
ts.end_hr,
ts.end_min,
se.room_number,
se.building,
i.name instructor_name,
st.dept_name
FROM section se
JOIN course c ON se.course_id = c.course_id
JOIN teaches te ON se.course_id = te.course_id
    AND se.sec_id = te.sec_id
    AND se.semester = te.semester
    AND se.year = te.year
JOIN instructor i ON te.id = i.id
JOIN takes ta ON se.sec_id = ta.sec_id
    AND se.semester = ta.semester
    AND se.year = ta.year
    AND se.course_id = ta.course_id
JOIN student st ON ta.id = st.id
LEFT JOIN time_slot ts ON se.time_slot_id = ts.time_slot_id;

```

Bước 2: Tạo thủ tục thủ tục lọc dữ liệu sử dụng OPEN-FOR-USING Statement trong oracle:

```

CREATE OR REPLACE PROCEDURE SP_LOC_DU_LIEU_BAI2(field_name IN
VARCHAR2, field_value IN VARCHAR, mycursor OUT SYS_REFCURSOR)
AS str_query VARCHAR (1000 );
BEGIN
    str_query := 'SELECT * FROM TONG_HOP_CAU2 WHERE ' || field_name || ' LIKE
    '%' || field_value || '%';
    OPEN mycursor FOR str_query;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Truong khong ton tai: ' || field_name);
END;

```

Chạy thử kiểm tra với cột 'dept_name' có giá trị 'Physics':

Test script DBMS Output Statistics Profiler Trace

```

1 begin
2     -- Call the procedure
3     SP_LOC_DU_LIEU_BAI2(field_name => :field_name,
4                         field_value => :field_value,
5                         mycursor => :mycursor);
6 end;

```

Variable	Type	Value
field_name	String	dept_name
field_value	String	Physics
mycursor	Cursor	<Cursor>

Kết quả:

	YEAR	SEMESTER	TITLE	DAY	START_HR	START_MIN	END_HR	END_MIN	ROOM_NUMBER	BUILDING	INSTRUCTOR_NAME	DEPT_NAME
1	2009	Fall	Image Processing	F	11	0	11	50	375	Chandler	Romero	Physics
2	2009	Fall	Image Processing	W	11	0	11	50	375	Chandler	Romero	Physics
3	2009	Fall	Image Processing	M	11	0	11	50	375	Chandler	Romero	Physics
4	2002	Fall	Image Processing	F	11	0	11	50	183	Taylor	Romero	Physics
5	2002	Fall	Image Processing	W	11	0	11	50	183	Taylor	Romero	Physics
6	2002	Fall	Image Processing	M	11	0	11	50	183	Taylor	Romero	Physics
7	2009	Fall	Music 2 New for y	W	10	0	12	30	143	Lamberton	Mingoz	Physics
8	2005	Fall	Embedded System						143	Lamberton	Mingoz	Physics
9	2008	Spring	Embedded System						700	Bronfman	Mingoz	Physics
10	2006	Spring	Video Gaming	F	16	0	16	50	134	Lamberton	Mingoz	Physics
11	2006	Spring	Video Gaming	W	16	0	16	50	134	Lamberton	Mingoz	Physics

b. Các câu lệnh trong SQL server:

Bước 1: tạo view tong_hop_cau2:

```

CREATE VIEW TONG_HOP_CAU2 AS
SELECT st.id,
       st.name studnet_name,
       se.year,
       se.semester,
       c.title,
       ts.day,
       ts.start_hr,
       ts.start_min,
       ts.end_hr,
       ts.end_min,
       se.room_number,
       se.building,
       i.name instructor_name,
       st.dept_name
FROM section se
JOIN course c ON se.course_id = c.course_id

```

```

JOIN teaches te ON se.course_id = te.course_id
AND se.sec_id = te.sec_id
AND se.semester = te.semester
AND se.year = te.year
JOIN instructor i ON te.id = i.id
JOIN takes ta ON se.sec_id = ta.sec_id
AND se.semester = ta.semester
AND se.year = ta.year
AND se.course_id = ta.course_id
JOIN student st ON ta.id = st.id

LEFT JOIN time_slot ts ON se.time_slot_id = ts.time_slot_id;

```

Bước 2: Tạo thủ tục:

```

USE [CSDLNC]
GO
/***** Object: StoredProcedure [dbo].[SP_LOC_DU_LIEU_BAI2]  Script Date:
6/7/2019 4:19:21 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[SP_LOC_DU_LIEU_BAI2](@field_name
VARCHAR(4000), @field_value VARCHAR)
AS
BEGIN DECLARE @str_query VARCHAR (1000 );

SET NOCOUNT ON;
SET @str_query = 'SELECT * FROM TONG_HOP_CAU2 WHERE ' +
isnull(CONVERT(varchar,@field_name), '') + ' LIKE "%' + isnull(CONVERT(varchar,
@field_value), '') + '%"';
EXECUTE (@str_query);

END;

```

Chạy thử:

```
exec SP_LOC_DU_LIEU_BAI2 @field_name = 'dept_name', @field_value = 'Physics'
```

Kết quả:

Results

Messages

	id	studnet_name	year	semester	title	day	start_hr	start_min	end_hr	end_min	room_number	building	instructor_name	dept_name
121	10705	Terauchi	2007	Spring	Plasma Physics	T	10	30	11	45	183	Taylor	Gustafsson	Physics
122	10705	Terauchi	2004	Fall	Video Gaming	F	13	0	13	50	113	Saucon	D'Agostino	Physics
123	10705	Terauchi	2004	Fall	Video Gaming	M	13	0	13	50	113	Saucon	D'Agostino	Physics
124	10705	Terauchi	2004	Fall	Video Gaming	W	13	0	13	50	113	Saucon	D'Agostino	Physics
125	10705	Terauchi	2005	Spring	Geology	F	13	0	13	50	145	Fairchild	D'Agostino	Physics
126	10705	Terauchi	2005	Spring	Geology	M	13	0	13	50	145	Fairchild	D'Agostino	Physics
127	10705	Terauchi	2005	Spring	Geology	W	13	0	13	50	145	Fairchild	D'Agostino	Physics

Query executed successfully.

DESKTOP-7BSH68L (14.0 RTM)

DESKTOP-7BSH68L\Minh N...

CSDLNC

00:00:00

10866 rows

Query executed successfully. DESKTOP-7BSH68L (14.0 RTM) DESKTOP-7BSH68L\Minh N... CSDLNC 00:00:00 10866 rows

4.3 Bài 3

Đề bài: Viết thủ tục SP_LOC_DU_LIEU cho phép nhập vào một biến kiểu table gồm 2 trường: tên trường và một giá trị của trường. Kết quả trả về là dữ liệu sau khi lọc theo danh sách các giá trị của các trường dữ liệu đó.

Các bước làm:

Bước 1: Tạo kiểu dữ liệu bảng gồm 2 trường: tên trường và một giá trị của trường để input.

Bước 2: Tạo hàm thủ tục lọc dữ liệu theo kiểu bảng sử dụng lại view [TONG_HOP_CAU2](#).

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo type bảng tên `my_table`

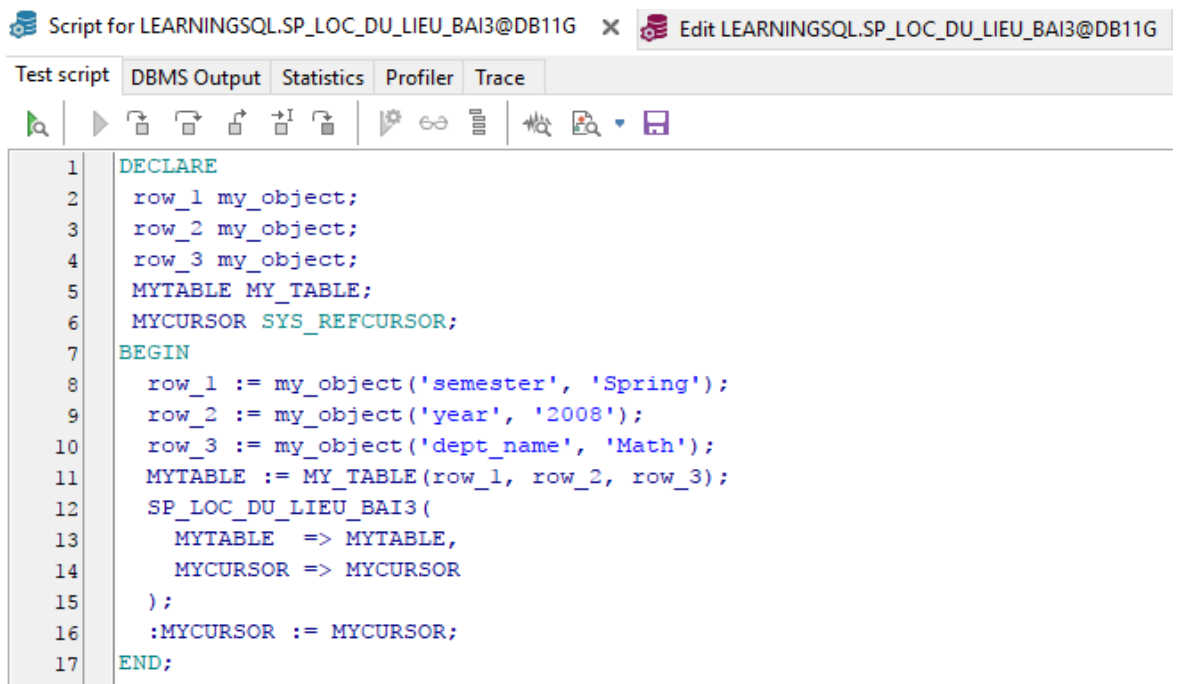
```
CREATE OR REPLACE TYPE my_object  
AS OBJECT(field_name VARCHAR(50),  
          field_value VARCHAR(50))
```

```
CREATE OR REPLACE TYPE my_table  
AS TABLE OF my_object
```

Bước 2: Tạo thủ tục

```
CREATE OR REPLACE PROCEDURE SP_LOC_DU_LIEU_BAI3 (mytable IN  
my_table, mycursor OUT SYS_REFCURSOR)  
AS str_query VARCHAR (1000);  
BEGIN  
    str_query := 'SELECT * FROM TONG_HOP_CAU2 WHERE 0=0';  
    FOR indx IN mytable.FIRST .. mytable.LAST LOOP  
        str_query := str_query || ' AND ' || mytable(indx).field_name || ' LIKE "%'  
|| mytable(indx).field_value || '%" ' ;  
    END LOOP;  
    OPEN mycursor FOR str_query;  
    EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE('Dieu kien dau vao khong dung!');  
END;
```


Chạy thử: Lọc theo cột 'semester', 'year', 'dept_name' với giá trị 'Spring', '2018', 'Math'



```

1 DECLARE
2   row_1 my_object;
3   row_2 my_object;
4   row_3 my_object;
5   MYTABLE MY_TABLE;
6   MYCURSOR SYS_REFCURSOR;
7 BEGIN
8   row_1 := my_object('semester', 'Spring');
9   row_2 := my_object('year', '2008');
10  row_3 := my_object('dept_name', 'Math');
11  MYTABLE := MY_TABLE(row_1, row_2, row_3);
12  SP_LOC_DU_LIEU_BAI3(
13    MYTABLE => MYTABLE,
14    MYCURSOR => MYCURSOR
15  );
16  :MYCURSOR := MYCURSOR;
17 END;

```

Kết quả:

	YEAR	SEMESTER	TITLE	DAY	START_HR	START_MIN	END_HR	END_MIN	ROOM_NUMBER	BUILDING	INSTRUCTOR_NAME	DEPT_NAME
1	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
2	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
3	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
4	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
5	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
6	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
7	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
8	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math
9	2008	Spring	Elastic Structures ...	F	13	0	13	50	812	Taylor	Dale	Math

b. Các câu lệnh SQL trong SQL server:

Bước 1: Tạo kiểu dữ liệu table:

```

CREATE TYPE MyTable
as table (
    field_name varchar(20),
    field_value varchar(20)
);

```

Bước 2: Tạo thủ tục lọc dữ liệu:

```

IF OBJECT_ID('SP_LOC_DU_LIEU_BAI3', 'P') IS NOT NULL
    DROP PROCEDURE SP_LOC_DU_LIEU_BAI3;
GO

```

```

CREATE PROCEDURE [dbo].[SP_LOC_DU_LIEU_BAI3] (@mytable MyTable
readonly)
AS
BEGIN
DECLARE @str_query VARCHAR (1000);
DECLARE @i int = 1;
DECLARE @numrows int = 0;
DECLARE @table table(RowNum int,col varchar(50),search varchar(50));
declare @field_name varchar(20)
declare @field_value varchar (20)

insert into @table select ROW_NUMBER() over (order by field_name), * from
@mytable

set @numrows = (select COUNT(*) from @mytable)
if @numrows > 0
    SET @str_query = 'SELECT * FROM TONG_HOP_CAU2 WHERE 0=0';
    begin
        while (@i <= @numrows)
            begin
                select @field_name = col, @field_value = search from @table
where RowNum = @i
                set @str_query = @str_query + 'and ' + isnull(@field_name, '') + '
LIKE "%' + isnull(@field_value, '') + '%"
                set @i = @i + 1
            end
        end
        EXECUTE (@str_query);

END

```

Chạy thử:

```

declare @table MyTable
insert into @table values('dept_name','Phy')
insert into @table values('semester','Spr')

exec SP_LOC_DU_LIEU_BAI3 @mytable = @table

```

Kết quả:

	id	studnet_name	year	semester	title	day	start_hr	start_min	end_hr	end_min	room_number	building	instructor_name	dept_name
1	41675	Wheeler	2007	Spring	The Music of the Ramones	F	13	0	13	50	180	Saucon	Lembr	Physics
2	41675	Wheeler	2007	Spring	The Music of the Ramones	M	13	0	13	50	180	Saucon	Lembr	Physics
3	41675	Wheeler	2007	Spring	The Music of the Ramones	W	13	0	13	50	180	Saucon	Lembr	Physics
4	34542	Basile	2007	Spring	The Music of the Ramones	F	13	0	13	50	180	Saucon	Lembr	Physics
5	34542	Basile	2007	Spring	The Music of the Ramones	M	13	0	13	50	180	Saucon	Lembr	Physics
6	34542	Basile	2007	Spring	The Music of the Ramones	W	13	0	13	50	180	Saucon	Lembr	Physics
7	27687	Yüksel	2007	Spring	The Music of the Ramones	F	13	0	13	50	180	Saucon	Lembr	Physics
8	27687	Yüksel	2007	Spring	The Music of the Ramones	M	13	0	13	50	180	Saucon	Lembr	Physics
9	27687	Yüksel	2007	Spring	The Music of the Ramones	W	13	0	13	50	180	Saucon	Lembr	Physics
10	18338	Kangs	2007	Spring	The Music of the Ramones	F	13	0	13	50	180	Saucon	Lembr	Physics

4.4 Bài 4

Đề bài: Sinh viên A muốn học môn ‘Mobile Computing’ hỏi A cần phải học qua những môn gì?

Các bước làm: Trong Oracle có hỗ trợ Hierarchical Queries ta truy vấn trực tiếp ra những môn sinh viên A cần học.

Thực hiện:

- Các câu lệnh SQL trong Oracle: Trong oracle có hỗ trợ Hierarchical Queries.

Câu lệnh SQL:

```
select c.course_id,p.prereq_id,c.title as course_chart
from ( course c left outer join prereq p on c.course_id=p.course_id )
start with c.title = 'Mobile Computing'

connect by prior p.prereq_id= c.course_id;
```

Kết quả:

	COURSE_ID	PREREQ_ID	COURSE_CHART
1	612	123	Mobile Computing
2	123		Differential Equations
3	810	966	Mobile Computing
4	966		Sanitary Engineering

- Các câu lệnh SQL trong SQL server:

Bước 1: tạo view nối bảng khóa học với bảng điều kiện học trước:

```
create view CAU_4 as
```

```
select c.course_id,p.prereq_id,c.title as course_chart

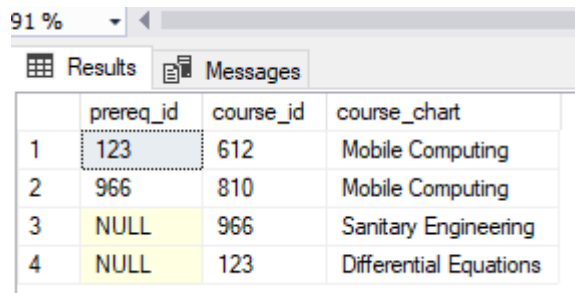
from ( course c left outer join prereq p on c.course_id=p.course_id )
```

Bước 1: Sử dụng with clause tương tự như connect by bên oracle

```
WITH n(prereq_id, course_id, course_chart) AS
  (SELECT prereq_id, course_id, course_chart
   FROM CAU_4
   WHERE course_chart = 'Mobile Computing'
   UNION ALL
   SELECT nplus1.prereq_id, nplus1.course_id, nplus1.course_chart
   FROM CAU_4 as nplus1, n
   WHERE n.prereq_id = nplus1.course_id)

SELECT prereq_id, course_id, course_chart FROM n
```

Kết quả:



	prereq_id	course_id	course_chart
1	123	612	Mobile Computing
2	966	810	Mobile Computing
3	NULL	966	Sanitary Engineering
4	NULL	123	Differential Equations

4.5 Bài 5

Đề bài: Cài đặt Trigger kiểm tra số lượng sinh viên đăng ký vượt quá sức chứa của phòng. Đưa ra thông báo không thành công khi sinh viên đăng ký môn học. Rollback khi có lỗi xảy ra.

Các bước làm: Cài đặt trigger kiểm tra các trường hợp sau:

Trường hợp sinh viên đăng ký rồi thì không được đăng kí nữa.

Trường hợp sinh viên chưa đăng ký môn học thì cho phép đăng ký. Khi đó nếu lớp vẫn còn chỗ thì sinh viên đăng kí thành công không thì rollback lại dữ liệu nhưng ban đầu.

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo triggers dang_ky_tin_chi:

```

CREATE OR REPLACE TRIGGER dang_ki_tin_chi
BEFORE INSERT ON takes FOR each ROW
DECLARE
    so_sv_hien_co NUMBER;
    so_sv_toi_da NUMBER;
    kiem_tra_dk_trung NUMBER;
BEGIN
    SELECT COUNT(*) INTO kiem_tra_dk_trung
    FROM TAKES t
    WHERE (t.ID = :NEW.ID)
        AND (t.COURSE_ID = :NEW.COURSE_ID)
        AND (t.SEC_ID = :NEW.SEC_ID)
        AND (t.SEMESTER = :NEW.SEMESTER)
        AND (t.YEAR = :NEW.YEAR);
    IF (kiem_tra_dk_trung > 0) THEN
        RAISE_APPLICATION_ERROR(-20191, 'sinh vien da dang ky. ');
    ELSE
        SELECT c.CAPACITY INTO so_sv_toi_da
        FROM SECTION s
        JOIN CLASSROOM c using (building, room_number)
        WHERE (s.COURSE_ID = :NEW.COURSE_ID)
            AND (s.SEC_ID = :NEW.SEC_ID)
            AND (s.SEMESTER = :NEW.SEMESTER)
            AND (s.YEAR = :NEW.YEAR);
        SELECT COUNT(*) INTO so_sv_hien_co
        FROM TAKES t
        WHERE (t.COURSE_ID = :NEW.COURSE_ID)
            AND (t.SEC_ID = :NEW.SEC_ID)
            AND (t.SEMESTER = :NEW.SEMESTER)
            AND (t.YEAR = :NEW.YEAR);
        IF (so_sv_hien_co + 1 < so_sv_toi_da) THEN
            DBMS_OUTPUT.PUT_LINE('DK thanh cong ');
        ELSE
            RAISE_APPLICATION_ERROR(-20192, 'lop da day' || so_sv_hien_co || '/' ||
so_sv_toi_da );
        END IF;
    END IF;
END;

```

Chạy thử:

```

INSERT INTO takes ( id, course_id, sec_id, year, semester )
VALUES ( '69730', '313', '1', 2010, 'Fall');

```

Kết quả:


Trường hợp thành công:

SQL

Output


Statistics

```
INSERT INTO takes ( id, course_id, sec_id, year, semester ) VALUES ( '69730', '313', '1', 2010, 'Fall');
select * from takes where id = 69730 and course_id = 313
```



	ID	COURSE_ID	SEC_ID	SEMESTER	YEAR	GRADE
▶ 1	69730	313	1	Fall	2010	

Trường hợp sinh viên đã đăng ký:




ORA-20201: sinh vien da dang ky.
ORA-06512: at "LEARNINGSQL.CHECK_STUDENT_PER_CLASSROOM", line 14
ORA-04088: error during execution of trigger 'LEARNINGSQL.CHECK_STUDENT_PER_CLASSROOM'

[View program sources of error stack?](#)

Trường hợp lớp đầy:

Error



ORA-20192: lop da day269/11
ORA-06512: at "LEARNINGSQL.DANG_KI_TIN_CHI", line 32
ORA-04088: error during execution of trigger 'LEARNINGSQL.DANG_KI_TIN_CHI'
ORA-06512: at "LEARNINGSQL.SP_DK_BAI9", line 10
ORA-06512: at line 3

[View program sources of error stack?](#)

b. Các câu lệnh SQL trong SQL server:

Câu lệnh SQL tạo trigger KiemTraTruocKhiDangKy:

```

USE [CSDLNC]
GO
/***** Object: Trigger [dbo].[KiemTraTruocKhiDangKy]  Script Date: 6/7/2019
4:28:34 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[KiemTraTruocKhiDangKy]
ON [dbo].[takes]
INSTEAD OF INSERT
AS
BEGIN

```

```

DECLARE @lastCapacity int
DECLARE @maxCapacity int
DECLARE @registered int
BEGIN TRANSACTION;
save transaction test_transaction
SELECT @registered = COUNT(*) FROM takes t
      WHERE (t.ID = (SELECT ID FROM inserted WHERE ID = inserted.ID))
      AND (t.course_id = (SELECT course_id FROM inserted WHERE
course_id = inserted.course_id))
      AND (t.sec_id = (SELECT sec_id FROM inserted WHERE sec_id =
inserted.sec_id))
      AND (t.semester = (SELECT semester FROM inserted WHERE semester
= inserted.semester))
      AND (t.year = (SELECT year FROM inserted WHERE year =
inserted.year))
  IF (@registered = 0)
  BEGIN
    insert into takes
    select * from inserted
    SELECT @maxCapacity = c.capacity FROM section s left outer join
classroom c
      ON (s.building = c.building and s.room_number =
c.room_number)
      WHERE (s.course_id = (SELECT course_id FROM inserted
WHERE course_id = inserted.course_id))
      AND (s.sec_id = (SELECT sec_id FROM inserted WHERE sec_id
= inserted.sec_id))
      AND (s.semester = (SELECT semester FROM inserted WHERE
semester = inserted.semester))
      AND (s.year = (SELECT year FROM inserted WHERE year =
inserted.year));
    SELECT @lastCapacity = COUNT(*) FROM takes t
      WHERE (t.course_id = (SELECT course_id FROM inserted
WHERE course_id = inserted.course_id))
      AND (t.sec_id = (SELECT sec_id FROM inserted WHERE sec_id
= inserted.sec_id))
      AND (t.semester = (SELECT semester FROM inserted WHERE
semester = inserted.semester))
      AND (t.year = (SELECT year FROM inserted WHERE year =
inserted.year));
    IF @lastCapacity + 1 <= @maxCapacity
    BEGIN
      PRINT('Dang ky lop thanh cong. ');
    END
  ELSE
  BEGIN
    PRINT('Lop hoc nay da het cho!');
    RAISERROR ('Lop hoc nay da het cho!',1,1)
    ROLLBACK TRANSACTION test_transaction;
  END
END

```

```

ELSE
    PRINT('Sinh vien da dang ky lop hoc nay!');
    RAISERROR ('Sinh vien da dang ky lop hoc nay!',1,1)
    commit

END

```

Chạy thử:

```

INSERT INTO takes ( id, course_id, sec_id, year, semester )
VALUES ( '69730', '313', '1', 2010, 'Fall');

```

Kết quả:

Trường hợp đăng kí thành công:



```

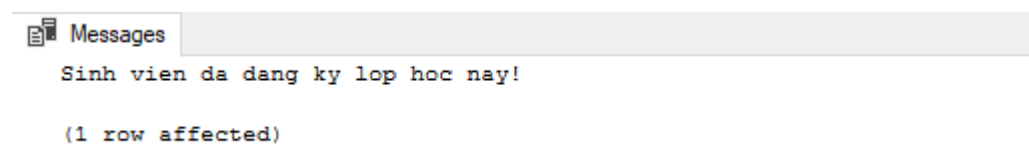
Messages

(1 row affected)
Dang ky lop thanh cong.

(1 row affected)

```

Trường hợp đăng kí rồi:



```

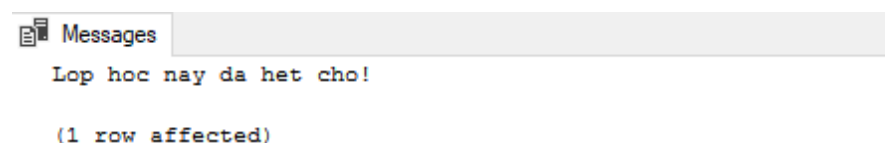
Messages

Sinh vien da dang ky lop hoc nay!

(1 row affected)

```

Trường hợp đăng lớp quá số người:



```

Messages

Lop hoc nay da het cho!

(1 row affected)

```

4.6 Bài 6

Đề bài: Viết thủ tục cho biết kết quả học tập của một sinh viên với:

Đầu vào: Mã sinh viên

Đầu ra: Mã sinh viên, Tên sinh viên, Số tín chỉ tích lũy, Điểm trung bình học kỳ và điểm trung bình tích lũy theo từng học kỳ.

Điều 23. Điểm trung bình học kỳ và điểm trung bình tích lũy

1. Điểm trung bình học kỳ (TBHK) và điểm trung bình tích lũy (TBTL) được tính theo công thức sau (làm tròn đến hai chữ số thập phân):

$$A = \frac{\sum_{i=1}^N a_i \times n_i}{\sum_{i=1}^N n_i}$$

trong đó:

A là điểm trung bình học kỳ hoặc điểm trung bình tích lũy

a_i là điểm học phần thứ i

n_i là số tín chỉ của học phần thứ i

N là số học phần tính điểm trung bình.

Các bước làm:

Bước 1: Chuyển học kỳ từ 2 trường mùa và năm thành 1 trường. ví dụ: 'Fall' và '2018' thành 20182.

Bước 2: Tổng hợp điểm lại với các view và bảng chuyen_he_so_4, course, student lại.

Bước 3: Tạo function để tính số tín chỉ tích lũy.

Bước 4: Tạo function để tính điểm cpa.

Bước 5: Tạo thủ tục show ra điểm trung bình học kỳ và điểm trung bình tích lũy theo từng học kỳ theo ứng với từng học sinh.

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo view conver học kỳ:

```
CREATE OR REPLACE VIEW CONVER_HOC_KI AS
SELECT id,
       course_id,
       sec_id,
       semester,
       year ,
       point,
```

```

CASE
    WHEN semester = 'Fall' THEN CONCAT( year, '2' )
    WHEN semester = 'Spring' THEN CONCAT( year, '1' )
END AS semester_number
FROM chuyen_he_so_4;

```

Bước 2: Tạo view TONG_HOP_DIEM

```

create or replace view tong_hop_diem as
select hk.id, st.name, hk.course_id, hk.Sec_id, hk.semester, hk.year, hk.point,
c.credits, hk.semester_number
  from conver_hoc_ki hk
 join course c on c.course_id = hk.course_id
 join student st on st.id = hk.id;

```

Bước 3: Tạo hàm tính số tín chỉ tích lũy.

```

create or replace function func_tich_luy(student_id NUMBER, semester_max
Number)
  return float is FunctionResult float;
begin
  select sum(pn.credits) into FunctionResult
  from
    (select th.id, th.name, th.course_id, th.credits, max(point) as max_point
     from Tong_hop_diem th
     where TO_NUMBER(th.semester_number) <=
TO_NUMBER(semester_max)
    group by th.course_id, th.name, th.credits, th.id) pn
  where pn.id = student_id
  and pn.max_point >= 1
  group by pn.id, pn.name;
  return (FunctionResult);
end func_tich_luy;

```

Bước 4: Tạo hàm tính cpa.

```

create or replace function func_cpa(student_id NUMBER, semester_max
Number)
  return float is FunctionResult float;
begin
  select round(sum(pn.max_point*pn.credits)/sum(pn.credits), 2) into
FunctionResult
  from
    (select th.id, th.name, th.course_id, th.credits, max(point) as
max_point
     from Tong_hop_diem th
     where TO_NUMBER(th.semester_number) <=
TO_NUMBER(semester_max)
    group by th.course_id, th.name, th.credits, th.id) pn
  where pn.id = student_id

```

```

        group by pn.id, pn.name;
        return (FunctionResult);
    end func_cpa;

```

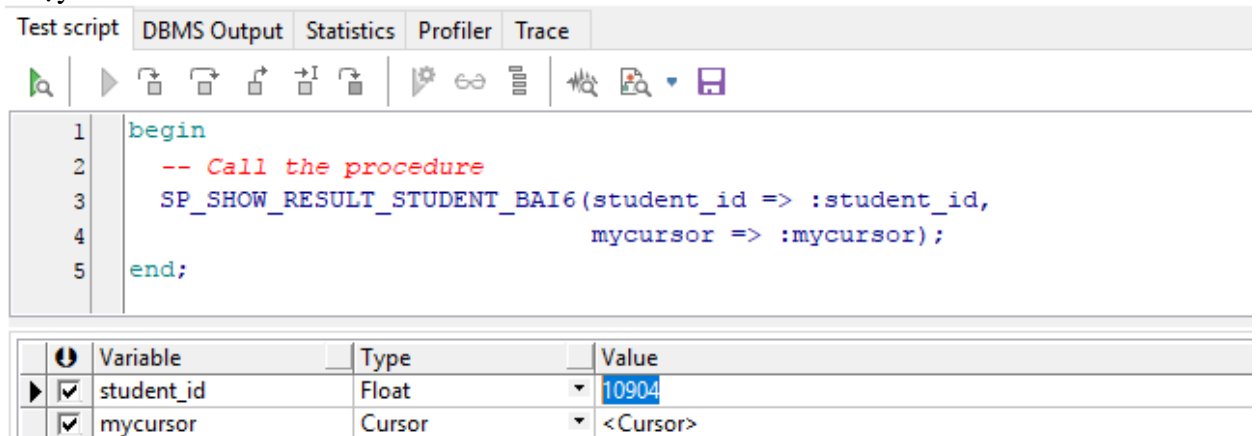
Bước 5: Tạo thủ tục show kết quả học tập của học sinh theo từng kì.

```

CREATE OR REPLACE PROCEDURE
SP_SHOW_RESULT_STUDENT_BAI6 ( student_id NUMBER, mycursor OUT
sys_refcursor )
IS
BEGIN
    OPEN mycursor FOR
        select
            th.id,
            th.name,
            th.semester,
            th.year,
            th.semester_number,
            round(sum(th.point*th.credits)/sum(th.credits) , 2) as gpa,
            sum(th.credits),
            Func_CPA(student_id,th.semester_number) as cpa,
            Func_tich_luy(student_id,th.semester_number) as Tong_tich_luy
        from Tong_hop_diem th
        where id = student_id group by th.semester_number, th.name, th.semester,
        th.year, th.id
        ORDER BY th.semester_number ASC;
    END;

```

Chạy thử:



The screenshot shows the SQL Developer interface. The 'Test script' tab is active, displaying the following SQL code:

```

1  begin
2      -- Call the procedure
3      SP_SHOW_RESULT_STUDENT_BAI6(student_id => :student_id,
4                                  mycursor => :mycursor);
5  end;

```

Below the script, the 'DBMS Output' tab shows the execution results in a table:

Variable	Type	Value
student_id	Float	10904
mycursor	Cursor	<Cursor>

Kết quả:

	ID	NAME	SEMESTER	YEAR	SEMESTER_NUMBER	GPA	SUM(TH.CREDITS)	CPA	TONG_TICH_LUY
1	10904	Jerns	Spring	2002	20021	3	4	3	4
2	10904	Jerns	Fall	2002	20022	3.43	7	3.27	11
3	10904	Jerns	Spring	2003	20031	4.17	9	3.68	20
4	10904	Jerns	Fall	2003	20032	3	4	3.56	24
5	10904	Jerns	Spring	2004	20041	4	3	3.61	27
6	10904	Jerns	Spring	2005	20051	3	4	3.53	31
7	10904	Jerns	Fall	2006	20062	3	4	3.47	35
8	10904	Jerns	Spring	2007	20071	3.5	3	3.47	38
9	10904	Jerns	Fall	2007	20072	2	3	3.37	41
10	10904	Jerns	Spring	2008	20081	3	3	3.34	44
11	10904	Jerns	Spring	2009	20091	1.5	4	3.19	48
12	10904	Jerns	Fall	2009	20092	3.5	6	3.24	51
13	10904	Jerns	Spring	2010	20101	1	6	3.08	51

b. Các câu lệnh SQL trong SQL server:

Bước 1: tạo view đánh số mã học kì:

```
CREATE VIEW CONVER_HOC_KI AS
SELECT id,
       course_id,
       sec_id,
       semester,
       year,
       point,
       CASE
         WHEN semester = 'Fall' THEN CONCAT( year, '2' )
         WHEN semester = 'Spring' THEN CONCAT( year, '1' )
       END AS semester_number
```

```
FROM chuyen_he_so_4;
```

Bước 2: Tạo view tổng hợp điểm thuận tiện cho việc truy vấn (thêm cột mã học kì so với conver hệ số 4):

```
create view tong_hop_diem as
select hk.id, st.name, hk.course_id, hk.Sec_id, hk.semester, hk.year, hk.point,
       c.credits, hk.semester_number
  from conver_hoc_ki hk
 join course c on c.course_id = hk.course_id

 join student st on st.id = hk.id;
```

Bước 3: Tạo function tính cpa:

```
if object_id('func_cpa', 'fn') is not null
  drop function func_cpa;
go
```

```

create function func_cpa(@student_id FLOAT, @semester_max Float)
    returns float as
begin declare @FunctionResult float;

    select @FunctionResult =
round(sum(pn.max_point*pn.credits)/sum(pn.credits), 2)
    from
    (select th.id, th.name, th.course_id, th.credits, max(point) as max_point
    from Tong_hop_diem th
    where th.semester_number <= @semester_max
    group by th.course_id, th.name, th.credits, th.id) pn
    where pn.id = @student_id
    group by pn.id, pn.name;
    return (@FunctionResult);
end;

```

Bước 4: tạo function tính tích lũy:

```

if object_id('func_tich_luy', 'fn') is not null
    drop function func_tich_luy;
go

create function func_tich_luy(@student_id FLOAT, @semester_max Float)
    returns float as
begin declare @FunctionResult float;

    select @FunctionResult = sum(pn.credits)
    from
    (select th.id, th.name, th.course_id, th.credits, max(point) as max_point
    from Tong_hop_diem th
    where th.semester_number <= @semester_max
    group by th.course_id, th.name, th.credits, th.id) pn
    where pn.id = @student_id
    and pn.max_point >= 1
    group by pn.id, pn.name;
    return (@FunctionResult);

end;

```

Bước 5: Tạo thủ tục sử dụng 2 function trên để output ra kết quả cần làm:

```

IF OBJECT_ID('SP_SHOW_RESULT_STUDENT_BAI6', 'P') IS NOT NULL
    DROP PROCEDURE SP_SHOW_RESULT_STUDENT_BAI6;
GO

CREATE PROCEDURE SP_SHOW_RESULT_STUDENT_BAI6 ( @student_id
FLOAT )
AS
BEGIN

```

```

SET NOCOUNT ON;
select
th.id,
th.name,
th.semester,
th.year,
th.semester_number,
round(sum(th.point*th.credits)/sum(th.credits) , 2) as gpa,
sum(th.credits),
dbo.Func_CPA( @student_id,th.semester_number) as cpa,
dbo.Func_tich_luy( @student_id,th.semester_number) as Tong_tich_luy
from Tong_hop_diem th
where id = @student_id group by th.semester_number, th.name, th.semester,
th.year, th.id
ORDER BY th.semester_number ASC;

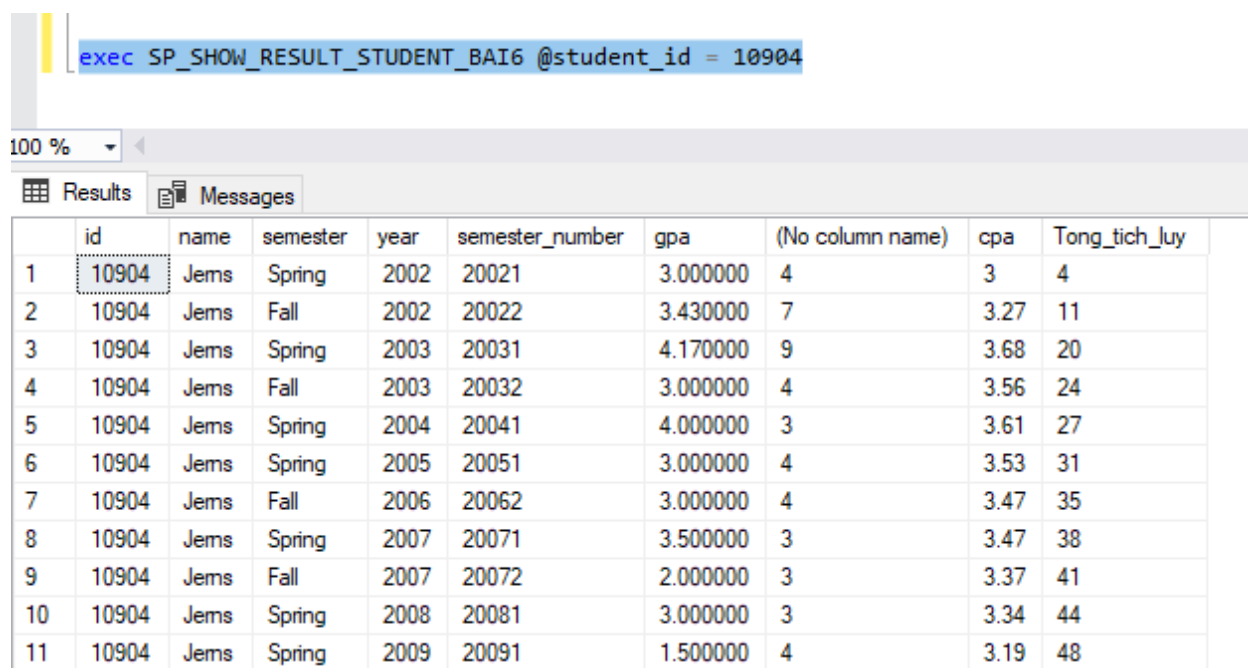
END;

```

Chạy thử:

```
exec SP_SHOW_RESULT_STUDENT_BAI6 @student_id = 10904
```

Kết quả:



The screenshot shows a SQL Server query window with the command `exec SP_SHOW_RESULT_STUDENT_BAI6 @student_id = 10904` executed. Below the command window, the 'Results' tab is active, displaying a table with 10 columns and 11 rows of data. The columns are: id, name, semester, year, semester_number, gpa, (No column name), cpa, and Tong_tich_luy. The data represents the academic performance of student 10904 (Jems) across 11 semesters from 2002 to 2009.

	id	name	semester	year	semester_number	gpa	(No column name)	cpa	Tong_tich_luy
1	10904	Jems	Spring	2002	20021	3.000000	4	3	4
2	10904	Jems	Fall	2002	20022	3.430000	7	3.27	11
3	10904	Jems	Spring	2003	20031	4.170000	9	3.68	20
4	10904	Jems	Fall	2003	20032	3.000000	4	3.56	24
5	10904	Jems	Spring	2004	20041	4.000000	3	3.61	27
6	10904	Jems	Spring	2005	20051	3.000000	4	3.53	31
7	10904	Jems	Fall	2006	20062	3.000000	4	3.47	35
8	10904	Jems	Spring	2007	20071	3.500000	3	3.47	38
9	10904	Jems	Fall	2007	20072	2.000000	3	3.37	41
10	10904	Jems	Spring	2008	20081	3.000000	3	3.34	44
11	10904	Jems	Spring	2009	20091	1.500000	4	3.19	48

4.7 Bài 7

Đề bài: Viết thủ tục đánh giá kết quả học tập của một sinh viên với:

Đầu vào: Mã sinh viên

Đầu ra: Xếp hạng trình độ sinh viên và xếp hạng học lực của sinh viên, biết rằng:

Điều 25. Xếp hạng trình độ và học lực cho sinh viên

1. Căn cứ vào số tín chỉ tích lũy, Nhà trường xếp hạng trình độ cho sinh viên sau mỗi học kỳ như trong Bảng 2.

Bảng 2: Xếp hạng trình độ của sinh viên

Trình độ	Số tín chỉ tích lũy		
	Cao đẳng 3 năm	Đại học 4 năm	Đại học 4,5-5 năm
Sinh viên năm thứ nhất	dưới 32 TC		
Sinh viên năm thứ hai	32 đến dưới 64 TC		
Sinh viên năm thứ ba	từ 64 TC	64 đến dưới 96 TC	
Sinh viên năm thứ tư	–	từ 96 TC	96 đến dưới 128 TC
Sinh viên năm thứ năm	–	–	từ 128 TC

3. Sau mỗi học kỳ, sinh viên được xếp hạng học lực căn cứ vào điểm trung bình tích lũy theo phân loại trong Bảng 3.

Bảng 3: Xếp hạng học lực sinh viên

Học lực	Loại	Điểm trung bình tích lũy
Bình thường	Xuất sắc	từ 3,60 đến 4,00
	Giỏi	từ 3,20 đến 3,59
	Khá	từ 2,50 đến 3,19
	Trung bình	từ 2,00 đến 2,49
Yếu kém	Yếu	từ 1,00 đến 1,99
	Kém	dưới 1,0

Các bước làm:

Bước 1: Tạo 1 hàm chuyển đổi số tín chỉ tích lũy sinh viên ứng với trình độ của sinh viên.

Bước 2: Tạo 1 hàm chuyển đổi số điểm trung bình cpa của sinh viên ra học lực của sinh viên.

Bước 3: Tạo thủ tục show ra kết quả trình độ và học lực của sinh viên (dựa theo hàm thủ tục bài 6).

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1:

```
CREATE OR REPLACE FUNCTION func_trinh_do_sv ( no_credits NUMBER )
RETURN VARCHAR2
IS
grade_point VARCHAR2 ( 20 ) := 'SV nam nhât';
BEGIN
    grade_point :=
    CASE
        WHEN no_credits < 32 THEN 'SV nam nhât'
        WHEN no_credits BETWEEN 32 AND 63 THEN 'SV nam hai'
        WHEN no_credits BETWEEN 64 AND 95 THEN 'SV nam ba'
        WHEN no_credits BETWEEN 96 AND 127 THEN 'SV nam bon'
        WHEN no_credits >= 128 THEN 'SV nam nam'
        ELSE 'Loi chuyen doi'
    END;
    RETURN grade_point;
EXCEPTION
    WHEN others THEN
        dbms_output.put_line ( 'ERR: ' || SQLERRM );
END;
```

Bước 2:

```
CREATE OR REPLACE FUNCTION func_hoc_luc_sv ( avg_credit_point NUMBER )
RETURN VARCHAR2
IS avg_point NUMBER := 0;
grade_point VARCHAR2 ( 20 ) := "";
BEGIN
    avg_point := round( avg_credit_point, 2 );
    grade_point :=
    CASE
        WHEN 3.6 <= avg_point THEN 'Xuat xac'
        WHEN 3.2 <= avg_point AND avg_point < 3.6 THEN 'gioi'
        WHEN 2.5 <= avg_point AND avg_point < 3.2 THEN 'kha'
        WHEN 2.0 <= avg_point AND avg_point < 2.5 THEN 'trung binh'
        WHEN 1.0 <= avg_point AND avg_point < 2.0 THEN 'yeu'
        WHEN avg_point < 1.0 THEN 'Kem'
        ELSE 'Loi chuyen doi!'
    END;
    RETURN grade_point;
EXCEPTION
    WHEN others THEN
        dbms_output.put_line ( 'Err: ' || SQLERRM );
END;
```

Bước 3:

```
CREATE OR REPLACE PROCEDURE SP_SHOW_RESULT_STUDENT_BAI7 (
student_id NUMBER, mycursor OUT sys_refcursor )
IS
```



```

BEGIN
OPEN mycursor FOR
select
th.id,
th.name,
th.semester,
th.year,
th.semester_number,
round(sum(th.point*th.credits)/sum(th.credits) , 2) as gpa,
sum(th.credits),
Func_CPA(student_id,th.semester_number) as cpa,
Func_tich_luy(student_id,th.semester_number) as Tong_tich_luy,
func_trinh_do_sv(Func_tich_luy(student_id,th.semester_number)),
func_hoc_luc_sv(Func_CPA(student_id,th.semester_number))
from Tong_hop_diem th
where id = student_id group by th.semester_number, th.name, th.semester, th.year,
th.id
ORDER BY th.semester_number ASC;
END;

```

Chạy thử:

Test script DBMS Output Statistics Profiler Trace

```

1 begin
2   -- Call the procedure
3   SP_SHOW_RESULT_STUDENT_BAI7(student_id => :student_id,
4                               mycursor => :mycursor);
5 end;

```

Variable	Type	Value
student_id	Float	10904
mycursor	Cursor	<Cursor>

Kết quả:

ID	NAME	SEMESTER	YEAR	SEMESTER_NUMBER	GPA	SUM(TH.CREDITS)	CPA	TONG_TICH_LUY	FUNC_TRINI	FUNC_HOC	
1	10904	Jerns	Spring	2002	20021	3	4	3	4	SV nam nha	kha
2	10904	Jerns	Fall	2002	20022	3.43	7	3.27	11	SV nam nha	gioi
3	10904	Jerns	Spring	2003	20031	4.17	9	3.68	20	SV nam nha	Xuat xac
4	10904	Jerns	Fall	2003	20032	3	4	3.56	24	SV nam nha	gioi
5	10904	Jerns	Spring	2004	20041	4	3	3.61	27	SV nam nha	Xuat xac
6	10904	Jerns	Spring	2005	20051	3	4	3.53	31	SV nam nha	gioi
7	10904	Jerns	Fall	2006	20062	3	4	3.47	35	SV nam hai	gioi
8	10904	Jerns	Spring	2007	20071	3.5	3	3.47	38	SV nam hai	gioi
9	10904	Jerns	Fall	2007	20072	2	3	3.37	41	SV nam hai	gioi
10	10904	Jerns	Spring	2008	20081	3	3	3.34	44	SV nam hai	gioi
11	10904	Jerns	Spring	2009	20091	1.5	4	3.19	48	SV nam hai	kha
12	10904	Jerns	Fall	2009	20092	3.5	6	3.24	51	SV nam hai	gioi
13	10904	Jerns	Spring	2010	20101	1	6	3.08	51	SV nam hai	kha

b. Các câu lệnh SQL trong SQL server:

Bước 1: tạo function chuyển đổi trình độ sv:

```
IF OBJECT_ID('func_trinh_do_sv', 'FN') IS NOT NULL
    DROP FUNCTION func_trinh_do_sv;
GO

CREATE FUNCTION func_trinh_do_sv ( @no_credits FLOAT )
RETURNS VARCHAR(4000)
AS
BEGIN
    DECLARE @grade_point VARCHAR ( 20 ) = 'SV nam nhât';

    SET @grade_point =
    CASE
        WHEN @no_credits < 32 THEN 'SV nam nhât'
        WHEN @no_credits BETWEEN 32 AND 63 THEN 'SV nam hai'
        WHEN @no_credits BETWEEN 64 AND 95 THEN 'SV nam ba'
        WHEN @no_credits BETWEEN 96 AND 127 THEN 'SV nam bon'
        WHEN @no_credits >= 128 THEN 'SV nam nam'
        ELSE 'Loi chuyen doi'
    END;
    RETURN @grade_point;
END;
```

Bước 2: function chuyển đổi học lực sinh viên.

```
IF OBJECT_ID('func_hoc_luc_sv', 'FN') IS NOT NULL
    DROP FUNCTION func_hoc_luc_sv;
GO

CREATE FUNCTION func_hoc_luc_sv ( @avg_credit_point FLOAT )
RETURNS VARCHAR(4000)
AS
BEGIN DECLARE @avg_point FLOAT = 0;
    DECLARE @grade_point VARCHAR ( 20 ) = "";

    SET @avg_point = round( @avg_credit_point, 2 );
    SET @grade_point =
    CASE
        WHEN 3.6 <= @avg_point THEN 'Xuat xac'
        WHEN 3.2 <= @avg_point AND @avg_point < 3.6 THEN 'gioi'
        WHEN 2.5 <= @avg_point AND @avg_point < 3.2 THEN 'kha'
        WHEN 2.0 <= @avg_point AND @avg_point < 2.5 THEN 'trung binh'
        WHEN 1.0 <= @avg_point AND @avg_point < 2.0 THEN 'yeu'
        ELSE 'Loi chuyen doi!'
    END;
    RETURN @grade_point;
```

END;

Bước 3: Tạo thủ tục tương tự như bài 6 thêm 2 cột sử dụng 2 function ở bước 1 và 2:

```
IF OBJECT_ID('SP_SHOW_RESULT_STUDENT_BAI7', 'P') IS NOT NULL
    DROP PROCEDURE SP_SHOW_RESULT_STUDENT_BAI7;
GO

CREATE PROCEDURE SP_SHOW_RESULT_STUDENT_BAI7 ( @student_id
FLOAT )
AS
BEGIN
SET NOCOUNT ON;
    select
    th.id,
    th.name,
    th.semester,
    th.year,
    th.semester_number,
    round(sum(th.point*th.credits)/sum(th.credits) , 2) as gpa,
    sum(th.credits),
    dbo.func_CPA(@student_id,th.semester_number) as cpa,
    dbo.func_tich_luy( @student_id,th.semester_number) as Tong_tich_luy,
    dbo.func_trinh_do_sv(dbo.Func_tich_luy(@student_id,th.semester_number)),
    dbo.func_hoc_luc_sv(dbo.Func_CPA(@student_id,th.semester_number))
    from Tong_hop_diem th
    where id = @student_id group by th.semester_number, th.name, th.semester,
    th.year, th.id
    ORDER BY th.semester_number ASC;

END;
```

Chạy thử:

```
exec SP_SHOW_RESULT_STUDENT_BAI7 @student_id = 10904
```

Kết quả:

100 %											
Results						Messages					
	id	name	semester	year	semester_number	gpa	(No column name)	cpa	Tong_tich_luy	(No column name)	(No column name)
1	10904	Jems	Spring	2002	20021	3.000000	4	3	4	SV nam nhat	kha
2	10904	Jems	Fall	2002	20022	3.430000	7	3.27	11	SV nam nhat	gioi
3	10904	Jems	Spring	2003	20031	4.170000	9	3.68	20	SV nam nhat	Xuat xac
4	10904	Jems	Fall	2003	20032	3.000000	4	3.56	24	SV nam nhat	gioi
5	10904	Jems	Spring	2004	20041	4.000000	3	3.61	27	SV nam nhat	Xuat xac
6	10904	Jems	Spring	2005	20051	3.000000	4	3.53	31	SV nam nhat	gioi
7	10904	Jems	Fall	2006	20062	3.000000	4	3.47	35	SV nam hai	gioi
8	10904	Jems	Spring	2007	20071	3.500000	3	3.47	38	SV nam hai	gioi
9	10904	Jems	Fall	2007	20072	2.000000	3	3.37	41	SV nam hai	gioi
10	10904	Jems	Spring	2008	20081	3.000000	3	3.34	44	SV nam hai	gioi
11	10904	Jems	Spring	2009	20091	1.500000	4	3.19	48	SV nam hai	kha

4.8 Bài 8

Đề bài: Đánh chỉ mục các bảng takes, student, advisor. So sánh tốc độ truy vấn sau khi đã thực hiện đánh chỉ mục.

Các bước làm:

Bước 1: Tạo 3 bảng y hệt 3 bảng takes, student, advisor.

Bước 2: Đánh chỉ mục 3 bảng vừa tạo. Với bảng ADVISOR_INDEX đã có khóa chính là S_ID nên ta đánh index vào I_ID. Ở bảng takes_index đã có ID, Course_id, sec_id, semester, year đều là khóa chính nên ta đánh index trường grade. Cuối cùng với bảng student có khóa chính là ID nên ta đánh index cho 3 cột còn lại name, dept_name, tot_cred.

Bước 3: So sánh tốc độ truy vấn giữa bảng cũ và bảng mới.

Thực hiện:

a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo 3 bảng mới:

```
create table takes_index
(ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
grade varchar(2),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id, sec_id, semester, year) references section
on delete cascade,
foreign key (ID) references student
on delete cascade
```

```

);
create table student_index
(ID    varchar(5),
name   varchar(20) not null,
dept_name  varchar(20),
tot_cred  numeric(3,0) check (tot_cred >= 0),
primary key (ID),
foreign key (dept_name) references department
on delete set null
);
create table advisor_index
(s_ID   varchar(5),
i_ID   varchar(5),
primary key (s_ID),
foreign key (i_ID) references instructor (ID)
on delete set null,
foreign key (s_ID) references student (ID)
on delete cascade) ;

insert into ADVISOR_INDEX select * from Advisor
insert into takes_index select * from takes
insert into student_index select * from student

```

Bước 2: Đánh index cho 3 bảng mới tạo:

```

create index index_for_advisor on ADVISOR_INDEX(I_ID);

create index index_for_takes on takes_index(grade);

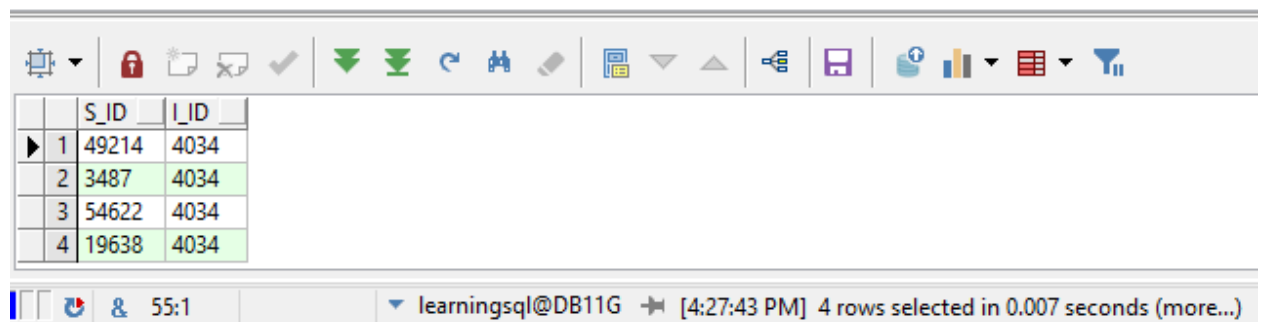
create index index_for_student on student_index(name, dept_name, tot_cred);

```

Chạy thử: Với trường hợp bảng advisor:

Không đánh index:

```
select * from advisor where i_id = 4034
```



The screenshot shows a database interface with a toolbar at the top. Below the toolbar is a table with 4 rows and 2 columns: S_ID and I_ID. The first row is highlighted in black, and the other three rows are highlighted in light green. The status bar at the bottom indicates that 4 rows were selected in 0.007 seconds.

	S_ID	I_ID
1	49214	4034
2	3487	4034
3	54622	4034
4	19638	4034

55:1 | learningsql@DB11G | [4:27:43 PM] 4 rows selected in 0.007 seconds (more...)

Có đánh index:

```
select * from ADVISOR_INDEX where i_id = 4034
```

```
select * from advisor where i_id = 4034
```

	S_ID	I_ID
1	7854	4034
2	1812	4034
3	22258	4034
4	57185	4034

learningsql@DB11G [4:29:21 PM] 4 rows selected in 0.006 seconds (more...)

Với bảng student:

Không đánh index

```
select * from student where name = 'Rowe' and dept_name = 'Geology'
```

	ID	NAME	DEPT_NAME	TOT_CRED
1	40682	Rowe	Geology	114

learningsql@DB11G [4:32:50 PM] 1 row selected in 0.008 seconds

Có đánh index:

```
select * from student_index where name = 'Rowe' and dept_name = 'Geology'
```

```
select * from student where name = 'Rowe' and dept_name = 'Geology'
```

	ID	NAME	DEPT_NAME	TOT_CRED
1	40682	Rowe	Geology	114

learningsql@DB11G [4:33:30 PM] 1 row selected in 0.007 seconds

Với bảng takes:

Không đánh index:

```
select * from takes where GRADE = 'A+'
```

	ID	COURSE_ID	SEC_ID	SEMESTER	YEAR	GRADE
1	68999	169	1	Spring	2007	A+
2	81207	239	1	Fall	2006	A+
3	68010	972	1	Spring	2009	A+
4	15517	489	1	Fall	2007	A+

55:1 learningsql@DB11G [4:35:59 PM] 4 rows selected in 0.011 seconds (more...)

Có đánh index:

```
select * from takes_index where GRADE = 'A+'
select * from takes where GRADE = 'A+'
```

	ID	COURSE_ID	SEC_ID	SEMESTER	YEAR	GRADE
1	79170	137	1	Spring	2002	A+
2	79170	366	1	Fall	2005	A+
3	79170	408	2	Spring	2003	A+
4	79170	603	1	Fall	2003	A+

54:1 learningsql@DB11G [4:36:45 PM] 4 rows selected in 0.009 seconds (more...)

Nhận xét: Việc đánh index cho các bảng có làm tăng tốc độ truy vấn nhưng không nhiều. Nguyên nhân là từ ban đầu các bảng đã được đánh index một số trường nhất định và dữ liệu cũng không quá lớn.

b. Các câu lệnh SQL trong SQL server:

Bước 1: Tạo 3 bảng mới

```
create table takes_test_index(
    ID varchar(5) not null,
    course_id varchar(8) not null,
    sec_id varchar(8) not null,
    semester varchar(6) not null,
    year numeric(4,0) not null,
    grade varchar(2),
    primary key(ID,course_id,sec_id,semester,year)
)
```

```
insert into takes_test_index
select * from takes
```

```
create table student_test_index(
```

```

ID varchar(5) not null,
name varchar(20) not null,
dept_name varchar(20),
tot_cred numeric(3,0)
primary key(ID)
)

```

```

insert into student_test_index
select * from student

```

```

create table advisor_test_index(
s_ID varchar(5) not null,
i_ID varchar(5) not null,
primary key(s_ID)
)

```

```

insert into advisor_test_index
select * from advisor

```

Bước 2: Đánh index

```

create index index_takes_2 on takes_test_index(grade)
create index index_student on student_test_index(name, dept_name, tot_cred)
create index index_adv on advisor_test_index(s_ID)

```

Chạy thử:

Với bảng takes:

```

set statistics time on
select * from takes where grade = 'B'
select * from takes_test_index where grade = 'B'
set statistics time off

```

```

(3334 rows affected)

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 123 ms.
SQL Server parse and compile time:
    CPU time = 0 ms,  elapsed time = 0 ms.

(3334 rows affected)

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 83 ms.
|

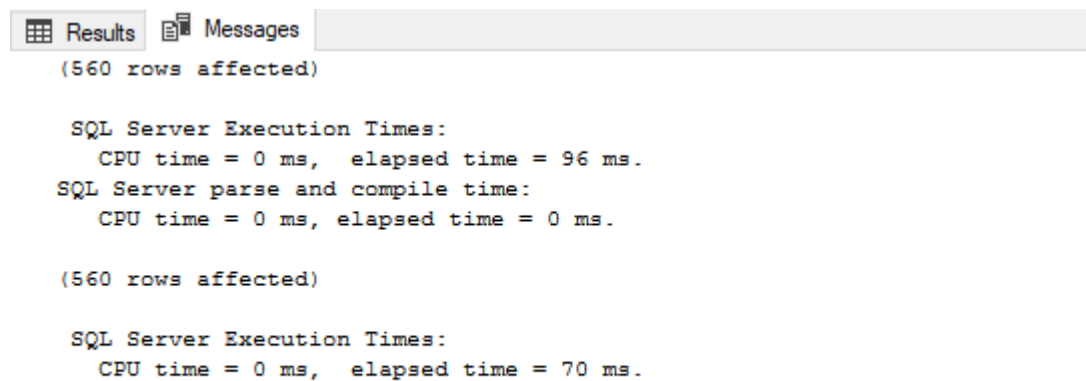
```


Nhận xét: chạy có đánh index nhanh hơn 83 ms < 123 ms nhưng nếu bình thường thì không nhận ra vì tốc độ truy vấn là rất nhanh.

Với bảng student:

```
set statistics time on
select * from student where tot_cred < 40
select * from student_test_index where tot_cred < 40

set statistics time off
```



The screenshot shows the SQL Server Results and Messages window. The 'Results' tab is active, displaying the execution times for two queries. The first query, 'select * from student where tot_cred < 40', took 96 ms. The second query, 'select * from student_test_index where tot_cred < 40', took 70 ms. Both queries affected 560 rows.

```
Results Messages
(560 rows affected)

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 96 ms.
SQL Server parse and compile time:
    CPU time = 0 ms,  elapsed time = 0 ms.

(560 rows affected)

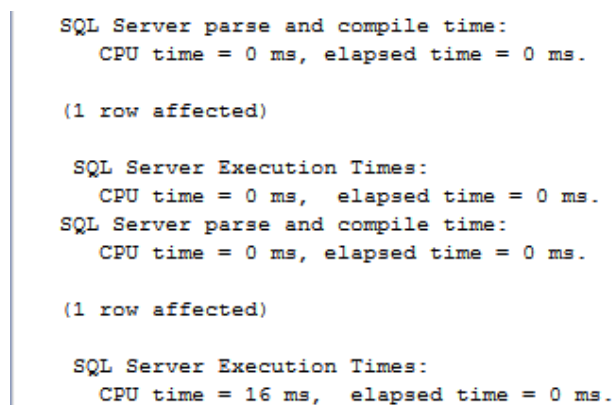
SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 70 ms.
```

Nhận xét: Truy vấn ở bảng đã đánh index nhanh hơn 70ms < 96 ms so với bảng chưa đánh index.

Với bảng advisor:

```
set statistics time on
select * from advisor where s_ID < 40
select * from advisor_test_index where s_ID < 40

set statistics time off
```



The screenshot shows the SQL Server Results and Messages window. The 'Results' tab is active, displaying the execution times for two queries. The first query, 'select * from advisor where s_ID < 40', took 0 ms. The second query, 'select * from advisor_test_index where s_ID < 40', took 16 ms. Both queries affected 1 row.

```
SQL Server parse and compile time:
    CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
    CPU time = 0 ms,  elapsed time = 0 ms.

(1 row affected)

SQL Server Execution Times:
    CPU time = 16 ms,  elapsed time = 0 ms.
```

Nhận xét: việc đánh index ở bảng này gần như không có ý nghĩa , tốc độ truy vấn rất nhanh.

4.9 Bài 9

Đề bài: Viết thủ tục cho phép sinh viên đăng ký khóa học với lựa chọn phòng và thời gian nào đó. Cài đặt các TRANSACTION để đảm bảo toàn vẹn dữ liệu và đưa ra thông báo lỗi khi có lỗi xảy ra.

Các làm: Viết một thủ tục để đăng ký khóa học. Do câu 5 ta đã viết 1 trigger lọc các trường hợp học sinh đã học trong kì đó và lớp hết chỗ lên giờ ta chỉ cần kiểm tra xem kì học đó có môn học đó được mở hay không.

Thực hiện:

- a. Các câu lệnh SQL trong Oracle:

Bước 1: Tạo procedure đăng ký học tập.

```
create or replace procedure SP_DK_BAI9(student_id in varchar2, course_id in
varchar2, sec_id in varchar2,
semester in varchar2, year_ in varchar2, room in varchar2, timer in varchar2)
as
    kiem_tra_room number;
begin
    savepoint TruocDK;
    select count(*) into kiem_tra_room from section where section.course_id =
course_id
    and section.room_number = room and section.year = year_ and
section.time_slot_id = timer;
    if(kiem_tra_room > 0) then
        insert into takes(id, course_id, sec_id, semester, year)
        values (student_id, course_id, sec_id, semester, year_);
    else
        dbms_output.put_line('khong dang ky duoc');
        rollback to TruocDk;
    end if;
end SP_DK_BAI9;
```

Chạy thử:

create table takes_index (... DDL.sql Edit LEARNINGSQL.SP_DK_BAI9@DB11G Script for LEARNINGSQL.SP_DK_BAI9@DB11G

Test script DBMS Output Statistics Profiler Trace

```

1 begin
2   -- Call the procedure
3   SP_DK_BAI9(student_id => :student_id,
4             course_id => :course_id,
5             sec_id => :sec_id,
6             semester => :semester,
7             year_ => :year_,
8             room => :room,
9             timer => :timer);
10 end;

```

Variable	Type	Value
student_id	String	69730
course_id	String	313
sec_id	String	1
semester	String	Fall
year_	String	2010
room	String	804
timer	String	N

Kết quả:

Trường hợp đăng ký thành công:

Test script DBMS Output Statistics Profiler Trace

Clear Buffer size 10000 ☒ Enabled

DK thanh cong

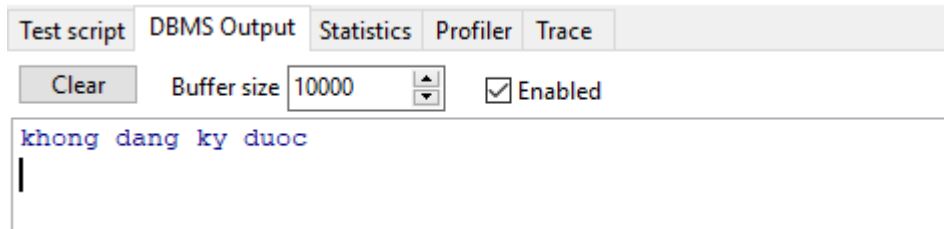
Trường hợp đã đăng ký:

Information

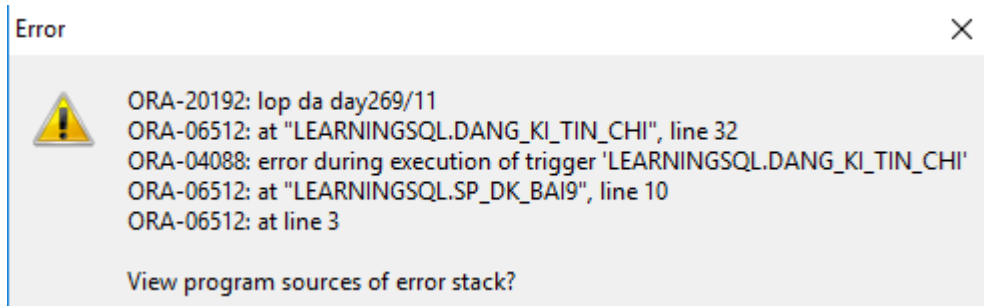
[5:31:29 PM] ORA-20191: sinh vien da dang ky.

OK

Trường hợp không có lớp trong thời gian đó:



Trường hợp lớp đã full sinh viên:



Trường hợp

b. Các câu lệnh SQL trong SQL server:

Bước 1: Tạo procedure kiểm tra đăng ký học tập:

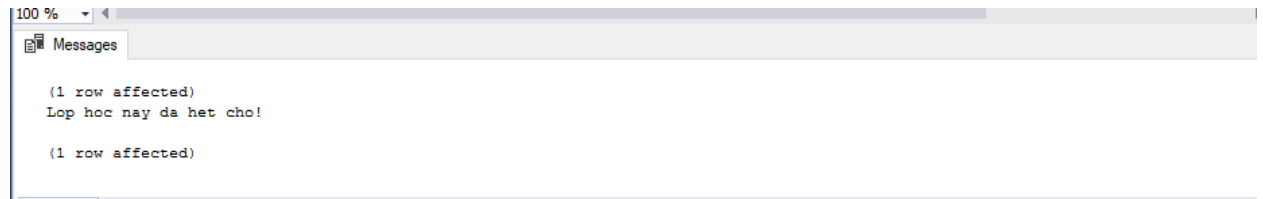
```
CREATE PROCEDURE DangKyHocTap(@student_id varchar(10), @course_id
varchar(5), @sec_id varchar(2),
@semester varchar(10), @year_ numeric, @room varchar(5), @timer varchar(5))
AS
BEGIN
    DECLARE @TruocDk int
    BEGIN TRANSACTION;
    SAVE TRANSACTION TruocKhiDangKy;
    SELECT @TruocDk = COUNT(*) from section where section.course_id =
@course_id
and section.room_number = @room and section.year = @year_ and
section.time_slot_id = @timer;
    IF @TruocDk > 0
    BEGIN
        INSERT INTO takes (ID, course_id, sec_id, semester, year) VALUES(
            @student_id, @course_id, @sec_id, @semester, @year_
        )
    END
    ELSE
        PRINT('Da co loi xay ra, giao tac tu dong quay lui.')
        RAISERROR ('hong co lich hoc.',1,1)
        commit
END
```

Chạy thử:

```
EXEC DangKyHocTap @student_id = '69730', @course_id = '313', @sec_id = '1',  
@semester = 'Fall', @year_ = 2010, @room = '804', @timer = 'N'
```


Kết quả:

Trường hợp hết chỗ:



```
100 %  
Messages  
(1 row affected)  
Lop hoc nay da het cho!  
(1 row affected)
```

Trường hợp đăng kí thành công:



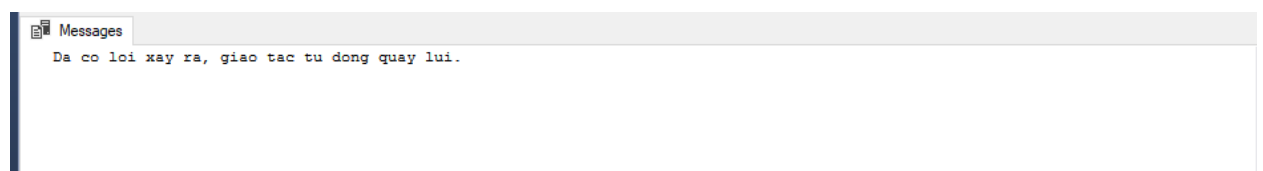
```
100 %  
Messages  
(1 row affected)  
Dang ky lop thanh cong.  
(1 row affected)
```

Trường hợp đã đăng kí rồi:



```
100 %  
Messages  
Sinh vien da dang ky lop hoc nay!  
(1 row affected)
```

Trường hợp không có lớp học để đăng kí:



```
Messages  
Da co loi xay ra, giao tac tu dong quay lui.
```

4.10 Bài 10

Đề bài: Yêu cầu chung: Thiết kế và cài đặt giao diện ứng dụng trên môi trường WinForms kết nối với cơ sở dữ liệu, bằng ngôn ngữ lập trình C# cho phép sinh viên có thể đăng ký học tập theo học chế tín chỉ.

Mô tả chi tiết:

Ứng dụng gồm 2 Forms:

- Form01: Chứa 1 Data Grid 1 bộ lọc. Data Grid hiển thị đầy đủ các trường thông tin: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện. Bộ lọc hỗ trợ lọc theo các trường: Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học. Thực hiện các chức năng xem, thêm, xóa, sửa đối với từng sinh viên, thao tác trên Form 02 dưới đây.

- Form02: Chứa các thông tin về Mã sinh viên, Họ tên sinh viên, Năm học, Kỳ học, Khóa học, Thời gian học, Phòng học, Giảng viên, Khoa viện của từng sinh viên. Cài đặt mối liên kết dữ liệu trên giao diện: chẳng hạn chọn khoa viện thì lọc được danh sách sinh viên thuộc khoa viện, chọn khóa học thì lọc được danh sách giảng viên có thể dạy khóa học đó. Thực hiện cập nhật dữ liệu trên giao diện và lưu vào cơ sở dữ liệu.

Lưu ý: Sinh viên tự lựa chọn và thiết kế các Controls trên Form phù hợp với dữ liệu được mô tả trong cơ sở dữ liệu đã cho trước.

Các bước làm:

Thực hiện:

a) Form 01 lọc sinh viên:

	student_name	year	semester	title	day	start_hr	start_min	end
▶	Zely	2009	Fall	Rock and Roll	F	11	0	11
	Zely	2009	Fall	Rock and Roll	M	11	0	11
	Zely	2009	Fall	Rock and Roll	W	11	0	11
	Zely	2009	Fall	International Trade				
	Zely	2007	Spring	Graph Theory	F	16	0	16
	Zely	2007	Spring	Graph Theory	M	16	0	16
	Zely	2007	Spring	Graph Theory	W	16	0	16
	Zely	2006	Spring	Graph Theory				
	Zely	2006	Spring	Compiler Design				
	Zely	2008	Spring	Embedded Syste...				

b) Form 02 sinh viên:

NguyenDucMinh_20152459_formSV

Khoa Viện: Mã SV: Tên SV:

Mã thời gian học: Năm học: Khoa viện:

Kì học: course_id: Mã giáo viên:

Phong_hoc_DK: Lich_hoc_dk: Tên giáo viên:

ID	student_name	year	semester	title	day	start_hr	start_min	end_hr
1018	Colin	2009	Fall	Image Processing	F	11	0	11
1018	Colin	2009	Fall	Image Processing	M	11	0	11
1018	Colin	2009	Fall	Image Processing	W	11	0	11
1018	Colin	2008	Fall	Elastic Structures	R	14	30	15
1018	Colin	2008	Fall	Elastic Structures	T	14	30	15
1018	Colin	2002	Fall	Drama	F	9	0	9
1018	Colin	2002	Fall	Drama	M	9	0	9
1018	Colin	2002	Fall	Drama	W	9	0	9