Experiment 4: Aggregate Functions, Group By and Having Clause

AIM

To study and implement aggregate functions, GROUP BY, and HAVING clause with suitable examples.

THEORY

Aggregate Functions

These perform calculations on a set of values and return a single value.

- **MIN()** – Smallest value
- **MAX()** – Largest value
- **COUNT()** – Number of rows
- **SUM()** – Total of values
- **AVG()** – Average of values

Syntax:

```
SELECT AGG_FUNC(column_name) FROM table_name WHERE condition;
```

GROUP BY

Groups records with the same values in specified columns. **Syntax:**

```
SELECT column_name, AGG_FUNC(column_name)
FROM table_name
```

```
GROUP BY column_name;
```

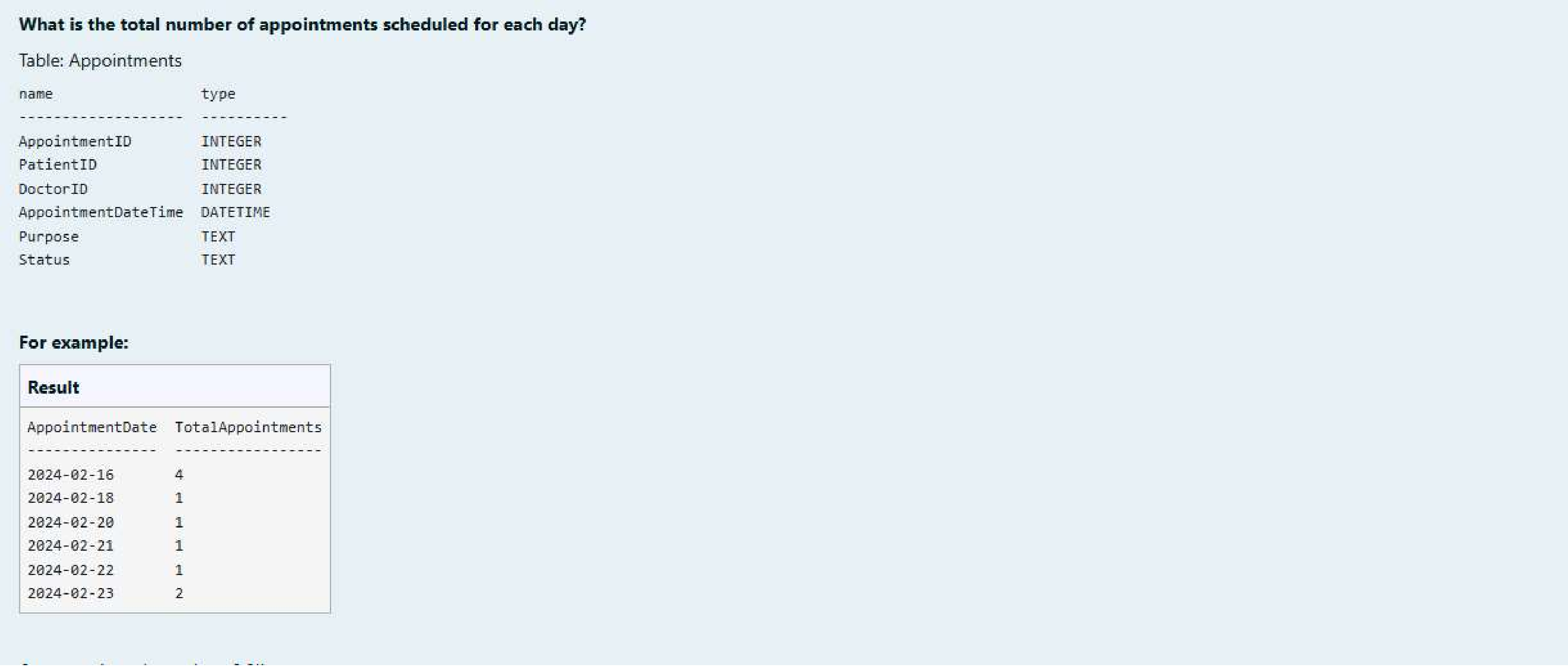
HAVING

Filters the grouped records based on aggregate conditions. **Syntax:**

```
SELECT column_name, AGG_FUNC(column_name)
FROM table_name
GROUP BY column_name
HAVING condition;
```

Question 1

--



What is the total number of appointments scheduled for each day?

Table: Appointments

```
name                type
------------------  ----------
AppointmentID       INTEGER
PatientID           INTEGER
DoctorID            INTEGER
AppointmentDateTime DATETIME
Purpose             TEXT
Status              TEXT
```

For example:

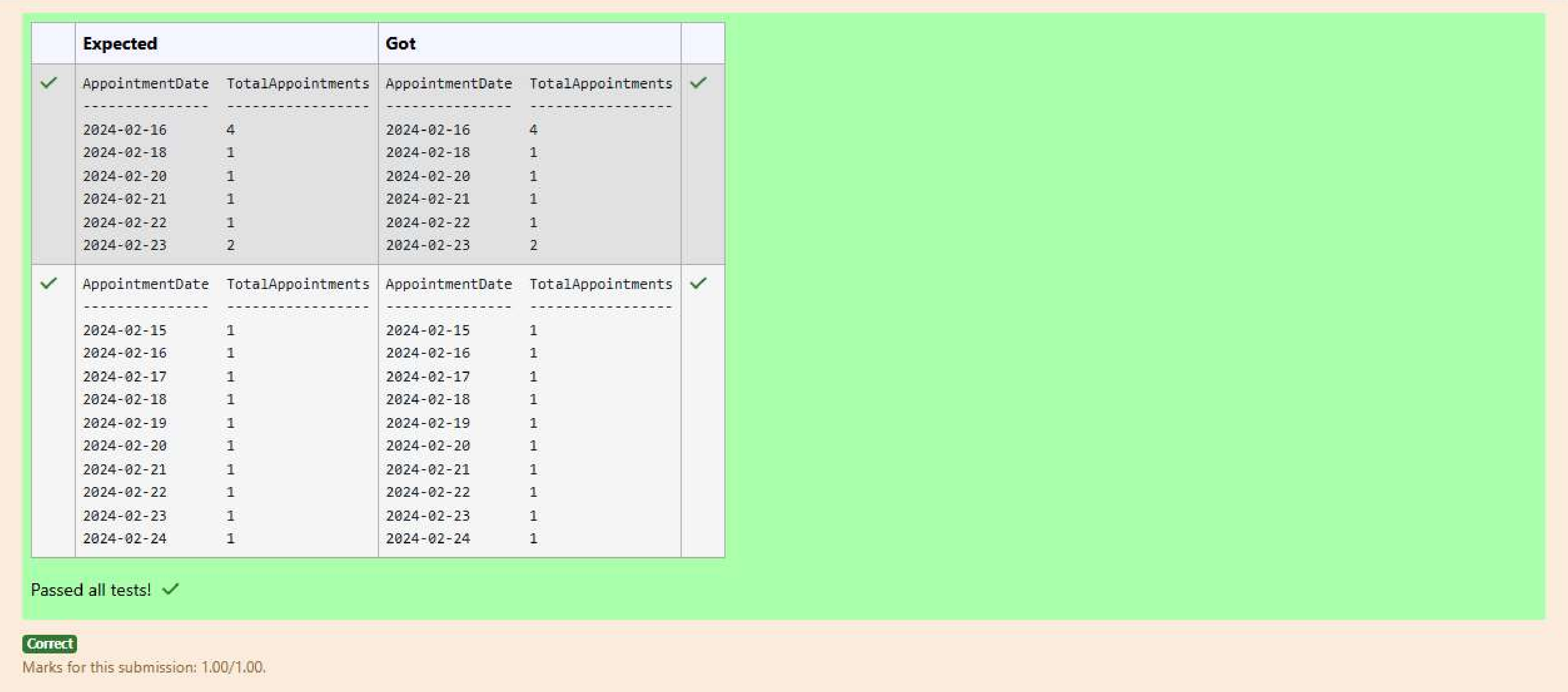
Result

```
AppointmentDate  TotalAppointments
---------------  -----------------
2024-02-16       4
2024-02-18       1
2024-02-20       1
2024-02-21       1
2024-02-22       1
2024-02-23       2
```

```
-- select date(AppointmentDateTime) as AppointmentDate, count(*) as  TotalAppointments

from Appointments
group by AppointmentDate;
```

Output:



	Expected	Got	
✓	AppointmentDate TotalAppointments --------------- ----------------- 2024-02-16 4 2024-02-18 1 2024-02-20 1 2024-02-21 1 2024-02-22 1 2024-02-23 2	AppointmentDate TotalAppointments --------------- ----------------- 2024-02-16 4 2024-02-18 1 2024-02-20 1 2024-02-21 1 2024-02-22 1 2024-02-23 2	✓
✓	AppointmentDate TotalAppointments --------------- ----------------- 2024-02-15 1 2024-02-16 1 2024-02-17 1 2024-02-18 1 2024-02-19 1 2024-02-20 1 2024-02-21 1 2024-02-22 1 2024-02-23 1 2024-02-24 1	AppointmentDate TotalAppointments --------------- ----------------- 2024-02-15 1 2024-02-16 1 2024-02-17 1 2024-02-18 1 2024-02-19 1 2024-02-20 1 2024-02-21 1 2024-02-22 1 2024-02-23 1 2024-02-24 1	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 2

--

Write a SQL query to find the shortest email address in the customer table?

Table: customer

```
name        type
----------  ----------
id          INTEGER
name        TEXT
city        TEXT
email       TEXT
phone       INTEGER
```

For example:

Result
name   email   min_email_length ---------- -------------- ---------------- Ravi Kumar ravi@gmail.com 14

```
-- select name,email,length(email) as  min_email_length
from customer
order by length(email)
limit 1;
```

Output:

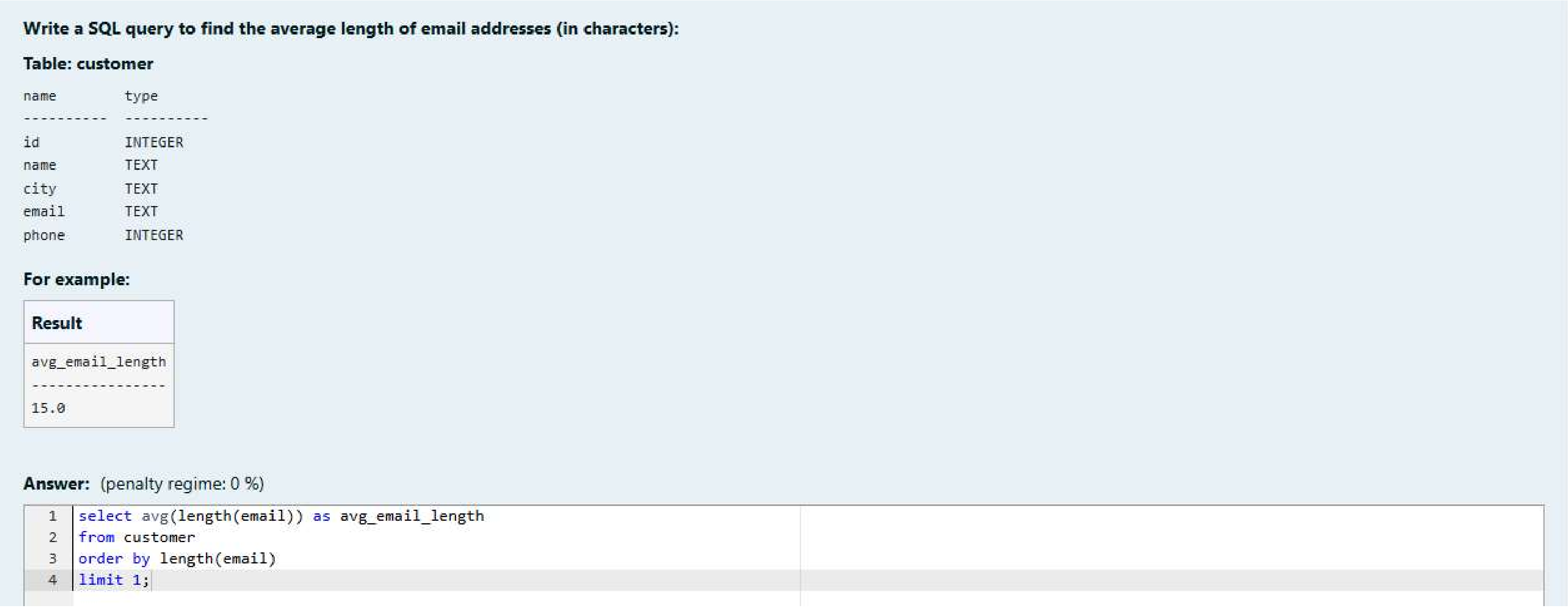
	Expected	Got	
✓	name email min_email_length ---------- -------------- ---------------- Ravi Kumar ravi@gmail.com 14	name email min_email_length ---------- -------------- ---------------- Ravi Kumar ravi@gmail.com 14	✓
✓	name email min_email_length --------------- ------------ ---------------- Esran Anton Roy roy@yahoo.in 12	name email min_email_length --------------- ------------ ---------------- Esran Anton Roy roy@yahoo.in 12	✓

Passed all tests! ✓

Correct

Question 3

--



Write a SQL query to find the average length of email addresses (in characters):

Table: customer

```
name        type
----------  ----------
id          INTEGER
name        TEXT
city        TEXT
email       TEXT
phone       INTEGER
```

For example:

Result
avg_email_length ---------------- 15.0

Answer: (penalty regime: 0 %)

```
select avg(length(email)) as avg_email_length
from customer
order by length(email)
limit 1;
```

```
-- select avg(length(email)) as avg_email_length
from customer
order by length(email)
limit 1;
```

Output:

	Expected	Got	
✓	avg_email_length ---------------- 15.0	avg_email_length ---------------- 15.0	✓
✓	avg_email_length ---------------- 15.4545454545455	avg_email_length ---------------- 15.4545454545455	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 4

--

Write a SQL query to calculate total purchase amount of all orders. Return total purchase amount.

Sample table: orders

```
ord_no     purch_amt   ord_date    customer_id  salesman_id
---------- ---------- ---------- ----------- -----------
70001       150.5       2012-10-05  3005        5002
70009       270.65      2012-09-10  3001        5005
70002       65.26       2012-10-05  3002        5001
```

For example:

Result
TOTAL ---------- 17541.18

```
-- select sum(purch_amt) as TOTAL
from orders
```

Output:

	Expected	Got	
✓	TOTAL ---------- 17541.18	TOTAL ---------- 17541.18	✓
✓	TOTAL ---------- 17841.18	TOTAL ---------- 17841.18	✓

Passed all tests! ✓

Correct

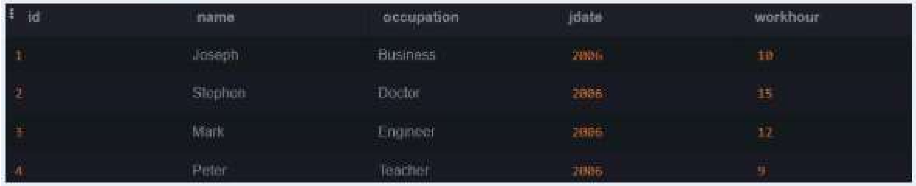
Marks for this submission: 1.00/1.00.

Question 5

--

Write the SQL query that accomplishes the grouping of data by joining date (jdate), calculates the minimum work hours for each date, and excludes dates where the minimum work hour is not less than 10.

Sample table: employee1



id	name	occupation	jdate	workhour
1	Joseph	Business	2006	10
2	Stephon	Doctor	2006	15
3	Mark	Engineer	2006	12
4	Peter	Teacher	2006	9

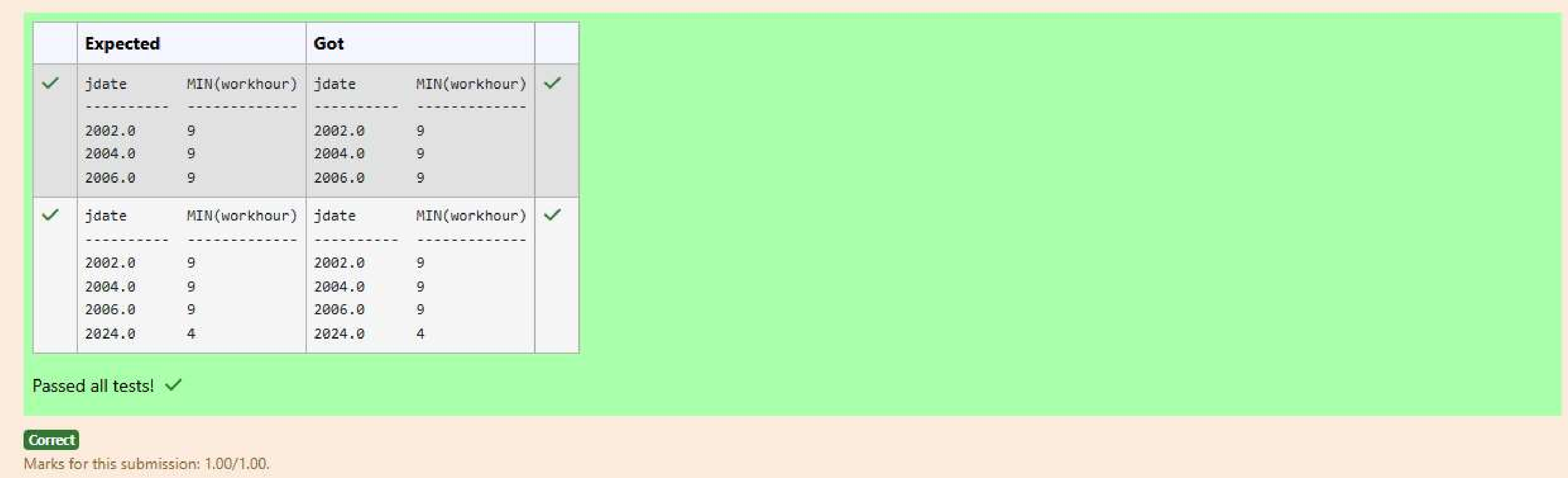
For example:

Result

```
jdate       MIN(workhour)
----------  -------------
2002.0      9
2004.0      9
2006.0      9
```

```
-- select jdate,MIN(workhour)
from employee1
group by jdate
having MIN(workhour) <10 ;
```

Output:



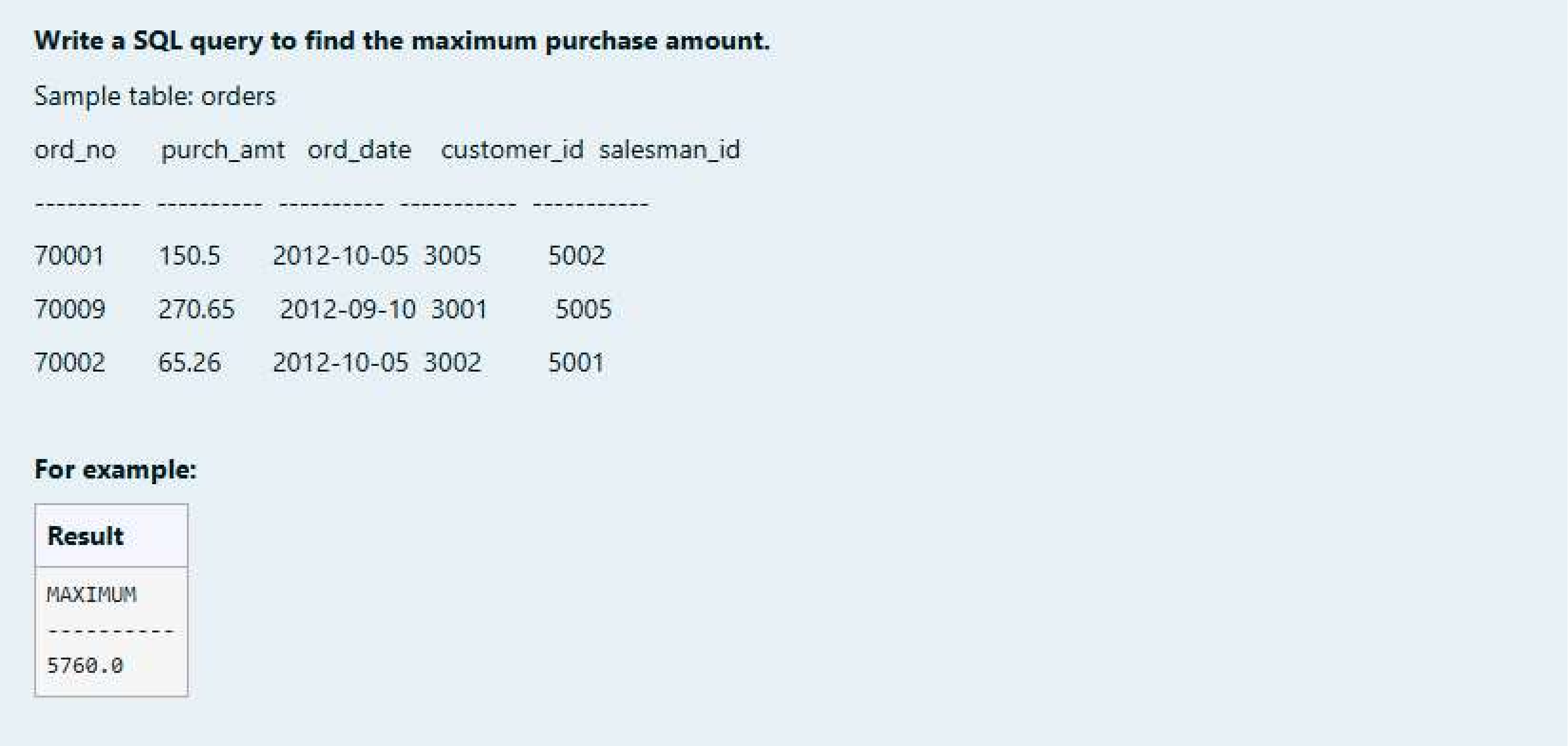
	Expected	Got	
✓	jdate MIN(workhour) ---------- ------------- 2002.0 9 2004.0 9 2006.0 9	jdate MIN(workhour) ---------- ------------- 2002.0 9 2004.0 9 2006.0 9	✓
✓	jdate MIN(workhour) ---------- ------------- 2002.0 9 2004.0 9 2006.0 9 2024.0 4	jdate MIN(workhour) ---------- ------------- 2002.0 9 2004.0 9 2006.0 9 2024.0 4	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 6



Write a SQL query to find the maximum purchase amount.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001

For example:

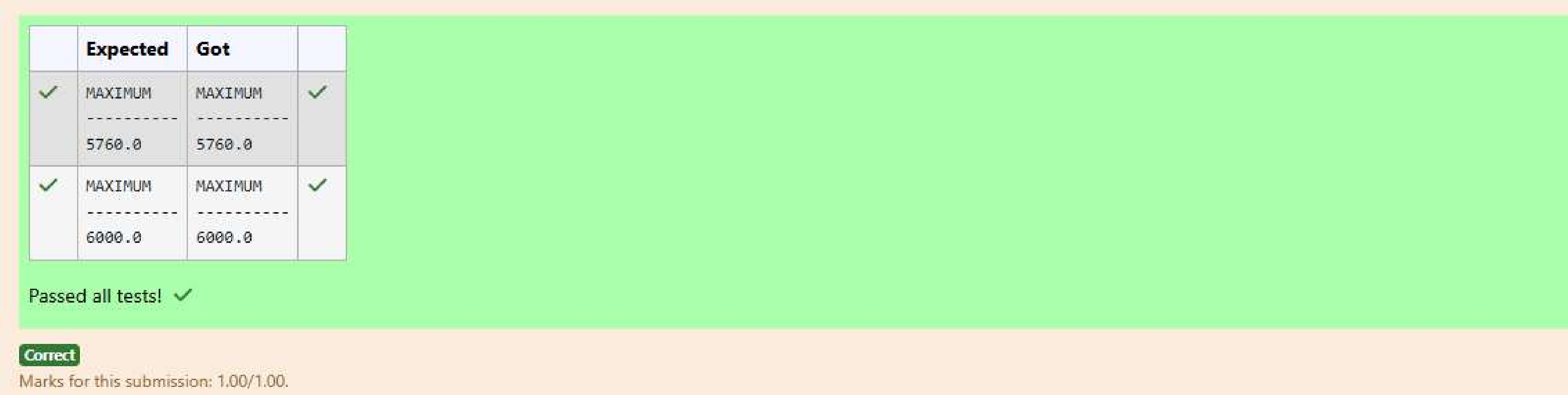
Result

```
MAXIMUM
----------
5760.0
```

--

```sql
-- select max(purch_amt) as MAXIMUM
from orders
order by purch_amt;
```

Output:



	Expected	Got	
✓	MAXIMUM ---------- 5760.0	MAXIMUM ---------- 5760.0	✓
✓	MAXIMUM ---------- 6000.0	MAXIMUM ---------- 6000.0	✓

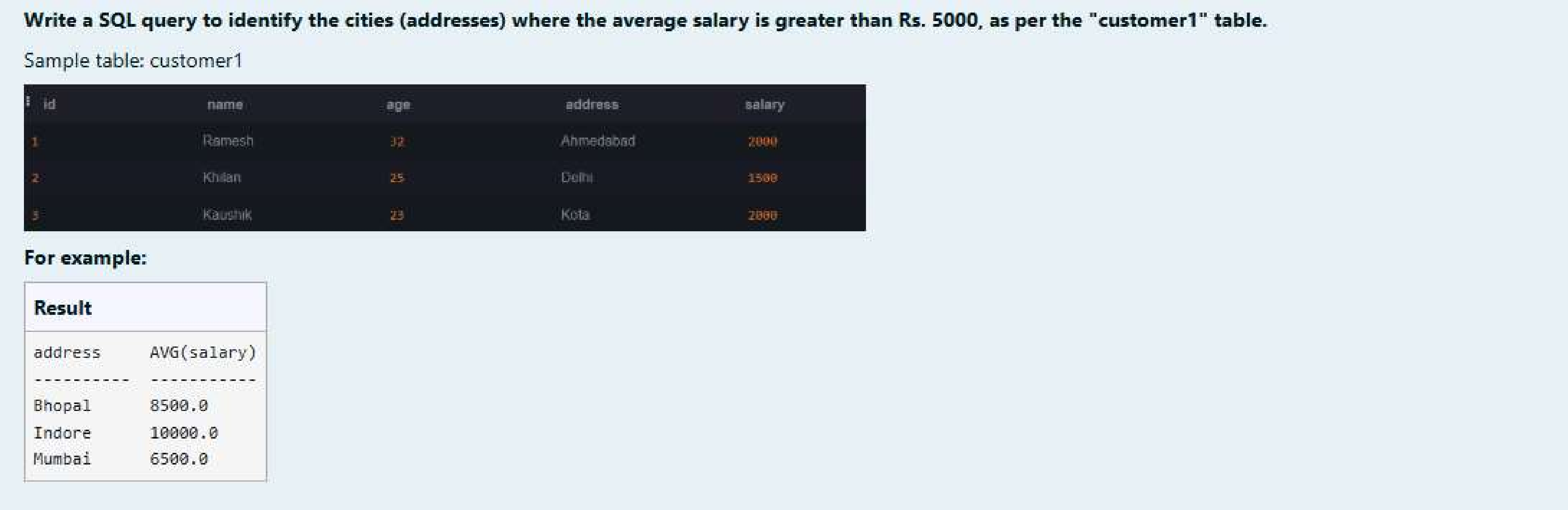
Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 7

--



Write a SQL query to identify the cities (addresses) where the average salary is greater than Rs. 5000, as per the "customer1" table.

Sample table: customer1

id	name	age	address	salary
1	Ramesh	32	Ahmedabad	2000
2	Khilan	25	Delhi	1500
3	Kaushik	23	Kota	2000

For example:

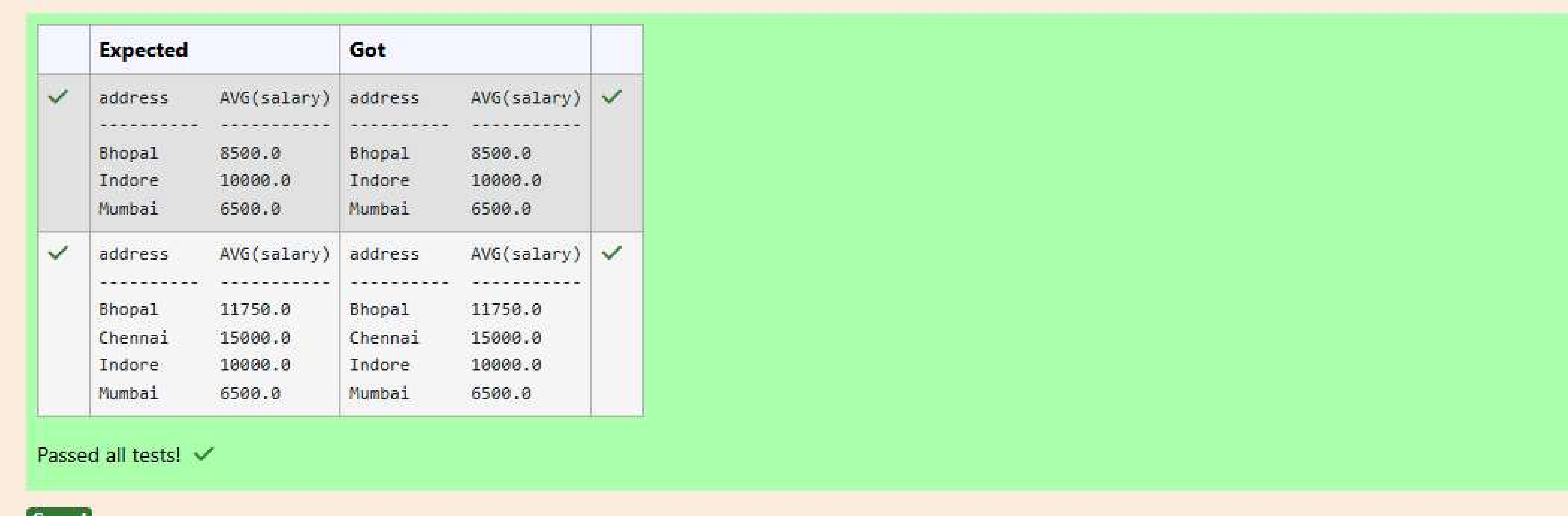
Result

```
address     AVG(salary)
----------  -----------
Bhopal      8500.0
Indore      10000.0
Mumbai      6500.0
```

```
-- select address,AVG(salary)
from customer1
```

```
group by address
having avg(salary) > 5000
```

Output:

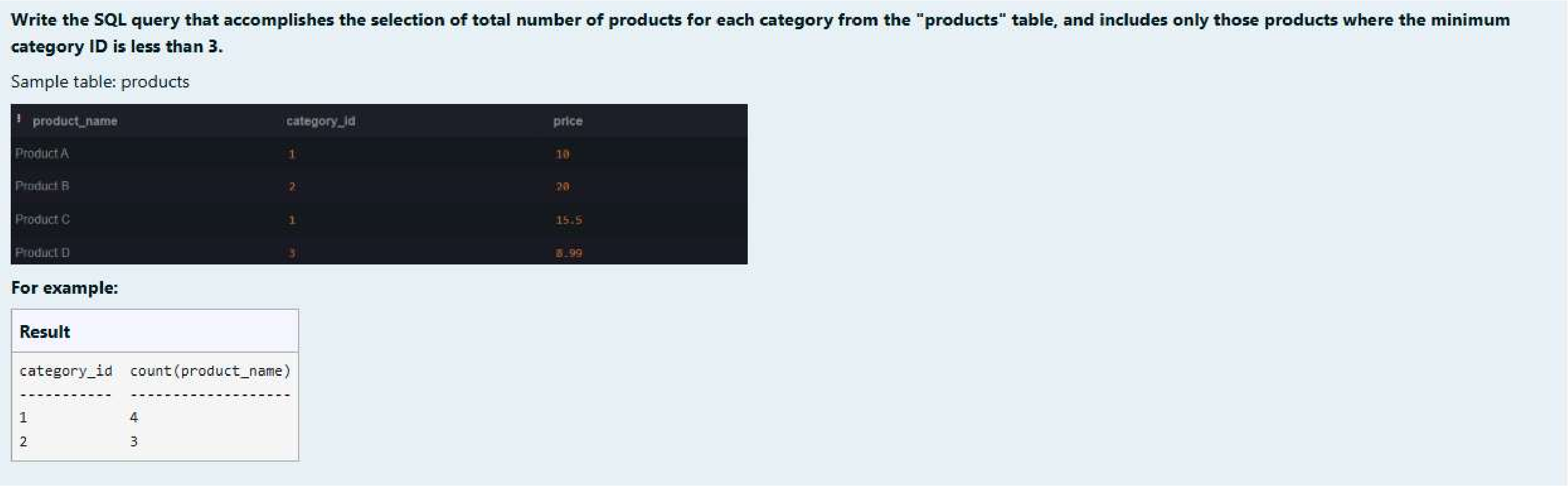


	Expected	Got	
✓	address AVG(salary) ---------- ----------- Bhopal 8500.0 Indore 10000.0 Mumbai 6500.0	address AVG(salary) ---------- ----------- Bhopal 8500.0 Indore 10000.0 Mumbai 6500.0	✓
✓	address AVG(salary) ---------- ----------- Bhopal 11750.0 Chennai 15000.0 Indore 10000.0 Mumbai 6500.0	address AVG(salary) ---------- ----------- Bhopal 11750.0 Chennai 15000.0 Indore 10000.0 Mumbai 6500.0	✓

Passed all tests! ✓

Question 8

--



Write the SQL query that accomplishes the selection of total number of products for each category from the "products" table, and includes only those products where the minimum category ID is less than 3.

Sample table: products

! product_name	category_id	price
Product A	1	10
Product B	2	20
Product C	1	15.5
Product D	3	8.99

For example:

Result

```
category_id  count(product_name)
-----------  -------------------
1            4
2            3
```

```
-- select category_id,count(product_name)
from products
group by category_id
having min(category_id) <3
```

Output:

	Expected	Got	
✓	category_id count(product_name) ----------- ------------------- 1 4 2 3	category_id count(product_name) ----------- ------------------- 1 4 2 3	✓
✓	category_id count(product_name) ----------- ------------------- 1 6 2 3	category_id count(product_name) ----------- ------------------- 1 6 2 3	✓

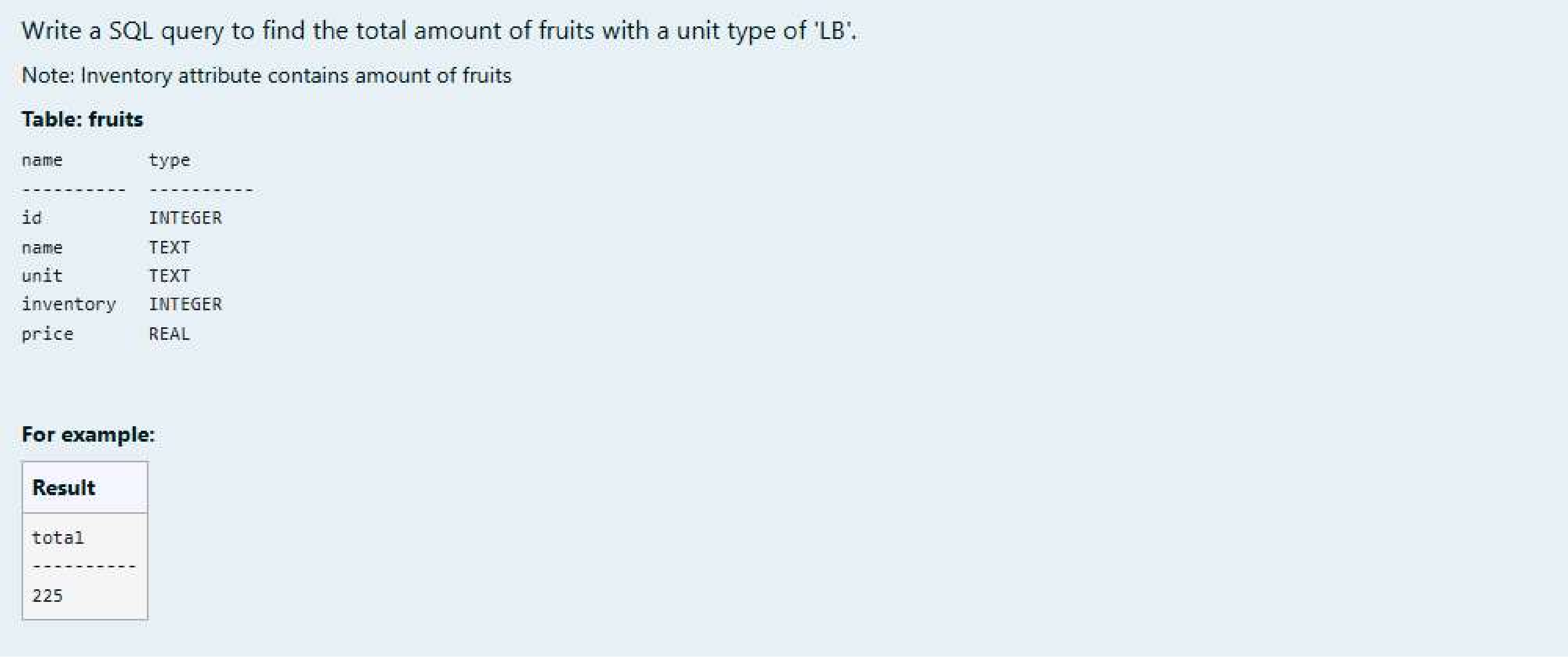
Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 9

--



Write a SQL query to find the total amount of fruits with a unit type of 'LB'.

Note: Inventory attribute contains amount of fruits

Table: fruits

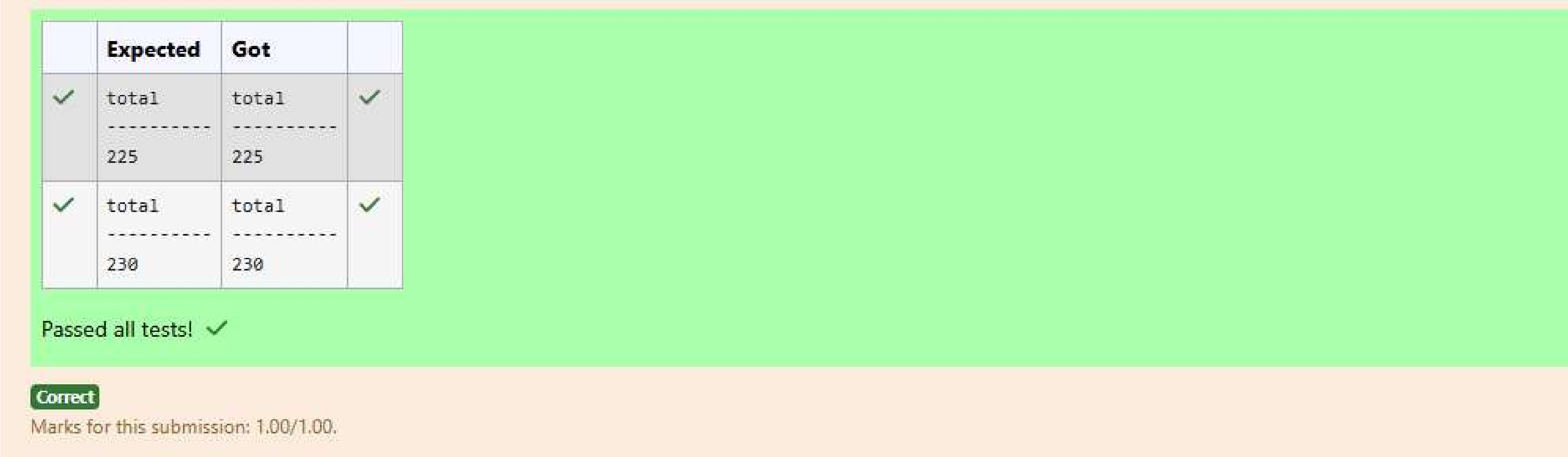
```
name        type
----------  ----------
id          INTEGER
name        TEXT
unit        TEXT
inventory   INTEGER
price       REAL
```

For example:

Result
total ---------- 225

```
-- select sum(inventory) as total
from fruits
where unit ='LB'
```

Output:



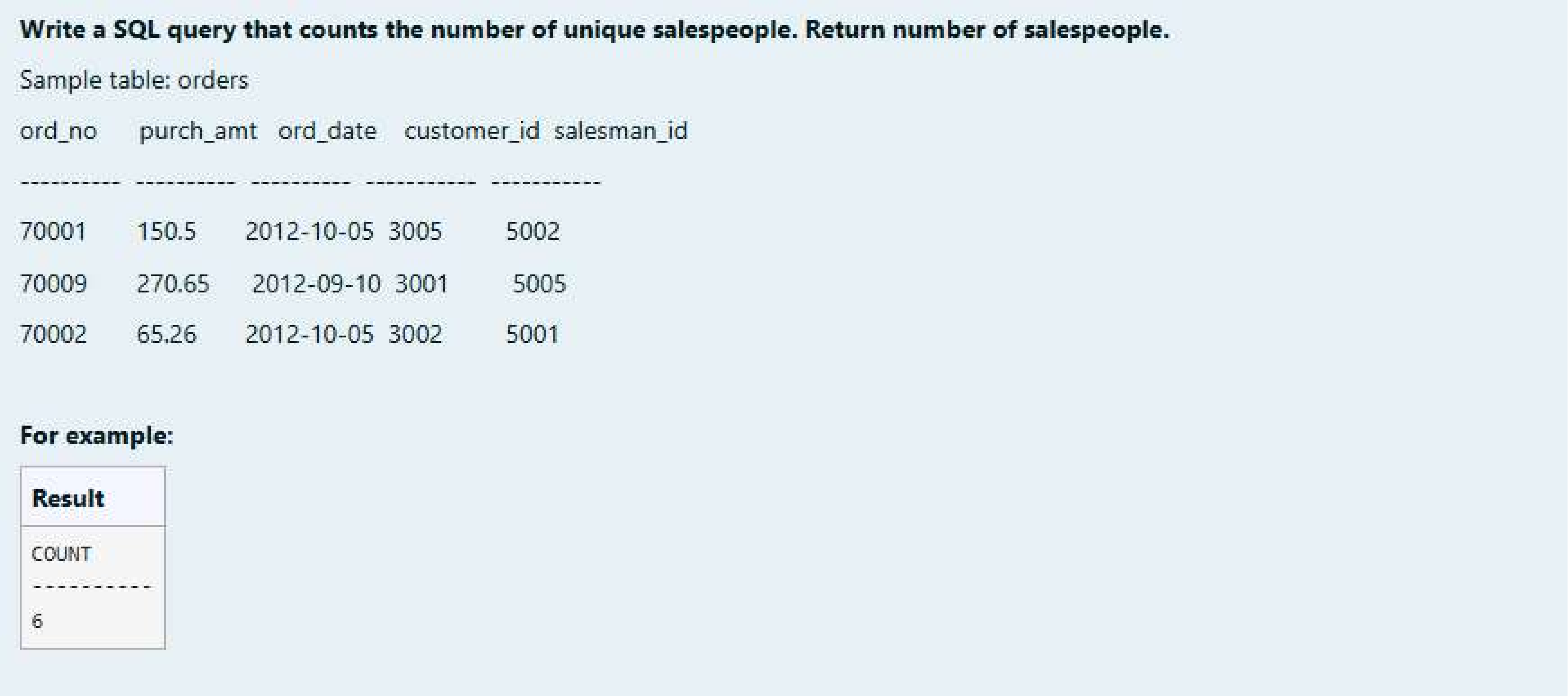
	Expected	Got	
✓	total ---------- 225	total ---------- 225	✓
✓	total ---------- 230	total ---------- 230	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 10



Write a SQL query that counts the number of unique salespeople. Return number of salespeople.

Sample table: orders

```
ord_no     purch_amt   ord_date    customer_id  salesman_id
---------- ----------  ----------  -----------  -----------
70001       150.5       2012-10-05  3005         5002
70009       270.65      2012-09-10  3001         5005
70002       65.26       2012-10-05  3002         5001
```

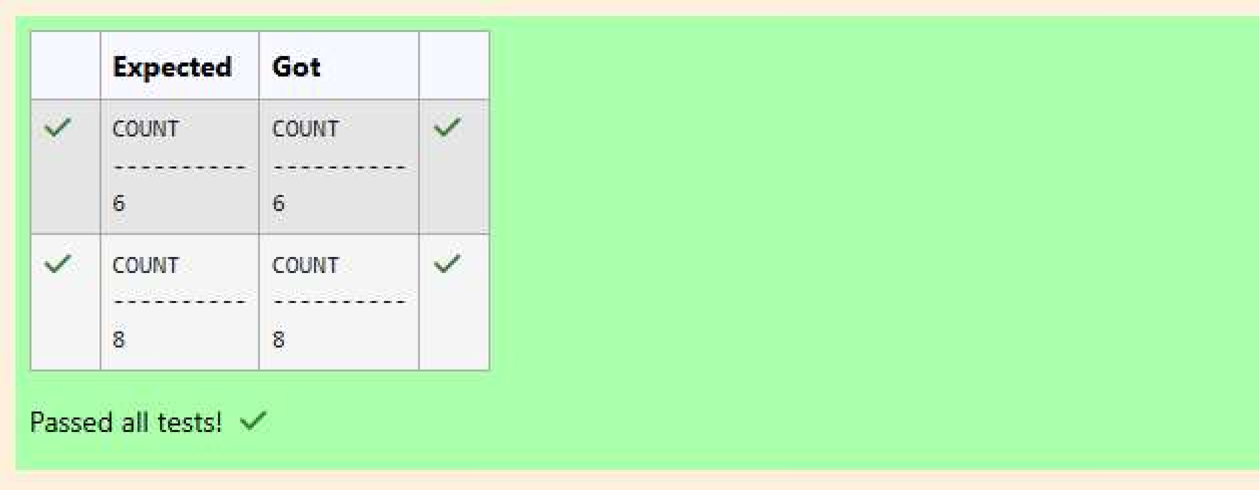
For example:

Result
COUNT ---------- 6

--

```
-- select count(distinct salesman_id) as COUNT
from orders
```

Output:



	Expected	Got	
✓	COUNT ---------- 6	COUNT ---------- 6	✓
✓	COUNT ---------- 8	COUNT ---------- 8	✓

Passed all tests! ✓

RESULT

Thus, the SQL queries to implement aggregate functions, GROUP BY, and HAVING clause have been executed successfully.