

DBMS - Homework 01

PART-01

Q1 [5 pt]

Find the names of all the instructors from Finance department.

Solution:

Relational Algebra Expression:

$\Pi(\text{name}) (\sigma (\text{dept_name} = \text{"finance"}) (\text{instructor}))$

SQL Query:

select name from instructor where dept_name='finance';

Output:

The screenshot shows the MySQL Workbench interface. The 'Query Editor' window contains the following SQL query:

```
1 • select name from instructor where dept_name='finance';
```

The 'Result Grid' window displays the results of the query:

name
Wu
Singh

The 'Table: instructor' window shows the schema for the instructor table:

Columns:	
ID	varchar(5) PK
name	varchar(20)
dept_name	varchar(20)
salary	decimal(8,2)

The 'Output' window shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	18:07:50	select name from instructor where dept_name='finance' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Q2 [5 pt]

Find the names of courses in Biology department which have 4 credits.

Solution:

Relational Algebra Expression:

$\Pi(\text{title})(\sigma(\text{dept_name} = \text{"biology"} \text{ and } \text{credits} = \text{"4"})(\text{course}))$

SQL Query:

select title from course where dept_name='Biology' and credits='4';

Output:

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the query: `select title from course where dept_name='Biology' and credits='4';`. The Results window displays the output of the query, showing two rows: 'Intro. to Biology' and 'Genetics'. The left sidebar shows the database schema for 'lab_university', including tables like 'advisor', 'classroom', 'course', 'department', and 'instructor'. The bottom status bar shows the execution time and the number of rows returned for each query step.

#	Time	Action	Message	Duration / Fetch
13	18:17:44	select * from instructor LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec
14	18:17:44	select * from course LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec
15	18:18:37	select title from course where dept_name='Biology' and credits='4' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Q3 [5 pt]

Display information about all courses whose name start with letter "I".

Solution:

Relational Algebra Expression:

$\Pi(\text{course}) (\sigma (\text{title like 'I\%'}) (\text{course}))$

SQL Query:

select * from course where title like 'I%';

Output:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `select * from course where title like 'I%';`. The Results window displays the following data:

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-319	Image Processing	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
22	19:03:12	select * from course where title like 'I%' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
23	19:03:45	select * from course where title like 'I%' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
24	19:03:46	select * from course where title like 'I%' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
25	19:03:47	select * from course where title like 'I%' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
26	19:03:47	select * from course where title like 'I%' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Q4 [5 pt]

Retrieve all course_id and title of all courses taken by the student with ID 12345.

Solution:

Relational Algebra Expression:

$\Pi(\text{takes.course_id}, \text{course.title})(\sigma(\text{ID} = '12345')(\text{course} \bowtie \text{takes on course_id}))$

SQL Query:

```
select takes.course_id as course_id, course.title as title from course inner join takes on
takes.course_id = course.course_id where ID = '12345';
```

Output:

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 select takes.course_id as course_id, course.title as title
2 from course inner join takes on takes.course_id = course.course_id where ID = '12345';
3
```

The Results window displays the following data:

course_id	title
CS-101	Intro. to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-347	Database System Concepts

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
55	19:32:50	select takes.course_id as course_id, course.title as title from course inner join t...	4 row(s) returned	0.015 sec / 0.000 sec
56	19:32:51	select takes.course_id as course_id, course.title as title from course inner join t...	4 row(s) returned	0.000 sec / 0.000 sec
57	19:33:48	select takes.course_id as course_id, course.title as title from course inner join t...	4 row(s) returned	0.000 sec / 0.000 sec
58	19:33:48	select * from takes where ID='12345' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
59	19:34:06	select takes.course_id as course_id, course.title as title from course inner join t...	4 row(s) returned	0.000 sec / 0.000 sec

Q5 [10 pt]

Retrieve distinct course_id and names of all the courses taught in 2009 Spring and 2010 Spring.

Solution:

Relational Algebra Expression:

$\Pi(\text{section.course_id}, \text{course.title})(\sigma(\text{year in (2009,2010)} \text{ and semester = "Spring"})(\text{course} \bowtie \text{section on course_id}))$

SQL Query:

```
select distinct section.course_id, course.title as title from course inner join section on
course.course_id = section.course_id where year in (2009,2010) and semester = 'Spring';
```

Output:

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 select distinct section.course_id, course.title as title from course inner join
2 section on course.course_id = section.course_id where year in (2009,2010) and semester = 'Spring';
3
4
5
6
7
```

The Results window displays the following data:

course_id	title
CS-101	Intro. to Computer Science
FIN-201	Investment Banking
MU-199	Music Video Production
HIS-351	World History
CS-190	Game Design
CS-319	Image Processing
EE-181	Intro. to Digital Systems
CS-315	Robotics

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	15:07:35	select distinct course_id from section where year in (2009,2010) and semester = 'Spring';	8 row(s) returned	0.000 sec / 0.000 sec
2	15:08:02	select * from section LIMIT 0, 1000	19 row(s) returned	0.000 sec / 0.000 sec
3	15:09:11	select distinct course_id from section where year in (2009,2010) and semester = 'Spring';	8 row(s) returned	0.000 sec / 0.000 sec
4	15:14:53	select distinct section.course_id, course.title as title from course inner join section on course.course_id = section.course_id where year in (2009,2010) and semester = 'Spring';	8 row(s) returned	0.000 sec / 0.000 sec

Q6 [10 pt]

Find course_id and name of all courses that taught by instructor from history department.

Solution:

Relational Algebra Expression:

$\Pi(\text{course.course_id}, \text{course.title})(\sigma_{\text{dept_name} = \text{"History"}}(\text{course} \bowtie \text{instructor on dept_name}) \bowtie \text{teaches on ID})$

SQL Query:

```
select course.course_id, title from course inner join instructor on course.dept_name =  
instructor.dept_name inner join teaches on instructor.ID = teaches.ID where  
instructor.dept_name = 'History';
```

Output:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
9  
10 • select distinct section.course_id, course.title as title from course inner join section  
11 on course.course_id = section.course_id where year in (2009,2010) and semester = 'Spring';  
12  
13  
14 • select course.course_id, title from course  
15 inner join instructor on course.dept_name = instructor.dept_name  
16 inner join teaches on instructor.ID = teaches.ID where instructor.dept_name = 'History';  
17
```

The Results window shows the following output:

course_id	title
HIS-351	World History

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
50	16:33:50	select * from course LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec
51	16:33:52	select * from instructor LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec
52	16:33:55	select * from teaches LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
53	16:34:17	select course.course_id, title from course inner join instructor on course.dept_n...	1 row(s) returned	0.000 sec / 0.000 sec
54	16:35:09	select course course_id, title from course inner join instructor on course.dept_n...	1 row(s) returned	0.016 sec / 0.000 sec

PART-02

Q7[15 pt]

Write SQL DDL statements to create the above tables. Make sure that you capture the primary and foreign key constraints (if applicable), choose appropriate domain (data) type and constraints for each attribute.

Solution:

Salesman Table

SQL Query to create table:

```
create table salesman (salesman_id int,  
                      salesman_name varchar(30),  
                      city varchar(30),  
                      commission decimal(10, 2),  
                      primary key (salesman_id));
```

SQL Query to insert values into the table:

```
insert into salesman values ('5001', 'James Hoog', 'New York','0.15');  
insert into salesman values ('5002', 'Nail Knite', 'Paris','0.13');  
insert into salesman values ('5005', 'Pit Alex', 'London','0.11');  
insert into salesman values ('5006', 'Mc Lyon', 'Paris','0.14');  
insert into salesman values ('5003', 'Lauson Hen', null,'0.12');  
insert into salesman values ('5007', 'Paul Adam', 'Rome','0.13');
```

Output:

The screenshot displays the MySQL Workbench interface. The 'SCHEMAS' pane on the left shows the 'sales' database selected. The main editor window contains the SQL query: `select * from salesman;`. The 'Result Grid' pane shows the output of this query, displaying 6 rows of data for the 'salesman' table. The 'Output' pane at the bottom shows the execution log, including the insertion of 5 rows and the final selection of all 6 rows.

salesman_id	salesman_name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5003	Lauson Hen	NULL	0.12
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13

#	Time	Action	Message	Duration / Fetch
27	19:34:52	insert into salesman values ('5005', 'Pit Alex', 'London','0.11')	1 row(s) affected	0.015 sec
28	19:34:52	insert into salesman values ('5006', 'Mc Lyon', 'Paris','0.14')	1 row(s) affected	0.000 sec
29	19:34:52	insert into salesman values ('5003', 'Lauson Hen', null,'0.12')	1 row(s) affected	0.000 sec
30	19:34:52	insert into salesman values ('5007', 'Paul Adam', 'Rome','0.13')	1 row(s) affected	0.000 sec
31	19:35:18	select * from salesman LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Customer Table

SQL Query to create table:

```
create table customer (customer_id int,  
                        customer_name varchar(30),  
                        city varchar(30),  
                        grade int,  
                        primary key (customer_id));
```

SQL Query to insert values into the table:

```
insert into customer values ('3002', 'Nick Rimando', 'New York', '100');  
insert into customer values ('3005', 'Brad Guzan', 'London', '200');  
insert into customer values ('3001', 'Brad Davis', 'New York', null);  
insert into customer values ('3004', 'Fabian Johns', 'Paris', '300');
```

Output:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'lab_university' expanded, showing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main query editor shows a query: `select * from customer;`. Below the query editor, the 'Result Grid' displays the data from the 'customer' table:

customer_id	customer_name	city	grade
3001	Brad Davis	New York	NULL
3002	Nick Rimando	New York	100
3004	Fabian Johns	Paris	300
3005	Brad Guzan	London	200
NULL	NULL	NULL	NULL

The bottom panel shows the 'Output' tab with a table of execution results:

#	Time	Action	Message	Duration / Fetch
33	19:37:42	insert into customer values ('3002', 'Nick Rimando', 'New York', '100')	1 row(s) affected	0.016 sec
34	19:37:47	insert into customer values ('3005', 'Brad Guzan', 'London', '200')	1 row(s) affected	0.016 sec
35	19:37:51	insert into customer values ('3001', 'Brad Davis', 'New York', null)	1 row(s) affected	0.000 sec
36	19:37:55	insert into customer values ('3004', 'Fabian Johns', 'Paris', '300')	1 row(s) affected	0.016 sec
37	19:38:09	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Order Table

SQL Query to create table:

```
create table orders (order_no int,  
                    purchase_amount decimal(10, 2),  
                    order_date date,  
                    customer_id int,  
                    salesman_id int,  
                    primary key (order_no),  
                    foreign key (customer_id) references customer(customer_id) on delete  
cascade,  
                    foreign key (salesman_id) references salesman(salesman_id) on delete cascade  
);
```

SQL Query to insert values into the table:

```
insert into orders values ('7001', '150.5', '2016-06-27','3005','5002');  
insert into orders values ('7002', '270.65', '2016-07-17','3002','5001');  
insert into orders values ('7003', '110.5', '2016-07-27','3005','5002');  
insert into orders values ('7004', '948.5', '2016-10-05','3004','5006');  
insert into orders values ('7005', '250.45', '2016-10-10','3001',null);  
insert into orders values ('7006', '79.29', '2016-10-10','3002','5001');
```

Output:

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'lab_university' database with tables 'customer' and 'orders'. The 'orders' table structure is detailed in the 'Table: orders' pane, showing columns: order_no (int PK), purchase_amount (decimal(10,2)), order_date (date), customer_id (int), and salesman_id (int). The main editor shows a SQL query: 'select * from orders;'. Below the query, the 'Result Grid' displays 6 rows of data. The 'Output' pane at the bottom shows the execution log with 5 messages, including 4 insertions and 1 select query.

order_no	purchase_amount	order_date	customer_id	salesman_id
7001	150.50	2016-06-27	3005	5002
7002	270.65	2016-07-17	3002	5001
7003	110.50	2016-07-27	3005	5002
7004	948.50	2016-10-05	3004	5006
7005	250.45	2016-10-10	3001	NULL
7006	79.29	2016-10-10	3002	5001

#	Time	Action	Message	Duration / Fetch
46	20:40:07	insert into orders values ('7003','110.5','2016-07-27','3005','5002')	1 row(s) affected	0.000 sec
47	20:40:07	insert into orders values ('7004','948.5','2016-10-05','3004','5006')	1 row(s) affected	0.016 sec
48	20:40:07	insert into orders values ('7005','250.45','2016-10-10','3001',null)	1 row(s) affected	0.000 sec
49	20:40:07	insert into orders values ('7006','79.29','2016-10-10','3002','5001')	1 row(s) affected	0.000 sec
50	20:41:24	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Q8 [5 pt]

Retrieve customer id of all customers from orders table without any repeats.

Solution:

Relational Algebra Expression:

$\Pi(\text{customer_id})(\sigma(\text{orders}))$

SQL Query:

```
select distinct customer_id from orders;
```

Output:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `select distinct customer_id from orders;`. The result grid displays the following data:

customer_id
3001
3002
3004
3005

The left sidebar shows the database schema for 'lab_university', with the 'orders' table selected. The 'orders' table structure is shown below:

Column	Type
order_no	int PK
purchase_amount	decimal(10,2)
order_date	date
customer_id	int
salesman_id	int

The bottom output pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
51	20:45:53	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
52	20:46:22	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
53	20:46:51	select distinct customer_id from orders LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
54	20:47:40	select * from orders LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
55	20:47:49	select distinct customer_id from orders LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Q9 [5 pt]

Display the names and city of salesman, who doesn't belong to the city of Paris.

Solution:

Relational Algebra Expression:

$\Pi(\text{customer_name}, \text{city})(\sigma(\text{city} \neq \text{Paris})(\text{customer}))$

SQL Query:

`select customer_name, city from customer where not city = 'Paris';`

Output:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 select *from customer;
2 select customer_name, city from customer where not city = 'Paris';
3
```

The Results window displays the output of the second query:

customer_name	city
Brad Davis	New York
Nick Rimando	New York
Brad Guzan	London

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
55	20:47:49	select distinct customer_id from orders LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
56	21:10:46	select *from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
57	21:12:29	select customer_name, city from customer where not city = 'Paris' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
58	21:12:34	select *from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
59	21:12:44	select customer_name, city from customer where not city = 'Paris' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Q10 [5 pt] Find that customer with all information who does not get any grade except NULL.

Solution:

Relational Algebra Expression:

$\Pi(\text{customer_id}, \text{customer_name}, \text{city})(\sigma(\text{grade} = \text{Null})(\text{customer}))$

SQL Query:

select customer_id, customer_name, city from customer where grade is null;

Output:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view showing 'lab_university' and 'sales' databases. Under 'sales', the 'customer' table is selected, showing its columns: order_no, purchase_amount, order_date, customer_id, and salesman_id. The main editor window shows a SQL query: `select customer_id, customer_name, city from customer where grade is null;`. The 'Result Grid' shows the query results with columns 'customer_id', 'customer_name', and 'city'. The first row is highlighted, showing '3001', 'Brad Davis', and 'New York'. The bottom panel shows the 'Output' tab with a table of query actions and their results. The table has columns: #, Time, Action, Message, and Duration / Fetch. The last row (61) shows the execution of the query: 'select customer_id, customer_name, city from customer where grade is null LIMIT 0, 1000', returning 1 row(s).

#	Time	Action	Message	Duration / Fetch
57	21:12:29	select customer_name, city from customer where not city = 'Paris' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
58	21:12:34	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
59	21:12:44	select customer_name, city from customer where not city = 'Paris' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
60	21:28:43	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
61	21:29:31	select customer_id, customer_name, city from customer where grade is null LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Q11[20 pt] Consider the following relational algebra expression.

$$\pi_{\text{salesman_id,name,city,commission}}(\sigma_{(\text{commission} \geq 0.10)} \text{salesman}) \cap (\text{commission} \leq 0.12)$$

- 1) How many attributes will the result have?

Solution: Salesman_id , name , city and commission → 4 attributes.

- 2) Write in English what question the expression is trying to answer (e.g. describe what would be the result of the expression).

Solution:

Select salesman id, name, city, and commission from salesman table where commission is greater than or equal to 0.10 and commission is lesser than or equal to 0.12. The output will have four columns that is salesman id, name, city and commission. The row data will be based on the condition → commission is greater than or equal to 0.10 and commission is lesser than or equal to 0.12. In simpler terms salesman id, name and city of the individual will be displayed based on the above condition.

- 3) Translate the expressions into SQL.

Solution:

```
select salesman_id,name,city,commission from  
salesman where commission >=0.10 and commission<=0.12
```