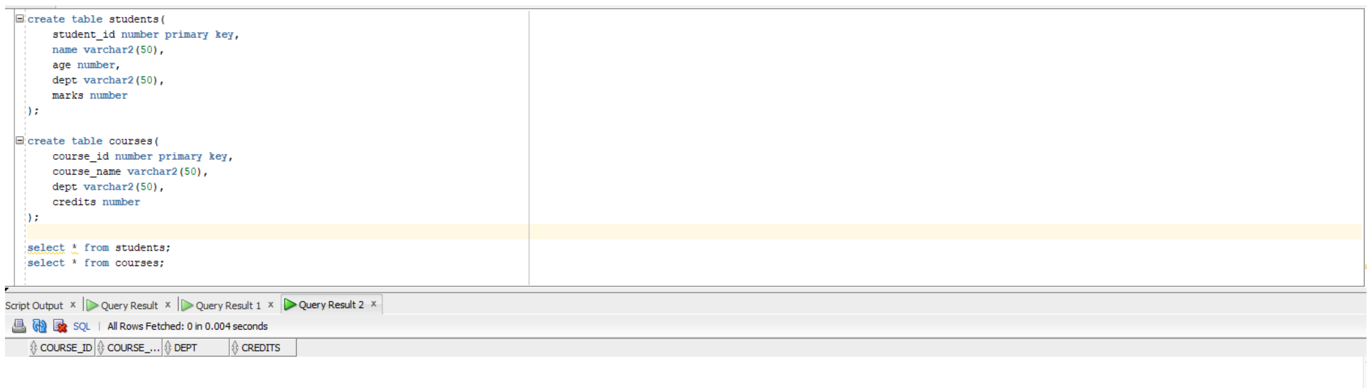
DBMS LAB 7

Siddharth Karmokar

- 123CS0061

Assignment 1

Step 1

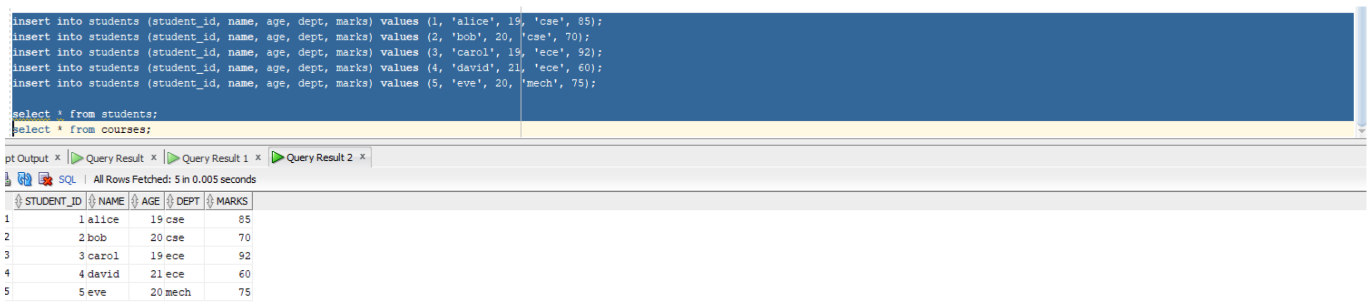


```
create table students(
    student_id number primary key,
    name varchar2(50),
    age number,
    dept varchar2(50),
    marks number
);

create table courses(
    course_id number primary key,
    course_name varchar2(50),
    dept varchar2(50),
    credits number
);

select * from students;
select * from courses;
```

Script Output | Query Result | Query Result 1 | Query Result 2

All Rows Fetched: 0 in 0.004 seconds

COURSE_ID	COURSE_...	DEPT	CREDITS

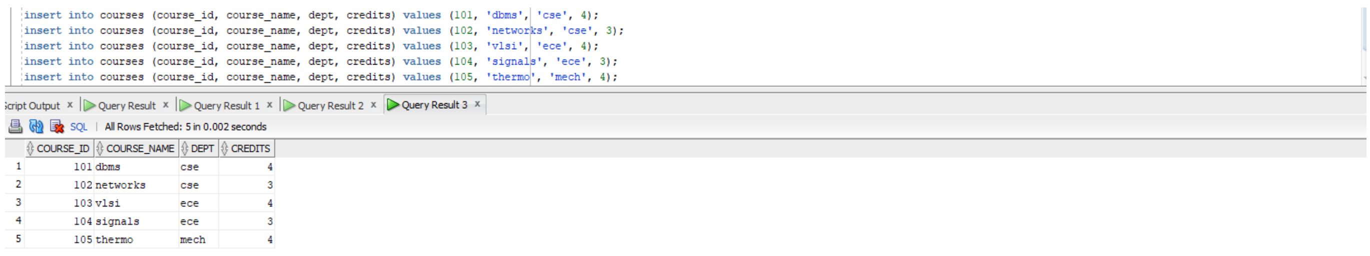


```
insert into students (student_id, name, age, dept, marks) values (1, 'alice', 19, 'cse', 85);
insert into students (student_id, name, age, dept, marks) values (2, 'bob', 20, 'cse', 70);
insert into students (student_id, name, age, dept, marks) values (3, 'carol', 19, 'ece', 92);
insert into students (student_id, name, age, dept, marks) values (4, 'david', 21, 'ece', 60);
insert into students (student_id, name, age, dept, marks) values (5, 'eve', 20, 'mech', 75);

select * from students;
select * from courses;
```

pt Output | Query Result | Query Result 1 | Query Result 2

All Rows Fetched: 5 in 0.005 seconds

	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	2	bob	20	cse	70
3	3	carol	19	ece	92
4	4	david	21	ece	60
5	5	eve	20	mech	75



```
insert into courses (course_id, course_name, dept, credits) values (101, 'dbms', 'cse', 4);
insert into courses (course_id, course_name, dept, credits) values (102, 'networks', 'cse', 3);
insert into courses (course_id, course_name, dept, credits) values (103, 'vlsi', 'ece', 4);
insert into courses (course_id, course_name, dept, credits) values (104, 'signals', 'ece', 3);
insert into courses (course_id, course_name, dept, credits) values (105, 'thermo', 'mech', 4);
```

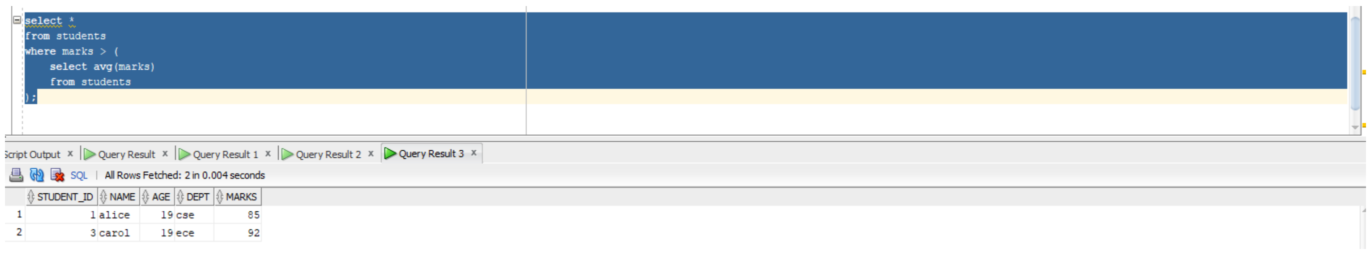
Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

All Rows Fetched: 5 in 0.002 seconds

	COURSE_ID	COURSE_NAME	DEPT	CREDITS
1	101	dbms	cse	4
2	102	networks	cse	3
3	103	vlsi	ece	4
4	104	signals	ece	3
5	105	thermo	mech	4

Step 3

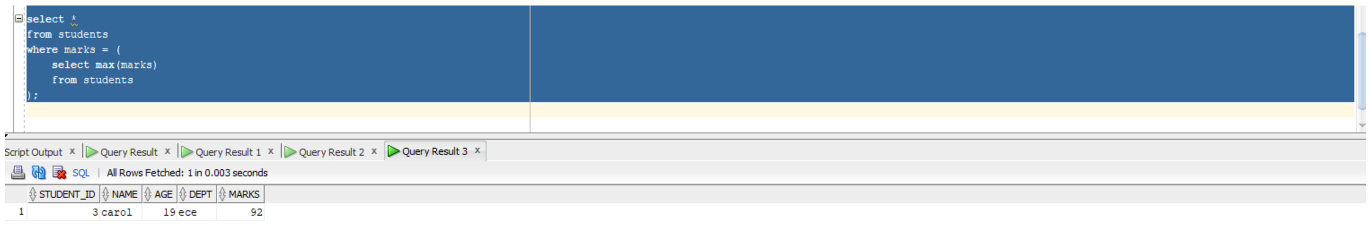
Part A: Basic SubQueries



```sql
select *
from students
where marks > (
    select avg(marks)
    from students
);
```

All Rows Fetched: 2 in 0.004 seconds

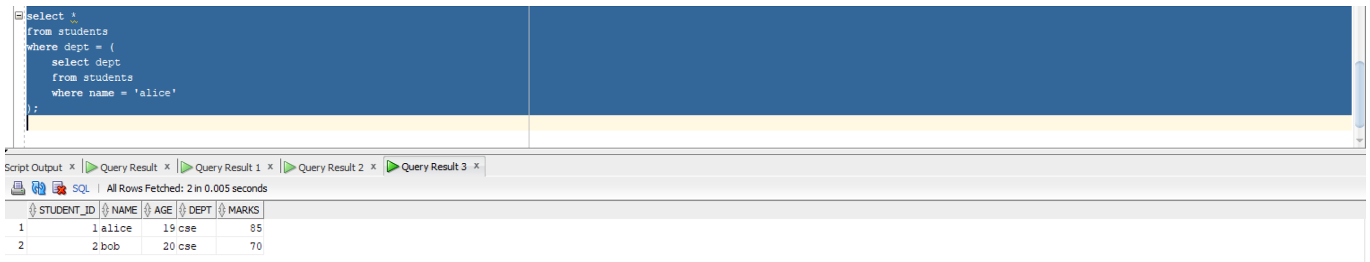
	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	3	carol	19	ece	92



```sql
select *
from students
where marks = (
    select max(marks)
    from students
);
```

All Rows Fetched: 1 in 0.003 seconds

	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	3	carol	19	ece	92

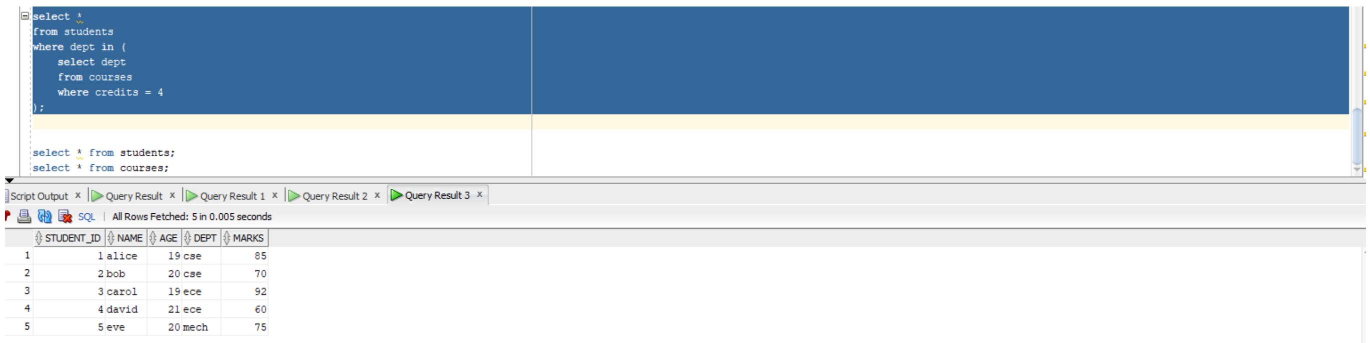


```sql
select *
from students
where dept = (
    select dept
    from students
    where name = 'alice'
);
```

All Rows Fetched: 2 in 0.005 seconds

	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	2	bob	20	cse	70

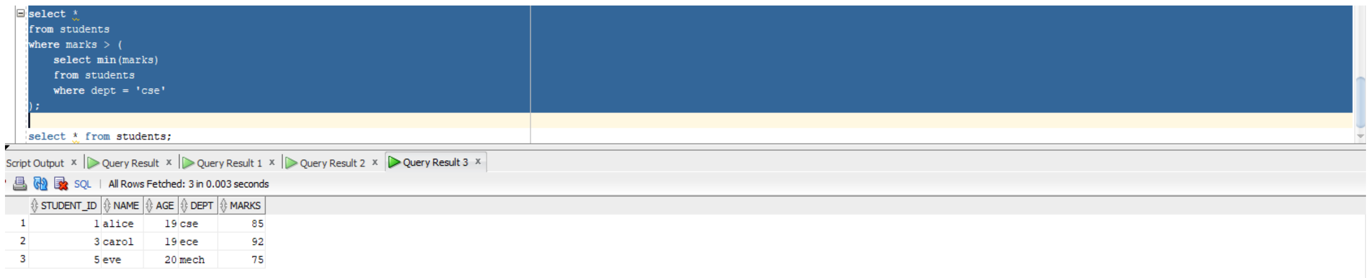
Part B: Multi-row Subqueries



```sql
select *
from students
where dept in (
    select dept
    from courses
    where credits = 4
);


select * from students;
select * from courses;
```

All Rows Fetched: 5 in 0.005 seconds

	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	2	bob	20	cse	70
3	3	carol	19	ece	92
4	4	david	21	ece	60
5	5	eve	20	mech	75

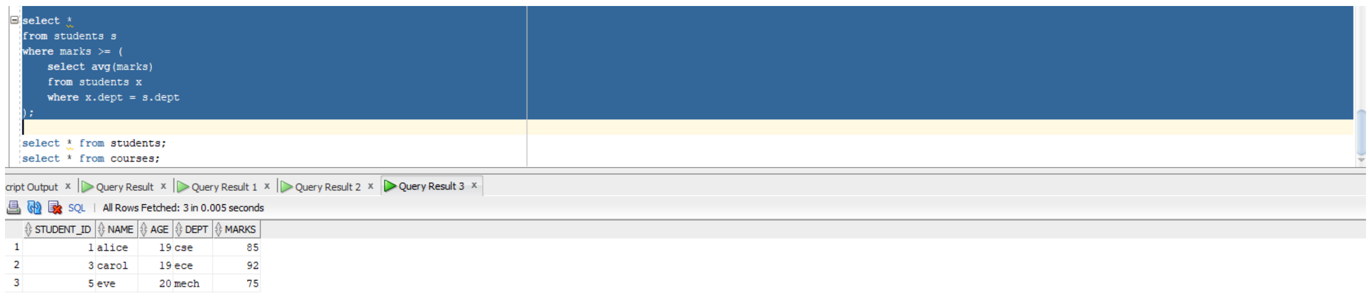


```sql
select *
from students
where marks > (
    select min(marks)
    from students
    where dept = 'cse'
);

select * from students;
```

All Rows Fetched: 3 in 0.003 seconds

	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	3	carol	19	ece	92
3	5	eve	20	mech	75

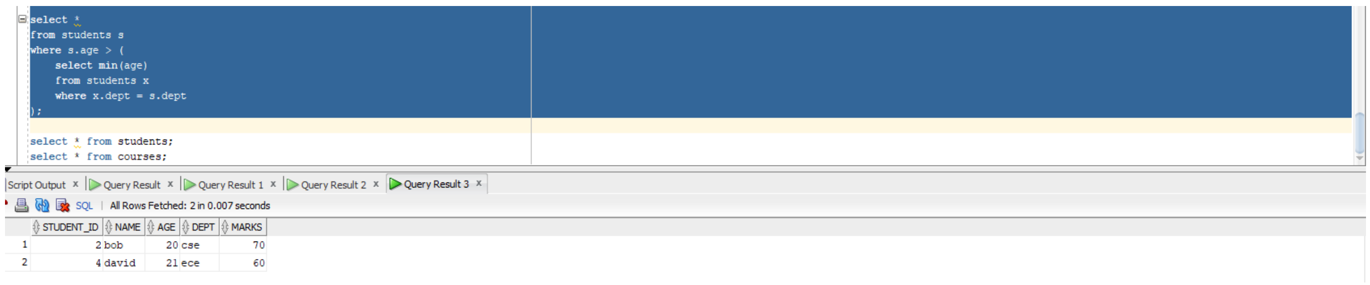
Part C: Correlated Subqueries



```sql
select *
from students s
where marks >= (
    select avg(marks)
    from students x
    where x.dept = s.dept
);

select * from students;
select * from courses;
```

All Rows Fetched: 3 in 0.005 seconds

```
  STUDENT_ID NAME  AGE DEPT MARKS
1          1 alice  19 cse     85
2          3 carol  19 ece     92
3          5 eve    20 mech    75
```

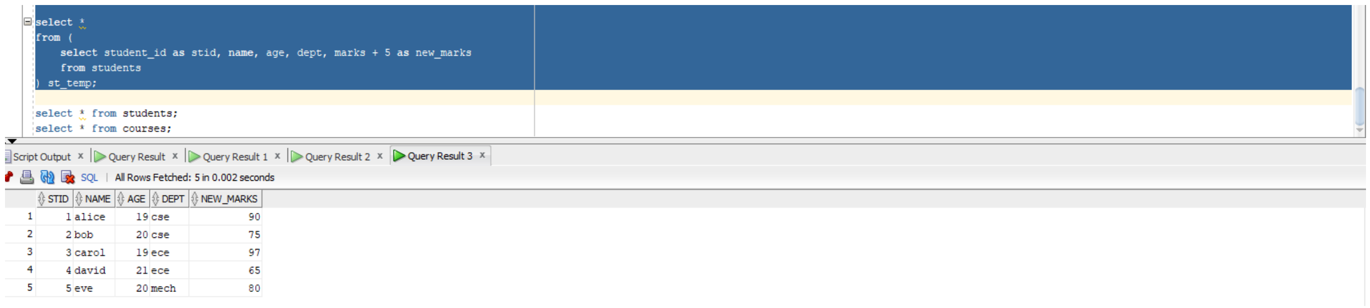


```sql
select *
from students s
where s.age > (
    select min(age)
    from students x
    where x.dept = s.dept
);

select * from students;
select * from courses;
```

All Rows Fetched: 2 in 0.007 seconds

```
  STUDENT_ID NAME  AGE DEPT MARKS
1          2 bob    20 cse     70
2          4 david  21 ece     60
```

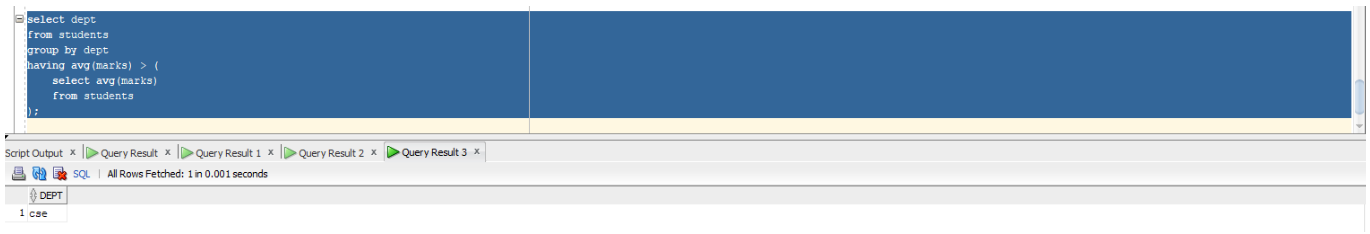
Part D: Placement Subqueries



```sql
select *
from (
    select student_id as stid, name, age, dept, marks + 5 as new_marks
    from students
) st_temp;

select * from students;
select * from courses;
```

All Rows Fetched: 5 in 0.002 seconds

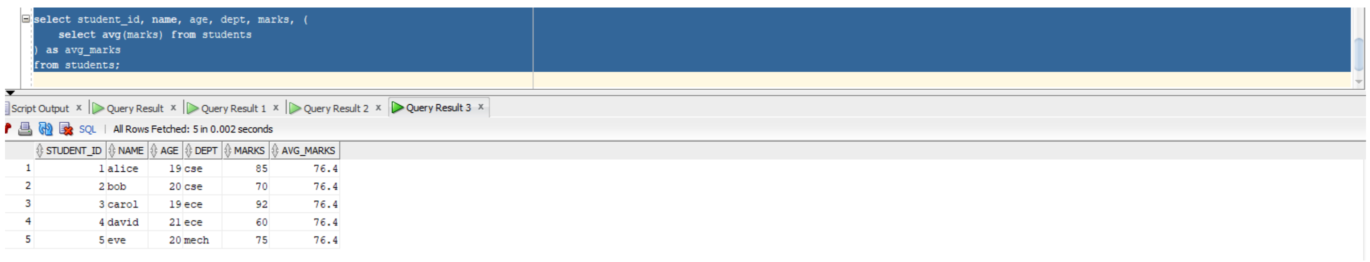
```
  STID NAME  AGE DEPT NEW_MARKS
1    1 alice  19 cse         90
2    2 bob    20 cse         75
3    3 carol  19 ece         97
4    4 david  21 ece         65
5    5 eve    20 mech        80
```



```sql
select dept
from students
group by dept
having avg(marks) > (
    select avg(marks)
    from students
);
```

All Rows Fetched: 1 in 0.001 seconds

```
  DEPT
1 cse
```

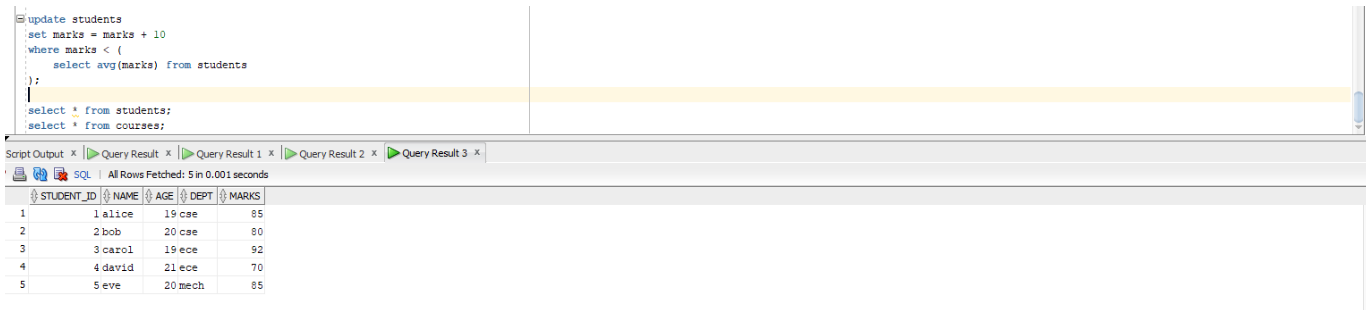


```sql
select student_id, name, age, dept, marks, (
    select avg(marks) from students
) as avg_marks
from students;
```

All Rows Fetched: 5 in 0.002 seconds

```
  STUDENT_ID NAME  AGE DEPT MARKS AVG_MARKS
1          1 alice  19 cse     85      76.4
2          2 bob    20 cse     70      76.4
3          3 carol  19 ece     92      76.4
4          4 david  21 ece     60      76.4
5          5 eve    20 mech    75      76.4
```

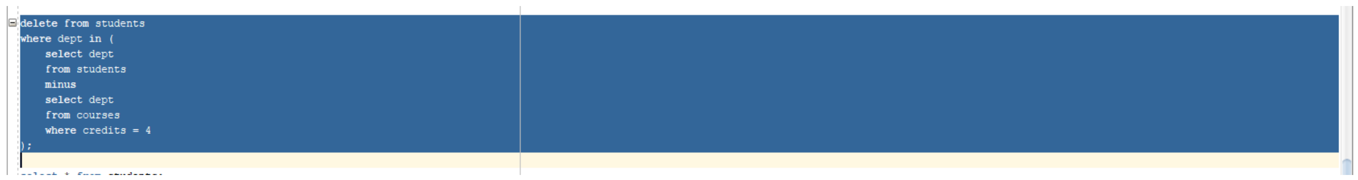
Part E: Placement Subqueries



```
update students
set marks = marks + 10
where marks < (
    select avg(marks) from students
);

select * from students;
select * from courses;
```

All Rows Fetched: 5 in 0.001 seconds

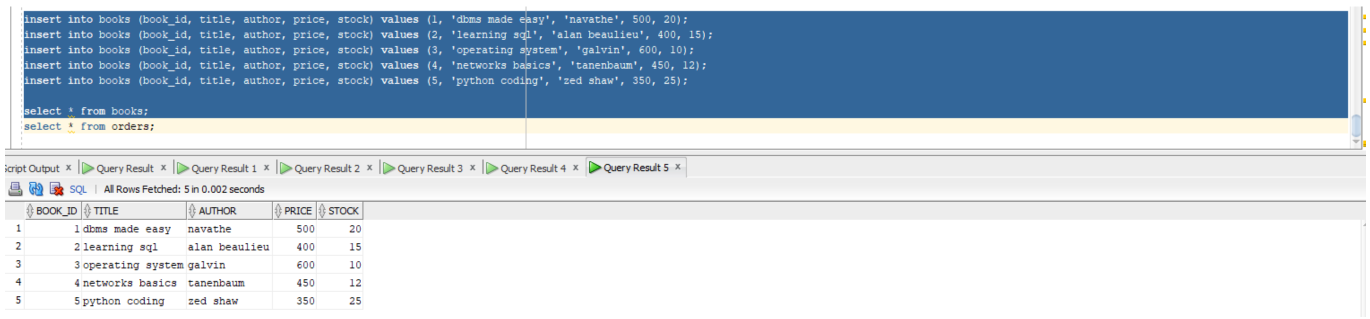
	STUDENT_ID	NAME	AGE	DEPT	MARKS
1	1	alice	19	cse	85
2	2	bob	20	cse	80
3	3	carol	19	ece	92
4	4	david	21	ece	70
5	5	eve	20	mech	85



```
delete from students
where dept in (
    select dept
    from students
    minus
    select dept
    from courses
    where credits = 4
);

select * from students;
```

Assignment 2

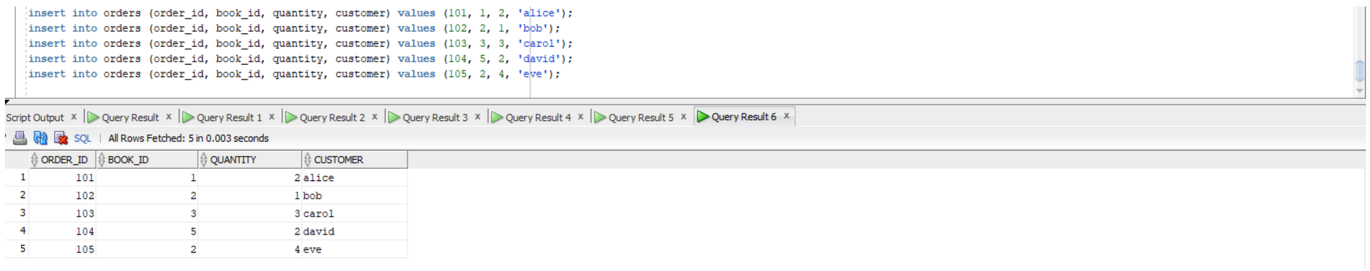
Step 1



```
insert into books (book_id, title, author, price, stock) values (1, 'dbms made easy', 'navathe', 500, 20);
insert into books (book_id, title, author, price, stock) values (2, 'learning sql', 'alan beaulieu', 400, 15);
insert into books (book_id, title, author, price, stock) values (3, 'operating system', 'galvin', 600, 10);
insert into books (book_id, title, author, price, stock) values (4, 'networks basics', 'tanenbaum', 450, 12);
insert into books (book_id, title, author, price, stock) values (5, 'python coding', 'zed shaw', 350, 25);

select * from books;
select * from orders;
```

All Rows Fetched: 5 in 0.002 seconds

	BOOK_ID	TITLE	AUTHOR	PRICE	STOCK
1	1	dbms made easy	navathe	500	20
2	2	learning sql	alan beaulieu	400	15
3	3	operating system	galvin	600	10
4	4	networks basics	tanenbaum	450	12
5	5	python coding	zed shaw	350	25



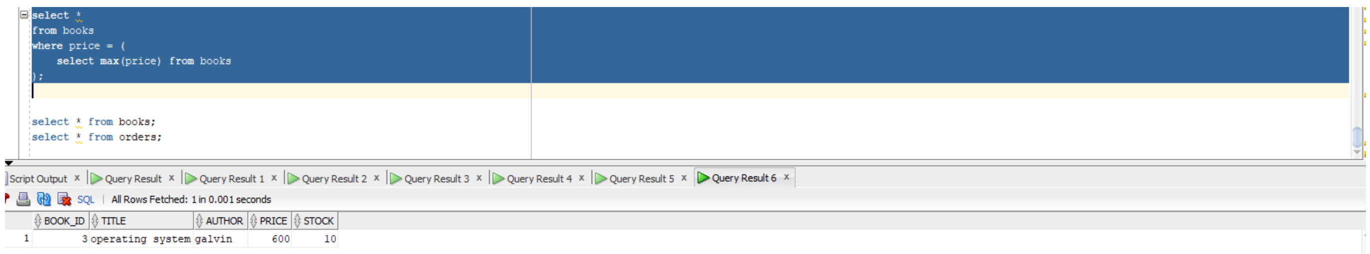
```
insert into orders (order_id, book_id, quantity, customer) values (101, 1, 2, 'alice');
insert into orders (order_id, book_id, quantity, customer) values (102, 2, 1, 'bob');
insert into orders (order_id, book_id, quantity, customer) values (103, 3, 3, 'carol');
insert into orders (order_id, book_id, quantity, customer) values (104, 5, 2, 'david');
insert into orders (order_id, book_id, quantity, customer) values (105, 2, 4, 'eve');
```

All Rows Fetched: 5 in 0.003 seconds

	ORDER_ID	BOOK_ID	QUANTITY	CUSTOMER
1	101	1	2	alice
2	102	2	1	bob
3	103	3	3	carol
4	104	5	2	david
5	105	2	4	eve

Step 3

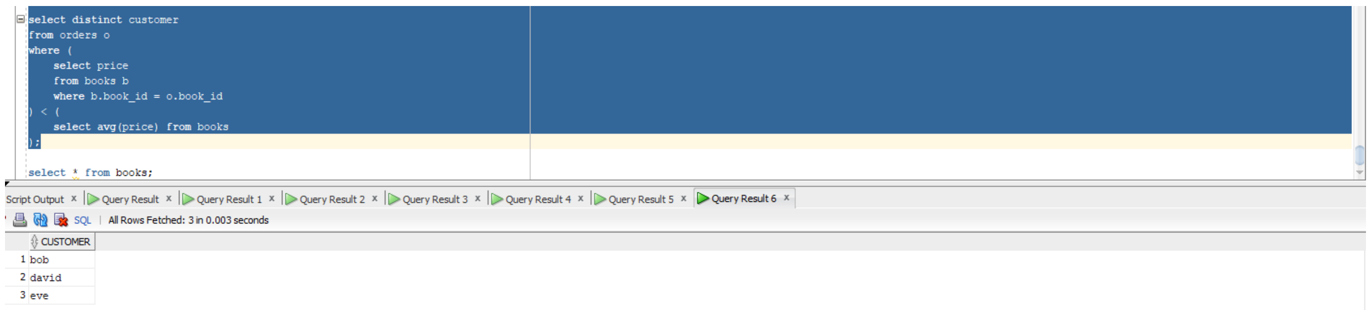
Part A: Basic SubQueries



```sql
select *
from books
where price = (
    select max(price) from books
);

select * from books;
select * from orders;
```

All Rows Fetched: 1 in 0.001 seconds

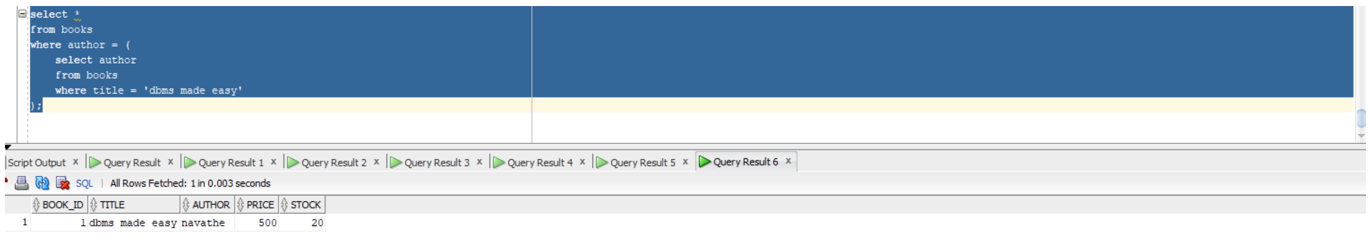
```
  BOOK_ID TITLE            AUTHOR PRICE STOCK
1       3 operating system galvin   600    10
```



```sql
select distinct customer
from orders o
where (
    select price
    from books b
    where b.book_id = o.book_id
) < (
    select avg(price) from books
);

select * from books;
```

All Rows Fetched: 3 in 0.003 seconds

```
  CUSTOMER
1 bob
2 david
3 eve
```

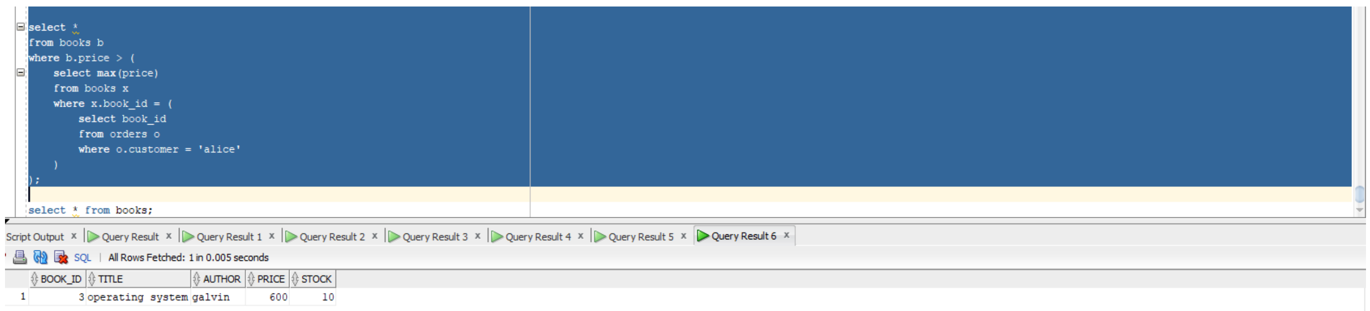


```sql
select *
from books
where author = (
    select author
    from books
    where title = 'dbms made easy'
);
```

All Rows Fetched: 1 in 0.003 seconds

```
  BOOK_ID TITLE          AUTHOR  PRICE STOCK
1       1 dbms made easy navathe   500    20
```

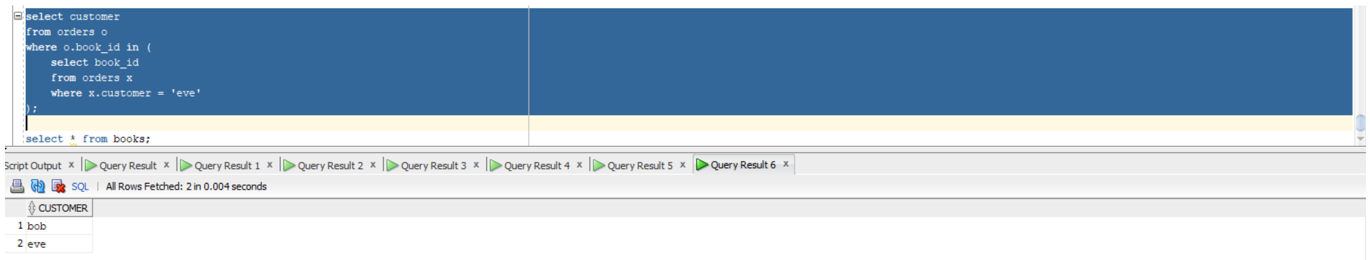
Part B: Multi-row Subqueries



```sql
select *
from books b
where b.price > (
    select max(price)
    from books x
    where x.book_id = (
        select book_id
        from orders o
        where o.customer = 'alice'
    )
);

select * from books;
```

All Rows Fetched: 1 in 0.005 seconds

```
  BOOK_ID TITLE            AUTHOR PRICE STOCK
1       3 operating system galvin   600    10
```

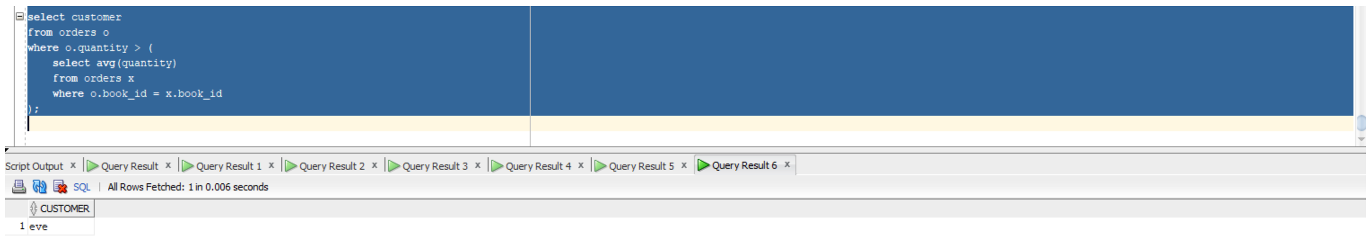


```sql
select customer
from orders o
where o.book_id in (
    select book_id
    from orders x
    where x.customer = 'eve'
);

select * from books;
```

All Rows Fetched: 2 in 0.004 seconds

```
  CUSTOMER
1 bob
2 eve
```

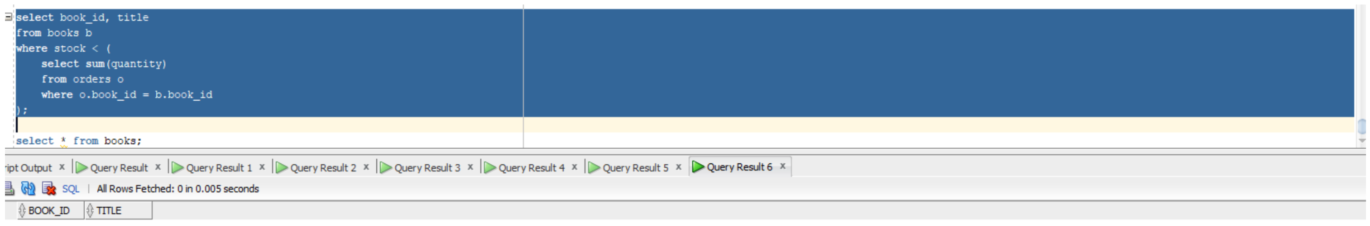
Part C: Correlated Subqueries



```sql
select customer
from orders o
where o.quantity > (
    select avg(quantity)
    from orders x
    where o.book_id = x.book_id
);
```

All Rows Fetched: 1 in 0.006 seconds

```
  CUSTOMER
1 eve
```

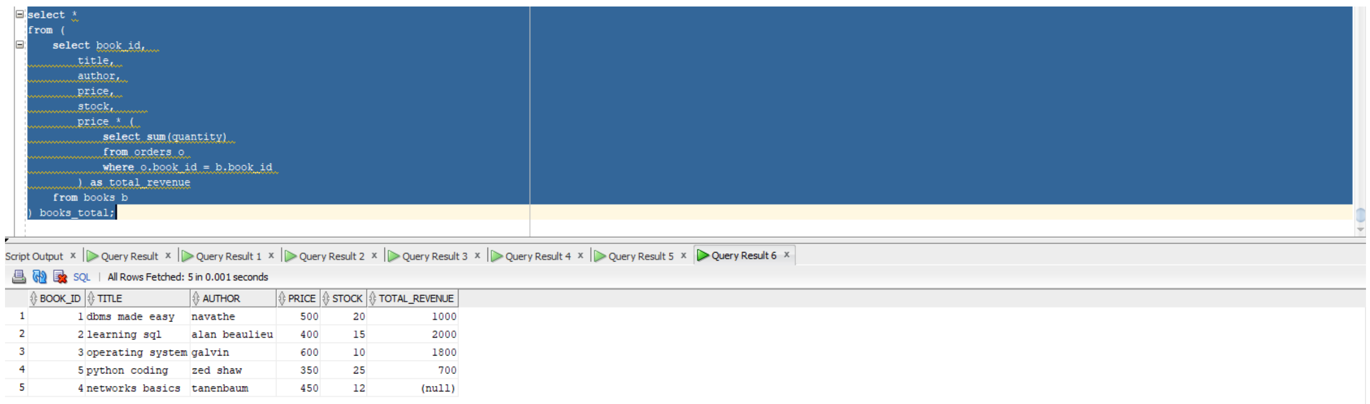


```sql
select book_id, title
from books b
where stock < (
    select sum(quantity)
    from orders o
    where o.book_id = b.book_id
);

select * from books;
```

All Rows Fetched: 0 in 0.005 seconds

```
  BOOK_ID  TITLE
```

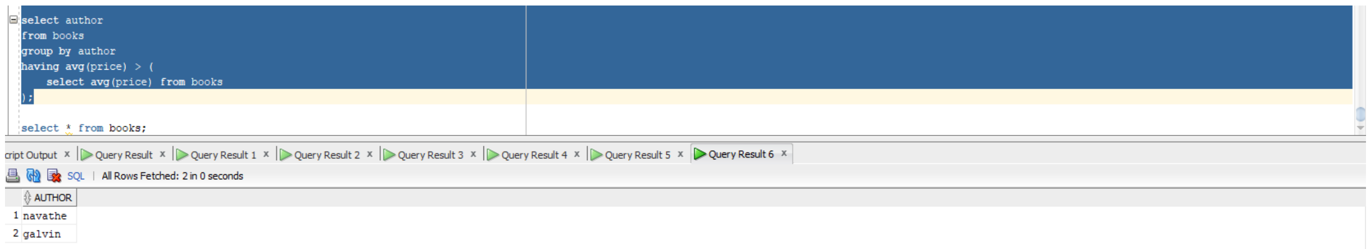
Part D: Placement Subqueries



```sql
select *
from (
    select book_id,
        title,
        author,
        price,
        stock,
        price * (
            select sum(quantity)
            from orders o
            where o.book_id = b.book_id
        ) as total_revenue
    from books b
) books_total;
```

All Rows Fetched: 5 in 0.001 seconds

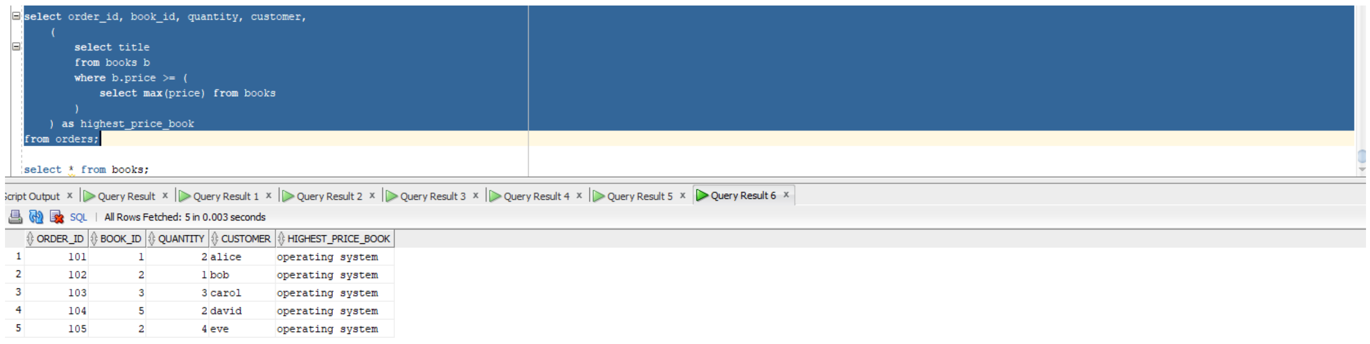
```
  BOOK_ID TITLE            AUTHOR        PRICE STOCK TOTAL_REVENUE
1       1 dbms made easy   navathe         500    20          1000
2       2 learning sql     alan beaulieu   400    15          2000
3       3 operating system galvin          600    10          1800
4       5 python coding    zed shaw        350    25           700
5       4 networks basics  tanenbaum       450    12        (null)
```



```sql
select author
from books
group by author
having avg(price) > (
    select avg(price) from books
);

select * from books;
```

All Rows Fetched: 2 in 0 seconds

```
  AUTHOR
1 navathe
2 galvin
```

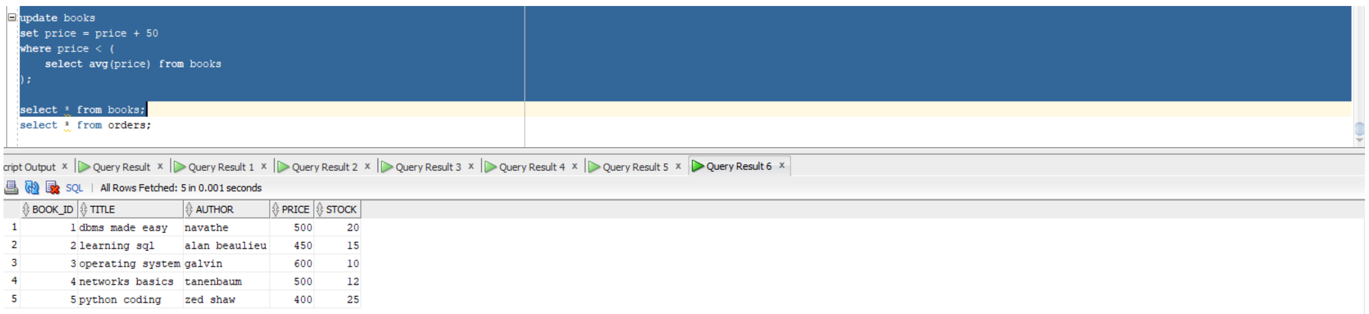


```sql
select order_id, book_id, quantity, customer,
    (
        select title
        from books b
        where b.price >= (
            select max(price) from books
        )
    ) as highest_price_book
from orders;

select * from books;
```

All Rows Fetched: 5 in 0.003 seconds

```
  ORDER_ID BOOK_ID QUANTITY CUSTOMER HIGHEST_PRICE_BOOK
1      101       1        2 alice    operating system
2      102       2        1 bob      operating system
3      103       3        3 carol    operating system
4      104       5        2 david    operating system
5      105       2        4 eve      operating system
```

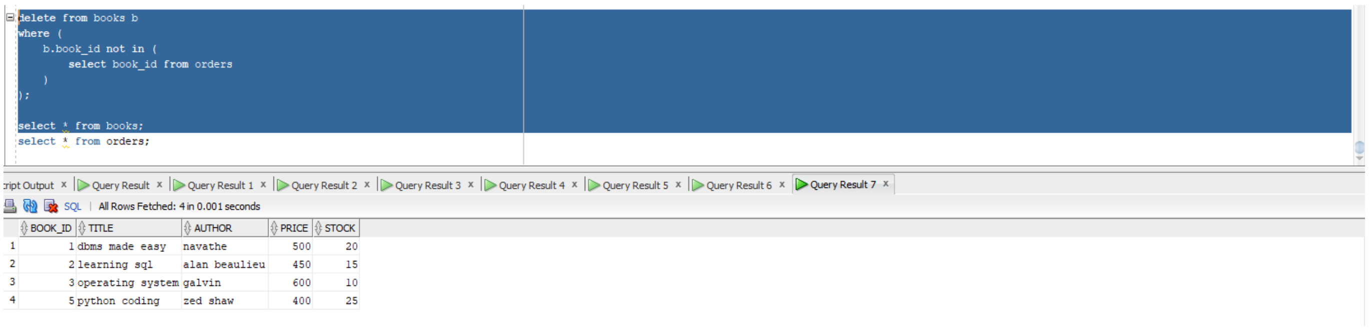
Part E: Placement Subqueries



```
update books
set price = price + 50
where price < (
    select avg(price) from books
);

select * from books;
select * from orders;
```

All Rows Fetched: 5 in 0.001 seconds

	BOOK_ID	TITLE	AUTHOR	PRICE	STOCK
1	1	dbms made easy	navathe	500	20
2	2	learning sql	alan beaulieu	450	15
3	3	operating system	galvin	600	10
4	4	networks basics	tanenbaum	500	12
5	5	python coding	zed shaw	400	25



```
delete from books b
where (
    b.book_id not in (
        select book_id from orders
    )
);

select * from books;
select * from orders;
```

All Rows Fetched: 4 in 0.001 seconds

	BOOK_ID	TITLE	AUTHOR	PRICE	STOCK
1	1	dbms made easy	navathe	500	20
2	2	learning sql	alan beaulieu	450	15
3	3	operating system	galvin	600	10
4	5	python coding	zed shaw	400	25