Sample Data Recap

customers

id name email

1 John Doe john@example.com

2 Jane Smith jane@example.com

products

id name price customer\_id

1 iPhone 15 999 1

2 Samsung TV 700 2

3 MacBook Pro 1800 1

4 Dell Monitor 300 NULL

1-50 Examples

1. Get all customers

SQL: SELECT \* FROM customers;

Laravel: Customer::all();

Output: 2 rows, John Doe, Jane Smith

2. Get all products

SQL: SELECT \* FROM products;

Laravel: Product::all();

Output: 4 rows, iPhone, Samsung TV, MacBook, Dell Monitor

3. Get customer names only

SQL: SELECT name FROM customers;

Laravel: Customer::select('name')->get();

Output: John Doe, Jane Smith

4. Get product prices over 1000

SQL: SELECT \* FROM products WHERE price > 1000;

Laravel: Product::where('price', '>', 1000)->get();

Output: MacBook Pro

5. Get customers with email ending 'example.com'

SQL: SELECT \* FROM customers WHERE email LIKE '%example.com';

Laravel: Customer::where('email', 'like', '%example.com')->get();

Output: Both customers

6. Get products where customer\_id is null

SQL: SELECT \* FROM products WHERE customer\_id IS NULL;

Laravel: Product::whereNull('customer\_id')->get();

Output: Dell Monitor

7. Join customers and products

SQL: SELECT customers.name, products.name FROM customers JOIN products ON customers.id = products.customer\_id;

Laravel: Customer::with('products')->get();

Output: John Doe → iPhone, MacBook; Jane Smith → Samsung TV

8. Count total customers

SQL: SELECT COUNT(\*) FROM customers;

Laravel: Customer::count();

Output: 2

9. Count total products

SQL: SELECT COUNT(\*) FROM products;

Laravel: Product::count();

Output: 4

10. Get max product price

SQL: SELECT MAX(price) FROM products;

Laravel: Product::max('price');

Output: 1800

11. Get min product price

SQL: SELECT MIN(price) FROM products;

Laravel: Product::min('price');

Output: 300

12. Average product price

SQL: SELECT AVG(price) FROM products;

Laravel: Product::avg('price');

Output: 949.75

13. Sum of all product prices

SQL: SELECT SUM(price) FROM products;

Laravel: Product::sum('price');

Output: 3799

14. Group products by customer\_id, count products per customer

SQL: SELECT customer\_id, COUNT(\*) FROM products GROUP BY customer\_id;

Laravel: Product::select('customer\_id', DB::raw('COUNT(\*) as count'))->groupBy('customer\_id')->get();

Output: customer 1 → 2, customer 2 → 1, NULL → 1

15. Group products by customer\_id having count > 1

SQL: SELECT customer\_id FROM products GROUP BY customer\_id HAVING COUNT(\*) > 1;

Laravel: Product::groupBy('customer\_id')->havingRaw('COUNT(\*) > 1')->get();

Output: customer 1 only

16. Insert new customer

SQL: INSERT INTO customers (name, email) VALUES ('Alice', 'alice@example.com');

Laravel: Customer::create(['name' => 'Alice', 'email' => 'alice@example.com']);

Output: New row added

17. Insert new product for Alice (customer\_id = 3)

SQL: INSERT INTO products (name, price, customer\_id) VALUES ('iPad', 600, 3);

Laravel: Product::create(['name' => 'iPad', 'price' => 600, 'customer\_id' => 3]);

Output: New product added

18. Update Jane's email

SQL: UPDATE customers SET email = 'jane\_new@example.com' WHERE id = 2;

Laravel: Customer::where('id', 2)->update(['email' => 'jane\_new@example.com']);

Output: 1 row updated

19. Update price of Dell Monitor to 350

SQL: UPDATE products SET price = 350 WHERE name = 'Dell Monitor';

Laravel: Product::where('name', 'Dell Monitor')->update(['price' => 350]);

Output: 1 row updated

20. Delete products cheaper than 400

SQL: DELETE FROM products WHERE price < 400;

Laravel: Product::where('price', '<', 400)->delete();

Output: Dell Monitor deleted

21. Get customers without products

SQL: SELECT \* FROM customers WHERE NOT EXISTS (SELECT 1 FROM products WHERE products.customer\_id = customers.id);

Laravel: Customer::doesntHave('products')->get();

Output: If Alice is inserted with no products, she will appear here.

22. Get products ordered by price descending

SQL: SELECT \* FROM products ORDER BY price DESC;

Laravel: Product::orderBy('price', 'desc')->get();

Output: MacBook Pro, iPhone 15, Samsung TV, Dell Monitor

23. Get top 2 expensive products

SQL: SELECT \* FROM products ORDER BY price DESC LIMIT 2;

Laravel: Product::orderBy('price', 'desc')->limit(2)->get();

Output: MacBook Pro, iPhone 15

24. Find product by name 'iPhone 15'

SQL: SELECT \* FROM products WHERE name = 'iPhone 15';

Laravel: Product::where('name', 'iPhone 15')->first();

Output: iPhone 15 product details

25. Find customers whose name starts with 'J'

SQL: SELECT \* FROM customers WHERE name LIKE 'J%';

Laravel: Customer::where('name', 'like', 'J%')->get();

Output: John Doe, Jane Smith

26. Find products with price between 600 and 1000

SQL: SELECT \* FROM products WHERE price BETWEEN 600 AND 1000;

Laravel: Product::whereBetween('price', [600, 1000])->get();

Output: iPhone 15, Samsung TV

27. Get distinct product names

SQL: SELECT DISTINCT name FROM products;

Laravel: Product::select('name')->distinct()->get();

Output: iPhone 15, Samsung TV, MacBook Pro, Dell Monitor

28. Count customers with email not null

SQL: SELECT COUNT(\*) FROM customers WHERE email IS NOT NULL;

Laravel: Customer::whereNotNull('email')->count();

Output: 2

29. Find products without customer (customer\_id is null)

SQL: SELECT \* FROM products WHERE customer\_id IS NULL;

Laravel: Product::whereNull('customer\_id')->get();

Output: Dell Monitor

30. Find customers who have products

SQL: SELECT \* FROM customers WHERE EXISTS (SELECT 1 FROM products WHERE products.customer\_id = customers.id);

Laravel: Customer::has('products')->get();

Output: John Doe, Jane Smith

31. Paginate products, 2 per page (page 1)

SQL: SELECT \* FROM products LIMIT 2 OFFSET 0;

Laravel: Product::paginate(2);

Output: First two products (iPhone 15, Samsung TV)

32. Get products with names not like '%TV%'

SQL: SELECT \* FROM products WHERE name NOT LIKE '%TV%';

Laravel: Product::where('name', 'not like', '%TV%')->get();

Output: iPhone 15, MacBook Pro, Dell Monitor

33. Find max product price per customer

SQL: SELECT customer\_id, MAX(price) FROM products GROUP BY customer\_id;

Laravel: Product::select('customer\_id', DB::raw('MAX(price) as max\_price'))->groupBy('customer\_id')->get();

Output: customer 1 → 1800, customer 2 → 700, NULL → 300

34. Find customers and count of their products

SQL: SELECT customers.name, COUNT(products.id) FROM customers LEFT JOIN products ON customers.id = products.customer\_id GROUP BY customers.id;

Laravel: Customer::withCount('products')->get();

Output: John Doe → 2, Jane Smith → 1

35. Find average price of products per customer

SQL: SELECT customer\_id, AVG(price) FROM products GROUP BY customer\_id;

Laravel: Product::select('customer\_id', DB::raw('AVG(price) as avg\_price'))->groupBy('customer\_id')->get();

Output: customer 1 → 1399.5, customer 2 → 700, NULL → 300

36. Update customer name where email is 'john@example.com'

SQL: UPDATE customers SET name = 'John Updated' WHERE email = 'john@example.com';

Laravel: Customer::where('email', 'john@example.com')->update(['name' => 'John Updated']);

Output: 1 row updated

37. Delete customer where name is 'Alice'

SQL: DELETE FROM customers WHERE name = 'Alice';

Laravel: Customer::where('name', 'Alice')->delete();

Output: 1 row deleted

38. Get products created in last 7 days (assuming timestamps)

SQL: SELECT \* FROM products WHERE created\_at >= NOW() - INTERVAL 7 DAY;

Laravel: Product::where('created\_at', '>=', now()->subDays(7))->get();

Output: Depends on data, usually recent products

39. Count distinct customers who bought products

SQL: SELECT COUNT(DISTINCT customer\_id) FROM products;

Laravel: Product::distinct('customer\_id')->count('customer\_id');

Output: 2

40. Get first product ordered by name asc

SQL: SELECT \* FROM products ORDER BY name ASC LIMIT 1;

Laravel: Product::orderBy('name')->first();

Output: Dell Monitor

41. Find customers with email either 'john@example.com' or 'jane@example.com'

SQL: SELECT \* FROM customers WHERE email IN ('john@example.com', 'jane@example.com');

Laravel: Customer::whereIn('email', ['john@example.com', 'jane@example.com'])->get();

Output: John Doe, Jane Smith

42. Find products whose price is not in (300, 700)

SQL: SELECT \* FROM products WHERE price NOT IN (300, 700);

Laravel: Product::whereNotIn('price', [300, 700])->get();

Output: iPhone 15, MacBook Pro

43. Get products where name contains 'Pro'

SQL: SELECT \* FROM products WHERE name LIKE '%Pro%';

Laravel: Product::where('name', 'like', '%Pro%')->get();

Output: MacBook Pro

44. Get customers ordered by name descending

SQL: SELECT \* FROM customers ORDER BY name DESC;

Laravel: Customer::orderBy('name', 'desc')->get();

Output: Jane Smith, John Doe

45. Insert new product with default customer\_id NULL

SQL: INSERT INTO products (name, price) VALUES ('Google Pixel', 800);

Laravel: Product::create(['name' => 'Google Pixel', 'price' => 800]);

Output: New product added with customer\_id null

46. Count how many products have customer\_id null

SQL: SELECT COUNT(\*) FROM products WHERE customer\_id IS NULL;

Laravel: Product::whereNull('customer\_id')->count();

Output: Depends on data, example 1

47. Update product price by 10% where price < 1000

SQL: UPDATE products SET price = price \* 1.1 WHERE price < 1000;

Laravel: Product::where('price', '<', 1000)->update(['price' => DB::raw('price \* 1.1')]);

Output: Updated products with price increased

48. Select customers with no email (email IS NULL)

SQL: SELECT \* FROM customers WHERE email IS NULL;

Laravel: Customer::whereNull('email')->get();

Output: If exists

You said:

more

ChatGPT said:

Sure! Continuing with examples 49 to 100 — still with SQL + Laravel Eloquent + expected output based on sample data.

49-100 Examples (continued)

49. Find products with price between 500 and 1500

SQL:

SELECT \* FROM products WHERE price BETWEEN 500 AND 1500;

Laravel:

Product::whereBetween('price', [500, 1500])->get();

Output: iPhone 15 (999), Samsung TV (700), Google Pixel (800 if inserted)

50. Find customers whose name contains 'Smith'

SQL:

SELECT \* FROM customers WHERE name LIKE '%Smith%';

Laravel:

Customer::where('name', 'like', '%Smith%')->get();

Output: Jane Smith

51. Get latest 3 products by creation date

SQL:

SELECT \* FROM products ORDER BY created\_at DESC LIMIT 3;

Laravel:

Product::orderBy('created\_at', 'desc')->limit(3)->get();

Output: Latest 3 products (depends on data)

52. Get total price of all products for each customer

SQL:

SELECT customer\_id, SUM(price) as total\_price FROM products GROUP BY customer\_id;

Laravel:

Product::select('customer\_id', DB::raw('SUM(price) as total\_price'))->groupBy('customer\_id')->get();

Output: customer 1 → sum, customer 2 → sum, NULL → sum

53. Get products where name starts with 'M' or 'D'

SQL:

SELECT \* FROM products WHERE name LIKE 'M%' OR name LIKE 'D%';

Laravel:

Product::where('name', 'like', 'M%')->orWhere('name', 'like', 'D%')->get();

Output: MacBook Pro, Dell Monitor

54. Get products with price NOT between 300 and 800

SQL:

SELECT \* FROM products WHERE price NOT BETWEEN 300 AND 800;

Laravel:

Product::whereNotBetween('price', [300, 800])->get();

Output: MacBook Pro (1800), iPhone 15 (999 if > 800)

55. Find customers ordered by email ascending

SQL:

SELECT \* FROM customers ORDER BY email ASC;

Laravel:

Customer::orderBy('email', 'asc')->get();

Output: Jane Smith, John Doe (alphabetical by email)

56. Find products with NULL or empty customer\_id

SQL:

SELECT \* FROM products WHERE customer\_id IS NULL OR customer\_id = '';

Laravel:

Product::whereNull('customer\_id')->orWhere('customer\_id', '')->get();

Output: Dell Monitor (and any others if present)

57. Count customers grouped by email domain

SQL:

SELECT SUBSTRING\_INDEX(email, '@', -1) AS domain, COUNT(\*) FROM customers GROUP BY domain;

Laravel:

Customer::select(DB::raw("SUBSTRING\_INDEX(email, '@', -1) as domain"), DB::raw('COUNT(\*) as count'))->groupBy('domain')->get();

Output: example.com → 2

58. Find products with price multiple of 100

SQL:

SELECT \* FROM products WHERE price % 100 = 0;

Laravel:

Product::whereRaw('price % 100 = 0')->get();

Output: iPhone 15 (999 no), Samsung TV (700 yes), MacBook Pro (1800 yes), Dell Monitor (300 yes)

59. Find customers with email not like '%example.com'

SQL:

SELECT \* FROM customers WHERE email NOT LIKE '%example.com';

Laravel:

Customer::where('email', 'not like', '%example.com')->get();

Output: None in sample data

60. Update product price to NULL where price < 400

SQL:

UPDATE products SET price = NULL WHERE price < 400;

Laravel:

Product::where('price', '<', 400)->update(['price' => null]);

Output: Dell Monitor price null

61. Delete customers who have no products

SQL:

DELETE FROM customers WHERE id NOT IN (SELECT DISTINCT customer\_id FROM products);

Laravel:

Customer::doesntHave('products')->delete();

Output: Deletes customers without products (if any)

62. Find products with price in (300, 700, 999)

SQL:

SELECT \* FROM products WHERE price IN (300, 700, 999);

Laravel:

Product::whereIn('price', [300, 700, 999])->get();

Output: Dell Monitor, Samsung TV, iPhone 15

63. Find products with price not null and price > 500

SQL:

SELECT \* FROM products WHERE price IS NOT NULL AND price > 500;

Laravel:

Product::whereNotNull('price')->where('price', '>', 500)->get();

Output: iPhone 15, Samsung TV, MacBook Pro

64. Select customers and their product count ordered by product count desc

SQL:

SELECT customers.name, COUNT(products.id) as total\_products FROM customers LEFT JOIN products ON customers.id = products.customer\_id GROUP BY customers.id ORDER BY total\_products DESC;

Laravel:

Customer::withCount('products')->orderBy('products\_count', 'desc')->get();

Output: John Doe → 2, Jane Smith → 1

65. Find customers with email ending with '.com' or '.net'

SQL:

SELECT \* FROM customers WHERE email LIKE '%.com' OR email LIKE '%.net';

Laravel:

Customer::where('email', 'like', '%.com')->orWhere('email', 'like', '%.net')->get();

Output: John Doe, Jane Smith (both .com)

66. Select products with id between 1 and 3

SQL:

SELECT \* FROM products WHERE id BETWEEN 1 AND 3;

Laravel:

Product::whereBetween('id', [1, 3])->get();

Output: iPhone 15, Samsung TV, MacBook Pro

67. Get customer email by name 'John Doe'

SQL:

SELECT email FROM customers WHERE name = 'John Doe';

Laravel:

Customer::where('name', 'John Doe')->value('email');

Output: john@example.com

68. Get product names ordered alphabetically

SQL:

SELECT name FROM products ORDER BY name ASC;

Laravel:

Product::orderBy('name')->pluck('name');

Output: Dell Monitor, iPhone 15, MacBook Pro, Samsung TV

69. Insert multiple products in one query

SQL:

INSERT INTO products (name, price) VALUES ('AirPods', 199), ('Apple Watch', 399);

Laravel:

Product::insert([

['name' => 'AirPods', 'price' => 199],

['name' => 'Apple Watch', 'price' => 399]

]);

Output: Two new products inserted

70. Find products with price not equal to 700

SQL:

SELECT \* FROM products WHERE price != 700;

Laravel:

Product::where('price', '!=', 700)->get();

Output: iPhone 15, MacBook Pro, Dell Monitor

71. Select customers who bought iPhone 15

SQL:

SELECT customers.\* FROM customers JOIN products ON customers.id = products.customer\_id WHERE products.name = 'iPhone 15';

Laravel:

Customer::whereHas('products', fn($q) => $q->where('name', 'iPhone 15'))->get();

Output: John Doe

72. Select products without a customer (customer\_id IS NULL)

SQL:

SELECT \* FROM products WHERE customer\_id IS NULL;

Laravel:

Product::whereNull('customer\_id')->get();

Output: Dell Monitor

73. Count products per customer and filter customers having more than 1 product

SQL:

SELECT customer\_id, COUNT(\*) as product\_count FROM products GROUP BY customer\_id HAVING product\_count > 1;

Laravel:

Product::select('customer\_id', DB::raw('COUNT(\*) as product\_count'))->groupBy('customer\_id')->having('product\_count', '>', 1)->get();

Output: customer 1 → 2

74. Find products with price greater than average price

SQL:

SELECT \* FROM products WHERE price > (SELECT AVG(price) FROM products);

Laravel:

$avg = Product::avg('price');

Product::where('price', '>', $avg)->get();

Output: MacBook Pro (1800)

75. Get products with names in ('iPhone 15', 'MacBook Pro')

SQL:

SELECT \* FROM products WHERE name IN ('iPhone 15', 'MacBook Pro');

Laravel:

Product::whereIn('name', ['iPhone 15', 'MacBook Pro'])->get();

Output: iPhone 15, MacBook Pro

76. Update product price adding 100 where price less than 1000

SQL:

UPDATE products SET price = price + 100 WHERE price < 1000;

Laravel:

Product::where('price', '<', 1000)->update(['price' => DB::raw('price + 100')]);

Output: Samsung TV price 700→800, Dell Monitor 300→400

77. Delete customers with name starting with 'A'

SQL:

DELETE FROM customers WHERE name LIKE 'A%';

Laravel:

Customer::where('name', 'like', 'A%')->delete();

Output: Alice deleted (if exists)

78. Find customers with null or empty email

SQL:

SELECT \* FROM customers WHERE email IS NULL OR email = '';

Laravel:

Customer::whereNull('email')->orWhere('email', '')->get();

Output: None in sample

79. Get products ordered by price asc, skip first 2

SQL:

SELECT \* FROM products ORDER BY price ASC LIMIT 2 OFFSET 2;

Laravel:

Product::orderBy('price')->skip(2)->take(2)->get();

Output: MacBook Pro, iPhone 15 (if sorted ascending by price)

80. Count distinct product prices

SQL:

SELECT COUNT(DISTINCT price) FROM products;

Laravel:

Product::distinct('price')->count('price');

Output: 4

81. Find products created today (assumes created\_at column)

SQL:

SELECT \* FROM products WHERE DATE(created\_at) = CURDATE();

Laravel:

Product::whereDate('created\_at', now()->toDateString())->get();

Output: Depends on data

82. Find customers with email containing 'john'

SQL:

SELECT \* FROM customers WHERE email LIKE '%john%';

Laravel:

Customer::where('email', 'like', '%john%')->get();

Output: John Doe

83. Get products grouped by customer with total products and total price

SQL:

SELECT customer\_id, COUNT(\*) as product\_count, SUM(price) as total\_price FROM products GROUP BY customer\_id;

Laravel:

Product::select('customer\_id', DB::raw('COUNT(\*) as product\_count'), DB::raw('SUM(price) as total\_price'))->groupBy('customer\_id')->get();

Output: customer 1 → 2 products, 2799 price etc.

84. Get customer with the most expensive product

SQL:

SELECT customers.\* FROM customers JOIN products ON customers.id = products.customer\_id ORDER BY products.price DESC LIMIT 1;

Laravel:

$maxPrice = Product::max('price');

Customer::whereHas('products', fn($q) => $q->where('price', $maxPrice))->first();

Output: John Doe (MacBook Pro)

85. Get customers and their products with eager loading

SQL:

(No direct SQL, Laravel optimized)

Laravel:

Customer::with('products')->get();

Output: Customer and nested products

86. Find product with price closest to 1000 (less than or equal)

SQL:

SELECT \* FROM products WHERE price <= 1000 ORDER BY price DESC LIMIT 1;

Laravel:

Product::where('price', '<=', 1000)->orderBy('price', 'desc')->first();

Output: iPhone 15 (999)

87. Count customers with multiple emails (assuming email stored as CSV, unrealistic but example)

SQL:

SELECT COUNT(\*) FROM customers WHERE email LIKE '%,%';

Laravel:

Customer::where('email', 'like', '%,%')->count();

Output: 0 (sample data)

88. Get products excluding 'Samsung TV'

SQL:

SELECT \* FROM products WHERE name != 'Samsung TV';

Laravel:

Product::where('name', '!=', 'Samsung TV')->get();

Output: iPhone 15, MacBook Pro, Dell Monitor

89. Find products with price greater than 500 and less than 1500

SQL:

SELECT \* FROM products WHERE price > 500 AND price < 1500;

Laravel:

Product::where('price', '>', 500)->where('price', '<', 1500)->get();

Output: iPhone 15, Samsung TV, Google Pixel (if exists)

90. Get customers with email domain 'example.com'

SQL:

SELECT \* FROM customers WHERE email LIKE '%@example.com';

Laravel:

Customer::where('email', 'like', '%@example.com')->get();

Output: John Doe, Jane Smith

91. Select first 5 customers

SQL:

SELECT \* FROM customers LIMIT 5;

Laravel:

Customer::limit(5)->get();

Output: Both customers (only 2 exist)

92. Select last 3 products (order by id desc)

SQL:

SELECT \* FROM products ORDER BY id DESC LIMIT 3;

Laravel:

Product::orderBy('id', 'desc')->limit(3)->get();

Output: MacBook Pro, Samsung TV, Dell Monitor

93. Get products with prices not equal to 999 or 700

SQL:

SELECT \* FROM products WHERE price NOT IN (999, 700);

Laravel:

Product::whereNotIn('price', [999, 700])->get();

Output: MacBook Pro, Dell Monitor

94. Find customers who do not have an email

SQL:

SELECT \* FROM customers WHERE email IS NULL;

Laravel:

Customer::whereNull('email')->get();

Output: None in sample data

95. Find customers with names not equal 'John Doe'

SQL:

SELECT \* FROM customers WHERE name != 'John Doe';

Laravel:

Customer::where('name', '!=', 'John Doe')->get();

Output: Jane Smith

96. Find products with price divisible by 300

SQL:

SELECT \* FROM products WHERE price % 300 = 0;

Laravel:

Product::whereRaw('price % 300 = 0')->get();

Output: MacBook Pro (1800), Dell Monitor (300)

97. Find products with name length greater than 8

SQL:

SELECT \* FROM products WHERE LENGTH(name) > 8;

Laravel:

Product::whereRaw('LENGTH(name) > 8')->get();

Output: MacBook Pro, Samsung TV, Dell Monitor

98. Select customers with id less than 5 ordered by name

SQL:

SELECT \* FROM customers WHERE id < 5 ORDER BY name ASC;

Laravel:

Customer::where('id', '<', 5)->orderBy('name')->get();

Output: John Doe, Jane Smith

99. Update product name to lowercase

SQL:

UPDATE products SET name = LOWER(name);

Laravel:

Product::update(['name' => DB::raw('LOWER(name)')]);

Output: Product names updated to lowercase

100. Select all products and customers as JSON with relationship

SQL: (Not SQL but Laravel)

Laravel:

return Customer::with('products')->get()->toJson();

Output: JSON array of customers and nested products