**Get all the products available in the market.**

SELECT \* FROM `farmers\_market.product`

Select five product IDs and their associated product

names from the product table.

SELECT product\_id, product\_name

FROM `farmers\_market.product`

LIMIT 5

If I ask you to skip the first 2 records and display the 10

rows afterwards?

SELECT product\_id, product\_name

FROM `farmers\_market.product`

LIMIT 10

OFFSET 2

Sort this entire data in ascending order of by a particular

column, in this case ‘product\_id’

SELECT product\_id, product\_name

FROM `farmers\_market.product`

ORDER BY product\_id

Show only the first 10 of the sorted data

SELECT product\_id, product\_name

FROM `farmers\_market.product`

ORDER BY product\_id

LIMIT 10

Question: Explore vendor\_booth\_assignments.

List down 10 rows of farmer’s market vendor booth assignments, displaying the

market date, vendor ID, and booth number from the vendor\_booth\_assignments

table.

SELECT

market\_date,

vendor\_id,

booth\_number

FROM

farmers\_market.vendor\_booth\_assignments

LIMIT 10

Sort the data by date.

SELECT

market\_date,

vendor\_id,

booth\_number

FROM

farmers\_market.vendor\_booth\_assignments

ORDER BY market\_date

Limit 10

For each market\_date in the vendor\_booth\_invetory, sort

the data by vendor\_id in descending order.

SELECT market\_date,

vendor\_id,

booth\_number

FROM

farmers\_market.vendor\_booth\_assignments

ORDER BY

market\_date,

vendor\_id DESC

Question: In the customer purchases, we have quantity and cost

per qty separate,

Query the total amount that the customer has paid along with

date, customer id, vendor\_id, qty, cost per qty and the total amt.?

SELECT

market\_date,

customer\_id,

vendor\_id,

quantity,

cost\_to\_customer\_per\_qty,

quantity \* cost\_to\_customer\_per\_qty

FROM farmers\_market.customer\_purchases

**# With Alias**

SELECT

market\_date,

customer\_id,

vendor\_id,

quantity,

cost\_to\_customer\_per\_qty,

(quantity \* cost\_to\_customer\_per\_qty) as price

FROM farmers\_market.customer\_purchases;

Round off all the values in the result set to 2 decimal

places?

SELECT

market\_date,

customer\_id,

vendor\_id,

ROUND(quantity \* cost\_to\_customer\_per\_qty, 2) AS price

FROM farmers\_market.customer\_purchases

Other function like ceil, floor

SELECT

market\_date,

customer\_id,

vendor\_id,

ROUND(quantity \* cost\_to\_customer\_per\_qty, 2) AS price

ceil(quantity \* cost\_to\_customer\_per\_qty) AS price\_ceil

floor(quantity \* cost\_to\_customer\_per\_qty) AS price\_floor

FROM farmers\_market.customer\_purchases;

Merge each customer’s name into a single column that contains

the first name, then a space, and then the last name.

SELECT

customer\_id,

CONCAT(customer\_first\_name, " ", customer\_last\_name) AS full\_name

FROM farmers\_market.customer

In a customer’s full\_name, we want the first\_name to be in upper case followed

by the last\_name as it is.

SELECT

customer\_id,

CONCAT(UPPER(customer\_first\_name), " ", customer\_last\_name) AS full\_name

FROM farmers\_market.customer

Display only the first character of someone’s last\_name?

SELECT

customer\_id,

SUBSTR(customer\_last\_name, 1, 1) as last\_name\_1\_char

FROM farmers\_market.customer;

Capitalize the full\_name of a customer. First character of the

first\_name & last\_name should be in uppercase. All other

characters should be in lowercase.

Using upper, lower, substr, concat

SELECT

customer\_id,

customer\_first\_name,

customer\_last\_name,

CONCAT(UPPER(SUBSTR(customer\_first\_name, 1, 1)),

LOWER(SUBSTR(customer\_first\_name, 2)), " ",

UPPER(SUBSTR(customer\_last\_name, 1, 1)),

LOWER(SUBSTR(customer\_last\_name, 2))) AS capitalised\_full\_name

FROM `farmers\_market.customer`

# Simpler function using **INITCAP** in BigQuery

SELECT

customer\_id,

customer\_first\_name,

customer\_last\_name,

CONCAT(INITCAP(customer\_first\_name)," ", INITCAP(customer\_last\_name)) AS capitalised\_full\_name

FROM `farmers\_market.customer`

LIMIT 5

Filtering and Subquery:

Print a report of everything the customer\_id 4 has ever purchased

at the market, sorted by date. Add total\_amt column as well for

each purchase.

SELECT

customer\_id, market\_date, quantity,

cost\_to\_customer\_per\_qty,

ROUND(quantity \* cost\_to\_customer\_per\_qty, 2) AS total\_amt

FROM `farmers\_market.customer\_purchases`

WHERE customer\_id = 4

ORDER BY market\_date ASC

Get all the product info for products with id between 3 and 8 (not

inclusive) and of products with id 10.

SELECT \*

FROM `farmers\_market.product`

WHERE (product\_id > 3 AND product\_id < 8) OR product\_id =10

Find the booth assignments for vendor\_id 7 for all dates between

April 3, 2019 and May 16, 2019, including the 2 dates.

SELECT \*

FROM `farmers\_market.vendor\_booth\_assignments`

WHERE vendor\_id = 7 AND

(market\_date >= "2019-04-03" AND market\_date <= "2019-05-18")

SELECT \*

FROM `farmers\_market.vendor\_booth\_assignments`

WHERE vendor\_id = 7 AND

(market\_date BETWEEN "2019-04-03" AND "2019-05-18")

Find the customer detail with the first name of “Carlos” or the last

name of “Diaz,”:

SELECT

customer\_id,

customer\_first\_name,

customer\_last\_name

FROM farmers\_market.customer

Question: You want to get data about a customer you knew as

Jerry but you are not sure if they are listed as Jeremy or Jeremiah or

Jerry. Get all customers whose name starts with “jer”.

SELECT \*

FROM `farmers\_market.customer`

WHERE lower(customer\_first\_name) LIKE "jer%"

What if we want a customers with minimum 5 characters and first 3 chars as “jer”

SELECT \*

FROM `farmers\_market.customer`

WHERE lower(customer\_first\_name) LIKE "jer\_\_%"

WHERE

customer\_last\_name = 'Diaz'

OR customer\_last\_name = 'Edwards'

OR customer\_last\_name = 'Wilson'

**Using IN**

SELECT

customer\_id,

customer\_first\_name,

customer\_last\_name

FROM farmers\_market.customer

WHERE

customer\_last\_name IN ('Diaz' , 'Edwards', 'Wilson')

SELECT

customer\_id,

customer\_first\_name,

customer\_last\_name

FROM farmers\_market.customer

WHERE LOWER(customer\_last\_name) IN ('diaz' , 'edwards', 'Wilson')

**You can also use**

UPPER(customer\_last\_name) IN ('DIAZ' , 'EDWARDS', 'WILSON')

Find all of the products from the product table which don’t have

sizes mentioned.

SELECT \*

FROM farmers\_market.product

WHERE

product\_size IS NULL

OR TRIM(product\_size) = “”

Put the total cost to customer purchases into bins of -

● under $5.00,

● $5.00–$9.99,

● $10.00–$19.99, or

● $20.00 and over

SELECT

market\_date, customer\_id,

quantity, cost\_to\_customer\_per\_qty,

CASE

WHEN quantity \* cost\_to\_customer\_per\_qty < 5 THEN "Under $5"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 5 AND 9.99 THEN "$5 - $9.99"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 10 AND 19.99 THEN "$10 - $19.99"

ELSE "Above $20"

END AS price\_bin

FROM `farmers\_market.customer\_purchases`;

Find out which vendors primarily sell fresh products and which

don’t

SELECT

vendor\_id,

vendor\_name,

vendor\_type,

CASE

WHEN LOWER(vendor\_type) LIKE '%fresh%'

THEN 'Fresh Produce'

ELSE 'Other' END AS category

FROM farmers\_market.vendor;

What if we want to add 1 for vendors who sell fresh products and 0

for those who don’t?

SELECT vendor\_id, vendor\_name, vendor\_type,

CASE

WHEN LOWER(vendor\_type) LIKE "%fresh%" THEN 1

ELSE 0

END AS category

FROM `farmers\_market.vendor`;

We can also do the same using an IF statement -

SELECT \*,

IF(LOWER(vendor\_type) LIKE "%fresh%", "Fresh", "Not Fresh") AS type

FROM `farmers\_market.vendor`

SELECT \*,

IF(LOWER(vendor\_type) LIKE "%fresh%", 1, 0) AS type FROM

`farmers\_market.vendor`;

Find out purchases made at the market on days when it rained.

SELECT

\*

FROM `farmers\_market.customer\_purchases`

WHERE market\_date IN (

SELECT

market\_date

FROM `farmers\_market.mdi`

WHERE market\_rain\_flag = 1)

List down all the product details where product\_category\_name contains

fresh in it.

SELECT \*

FROM `farmers\_market.product`

WHERE product\_category\_id IN (

SELECT product\_category\_id

FROM `farmers\_market.product\_category`

WHERE lower(product\_category\_name) LIKE "%fresh%"

)

List all the products along with their product\_category names.

SELECT

p.product\_id,

p.product\_name,

pc.product\_category\_id,

pc.product\_category\_name

FROM `farmers\_market.product` AS p

LEFT JOIN `farmers\_market.product\_category` AS pc

ON p.product\_category\_id = pc.product\_category\_id

Find out all the customers who have not ordered anything yet.

SELECT

c.customer\_id

FROM `farmers\_market.customer` AS c

LEFT JOIN `farmers\_market.customer\_purchases` AS cp

ON c.customer\_id = cp.customer\_id

WHERE cp.customer\_id IS NULL

Question: List all the products along with

their product category name.

SELECT \* FROM

farmers\_market.product

LEFT JOIN farmers\_market.product\_category

ON product.product\_category\_id = product\_category.product\_category\_id

**With table aliasing:**

SELECT

p.product\_id,

p.product\_name,

pc.product\_category\_id,

pc.product\_category\_name

FROM farmers\_market.product AS p

LEFT JOIN farmers\_market.product\_category AS pc

ON p.product\_category\_id = pc.product\_category\_id

Get a list of customers' zip codes who made a

purchase on 2019-04-06.

select distinct c.customer\_zip from

farmers\_market.customer c

join farmers\_market.customer\_purchases cp

on c.customer\_id = cp.customer\_id

where cp.market\_date = '2019-04-06'

Question: Get all the Customers who have

purchased nothing from the market yet.

select c.\*, cp.\*

from farmers\_market.customer c

left join farmers\_market.customer\_purchases cp on

c.customer\_id = cp.customer\_id

where cp.product\_id is NULL

**#Only customer information**

select c.\*

from farmers\_market.customer c

left join farmers\_market.customer\_purchases cp on

c.customer\_id = cp.customer\_id

where cp.product\_id is NULL;

Q: A simpler question: List all the customers and their associated

purchases?

SELECT \*

FROM farmers\_market.customer\_purchases AS cp

RIGHT JOIN farmers\_market.customer AS c

ON c.customer\_id = cp.customer\_id

Note: You can also use left join by swapping the tables

Question: Let’s say we want details about all farmer’s market booths

(even if not assigned) and every vendor booth assignment for every

market date with the details of the vendor.

SELECT

b.booth\_number,

b.booth\_type,

vba.market\_date,

v.vendor\_id,

v.vendor\_name,

v.vendor\_type

FROM farmers\_market.booth AS b

LEFT JOIN farmers\_market.vendor\_booth\_assignments AS vba

ON b.booth\_number = vba.booth\_number

LEFT JOIN farmers\_market.vendor AS v

ON v.vendor\_id = vba.vendor\_id;

Find all customers who have deleted their account and also those who

are yet to make a purchase along with active customers

**# Outer join(Big Query)**

select \* from temp\_sales.transaction t

full outer join temp\_sales.customer\_info c

on t.Customer\_id=c.customer\_id

order by c.customer\_id

Find all customers who have deleted their account and also those who

are yet to make a purchase along with active customers

**# Outer join My SQL Workbench**

(select \* from temp\_sales.transaction t left join temp\_sales.customer\_info c

on t.Customer\_id=c.customer\_id)

union

(select \* from temp\_sales.transaction t right join temp\_sales.customer\_info c

on t.Customer\_id=c.customer\_id)

Group by :

Question: Get a list of the customers who

made purchases on each market date.

SELECT

market\_date,

customer\_id

FROM farmers\_market.customer\_purchases

ORDER BY market\_date, customer\_id;

Question: Count the number of purchases

each customer made per market date.

SELECT

market\_date,

customer\_id,

COUNT(\*) AS num\_purchases

FROM farmers\_market.customer\_purchases

GROUP BY market\_date, customer\_id;

Alternate Question: Calculate the total quantity

purchased by each customer per market\_date.

SELECT

market\_date,

customer\_id,

SUM(quantity) AS total\_qty\_purchased

FROM farmers\_market.customer\_purchases

GROUP BY market\_date, customer\_id

Slightly complex question: how many different

kinds of products were purchased by each

customer on each market date:

SELECT

market\_date,

customer\_id,

COUNT(DISTINCT product\_id) AS

different\_products\_purchased

FROM farmers\_market.customer\_purchases

GROUP BY market\_date, customer\_id

ORDER BY market\_date;

Question: Calculate the total price paid by

customer\_id 3 per market\_date.

SELECT

market\_date,

SUM(quantity \*

cost\_to\_customer\_per\_qty) AS total\_spent

FROM farmers\_market.customer\_purchases

WHERE

customer\_id = 3

GROUP BY market\_date

ORDER BY market\_date;

Question: What if we wanted to find out how

much each customer had spent at each vendor,

regardless of date?

SELECT

customer\_id,

vendor\_id,

SUM(quantity \* cost\_to\_customer\_per\_qty) AS total\_spent

FROM farmers\_market.customer\_purchases

GROUP BY customer\_id, vendor\_id

ORDER BY customer\_id, vendor\_id;

Let’s add some customer details and vendor details to these results.

Customer details are in the customer table and vendor details are in the vendor

table.

SELECT

c.customer\_first\_name,

c.customer\_last\_name,

cp.customer\_id,

v.vendor\_name,

cp.vendor\_id,

ROUND(SUM(quantity \* cost\_to\_customer\_per\_qty), 2) AS total\_spent

FROM farmers\_market.customer c

LEFT JOIN farmers\_market.customer\_purchases cp

ON c.customer\_id = cp.customer\_id

LEFT JOIN farmers\_market.vendor v

ON cp.vendor\_id = v.vendor\_id

GROUP BY

cp.customer\_id,

cp.vendor\_id

ORDER BY cp.customer\_id, cp.vendor\_id;

Question: We want to get the most and least expensive

items per product category, considering the fact that

each vendor sets their own prices and can adjust prices

per customer.

SELECT

p.product\_category\_id,

MIN(vi.original\_price) AS minimum\_price,

MAX(vi.original\_price) AS maximum\_price

FROM farmers\_market.vendor\_inventory AS vi

INNER JOIN farmers\_market.product AS p

ON vi.product\_id = p.product\_id

GROUP BY p.product\_category\_id;

Question: Count how many products(can be duplicates) were for sale

on each market date:

SELECT

market\_date,

COUNT(product\_id) AS product\_count

FROM farmers\_market.vendor\_inventory

GROUP BY market\_date

ORDER BY market\_date;

how many different products each vendor offered.

between - 2019-05-02 and 2019-05-16?

SELECT

vendor\_id,

COUNT(DISTINCT product\_id) AS different\_products\_offered

FROM farmers\_market.vendor\_inventory

WHERE

market\_date BETWEEN '2019-05-02' AND '2019-05-16'

GROUP BY vendor\_id

ORDER BY vendor\_id;

Filter out vendors who brought at least 100 items from the farmer’s

market over the period - 2019-05-02 and 2022-05-16.

**Using Having clause**

SELECT

vendor\_id,

count(product\_id) AS products\_brought,

FROM farmers\_market.vendor\_inventory

WHERE market\_date BETWEEN '2019-04-03' AND '2019-05-16’

GROUP BY vendor\_id

**HAVING** products\_brought >= 100

ORDER BY vendor\_id;

**Using subqueries**

select x.vendor\_id, x.products\_brought from

(

select vendor\_id, count(product\_id) as products\_brought

from `farmers\_market.vendor\_inventory`

where market\_date between "2019-04-02" and "2022-05-16"

group by vendor\_id

) as x

where x.products\_brought>100;

Order of execution:

• FROM, including JOINs

• WHERE

• GROUP BY

• HAVING

• WINDOW functions

• SELECT

• DISTINCT

• UNION

• ORDER BY

• OFFSET

• LIMIT