

DBMS Lab 4

Name: Paruchuri Anshul

SRN: PES2UG23CS405

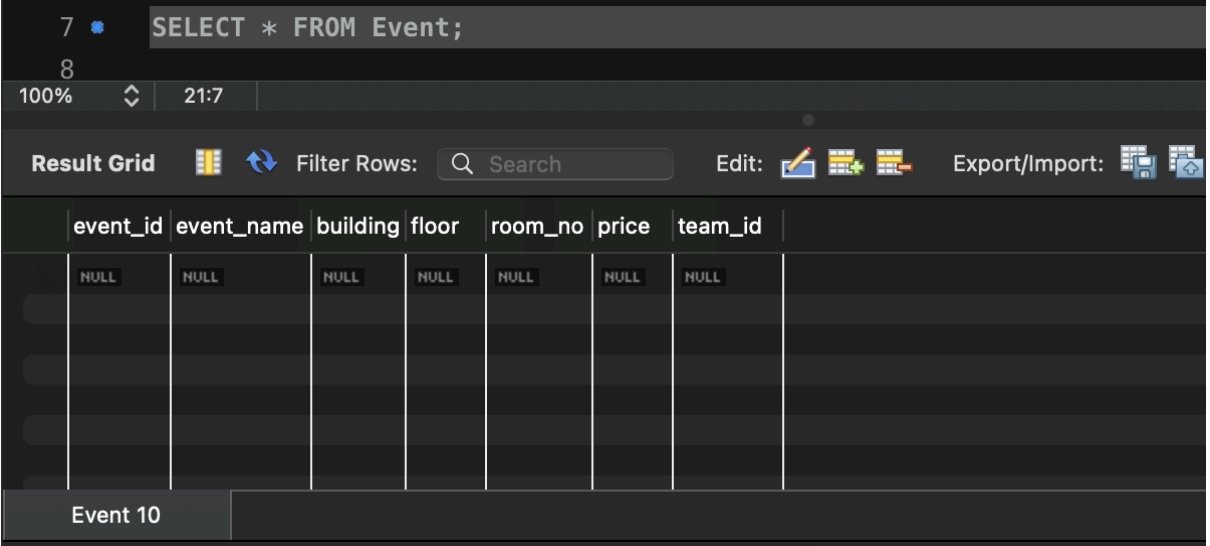
Section: G

Task 1

1. Insert a new event called "AI Hackathon" conducted by team 'T4' in 'Seminar Hall', floor '2', room 205, priced at 900.00

Answer:

Before



7 `SELECT * FROM Event;`

8

100% 21:7

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

event_id	event_name	building	floor	room_no	price	team_id
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Event 10

SQL Commands

first add team t4 to Team table

```
INSERT INTO Team(team_id, team_name, team_type)
```

```
VALUES(1, 'T4', 'ORG');
```

add given values to Event table

```
INSERT INTO Event(event_id, event_name, building, floor, room_no, price, team_id)
```

```
VALUES (1, 'AI Hackathon', 'Seminar Hall', 2, 205, 900.00, 1);
```

After:

```
9 • INSERT INTO Event(event_id, event_name, building, floor, room_no, price, team_id)
10 VALUES (1, 'AI Hackathon', 'Seminar Hall', 2, 205, 900.00, 1);
11
12 • SELECT * FROM Event;
```

100% 21:12

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

event_id	event_name	building	floor	room_no	price	team_id
1	AI Hackathon	Seminar Hall	2	205	900.00	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Update the quantity of 'Mushroom Risotto' in stall 'S1' to 25.

Before:

```
33 • SELECT * FROM Stall_Items;
34
```

100% 27:33

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

stall_id	item_name	stock	price_per_u...
NULL	NULL	NULL	NULL

SQL Command:

```
INSERT INTO Stall(stall_id, name)
VALUES (1, 'S1');
```

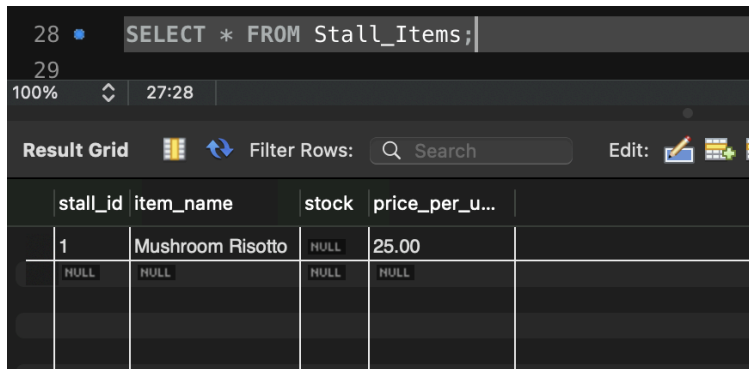
```
INSERT INTO Item(name)
VALUES ('Mushroom Risotto');
```

```
DESCRIBE Stall_Items;
ALTER TABLE Stall_Items
ADD COLUMN price_per_unit FLOAT(5,2);
```

```
INSERT INTO Stall_Items(stall_id, item_name, price_per_unit)
VALUES (1, "Mushroom Risotto", 25);
```

```
SELECT * FROM Stall_Items;
```

After:

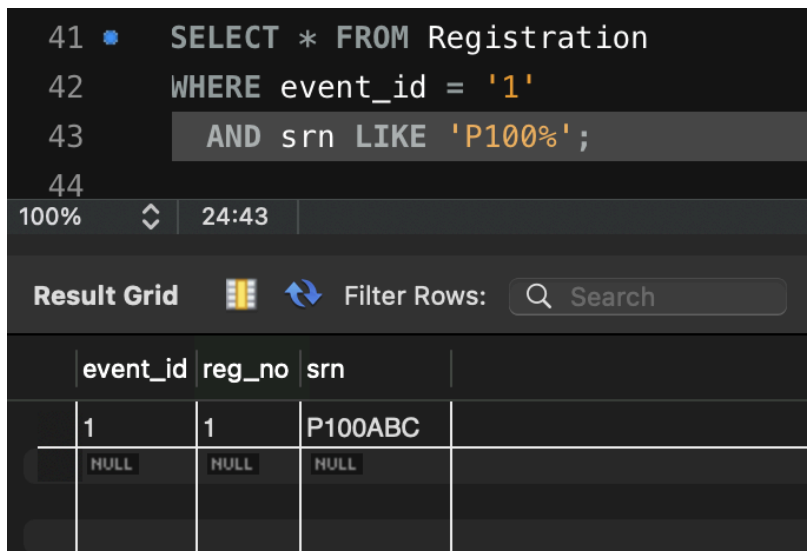


The screenshot shows a SQL query editor with the query `SELECT * FROM Stall_Items;` and its result grid. The result grid has columns: `stall_id`, `item_name`, `stock`, and `price_per_u...`. The first row shows `1` for `stall_id`, `Mushroom Risotto` for `item_name`, `NULL` for `stock`, and `25.00` for `price_per_u...`. The second row shows `NULL` for `stall_id`, `NULL` for `item_name`, `NULL` for `stock`, and `NULL` for `price_per_u...`.

stall_id	item_name	stock	price_per_u...
1	Mushroom Risotto	NULL	25.00
NULL	NULL	NULL	NULL

3. Delete all registrations where the event ID is 'E1' and SRN starts with 'P100'

Before:



The screenshot shows a SQL query editor with the query `SELECT * FROM Registration WHERE event_id = '1' AND srn LIKE 'P100%';` and its result grid. The result grid has columns: `event_id`, `reg_no`, and `srn`. The first row shows `1` for `event_id`, `1` for `reg_no`, and `P100ABC` for `srn`. The second row shows `NULL` for `event_id`, `NULL` for `reg_no`, and `NULL` for `srn`.

event_id	reg_no	srn
1	1	P100ABC
NULL	NULL	NULL

SQL Commands:

```
DESCRIBE Participant;  
INSERT INTO Participant(srn, name, gender, department, semester)  
VALUES ('P100ABC', 'Pranav', 'M', 'CSE', 5);
```

```
# add dummy value  
INSERT INTO Registration(event_id, reg_no, srn)  
VALUES (1, 001, 'P100ABC');
```

```
SELECT * FROM Registration  
WHERE event_id = '1'  
AND srn LIKE 'P100%';
```

```
# now delete it  
DELETE FROM Registration  
WHERE event_id = '1'
```

AND srn LIKE 'P100%';

```
SELECT * FROM Registration
WHERE event_id = '1'
AND srn LIKE 'P100%';
```

After:

```
50 SELECT * FROM Registration
51 WHERE event_id = '1'
52 AND srn LIKE 'P100%';
53
```

100% 1:33

Result Grid Filter Rows: Search

event_id	reg_no	srn
NULL	NULL	NULL

4. Insert a new purchase: 'P1017' buys 3 'Fish Tacos' from stall 'S6' at '2025-07-10 14:00:00'

Before:

```
73 SELECT * FROM Purchased
74 WHERE srn LIKE 'P1017';
75
```

100% 24:74

Result Grid Filter Rows: Search

srn	stall_id	item_name	time_stamp	quantity
NULL	NULL	NULL	NULL	NULL

SQL Commands:

add dummy value

```
INSERT INTO Participant(srn, name, gender, department, semester)
```

```
VALUES ('P1017', 'Rajesh', 'M', 'CSE', 5);
```

```
INSERT INTO Registration(event_id, reg_no, srn)
```

```
VALUES (1, 002, 'P1017');
```

add stall 6 and fish taco

```
INSERT INTO Stall(stall_id, name)
```

```
VALUES (6, 'S6');
```

```
INSERT INTO Item(name)
```

```
VALUES ('Fish Taco');
```

```
INSERT INTO Stall_Items(stall_id, item_name, price_per_unit)
```

```
VALUES (6, "Fish Taco", 50);
```

```
SELECT * FROM Stall_Items;
```

```
SELECT * FROM Purchased
```

```
WHERE srn LIKE 'P1017';
```

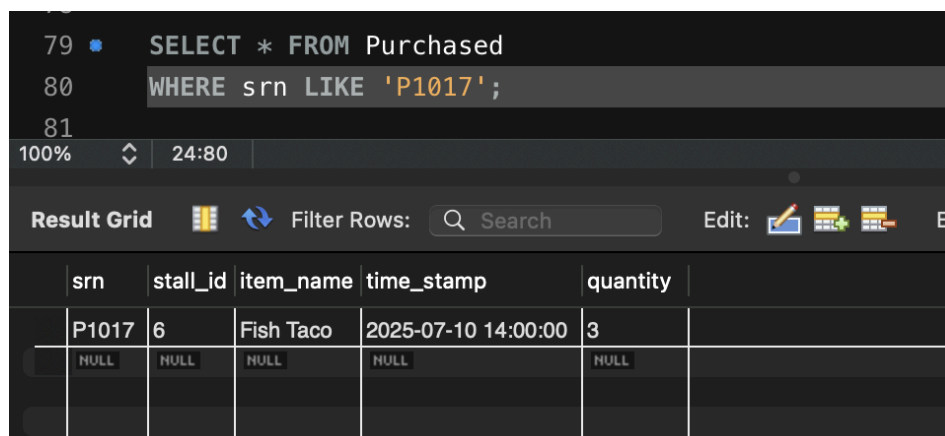
```
INSERT INTO Purchased(srn, stall_id, item_name, time_stamp, quantity)
```

```
Values('P1017', 6, 'Fish Taco', '2025-07-10 14:00:00', 3);
```

```
SELECT * FROM Purchased
```

```
WHERE srn LIKE 'P1017';
```

After:



```
79 SELECT * FROM Purchased
80 WHERE srn LIKE 'P1017';
81
```

100% 24:80

Result Grid Filter Rows: Search Edit:

	srn	stall_id	item_name	time_stamp	quantity
	P1017	6	Fish Taco	2025-07-10 14:00:00	3
	NULL	NULL	NULL	NULL	NULL

Task 2

- Retrieve participants's SRN who are registered only for event 'E2' or 'E5', but not both (using SET operations).

Before:

```
123 -- Event 5 registrations
124 • INSERT INTO Registration(event_id, reg_no, srn)
125     VALUES (5, 006, 'P1004'),
126             (5, 007, 'P1005'),
127             (5, 008, 'P1002'); -- note: P1002 appears in both 2 & 5
128
129 -- Event 2 regs
130 • INSERT INTO Registration(event_id, reg_no, srn)
131     VALUES (2, 009, 'P1001'),
132             (2, 010, 'P1002'),
133             (2, 011, 'P1003');
134
135 • SELECT * from REGISTRATION WHERE event_id = 2;
```

100% 47:135

Result Grid

reg_no	event_id	srn
9	2	P1001
10	2	P1002
11	2	P1003
NULL	NULL	NULL

SQL Command:

```
SELECT srn
FROM Registration
WHERE event_id = '2'
AND srn NOT IN (
    SELECT srn FROM Registration WHERE event_id = '5'
)
UNION
SELECT srn
FROM Registration
WHERE event_id = '5'
AND srn NOT IN (
    SELECT srn FROM Registration WHERE event_id = '2'
);
```

After:

```
136  -- Retrieve participants's SRN who are registered only for event 'E2' or 'E5', but not both (using SET operations).
137  SELECT srn
138  FROM Registration
139  WHERE event_id = '2'
140  AND srn NOT IN (
141      SELECT srn FROM Registration WHERE event_id = '5'
142  )
143  UNION
144  SELECT srn
145  FROM Registration
146  WHERE event_id = '5'
147  AND srn NOT IN (
148      SELECT srn FROM Registration WHERE event_id = '2'
149  );
```

100% 5:149

Result Grid Filter Rows: Search Export:

srn
P1001
P1003
P1004
P1005

6. Display all participants and the names of all their visitors(if any) with a count of visitors.

SQL Command:

```
SELECT
    p.srn,
    p.name AS participant_name,
    v.name AS visitor_name,
    COUNT(v.name) OVER (PARTITION BY p.srn) AS visitor_count
FROM Participant p
LEFT JOIN Visitor v
    ON p.srn = v.participant_srn
ORDER BY p.srn;
```

```
160  SELECT
161      p.srn,
162      p.name AS participant_name,
163      v.name AS visitor_name,
164      COUNT(v.name) OVER (PARTITION BY p.srn) AS visitor_count
165  FROM Participant p
166  LEFT JOIN Visitor v
167      ON p.srn = v.participant_srn
168  ORDER BY p.srn;
```

100% 8:160

Result Grid Filter Rows: Search Export:

srn	participant_na...	visitor_name	visitor_cou...
P1001	Aarav	Divya	2
P1001	Aarav	Kiran	2
P1002	Ananya	NULL	0
P1003	Ravi	NULL	0
P1004	Sneha	Rohit	1
P1005	Vikram	Rohan	1
P1006	Meera	NULL	0
P100ABC	Pranav	NULL	0
P1017	Rajesh	NULL	0

7. List events that have equal number of male and female participants
SQL Commands:

```
SELECT e.event_id, e.event_name
FROM Event e
JOIN Registration r ON e.event_id = r.event_id
JOIN Participant p ON r.srn = p.srn
GROUP BY e.event_id, e.event_name
HAVING SUM(p.gender = 'M') = SUM(p.gender = 'F');
```

```
170 -- List events that have equal number of male and female participants
171 -- adding
172 • INSERT INTO Registration(event_id, reg_no, srn)
173   VALUES (3, 012, 'P1001'),
174           (3, 013, 'P1002'),
175           (3, 014, 'P1003'),
176           (3, 015, 'P1004');
177
178 • SELECT e.event_id, e.event_name
179   FROM Event e
180  JOIN Registration r ON e.event_id = r.event_id
181  JOIN Participant p ON r.srn = p.srn
182  GROUP BY e.event_id, e.event_name
183  HAVING SUM(p.gender = 'M') = SUM(p.gender = 'F');
184
185
```

100% 50:183

Result Grid Filter Rows: Search Export:

event_id	event_name
3	Math Quiz

8. Display each event's name and a binary indicator of whether it occurred after the Golden Jubilee (year > 2047)

SQL Commands:

```
SELECT
    e.event_name,
    CASE
        WHEN f.year > 2047 THEN 1
        ELSE 0
    END AS after_golden_jubilee
FROM Event e
JOIN Team t ON e.team_id = t.team_id
JOIN Fest f ON t.fest_id = f.fest_id;
```


188

•

INSERT INTO Fest(fest_id, fest_name, year, head_team_id)

189

VALUES (1, 'TechFest', 2045, 1),

190

(2, 'FutureFest', 2050, 2);

191

192

•

UPDATE Team SET fest_id = 1 WHERE team_id = 1;

193

•

UPDATE Team SET fest_id = 2 WHERE team_id = 2;

194

•

UPDATE Team SET fest_id = 2 WHERE team_id = 3;

195

196

•

SELECT

197

e.event_name,

198

CASE

199

WHEN f.year > 2047 THEN 1

200

ELSE 0

201

END AS after_golden_jubilee

202

FROM Event e

203

JOIN Team t ON e.team_id = t.team_id

204

JOIN Fest f ON t.fest_id = f.fest_id;

205



206

100%


↕

8:196

Result Grid

Filter Rows:

Export: 

event_name	after_golden_jubil...	
AI Hackathon	0	
Robotics Expo	0	
Code Sprint	1	
Cybersecurity Workshop	1	
Math Quiz	1	
Golden Jubilee Special	1	

Result 36