

SQL Assignment

Name - Rohan Das

Employee Id- 2629311

Assessment report presented by Considering the below Tables:

employee table:

	Field	Type	Null	Key	Default	Extra
►	emp_id	int	YES		NULL	
	emp_name	varchar(50)	YES		NULL	
	emp_sal	double	YES		NULL	

customer table:

	Field	Type	Null	Key	Default	Extra
►	id	int	NO	PRI	NULL	
	name	varchar(32)	YES		NULL	
	age	int	YES		NULL	
	city	varchar(50)	YES		NULL	

product table:

	Field	Type	Null	Key	Default	Extra
►	id	int	NO		NULL	
	name	varchar(32)	YES		NULL	
	price	double	YES		NULL	
	quantity	int	YES		NULL	
	cust_id	int	YES		NULL	

The customer and product table are being used for showing join operation while the employee table is for showing max, min functions.

Q1. Subquery to find max and min salary

Minimum:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

No object selected

Result Grid

Filter Rows

Export

Wrap Cell Contents

employee 15 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
9	16:39:42	select * from employee LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
10	16:40:11	select emp_sal as minimum_salary from employee where emp_sal = (select min(emp_sal) from employee) LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

31°C Mostly sunny 4:41 PM 2/24/2025

Maximum:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

No object selected

Result Grid

Filter Rows

Export

Wrap Cell Contents

employee 15 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	16:40:11	select emp_sal as minimum_salary from employee where emp_sal = (select min(emp_sal) from employee) LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
11	16:42:33	select emp_sal as maximum_salary from employee where emp_sal = (select max(emp_sal) from employee) LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

31°C Mostly sunny 4:42 PM 2/24/2025

Q2. Find the second highest salary using subquery or limit.

Subquery:

The screenshot shows the MySQL Workbench interface with a script editor containing the following SQL code:

```
27 emp_name varchar(10),
28 emp_sal double);
29
30 insert into employee
31 values
32 (1001, 'Raghu', 50000),
33 (1002, 'Ram', 35000),
34 (1003, 'Rajan', 60000),
35 (1004, 'Nitin', 85000),
36 (1005, 'Nisha', 80000),
37 (1006, 'Abhinav', 40000),
38 (1007, 'Rohan', 25000);
39
40 desc employee;
41
42 select * from employee;
43
44 select emp_sal as minimum_salary
45 from employee
46 where emp_sal = (select min(emp_sal) from employee);
47
48 select emp_sal as maximum_salary
49 from employee
50 where emp_sal = (select max(emp_sal) from employee);
51
52 select max(emp_sal) as second_highest_salary from employee where emp_sal < (select max(emp_sal) from employee);
```

The output pane shows the result of the last query, which is a single row with the value 80000. The status bar at the bottom indicates that the query was executed successfully, returning 1 row(s) in 0.000 sec.

Limit:

The screenshot shows the MySQL Workbench interface with a script editor containing the following SQL code:

```
33 (1002, 'Ram', 35000),
34 (1003, 'Rajan', 60000),
35 (1004, 'Nitin', 85000),
36 (1005, 'Nisha', 80000),
37 (1006, 'Abhinav', 40000),
38 (1007, 'Rohan', 25000);
39
40 desc employee;
41
42 select * from employee;
43
44 select emp_sal as minimum_salary
45 from employee
46 where emp_sal = (select min(emp_sal) from employee);
47
48 select emp_sal as maximum_salary
49 from employee
50 where emp_sal = (select max(emp_sal) from employee);
51
52 select max(emp_sal) as second_highest_salary from employee where emp_sal < (select max(emp_sal) from employee);
53
54 select emp_sal as second_highest_salary from employee order by emp_sal desc limit 1, 1;
```

The output pane shows the result of the last query, which is a single row with the value 80000. The status bar at the bottom indicates that the query was executed successfully, returning 1 row(s) in 0.000 sec.

Q3. Write a query for implementing all the joins with order or group by.

Inner Join with order by:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
quantity INT,  
cust_id INT);  
insert into customer values  
(1, 'Rohan', 24, 'Kolkata'),  
(2, 'Aman', 23, 'Patna'),  
(3, 'Prakash', 22, 'Behala'),  
(4, 'Tathagata', 23, 'Bankura'),  
(5, 'Abir', 22, 'Kol'),  
(6, 'John', 35, 'ABC');  
INSERT INTO product values  
(1001, 'Rice', 60, 2, 1),  
(1002, 'Dal', 90, 1, 2),  
(1003, 'Besan', 30, 3, 3),  
(1004, 'Pen', 350, 5, 4),  
(1005, 'Water', 50, 1, 5),  
(1006, 'Water', 50, 1, 1),  
(1007, 'Pen', 350, 5, NULL),  
(1008, 'newitem', 450, 10, NULL);  
  
select *  
from customer as c inner join product as p  
on c.id = p.cust_id  
order by p.price*p.quantity desc;
```

The Results grid shows the output of the inner join query:

	id	name	age	city	id	name	price	quantity	cust_id
1	4	Tathagata	23	Bankura	1004	Pen	350	5	4
2	1	Rohan	24	Kolkata	1001	Rice	60	2	1
3	2	Aman	23	Patna	1002	Dal	90	1	2
4	3	Prakash	22	Behala	1003	Besan	30	3	3
5	5	Abir	22	Kol	1005	Water	50	1	5
6	1	Rohan	24	Kolkata	1006	Water	50	1	1

The Output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
29	17:06.04	select * from customer as c inner join product as p on c.id = p.cust_id order by p.price*p.quantity desc LIMIT 0.1000	6 row(s) returned	0.000 sec / 0.000 sec
30	17:06.46	select * from customer as c inner join product as p on c.id = p.cust_id order by p.price*p.quantity desc LIMIT 0.1000	6 row(s) returned	0.000 sec / 0.000 sec

Left join:

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
(2, 'Aman', 23, 'Patna'),  
(3, 'Prakash', 22, 'Behala'),  
(4, 'Tathagata', 23, 'Bankura'),  
(5, 'Abir', 22, 'Kol'),  
(6, 'John', 35, 'ABC');  
INSERT INTO product values  
(1001, 'Rice', 60, 2, 1),  
(1002, 'Dal', 90, 1, 2),  
(1003, 'Besan', 30, 3, 3),  
(1004, 'Pen', 350, 5, 4),  
(1005, 'Water', 50, 1, 5),  
(1006, 'Water', 50, 1, 1),  
(1007, 'Pen', 350, 5, NULL),  
(1008, 'newitem', 450, 10, NULL);  
  
select *  
from customer as c inner join product as p  
on c.id = p.cust_id  
order by p.price*p.quantity desc;  
  
select *  
from customer as c left join product as p  
on c.id = p.cust_id;
```

The Results grid shows the output of the left join query:

	id	name	age	city	id	name	price	quantity	cust_id
1	1	Rohan	24	Kolkata	1006	Water	50	1	1
2	1	Aman	23	Patna	1002	Dal	90	1	2
3	3	Prakash	22	Behala	1003	Besan	30	3	3
4	4	Tathagata	23	Bankura	1004	Pen	350	5	4
5	5	Abir	22	Kol	1005	Water	50	1	5
6	6	John	35	ABC					

The Output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
35	17:16.40	select * from customer as c right join product as p on c.id = p.cust_id LIMIT 0.1000	6 row(s) returned	0.000 sec / 0.000 sec
36	17:17.06	select * from customer as c left join product as p on c.id = p.cust_id LIMIT 0.1000	7 row(s) returned	0.000 sec / 0.000 sec

Right Join:

The screenshot shows MySQL Workbench with a query window containing the following SQL code:

```
18 (6, 'John', 35, 'ABC');
19 * INSERT INTO product values
20 (1001, 'Rice', 60, 2, 1),
21 (1002, 'Dal', 90, 1, 2),
22 (1003, 'Besan', 30, 3, 3),
23 (1004, 'Pen', 350, 5, 4),
24 (1005, 'Water', 50, 1, 5),
25 (1006, 'Water', 50, 1, 1),
26 (1007, 'Pen', 350, 5, NULL),
27 (1008, 'newitem', 450, 10, NULL);
28
29 * select *
30 from customer as c inner join product as p
31 on c.id = p.cust_id
32 order by p.price*p.quantity desc;
33
34 * select *
35 from customer as c left join product as p
36 on c.id = p.cust_id;
37
38 * select *
39 from customer as c right join product as p
40 on c.id = p.cust_id;
```

The Results grid shows the output of the Right Join query, displaying 10 rows. The first 5 rows are from the 'customer' table, and the last 5 rows are from the 'product' table. The 'id' column is highlighted in blue.

id	name	age	city	id	name	price	quantity	cust_id
1	Rohan	24	Kolkata	1001	Rice	60	2	1
2	Anan	23	Patna	1002	Dal	90	1	2
3	Prakash	22	Behala	1003	Besan	30	3	3
4	Tatpaga	23	Bankura	1004	Pen	350	5	4
5	Abr	22	Kol	1005	Water	50	1	5
6	Rohan	24	Kolkata	1006	Water	50	1	1
7	John	35	ABC	1007	Pen	350	5	
8				1008	newitem	450	10	
9								
10								

The Output pane shows the execution details of the query, including the time taken (17:17:43) and the number of rows returned (10 rows).

Full Join:

The screenshot shows MySQL Workbench with a query window containing the following SQL code:

```
29 * select *
30 from customer as c inner join product as p
31 on c.id = p.cust_id
32 order by p.price*p.quantity desc;
33
34 * select *
35 from customer as c left join product as p
36 on c.id = p.cust_id;
37
38 * select *
39 from customer as c right join product as p
40 on c.id = p.cust_id;
41
42 * select *
43 from customer as c left join product as p
44 on c.id = p.cust_id
45 union
46 select *
47 from customer as c right join product as p
48 on c.id = p.cust_id;
```

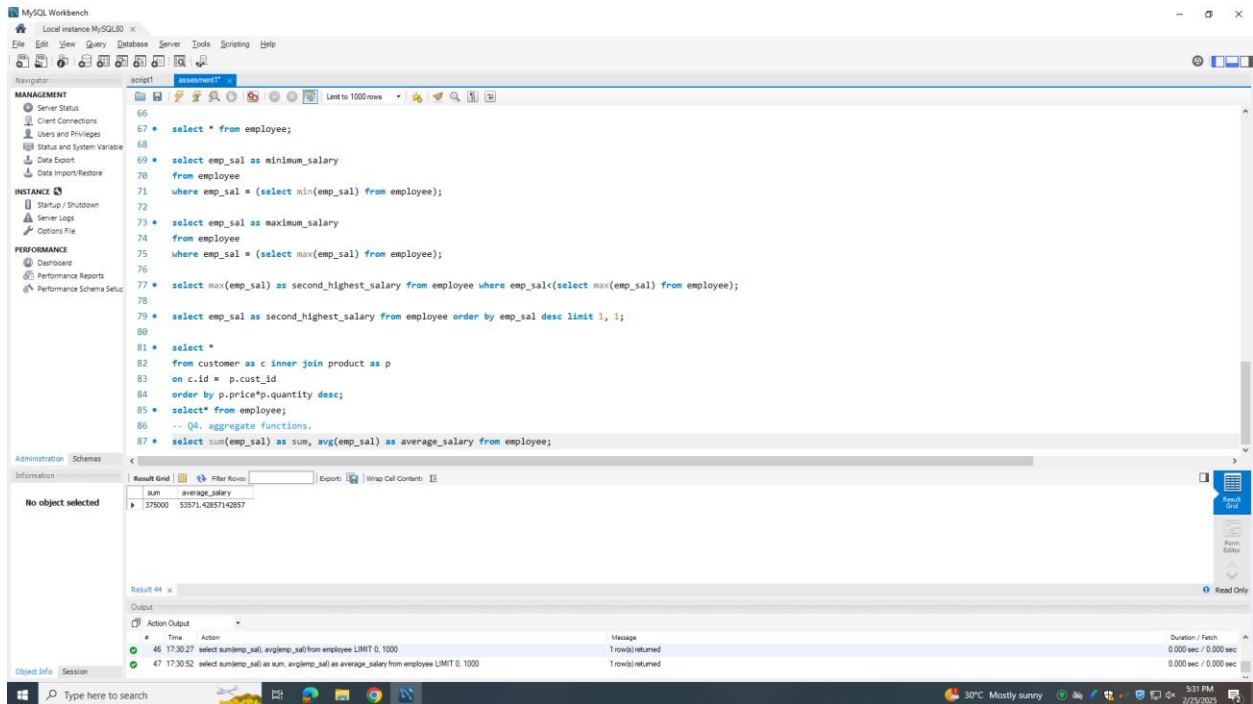
The Results grid shows the output of the Full Join query, displaying 10 rows. The first 5 rows are from the 'customer' table, and the last 5 rows are from the 'product' table. The 'id' column is highlighted in blue.

id	name	age	city	id	name	price	quantity	cust_id
1	Rohan	24	Kolkata	1006	Water	50	1	1
2	Anan	23	Patna	1002	Dal	90	1	2
3	Prakash	22	Behala	1003	Besan	30	3	3
4	Tatpaga	23	Bankura	1004	Pen	350	5	4
5	Abr	22	Kol	1005	Water	50	1	5
6	John	35	ABC					
7				1007	Pen	350	5	
8				1008	newitem	450	10	
9								
10								

The Output pane shows the execution details of the query, including the time taken (17:17:53) and the number of rows returned (10 rows).

Q4. Write a query to find all aggregate functions.

sum() and avg() aggregate functions:



As of now I cannot demonstrate count() function using the Group By as there is no common attribute in the table. But the query looks something like this:

Select <common attribute>, sum(emp_sal) as sum, avg(emp_sal) as average_salary from employee group by <common attribute>;

Q5. Write a Query using all string functions.

Demonstrating concat(), substring(), trim() functions.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
74 from employee
75 where emp_sal = (select max(emp_sal) from employee);
76
77 • select max(emp_sal) as second_highest_salary from employee where emp_sal=(select max(emp_sal) from employee);
78
79 • select emp_sal as second_highest_salary from employee order by emp_sal desc limit 1, 1;
80
81 • select *
82 from customer as c inner join product as p
83 on c.id = p.cust_id
84 order by p.price*p.quantity desc;
85 • select* from employee;
86 -- Q4. aggregate functions.
87 • select sum(emp_sal) as sum, avg(emp_sal) as average_salary from employee group by salary;
88
89 -- Q5. String functions
90 • select concat(emp_id, '-', emp_name) as unique_name,
91 substring(emp_name, 1, 3) as first_three,
92 trim('100' from emp_id) as trimmed_id
93 from employee;
94
95
```

The Results grid shows the output of the query:

unique_name	first_three	trimmed_id
1001_Raghu	Rag	1
1002_Ram	Ram	2
1003_Rajan	Raj	3
1004_Ritesh	Rit	4
1005_Neha	Neh	5
1006_Ashwini	Ash	6
1007_Rohan	Roh	7

The Output pane shows the execution of the query:

```
51 17:40:27 select substring(emp_name, 1, 3) as first_three from employee LIMIT 0, 1000
Message
7 rows(s) returned
0.000 sec / 0.000 sec

52 17:42:13 select concat(emp_id, '-', emp_name) as unique_name, substring(emp_name, 1, 3) as first_three, trim('100' from emp_id) as trimmed_id
Message
7 rows(s) returned
0.000 sec / 0.000 sec
```