

Step 1 : Create Database :

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the schema is set to 'elevate_lab'. The 'Query' tab contains the following SQL code:

```
1 • create database elevate_lab;
2 • use elevate_lab;
3 • select * from ecommercepk1;
4
5 • # 1.Use Select,Where, Orderby, groupby.
6 • # select
7 • select * from ecommercepk1;
```

The 'Result Grid' pane displays the results of the 'select * from ecommercepk1;' query. The data is as follows:

Page Name	Day	Amount spent (PKR)	Reporting starts	Reporting ends
censored	10-06-2025	479.85	10-06-2025	10-06-2025
censored	09-06-2025	2807.17	09-06-2025	09-06-2025
censored	08-06-2025	2051.74	08-06-2025	08-06-2025
censored	07-06-2025	2392.81	07-06-2025	07-06-2025
censored	06-06-2025	1279.79	06-06-2025	06-06-2025
censored	30-05-2025	2025.59	30-05-2025	30-05-2025
censored	29-05-2025	2329.9	29-05-2025	29-05-2025
censored	28-05-2025	1653.92	28-05-2025	28-05-2025
censored	26-05-2025	243.41	26-05-2025	26-05-2025
censored	25-05-2025	742.82	25-05-2025	25-05-2025

Step 2 : Use Select , where :

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the schema is set to 'elevate_lab'. The 'Query' tab contains the following SQL code:

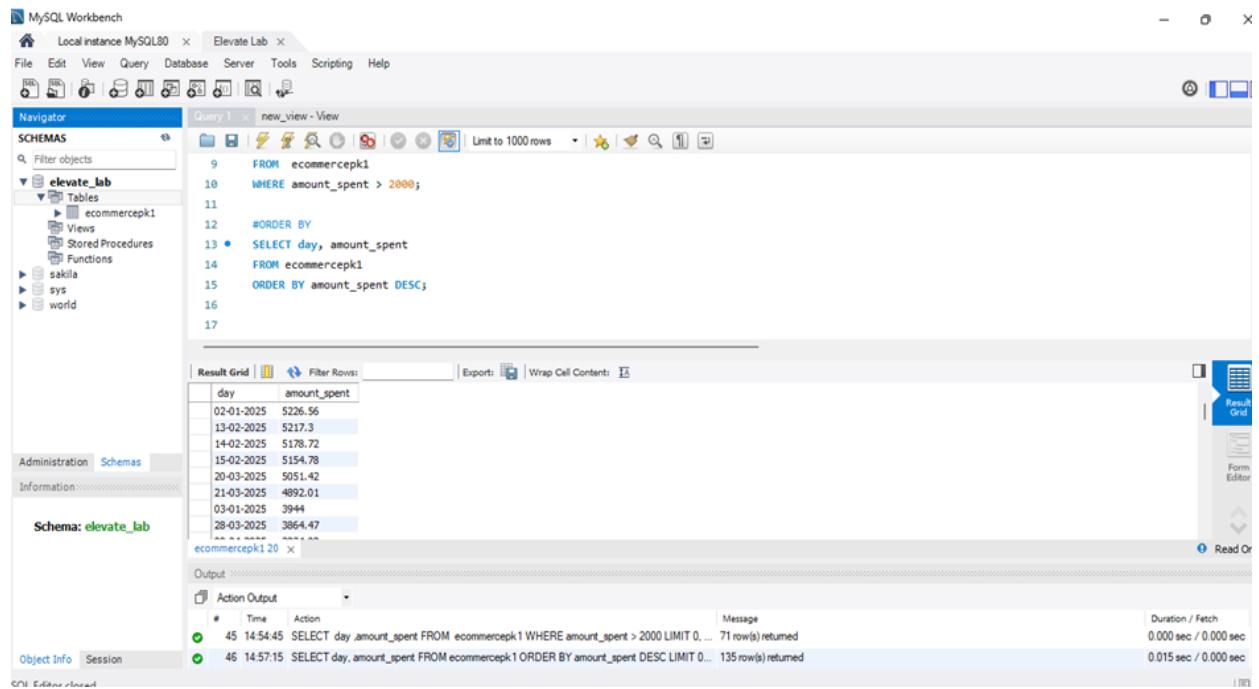
```
1 • create database Elevate_Lab;
2 • use Elevate_Lab;
3
4 • select * from ecommercepk1;
5
6 • SELECT day, amount_spent
7   FROM ecommercepk1
8   WHERE amount_spent > 2000;
```

The 'Result Grid' pane displays the results of the modified 'select' query. The data is as follows:

day	amount_spent
09-06-2025	2807.17
08-06-2025	2051.74
07-06-2025	2392.81
30-05-2025	2025.59
29-05-2025	2329.9
18-04-2025	2148.22
16-04-2025	2885.34
15-04-2025	2991.05

The 'Output' pane shows the message: '135 row(s) returned'.

Step 3 : Use Order by:



The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```
9   FROM ecommercepk1
10  WHERE amount_spent > 2000;
11
12  #ORDER BY
13 *  SELECT day, amount_spent
14  FROM ecommercepk1
15  ORDER BY amount_spent DESC;
16
17
```

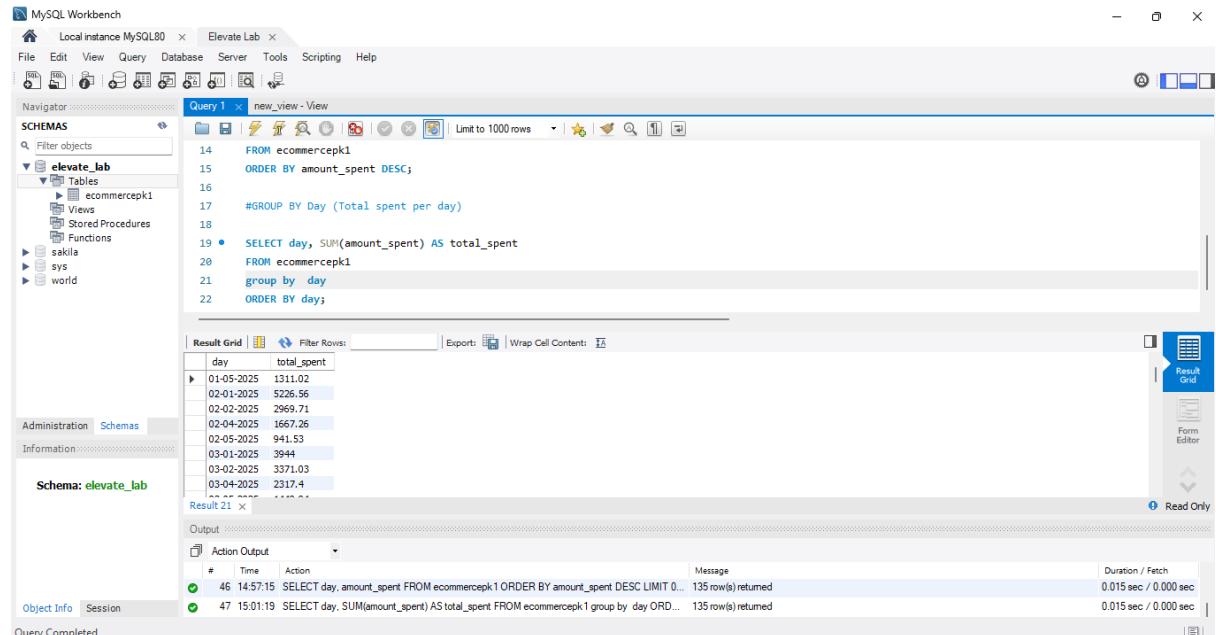
The result grid displays the following data:

day	amount_spent
02-01-2025	5226.56
13-02-2025	5217.3
14-02-2025	5178.72
15-02-2025	5154.78
20-03-2025	5051.42
21-03-2025	4892.01
03-01-2025	3944
28-03-2025	3864.47

The output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
45	14:54:45	SELECT day,amount_spent FROM ecommercepk1 WHERE amount_spent > 2000 LIMIT 0... 71 row(s) returned		0.000 sec / 0.000 sec
46	14:57:15	SELECT day,amount_spent FROM ecommercepk1 ORDER BY amount_spent DESC LIMIT 0... 135 row(s) returned		0.015 sec / 0.000 sec

Step 4 : Use Group by:



The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:

```
14  FROM ecommercepk1
15  ORDER BY amount_spent DESC;
16
17  #GROUP BY Day (Total spent per day)
18
19 *  SELECT day, SUM(amount_spent) AS total_spent
20  FROM ecommercepk1
21  group by day
22  ORDER BY day;
```

The result grid displays the following data:

day	total_spent
01-05-2025	1311.02
02-01-2025	5226.56
02-02-2025	2969.71
02-04-2025	1667.26
02-05-2025	941.53
03-01-2025	3944
03-02-2025	3371.03
03-04-2025	2317.4

The output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
46	14:57:15	SELECT day,amount_spent FROM ecommercepk1 ORDER BY amount_spent DESC LIMIT 0... 135 row(s) returned		0.015 sec / 0.000 sec
47	15:01:19	SELECT day, SUM(amount_spent) AS total_spent FROM ecommercepk1 group by day ORD... 135 row(s) returned		0.015 sec / 0.000 sec

Step 5 : Create new Table and insert values:

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'elevate_lab' schema, there is a 'Tables' folder containing 'ecommercepk1' and 'page_details'. The 'page_details' table is selected. In the Query Editor (Query 1), the following SQL code is written:

```
CREATE TABLE page_details (
    page_name VARCHAR(50),
    category VARCHAR(50)
);
# insert values
INSERT INTO page_details (page_name, category)
VALUES
('censored', 'Shopping'),
('ABC Store', 'Grocery'),
('XYZ Ads', 'Electronics');
select * from page_details;
```

The Result Grid shows the inserted data:

page_name	category
censored	Shopping
ABC Store	Grocery
XYZ Ads	Electronics
censored	Shopping
ABC Store	Grocery
XYZ Ads	Electronics

Step 6 : Use join: Inner join :

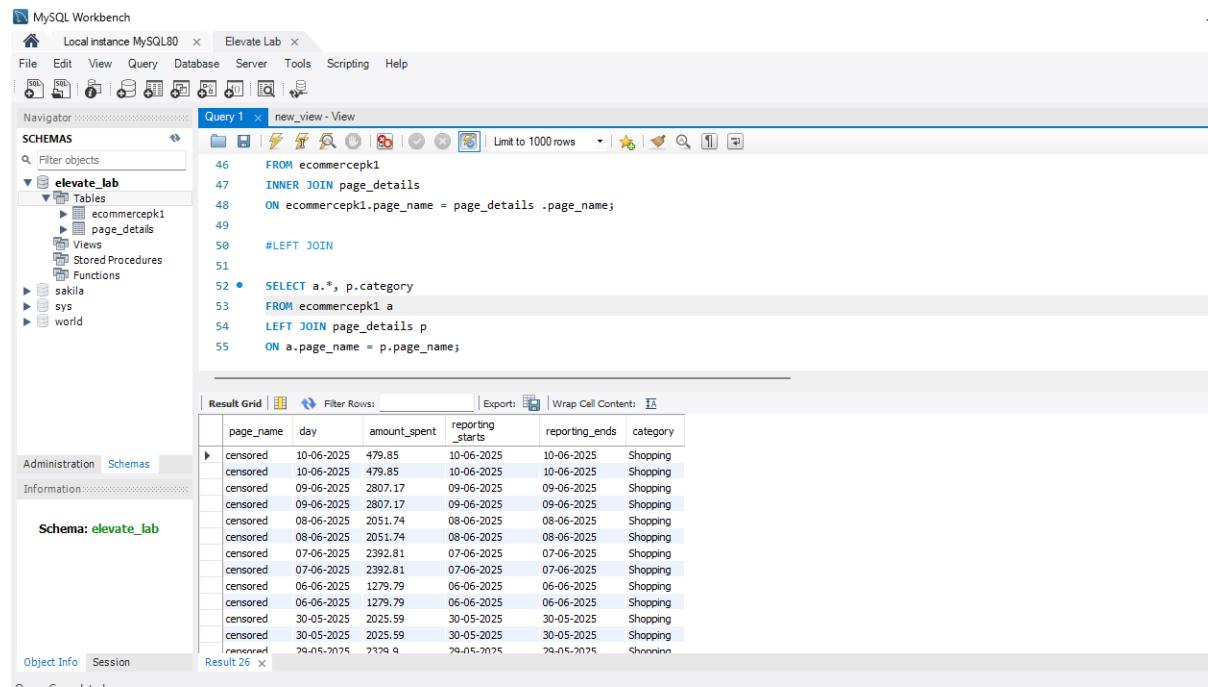
The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'elevate_lab' schema, there is a 'Tables' folder containing 'ecommercepk1' and 'page_details'. Both are selected. In the Query Editor (Query 1), the following SQL code is written:

```
#INNER JOIN
SELECT ecommercepk1.day, ecommercepk1.amount_spent, page_details .category
FROM ecommercepk1
INNER JOIN page_details
ON ecommercepk1.page_name = page_details .page_name;
```

The Result Grid shows the joined data:

day	amount_spent	category
10-06-2025	479.85	Shopping
10-06-2025	479.85	Shopping
09-06-2025	2807.17	Shopping
09-06-2025	2807.17	Shopping
08-06-2025	2051.74	Shopping
08-06-2025	2051.74	Shopping
07-06-2025	2392.81	Shopping
07-06-2025	2392.81	Shopping
06-06-2025	1279.79	Shopping
06-06-2025	1279.79	Shopping
30-05-2025	2025.59	Shopping
30-05-2025	2025.59	Shopping
29-05-2025	2329.9	Shopping

Step 6.1 : Left join :



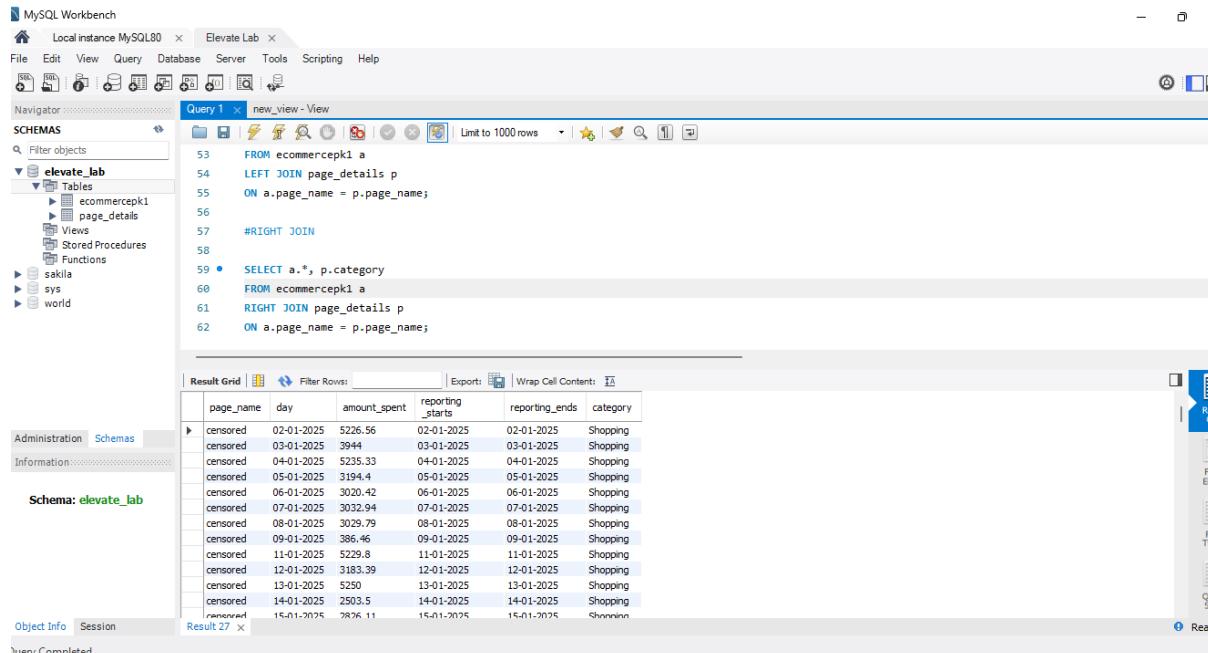
The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1" containing the following SQL code:

```
46 FROM ecommercepk1
47 INNER JOIN page_details
48 ON ecommercepk1.page_name = page_details .page_name;
49
50 #LEFT JOIN
51
52 • SELECT a.*, p.category
53 FROM ecommercepk1 a
54 LEFT JOIN page_details p
55 ON a.page_name = p.page_name;
```

The result grid displays the following data:

page_name	day	amount_spent	reporting_starts	reporting_ends	category
censored	10-06-2025	479.85	10-06-2025	10-06-2025	Shopping
censored	10-06-2025	479.85	10-06-2025	10-06-2025	Shopping
censored	09-06-2025	2807.17	09-06-2025	09-06-2025	Shopping
censored	09-06-2025	2807.17	09-06-2025	09-06-2025	Shopping
censored	08-06-2025	2051.74	08-06-2025	08-06-2025	Shopping
censored	08-06-2025	2051.74	08-06-2025	08-06-2025	Shopping
censored	07-06-2025	2392.81	07-06-2025	07-06-2025	Shopping
censored	07-06-2025	2392.81	07-06-2025	07-06-2025	Shopping
censored	06-06-2025	1279.79	06-06-2025	06-06-2025	Shopping
censored	06-06-2025	1279.79	06-06-2025	06-06-2025	Shopping
censored	30-05-2025	2025.59	30-05-2025	30-05-2025	Shopping
censored	30-05-2025	2025.59	30-05-2025	30-05-2025	Shopping
censored	29-05-2025	7179.9	29-05-2025	29-05-2025	Shopping

Step 6.2 : Use Right Join :



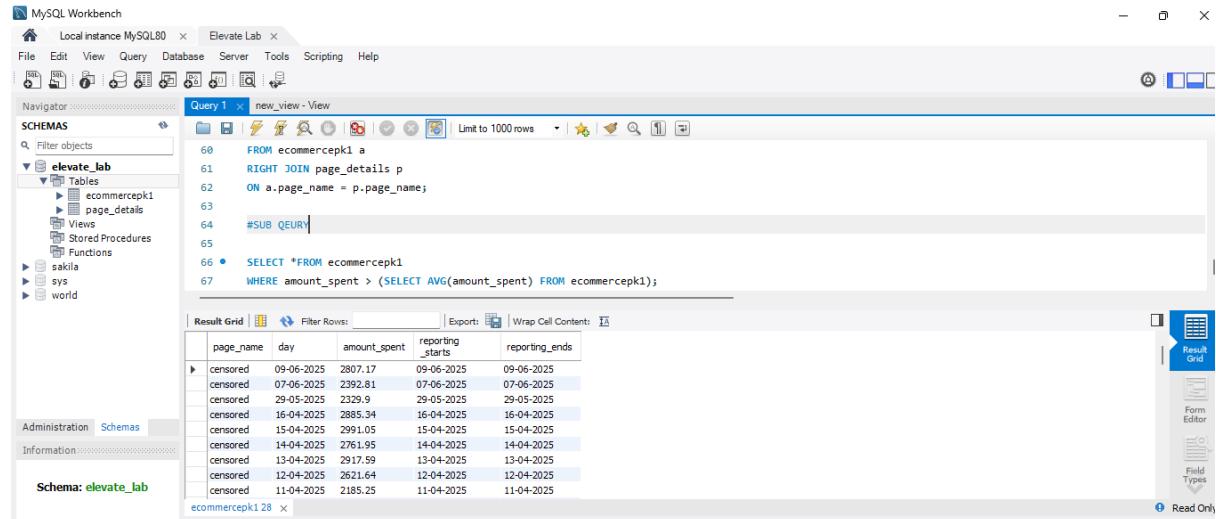
The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1" containing the following SQL code:

```
53 FROM ecommercepk1 a
54 LEFT JOIN page_details p
55 ON a.page_name = p.page_name;
56
57 #RIGHT JOIN
58
59 • SELECT a.*, p.category
60 FROM ecommercepk1 a
61 RIGHT JOIN page_details p
62 ON a.page_name = p.page_name;
```

The result grid displays the following data:

page_name	day	amount_spent	reporting_starts	reporting_ends	category
censored	02-01-2025	5226.56	02-01-2025	02-01-2025	Shopping
censored	03-01-2025	3944	03-01-2025	03-01-2025	Shopping
censored	04-01-2025	5235.33	04-01-2025	04-01-2025	Shopping
censored	05-01-2025	3194.4	05-01-2025	05-01-2025	Shopping
censored	06-01-2025	3020.42	06-01-2025	06-01-2025	Shopping
censored	07-01-2025	3032.94	07-01-2025	07-01-2025	Shopping
censored	08-01-2025	3029.79	08-01-2025	08-01-2025	Shopping
censored	09-01-2025	386.46	09-01-2025	09-01-2025	Shopping
censored	11-01-2025	5229.8	11-01-2025	11-01-2025	Shopping
censored	12-01-2025	3183.39	12-01-2025	12-01-2025	Shopping
censored	13-01-2025	5250	13-01-2025	13-01-2025	Shopping
censored	14-01-2025	2503.5	14-01-2025	14-01-2025	Shopping
censored	15-01-2025	2826.11	15-01-2025	15-01-2025	Shopping

Step 7: Write Subqueries :



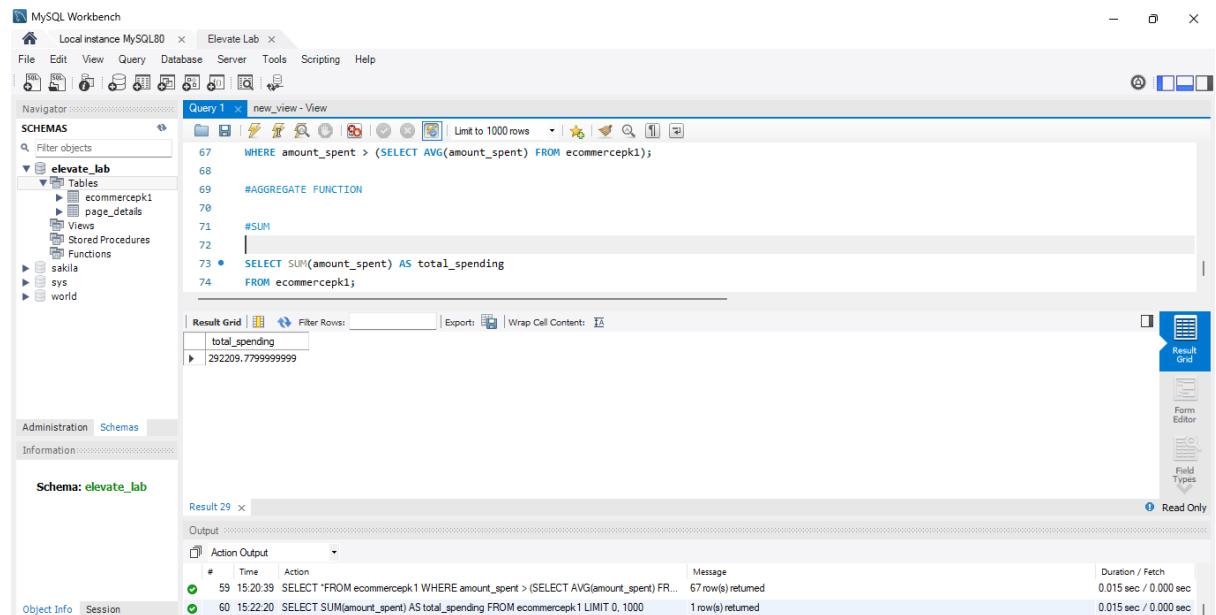
The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The schema is set to "elevate_lab". The query is:

```
60 FROM ecommercepk1 a
61 RIGHT JOIN page_details p
62 ON a.page_name = p.page_name;
63
64 #SUB QUERY
65
66 • SELECT *FROM ecommercepk1
67 WHERE amount_spent > (SELECT AVG(amount_spent) FROM ecommercepk1);
```

The result grid shows the following data:

page_name	day	amount_spent	reporting_starts	reporting_ends
censored	09-06-2025	2807.17	09-06-2025	09-06-2025
censored	07-06-2025	2392.81	07-06-2025	07-06-2025
censored	29-05-2025	2329.9	29-05-2025	29-05-2025
censored	16-04-2025	2885.34	16-04-2025	16-04-2025
censored	15-04-2025	2991.05	15-04-2025	15-04-2025
censored	14-04-2025	2761.95	14-04-2025	14-04-2025
censored	13-04-2025	2917.59	13-04-2025	13-04-2025
censored	12-04-2025	2621.64	12-04-2025	12-04-2025
censored	11-04-2025	2185.25	11-04-2025	11-04-2025

Step 8 : Use Aggregate Function(sum) :



The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The schema is set to "elevate_lab". The query is:

```
67 WHERE amount_spent > (SELECT AVG(amount_spent) FROM ecommercepk1);
68
69 #AGGREGATE FUNCTION
70
71 #SUM
72
73 • SELECT SUM(amount_spent) AS total_spending
74 FROM ecommercepk1;
```

The result grid shows the following data:

total_spending
292209.7799999999

The output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
59	15:20:39	SELECT *FROM ecommercepk1 WHERE amount_spent > (SELECT AVG(amount_spent) FR...	67 row(s) returned	0.015 sec / 0.000 sec
60	15:22:20	SELECT SUM(amount_spent) AS total_spending FROM ecommercepk1 LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

Step 8.1 : Use Aggregate Function(Average) :

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `elevate_lab` with tables `ecommercepk1` and `page_details`.
- Query Editor (Query 1):** Contains the following SQL code:

```
72
73 •  SELECT SUM(amount_spent) AS total_spending
74   FROM ecommercepk1;
75
76 #AVERAGE
77
78 •  SELECT AVG(amount_spent) AS avg_spending
79   FROM ecommercepk1;
```
- Result Grid:** Displays the result of the second query:

avg_spending
2164.516888888884
- Output Panel (Result 30):** Shows the execution log:

Action	Time	Message	Duration / Fetch
60	15:22:20	SELECT SUM(amount_spent) AS total_spending FROM ecommercepk1 LIMIT 0, 1000	1row(s) returned 0.015 sec / 0.000 sec
61	15:24:39	SELECT AVG(amount_spent) AS avg_spending FROM ecommercepk1 LIMIT 0, 1000	1row(s) returned 0.015 sec / 0.000 sec
- Status Bar:** Shows "Query Completed".

Step 8.2 : Use Aggregate Function(Max,Min) :

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `elevate_lab` with tables `ecommercepk1` and `page_details`.
- Query Editor (Query 1):** Contains the following SQL code:

```
77
78 •  SELECT AVG(amount_spent) AS avg_spending
79   FROM ecommercepk1;
80
81 #MAX MIN
82
83 •  SELECT MAX(amount_spent), MIN(amount_spent)
84   FROM ecommercepk1;
```
- Result Grid:** Displays the results of the third query:

MAX(amount_spent)	MIN(amount_spent)
5410.61	78.4
- Output Panel (Result 31):** Shows the execution log:

Action	Time	Message	Duration / Fetch
61	15:24:39	SELECT AVG(amount_spent) AS avg_spending FROM ecommercepk1 LIMIT 0, 1000	1row(s) returned 0.015 sec / 0.000 sec
62	15:26:39	SELECT MAX(amount_spent), MIN(amount_spent) FROM ecommercepk1 LIMIT 0, 1000	1row(s) returned 0.016 sec / 0.000 sec
- Status Bar:** Shows "Query Completed".

Step 9 : Create View :

The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1 - new_view - View'. The code entered is:

```
#CREATE VIEW FOR analyze
CREATE VIEW high_spend_days AS
SELECT page_name, day, amount_spent
FROM ecommercepk1
WHERE amount_spent > 2000;
SELECT * FROM high_spend_days;
```

The 'Result Grid' shows the results of the SELECT query:

page_name	day	amount_spent
censored	09-06-2025	2807.17
censored	08-06-2025	2051.74
censored	07-06-2025	2392.81
censored	30-05-2025	2025.59
censored	29-05-2025	2329.9
censored	18-04-2025	2148.22
censored	16-04-2025	2885.34
censored	15-04-2025	2991.05
censored	14-04-2025	2761.95
censored	13-04-2025	2917.59

The 'Object Info' tab shows the history of actions:

#	Time	Action	Message	Duration / Fetch
63	15:28:28	CREATE VIEW high_spend_days AS SELECT page_name, day, amount_spent FROM ecommercepk1 WHERE amount_spent > 2000;	0 row(s) affected	0.063 sec
64	15:28:59	SELECT * FROM high_spend_days LIMIT 0, 1000	71 row(s) returned	0.015 sec / 0.000 sec

Step 10 : Index Analysis:

The screenshot shows the MySQL Workbench interface with a query editor window titled 'Query 1 - new_view - View'. The code entered is:

```
WHERE amount_spent > 2000;
SELECT * FROM high_spend_days;
#CREATE INDEX FOR OPTIMIZATION
CREATE INDEX idx_day ON ecommercepk1 (day(10));
CREATE INDEX idx_amount ON ecommercepk1 (amount_spent );
SHOW INDEXES FROM ecommercepk1;
```

The 'Result Grid' shows the index information:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
ecommercepk1	1	idx_day	1	day	A	135	10	NULL	YES	BTREE		YES	NULL	
ecommercepk1	1	idx_amount	1	amount_spent	A	135	NULL	NULL	YES	BTREE		YES	NULL	

The 'Object Info' tab shows the history of actions:

#	Time	Action	Message	Duration
68	15:35:09	CREATE INDEX idx_amount ON ecommercepk1(amount_spent)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.1
69	15:36:19	SHOW INDEXES FROM ecommercepk1	2 row(s) returned	0.0