Task 2: SQL Basics (Tuomas Pasanen)

This exercise just simply has pictures of the commands, under the specified task. Database is the same one detailed in task 1.

1. Write a SQL statement to display columns name and commission for all the salesmen.

```
mybusiness=> SELECT name, commission FROM salesman;
    name     | commission
-------------+------------
 James Hoow  |       0.15
 Nail Knite  |       0.13
 Pit Alex    |       0.11
 Mc Lyon     |       0.14
 Paul Adam   |       0.13
 Lauson Hen  |       0.12
 Ben Johnson |       0.13
 Sam Lawson  |       0.11
(8 rows)
```

2. Find the salespeople who lives in the City of 'Paris'. Return salesperson's name, city.

```
mybusiness=> SELECT name, city FROM salesman WHERE city='Paris';
    name     | city
-------------+-------
 Nail Knite | Paris
 Mc Lyon    | Paris
(2 rows)
```

3. Find the details of all employees whose name includes ‘James’ or ‘Adam’. Return salesman_id, name.

```
mybusiness=> SELECT salesman_id, name FROM salesman WHERE name LIKE '%James%' OR name LIKE '%Adam%';
 salesman_id |    name
-------------+------------
        5001 | James Hoow
        5007 | Paul Adam
(2 rows)
```

4. Find the details of those salespeople whose name starts with any letter within 'A' and 'L' (not inclusive). Return salesman_id, name, city, commission.

```
mybusiness=> SELECT * FROM salesman WHERE name >='A' AND name <'M';
 salesman_id |    name     |   city    | commission
-------------+-------------+-----------+------------
        5001 | James Hoow  | New York  |       0.15
        5003 | Lauson Hen  | San Jose  |       0.12
        5010 | Ben Johnson | San Jose  |       0.13
(3 rows)
```

5. Find those salesmen whose commission is greater than or equal to 0.13. Return name, commission.

```
mybusiness=> SELECT name, commission FROM salesman WHERE commission >= 0.13;
    name     | commission
-------------+------------
 James Hoow  |       0.15
 Nail Knite  |       0.13
 Mc Lyon     |       0.14
 Paul Adam   |       0.13
 Ben Johnson |       0.13
(5 rows)
```

6. Find the orders, which are delivered by a salesperson of ID. 5001. Return ord_no,

ord_date, purch_amt

```
mybusiness=> SELECT ord_no, ord_date, purch_amt FROM orders WHERE salesman_id = 5001;
 ord_no |  ord_date  | purch_amt
--------+------------+-----------
  70002 | 2012-10-05 |     65.26
  70005 | 2012-07-27 |    2400.6
  70008 | 2012-09-10 |      5760
  70013 | 2012-04-25 |    3045.6
(4 rows)
```

7. Find the orders, which are delivered by a salesperson of ID. 5001 and purchace amount is more than 1000. Return ord_no, ord_date, purch_amt

```
mybusiness=> SELECT ord_no, ord_date, purch_amt FROM orders WHERE salesman_id = 5001 AND purch_amt > 1000.0;
 ord_no |  ord_date  | purch_amt
--------+------------+-----------
  70005 | 2012-07-27 |    2400.6
  70008 | 2012-09-10 |      5760
  70013 | 2012-04-25 |    3045.6
(3 rows)
```

8. Find the products whose price is in the range 1000 to 4000. Begin and end values are included. Return ord_no, ord_date, purch_amt.

```
mybusiness=> SELECT ord_no, ord_date, purch_amt FROM orders WHERE purch_amt >= 1000 AND purch_amt <= 4000;
 ord_no |  ord_date  | purch_amt
--------+------------+-----------
  70005 | 2012-07-27 |    2400.6
  70010 | 2012-10-10 |   1983.43
  70003 | 2012-10-10 |    2480.4
  70013 | 2012-04-25 |    3045.6
  70014 | 2012-06-25 |    1786.4
(5 rows)
```

9. Write a SQL query to find all the orders which purchase amount is less than 500€ and done before October 2012 or which purchase amount is greater than 2000 and done in October 2012. Return ord_no, purch_amt, ord_date.

```
mybusiness=> SELECT ord_no, purch_amt, ord_date FROM orders WHERE
(purch_amt < 500 and EXTRACT(MONTH FROM ord_date) < 10) OR
(purch_amt > 2000 AND EXTRACT(MONTH FROM ord_date)=10);
 ord_no | purch_amt |  ord_date
--------+-----------+------------
  70009 |    270.65 | 2012-09-10
  70004 |     110.5 | 2012-08-17
  70003 |    2480.4 | 2012-10-10
  70012 |    250.45 | 2012-06-27
  70011 |     75.29 | 2012-08-17
(5 rows)
```

10. Update salesman whose id is 5007 name to be Paul White and his city to London.

```
mybusiness=> UPDATE salesman
mybusiness-> SET name = 'Paul White', city = 'London'
mybusiness-> WHERE salesman_id = 5007;
UPDATE 1
mybusiness=> SELECT * FROM salesman;
 salesman_id |    name     |   city   | commission
-------------+-------------+----------+------------
        5001 | James Hoow  | 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  | San Jose |       0.12
        5010 | Ben Johnson | San Jose |       0.13
        5011 | Sam Lawson  | Santiago |       0.11
        5007 | Paul White  | London   |       0.13
(8 rows)
```

11. Update customer's whose id is 3005 grade to be 300.

```
mybusiness=> UPDATE customer
mybusiness-> SET grade = 300
mybusiness-> WHERE customer_id = 3005;
UPDATE 1
mybusiness=> SELECT * FROM customer
mybusiness-> ;
 customer_id |   cust_name    |    city    | grade | salesman_id
-------------+----------------+------------+-------+-------------
        3002 | Nick Rimando   | New York   |   100 |        5001
        3007 | Brad Davis     | New York   |   200 |        5001
        3008 | Julian Green   | London     |   300 |        5002
        3004 | Fabian Johnson | Paris      |   300 |        5006
        3009 | Geoff Cameron  | Berlin     |   100 |        5003
        3003 | Jozy Altidor   | Moscow     |   200 |        5007
        3001 | Brad Guzan     | London     |   300 |        5005
        3010 | Marion Cameron | San Jose   |   300 |        5010
        3005 | Graham Zusi    | California |   300 |        5002
(9 rows)
```

12. Change salesman whose id is 5007 id to be 5009.

First drop the constraints:

```
mybusiness=> ALTER TABLE customer DROP CONSTRAINT customer_salesman_id_fkey;
ALTER TABLE
```

```
mybusiness=> ALTER TABLE orders DROP CONSTRAINT orders_salesman_id_fkey;
ALTER TABLE
```

Now alter the salesman_id:

```
mybusiness=> UPDATE salesman
mybusiness-> SET salesman_id = 5009
mybusiness-> WHERE salesman_id = 5007;
UPDATE 1
mybusiness=> SELECT * FROM salesman;
 salesman_id |    name     |   city   | commission
-------------+-------------+----------+------------
        5001 | James Hoow  | 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  | San Jose |       0.12
        5010 | Ben Johnson | San Jose |       0.13
        5011 | Sam Lawson  | Santiago |       0.11
        5009 | Paul White  | London   |       0.13
(8 rows)
```

```
mybusiness=> UPDATE customer
mybusiness-> SET salesman_id = 5009
mybusiness-> WHERE salesman_id = 5007;
UPDATE 1
```

```
mybusiness=> UPDATE orders
mybusiness-> SET salesman_id = 5009
mybusiness-> WHERE salesman_id = 5007;
UPDATE 1
```

Now add the constraints back:

```
mybusiness=> ALTER TABLE customer
mybusiness-> ADD FOREIGN KEY (salesman_id)
mybusiness-> REFERENCES salesman(salesman_id);
ALTER TABLE
```

```
mybusiness=> ALTER TABLE orders
mybusiness-> ADD FOREIGN KEY (salesman_id)
mybusiness-> REFERENCES salesman(salesman_id);
ALTER TABLE
```

The constraints in action:

```
mybusiness=> \d customer
                                Table "public.customer"
   Column    |  Type   | Collation | Nullable |                  Default
-------------+---------+-----------+----------+-----------------------------------------------
 customer_id | integer |           | not null | nextval('customer_customer_id_seq'::regclass)
 cust_name   | text    |           |          |
 city        | text    |           |          |
 grade       | integer |           |          |
 salesman_id | integer |           |          |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (customer_id)
Foreign-key constraints:
    "customer_salesman_id_fkey" FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
Referenced by:
    TABLE "orders" CONSTRAINT "orders_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
```

13. Delete order number 70014 from orders.

```
mybusiness=> DELETE FROM orders
mybusiness-> WHERE ord_no = 70014;
DELETE 1
mybusiness=> SELECT * FROM orders;
 ord_no | purch_amt |  ord_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
  70013 |    3045.6 | 2012-04-25 |        3002 |        5001
  70011 |     75.29 | 2012-08-17 |        3003 |        5009
(12 rows)
```

14. Delete salesman whose id is 5010 and all the customers who has the reference to this salesman.

```
mybusiness=> DELETE FROM customer
mybusiness-> WHERE salesman_id = 5010;
DELETE 1
```

```
mybusiness=> DELETE FROM salesman
mybusiness-> WHERE salesman_id = 5010;
DELETE 1
```

15. Write a SQL query to calculate average purchase amount of all orders. Return average purchase amount

```
mybusiness=> SELECT AVG(purch_amt) FROM orders;
          avg
----------------------
 1461.7650000000000000
(1 row)
```

Rounded:

```
mybusiness=> SELECT ROUND(AVG(purch_amt)) FROM orders;
 round
-------
  1462
(1 row)
```

16. Write a SQL query to calculate the average price for purchase amount of salesman 5001.

```
mybusiness=> SELECT ROUND(AVG(purch_amt), 2) FROM orders
mybusiness-> WHERE salesman_id = 5001;
  round
---------
 2817.87
(1 row)
```

17. Write a SQL query to calculate total purchase amount of all orders. Return total purchase amount.

```
mybusiness=> SELECT SUM(purch_amt) FROM orders;
   sum
----------
 17541.18
(1 row)
```

18. Write a SQL query to count the number of orders.

```
mybusiness=> SELECT COUNT(*) FROM orders;
 count
-------
    12
(1 row)
```

19. Write a SQL query to count the number of unique salespeople. Return number of salespeople.

```
mybusiness=> SELECT COUNT(DISTINCT salesman_id)
mybusiness-> FROM salesman;
 count
-------
     7
(1 row)
```

Since salesman_id is a primary key, we don't necessarily have to do DISTINCT(). There won't be duplicates of id's in any case.

20. Write a SQL query to count the number of orders after 2012-07-01.

```
mybusiness=> SELECT COUNT(*) FROM orders
mybusiness-> WHERE ord_date > '2012-07-01';
 count
-------
    10
(1 row)
```

21. Write a SQL query to count the number of orders in October 2012.

```
mybusiness=> SELECT COUNT(*) FROM orders
mybusiness-> WHERE EXTRACT(MONTH FROM ord_date) = 10;
 count
-------
     4
```

22. Write a SQL query to find the maximum purchase amount.

```
mybusiness=> SELECT MAX(purch_amt) FROM orders;
 max
------
 5760
(1 row)
```

23. Write a SQL query to find the lowest purchase amount ordered by each customer. Return customer ID, minimum purchase amount.

```
mybusiness=> SELECT customer_id, MIN(purch_amt) FROM orders
mybusiness-> GROUP BY customer_id
mybusiness-> ORDER BY customer_id;
 customer_id |   min
-------------+---------
        3001 |  270.65
        3002 |   65.26
        3003 |   75.29
        3004 | 1983.43
        3005 |   150.5
        3007 |  2400.6
        3008 |  250.45
        3009 |   110.5
(8 rows)
```

Ordered by the customer_id.

24. Write a SQL query to find the highest purchase amount ordered by each customer on a particular date. Return, order date and highest purchase amount.

```
mybusiness=> SELECT ord_date, MAX(purch_amt) FROM orders
mybusiness-> GROUP BY ord_date
mybusiness-> ORDER BY ord_date;
  ord_date   |  max
-------------+--------
 2012-04-25 | 3045.6
 2012-06-27 | 250.45
 2012-07-27 | 2400.6
 2012-08-17 |  110.5
 2012-09-10 |   5760
 2012-10-05 |  150.5
 2012-10-10 | 2480.4
(7 rows)
```

25. Write a SQL query to find highest order (purchase) amount by each customer in a particular order date. Filter the result by highest order (purchase) amount above 2000.00. Return customer id, order date and maximum purchase amount.

```
mybusiness=> SELECT customer_id, ord_date, MAX(purch_amt)
mybusiness-> FROM orders
mybusiness-> GROUP BY ord_date, customer_id, purch_amt HAVING purch_amt > 2000.00
mybusiness-> ORDER BY ord_date;
 customer_id |  ord_date   |  max
-------------+-------------+--------
        3002 | 2012-04-25 | 3045.6
        3007 | 2012-07-27 | 2400.6
        3002 | 2012-09-10 |   5760
        3009 | 2012-10-10 | 2480.4
(4 rows)
```

26. Write a SQL query to find the maximum order (purchase) amount in the range 2000, 4000 (Begin and end values are included.) by combination of each customer and order date.

```
mybusiness=> SELECT customer_id, ord_date, MAX(purch_amt)
mybusiness-> FROM orders
mybusiness-> GROUP BY ord_date, customer_id, purch_amt
mybusiness-> HAVING purch_amt BETWEEN 2000 AND 4000
mybusiness-> ORDER BY ord_date;
 customer_id |  ord_date  |  max
-------------+------------+--------
        3002 | 2012-04-25 | 3045.6
        3007 | 2012-07-27 | 2400.6
        3009 | 2012-10-10 | 2480.4
(3 rows)
```

27. Write a SQL query to find the maximum order (purchase) amount generated by each salesperson. Filter the rows for the salesperson ID is in the range 5003 and 5008 (Begin and end values are included.). Return salesperson id and maximum purchase amount.

```
mybusiness=> SELECT salesman_id, MAX(purch_amt)
mybusiness-> FROM orders
mybusiness-> WHERE salesman_id BETWEEN 5003 AND 5008
mybusiness-> GROUP BY salesman_id;
 salesman_id |   max
-------------+---------
        5003 |  2480.4
        5005 |  270.65
        5006 | 1983.43
(3 rows)
```

28. Write a SQL query to count all the orders generated on '2012-08-17'. Return number of orders.

```
mybusiness=> SELECT COUNT(*) FROM orders
mybusiness-> WHERE ord_date = '2012-08-17';
 count
-------
     2
```

29. Write a SQL query to calculate average purchase amount of each salesman. Return salesman id and average purchase amount.

```
mybusiness=> SELECT salesman_id, ROUND(AVG(purch_amt), 2) FROM orders
mybusiness-> GROUP BY salesman_id;
 salesman_id |  round
-------------+---------
        5001 | 2817.87
        5005 |  270.65
        5006 | 1983.43
        5009 |   75.29
        5003 | 1295.45
        5002 |  449.82
(6 rows)
```

30. Sort the previous result in decreasing order by the average purchase amount.

```
mybusiness=> SELECT salesman_id, ROUND(AVG(purch_amt), 2) FROM orders
mybusiness-> GROUP BY salesman_id
mybusiness-> ORDER BY round DESC;
 salesman_id |  round
-------------+---------
        5001 | 2817.87
        5006 | 1983.43
        5003 | 1295.45
        5002 |  449.82
        5005 |  270.65
        5009 |   75.29
(6 rows)
```

“round” is the name of the result of the average, so we are ordering by it, instead of purch_amt.

31. Write a SQL query to find all the orders. Instead of showing salesman id you should show salesman’s name. Return ord_no, purch_amt, ord_date, customer_id and salesman name. You need to join orders and salesman tables.

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id, salesman.name AS salesman_name
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id;
 ord_no | purch_amt |  ord_date  | customer_id | salesman_name
--------+-----------+------------+-------------+---------------
  70001 |     150.5 | 2012-10-05 |        3005 | Nail Knite
  70009 |    270.65 | 2012-09-10 |        3001 | Pit Alex
  70002 |     65.26 | 2012-10-05 |        3002 | James Hoow
  70004 |     110.5 | 2012-08-17 |        3009 | Lauson Hen
  70007 |     948.5 | 2012-09-10 |        3005 | Nail Knite
  70005 |    2400.6 | 2012-07-27 |        3007 | James Hoow
  70008 |      5760 | 2012-09-10 |        3002 | James Hoow
  70010 |   1983.43 | 2012-10-10 |        3004 | Mc Lyon
  70003 |    2480.4 | 2012-10-10 |        3009 | Lauson Hen
  70012 |    250.45 | 2012-06-27 |        3008 | Nail Knite
  70013 |    3045.6 | 2012-04-25 |        3002 | James Hoow
  70011 |     75.29 | 2012-08-17 |        3003 | Paul White
(12 rows)
```

I used the AS keyword to import the salesman name as “salesman_name”, instead of just “name”.

32. Write a SQL query to find all the orders. Instead of showing salesman id you should show salesman’s name and instead of showing customer id you should show customer name. Return ord_no, purch_amt, ord_date, customer name and salesman name. You need to join orders, salesman and customers tables.

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id,
mybusiness-> salesman.name AS salesman_name,
mybusiness-> customer.cust_name AS customer_name
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id
mybusiness-> INNER JOIN customer
mybusiness-> ON orders.customer_id = customer.customer_id;
 ord_no | purch_amt |  ord_date  | customer_id | salesman_name | customer_name
--------+-----------+------------+-------------+---------------+---------------
  70001 |     150.5 | 2012-10-05 |        3005 | Nail Knite    | Graham Zusi
  70009 |    270.65 | 2012-09-10 |        3001 | Pit Alex      | Brad Guzan
  70002 |     65.26 | 2012-10-05 |        3002 | James Hoow    | Nick Rimando
  70004 |     110.5 | 2012-08-17 |        3009 | Lauson Hen    | Geoff Cameron
  70007 |     948.5 | 2012-09-10 |        3005 | Nail Knite    | Graham Zusi
  70005 |    2400.6 | 2012-07-27 |        3007 | James Hoow    | Brad Davis
  70008 |      5760 | 2012-09-10 |        3002 | James Hoow    | Nick Rimando
  70010 |   1983.43 | 2012-10-10 |        3004 | Mc Lyon       | Fabian Johnson
  70003 |    2480.4 | 2012-10-10 |        3009 | Lauson Hen    | Geoff Cameron
  70012 |    250.45 | 2012-06-27 |        3008 | Nail Knite    | Julian Green
  70013 |    3045.6 | 2012-04-25 |        3002 | James Hoow    | Nick Rimando
  70011 |     75.29 | 2012-08-17 |        3003 | Paul White    | Jozy Altidor
(12 rows)
```

33. Find the salesperson and customer who belongs to same city. Return Salesman, cust_name and city.

```
mybusiness=> SELECT salesman.name AS salesman_name,
mybusiness-> customer.cust_name AS customer_name,
mybusiness-> salesman.city
mybusiness-> FROM salesman
mybusiness-> INNER JOIN customer
mybusiness-> ON salesman.city = customer.city;
 salesman_name | customer_name  |   city
---------------+----------------+----------
 James Hoow    | Nick Rimando   | New York
 James Hoow    | Brad Davis     | New York
 Paul White    | Julian Green   | London
 Pit Alex      | Julian Green   | London
 Mc Lyon       | Fabian Johnson | Paris
 Nail Knite    | Fabian Johnson | Paris
 Paul White    | Brad Guzan     | London
 Pit Alex      | Brad Guzan     | London
(8 rows)
```

34. Calculate the average purchase amount of each salesman. Return salesman id, salesman name and average purchase.

My solution rounded to 2 decimal places.

```
mybusiness=> SELECT orders.salesman_id,
mybusiness-> salesman.name AS salesman_name,
mybusiness-> ROUND(AVG(orders.purch_amt), 2) AS average_sales
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id
mybusiness-> GROUP BY orders.salesman_id, salesman.name;
 salesman_id | salesman_name | average_sales
-------------+---------------+---------------
        5006 | Mc Lyon       |       1983.43
        5005 | Pit Alex      |        270.65
        5002 | Nail Knite    |        449.82
        5001 | James Hoow    |       2817.87
        5009 | Paul White    |         75.29
        5003 | Lauson Hen    |       1295.45
(6 rows)
```

35. Change the previous so that it also shows those salesman who have not sold anything.

```
mybusiness=> SELECT salesman.salesman_id,
mybusiness-> salesman.name AS salesman_name,
mybusiness-> ROUND(AVG(orders.purch_amt)) AS average_sales
mybusiness-> FROM salesman
mybusiness-> LEFT JOIN orders
mybusiness-> ON salesman.salesman_id = orders.salesman_id
mybusiness-> GROUP BY salesman.salesman_id, salesman.name;
 salesman_id | salesman_name | average_sales
-------------+---------------+---------------
        5001 | James Hoow    |          2818
        5006 | Mc Lyon       |          1983
        5009 | Paul White    |            75
        5002 | Nail Knite    |           450
        5003 | Lauson Hen    |          1295
        5011 | Sam Lawson    |
        5005 | Pit Alex      |           271
(7 rows)
```

36. Write a SQL query to find the salesperson(s) and the customer(s) he handle. Return Customer Name, city, Salesman, commission. You need information from salesman and customer tables.

```
mybusiness=> SELECT customer.cust_name, customer.city,
mybusiness-> salesman.name AS salesman_name,
mybusiness-> commission
mybusiness-> FROM salesman
mybusiness-> INNER JOIN customer
mybusiness-> ON salesman.salesman_id = customer.salesman_id;
   cust_name    |    city    | salesman_name | commission
----------------+------------+---------------+------------
 Nick Rimando   | New York   | James Hoow    |       0.15
 Brad Davis     | New York   | James Hoow    |       0.15
 Julian Green   | London     | Nail Knite    |       0.13
 Fabian Johnson | Paris      | Mc Lyon       |       0.14
 Geoff Cameron  | Berlin     | Lauson Hen    |       0.12
 Brad Guzan     | London     | Pit Alex      |       0.11
 Graham Zusi    | California | Nail Knite    |       0.13
 Jozy Altidor   | Moscow     | Paul White    |       0.13
(8 rows)
```

37. Write a SQL query to find those salespersons do not live in the same city where their customers live and received a commission from the company more than 12%. Return Customer Name, customer city, Salesman, salesman city, commission.

```
mybusiness=> SELECT customer.cust_name AS customer_name,
mybusiness-> customer.city AS customer_city,
mybusiness-> salesman.name AS salesman_name,
mybusiness-> salesman.city AS salesman_city,
mybusiness-> salesman.commission
mybusiness-> FROM salesman
mybusiness-> INNER JOIN customer
mybusiness-> ON salesman.salesman_id = customer.salesman_id
mybusiness-> WHERE salesman.city != customer.city AND commission > 0.12;
 customer_name | customer_city | salesman_name | salesman_city | commission
---------------+---------------+---------------+---------------+------------
 Julian Green  | London        | Nail Knite    | Paris         |       0.13
 Graham Zusi   | California    | Nail Knite    | Paris         |       0.13
 Jozy Altidor  | Moscow        | Paul White    | London        |       0.13
(3 rows)
```

38. Find all the orders issued by the salesman 'Paul Adam'. Return ord_no, purch_amt, ord_date, customer_id and salesman_id. You can use subquery or join.

Paul Adam doesn't exist, So I used Paul White. (See task 10)

With join:

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id, orders.salesman_id
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id
mybusiness-> WHERE salesman.name = 'Paul White';
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70011 |     75.29 | 2012-08-17 |        3003 |        5009
(1 row)
```

With subquery:

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id, orders.salesman_id
mybusiness-> FROM orders
mybusiness-> WHERE orders.salesman_id =
mybusiness-> (SELECT salesman.salesman_id FROM salesman WHERE salesman.name = 'Paul White');
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70011 |     75.29 | 2012-08-17 |        3003 |        5009
(1 row)
```

39. Write a SQL query to find all the orders, which are generated by those salespeople, who live in the city of London. Return ord_no, purch_amt, ord_date, customer_id, salesman_id. You can use subquery or join.

With join:

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id, orders.salesman_id
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id
mybusiness-> WHERE salesman.city = 'London';
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70009 |    270.65 | 2012-09-10 |        3001 |        5005
  70011 |     75.29 | 2012-08-17 |        3003 |        5009
(2 rows)
```

With subquery:

```
mybusiness=> SELECT orders.ord_no, orders.purch_amt, orders.ord_date, orders.customer_id, orders.salesman_id
mybusiness-> FROM orders
mybusiness-> WHERE orders.salesman_id IN
mybusiness-> (SELECT salesman.salesman_id FROM salesman WHERE salesman.city = 'London');
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70009 |    270.65 | 2012-09-10 |        3001 |        5005
  70011 |     75.29 | 2012-08-17 |        3003 |        5009
(2 rows)
```

40. Write a SQL query to find the orders generated by the salespeople who works for customers whose id is 3007. Return ord_no, purch_amt, ord_date, customer_id, salesman_id. A customer can works only with a salespeople.

```
mybusiness=> SELECT * FROM
mybusiness-> orders
mybusiness-> WHERE salesman_id IN
mybusiness-> (SELECT customer.salesman_id FROM customer WHERE customer.customer_id = 3007);
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70002 |     65.26 | 2012-10-05 |        3002 |        5001
  70005 |    2400.6 | 2012-07-27 |        3007 |        5001
  70008 |      5760 | 2012-09-10 |        3002 |        5001
  70013 |    3045.6 | 2012-04-25 |        3002 |        5001
(4 rows)
```

41. Write a SQL query to find the order values greater than the average order value of 10th October 2012. Return ord_no, purch_amt, ord_date, customer_id, salesman_id.

```
mybusiness=> SELECT *
mybusiness-> FROM orders
mybusiness-> WHERE purch_amt >
mybusiness-> (SELECT AVG(purch_amt) FROM orders WHERE ord_date = '2012-10-10');
 ord_no | purch_amt |  ord_date  | customer_id | salesman_id
--------+-----------+------------+-------------+-------------
  70005 |    2400.6 | 2012-07-27 |        3007 |        5001
  70008 |      5760 | 2012-09-10 |        3002 |        5001
  70003 |    2480.4 | 2012-10-10 |        3009 |        5003
  70013 |    3045.6 | 2012-04-25 |        3002 |        5001
(4 rows)
```

42. Write a SQL query to find the commission of the salespeople work in Paris City (i.e. whose customer is in Paris). Return salesman id, salesman name and commission.

```
mybusiness=> SELECT salesman_id,
mybusiness-> name AS salesman_name,
mybusiness-> commission
mybusiness-> FROM salesman
mybusiness-> WHERE salesman_id IN
mybusiness-> (SELECT salesman_id FROM customer WHERE city = 'Paris');
 salesman_id | salesman_name | commission
-------------+---------------+------------
        5006 | Mc Lyon       |       0.14
(1 row)
```

43. Create a view for those salespersons living in the city 'Paris’

```
mybusiness=> CREATE VIEW Parisians_view AS
mybusiness-> SELECT *
mybusiness-> FROM salesman
mybusiness-> WHERE city = 'Paris';
CREATE VIEW
mybusiness=> SELECT * FROM parisians_view;
 salesman_id |    name    | city  | commission
-------------+------------+-------+------------
        5002 | Nail Knite | Paris |       0.13
        5006 | Mc Lyon    | Paris |       0.14
(2 rows)
```

44. create a view to compute average purchase amount and total purchase amount for each salesperson. Return name, average purchase and total purchase amount. (Assume all names are unique).

```
mybusiness=> DROP VIEW averages_view ;
DROP VIEW
mybusiness=> CREATE VIEW averages_view AS
mybusiness-> SELECT salesman.name, ROUND(AVG(purch_amt), 2), SUM(purch_amt)
mybusiness-> FROM orders
mybusiness-> INNER JOIN salesman
mybusiness-> ON orders.salesman_id = salesman.salesman_id
mybusiness-> GROUP BY name;
CREATE VIEW
mybusiness=> SELECT * FROM averages_view;
    name    |  round  |   sum
------------+---------+----------
 Paul White |   75.29 |    75.29
 Pit Alex   |  270.65 |   270.65
 Nail Knite |  449.82 |  1349.45
 James Hoow | 2817.87 | 11271.46
 Lauson Hen | 1295.45 |   2590.9
 Mc Lyon    | 1983.43 |  1983.43
(6 rows)
```