Full stack Development

Week:14

14.a

From the following tables write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust_name and city.

Sample 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
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
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		5005

Sample Solution:

```
SELECT salesman.name AS "Salesman",

customer.cust_name, customer.city

FROM salesman,customer

WHERE salesman.city=customer.city;
```

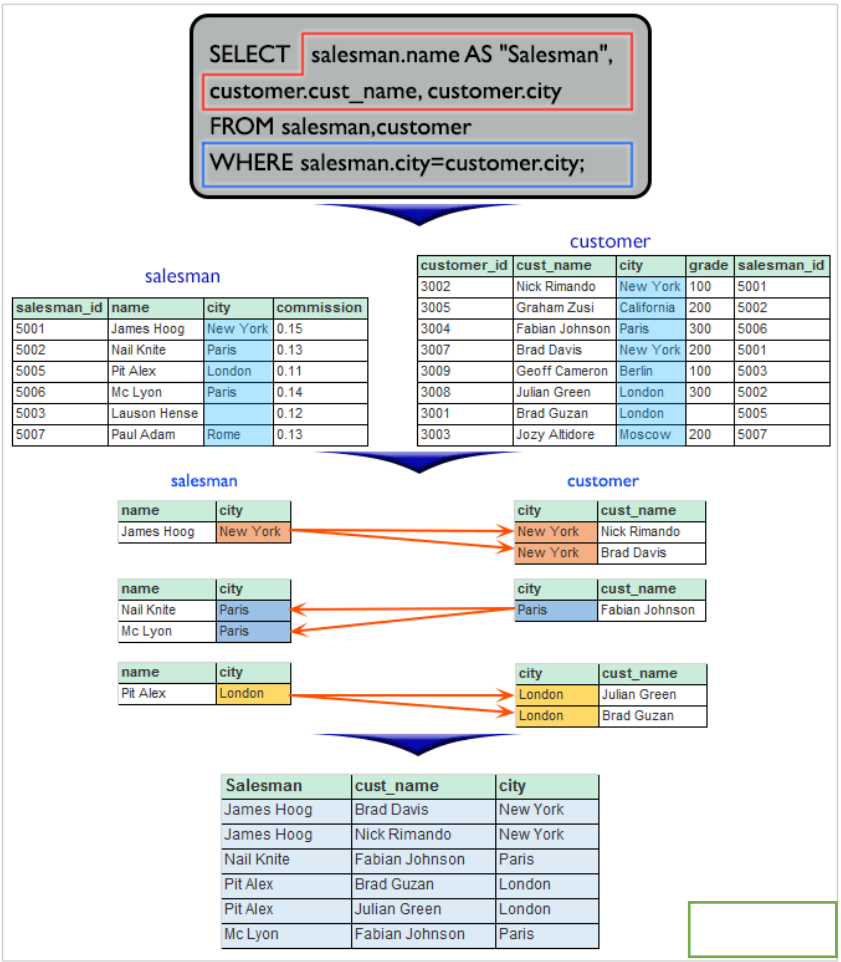
Output of the Query:

```
Salesman        cust_name       city
James Hoog      Nick Rimando    New York
James Hoog      Brad Davis      New York
Pit Alex        Julian Green    London
Mc Lyon         Fabian Johnson  Paris
Nail Knite      Fabian Johnson  Paris
Pit Alex        Brad Guzan      London
```

Explanation:

The said SQL query is selecting the name of the salesman, the customer's name, and the customer's city from the salesman and customer tables, and only displaying results where the city of the salesman matches the city of the customer. The column "name" of the "salesman" table is given an alias of "Salesman".

Visual Explanation:



14.b

From the following tables write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord_no, purch_amt, cust_name, city.

Sample table: 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
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
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		5005

Sample Solution:

```
SELECT  a.ord_no,a.purch_amt,

b.cust_name,b.city

FROM orders a,customer b

WHERE a.customer_id=b.customer_id

AND a.purch_amt BETWEEN 500 AND 2000;
```

Output of the Query:

```
ord_no   purch_amt        cust_name        city
70007    948.50           Graham Zusi      California
70010    1983.43          Fabian Johnson   Paris
```

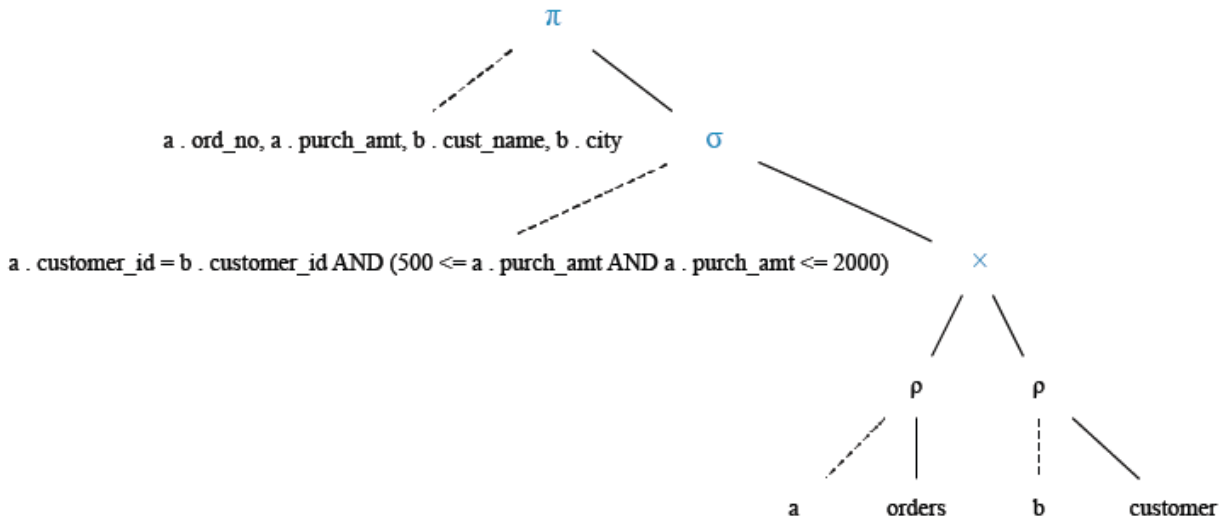
Explanation:

The said SQL query is selecting the order number, purchase amount, customer name and customer city from the "orders" table as alias a and "customer" table as alias b, and only displaying results where the customer ID of the "orders" table matches the customer ID of the "customer" table and the purchase amount is between 500 and 2000.

Relational Algebra Expression:

$$\pi_{a.ord\_no,\ a.purch\_amt,\ b.cust\_name,\ b.city}$$
$$\sigma_{a.customer\_id = b.customer\_id\ AND\ (500 <= a.purch\_amt\ AND\ a.purch\_amt <= 2000)}$$
$$(\rho_{a}\ orders \times$$
$$\rho_{b}\ customer)$$

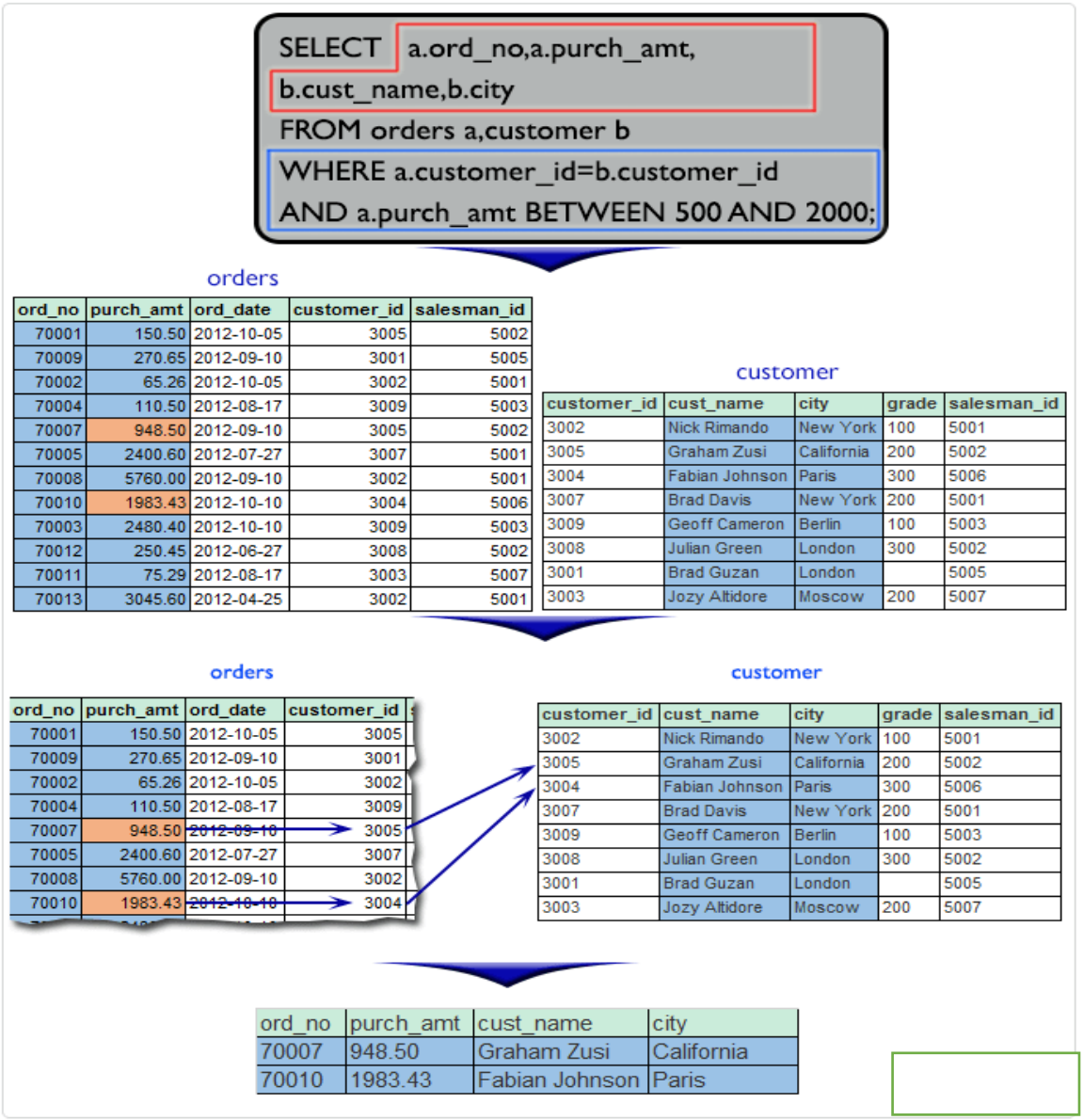
Relational Algebra Tree:



Explanation:

```
SELECT a.ord_no,a.purch_amt,
b.cust_name,b.city
FROM orders a,customer b
WHERE a.customer_id=b.customer_id
AND a.purch_amt BETWEEN 500 AND 2000;
```

Visual Explanation:



ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.50	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.50	2012-08-17	3009	5003
70007	948.50	2012-09-10	3005	5002
70005	2400.60	2012-07-27	3007	5001
70008	5760.00	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.40	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.60	2012-04-25	3002	5001

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3004	Fabian Johnson	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Cameron	Berlin	100	5003
3008	Julian Green	London	300	5002
3001	Brad Guzan	London		5005
3003	Jozy Altidore	Moscow	200	5007

ord_no	purch_amt	cust_name	city
70007	948.50	Graham Zusi	California
70010	1983.43	Fabian Johnson	Paris

14.c

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 determine whether any of the existing customers have placed an order or not.

Sample table: 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
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
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		5005

Sample Solution:

```
SELECT a.cust_name,a.city, b.ord_no,

b.ord_date,b.purch_amt AS "Order Amount"

FROM customer a
```

```
LEFT OUTER JOIN orders b

ON a.customer_id=b.customer_id

order by b.ord_date;
```

Output of the Query:

```
cust_name       city            ord_no  ord_date        Order Amount
Nick Rimando    New York        70013   2012-04-25      3045.60
Julian Green    London          70012   2012-06-27      250.45
Brad Davis      New York        70005   2012-07-27      2400.60
Jozy Altidor    Moscow          70011   2012-08-17      75.29
Geoff Cameron   Berlin          70004   2012-08-17      110.50
Brad Guzan      London          70009   2012-09-10      270.65
Nick Rimando    New York        70008   2012-09-10      5760.00
Graham Zusi     California      70007   2012-09-10      948.50
Graham Zusi     California      70001   2012-10-05      150.50
Nick Rimando    New York        70002   2012-10-05      65.26
Fabian Johnson  Paris           70010   2012-10-10      1983.43
Geoff Cameron   Berlin          70003   2012-10-10      2480.40
```

Explanation:

The said SQL query is selecting the customer name, city from the customer table aliased as a and the order number, order date and purchase amount from the orders table aliased as b. It is joining these tables on the 'customer_id' column, and the results are ordered by the 'ord_date' column.
Additionally, it is using a LEFT OUTER JOIN which will retrieve all records from the left table and the matching records from the right table. If no match is found on the right table, it will return NULL for the right table's fields.

Visual Explanation:

```
SELECT a.cust_name,a.city,b.ord_no,
b.ord_date,b.purch_amt AS "Order Amount"
FROM customer a
LEFT OUTER JOIN orders b
ON a.customer_id=b.customer_id
order by b.ord_date;
```

customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3004	Fabian Johnson	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Cameron	Berlin	100	5003
3008	Julian Green	London	300	5002
3001	Brad Guzan	London		5005
3003	Jozy Altidore	Moscow	200	5007

orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.50	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.50	2012-08-17	3009	5003
70007	948.50	2012-09-10	3005	5002
70005	2400.60	2012-07-27	3007	5001
70008	5760.00	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.40	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.60	2012-04-25	3002	5001

cust_name	city	customer_id	customer_id	...	ord_no	ord_date	purch_amt
Nick Rimando	New York	3002	3002	...	70002	2012-10-05	65.26
			3002	...	70008	2012-09-10	5760.00
			3002	...	70013	2012-04-25	3045.60
Graham Zusi	California	3005	3005	...	70001	2012-10-05	150.50
			3005	...	70007	2012-09-10	948.50
Fabian Johnson	Paris	3004	3004	...	70010	2012-10-10	1983.43
Brad Davis	New York	3007	3007	...	70005	2012-07-27	2400.60
Geoff Cameron	Berlin	3009	3009	...	70004	2012-08-17	110.50
			3009	...	70003	2012-10-10	2480.40
Julian Green	London	3008	3008	...	70012	2012-06-27	250.45
Brad Guzan	London	3001	3001	...	70009	2012-09-10	270.65
Jozy Altidore	Moscow	3003	3003	...	70011	2012-08-17	75.29

cust_name	city	ord_no	ord_date	Order Amount
Nick Rimando	New York	70013	4/25/2012	3045.6
Julian Green	London	70012	6/27/2012	250.45
Brad Davis	New York	70005	7/27/2012	2400.6
Jozy Altidore	Moscow	70011	8/17/2012	75.29
Geoff Cameron	Berlin	70004	8/17/2012	110.5
Nick Rimando	New York	70008	9/10/2012	5760
Brad Guzan	London	70009	9/10/2012	270.65
Graham Zusi	California	70007	9/10/2012	948.5
Graham Zusi	California	70001	10/5/2012	150.5
Nick Rimando	New York	70002	10/5/2012	65.26
Geoff Cameron	Berlin	70003	10/10/2012	2480.4
Fabian Johnson	Paris	70010	10/10/2012	1983.43

14.d

From the following tables write a SQL query to calculate and find the average price of items of each company higher than or equal to Rs. 350. Return average value and company name.

Sample table: company_mast

```
COM_ID COM_NAME
------ -------------
    11 Samsung
    12 iBall
    13 Epsion
    14 Zebronics
    15 Asus
    16 Frontech
```

Sample table: item_mast

```
PRO_ID PRO_NAME                   PRO_PRICE    PRO_COM
------- ------------------------- ------------ ----------
    101 Mother Board                  3200.00         15
    102 Key Board                      450.00         16
    103 ZIP drive                      250.00         14
    104 Speaker                        550.00         16
    105 Monitor                       5000.00         11
    106 DVD drive                      900.00         12
    107 CD drive                       800.00         12
    108 Printer                       2600.00         13
    109 Refill cartridge               350.00         13
    110 Mouse                          250.00         12
```

Sample Solution:

```
SELECT AVG(pro_price), company_mast.com_name
  FROM item_mast INNER JOIN company_mast
  ON item_mast.pro_com= company_mast.com_id
  GROUP BY company_mast.com_name
  HAVING AVG(pro_price) >= 350;
```

Output of the Query:

```
          avg           | com_name
------------------------+----------
 5000.0000000000000000 | Samsung
  650.0000000000000000 | iBall
 1475.0000000000000000 | Epsion
  500.0000000000000000 | Frontech
 3200.0000000000000000 | Asus
(5 rows)
```

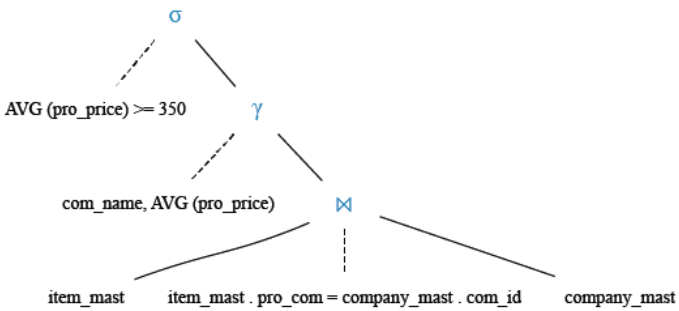
Explanation:

The said SQL query is selecting the average price (AVG(pro_price)) of items and the name of the company (company_mast.com_name) they are associated with, by joining the item_mast and company_mast tables on the pro_com column in the item_mast table and the com_id column in the company_mast table. The results are grouped by the company name and only showing the results where the average price is greater than or equal to 350.

Relational Algebra Expression:

$$\sigma_{AVG\ (pro\_price)\ >=\ 350}$$
$$\gamma_{com\_name\ ,\ AVG\ (pro\_price)}\ (item\_mast \bowtie item\_mast\ .\ pro\_com = company\_mast\ .\ com\_id\ company\_mast)$$

Relational Algebra Tree:



Results:

Thus, in the above SQL queries successfully executed without errors

Using XAMPP Server with Mysql Dashboard on SQL operations