**Questions**

1. Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. This time only for the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

**SELECT region.name as Region\_name,sales\_reps.name as Rep\_name,accounts.name as Account\_name**

**FROM sales\_reps**

**JOIN region**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**WHERE region.name = 'Midwest'**

**ORDER BY Account\_name;**

1. Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a first name starting with S and in the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

**SELECT region.name as Region\_name,sales\_reps.name as Rep\_name,accounts.name as Account\_name**

**FROM sales\_reps**

**JOIN region**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**WHERE (sales\_reps.name LIKE 'S%' AND region.name = 'Midwest')**

**ORDER BY Account\_name;**

1. Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a **last** name starting with K and in the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

**SELECT region.name as Region\_name,sales\_reps.name as Rep\_name,accounts.name as Account\_name**

**FROM sales\_reps**

**JOIN region**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**WHERE (sales\_reps.name LIKE '% K%' AND region.name = 'Midwest')**

**ORDER BY Account\_name;**

1. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total\_amt\_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. In order to avoid a division by zero error, adding .01 to the denominator here is helpful total\_amt\_usd/(total+0.01).

**SELECT region.name as region\_name,accounts.name as account\_name,(total\_amt\_usd/(total+0.01)) as unit\_price**

**FROM region**

**JOIN sales\_reps**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**JOIN orders**

**ON accounts.id=orders.account\_id**

**WHERE orders.standard\_qty>100**

1. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total\_amt\_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100 and the **poster order quantity** exceeds 50. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. Sort for the smallest **unit price** first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total\_amt\_usd/(total+0.01).

**SELECT region.name as region\_name,accounts.name as account\_name,(total\_amt\_usd/(total+0.01)) as unit\_price**

**FROM region**

**JOIN sales\_reps**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**JOIN orders**

**ON accounts.id=orders.account\_id**

**WHERE orders.standard\_qty>100 AND orders.poster\_qty > 50**

**ORDER BY unit\_price**

1. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total\_amt\_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100 and the **poster order quantity** exceeds 50. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. Sort for the largest **unit price** first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total\_amt\_usd/(total+0.01).

**SELECT region.name as region\_name,accounts.name as account\_name,(total\_amt\_usd/(total+0.01)) as unit\_price**

**FROM region**

**JOIN sales\_reps**

**ON region.id=sales\_reps.region\_id**

**JOIN accounts**

**ON sales\_reps.id=accounts.sales\_rep\_id**

**JOIN orders**

**ON accounts.id=orders.account\_id**

**WHERE orders.standard\_qty>100 AND orders.poster\_qty > 50**

**ORDER BY unit\_price DESC**

1. What are the different **channel**s used by **account id** 1001? Your final table should have only 2 columns: **account name** and the different **channel**s. You can try **SELECT DISTINCT** to narrow down the results to only the unique values.

**SELECT DISTINCT accounts.name as Account\_name, web\_events.channel as Channel**

**FROM accounts**

**JOIN web\_events**

**ON web\_events.account\_id=accounts.id**

**WHERE accounts.id=1001**

1. Find all the orders that occurred in 2015. Your final table should have 4 columns: **occurred\_at**, **account name**, **order total**, and **order total\_amt\_usd**.

SELECT orders.occurred\_at as Date, accounts.name as Account\_name,orders.total as Order\_Total, orders.total\_amt\_usd as Order\_Total\_USD

FROM orders

JOIN accounts

ON accounts.id=orders.account\_id

WHERE (SELECT EXTRACT(YEAR FROM orders.occurred\_at)) =2015