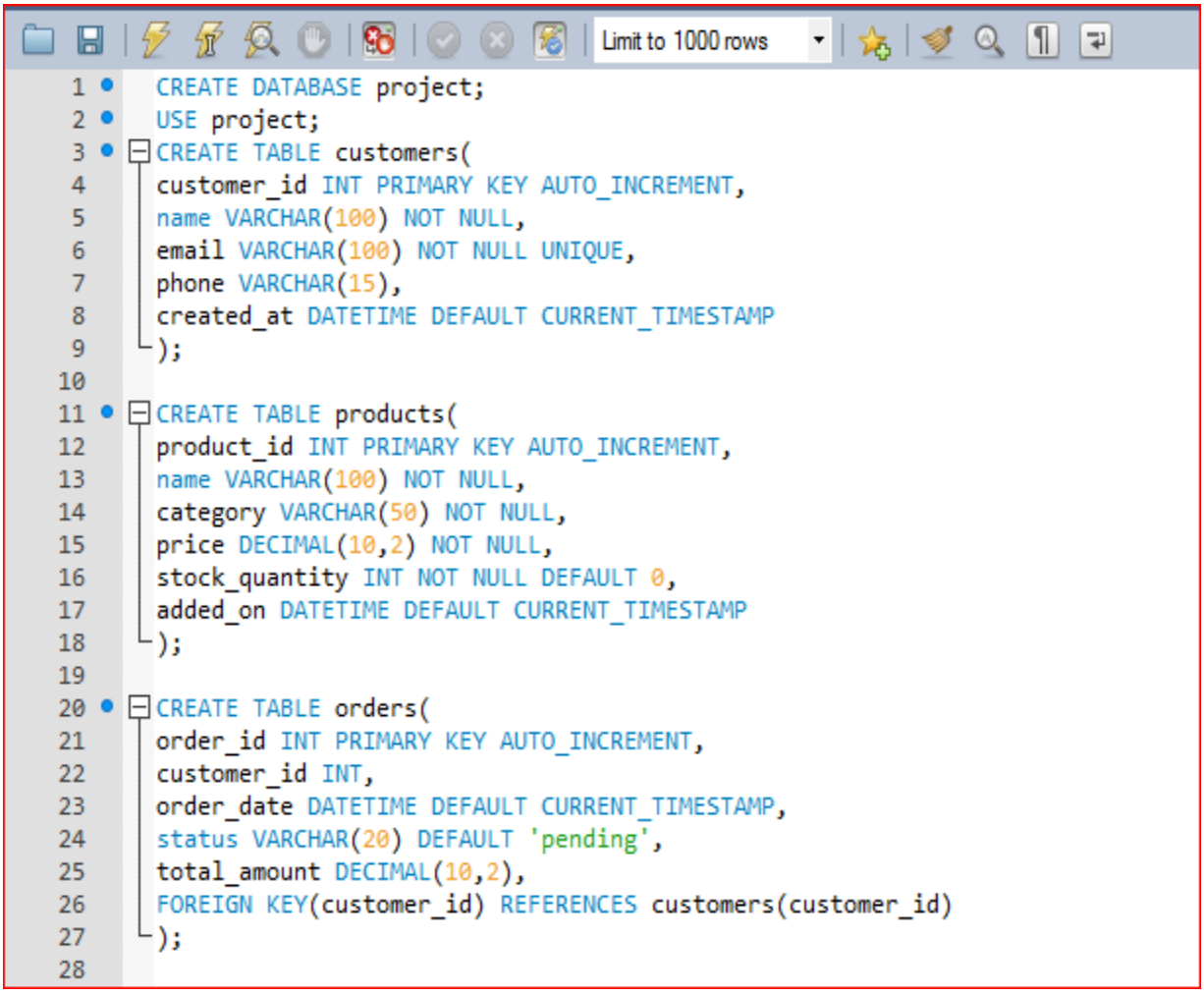
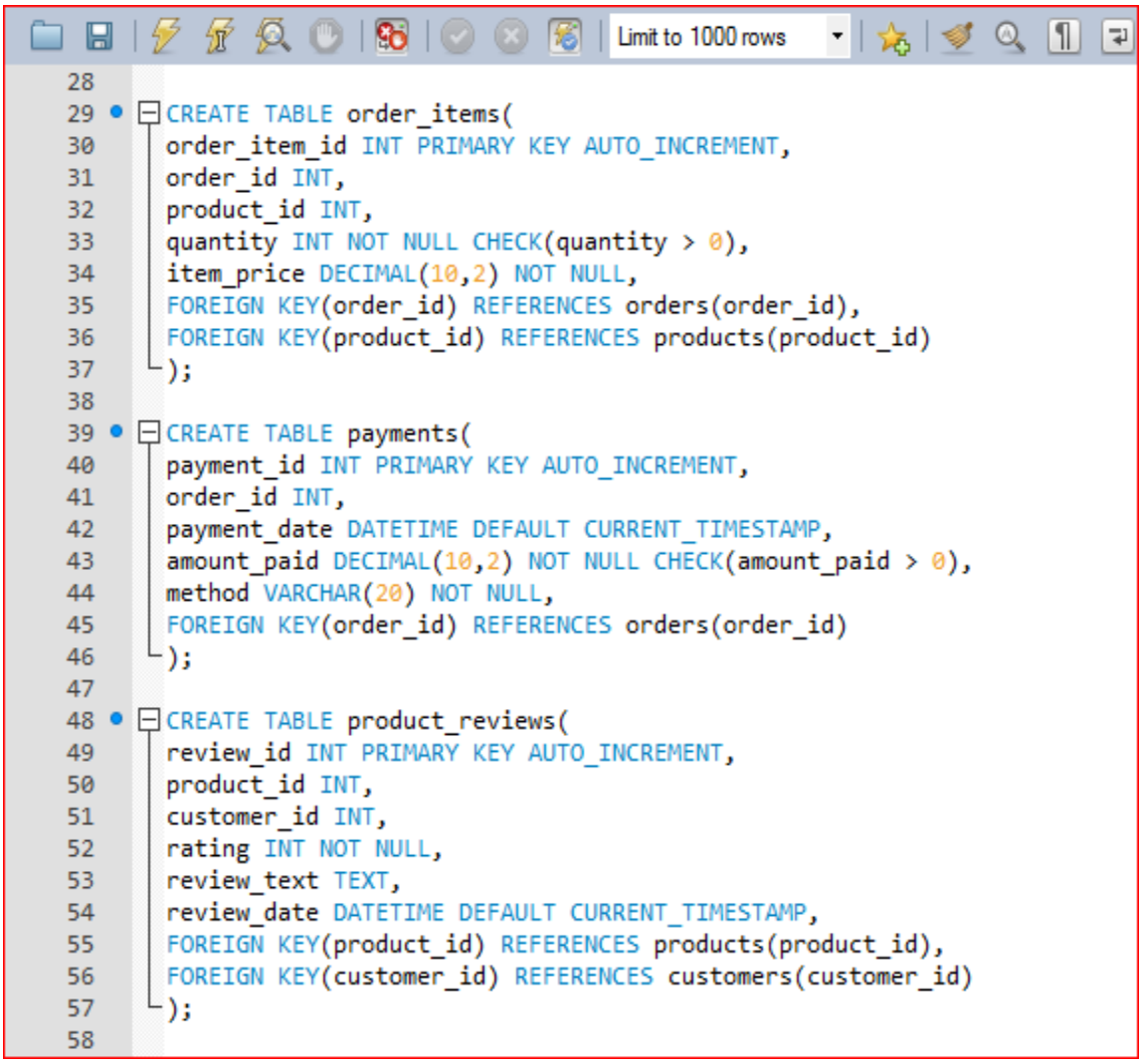
SQL PROJECT

Creating a database and defining database tables (DDLs)

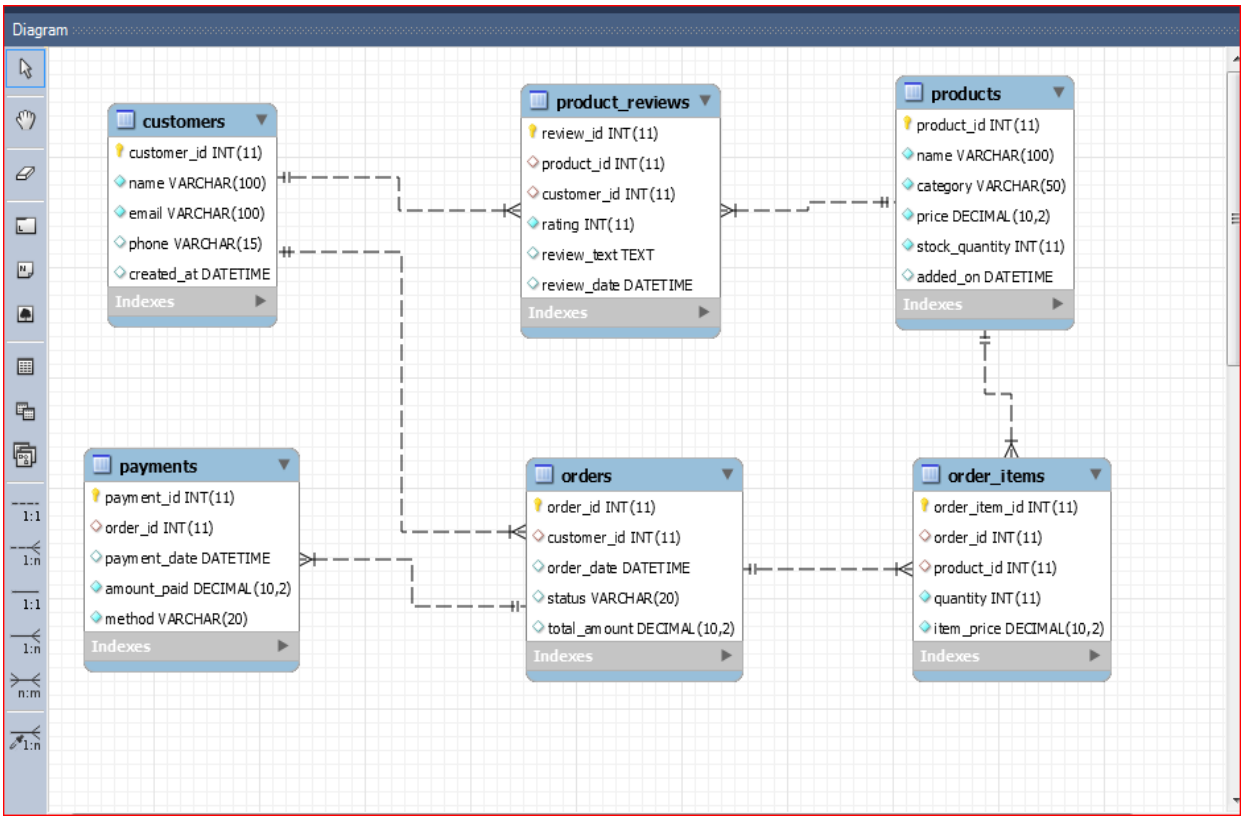


```sql
CREATE DATABASE project;
USE project;
CREATE TABLE customers(
customer_id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL UNIQUE,
phone VARCHAR(15),
created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE products(
product_id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
category VARCHAR(50) NOT NULL,
price DECIMAL(10,2) NOT NULL,
stock_quantity INT NOT NULL DEFAULT 0,
added_on DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE orders(
order_id INT PRIMARY KEY AUTO_INCREMENT,
customer_id INT,
order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
status VARCHAR(20) DEFAULT 'pending',
total_amount DECIMAL(10,2),
FOREIGN KEY(customer_id) REFERENCES customers(customer_id)
);

```




```
28
29  CREATE TABLE order_items(
30  order_item_id INT PRIMARY KEY AUTO_INCREMENT,
31  order_id INT,
32  product_id INT,
33  quantity INT NOT NULL CHECK(quantity > 0),
34  item_price DECIMAL(10,2) NOT NULL,
35  FOREIGN KEY(order_id) REFERENCES orders(order_id),
36  FOREIGN KEY(product_id) REFERENCES products(product_id)
37  );
38
39  CREATE TABLE payments(
40  payment_id INT PRIMARY KEY AUTO_INCREMENT,
41  order_id INT,
42  payment_date DATETIME DEFAULT CURRENT_TIMESTAMP,
43  amount_paid DECIMAL(10,2) NOT NULL CHECK(amount_paid > 0),
44  method VARCHAR(20) NOT NULL,
45  FOREIGN KEY(order_id) REFERENCES orders(order_id)
46  );
47
48  CREATE TABLE product_reviews(
49  review_id INT PRIMARY KEY AUTO_INCREMENT,
50  product_id INT,
51  customer_id INT,
52  rating INT NOT NULL,
53  review_text TEXT,
54  review_date DATETIME DEFAULT CURRENT_TIMESTAMP,
55  FOREIGN KEY(product_id) REFERENCES products(product_id),
56  FOREIGN KEY(customer_id) REFERENCES customers(customer_id)
57  );
58
```

EER Model Diagram



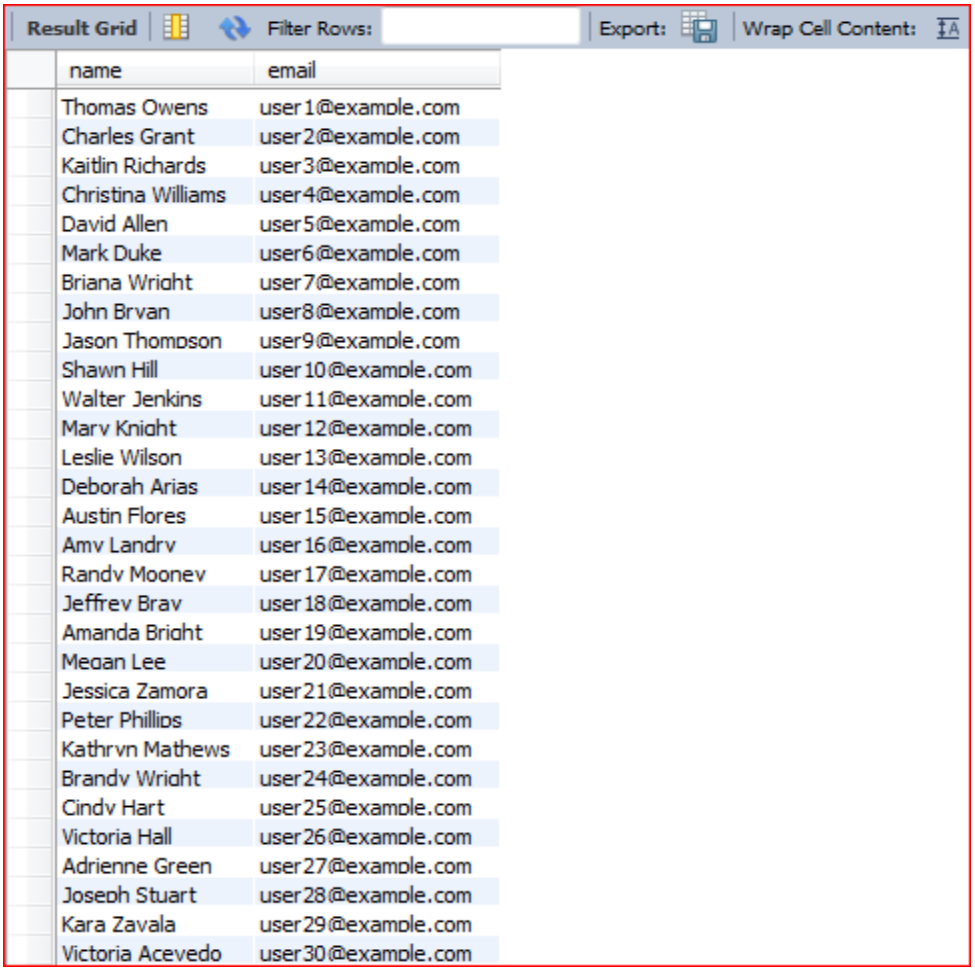
Level 1: Basics

1. Retrieve customer names and emails for email marketing

Query -

```
SELECT name, email

FROM customers;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

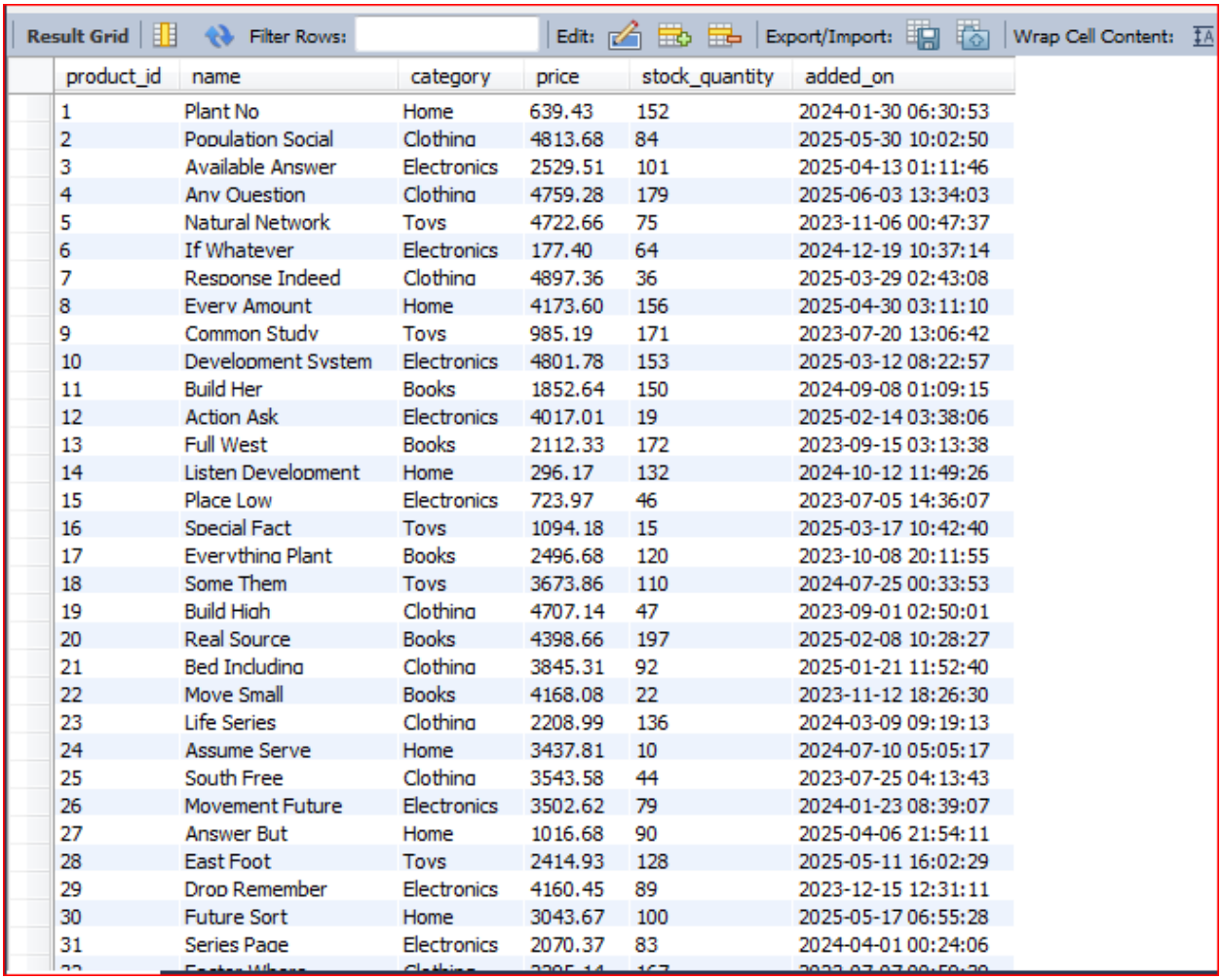
name	email
Thomas Owens	user1@example.com
Charles Grant	user2@example.com
Kaitlin Richards	user3@example.com
Christina Williams	user4@example.com
David Allen	user5@example.com
Mark Duke	user6@example.com
Briana Wright	user7@example.com
John Bryan	user8@example.com
Jason Thompson	user9@example.com
Shawn Hill	user10@example.com
Walter Jenkins	user11@example.com
Mary Knight	user12@example.com
Leslie Wilson	user13@example.com
Deborah Arias	user14@example.com
Austin Flores	user15@example.com
Amy Landry	user16@example.com
Randy Mooney	user17@example.com
Jeffrey Bray	user18@example.com
Amanda Bright	user19@example.com
Megan Lee	user20@example.com
Jessica Zamora	user21@example.com
Peter Phillips	user22@example.com
Kathryn Mathews	user23@example.com
Brandy Wright	user24@example.com
Cindy Hart	user25@example.com
Victoria Hall	user26@example.com
Adrienne Green	user27@example.com
Joseph Stuart	user28@example.com
Kara Zavala	user29@example.com
Victoria Acevedo	user30@example.com

2 - View complete product catalog with all available details.

Query -

SELECT * FROM products;

Output -



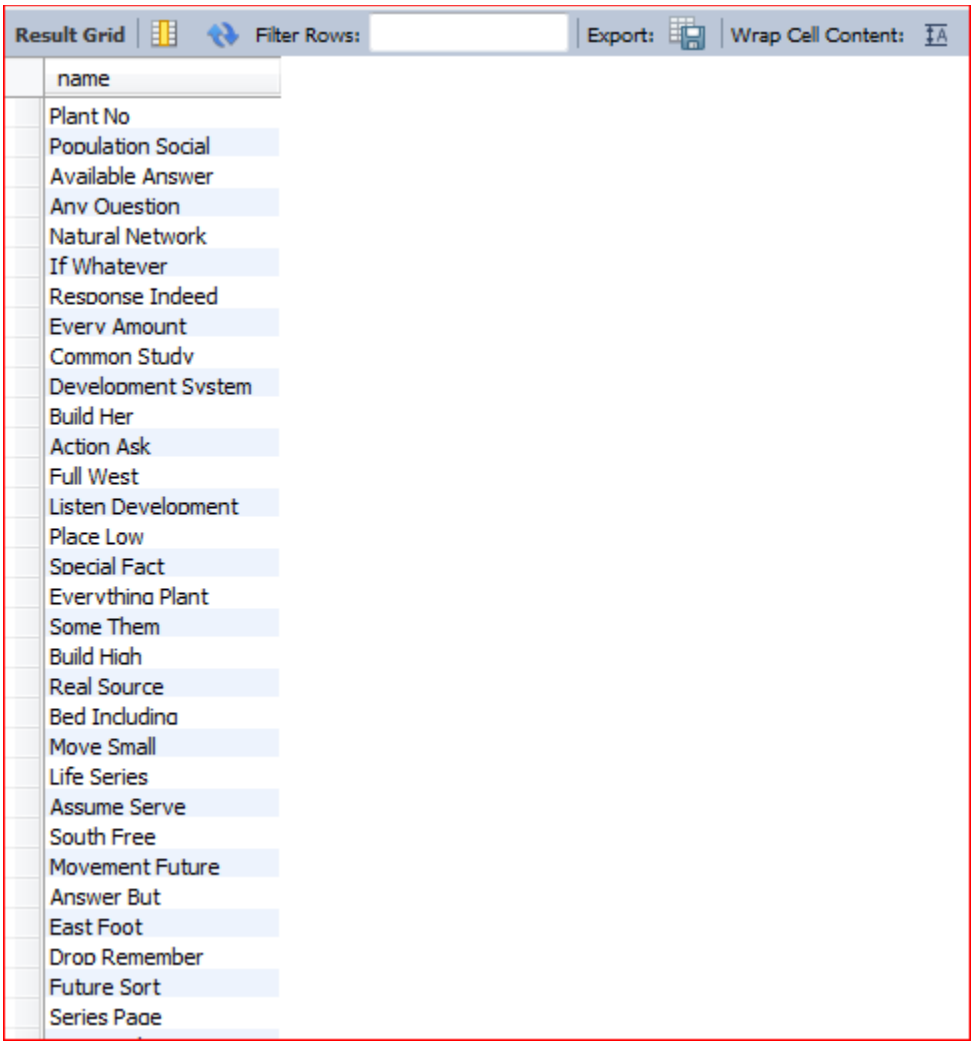
product_id	name	category	price	stock_quantity	added_on
1	Plant No	Home	639.43	152	2024-01-30 06:30:53
2	Population Social	Clothing	4813.68	84	2025-05-30 10:02:50
3	Available Answer	Electronics	2529.51	101	2025-04-13 01:11:46
4	Any Question	Clothing	4759.28	179	2025-06-03 13:34:03
5	Natural Network	Toys	4722.66	75	2023-11-06 00:47:37
6	If Whatever	Electronics	177.40	64	2024-12-19 10:37:14
7	Response Indeed	Clothing	4897.36	36	2025-03-29 02:43:08
8	Every Amount	Home	4173.60	156	2025-04-30 03:11:10
9	Common Study	Toys	985.19	171	2023-07-20 13:06:42
10	Development System	Electronics	4801.78	153	2025-03-12 08:22:57
11	Build Her	Books	1852.64	150	2024-09-08 01:09:15
12	Action Ask	Electronics	4017.01	19	2025-02-14 03:38:06
13	Full West	Books	2112.33	172	2023-09-15 03:13:38
14	Listen Development	Home	296.17	132	2024-10-12 11:49:26
15	Place Low	Electronics	723.97	46	2023-07-05 14:36:07
16	Special Fact	Toys	1094.18	15	2025-03-17 10:42:40
17	Everything Plant	Books	2496.68	120	2023-10-08 20:11:55
18	Some Them	Toys	3673.86	110	2024-07-25 00:33:53
19	Build High	Clothing	4707.14	47	2023-09-01 02:50:01
20	Real Source	Books	4398.66	197	2025-02-08 10:28:27
21	Bed Including	Clothing	3845.31	92	2025-01-21 11:52:40
22	Move Small	Books	4168.08	22	2023-11-12 18:26:30
23	Life Series	Clothing	2208.99	136	2024-03-09 09:19:13
24	Assume Serve	Home	3437.81	10	2024-07-10 05:05:17
25	South Free	Clothing	3543.58	44	2023-07-25 04:13:43
26	Movement Future	Electronics	3502.62	79	2024-01-23 08:39:07
27	Answer But	Home	1016.68	90	2025-04-06 21:54:11
28	East Foot	Toys	2414.93	128	2025-05-11 16:02:29
29	Drop Remember	Electronics	4160.45	89	2023-12-15 12:31:11
30	Future Sort	Home	3043.67	100	2025-05-17 06:55:28
31	Series Page	Electronics	2070.37	83	2024-04-01 00:24:06

3. List all unique product categories.

Query -

SELECT DISTINCT name FROM products;

Output -

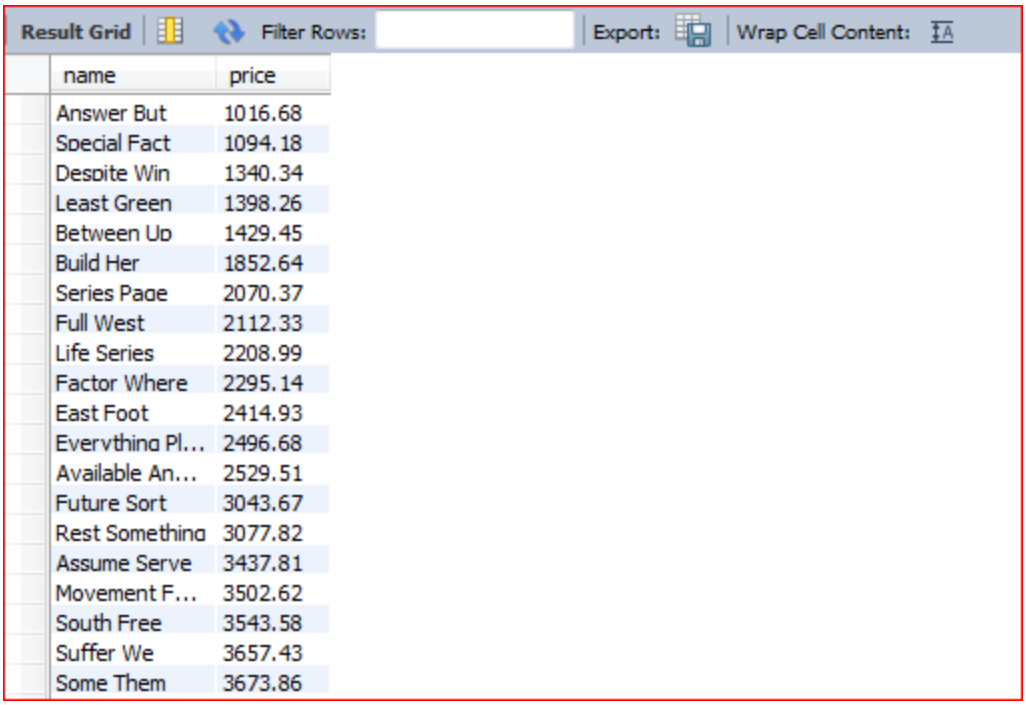

Result Grid
Filter Rows:
Export:
Wrap Cell Content:
name
Plant No
Population Social
Available Answer
Any Question
Natural Network
If Whatever
Response Indeed
Every Amount
Common Study
Development System
Build Her
Action Ask
Full West
Listen Development
Place Low
Special Fact
Everything Plant
Some Them
Build High
Real Source
Bed Including
Move Small
Life Series
Assume Serve
South Free
Movement Future
Answer But
East Foot
Drop Remember
Future Sort
Series Page

4. *Show all products priced above ₹1,000*.

Query -

```
SELECT name, price

FROM products

WHERE price >= 1000

ORDER BY price;
```

Output -



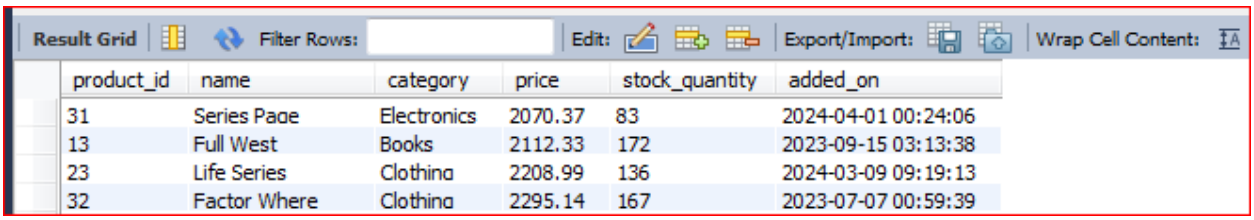
name	price
Answer But	1016.68
Special Fact	1094.18
Despite Win	1340.34
Least Green	1398.26
Between Up	1429.45
Build Her	1852.64
Series Page	2070.37
Full West	2112.33
Life Series	2208.99
Factor Where	2295.14
East Foot	2414.93
Everything Pl...	2496.68
Available An...	2529.51
Future Sort	3043.67
Rest Something	3077.82
Assume Serve	3437.81
Movement F...	3502.62
South Free	3543.58
Suffer We	3657.43
Some Them	3673.86

5. Display products within a mid-range price bracket (₹2,000 to ₹5,000).

Query -

```
select * from products
where price >= 2000 and price <= 5000
order by price ASC;
```

Output -



product_id	name	category	price	stock_quantity	added_on
31	Series Page	Electronics	2070.37	83	2024-04-01 00:24:06
13	Full West	Books	2112.33	172	2023-09-15 03:13:38
23	Life Series	Clothing	2208.99	136	2024-03-09 09:19:13
32	Factor Where	Clothing	2295.14	167	2023-07-07 00:59:39

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	stock_quantity	added_on
28	East Foot	Toys	2414.93	128	2025-05-11 16:02:29
17	Everything Plant	Books	2496.68	120	2023-10-08 20:11:55
3	Available Answer	Electronics	2529.51	101	2025-04-13 01:11:46
30	Future Sort	Home	3043.67	100	2025-05-17 06:55:28
49	Rest Something	Books	3077.82	181	2023-08-09 03:42:42
24	Assume Serve	Home	3437.81	10	2024-07-10 05:05:17
26	Movement Future	Electronics	3502.62	79	2024-01-23 08:39:07
25	South Free	Clothing	3543.58	44	2023-07-25 04:13:43
45	Suffer We	Electronics	3657.43	113	2024-07-25 12:40:57
18	Some Them	Toys	3673.86	110	2024-07-25 00:33:53
46	Age Treatment	Home	3728.83	160	2023-12-01 10:51:52
21	Bed Including	Clothing	3845.31	92	2025-01-21 11:52:40
42	Weight Enter	Clothing	3875.18	60	2024-01-26 08:21:55
12	Action Ask	Electronics	4017.01	19	2025-02-14 03:38:06
39	Data Add	Books	4063.38	62	2025-05-26 23:39:16
29	Drop Remember	Electronics	4160.45	89	2023-12-15 12:31:11
22	Move Small	Books	4168.08	22	2023-11-12 18:26:30
8	Every Amount	Home	4173.60	156	2025-04-30 03:11:10
48	Group But	Clothing	4180.23	141	2023-11-29 01:38:12
33	Involve Officer	Clothing	4244.09	112	2024-03-22 11:36:17
20	Real Source	Books	4398.66	197	2025-02-08 10:28:27
38	Study Total	Toys	4413.68	31	2023-11-08 08:25:07
41	Serious Recognize	Electronics	4523.10	112	2023-08-20 06:16:55
19	Build High	Clothing	4707.14	47	2023-09-01 02:50:01
5	Natural Network	Toys	4722.66	75	2023-11-06 00:47:37
35	Fire Often	Electronics	4734.89	198	2023-10-22 04:15:51
4	Any Question	Clothing	4759.28	179	2025-06-03 13:34:03
10	Development Sy...	Electronics	4801.78	153	2025-03-12 08:22:57
2	Population Social	Clothing	4813.68	84	2025-05-30 10:02:50
7	Response Indeed	Clothing	4897.36	36	2025-03-29 02:43:08
NULL	NULL	NULL	NULL	NULL	NULL

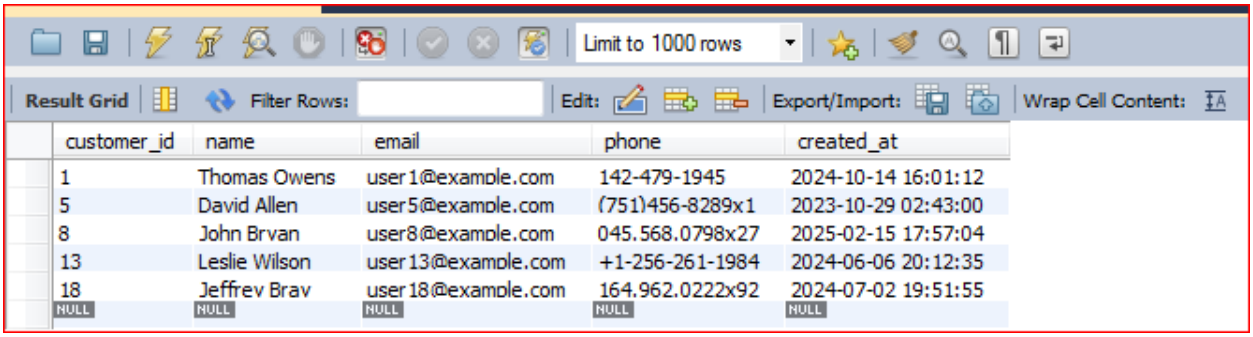
6. Fetch data for specific customer IDs (e.g., from loyalty program list).

Query -

--Assuming we have loyalty list with customer_id (1,5,8,13,18).

```
select * from customers
where customer_id in (1,5,8,13,18);
```

Output -



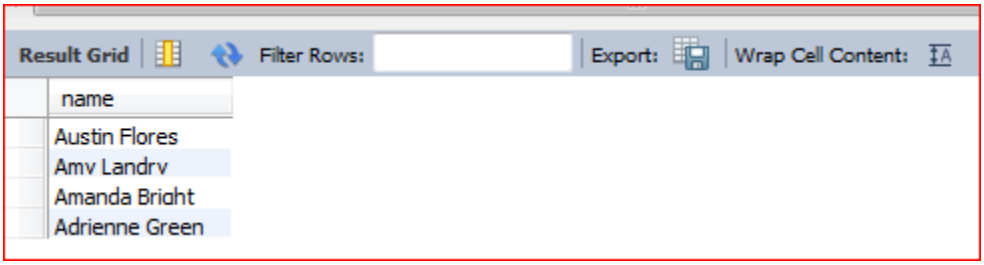
customer_id	name	email	phone	created_at
1	Thomas Owens	user1@example.com	142-479-1945	2024-10-14 16:01:12
5	David Allen	user5@example.com	(751)456-8289x1	2023-10-29 02:43:00
8	John Bryan	user8@example.com	045.568.0798x27	2025-02-15 17:57:04
13	Leslie Wilson	user13@example.com	+1-256-261-1984	2024-06-06 20:12:35
18	Jeffrey Bray	user18@example.com	164.962.0222x92	2024-07-02 19:51:55
NULL	NULL	NULL	NULL	NULL

7. **Identify customers whose names start with the letter 'A',**

Query -

```
select name from customers

where name like 'A%' ;
```

Output -



name
Austin Flores
Amy Landry
Amanda Bright
Adrienne Green

8. List electronics products priced under ₹3,000.

Query -

```
SELECT  * FROM products

WHERE category = 'Electronics' and price < 3000;
```

Output -

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	stock_quantity	added_on
3	Available Answer	Electronics	2529.51	101	2025-04-13 01:11:46
6	If Whatever	Electronics	177.40	64	2024-12-19 10:37:14
15	Place Low	Electronics	723.97	46	2023-07-05 14:36:07
31	Series Page	Electronics	2070.37	83	2024-04-01 00:24:06
34	Despite Win	Electronics	1340.34	64	2024-11-27 06:55:45
44	Actually Term	Electronics	396.11	85	2023-11-02 13:09:20
47	Southern Thing	Electronics	512.46	40	2024-02-28 17:57:38
NULL	NULL	NULL	NULL	NULL	NULL

9. Display product names and prices in descending order of price.

Query -

```
SELECT name,price

FROM products

ORDER BY price DESC;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	price
Response Indeed	4897.36
Population Social	4813.68
Development System	4801.78
Any Question	4759.28
Fire Often	4734.89
Natural Network	4722.66
Build High	4707.14
Serious Recognize	4523.10

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

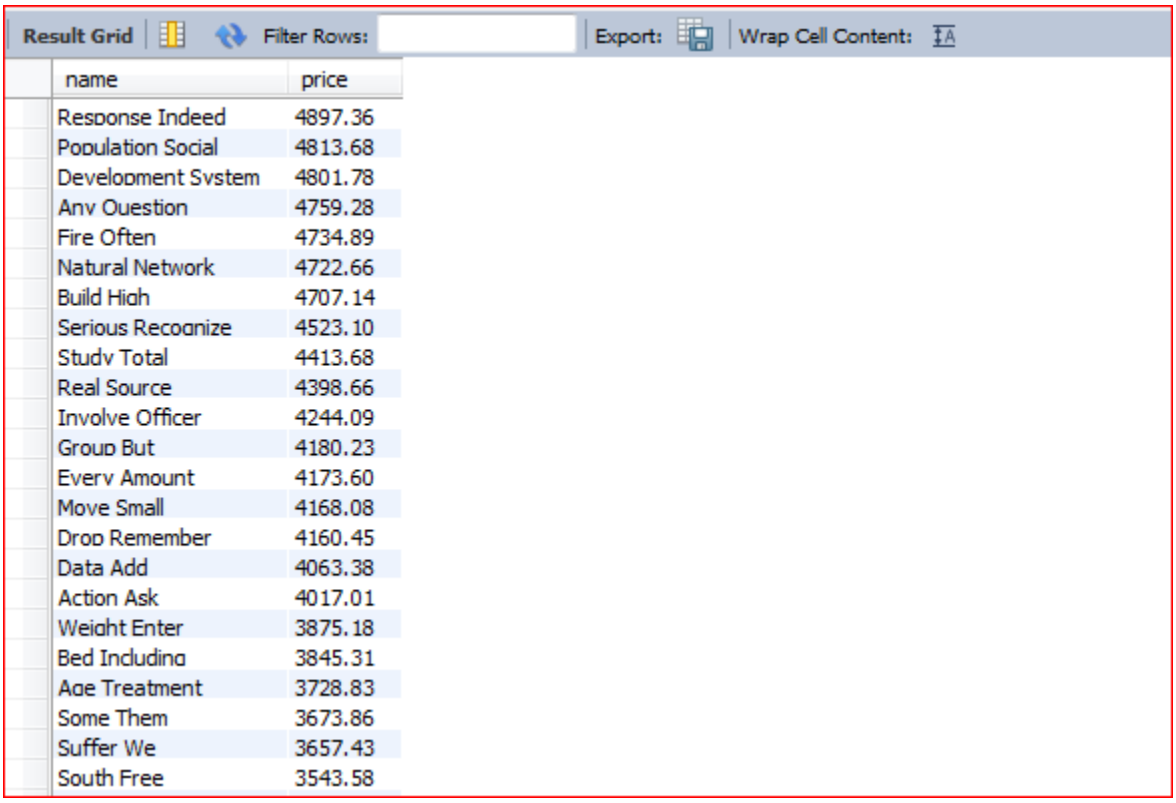
name	price
Between Up	1429.45
Least Green	1398.26
Despite Win	1340.34
Special Fact	1094.18
Answer But	1016.68
Common Study	985.19
Toward Minute	857.85
Stock Article	834.75
Place Low	723.97
Plant No	639.43
Southern Thing	512.46
Television Stock	421.73
Actually Term	396.11
Listen Development	296.17
If Whatever	177.40

10. Display product names and prices, sorted by price and then by name.

Query -

```
SELECT name,price

FROM  products

ORDER BY price DESC, name;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	price
Response Indeed	4897.36
Population Social	4813.68
Development System	4801.78
Any Question	4759.28
Fire Often	4734.89
Natural Network	4722.66
Build High	4707.14
Serious Recognize	4523.10
Study Total	4413.68
Real Source	4398.66
Involve Officer	4244.09
Group But	4180.23
Every Amount	4173.60
Move Small	4168.08
Drop Remember	4160.45
Data Add	4063.38
Action Ask	4017.01
Weight Enter	3875.18
Bed Including	3845.31
Age Treatment	3728.83
Some Them	3673.86
Suffer We	3657.43
South Free	3543.58

Level 2: Filtering and Formatting

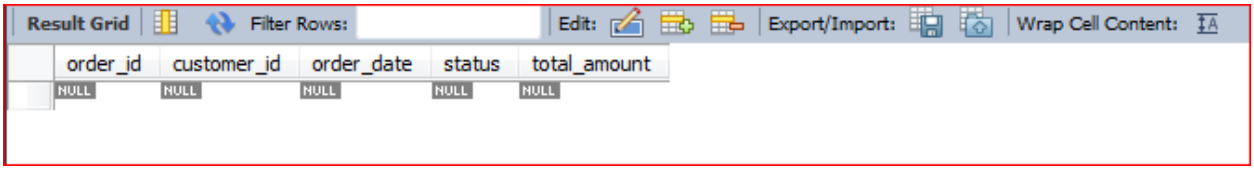
1. Retrieve orders where customer information is missing (possibly due to data migration or deletion).

Query -

SELECT * FROM orders

WHERE customer_id IS NULL;

Output -



** no data found in orders table where customers id column has null values.

2. Display customer names and emails using column aliases for front end readability.

Query -

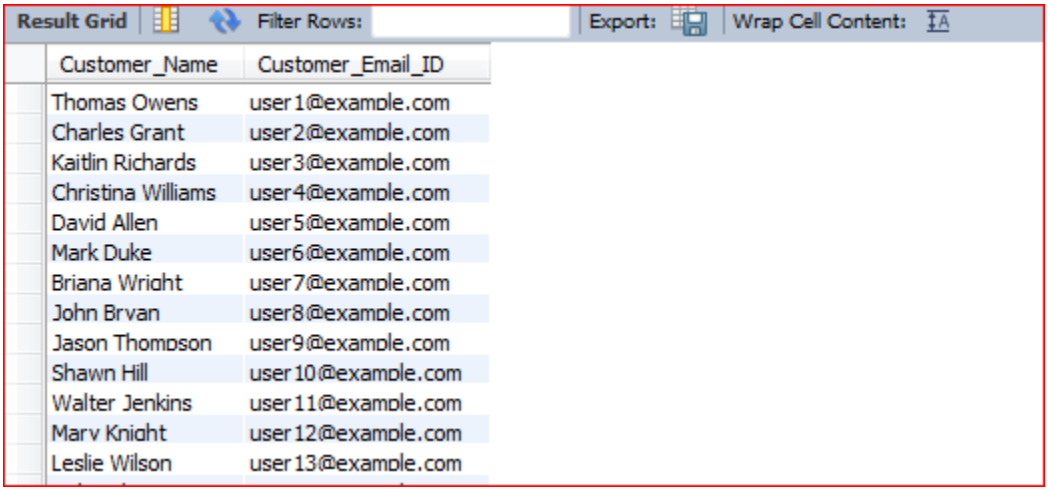
SELECT

name AS Customer_Name,

email AS Customer_Email_ID

FROM customers;

Output -



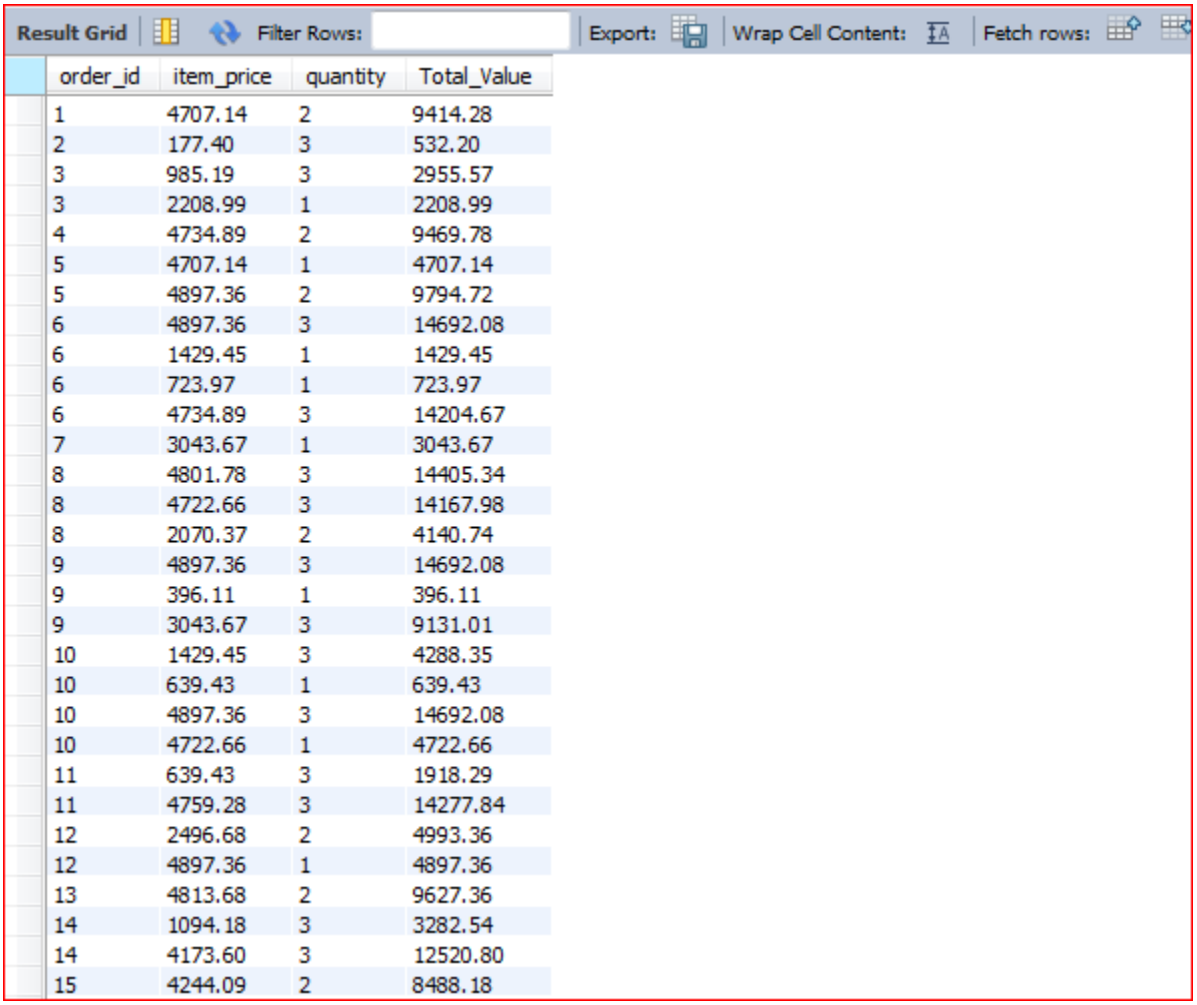
3. Calculate total value per item ordered by multiplying quantity and item price.

Query -

SELECT

```
    order_id,
    item_price,
    quantity,
    (item_price * quantity) AS Total_Value
FROM order_items;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

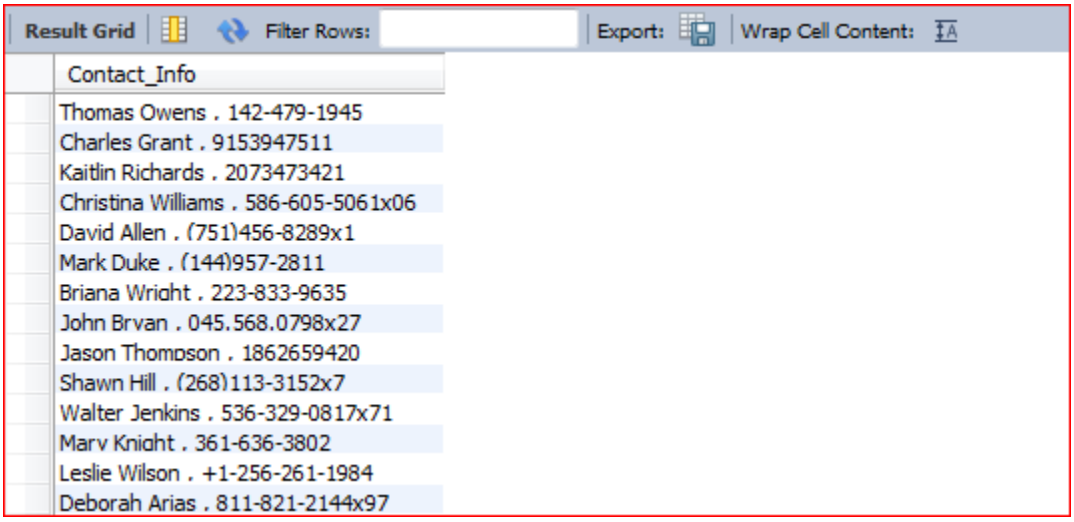
order_id	item_price	quantity	Total_Value
1	4707.14	2	9414.28
2	177.40	3	532.20
3	985.19	3	2955.57
3	2208.99	1	2208.99
4	4734.89	2	9469.78
5	4707.14	1	4707.14
5	4897.36	2	9794.72
6	4897.36	3	14692.08
6	1429.45	1	1429.45
6	723.97	1	723.97
6	4734.89	3	14204.67
7	3043.67	1	3043.67
8	4801.78	3	14405.34
8	4722.66	3	14167.98
8	2070.37	2	4140.74
9	4897.36	3	14692.08
9	396.11	1	396.11
9	3043.67	3	9131.01
10	1429.45	3	4288.35
10	639.43	1	639.43
10	4897.36	3	14692.08
10	4722.66	1	4722.66
11	639.43	3	1918.29
11	4759.28	3	14277.84
12	2496.68	2	4993.36
12	4897.36	1	4897.36
13	4813.68	2	9627.36
14	1094.18	3	3282.54
14	4173.60	3	12520.80
15	4244.09	2	8488.18

4. Combine customer name and phone number in a single column.

Query -

```
SELECT CONCAT(name, ' , ', phone) AS Contact_Info
FROM customers;
```

Output -

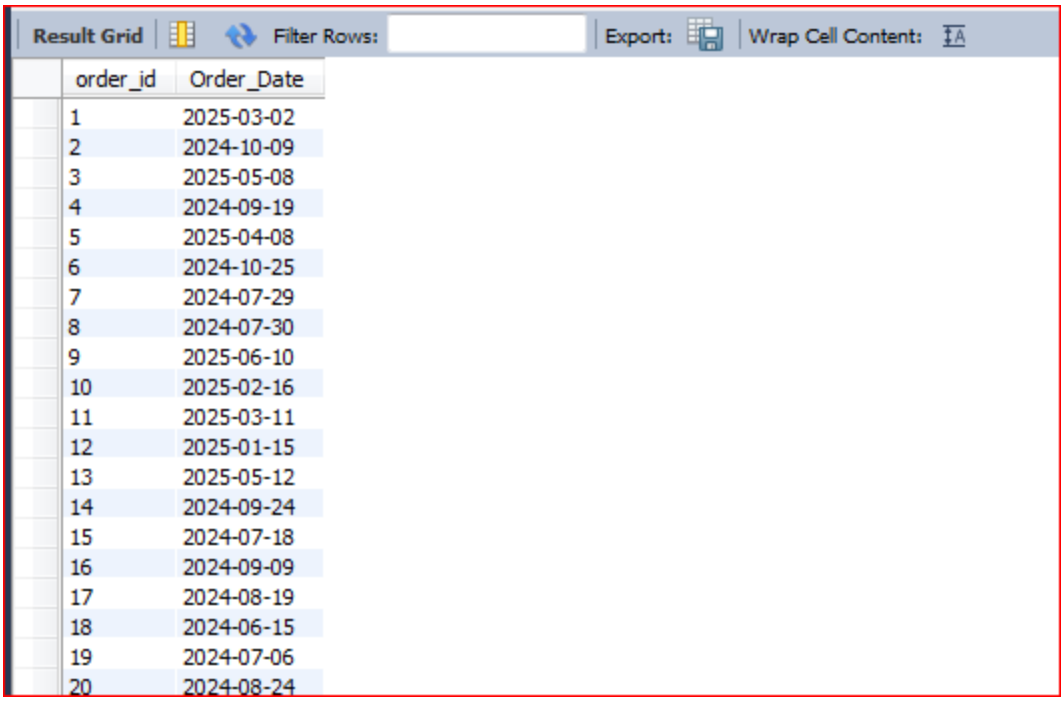


5. Extract only the date part from order timestamps for date-wise reporting.

Query -

```
SELECT
        order_id,
        DATE(order_date) AS Order_Date
FROM orders;
```

Output -



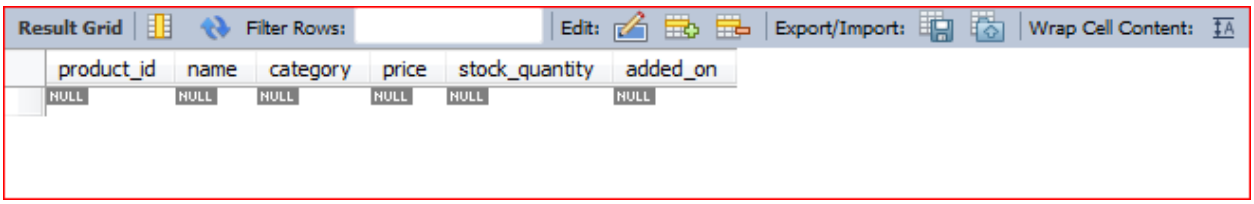
order_id	Order_Date
1	2025-03-02
2	2024-10-09
3	2025-05-08
4	2024-09-19
5	2025-04-08
6	2024-10-25
7	2024-07-29
8	2024-07-30
9	2025-06-10
10	2025-02-16
11	2025-03-11
12	2025-01-15
13	2025-05-12
14	2024-09-24
15	2024-07-18
16	2024-09-09
17	2024-08-19
18	2024-06-15
19	2024-07-06
20	2024-08-24

6. List products that do not have any stock left.

Query -

```
SELECT * FROM products

WHERE stock_quantity = 0;
```

Output -



product_id	name	category	price	stock_quantity	added_on
NULL	NULL	NULL	NULL	NULL	NULL

** No products found which do not have any stocks left.

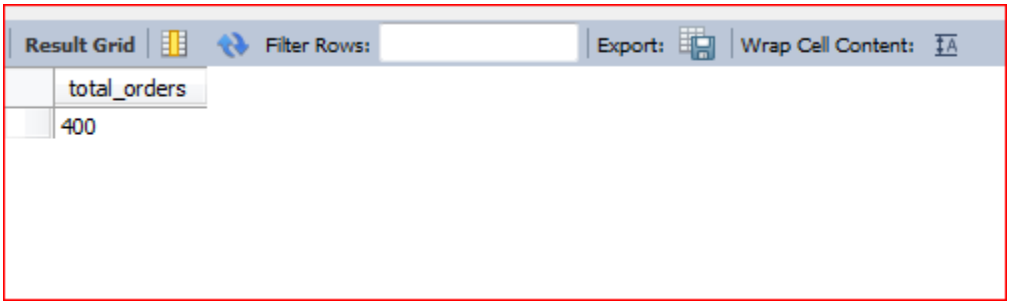
Level 3: Aggregations

1. Count the total number of orders placed.

Query -

```
SELECT COUNT(order_id) AS total_orders
FROM orders;
```

Output -

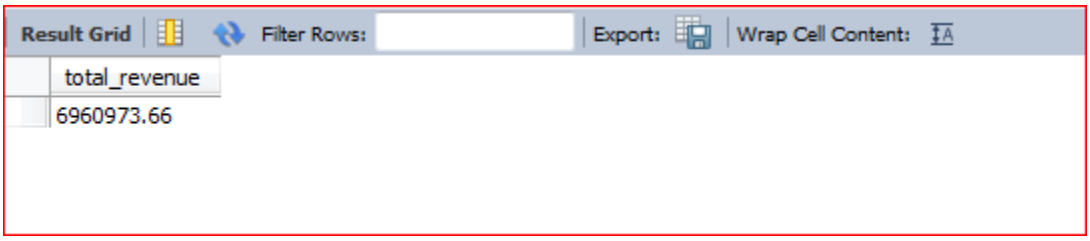


2. Calculate the total revenue collected from all orders.

Query -

```
SELECT SUM(total_amount) AS total_revenue
FROM orders;
```

Output -

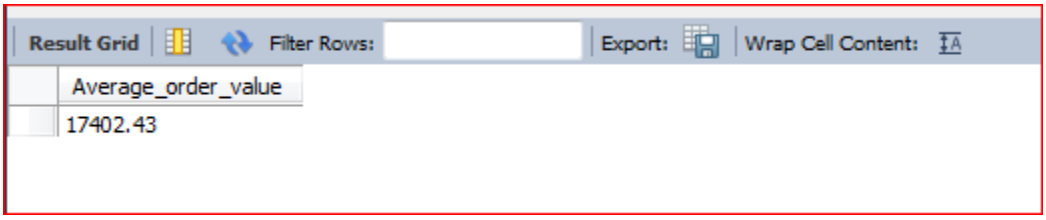


3. Calculate the average order value.

Query -

```
SELECT ROUND(AVG(total_amount),2) AS Average_order_value
FROM orders;
```

Output -

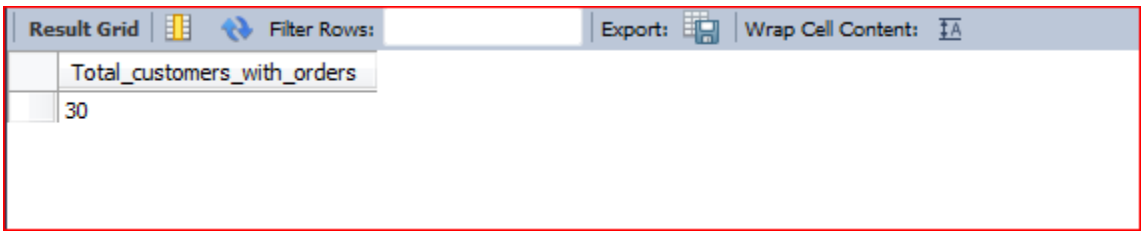


4. Count the number of customers who have placed at least one order.

Query -

```
SELECT COUNT(DISTINCT customer_id) AS Total_customers_with_orders
FROM orders;
```

Output -

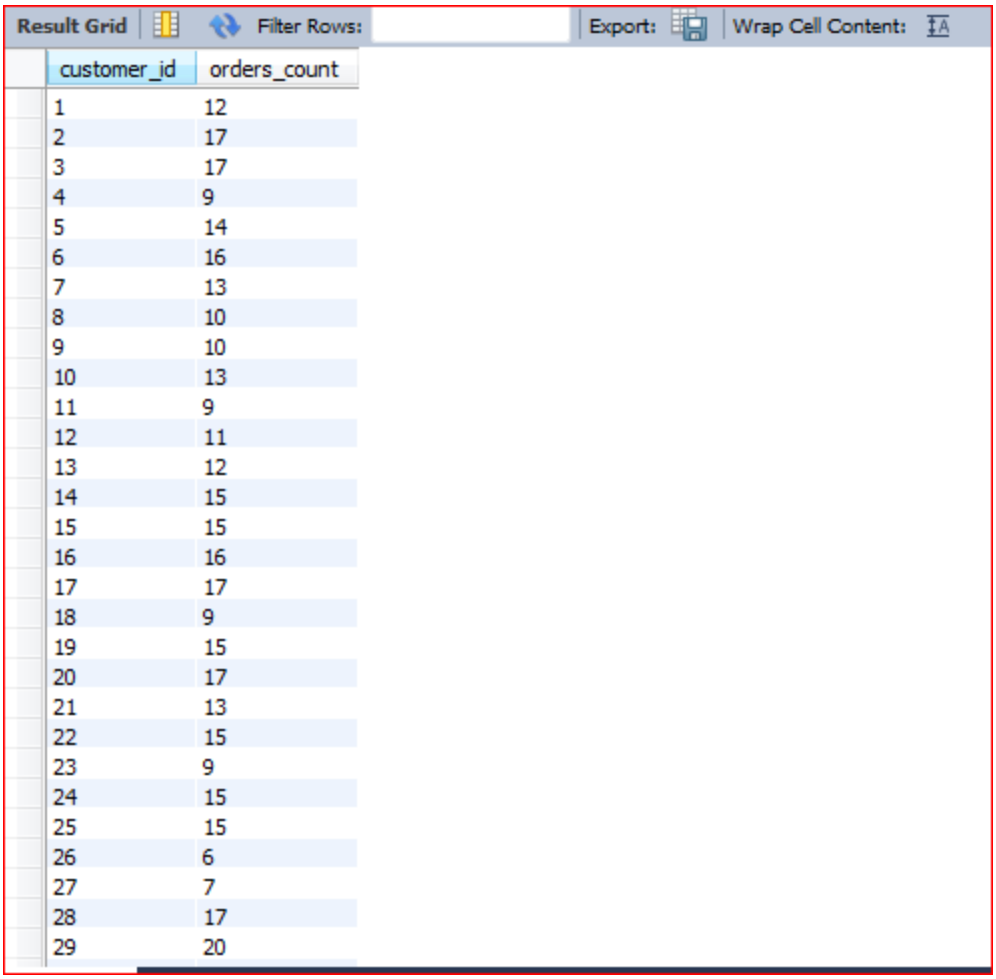


5. Find the number of orders placed by each customer.

Query -

```
SELECT
        customer_id,
        COUNT(*) AS orders_count
FROM orders
GROUP BY customer_id
ORDER BY customer_id;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

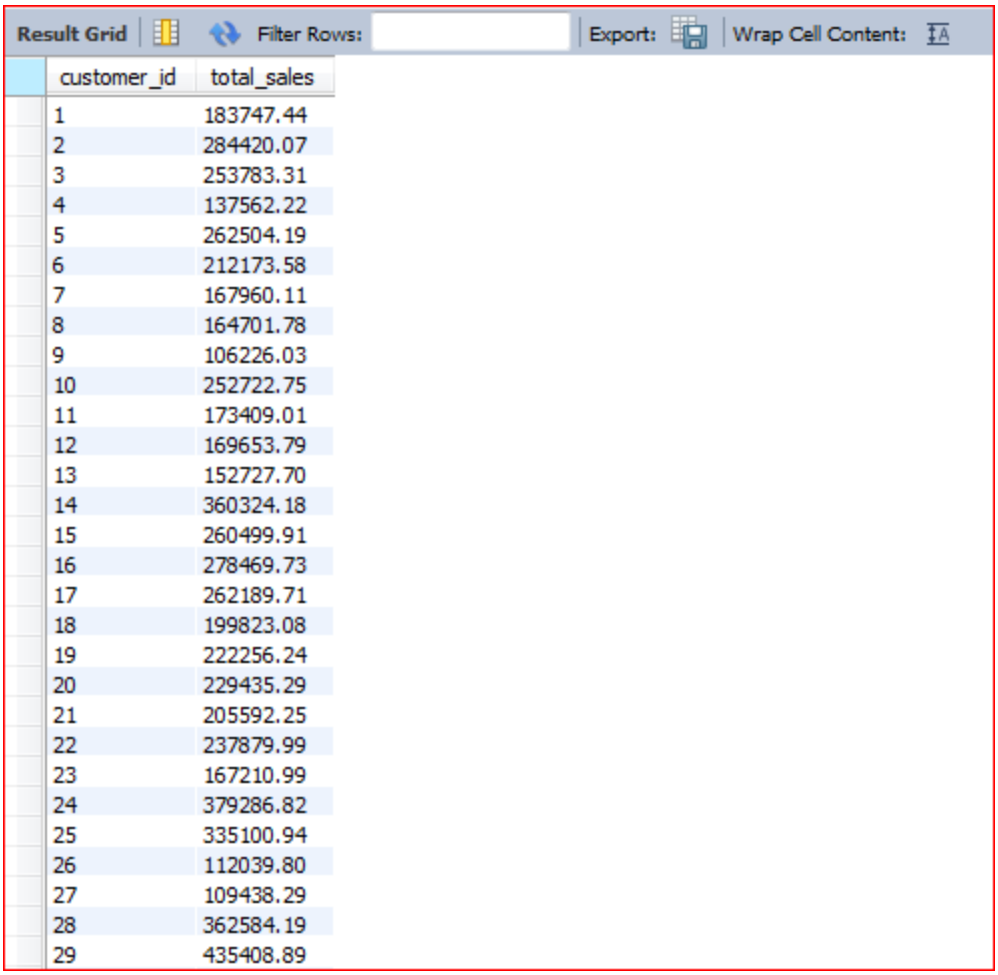
customer_id	orders_count
1	12
2	17
3	17
4	9
5	14
6	16
7	13
8	10
9	10
10	13
11	9
12	11
13	12
14	15
15	15
16	16
17	17
18	9
19	15
20	17
21	13
22	15
23	9
24	15
25	15
26	6
27	7
28	17
29	20

5. Find total sales amount made by each customer.

Query -

```
SELECT
        customer_id,
        SUM(total_amount) AS total_sales
FROM orders
GROUP BY customer_id
ORDER BY customer_id;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

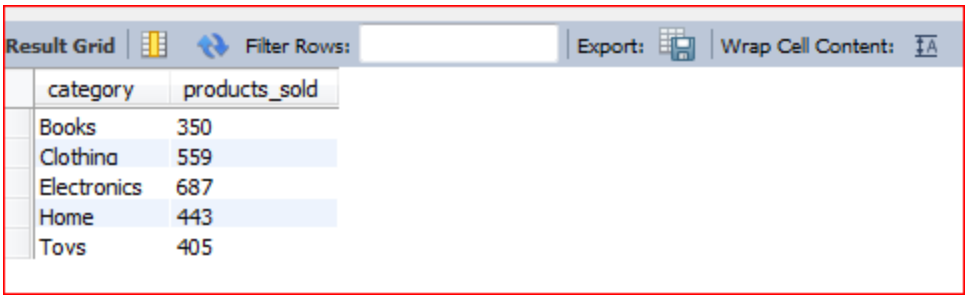
customer_id	total_sales
1	183747.44
2	284420.07
3	253783.31
4	137562.22
5	262504.19
6	212173.58
7	167960.11
8	164701.78
9	106226.03
10	252722.75
11	173409.01
12	169653.79
13	152727.70
14	360324.18
15	260499.91
16	278469.73
17	262189.71
18	199823.08
19	222256.24
20	229435.29
21	205592.25
22	237879.99
23	167210.99
24	379286.82
25	335100.94
26	112039.80
27	109438.29
28	362584.19
29	435408.89

7. List the number of products sold per category.

Query -

```
SELECT
        P.category,
        SUM(OI.quantity) AS products_sold
FROM products AS P
JOIN order_items AS OI
ON P.product_id = OI.product_id
GROUP BY P.category
ORDER BY P.category;
```

Output -



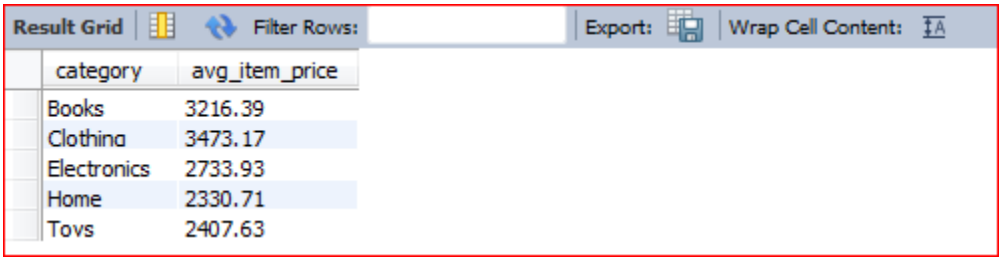
category	products_sold
Books	350
Clothing	559
Electronics	687
Home	443
Toys	405

8. Find the average item price per category.

Query -

```
SELECT
        P.category,
        ROUND(AVG(OI.item_price),2) AS avg_item_price
FROM products AS P
JOIN order_items AS OI
ON P.product_id = OI.product_id
GROUP BY P.category
ORDER BY P.category;
```

Output -



category	avg_item_price
Books	3216.39
Clothing	3473.17
Electronics	2733.93
Home	2330.71
Toys	2407.63

9. Show number of orders placed per day.

Query -

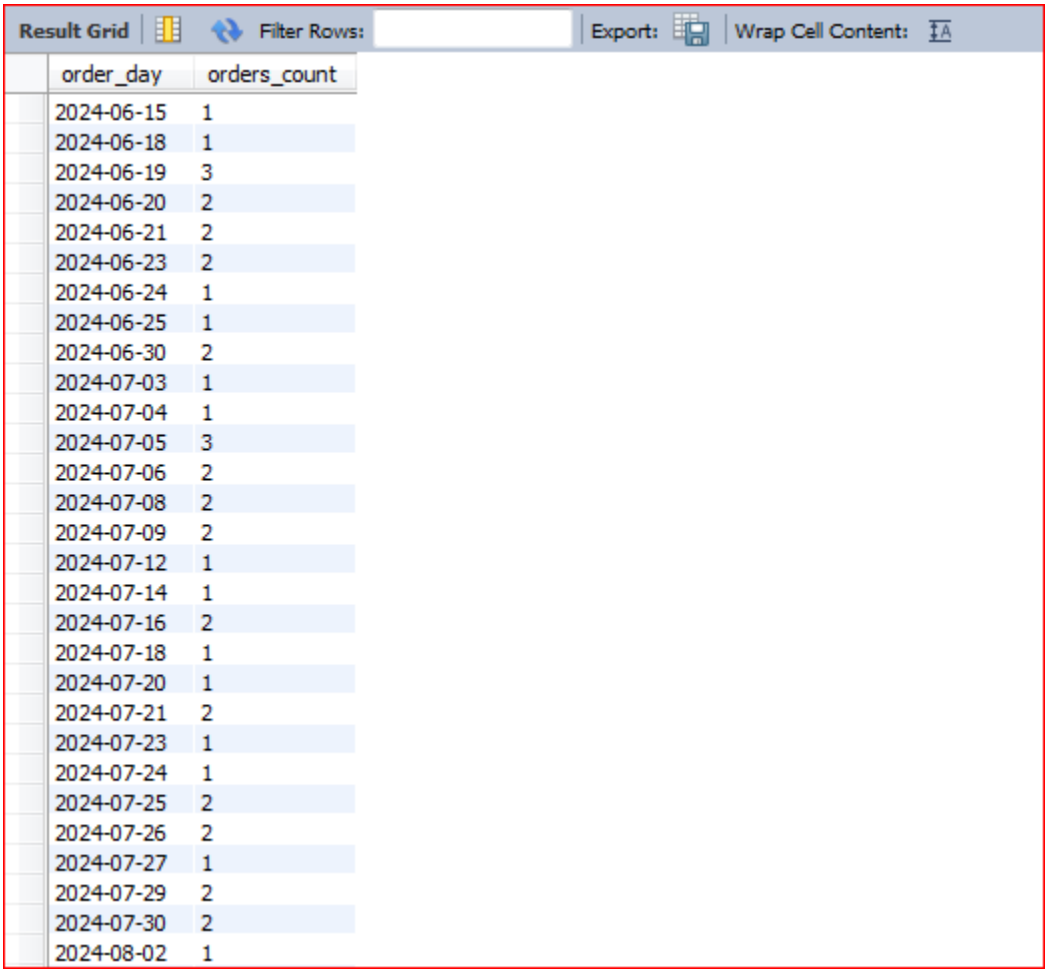
```
SELECT
        DATE(order_date) AS order_day,
        COUNT(*) AS orders_count
```

FROM orders

GROUP BY DATE(order_date)

ORDER BY order_day;

Output -


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_day	orders_count
2024-06-15	1
2024-06-18	1
2024-06-19	3
2024-06-20	2
2024-06-21	2
2024-06-23	2
2024-06-24	1
2024-06-25	1
2024-06-30	2
2024-07-03	1
2024-07-04	1
2024-07-05	3
2024-07-06	2
2024-07-08	2
2024-07-09	2
2024-07-12	1
2024-07-14	1
2024-07-16	2
2024-07-18	1
2024-07-20	1
2024-07-21	2
2024-07-23	1
2024-07-24	1
2024-07-25	2
2024-07-26	2
2024-07-27	1
2024-07-29	2
2024-07-30	2
2024-08-02	1

order_day	orders_count
2025-05-18	1
2025-05-20	2
2025-05-21	1
2025-05-22	1
2025-05-23	1
2025-05-25	1
2025-05-28	1
2025-05-30	1
2025-05-31	1
2025-06-04	1
2025-06-05	1
2025-06-06	5
2025-06-08	1
2025-06-09	3
2025-06-10	2
2025-06-11	3
2025-06-13	2

10. List total payments received per payment method.

Query -

```
SELECT
        method,
        SUM(amount_paid) AS total_payments_received
FROM payments
GROUP BY method
ORDER BY total_payments_received DESC;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

method	total_payments_received
Debit Card	1930577.88
Credit Card	1754603.10
Net Banking	1658383.90
UPI	1617408.78

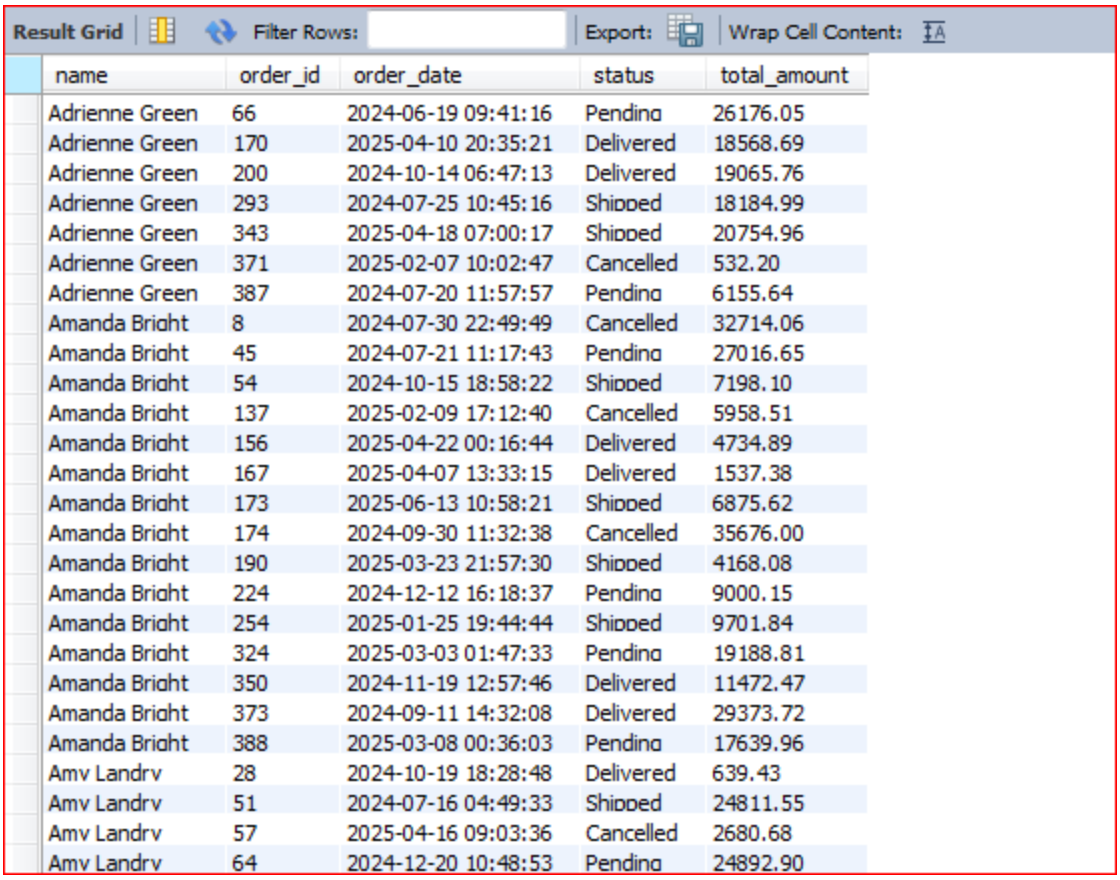
Level 4: Multi-Table Queries (JOINS)

1. Retrieve order details along with the customer name (INNER JOIN).

Query -

```
SELECT
        C.name,
        O.order_id,
        O.order_date,
        O.status,
        O.total_amount
FROM orders AS O
INNER JOIN customers AS C
ON O.customer_id = C.customer_id
ORDER BY C.name;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

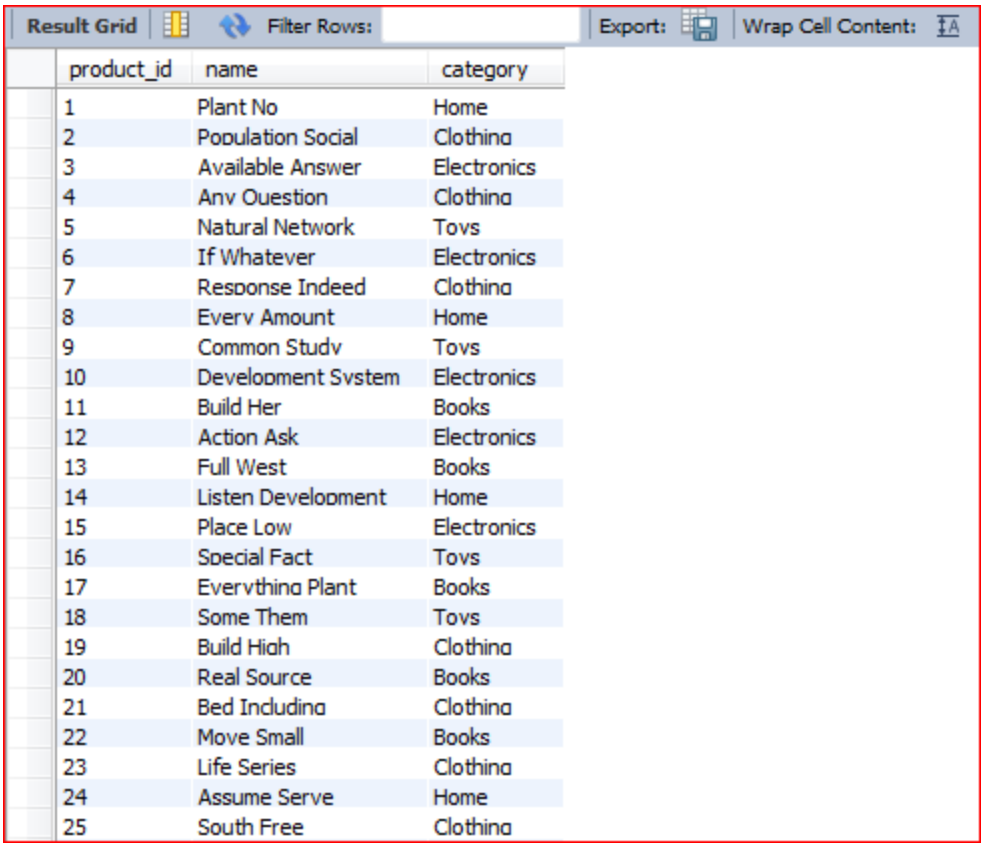
name	order_id	order_date	status	total_amount
Adrienne Green	66	2024-06-19 09:41:16	Pending	26176.05
Adrienne Green	170	2025-04-10 20:35:21	Delivered	18568.69
Adrienne Green	200	2024-10-14 06:47:13	Delivered	19065.76
Adrienne Green	293	2024-07-25 10:45:16	Shipped	18184.99
Adrienne Green	343	2025-04-18 07:00:17	Shipped	20754.96
Adrienne Green	371	2025-02-07 10:02:47	Cancelled	532.20
Adrienne Green	387	2024-07-20 11:57:57	Pending	6155.64
Amanda Bright	8	2024-07-30 22:49:49	Cancelled	32714.06
Amanda Bright	45	2024-07-21 11:17:43	Pending	27016.65
Amanda Bright	54	2024-10-15 18:58:22	Shipped	7198.10
Amanda Bright	137	2025-02-09 17:12:40	Cancelled	5958.51
Amanda Bright	156	2025-04-22 00:16:44	Delivered	4734.89
Amanda Bright	167	2025-04-07 13:33:15	Delivered	1537.38
Amanda Bright	173	2025-06-13 10:58:21	Shipped	6875.62
Amanda Bright	174	2024-09-30 11:32:38	Cancelled	35676.00
Amanda Bright	190	2025-03-23 21:57:30	Shipped	4168.08
Amanda Bright	224	2024-12-12 16:18:37	Pending	9000.15
Amanda Bright	254	2025-01-25 19:44:44	Shipped	9701.84
Amanda Bright	324	2025-03-03 01:47:33	Pending	19188.81
Amanda Bright	350	2024-11-19 12:57:46	Delivered	11472.47
Amanda Bright	373	2024-09-11 14:32:08	Delivered	29373.72
Amanda Bright	388	2025-03-08 00:36:03	Pending	17639.96
Amy Landry	28	2024-10-19 18:28:48	Delivered	639.43
Amy Landry	51	2024-07-16 04:49:33	Shipped	24811.55
Amy Landry	57	2025-04-16 09:03:36	Cancelled	2680.68
Amy Landry	64	2024-12-20 10:48:53	Pending	24892.90

2. Get list of products that have been sold (INNER JOIN with order_items).

Query -.

```
SELECT DISTINCT
        P.product_id,
        P.name,
        P.category
FROM products AS P
INNER JOIN order_items AS OI
ON P.product_id = OI.product_id;
```

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	name	category
1	Plant No	Home
2	Population Social	Clothing
3	Available Answer	Electronics
4	Any Question	Clothing
5	Natural Network	Toys
6	If Whatever	Electronics
7	Response Indeed	Clothing
8	Every Amount	Home
9	Common Study	Toys
10	Development System	Electronics
11	Build Her	Books
12	Action Ask	Electronics
13	Full West	Books
14	Listen Development	Home
15	Place Low	Electronics
16	Special Fact	Toys
17	Everything Plant	Books
18	Some Them	Toys
19	Build High	Clothing
20	Real Source	Books
21	Bed Including	Clothing
22	Move Small	Books
23	Life Series	Clothing
24	Assume Serve	Home
25	South Free	Clothing

product_id	name	category
26	Movement Future	Electronics
27	Answer But	Home
28	East Foot	Toys
29	Drop Remember	Electronics
30	Future Sort	Home
31	Series Page	Electronics
32	Factor Where	Clothing
33	Involve Officer	Clothing
34	Despite Win	Electronics
35	Fire Often	Electronics
36	Television Stock	Clothing
37	Between Up	Toys
38	Study Total	Toys
39	Data Add	Books
40	Toward Minute	Clothing
41	Serious Recognize	Electronics
42	Weight Enter	Clothing
43	Least Green	Toys
44	Actually Term	Electronics
45	Suffer We	Electronics
46	Age Treatment	Home
47	Southern Thing	Electronics
48	Group But	Clothing
49	Rest Something	Books
50	Stock Article	Home

3. List all orders with their payment method (INNER JOIN).

Query -

```
SELECT
        O.order_id,
        O.order_date,
        O.total_amount,
        PM.method
FROM orders AS O
INNER JOIN payments AS PM
ON O.order_id = PM.order_id;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_date	total_amount	method
1	2025-03-02 07:20:11	9414.28	Credit Card
2	2024-10-09 18:08:21	532.20	Net Banking
3	2025-05-08 00:08:27	5164.56	Credit Card
4	2024-09-19 22:16:13	9469.78	UPI
5	2025-04-08 18:02:06	14501.86	UPI
6	2024-10-25 07:33:59	31050.17	UPI
7	2024-07-29 11:58:47	3043.67	UPI
8	2024-07-30 22:49:49	32714.06	Net Banking
9	2025-06-10 17:00:25	24219.20	Net Banking
10	2025-02-16 12:45:59	24342.52	Debit Card
11	2025-03-11 14:22:46	16196.13	Credit Card
12	2025-01-15 13:22:53	9890.72	Credit Card
13	2025-05-12 13:04:29	9627.36	UPI
14	2024-09-24 21:21:38	15803.34	UPI
15	2024-07-18 03:18:24	12421.88	UPI
16	2024-09-09 03:00:40	22856.73	UPI
17	2024-08-19 21:17:57	11173.77	UPI
18	2024-06-15 12:57:04	32001.24	Net Banking
19	2024-07-06 15:28:24	843.46	Net Banking
20	2024-08-24 04:48:37	6400.68	Debit Card

4. Get list of customers and their orders (LEFT JOIN)

Used to find all customers and see who has or hasn't placed orders.

Query -

```
SELECT
```

```
    C.customer_id,
    C.name,
    O.order_id,
    O.order_date,
    O.total_amount
FROM customers AS C
LEFT JOIN orders AS O
ON C.customer_id = O.customer_id
ORDER BY C.customer_id;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

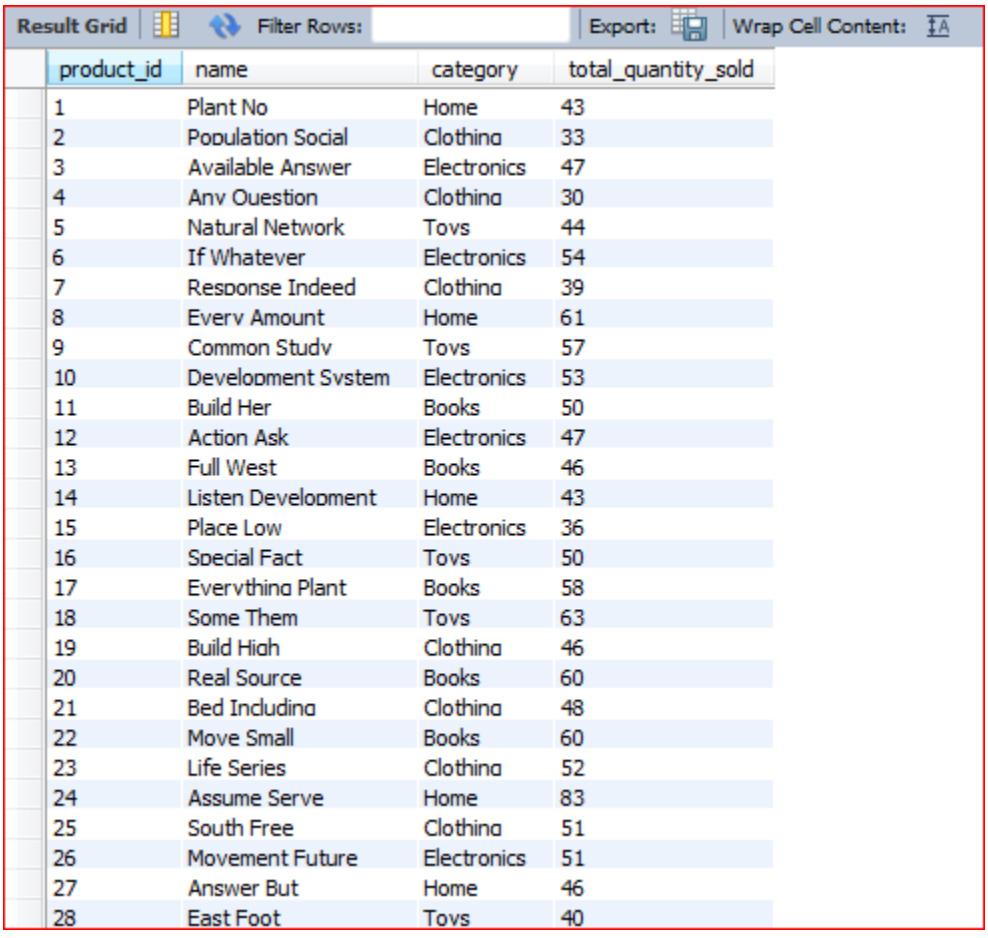
customer_id	name	order_id	order_date	total_amount
1	Thomas Owens	14	2024-09-24 21:21:38	15803.34
1	Thomas Owens	17	2024-08-19 21:17:57	11173.77
1	Thomas Owens	61	2024-12-26 23:21:58	13053.00
1	Thomas Owens	76	2025-01-14 22:59:54	23506.81
1	Thomas Owens	92	2024-09-26 11:33:58	834.75
1	Thomas Owens	109	2025-03-17 04:56:18	12190.14
1	Thomas Owens	127	2025-06-10 18:01:32	20160.64
1	Thomas Owens	135	2025-03-11 22:58:23	8891.79
1	Thomas Owens	144	2024-09-19 09:18:22	12093.54
1	Thomas Owens	221	2024-09-25 05:49:39	12437.61
1	Thomas Owens	278	2025-02-01 20:05:38	32015.16
1	Thomas Owens	379	2025-03-22 15:57:58	21586.89
2	Charles Grant	56	2024-06-23 15:59:57	6828.61
2	Charles Grant	169	2025-02-11 04:04:38	9426.30
2	Charles Grant	209	2024-10-16 10:07:48	7690.62
2	Charles Grant	223	2024-11-25 01:32:18	41020.75
2	Charles Grant	230	2024-07-25 22:19:17	22655.32
2	Charles Grant	232	2025-05-02 17:14:21	9583.84
2	Charles Grant	248	2025-06-06 14:36:15	9829.50
2	Charles Grant	283	2025-03-22 17:37:20	14493.79
2	Charles Grant	315	2025-01-27 15:23:27	42056.04
2	Charles Grant	319	2025-04-22 19:16:07	1024.92
2	Charles Grant	329	2025-03-19 10:36:50	31387.01
2	Charles Grant	334	2025-01-27 04:14:50	27857.86
2	Charles Grant	335	2024-08-29 05:53:29	11713.56

5. List all products along with order item quantity (LEFT JOIN).

Query -

```
SELECT
        P.product_id,
        P.name,
        P.category,
        SUM(OI.quantity AS total_quantity_sold
FROM products AS P
LEFT JOIN order_items AS OI
ON P.product_id = OI.product_id
GROUP BY P.product_id, P.name, P.category;
```

Output -



product_id	name	category	total_quantity_sold
1	Plant No	Home	43
2	Population Social	Clothing	33
3	Available Answer	Electronics	47
4	Any Question	Clothing	30
5	Natural Network	Toys	44
6	If Whatever	Electronics	54
7	Response Indeed	Clothing	39
8	Every Amount	Home	61
9	Common Study	Toys	57
10	Development System	Electronics	53
11	Build Her	Books	50
12	Action Ask	Electronics	47
13	Full West	Books	46
14	Listen Development	Home	43
15	Place Low	Electronics	36
16	Special Fact	Toys	50
17	Everything Plant	Books	58
18	Some Them	Toys	63
19	Build High	Clothing	46
20	Real Source	Books	60
21	Bed Including	Clothing	48
22	Move Small	Books	60
23	Life Series	Clothing	52
24	Assume Serve	Home	83
25	South Free	Clothing	51
26	Movement Future	Electronics	51
27	Answer But	Home	46
28	East Foot	Toys	40

6. List all payments including those with no matching orders (RIGHT JOIN).

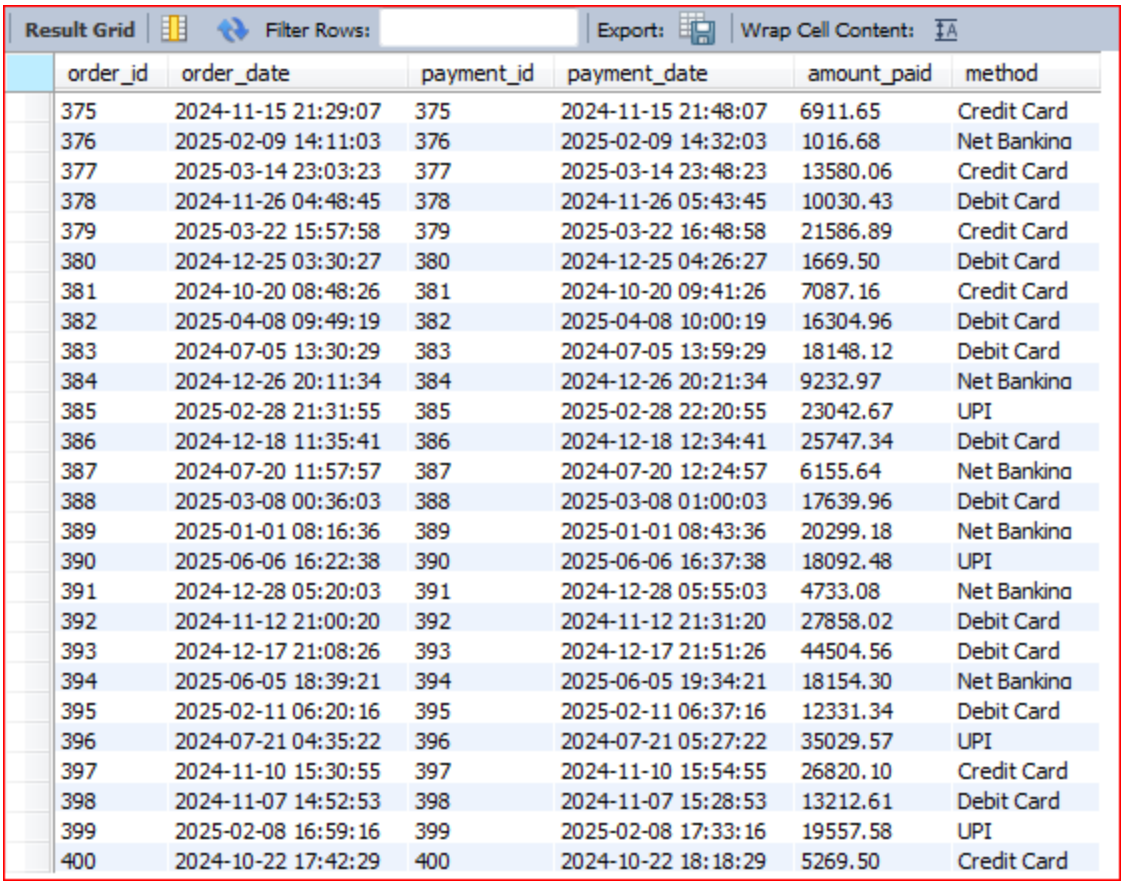
Query -

```
SELECT
        O.order_id,
        O.order_date,
        P.payment_id,
        P.payment_date,
        P.amount_paid,
        P.method
FROM orders AS O
RIGHT JOIN payments AS P
ON O.order_id = P.order_id;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_date	payment_id	payment_date	amount_paid	method
1	2025-03-02 07:20:11	1	2025-03-02 08:16:11	9414.28	Credit Card
2	2024-10-09 18:08:21	2	2024-10-09 19:00:21	532.20	Net Banking
3	2025-05-08 00:08:27	3	2025-05-08 00:55:27	5164.56	Credit Card
4	2024-09-19 22:16:13	4	2024-09-19 22:28:13	9469.78	UPI
5	2025-04-08 18:02:06	5	2025-04-08 18:26:06	14501.86	UPI
6	2024-10-25 07:33:59	6	2024-10-25 08:16:59	31050.17	UPI
7	2024-07-29 11:58:47	7	2024-07-29 12:45:47	3043.67	UPI
8	2024-07-30 22:49:49	8	2024-07-30 23:42:49	32714.06	Net Banking
9	2025-06-10 17:00:25	9	2025-06-10 17:36:25	24219.20	Net Banking
10	2025-02-16 12:45:59	10	2025-02-16 13:43:59	24342.52	Debit Card
11	2025-03-11 14:22:46	11	2025-03-11 15:05:46	16196.13	Credit Card
12	2025-01-15 13:22:53	12	2025-01-15 14:00:53	9890.72	Credit Card
13	2025-05-12 13:04:29	13	2025-05-12 14:01:29	9627.36	UPI
14	2024-09-24 21:21:38	14	2024-09-24 22:11:38	15803.34	UPI
15	2024-07-18 03:18:24	15	2024-07-18 03:45:24	12421.88	UPI
16	2024-09-09 03:00:40	16	2024-09-09 03:38:40	22856.73	UPI
17	2024-08-19 21:17:57	17	2024-08-19 21:49:57	11173.77	UPI
18	2024-06-15 12:57:04	18	2024-06-15 13:45:04	32001.24	Net Banking
19	2024-07-06 15:28:24	19	2024-07-06 15:43:24	843.46	Net Banking
20	2024-08-24 04:48:37	20	2024-08-24 04:59:37	6400.68	Debit Card
21	2025-03-12 18:53:46	21	2025-03-12 19:48:46	25364.11	Debit Card
22	2025-02-21 11:21:06	22	2025-02-21 12:05:06	21281.44	Credit Card
23	2025-03-02 14:33:23	23	2025-03-02 15:03:23	18045.05	Debit Card
24	2024-06-19 07:09:04	24	2024-06-19 07:28:04	22882.19	UPI
25	2024-07-24 12:56:24	25	2024-07-24 13:17:24	23749.66	UPI
26	2025-02-16 23:16:49	26	2025-02-16 23:26:49	35167.92	Credit Card



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_date	payment_id	payment_date	amount_paid	method
375	2024-11-15 21:29:07	375	2024-11-15 21:48:07	6911.65	Credit Card
376	2025-02-09 14:11:03	376	2025-02-09 14:32:03	1016.68	Net Banking
377	2025-03-14 23:03:23	377	2025-03-14 23:48:23	13580.06	Credit Card
378	2024-11-26 04:48:45	378	2024-11-26 05:43:45	10030.43	Debit Card
379	2025-03-22 15:57:58	379	2025-03-22 16:48:58	21586.89	Credit Card
380	2024-12-25 03:30:27	380	2024-12-25 04:26:27	1669.50	Debit Card
381	2024-10-20 08:48:26	381	2024-10-20 09:41:26	7087.16	Credit Card
382	2025-04-08 09:49:19	382	2025-04-08 10:00:19	16304.96	Debit Card
383	2024-07-05 13:30:29	383	2024-07-05 13:59:29	18148.12	Debit Card
384	2024-12-26 20:11:34	384	2024-12-26 20:21:34	9232.97	Net Banking
385	2025-02-28 21:31:55	385	2025-02-28 22:20:55	23042.67	UPI
386	2024-12-18 11:35:41	386	2024-12-18 12:34:41	25747.34	Debit Card
387	2024-07-20 11:57:57	387	2024-07-20 12:24:57	6155.64	Net Banking
388	2025-03-08 00:36:03	388	2025-03-08 01:00:03	17639.96	Debit Card
389	2025-01-01 08:16:36	389	2025-01-01 08:43:36	20299.18	Net Banking
390	2025-06-06 16:22:38	390	2025-06-06 16:37:38	18092.48	UPI
391	2024-12-28 05:20:03	391	2024-12-28 05:55:03	4733.08	Net Banking
392	2024-11-12 21:00:20	392	2024-11-12 21:31:20	27858.02	Debit Card
393	2024-12-17 21:08:26	393	2024-12-17 21:51:26	44504.56	Debit Card
394	2025-06-05 18:39:21	394	2025-06-05 19:34:21	18154.30	Net Banking
395	2025-02-11 06:20:16	395	2025-02-11 06:37:16	12331.34	Debit Card
396	2024-07-21 04:35:22	396	2024-07-21 05:27:22	35029.57	UPI
397	2024-11-10 15:30:55	397	2024-11-10 15:54:55	26820.10	Credit Card
398	2024-11-07 14:52:53	398	2024-11-07 15:28:53	13212.61	Debit Card
399	2025-02-08 16:59:16	399	2025-02-08 17:33:16	19557.58	UPI
400	2024-10-22 17:42:29	400	2024-10-22 18:18:29	5269.50	Credit Card

7. Combine data from three tables: customer, order, and payment.

Query -.

```
SELECT
        C.customer_id,
        C.name,
        O.order_id,
        O.order_date,
        O.total_amount,
        PM.payment_date,
        PM.amount_paid,
        PM.method
FROM customers AS C
```

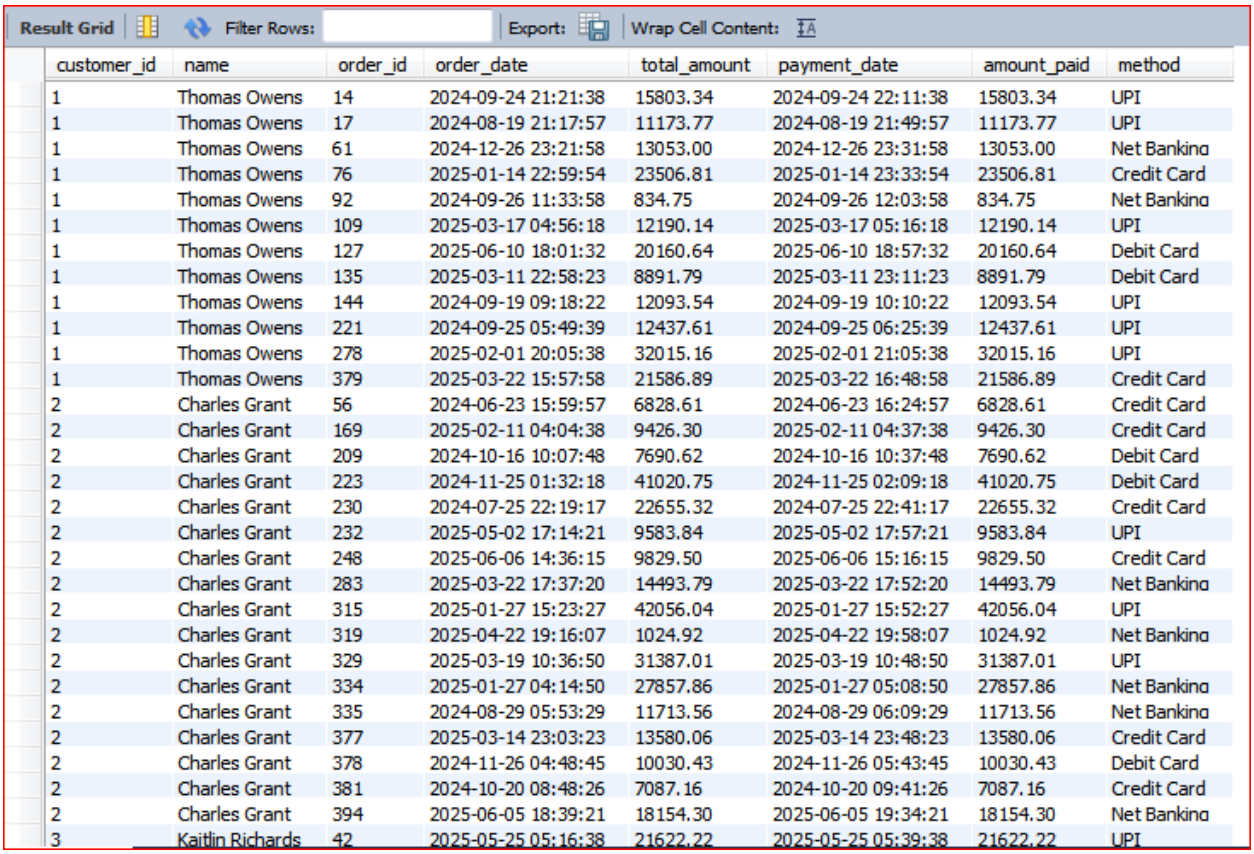
INNER JOIN orders AS O

ON C.customer_id = O.customer_id

INNER JOIN payments AS PM

ON O.order_id = PM.order_id;

Output -



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	name	order_id	order_date	total_amount	payment_date	amount_paid	method
1	Thomas Owens	14	2024-09-24 21:21:38	15803.34	2024-09-24 22:11:38	15803.34	UPI
1	Thomas Owens	17	2024-08-19 21:17:57	11173.77	2024-08-19 21:49:57	11173.77	UPI
1	Thomas Owens	61	2024-12-26 23:21:58	13053.00	2024-12-26 23:31:58	13053.00	Net Banking
1	Thomas Owens	76	2025-01-14 22:59:54	23506.81	2025-01-14 23:33:54	23506.81	Credit Card
1	Thomas Owens	92	2024-09-26 11:33:58	834.75	2024-09-26 12:03:58	834.75	Net Banking
1	Thomas Owens	109	2025-03-17 04:56:18	12190.14	2025-03-17 05:16:18	12190.14	UPI
1	Thomas Owens	127	2025-06-10 18:01:32	20160.64	2025-06-10 18:57:32	20160.64	Debit Card
1	Thomas Owens	135	2025-03-11 22:58:23	8891.79	2025-03-11 23:11:23	8891.79	Debit Card
1	Thomas Owens	144	2024-09-19 09:18:22	12093.54	2024-09-19 10:10:22	12093.54	UPI
1	Thomas Owens	221	2024-09-25 05:49:39	12437.61	2024-09-25 06:25:39	12437.61	UPI
1	Thomas Owens	278	2025-02-01 20:05:38	32015.16	2025-02-01 21:05:38	32015.16	UPI
1	Thomas Owens	379	2025-03-22 15:57:58	21586.89	2025-03-22 16:48:58	21586.89	Credit Card
2	Charles Grant	56	2024-06-23 15:59:57	6828.61	2024-06-23 16:24:57	6828.61	Credit Card
2	Charles Grant	169	2025-02-11 04:04:38	9426.30	2025-02-11 04:37:38	9426.30	Credit Card
2	Charles Grant	209	2024-10-16 10:07:48	7690.62	2024-10-16 10:37:48	7690.62	Debit Card
2	Charles Grant	223	2024-11-25 01:32:18	41020.75	2024-11-25 02:09:18	41020.75	Debit Card
2	Charles Grant	230	2024-07-25 22:19:17	22655.32	2024-07-25 22:41:17	22655.32	Credit Card
2	Charles Grant	232	2025-05-02 17:14:21	9583.84	2025-05-02 17:57:21	9583.84	UPI
2	Charles Grant	248	2025-06-06 14:36:15	9829.50	2025-06-06 15:16:15	9829.50	Credit Card
2	Charles Grant	283	2025-03-22 17:37:20	14493.79	2025-03-22 17:52:20	14493.79	Net Banking
2	Charles Grant	315	2025-01-27 15:23:27	42056.04	2025-01-27 15:52:27	42056.04	UPI
2	Charles Grant	319	2025-04-22 19:16:07	1024.92	2025-04-22 19:58:07	1024.92	Net Banking
2	Charles Grant	329	2025-03-19 10:36:50	31387.01	2025-03-19 10:48:50	31387.01	UPI
2	Charles Grant	334	2025-01-27 04:14:50	27857.86	2025-01-27 05:08:50	27857.86	Net Banking
2	Charles Grant	335	2024-08-29 05:53:29	11713.56	2024-08-29 06:09:29	11713.56	Net Banking
2	Charles Grant	377	2025-03-14 23:03:23	13580.06	2025-03-14 23:48:23	13580.06	Credit Card
2	Charles Grant	378	2024-11-26 04:48:45	10030.43	2024-11-26 05:43:45	10030.43	Debit Card
2	Charles Grant	381	2024-10-20 08:48:26	7087.16	2024-10-20 09:41:26	7087.16	Credit Card
2	Charles Grant	394	2025-06-05 18:39:21	18154.30	2025-06-05 19:34:21	18154.30	Net Banking
3	Kaitlin Richards	42	2025-05-25 05:16:38	21622.22	2025-05-25 05:39:38	21622.22	UPI

Level 5: Subqueries (Inner Queries)

1. List all products priced above the average product price.

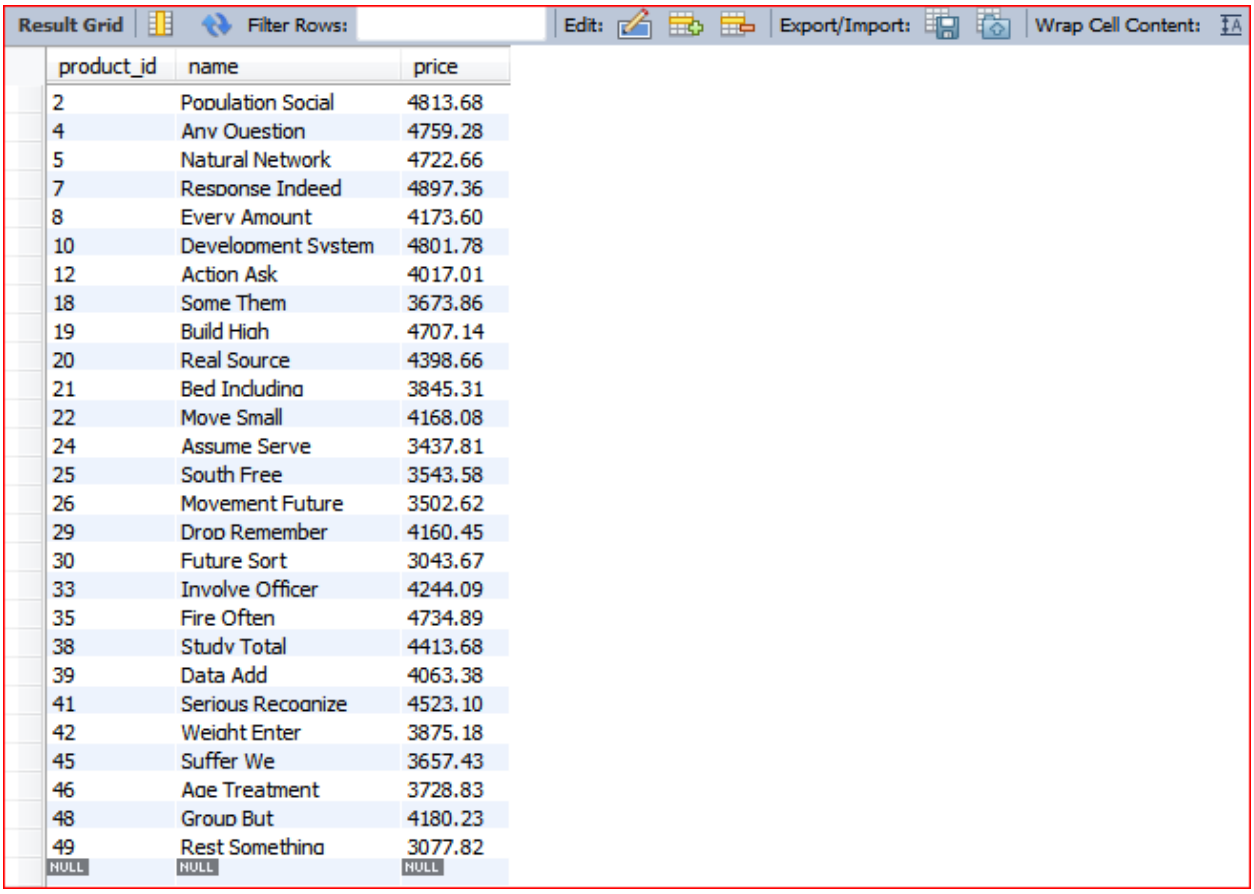
Query -.

SELECT product_id, name, price

FROM products

```
WHERE price >
(
SELECT AVG(price) FROM products
);
```

Output -



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	price
2	Population Social	4813.68
4	Any Question	4759.28
5	Natural Network	4722.66
7	Response Indeed	4897.36
8	Every Amount	4173.60
10	Development System	4801.78
12	Action Ask	4017.01
18	Some Them	3673.86
19	Build High	4707.14
20	Real Source	4398.66
21	Bed Including	3845.31
22	Move Small	4168.08
24	Assume Serve	3437.81
25	South Free	3543.58
26	Movement Future	3502.62
29	Drop Remember	4160.45
30	Future Sort	3043.67
33	Involve Officer	4244.09
35	Fire Often	4734.89
38	Study Total	4413.68
39	Data Add	4063.38
41	Serious Recognize	4523.10
42	Weight Enter	3875.18
45	Suffer We	3657.43
46	Age Treatment	3728.83
48	Group But	4180.23
49	Rest Something	3077.82
NULL	NULL	NULL

2. Find customers who have placed at least one order.

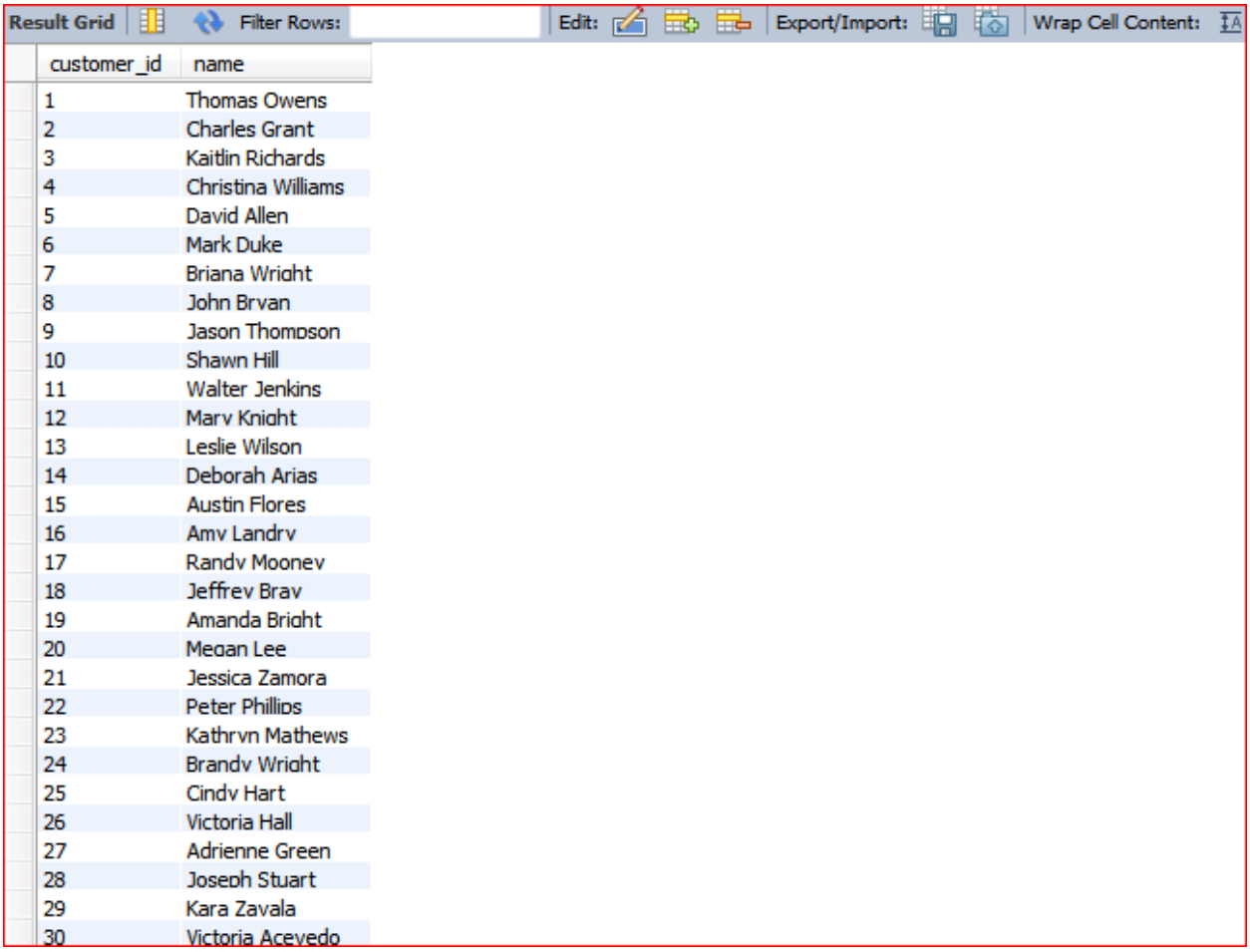
Query -

```
SELECT customer_id, name
FROM customers
WHERE customer_id IN
(
SELECT DISTINCT customer_id
```

FROM orders

);

Output -



customer_id	name
1	Thomas Owens
2	Charles Grant
3	Kaitlin Richards
4	Christina Williams
5	David Allen
6	Mark Duke
7	Briana Wright
8	John Bryan
9	Jason Thompson
10	Shawn Hill
11	Walter Jenkins
12	Mary Knight
13	Leslie Wilson
14	Deborah Arias
15	Austin Flores
16	Amy Landry
17	Randy Mooney
18	Jeffrey Bray
19	Amanda Bright
20	Megan Lee
21	Jessica Zamora
22	Peter Phillips
23	Kathryn Mathews
24	Brandy Wright
25	Cindy Hart
26	Victoria Hall
27	Adrienne Green
28	Joseph Stuart
29	Kara Zavala
30	Victoria Acevedo

3. Show orders whose total amount is above the average for that customer.

Query -

```
SELECT
        O.order_id,
        O.customer_id,
        O.total_amount
FROM orders AS O
WHERE O.total_amount >
```

(

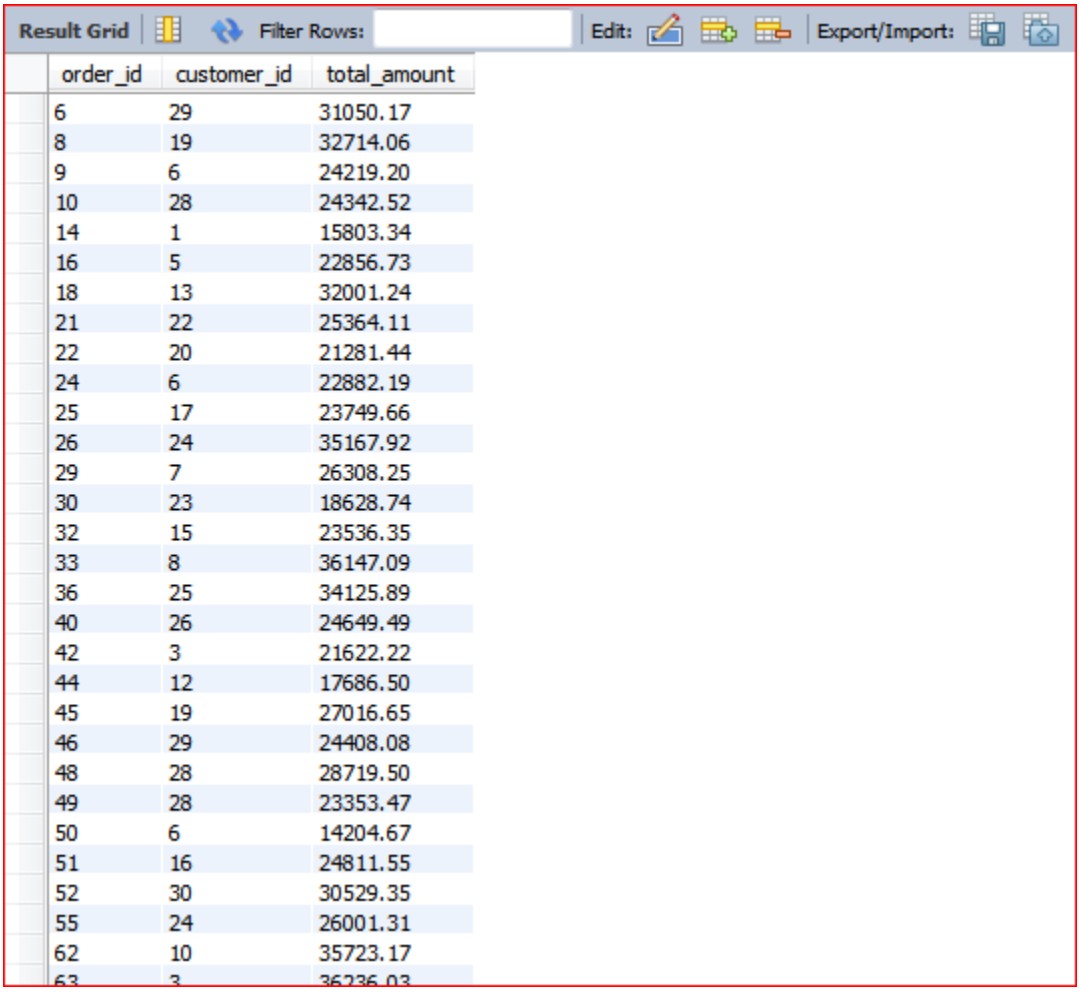
SELECT AVG(OD.total_amount)

FROM orders AS OD

WHERE OD.customer_id = O.customer_id

);

Output -


Result Grid | Filter Rows: | Edit: | Export/Import:

order_id	customer_id	total_amount
6	29	31050.17
8	19	32714.06
9	6	24219.20
10	28	24342.52
14	1	15803.34
16	5	22856.73
18	13	32001.24
21	22	25364.11
22	20	21281.44
24	6	22882.19
25	17	23749.66
26	24	35167.92
29	7	26308.25
30	23	18628.74
32	15	23536.35
33	8	36147.09
36	25	34125.89
40	26	24649.49
42	3	21622.22
44	12	17686.50
45	19	27016.65
46	29	24408.08
48	28	28719.50
49	28	23353.47
50	6	14204.67
51	16	24811.55
52	30	30529.35
55	24	26001.31
62	10	35723.17
63	3	36236.03

4. Display customers who haven't placed any orders.

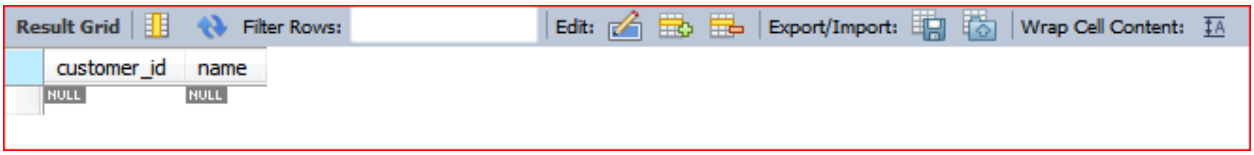
Query -

SELECT customer_id, name

FROM customers

```
WHERE customer_id NOT IN (
   SELECT customer_id
   FROM orders
);
```

Output -



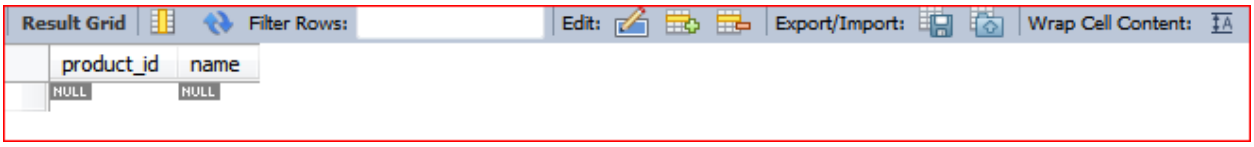
** No Customers found who haven't placed any orders.

5. Show products that were never ordered.

Query -

```
SELECT product_id, name
FROM products
WHERE product_id NOT IN (
   SELECT DISTINCT product_id
   FROM order_items
);
```

Output -



6. Show highest value order per customer.

Query -

```
SELECT order_id,customer_id,total_amount
FROM orders
WHERE (customer_id, total_amount) IN
```

(

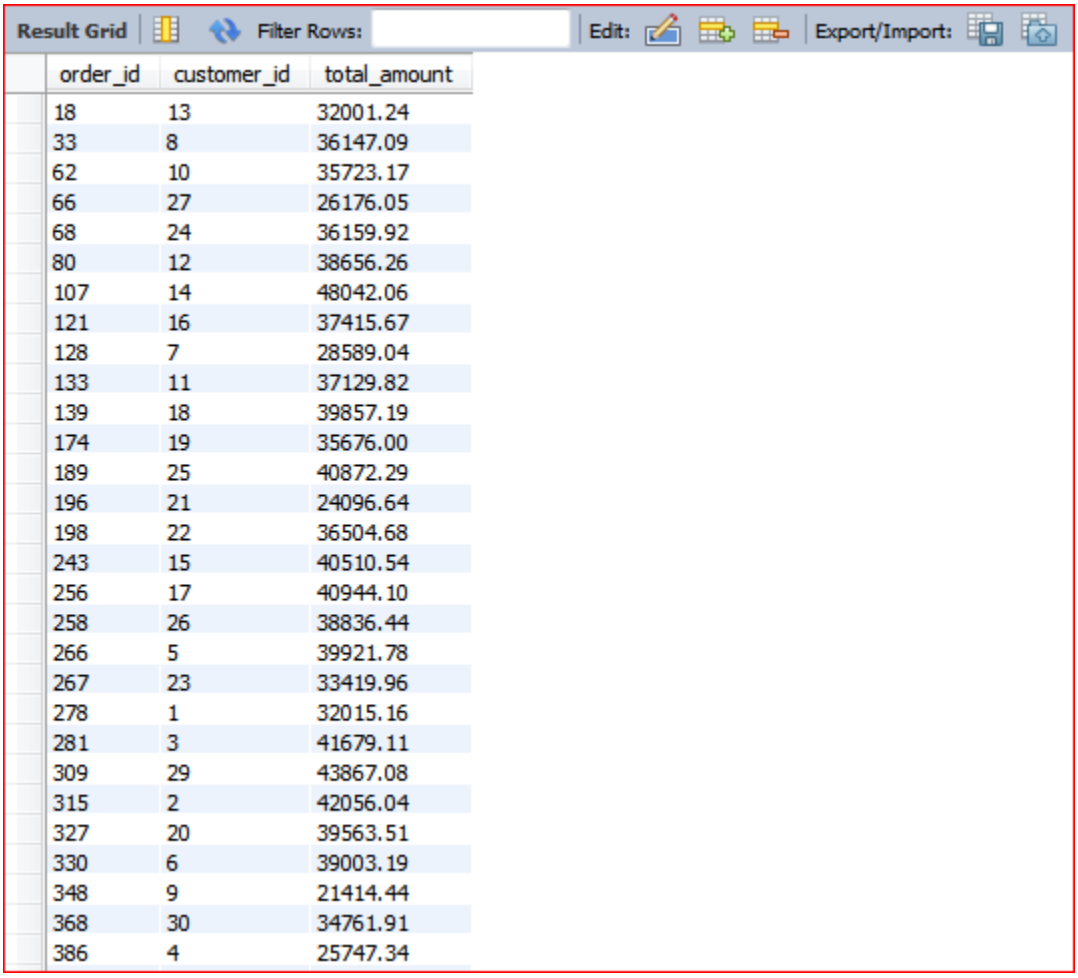
SELECT customer_id, MAX(total_amount)

FROM orders

GROUP BY customer_id

);

Output -



order_id	customer_id	total_amount
18	13	32001.24
33	8	36147.09
62	10	35723.17
66	27	26176.05
68	24	36159.92
80	12	38656.26
107	14	48042.06
121	16	37415.67
128	7	28589.04
133	11	37129.82
139	18	39857.19
174	19	35676.00
189	25	40872.29
196	21	24096.64
198	22	36504.68
243	15	40510.54
256	17	40944.10
258	26	38836.44
266	5	39921.78
267	23	33419.96
278	1	32015.16
281	3	41679.11
309	29	43867.08
315	2	42056.04
327	20	39563.51
330	6	39003.19
348	9	21414.44
368	30	34761.91
386	4	25747.34

7. Highest Order Per Customer (Including Names)

Used to identify the largest transaction made by each customer. Outputs name as well.

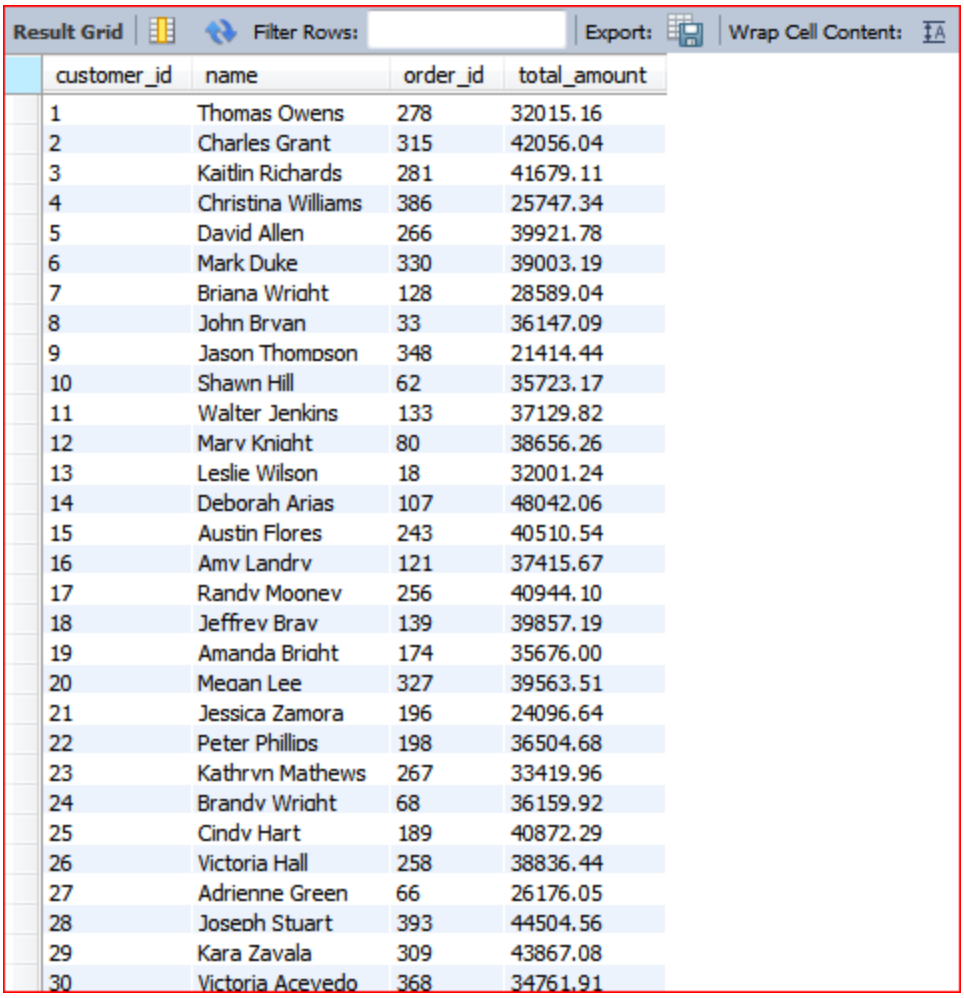
Query -

SELECT

C.customer_id,

```
        C.name,
        O.order_id,
        O.total_amount
FROM orders AS O
JOIN customers AS C
ON O.customer_id = C.customer_id
WHERE (O.customer_id, O.total_amount) IN
(
   SELECT customer_id, MAX(total_amount)
   FROM orders
   GROUP BY customer_id
);
```

Output -



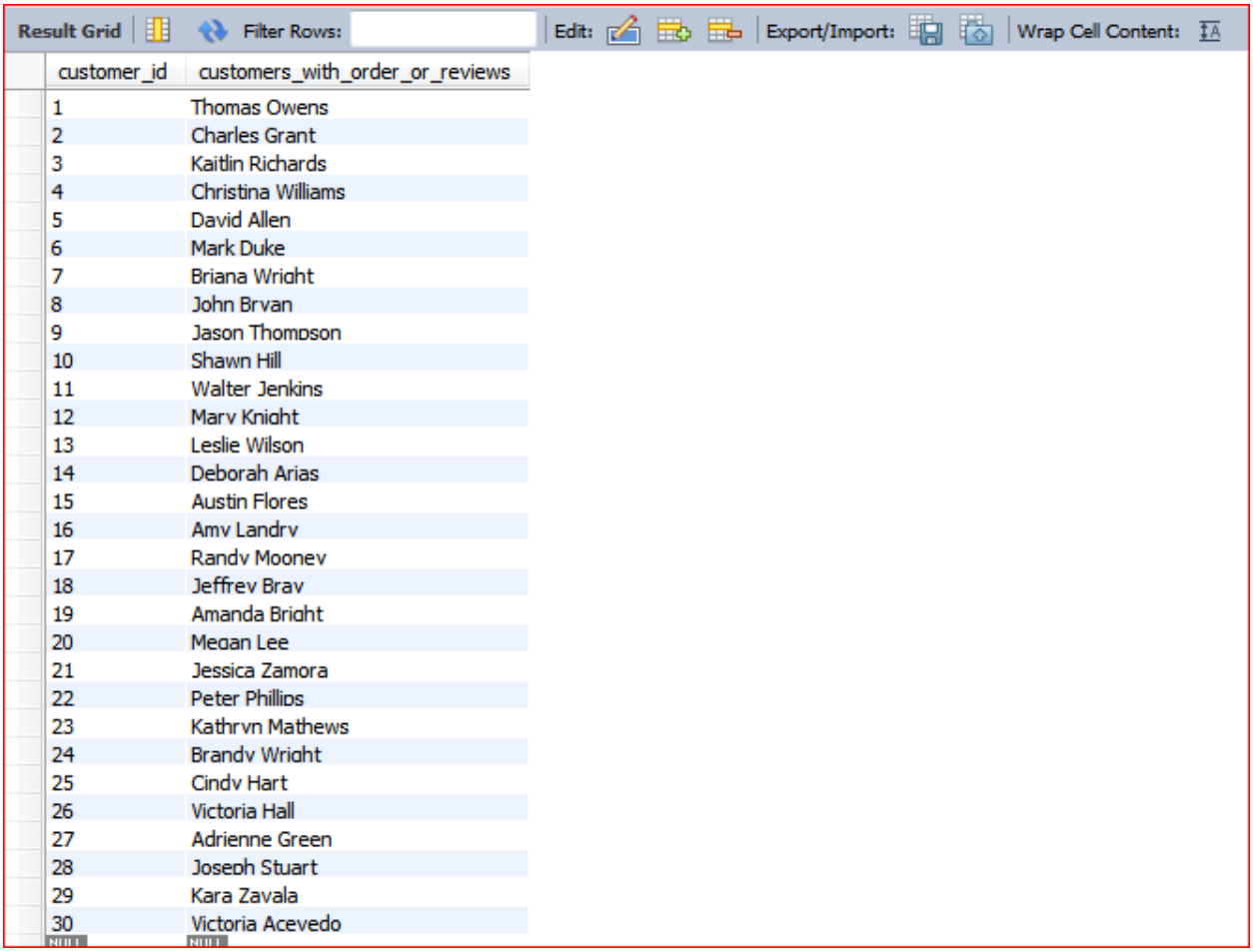
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	name	order_id	total_amount
1	Thomas Owens	278	32015.16
2	Charles Grant	315	42056.04
3	Kaitlin Richards	281	41679.11
4	Christina Williams	386	25747.34
5	David Allen	266	39921.78
6	Mark Duke	330	39003.19
7	Briana Wright	128	28589.04
8	John Bryan	33	36147.09
9	Jason Thompson	348	21414.44
10	Shawn Hill	62	35723.17
11	Walter Jenkins	133	37129.82
12	Mary Knight	80	38656.26
13	Leslie Wilson	18	32001.24
14	Deborah Arias	107	48042.06
15	Austin Flores	243	40510.54
16	Amy Landry	121	37415.67
17	Randy Mooney	256	40944.10
18	Jeffrey Bray	139	39857.19
19	Amanda Bright	174	35676.00
20	Megan Lee	327	39563.51
21	Jessica Zamora	196	24096.64
22	Peter Phillips	198	36504.68
23	Kathryn Mathews	267	33419.96
24	Brandy Wright	68	36159.92
25	Cindy Hart	189	40872.29
26	Victoria Hall	258	38836.44
27	Adrienne Green	66	26176.05
28	Joseph Stuart	393	44504.56
29	Kara Zavala	309	43867.08
30	Victoria Acevedo	368	34761.91

Level 6: Set Operations

1. List all customers who have either placed an order or written a product review

```
SELECT
        C.customer_id
        C.name as customers_with_order_or_reviews
FROM customers AS C
WHERE C.customer_id IN (
        SELECT customer_id FROM orders
        UNION
        SELECT customer_id FROM product_reviews
);
```



customer_id	customers_with_order_or_reviews
1	Thomas Owens
2	Charles Grant
3	Kaitlin Richards
4	Christina Williams
5	David Allen
6	Mark Duke
7	Briana Wright
8	John Bryan
9	Jason Thompson
10	Shawn Hill
11	Walter Jenkins
12	Mary Knight
13	Leslie Wilson
14	Deborah Arias
15	Austin Flores
16	Amy Landry
17	Randy Mooney
18	Jeffrey Bray
19	Amanda Bright
20	Megan Lee
21	Jessica Zamora
22	Peter Phillips
23	Kathryn Mathews
24	Brandy Wright
25	Cindy Hart
26	Victoria Hall
27	Adrienne Green
28	Joseph Stuart
29	Kara Zavala
30	Victoria Acevedo

2. List all customers who have placed an order as well as reviewed a product [intersect not supported]

Query -

```
SELECT DISTINCT C.customer_id, C.name AS customers_with_order_and_reviews
FROM customers AS C
INNER JOIN orders AS O
ON C.customer_id = O.customer_id
INNER JOIN product_reviews AS R
ON C.customer_id = R.customer_id;
```

Output -

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	customers_with_order_and_reviews
1	Thomas Owens
2	Charles Grant
4	Christina Williams
6	Mark Duke
7	Briana Wright
9	Jason Thompson
10	Shawn Hill
11	Walter Jenkins
13	Leslie Wilson
14	Deborah Arias
15	Austin Flores
17	Randy Mooney
19	Amanda Bright
20	Megan Lee
21	Jessica Zamora
22	Peter Phillips
23	Kathryn Mathews
25	Cindy Hart
26	Victoria Hall
27	Adrienne Green
28	Joseph Stuart
29	Kara Zavala
30	Victoria Acevedo