# Matt Markezin-Press

## 1. Write the SQL to generate a report of the total quantity and amount paid for each purchase. The result should have the following columns. (Difficulty: Easy)

* customer\_id
* first\_name
* last\_name
* email
* purchase\_id
* purchase\_time
* total\_quantity
* total\_amount\_paid

SELECT customer.first\_name, customer.last\_name, customer.email, purchase.purchase\_id, purchase.purchase\_time,

sum(purchase\_item.total\_amount\_paid) as total\_amount\_paid, sum(purchase\_item.quantity) as total\_quantity

FROM purchase

JOIN purchase\_item

ON purchase.purchase\_id=purchase\_item.purchase\_id

JOIN customer

on purchase.customer\_id=customer.customer\_id

GROUP BY purchase.purchase\_id;

## 2. Write the SQL to generate a report of customers who have made more than 1 purchase, sorted by the total number of purchases in descending order. The result should have the following columns. (Difficulty: Medium)

* customer\_id
* first\_name
* last\_name
* email
* number\_of\_purchases

SELECT customer.first\_name, customer.last\_name, customer.email, count(customer.customer\_id) as number\_of\_purchases

FROM purchase

JOIN customer

ON purchase.customer\_id=customer.customer\_id

GROUP BY purchase.customer\_id

HAVING count(customer.customer\_id) > 1

## 3. Write the SQL to generate a report of the number of customers who have purchased each sku. Any skus with no purchases don't need to be included in the report. The result should have the resulting columns. (Difficulty: Medium)

* sku
* total\_number\_of\_customers

SELECT sku, count(purchase.customer\_id) as total\_number\_of\_customers

FROM purchase\_item

JOIN purchase

ON purchase\_item.purchase\_id=purchase.purchase\_id

GROUP BY sku

HAVING total\_number\_of\_customers > 0

## 4. Write the SQL to generate a report of all customers and the total amount they’ve spent. Users who haven’t made any purchases should still be included in the report. The result should have the following columns. (Difficulty: Medium)

* customer\_id
* first\_name
* last\_name
* email
* total\_amount\_paid

SELECT customer.customer\_id, customer.first\_name, customer.last\_name, customer.email, sum(purchase\_item.total\_amount\_paid) as total\_amount\_paid

FROM purchase

JOIN purchase\_item

ON purchase.purchase\_id=purchase\_item.purchase\_id

FULL OUTER JOIN customer

ON purchase.customer\_id=customer.customer\_id

GROUP BY customer.customer\_id

## 5. Write the SQL to show the some basic statistics about two types of purchases: those that include a bike (has a purchase\_item with sku = ‘bike’) and those that do not. For these two cases, we’re interested in knowing how many purchases there are, the average amount paid across those purchases (known as AOV for “average order value”), and the average quantity in each purchase. The result should be two rows with the following columns. (Difficulty: Hard)

* purchase\_has\_bike (boolean)
* num\_purchases
* avg\_amount\_paid
* avg\_quantity

WITH bike\_purchases (quantity, amount\_paid, purchase\_has\_bike) AS

(

SELECT purchase\_item.quantity, purchase\_item.total\_amount\_paid,

CASE

WHEN purchase\_item.sku = 'bike' THEN 'Yes'

ELSE 'No'

END AS purchase\_has\_bike

FROM purchase

JOIN purchase\_item

ON purchase.purchase\_id=purchase\_item.purchase\_id

)

SELECT purchase\_has\_bike, count(purchase\_has\_bike) as num\_purchases, avg(amount\_paid) as avg\_amount\_paid, avg(quantity) as avg\_quantity

FROM bike\_purchases

GROUP BY purchase\_has\_bike

## 6. Write the SQL to generate a report of customers with their most recent purchase. You don’t need to include customers who haven’t made a purchase. The result should have the following columns. (Hint: you can assume the underlying database supports [window functions](https://www.postgresql.org/docs/10/static/functions-window.html). But it is still possible without window functions.) (Difficulty: Hard)

* customer\_id
* first\_name
* last\_name
* email
* purchase\_id
* purchase\_time
* purchase\_quantity
* purchase\_amount\_paid

WITH purchase\_group (purchase\_id, purchase\_time, customer\_id, purchase\_quantity, purchase\_amount\_paid) AS

(

SELECT purchase.purchase\_id, purchase\_time, customer\_id, sum(purchase\_item.quantity) as purchase\_quantity, sum(purchase\_item.total\_amount\_paid) as purchase\_amount\_paid

FROM purchase

JOIN purchase\_item

ON purchase.purchase\_id=purchase\_item.purchase\_id

GROUP BY purchase.purchase\_id

)

SELECT first\_name, last\_name, email, purchase\_id, most\_recent\_purchase, purchase\_group.customer\_id, purchase\_quantity, purchase\_amount\_paid

FROM purchase\_group

JOIN

(

SELECT customer\_id, max(purchase\_group.purchase\_time) as most\_recent\_purchase

FROM purchase\_group

GROUP BY purchase\_group.customer\_id

) as recent\_purchase

ON purchase\_group.purchase\_time=recent\_purchase.most\_recent\_purchase

JOIN customer

ON customer.customer\_id=purchase\_group.customer\_id