```sql
create database mydb;

/*
Table: salesman

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

create table salesman
(
     salesman_id int primary key,
     name varchar(255),
     city varchar(255),
     commission float(53)
);

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

/*
Table: customer

customer_id  customer_name  city       grade       salesman_id
-----------  ------------  ----------  ----------  -----------
3002         Nick Rimando  New York    100         5001
3005         Graham Zusi   California  200         5002
3001         Brad Guzan    London                  5005
3004         Fabian Johns  Paris       300         5006
3007         Brad Davis    New York    200         5001
3009         Geoff Camero  Berlin      100         5003
3008         Julian Green  London      300         5002
3003         Jozy Altidor  Moscow      200         5007
*/
```

```sql
create table customer
(
      customer_id int primary key,
      customer_name varchar(255),
      city varchar(255),
      grade int,
      salesman_id int
);

insert into customer values (3002,'Nick Rimando','New York',100,5001),
                            (3005,'Graham Zusi','California',200,5002),
                            (3001,'Brad Guzan','London',null,5005),
                            (3004,'Fabian Johns','Paris',300, 5006),
                            (3007,'Brad Davis','New York',200,5001),
                            (3009,'Geoff Camero','Berlin',100,5003),
                            (3008,'Julian Green','London',300,5002),
                            (3003,'Jozy Altidor','Moscow',200,5007);

/*
Table: orders

order_no     purchase_amount   order_date  customer_id  salesman_id
----------  ----------------  ----------  -----------  -----------
70001         150.5            2012-10-05      3005         5002
70009         270.65           2012-09-10      3001         5005
70002         65.26            2012-10-05      3002         5001
70004         110.5            2012-08-17      3009         5003
70007         948.5            2012-09-10      3005         5002
70005         2400.6           2012-07-27      3007         5001
70008         5760             2012-09-10      3002         5001
70010         1983.43          2012-10-10      3004         5006
70003         2480.4           2012-10-10      3009         5003
70012         250.45           2012-06-27      3008         5002
70011         75.29            2012-08-17      3003         5007
70013         3045.6           2012-04-25      3002         5001
*/

create table orders
(
      order_no int primary key,
      purchase_amount float(53),
      order_date date,
      customer_id int,
      salesman_id int
);
```

```
insert into orders values (70001, 150.5,  '2012-10-05',  3005,  5002),
                          (70009, 270.65, '2012-09-10',  3001,  5005),
                          (70002, 65.26,  '2012-10-05',  3002,  5001),
                          (70004, 110.5,  '2012-08-17',  3009,  5003),
                          (70007, 948.5,  '2012-09-10',  3005,  5002),
                          (70005, 2400.6, '2012-07-27',  3007,  5001),
                          (70008, 5760,   '2012-09-10',  3002,  5001),
                          (70010, 1983.43,'2012-10-10',  3004,  5006),
                          (70003, 2480.4, '2012-10-10',  3009,  5003),
                          (70012, 250.45, '2012-06-27',  3008,  5002),
                          (70011, 75.29,  '2012-08-17',  3003,  5007),
                          (70013, 3045.6, '2012-04-25',  3002,  5001);

/*
Table: company

COMPANY_ID    COMPANY_NAME
----------    -------------
    11         Samsung
    12         iBall
    13         Epsion
    14         Zebronics
    15         Asus
    16         Frontech
*/

create table company
(
     COMPANY_ID int primary key,
     COMPANY_NAME varchar(255)
);

insert into company values (11, 'Samsung'),
                           (12, 'iBall'),
                           (13, 'Epsion'),
                           (14, 'Zebronics'),
                           (15, 'Asus'),
                           (16, 'Frontech');

/*
Table: product

PRODUCT_ID   PRODUCT_NAME              PRODUCT_PRICE    PRODUCT_COMPANY
----------  -------------------      --------------    ---------------
    101       Mother Board                3200              15
    102       Keyboard                    450               16
```

```
    103       ZIP drive                 250             14
    104       Speaker                   550             16
    105       Monitor                   5000            11
    106       DVD drive                 900             12
    107       CD drive                  800             12
    108       Printer                   2600            13
    109       Refill cartridge          350             13
    110       Mouse                     250             12
*/

create table product
(
     PRODUCT_ID int PRIMARY KEY,
     PRODUCT_NAME varchar(255),
     PRODUCT_PRICE int,
     PRODUCT_COMPANY int
);

insert into product values (101, 'Mother Board',     3200, 15),
                           (102, 'Keyboard',         450,  16),
                           (103, 'ZIP drive',        250,  14),
                           (104, 'Speaker',          550,  16),
                           (105, 'Monitor',          5000, 11),
                           (106, 'DVD drive',        900,  12),
                           (107, 'CD drive',         800,  12),
                           (108, 'Printer',          2600, 13),
                           (109, 'Refill cartridge', 350,  13),
                           (110, 'Mouse',            250,  12);

/*
Table: department

DEPT_CODE    DEPT_NAME        DEPT_BUDGET
--------   ---------------   -------------
    57        IT                 65000
    63        Finance            15000
    47        HR                 240000
    27        RD                 55000
    89        QC                 75000
*/

create table department
(
     DEPT_CODE int primary key,
     DEPT_NAME varchar(255),
     DEPT_BUDGET INT
```

```
);

insert into department values (57, 'IT',      65000),
                              (63, 'Finance', 15000),
                              (47, 'HR',      240000),
                              (27, 'RD',      55000),
                              (89, 'QC',      75000);

/*
Table: employee

EMPLOYEE_ID      EMPLOYEE_FNAME    EMPLOYEE_LNAME       EMPLOYEE_DEPT
-----------     ---------------   ----------------      -------------
   127323            Michale         Robbin                  57
   526689            Carlos          Snares                  63
   843795            Enric           Dosio                   57
   328717            John            Snares                  63
   444527            Joseph          Dosni                   47
   659831            Zanifer         Emily                   47
   847674            Kuleswar        Sitaraman               57
   748681            Henrey          Gabriel                 47
   555935            Alex            Manuel                  57
   539569            George          Mardy                   27
   733843            Mario           Saule                   63
   631548            Alan            Snappy                  27
   839139            Maria           Foster                  57
*/

create table employee
(
      EMPLOYEE_ID int primary key,
      EMPLOYEE_FNAME varchar(255),
      EMPLOYEE_LNAME varchar(255),
      EMPLOYEE_DEPT int
);

insert into employee values (127323, 'Michale',  'Robbin',     57),
                            (526689, 'Carlos',   'Snares',     63),
                            (843795, 'Enric',    'Dosio',      57),
                            (328717, 'John',     'Snares',     63),
                            (444527, 'Joseph',   'Dosni',      47),
                            (659831, 'Zanifer',  'Emily',      47),
                            (847674, 'Kuleswar', 'Sitaraman',  57),
                            (748681, 'Henrey',   'Gabriel',    47),
                            (555935, 'Alex',     'Manuel',     57),
                            (539569, 'George',   'Mardy',      27),
```

```
                        (733843, 'Mario',    'Saule',      63),
                        (631548, 'Alan',     'Snappy',     27),
                        (839139, 'Maria',    'Foster',     57);

/* Write a SQL statement to prepare a list with salesman name, customer
name and their cities for the salesmen and customer who belongs to the
same city. */

SELECT s.name, c.customer_name, c.city
  FROM salesman s INNER JOIN customer c ON s.city = c.city;

/* Write a SQL statement to make a list with order no, purchase amount,
customer name and their cities for those orders which order amount
between 500 and 2000 */

SELECT o.order_no, o.purchase_amount, c.customer_name, c.city
  FROM orders o INNER JOIN customer c ON o.customer_id = c.customer_id
     WHERE o.purchase_amount BETWEEN 500 AND 2000;

/* Write a SQL statement to know which salesman are working for which
customer. */

SELECT c.customer_name AS "Customer Name", s.name AS "Salesman"
  FROM customer c INNER JOIN salesman s
    ON c.salesman_id = s.salesman_id;

/* Write a SQL statement to find the list of customers who appointed a
salesman for their jobs who gets a commission from the company that is
more than 12%. */

SELECT c.customer_name AS "Customer Name", c.city,
       s.name AS "Salesman", s.commission
  FROM customer c INNER JOIN salesman s
    ON c.salesman_id = s.salesman_id WHERE s.commission > 0.12;

/* Write a SQL statement to find the list of customers who appointed a
salesman for their jobs who does not live in the same city where their
customer lives, and gets a commission is above 12%. */

SELECT c.customer_name AS "Customer Name", c.city AS "Customer City",
       s.name AS "Salesman", s.city AS "Salesman City", s.commission
  FROM customer c INNER JOIN salesman s
    ON c.salesman_id = s.salesman_id
     WHERE s.commission > 0.12 AND c.city != s.city;
```

```sql
/* Write a SQL statement to find the details of an order i.e. order number, order date, amount of order, which customer gives the order and which salesman works for that customer and how much commission he gets for an order. */

SELECT o.order_no, o.order_date, o.purchase_amount,
       c.customer_name AS "Customer Name",
       s.name AS "Salesman", s.commission
  FROM (orders o INNER JOIN customer c ON o.customer_id=c.customer_id)
       INNER JOIN salesman s ON o.salesman_id=s.salesman_id;

/* Write a SQL statement to the customers in ascending order who work either through a salesman or by own. */

SELECT c.customer_name AS "Customer Name"
  FROM customer c LEFT JOIN salesman s ON c.salesman_id=s.salesman_id
     ORDER BY c.customer_id ASC;

/* Write a SQL statement to make a list in ascending order for the customer who holds a grade less than 300 and works either through a salesman or by own. */

SELECT c.customer_name AS "Customer Name", c.grade
  FROM customer c LEFT JOIN salesman s ON c.salesman_id=s.salesman_id
  WHERE c.grade < 300 ORDER BY c.customer_id ASC;

/* Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to find that either any of the existing customers have placed no order or placed one or more orders. */

SELECT c.customer_name AS "Customer Name", c.city,
       o.order_no, o.order_date, o.purchase_amount
  FROM customer c LEFT OUTER JOIN orders o
     ON c.customer_id=o.customer_id ORDER BY o.order_date;

/* Write a SQL statement to make a report with customer name, city, order number, order date, order amount, salesman name and commission to find that either any of the existing customers have placed no order or placed one or more orders by their salesman or by own. */

SELECT c.customer_name AS "Customer Name", c.city,
       o.order_no, o.order_date, o.purchase_amount,
       s.name AS "Salesman", s.commission
  FROM customer c LEFT JOIN orders o ON c.customer_id=o.customer_id
          LEFT JOIN salesman s ON c.salesman_id=s.salesman_id;
```

```sql
/* Write a SQL statement to make a list in ascending order for the salesmen who works either for one or more customer or not yet join under any of the customers. */

SELECT s.name AS "Salesman"
  FROM salesman s LEFT JOIN customer c ON s.salesman_id=c.salesman_id
  ORDER BY c.salesman_id ASC;

/* Write a SQL statement to make a list for the salesmen who works either for one or more customer or not yet join under any of the customers who placed either one or more orders or no order to their supplier. */

SELECT s.name AS "Salesman"
  FROM salesman s LEFT JOIN customer c ON s.salesman_id=c.salesman_id
       LEFT JOIN orders o ON c.customer_id=o.customer_id;

/* Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customer. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated supplier. */

SELECT s.name AS "Salesman"
  FROM salesman s LEFT JOIN customer c ON s.salesman_id=c.salesman_id
        LEFT OUTER JOIN orders o ON c.customer_id=o.customer_id
  WHERE o.purchase_amount >= 2000 AND grade IS NOT NULL;

/* Write a SQL statement to display customer name, city, order no, order date, purchase amount for those customers from the existing list who placed one or more orders or which order(s) have been placed by the customer who is not on the list. */

SELECT c.customer_name AS "Customer Name", c.city,
       o.order_no, o.order_date, o.purchase_amount
  FROM customer c RIGHT JOIN orders o ON c.customer_id= o.customer_id;

/* Write a SQL statement to make a cartesian product between salesman and customer i.e. each salesman will appear for all customer and vice versa. */

SELECT s.name AS "Salesman", c.customer_name AS "Customer"
  FROM salesman s CROSS JOIN customer c;

/* Write a SQL statement to make a cartesian product between salesman and customer i.e. each salesman will appear for all customer and vice versa for that customer who belongs to a city. */
```

```sql
SELECT s.name AS "Salesman", c.customer_name AS "Customer"
  FROM salesman s CROSS JOIN customer c
  WHERE s.city IS NOT NULL;

/* Write a SQL statement to make a cartesian product between salesman
and customer i.e. each salesman will appear for all customer and vice
versa for those salesmen who belongs to a city and the customers who
must have a grade. */

SELECT s.name AS "Salesman", c.customer_name AS "Customer"
  FROM salesman s CROSS JOIN customer c
  WHERE s.city IS NOT NULL AND c.grade IS NOT NULL;

/* Write a SQL statement to make a cartesian product between salesman
and customer i.e. each salesman will appear for all customer and vice
versa for those salesmen who must belong a city which is not the same
as his customer and the customers should have an own grade. */

SELECT s.name AS "Salesman", c.customer_name AS "Customer"
  FROM salesman s CROSS JOIN customer c
  WHERE s.city IS NOT NULL AND s.city!=c.city AND c.grade IS NOT NULL;

/* Write a SQL query to display all the data from the product table,
including all the data for each product's producer company. */

SELECT * FROM product INNER JOIN company
     ON product.PRODUCT_COMPANY=company.COMPANY_ID;

/* Write a SQL query to display the name, price, and company name of all
the products. */

SELECT p.PRODUCT_NAME, p.PRODUCT_PRICE, c.COMPANY_NAME
  FROM product p INNER JOIN company c
    ON p.PRODUCT_COMPANY=c.COMPANY_ID;

/* Write a SQL query to display the average price of products of each
company */

SELECT c.COMPANY_NAME as 'Company',AVG(p.PRODUCT_PRICE) as 'Avg Price'
  FROM product p INNER JOIN company c
    ON p.PRODUCT_COMPANY=c.COMPANY_ID GROUP BY c.COMPANY_NAME;

/* Write a SQL query to display the names of the company whose products
have an average price larger than or equal to Rs. 350. */

SELECT c.COMPANY_NAME as 'Company',AVG(p.PRODUCT_PRICE) as 'Avg Price'
```

```
  FROM product p INNER JOIN company c
    ON p.PRODUCT_COMPANY=c.COMPANY_ID
      GROUP BY c.COMPANY_NAME HAVING AVG(p.PRODUCT_PRICE) > 350;

/* Write a SQL query to display the name of each company along with the
ID and price for their most expensive product. */

SELECT C.COMPANY_NAME, P.PRODUCT_ID, P.PRODUCT_NAME, P.PRODUCT_PRICE
  FROM product P INNER JOIN company C
    ON P.PRODUCT_COMPANY=C.COMPANY_ID
      AND P.PRODUCT_PRICE =
        (SELECT MAX(P.PRODUCT_PRICE) FROM product P
           WHERE P.PRODUCT_COMPANY = C.COMPANY_ID);

/* Write a query in SQL to display all the data of employees including
their department. */

SELECT * FROM employee e INNER JOIN department d
    ON e.EMPLOYEE_DEPT = d.DEPT_CODE;

/* Write a query in SQL to display the first name and last name of each
employee, along with the name and the budget for their department. */

SELECT e.EMPLOYEE_FNAME AS "First Name",
       e.EMPLOYEE_LNAME AS "Last Name",
       d.DEPT_NAME AS "Department Name",
       d.DEPT_BUDGET AS "Amount Allotted"
  FROM employee e INNER JOIN department d
    ON e.EMPLOYEE_DEPT = d.DEPT_CODE;

/* Write a query in SQL to find the first name and last name of employees
working for departments with a budget more than Rs. 50000. */

SELECT e.EMPLOYEE_FNAME AS "First Name",
       e.EMPLOYEE_LNAME AS "Last Name"
  FROM employee e INNER JOIN department d
    ON e.EMPLOYEE_DEPT = d.DEPT_CODE WHERE d.DEPT_BUDGET > 50000;

SELECT EMPLOYEE_FNAME as 'First Name',
       EMPLOYEE_LNAME as 'Last Name'
  FROM employee WHERE EMPLOYEE_DEPT IN
  (SELECT DEPT_CODE FROM department WHERE DEPT_BUDGET > 50000);

/* Write a query in SQL to find the names of departments where more than
two employees are working. */
```

```sql
SELECT d.DEPT_NAME as 'Department Name',
       COUNT(e.EMPLOYEE_ID) as 'No of Employees'
  FROM employee e INNER JOIN department d
    ON e.EMPLOYEE_DEPT = d.DEPT_CODE
  GROUP BY d.DEPT_NAME HAVING COUNT(e.EMPLOYEE_ID) > 2;

SELECT DEPT_NAME FROM department WHERE DEPT_CODE IN
     (SELECT EMPLOYEE_DEPT FROM employee
          GROUP BY EMPLOYEE_DEPT HAVING COUNT(*) > 2
     );

/* Write a query to display all the orders from the orders table issued
by the salesman 'Paul Adam'. */

SELECT * FROM orders WHERE salesman_id =
     (SELECT salesman_id FROM salesman WHERE name = 'Paul Adam');

/* Write a query to display all the orders for the salesman who belongs
to the city London. */

SELECT * FROM orders WHERE salesman_id =
 (SELECT salesman_id FROM salesman WHERE city = 'London');

/* Write a query to find all the orders issued against the salesman who
may works for customer whose id is 3007. */

SELECT * FROM orders WHERE salesman_id =
     (SELECT salesman_id FROM orders WHERE customer_id = 3007);

/* Write a query to display all the orders which values are greater than
the average order value for 10th October 2012. */

SELECT * FROM orders WHERE purchase_amount >
     (SELECT AVG(purchase_amount) FROM orders
          WHERE order_date = '2012-10-10');

--Write a query to find all orders attributed to a salesman in New York.

SELECT * FROM orders WHERE salesman_id IN
     (SELECT salesman_id FROM salesman WHERE city = 'New York');

/* Write a query to display the commission of all the salesmen servicing
customers in Paris. */

SELECT commission FROM salesman WHERE salesman_id IN
     (SELECT salesman_id FROM customer WHERE city = 'Paris');
```

```sql
/* Write a query to display all the customers whose id is 2001 below the
salesman ID of Mc Lyon. */

SELECT * FROM customer WHERE customer_id =
     (SELECT salesman_id - 2001 FROM salesman WHERE name = 'Mc Lyon');

/* Write a query to count the customers with grades above New York's
average. */

SELECT COUNT(*) FROM customer WHERE grade >
     (SELECT AVG(grade) FROM customer WHERE city = 'New York');

-- Write a query to display all customers with orders on 5 October 2012.

SELECT * FROM customer WHERE customer_id IN
     (SELECT customer_id FROM orders WHERE order_date = '2012-10-05');

SELECT c.customer_id, c.customer_name, c.city, c.grade, c.salesman_id
     FROM customer c inner join orders o on
          c.customer_id=o.customer_id
          where o.order_date = '2012-10-05';

/* Write a query to display all the customers with orders issued on date
17th August, 2012. */

SELECT * FROM customer WHERE customer_id IN
     (SELECT customer_id FROM orders WHERE order_date = '2012-08-17');

SELECT c.customer_id, c.customer_name, c.city, c.grade, c.salesman_id
  FROM customer c inner join orders o
  on c.customer_id = o.customer_id where o.order_date = '2012-08-17';

/* Write a query to find the name and numbers of all salesmen who had
more than one customer. */

SELECT salesman_id, name FROM salesman
  WHERE salesman_id IN
     (SELECT salesman_id FROM customer
          GROUP BY salesman_id HAVING COUNT(*) > 1);

/* Write a query to extract the data from the customer table if and only
if one or more of the customers in the customer table are located in
London. */

SELECT * FROM customer WHERE EXISTS
     (SELECT * FROM customer WHERE city = 'London');
```

```sql
SELECT * FROM customer WHERE
     (SELECT count(*) FROM customer WHERE city = 'London') >= 1;
/* Write a query to find the salesmen who have multiple customers. */

SELECT * FROM salesman WHERE salesman_id IN
     (SELECT salesman_id FROM customer
          GROUP BY salesman_id HAVING COUNT(*) > 1);
/* Write a query to find all the salesmen who worked for only one
customer. */

SELECT * FROM salesman WHERE salesman_id IN
     (SELECT salesman_id FROM customer
          GROUP BY salesman_id HAVING COUNT(customer_id) = 1);

/* Write a query that extract the rows of all salesmen who have customers
with more than one orders. */

SELECT * FROM salesman WHERE salesman_id IN
     (SELECT salesman_id FROM customer WHERE customer_id IN
          (SELECT customer_id FROM orders
               GROUP BY customer_id HAVING COUNT(*) > 1
          )
     );

/* Write a query to find all information of a salesman who lives in the
city where any of the customers lives. */

SELECT * FROM salesman WHERE city IN (SELECT city FROM customer);

SELECT * FROM salesman WHERE city = ANY (SELECT city FROM customer);

/* Write a query to find all the salesmen for whom there are customers
that follow them. */

SELECT * FROM salesman WHERE city IN (SELECT city FROM customer);

/* Write a query to display the customers who have a greater grade than
any customer who belongs to a city that is alphabetically lower than the
city New York. */

SELECT * FROM customer WHERE grade >
     ANY (SELECT grade FROM customer WHERE city < 'New York');

/* Write a query to display all the orders that had amounts that were
greater than at least one of the orders on September 10th 2012. */
```

```
SELECT * FROM orders WHERE purchase_amount > ANY
 (SELECT purchase_amount FROM orders WHERE order_date = '2012-09-10');

/* Write a query to find all orders with an amount smaller than any
amount for a customer in London. */

SELECT * FROM orders WHERE purchase_amount <
     ANY (SELECT purchase_amount FROM orders WHERE customer_id IN
            (SELECT customer_id FROM customer WHERE city = 'London'
            )
         );

/* Write a query to display all orders with an amount smaller than the
maximum amount for customers in London. */

SELECT * FROM orders WHERE purchase_amount <
     (SELECT MAX(purchase_amount) FROM orders WHERE customer_id IN
           (SELECT customer_id FROM customer WHERE city = 'London'
           )
     );

/* Write a query to display only those customers whose grade are, in
fact, higher than every customer in New York. */

SELECT * FROM customer WHERE grade >
     ALL (SELECT grade FROM customer WHERE city = 'New York');

/* Write a query to find only those customers whose grade are, higher
than every customer to the city New York. */

SELECT * FROM customer WHERE grade >
     ALL (SELECT grade FROM customer WHERE city = 'New York');

/* Write a query to get all the information for those customers whose
grade is not as the grade of customer who belongs to the city London */

SELECT * FROM customer WHERE grade !=
     ANY (SELECT grade FROM customer WHERE city = 'London');

/* Write a query to find all those customers whose grade are not as the
grade, belongs to the city Paris. */

SELECT * FROM customer WHERE grade !=
     ANY (SELECT grade FROM customer WHERE city = 'Paris');

SELECT * FROM customer WHERE grade NOT IN
```

```sql
     (SELECT grade FROM customer WHERE city = 'Paris');

/* Write a query to find all those customers who hold a different grade
than any customer of the city Dallas. */

SELECT * FROM customer WHERE grade NOT IN
     (SELECT grade FROM customer WHERE city = 'Dallas');

/* Write a query in SQL to find all the details of employees whose last
name is Gabriel or Dosio. */

SELECT * FROM employee WHERE EMPLOYEE_LNAME IN ('Gabriel', 'Dosio');

/* Write a query in SQL to display all the details of employees who works
in department 89 or 63. */

SELECT * FROM employee WHERE EMPLOYEE_DEPT IN (89, 63);

/* Write a query in SQL to find the departments whose budget is larger
than the average budget of all the departments. */

SELECT * FROM department WHERE DEPT_BUDGET >
     (SELECT AVG(DEPT_BUDGET) FROM department);

/* Write a query in SQL to find the department whose budget amount is
second lowest. */

SELECT DEPT_CODE FROM department WHERE DEPT_BUDGET =
          (SELECT MIN(DEPT_BUDGET) FROM department WHERE DEPT_BUDGET >
               (SELECT MIN(DEPT_BUDGET) FROM department)
          );

/* Write a query in SQL to find the first name and last name of employees
working for department whose budget amount is second lowest. */

SELECT EMPLOYEE_FNAME, EMPLOYEE_LNAME FROM employee
     WHERE EMPLOYEE_DEPT IN
     (SELECT DEPT_CODE FROM department WHERE DEPT_BUDGET =
          (SELECT MIN(DEPT_BUDGET) FROM department WHERE DEPT_BUDGET >
               (SELECT MIN(DEPT_BUDGET) FROM department)
          )
     );
```