Bases de Datos relacionales

2023-01-17

Introducción

Iniciemos creando una conexión al archivo SQL

```
mydb <- dbConnect(RSQLite::SQLite(), "my-db.sqlite")
dbListTables(mydb)
```

```
## [1] "company_mast"   "customer"        "emp_department" "
## [5] "item_mast"      "orders"          "salesman"
```

```
dbListFields(mydb, "salesman")
```

```
## [1] "salesman_id" "name"          "city"          "commissio
```

Tablas

Table 1: 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

Table 2: 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	NA	5005

Table 3: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.50	1349395200	3005	5002
70009	270.65	1347235200	3001	5005
70002	65.26	1349395200	3002	5001
70004	110.50	1345161600	3009	5003
70007	948.50	1347235200	3005	5002
70005	2400.60	1343347200	3007	5001
70008	5760.00	1347235200	3002	5001
70010	1983.43	1349827200	3004	5006
70003	2480.40	1349827200	3009	5003
70012	250.45	1340755200	3008	5002
70011	75.29	1345161600	3003	5007
70013	3045.60	1335312000	3002	5001

Table 4: company mast

com_id	com_name
11	Samsung
12	iBall
13	Epsion
14	Zebronics
15	Asus
16	Frontech

Table 5: item mast

pro_id	pro_name	pro_price	pro_com
101	Mother Board	3200	15
102	Key Board	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 6: emp department

dpt_code	dpt_name	dpt_allotment
57	IT	65000
63	Finance	15000
47	HR	240000
27	RD	55000
89	QC	75000

Table 7: emp details

emp_idno	emp_fname	emp_lname	emp_dept
127323	Michale	Robbin	57
526689	Carlos	Snares	63
843795	Enric	Dosio	57
328717	Jhon	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

Ejercicios

1. 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. (salesman, customer)
2. 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. (orders, customer)
3. From the following tables write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission. (customer, salesman)
4. From the following tables write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission. (customer, salesman)

5. From the following tables write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission. (customer, salesman)

6. From the following tables write a SQL query to find the details of an order. Return ord_no, ord_date, purch_amt, Customer Name, grade, Salesman, commission. (orders, customer, salesman)

7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned. (orders, customer, salesman)

8. From the following tables write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer_id. (customer, salesman)

9. From the following tables write a SQL query to find those customers with a grade less than 300. Return cust_name, customer city, grade, Salesman, salesmancity. The result should be ordered by ascending customer_id. (customer, salesman)

10. 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. (orders, customer)

11. SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves. (customer, orders, salesman)

12. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers. (customer, salesman)

13. From the following tables write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount. (customer, salesman, orders)

14. 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. (customer, salesman, orders)

15. For those customers from the existing list who put one or more orders, or which orders have been placed by the customer who is not on the list, create a report containing the customer name, city, order number, order date, and purchase amount. (customer, orders)

16. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade. (customer, orders)

17. Write a SQL query to combine each row of the salesman table with each row of the customer table. (salesman, customer)

18. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city. (salesman, customer)

```
dbDisconnect(mydb)
```