RDBMS Lab Evaluation

Batch 1 — Sets 1–15 · Solutions

CSE2132 RDBMS Lab · Labs 1–7. For each set this document gives the model SQL for every task together with the output MySQL 8 produced when the tasks were run in order on a freshly built database. Use it to check your own work after the evaluation.

Error tasks show the MySQL error number you should have noted in your comment: 1062 duplicate / UNIQUE, 1048 NOT NULL, 1452 foreign key (child row), 1451 foreign key (parent row), 3819 CHECK.

Set 01 – Merit & Dues Review
Set 02 – Cardiology Desk
Set 03 – Premium Customers
Set 04 – Account Review
Set 05 – Delhi & Mumbai Departures
Set 06 – Salary Review
Set 07 – Semester Planning
Set 08 – Doctor Directory
Set 09 – Order Desk
Set 10 – Transaction Audit
Set 11 – Passenger Manifest
Set 12 – Joiners & Cities
Set 13 – Exam Analysis
Set 14 – Patient Age Groups
Set 15 – Category Performance

Set 01 - Merit & Dues Review

University (`eval_university`) - tables: STUDENT, EXAM_RESULT

Task 1 - Retrieve data (5 marks)

> Table STUDENT. Show roll_no, sname, dept, cgpa of students who are in CSE or IT (use IN), have a CGPA between 7.5 and 9.0 (inclusive) and whose email is recorded. Sort by cgpa (highest first), then roll_no. Show only the first 4 rows (LIMIT).

```
SELECT roll_no, sname, dept, cgpa FROM STUDENT WHERE dept IN ('CSE','IT') AND cgpa BETWEEN 7.5 AND 9.0
AND email IS NOT NULL ORDER BY cgpa DESC, roll_no LIMIT 4;
```

4 row(s):

roll_no	sname	dept	cgpa
2118	Lakshmi Menon	IT	9.00
2101	Aarav Sharma	CSE	8.72
2106	Meera Nair	IT	8.72
2120	Zoya Khan	CSE	8.72

Task 2 - Summarise data (5 marks)

> Table STUDENT. (a) ONE row with: total_students (number of students), with_cgpa (students whose CGPA is recorded), avg_cgpa (average CGPA, ROUND to 2 decimals), total_due (sum of fee_due). (b) For each dept show dept, students (count) and avg_cgpa (ROUND 2). Keep only departments with 4 or more students. Sort by students (highest first), then dept.

```
SELECT COUNT(*) AS total_students, COUNT(cgpa) AS with_cgpa, ROUND(AVG(cgpa), 2) AS avg_cgpa,
SUM(fee_due) AS total_due FROM STUDENT;
```

1 row(s):

total_students	with_cgpa	avg_cgpa	total_due
20	19	7.99	80000.00

```
SELECT dept, COUNT(*) AS students, ROUND(AVG(cgpa), 2) AS avg_cgpa FROM STUDENT GROUP BY dept HAVING
COUNT(*) >= 4 ORDER BY students DESC, dept;
```

3 row(s):

dept	students	avg_cgpa
CSE	6	8.61
ECE	4	7.63
IT	4	8.56

Task 3 - Design & define (5 marks)

> ER fragment: a STUDENT can receive many SCHOLARSHIPs; each scholarship belongs to exactly one student. (a) Create table SCHOLARSHIP: sch_id INT primary key; roll_no INT NOT NULL, foreign key to STUDENT(roll_no); sch_name VARCHAR(30) NOT NULL; amount DECIMAL(8,2) with a CHECK that amount > 0; status VARCHAR(10) with DEFAULT 'Active'. (b) Add a column phone VARCHAR(15) to STUDENT. (c) Insert (sch_id 1, roll_no 2101, 'Merit', 50000) listing the columns and leaving status out. Then insert (2, 2102, 'Sports', -500): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE SCHOLARSHIP; and SELECT * FROM SCHOLARSHIP ORDER BY sch_id;

```
CREATE TABLE SCHOLARSHIP (sch_id INT PRIMARY KEY, roll_no INT NOT NULL, sch_name VARCHAR(30) NOT NULL,
amount DECIMAL(8,2) CHECK (amount > 0), status VARCHAR(10) DEFAULT 'Active', FOREIGN KEY (roll_no)
REFERENCES STUDENT(roll_no));
```

OK - 0 row(s) affected

```
ALTER TABLE STUDENT ADD COLUMN phone VARCHAR(15);
```

OK - 0 row(s) affected

```
INSERT INTO SCHOLARSHIP (sch_id, roll_no, sch_name, amount) VALUES (1, 2101, 'Merit', 50000);
```

OK - 1 row(s) affected

```
INSERT INTO SCHOLARSHIP (sch_id, roll_no, sch_name, amount) VALUES (2, 2102, 'Sports', -500);
```

Error number 3819: Check constraint 'scholarship_chk_1' is violated.

```
DESCRIBE SCHOLARSHIP;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
sch_id	int	NO	PRI	NULL	
roll_no	int	NO	MUL	NULL	
sch_name	varchar(30)	NO		NULL	
amount	decimal(8,2)	YES		NULL	
status	varchar(10)	YES		Active	

```
SELECT * FROM SCHOLARSHIP ORDER BY sch_id;
```

1 row(s):

sch_id	roll_no	sch_name	amount	status
1	2101	Merit	50000.00	Active

Task 4 - Modify data (5 marks)

> (a) Insert student 2121, 'Mohit Jain', ECE, semester 3, cgpa 7.50, city 'Pune', email 'mohit.jain@uni.edu' - list the columns and leave fee_due out. (b) Insert student 2122, 'Neha Jain', ECE, semester 3, cgpa 8.00, city 'Pune' with the SAME email as 2121: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Reduce fee_due by 10% for every ECE student who currently owes money (fee_due > 0). (d) Delete from EXAM_RESULT every row where marks IS NULL (student was absent). (e) Verify with: SELECT roll_no, sname, fee_due FROM STUDENT WHERE dept = 'ECE' ORDER BY roll_no; and SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;

```
INSERT INTO STUDENT (roll_no, sname, dept, semester, cgpa, city, email) VALUES (2121, 'Mohit Jain',
'ECE', 3, 7.50, 'Pune', 'mohit.jain@uni.edu');
```

OK - 1 row(s) affected

```
INSERT INTO STUDENT (roll_no, sname, dept, semester, cgpa, city, email) VALUES (2122, 'Neha Jain',
'ECE', 3, 8.00, 'Pune', 'mohit.jain@uni.edu');
```

Error number 1062: Duplicate entry 'mohit.jain@uni.edu' for key 'student.email'

```
UPDATE STUDENT SET fee_due = fee_due * 0.90 WHERE dept = 'ECE' AND fee_due > 0;
```

OK - 3 row(s) affected

```
DELETE FROM EXAM_RESULT WHERE marks IS NULL;
```

OK - 1 row(s) affected

```
SELECT roll_no, sname, fee_due FROM STUDENT WHERE dept = 'ECE' ORDER BY roll_no;
```

5 row(s):

roll_no	sname	fee_due
2104	Sneha Iyer	0.00
2105	Kabir Singh	22500.00
2111	Yash Patel	6750.00
2117	Aditya Bose	8100.00
2121	Mohit Jain	0.00

```
SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;
```

1 row(s):

rows_left
23

Set 02 - Cardiology Desk

Hospital (`eval_hospital`) - tables: APPOINTMENT, DOCTOR

Task 1 - Retrieve data (5 marks)

> Table APPOINTMENT. Show appt_id, doctor_id, patient_name, age, status of appointments of doctors 101, 102 or 103 (use IN) whose status is Completed or Scheduled (use IN) and whose patient age is between 40 and 70 (inclusive). Sort by age (oldest first), then appt_id. Show only the first 6 rows (LIMIT).

```
SELECT appt_id, doctor_id, patient_name, age, status FROM APPOINTMENT WHERE doctor_id IN (101,102,103)
AND status IN ('Completed','Scheduled') AND age BETWEEN 40 AND 70 ORDER BY age DESC, appt_id LIMIT 6;
```

6 row(s):

appt_id	doctor_id	patient_name	age	status
22	101	Mohammed Iqbal	64	Scheduled
2	101	Sunita Devi	62	Completed
1	101	Ramesh Kumar	58	Completed
23	102	Shalini Gupta	56	Completed
25	103	Jaspreet Singh	47	Completed
3	102	Alok Mishra	45	Completed

Task 2 - Summarise data (5 marks)

> Table APPOINTMENT. (a) ONE row with: total_appts, with_fee (appointments that have a fee recorded), revenue (sum of fee_charged), avg_age (ROUND 1). (b) For COMPLETED appointments, for each doctor_id show doctor_id, visits (count) and revenue (sum of fee_charged). Keep only doctors whose revenue is more than 1500. Sort by revenue (highest first), then doctor_id.

```
SELECT COUNT(*) AS total_appts, COUNT(fee_charged) AS with_fee, SUM(fee_charged) AS revenue,
ROUND(AVG(age), 1) AS avg_age FROM APPOINTMENT;
```

1 row(s):

total_appts	with_fee	revenue	avg_age
26	19	14750.00	38.7

```
SELECT doctor_id, COUNT(*) AS visits, SUM(fee_charged) AS revenue FROM APPOINTMENT WHERE status =
'Completed' GROUP BY doctor_id HAVING SUM(fee_charged) > 1500 ORDER BY revenue DESC, doctor_id;
```

5 row(s):

doctor_id	visits	revenue
101	2	2400.00
104	3	2100.00
103	2	2000.00
102	2	1800.00
109	4	1550.00

Task 3 - Design & define (5 marks)

> ER fragment: an APPOINTMENT can lead to many PRESCRIPTION lines; each line belongs to exactly one appointment. (a) Create table PRESCRIPTION: rx_id INT primary key; appt_id INT NOT NULL, foreign key to APPOINTMENT(appt_id); medicine VARCHAR(40) NOT NULL; days INT with CHECK (days BETWEEN 1 AND 30); refill CHAR(1) DEFAULT 'N'. (b) Add a column remarks VARCHAR(100) to APPOINTMENT. (c) Insert (1, appt 1, 'Aspirin', 10 days) listing the columns and leaving refill out. Then insert (2, appt 3, 'Statin', 45 days): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE PRESCRIPTION; and SELECT * FROM PRESCRIPTION ORDER BY rx_id;

```
CREATE TABLE PRESCRIPTION (rx_id INT PRIMARY KEY, appt_id INT NOT NULL, medicine VARCHAR(40) NOT NULL,
days INT CHECK (days BETWEEN 1 AND 30), refill CHAR(1) DEFAULT 'N', FOREIGN KEY (appt_id) REFERENCES
APPOINTMENT(appt_id));
```

OK - 0 row(s) affected

```
ALTER TABLE APPOINTMENT ADD COLUMN remarks VARCHAR(100);
```

OK - 0 row(s) affected

```
INSERT INTO PRESCRIPTION (rx_id, appt_id, medicine, days) VALUES (1, 1, 'Aspirin', 10);
```

OK - 1 row(s) affected

```
INSERT INTO PRESCRIPTION (rx_id, appt_id, medicine, days) VALUES (2, 3, 'Statin', 45);
```

Error number 3819: Check constraint 'prescription_chk_1' is violated.

```
DESCRIBE PRESCRIPTION;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
rx_id	int	NO	PRI	NULL	
appt_id	int	NO	MUL	NULL	
medicine	varchar(40)	NO		NULL	
days	int	YES		NULL	
refill	char(1)	YES		N	

```
SELECT * FROM PRESCRIPTION ORDER BY rx_id;
```

1 row(s):

rx_id	appt_id	medicine	days	refill
1	1	Aspirin	10	N

Task 4 - Modify data (5 marks)

> (a) Insert appointment 27: doctor 106, 'Sana Mirza', age 26, gender 'F', date '2026-08-16' - list the columns and leave status and fee_charged out (status must default to 'Scheduled'). (b) Insert appointment 28 for doctor 999: there is no such doctor, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Increase consult_fee by 100 for every Cardiology doctor (table DOCTOR). (d) Delete all Cancelled appointments. (e) Verify with: SELECT appt_id, status, fee_charged FROM APPOINTMENT WHERE appt_id = 27; SELECT doctor_id, consult_fee FROM DOCTOR ORDER BY doctor_id; and SELECT COUNT(*) AS rows_left FROM APPOINTMENT;

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date) VALUES (27, 106,
'Sana Mirza', 26, 'F', '2026-08-16');
```

OK - 1 row(s) affected

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date) VALUES (28, 999,
'Test Patient', 30, 'M', '2026-08-16');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_hospital` . `appointment` , CONSTRAINT `appointment_ibfk_1` FOREIGN KEY (`doctor_id`) REFERENCES `doctor` (`doctor_id`))

```
UPDATE DOCTOR SET consult_fee = consult_fee + 100 WHERE specialization = 'Cardiology';
```

OK - 2 row(s) affected

```
DELETE FROM APPOINTMENT WHERE status = 'Cancelled';
```

OK - 3 row(s) affected

```
SELECT appt_id, status, fee_charged FROM APPOINTMENT WHERE appt_id = 27;
```

1 row(s):

appt_id	status	fee_charged
27	Scheduled	NULL

```
SELECT doctor_id, consult_fee FROM DOCTOR ORDER BY doctor_id;
```

10 row(s):

doctor_id	consult_fee
101	1300.00
102	1000.00
103	1000.00
104	700.00
105	1500.00
106	600.00
107	800.00
108	500.00
109	400.00
110	900.00

```
SELECT COUNT(*) AS rows_left FROM APPOINTMENT;
```

1 row(s):

rows_left
24

Set 03 - Premium Customers

Retail Store (`eval_retail`) - tables: CUSTOMER, ORDERS

Task 1 - Retrieve data (5 marks)

> Table CUSTOMER. Show cust_id, cname, city, membership, credit_limit of Gold or Platinum members (use IN) whose credit_limit is between 40000 and 100000 (inclusive) and who do NOT live in Delhi. Sort by credit_limit (highest first), then cust_id. Show only the first 3 rows (LIMIT).

```
SELECT cust_id, cname, city, membership, credit_limit FROM CUSTOMER WHERE membership IN
('Gold','Platinum') AND credit_limit BETWEEN 40000 AND 100000 AND city <> 'Delhi' ORDER BY credit_limit
DESC, cust_id LIMIT 3;
```

3 row(s):

cust_id	cname	city	membership	credit_limit
7	Gita Bansal	Jaipur	Gold	60000.00
1	Aditi Rao	Bengaluru	Gold	50000.00
11	Komal Shah	Ahmedabad	Gold	45000.00

Task 2 - Summarise data (5 marks)

> Table ORDERS, Delivered orders only (status = 'Delivered'). (a) ONE row with: orders (count), with_discount (orders having discount_pct), avg_discount (ROUND 2, NULLs ignored), avg_discount_all (ROUND 2, NULL counted as 0 - use IFNULL), revenue (sum of qty *unit_price). (b) For each category show category, orders (count) and revenue (sum of qty* unit_price). Keep only categories whose revenue is more than 10000. Sort by revenue (highest first), then category.

```
SELECT COUNT(*) AS orders, COUNT(discount_pct) AS with_discount, ROUND(AVG(discount_pct), 2) AS
avg_discount, ROUND(AVG(IFNULL(discount_pct, 0)), 2) AS avg_discount_all, SUM(qty * unit_price) AS
revenue FROM ORDERS WHERE status = 'Delivered';
```

1 row(s):

orders	with_discount	avg_discount	avg_discount_all	revenue
17	9	5.67	3.00	277200.00

```
SELECT category, COUNT(*) AS orders, SUM(qty * unit_price) AS revenue FROM ORDERS WHERE status =
'Delivered' GROUP BY category HAVING SUM(qty * unit_price) > 10000 ORDER BY revenue DESC, category;
```

3 row(s):

category	orders	revenue
Electronics	7	241000.00
Clothing	5	14250.00
Footwear	3	14250.00

Task 3 - Design & define (5 marks)

> ER fragment: an ORDER can have many REVIEWs; each review belongs to one order. (a) Create table REVIEW: review_id INT primary key; order_id INT NOT NULL, foreign key to ORDERS(order_id); rating INT NOT NULL with CHECK (rating BETWEEN 1 AND 5); comments VARCHAR(200); review_status VARCHAR(10) DEFAULT 'Pending'. (b) Add a column email VARCHAR(60) to CUSTOMER. (c) Insert (1, order 1001, rating 5, 'Great laptop') listing the columns and leaving review_status out. Then insert (2, order 1003, rating 6, 'Wow'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE REVIEW; and SELECT * FROM REVIEW ORDER BY review_id;

```
CREATE TABLE REVIEW (review_id INT PRIMARY KEY, order_id INT NOT NULL, rating INT NOT NULL CHECK (rating
BETWEEN 1 AND 5), comments VARCHAR(200), review_status VARCHAR(10) DEFAULT 'Pending', FOREIGN KEY
(order_id) REFERENCES ORDERS(order_id));
```

OK - 0 row(s) affected

```
ALTER TABLE CUSTOMER ADD COLUMN email VARCHAR(60);
```

OK - 0 row(s) affected

```
INSERT INTO REVIEW (review_id, order_id, rating, comments) VALUES (1, 1001, 5, 'Great laptop');
```

OK - 1 row(s) affected

```
INSERT INTO REVIEW (review_id, order_id, rating, comments) VALUES (2, 1003, 6, 'Wow');
```

Error number 3819: Check constraint 'review_chk_1' is violated.

```
DESCRIBE REVIEW;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
review_id	int	NO	PRI	NULL	
order_id	int	NO	MUL	NULL	
rating	int	NO		NULL	
comments	varchar(200)	YES		NULL	
review_status	varchar(10)	YES		Pending	

```
SELECT * FROM REVIEW ORDER BY review_id;
```

1 row(s):

review_id	order_id	rating	comments	review_status
1	1001	5	Great laptop	Pending

Task 4 - Modify data (5 marks)

> (a) Insert customer 13, 'Mehul Doshi', city 'Surat', membership 'Silver', joined_on '2026-09-01' - list the columns and leave credit_limit out (it must default to 10000). (b) Insert an order (1029, cust_id 99, 'Mouse', 'Electronics', 1, 800, '2026-07-17', 'Placed'): customer 99 does not exist, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Increase unit_price by 10% for every Footwear order that is not Cancelled. (d) Delete every Cancelled order. (e) Verify with: SELECT cust_id, credit_limit FROM CUSTOMER WHERE cust_id = 13; SELECT order_id, unit_price FROM ORDERS WHERE category = 'Footwear' ORDER BY order_id; and SELECT COUNT(*) AS rows_left FROM ORDERS;

```
INSERT INTO CUSTOMER (cust_id, cname, city, membership, joined_on) VALUES (13, 'Mehul Doshi', 'Surat',
'Silver', '2026-09-01');
```

OK - 1 row(s) affected

```
INSERT INTO ORDERS (order_id, cust_id, item, category, qty, unit_price, order_date, status) VALUES
(1029, 99, 'Mouse', 'Electronics', 1, 800, '2026-07-17', 'Placed');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_retail` . `orders` , CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`cust_id`) REFERENCES `customer` (`cust_id`))

```
UPDATE ORDERS SET unit_price = unit_price * 1.10 WHERE category = 'Footwear' AND status <> 'Cancelled';
```

OK - 4 row(s) affected

```
DELETE FROM ORDERS WHERE status = 'Cancelled';
```

OK - 2 row(s) affected

```
SELECT cust_id, credit_limit FROM CUSTOMER WHERE cust_id = 13;
```

1 row(s):

cust_id	credit_limit
13	10000.00

```
SELECT order_id, unit_price FROM ORDERS WHERE category = 'Footwear' ORDER BY order_id;
```

4 row(s):

order_id	unit_price
1003	3520.00
1011	4510.00
1016	1045.00
1026	1045.00

```
SELECT COUNT(*) AS rows_left FROM ORDERS;
```

1 row(s):

rows_left
26

Set 04 - Account Review

Bank (`eval_bank`) - tables: ACCOUNT, TXN

Task 1 - Retrieve data (5 marks)

> Table ACCOUNT. Show acc_no, holder_name, acc_type, balance of Active accounts whose acc_type is Savings or Current (use IN) and whose balance is between 10000 and 300000 (inclusive). Sort by balance (highest first), then acc_no. Show only the first 5 rows (LIMIT).

```
SELECT acc_no, holder_name, acc_type, balance FROM ACCOUNT WHERE status = 'Active' AND acc_type IN
('Savings','Current') AND balance BETWEEN 10000 AND 300000 ORDER BY balance DESC, acc_no LIMIT 5;
```

5 row(s):

acc_no	holder_name	acc_type	balance
5008	Hina Qureshi	Current	215000.00
5001	Anil Chopra	Savings	125000.00
5014	Nandita Roy	Current	98000.00
5004	Divya Narang	Savings	76000.00
5007	Gaurav Anand	Savings	45250.50

Task 2 - Summarise data (5 marks)

> Table ACCOUNT. (a) For Active accounts, ONE row with: accounts (count), total_balance, avg_balance (ROUND 2), without_nominee (accounts with no nominee - hint: COUNT(*) minus COUNT(nominee)). (b) For ALL accounts, for each acc_type show acc_type, accounts (count) and total_balance. Keep only types whose total_balance is more than 300000. Sort by total_balance (highest first).

```
SELECT COUNT(*) AS accounts, SUM(balance) AS total_balance, ROUND(AVG(balance), 2) AS avg_balance,
COUNT(*) - COUNT(nominee) AS without_nominee FROM ACCOUNT WHERE status = 'Active';
```

1 row(s):

accounts	total_balance	avg_balance	without_nominee
12	2685051.25	223754.27	5

```
SELECT acc_type, COUNT(*) AS accounts, SUM(balance) AS total_balance FROM ACCOUNT GROUP BY acc_type
HAVING SUM(balance) > 300000 ORDER BY total_balance DESC;
```

3 row(s):

acc_type	accounts	total_balance
Fixed	2	1250000.00
Current	4	1163000.00
Savings	8	334051.25

Task 3 - Design & define (5 marks)

> ER fragment: an ACCOUNT can have many LOANs; each loan belongs to one account. (a) Create table LOAN: loan_id INT primary key; acc_no INT NOT NULL, foreign key to ACCOUNT(acc_no); principal DECIMAL(12,2) NOT NULL with CHECK (principal > 0); rate DECIMAL(4,2) DEFAULT 8.50; status VARCHAR(10) DEFAULT 'Open'. (b) Add a column email VARCHAR(60) to ACCOUNT. (c) Insert (1, account 5001, 200000) listing loan_id, acc_no, principal. Then insert (2, account 5003, 0): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE LOAN; and SELECT * FROM LOAN ORDER BY loan_id;

```
CREATE TABLE LOAN (loan_id INT PRIMARY KEY, acc_no INT NOT NULL, principal DECIMAL(12,2) NOT NULL CHECK
(principal > 0), rate DECIMAL(4,2) DEFAULT 8.50, status VARCHAR(10) DEFAULT 'Open', FOREIGN KEY (acc_no)
REFERENCES ACCOUNT(acc_no));
```

OK - 0 row(s) affected

```
ALTER TABLE ACCOUNT ADD COLUMN email VARCHAR(60);
```

OK - 0 row(s) affected

```
INSERT INTO LOAN (loan_id, acc_no, principal) VALUES (1, 5001, 200000);
```

OK - 1 row(s) affected

```
INSERT INTO LOAN (loan_id, acc_no, principal) VALUES (2, 5003, 0);
```

Error number 3819: Check constraint 'loan_chk_1' is violated.

```
DESCRIBE LOAN;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
loan_id	int	NO	PRI	NULL	
acc_no	int	NO	MUL	NULL	
principal	decimal(12,2)	NO		NULL	
rate	decimal(4,2)	YES		8.50	
status	varchar(10)	YES		Open	

```
SELECT * FROM LOAN ORDER BY loan_id;
```

1 row(s):

loan_id	acc_no	principal	rate	status
1	5001	200000.00	8.50	Open

Task 4 - Modify data (5 marks)

> (a) Open account 5015: 'Omkar Pillai', 'Chennai', 'Savings', opened_on '2026-10-01' - list the columns and leave balance, status and nominee out (balance defaults to 0, status to 'Active'). (b) Insert a transaction (29, account 9999, 'Deposit', 1000, '2026-10-02', 'Cash', NULL): account 9999 does not exist, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Credit 4% interest (balance *1.04) to every Active Savings account. (d) Delete every transaction of less than 1000. (e) Verify with: SELECT acc_no, balance, status FROM ACCOUNT WHERE acc_type = 'Savings' ORDER BY acc_no; and SELECT COUNT()* AS rows_left FROM TXN;

```
INSERT INTO ACCOUNT (acc_no, holder_name, branch_city, acc_type, opened_on) VALUES (5015, 'Omkar
Pillai', 'Chennai', 'Savings', '2026-10-01');
```

OK - 1 row(s) affected

```
INSERT INTO TXN (txn_id, acc_no, txn_type, amount, txn_date, mode, remarks) VALUES (29, 9999, 'Deposit',
1000, '2026-10-02', 'Cash', NULL);
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_bank` . `txn` , CONSTRAINT `txn_ibfk_1` FOREIGN KEY (`acc_no`) REFERENCES `account` (`acc_no`))

```
UPDATE ACCOUNT SET balance = balance * 1.04 WHERE acc_type = 'Savings' AND status = 'Active';
```

OK - 6 row(s) affected

```
DELETE FROM TXN WHERE amount < 1000;
```

OK - 1 row(s) affected

```
SELECT acc_no, balance, status FROM ACCOUNT WHERE acc_type = 'Savings' ORDER BY acc_no;
```

9 row(s):

acc_no	balance	status
5001	130000.00	Active
5003	8840.00	Active
5004	79040.00	Active
5006	0.00	Dormant
5007	47060.52	Active
5009	15600.00	Active
5011	2392.78	Active
5013	62000.00	Closed
5015	0.00	Active

```
SELECT COUNT(*) AS rows_left FROM TXN;
```

1 row(s):

rows_left
27

Set 05 - Delhi & Mumbai Departures

Airline (`eval_airline`) - tables: FLIGHT, BOOKING

Task 1 - Retrieve data (5 marks)

> Table FLIGHT. Show flight_no, airline, source, destination, base_fare of flights that depart from Delhi or Mumbai (use IN), have a base_fare between 3000 and 7000 (inclusive) and are not 'Cancelled'. Sort by base_fare (highest first), then flight_no.

```
SELECT flight_no, airline, source, destination, base_fare FROM FLIGHT WHERE source IN ('Delhi','Mumbai')
AND base_fare BETWEEN 3000 AND 7000 AND status <> 'Cancelled' ORDER BY base_fare DESC, flight_no;
```

5 row(s):

flight_no	airline	source	destination	base_fare
SJ101	SkyJet	Delhi	Mumbai	6500.00
AN301	AirNova	Delhi	Bengaluru	5800.00
SJ202	SkyJet	Mumbai	Chennai	5200.00
SJ303	SkyJet	Delhi	Jaipur	3400.00
BW601	BlueWing	Delhi	Jaipur	3000.00

Task 2 - Summarise data (5 marks)

> Table FLIGHT. (a) ONE row with: flights (count), total_seats (sum), avg_base_fare (ROUND 2), airlines (number of DISTINCT airlines), max_fare (highest base_fare). (b) For each airline show airline, flights (count) and seats (sum of total_seats). Keep only airlines with 3 or more flights. Sort by airline.

```
SELECT COUNT(*) AS flights, SUM(total_seats) AS total_seats, ROUND(AVG(base_fare), 2) AS avg_base_fare,
COUNT(DISTINCT airline) AS airlines, MAX(base_fare) AS max_fare FROM FLIGHT;
```

1 row(s):

flights	total_seats	avg_base_fare	airlines	max_fare
10	1592	5080.00	4	8100.00

```
SELECT airline, COUNT(*) AS flights, SUM(total_seats) AS seats FROM FLIGHT GROUP BY airline HAVING
COUNT(*) >= 3 ORDER BY airline;
```

2 row(s):

airline	flights	seats
AirNova	3	552
SkyJet	3	460

Task 3 - Design & define (5 marks)

> ER fragment: a FLIGHT has many CREW members; each crew member is rostered on exactly one flight (in this model). (a) Create table CREW: crew_id INT primary key; flight_no VARCHAR(8) NOT NULL, foreign key to FLIGHT(flight_no); crew_name VARCHAR(40) NOT NULL; role VARCHAR(15) DEFAULT 'Cabin Crew'; exp_yrs INT with CHECK (exp_yrs >= 0). (b) Add a column gate VARCHAR(5) to FLIGHT. (c) Insert (1, 'SJ101', 'Anita Roy', 6 years) listing crew_id, flight_no, crew_name, exp_yrs. Then insert (2, 'SJ101', 'Binod Das', -2 years): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE CREW; and SELECT * FROM CREW ORDER BY crew_id;

```
CREATE TABLE CREW (crew_id INT PRIMARY KEY, flight_no VARCHAR(8) NOT NULL, crew_name VARCHAR(40) NOT
NULL, role VARCHAR(15) DEFAULT 'Cabin Crew', exp_yrs INT CHECK (exp_yrs >= 0), FOREIGN KEY (flight_no)
REFERENCES FLIGHT(flight_no));
```

OK - 0 row(s) affected

```
ALTER TABLE FLIGHT ADD COLUMN gate VARCHAR(5);
```

OK - 0 row(s) affected

```
INSERT INTO CREW (crew_id, flight_no, crew_name, exp_yrs) VALUES (1, 'SJ101', 'Anita Roy', 6);
```

OK - 1 row(s) affected

```
INSERT INTO CREW (crew_id, flight_no, crew_name, exp_yrs) VALUES (2, 'SJ101', 'Binod Das', -2);
```

Error number 3819: Check constraint 'crew_chk_1' is violated.

```
DESCRIBE CREW;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
crew_id	int	NO	PRI	NULL	
flight_no	varchar(8)	NO	MUL	NULL	
crew_name	varchar(40)	NO		NULL	
role	varchar(15)	YES		Cabin Crew	
exp_yrs	int	YES		NULL	

```
SELECT * FROM CREW ORDER BY crew_id;
```

1 row(s):

crew_id	flight_no	crew_name	role	exp_yrs
1	SJ101	Anita Roy	Cabin Crew	6

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add three bookings on flight 'SJ101' (class 'Economy', status 'Confirmed'; list the columns): (27, 'Kavita Rao', 31, fare 6500, baggage 15), (28, 'Manoj Rao', 33, fare 6500, baggage NULL), (29, 'Lucky Rao', 6, fare 3250, baggage NULL). (b) Insert a booking (30, flight 'XX999', ...) for a flight that does not exist: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Increase base_fare by 5% for every flight departing on '2026-11-05'. (d) Delete every Waitlisted booking. (e) Verify with: SELECT flight_no, base_fare FROM FLIGHT WHERE dep_date = '2026-11-05' ORDER BY flight_no; and SELECT COUNT(*) AS bookings_left FROM BOOKING;

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, fare_paid, booking_status,
baggage_kg) VALUES (27, 'SJ101', 'Kavita Rao', 31, 'Economy', 6500, 'Confirmed', 15), (28, 'SJ101',
'Manoj Rao', 33, 'Economy', 6500, 'Confirmed', NULL), (29, 'SJ101', 'Lucky Rao', 6, 'Economy', 3250,
'Confirmed', NULL);
```

OK - 3 row(s) affected

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, booking_status) VALUES (30,
'XX999', 'Ghost Passenger', 30, 'Economy', 'Confirmed');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_airline` . `booking` , CONSTRAINT `booking_ibfk_1` FOREIGN KEY (`flight_no`) REFERENCES `flight` (`flight_no`))

```
UPDATE FLIGHT SET base_fare = base_fare * 1.05 WHERE dep_date = '2026-11-05';
```

OK - 2 row(s) affected

```
DELETE FROM BOOKING WHERE booking_status = 'Waitlisted';
```

OK - 2 row(s) affected

```
SELECT flight_no, base_fare FROM FLIGHT WHERE dep_date = '2026-11-05' ORDER BY flight_no;
```

2 row(s):

flight_no	base_fare
SJ303	3570.00
ZA902	8505.00

```
SELECT COUNT(*) AS bookings_left FROM BOOKING;
```

1 row(s):

bookings_left
27

Set 06 - Salary Review

Human Resources (`eval_hr`) - tables: EMPLOYEE, DEPARTMENT

Task 1 - Retrieve data (5 marks)

> Table EMPLOYEE. Show emp_id, ename, job, salary of Developers, Salesmen or Testers (use IN) whose salary is between 45000 and 80000 (inclusive) and whose city is recorded (not NULL). Sort by salary (highest first), then emp_id. Show only the first 5 rows (LIMIT).

```
SELECT emp_id, ename, job, salary FROM EMPLOYEE WHERE job IN ('Developer','Salesman','Tester') AND
salary BETWEEN 45000 AND 80000 AND city IS NOT NULL ORDER BY salary DESC, emp_id LIMIT 5;
```

5 row(s):

emp_id	ename	job	salary
7003	Chirag Shah	Developer	78000.00
7004	Dolly Sinha	Developer	78000.00
7005	Eklavya Rana	Tester	52000.00
7008	Harini Subbu	Salesman	47000.00
7010	Jaya Prakash	Salesman	46000.00

Task 2 - Summarise data (5 marks)

> Table EMPLOYEE. (a) ONE row with: total_emps (count), with_comm (employees who earn a commission), max_sal, min_sal, avg_sal (ROUND 2). (b) For employees who belong to a department (dept_id is not NULL), for each dept_id show dept_id, headcount (count) and total_salary (sum). Keep only departments whose total_salary is more than 250000. Sort by dept_id.

```
SELECT COUNT(*) AS total_emps, COUNT(commission) AS with_comm, MAX(salary) AS max_sal, MIN(salary) AS
min_sal, ROUND(AVG(salary), 2) AS avg_sal FROM EMPLOYEE;
```

1 row(s):

total_emps	with_comm	max_sal	min_sal	avg_sal
22	4	120000.00	35000.00	64727.27

```
SELECT dept_id, COUNT(*) AS headcount, SUM(salary) AS total_salary FROM EMPLOYEE WHERE dept_id IS NOT
NULL GROUP BY dept_id HAVING SUM(salary) > 250000 ORDER BY dept_id;
```

3 row(s):

dept_id	headcount	total_salary
10	5	413000.00
20	5	286000.00
40	4	270000.00

Task 3 - Design & define (5 marks)

> ER fragment: a DEPARTMENT runs many PROJECTs; each project belongs to one department. (a) Create table PROJECT: proj_id INT primary key; proj_name VARCHAR(30) NOT NULL UNIQUE; dept_id INT NOT NULL, foreign key to DEPARTMENT(dept_id); budget DECIMAL(10,2) with CHECK (budget > 0); status VARCHAR(10) DEFAULT 'Planned'. (b) Add a column email VARCHAR(60) to EMPLOYEE. (c) Insert (1, 'Apollo', dept 10, budget 500000) listing proj_id, proj_name, dept_id, budget. Then insert (2, 'Zeus', dept 10, budget -5): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE PROJECT; and SELECT * FROM PROJECT ORDER BY proj_id;

```
CREATE TABLE PROJECT (proj_id INT PRIMARY KEY, proj_name VARCHAR(30) NOT NULL UNIQUE, dept_id INT NOT
NULL, budget DECIMAL(10,2) CHECK (budget > 0), status VARCHAR(10) DEFAULT 'Planned', FOREIGN KEY
(dept_id) REFERENCES DEPARTMENT(dept_id));
```

OK - 0 row(s) affected

```
ALTER TABLE EMPLOYEE ADD COLUMN email VARCHAR(60);
```

OK - 0 row(s) affected

```
INSERT INTO PROJECT (proj_id, proj_name, dept_id, budget) VALUES (1, 'Apollo', 10, 500000);
```

OK - 1 row(s) affected

```
INSERT INTO PROJECT (proj_id, proj_name, dept_id, budget) VALUES (2, 'Zeus', 10, -5);
```

Error number 3819: Check constraint 'project_chk_1' is violated.

```
DESCRIBE PROJECT;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
proj_id	int	NO	PRI	NULL	
proj_name	varchar(30)	NO	UNI	NULL	
dept_id	int	NO	MUL	NULL	
budget	decimal(10,2)	YES		NULL	
status	varchar(10)	YES		Planned	

```
SELECT * FROM PROJECT ORDER BY proj_id;
```

1 row(s):

proj_id	proj_name	dept_id	budget	status
1	Apollo	10	500000.00	Planned

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two employees (list the columns, leave commission out): (7023, 'Wasim Akhtar', 'Salesman', 20, 44000, '2026-09-01', 'Mumbai') and (7024, 'Yamini Rao', 'HR Executive', 30, 47000, '2026-09-15', 'Jaipur'). (b) Insert employee 7025 'Zubin Das', 'Tester', dept 99, salary 40000, hired '2026-09-20': department 99 does not exist, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Give every Support Agent a 10% raise. (d) Delete every employee who is not assigned to a department (dept_id IS NULL). (e) Verify with: SELECT emp_id, salary FROM EMPLOYEE WHERE job = 'Support Agent' ORDER BY emp_id; and SELECT COUNT(*) AS emps_left FROM EMPLOYEE;

```
INSERT INTO EMPLOYEE (emp_id, ename, job, dept_id, salary, hire_date, city) VALUES (7023, 'Wasim
Akhtar', 'Salesman', 20, 44000, '2026-09-01', 'Mumbai'), (7024, 'Yamini Rao', 'HR Executive', 30, 47000,
'2026-09-15', 'Jaipur');
```

OK - 2 row(s) affected

```
INSERT INTO EMPLOYEE (emp_id, ename, job, dept_id, salary, hire_date) VALUES (7025, 'Zubin Das',
'Tester', 99, 40000, '2026-09-20');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_hr` . `employee` , CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`dept_id`) REFERENCES `department` (`dept_id`))

```
UPDATE EMPLOYEE SET salary = salary * 1.10 WHERE job = 'Support Agent';
```

OK - 3 row(s) affected

```
DELETE FROM EMPLOYEE WHERE dept_id IS NULL;
```

OK - 1 row(s) affected

```
SELECT emp_id, salary FROM EMPLOYEE WHERE job = 'Support Agent' ORDER BY emp_id;
```

3 row(s):

emp_id	salary
7019	38500.00
7020	39600.00
7021	38500.00

```
SELECT COUNT(*) AS emps_left FROM EMPLOYEE;
```

1 row(s):

emps_left
23

Set 07 - Semester Planning

University (`eval_university`) - tables: STUDENT, EXAM_RESULT

Task 1 - Retrieve data (5 marks)

> Table STUDENT. Show each DISTINCT combination of dept and semester among students whose CGPA is below 8 or not yet assigned (NULL), and whose name does NOT start with 'A' (NOT LIKE). Sort by dept, then semester. (Mind the brackets.)

```
SELECT DISTINCT dept, semester FROM STUDENT WHERE (cgpa < 8 OR cgpa IS NULL) AND sname NOT LIKE 'A%'
ORDER BY dept, semester;
```

9 row(s):

dept	semester
CE	3
CE	5
CSE	5
ECE	5
ECE	7
IT	7
ME	3
ME	5
ME	7

Task 2 - Summarise data (5 marks)

> Table STUDENT. (a) ONE row with: cities (number of DISTINCT cities), max_cgpa, min_cgpa, no_email (number of students with no email recorded - hint: COUNT(*) minus COUNT(email)). (b) For each semester show semester, students (count) and total_due (sum of fee_due). Keep only semesters whose total_due is more than 15000. Sort by semester.

```
SELECT COUNT(DISTINCT city) AS cities, MAX(cgpa) AS max_cgpa, MIN(cgpa) AS min_cgpa, COUNT(*) -
COUNT(email) AS no_email FROM STUDENT;
```

1 row(s):

cities	max_cgpa	min_cgpa	no_email
11	9.35	6.45	2

```
SELECT semester, COUNT(*) AS students, SUM(fee_due) AS total_due FROM STUDENT GROUP BY semester HAVING
SUM(fee_due) > 15000 ORDER BY semester;
```

2 row(s):

semester	students	total_due
5	7	42500.00
7	5	25500.00

Task 3 - Design & define (5 marks)

> ER fragment: each STUDENT is allotted at most ONE hostel room (one-to-one). (a) Create table HOSTEL_ALLOCATION: alloc_id INT primary key; roll_no INT NOT NULL UNIQUE, foreign key to STUDENT(roll_no); room_no VARCHAR(5) NOT NULL; block CHAR(1) DEFAULT 'A'; rent DECIMAL(8,2) with CHECK (rent >= 0). (b) Change STUDENT.city to VARCHAR(50) (MODIFY). (c) Insert (1, 2101, '101', rent 4000) listing the columns and leaving block out. Then insert (2, 2101, '102', rent 4500): same student again, it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE HOSTEL_ALLOCATION; and SELECT * FROM HOSTEL_ALLOCATION ORDER BY alloc_id;

```
CREATE TABLE HOSTEL_ALLOCATION (alloc_id INT PRIMARY KEY, roll_no INT NOT NULL UNIQUE, room_no
VARCHAR(5) NOT NULL, block CHAR(1) DEFAULT 'A', rent DECIMAL(8,2) CHECK (rent >= 0), FOREIGN KEY
(roll_no) REFERENCES STUDENT(roll_no));
```

OK - 0 row(s) affected

```
ALTER TABLE STUDENT MODIFY city VARCHAR(50);
```

OK - 0 row(s) affected

```
INSERT INTO HOSTEL_ALLOCATION (alloc_id, roll_no, room_no, rent) VALUES (1, 2101, '101', 4000);
```

OK - 1 row(s) affected

```
INSERT INTO HOSTEL_ALLOCATION (alloc_id, roll_no, room_no, rent) VALUES (2, 2101, '102', 4500);
```

Error number 1062: Duplicate entry '2101' for key 'hostel_allocation.roll_no'

```
DESCRIBE HOSTEL_ALLOCATION;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
alloc_id	int	NO	PRI	NULL	
roll_no	int	NO	UNI	NULL	
room_no	varchar(5)	NO		NULL	
block	char(1)	YES		A	
rent	decimal(8,2)	YES		NULL	

```
SELECT * FROM HOSTEL_ALLOCATION ORDER BY alloc_id;
```

1 row(s):

alloc_id	roll_no	room_no	block	rent
1	2101	101	A	4000.00

Task 4 - Modify data (5 marks)

> (a) Insert student 2121, 'Rhea Kapoor', CSE, semester 5, city 'Delhi' - list the columns and leave cgpa, email and fee_due out (they must take their defaults / NULL). (b) Insert into EXAM_RESULT (result_id 25, roll_no 9999, 'Databases', 'End', marks 70, max_marks 100, exam_date '2026-05-12'): there is no student 9999, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Add 0.25 to the cgpa of every CSE student whose cgpa is below 9. (d) Delete from EXAM_RESULT all rows of exam_type 'Mid'. (e) Verify with: SELECT roll_no, cgpa, fee_due FROM STUDENT WHERE dept = 'CSE' ORDER BY roll_no; and SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;

```
INSERT INTO STUDENT (roll_no, sname, dept, semester, city) VALUES (2121, 'Rhea Kapoor', 'CSE', 5,
'Delhi');
```

OK - 1 row(s) affected

```
INSERT INTO EXAM_RESULT (result_id, roll_no, subject, exam_type, marks, max_marks, exam_date) VALUES
(25, 9999, 'Databases', 'End', 70, 100, '2026-05-12');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_university` . `exam_result` , CONSTRAINT `exam_result_ibfk_1` FOREIGN KEY (`roll_no`) REFERENCES `student` (`roll_no`))

```
UPDATE STUDENT SET cgpa = cgpa + 0.25 WHERE dept = 'CSE' AND cgpa < 9;
```

OK - 4 row(s) affected

```
DELETE FROM EXAM_RESULT WHERE exam_type = 'Mid';
```

OK - 5 row(s) affected

```
SELECT roll_no, cgpa, fee_due FROM STUDENT WHERE dept = 'CSE' ORDER BY roll_no;
```

7 row(s):

roll_no	cgpa	fee_due
2101	8.97	0.00
2102	9.10	0.00
2103	7.70	12500.00
2110	9.35	0.00
2113	8.55	0.00
2120	8.97	0.00
2121	NULL	0.00

```
SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;
```

1 row(s):

rows_left
19

Set 08 - Doctor Directory

Hospital (`eval_hospital`) - tables: DOCTOR, APPOINTMENT

Task 1 - Retrieve data (5 marks)

> Table DOCTOR. Show doctor_id, dname, specialization, consult_fee of doctors who (have no phone number on file OR whose specialization ends with 'ics') AND whose consult_fee is 1000 or less. Sort by consult_fee (highest first), then doctor_id. (Mind the brackets: AND binds tighter than OR.)

```
SELECT doctor_id, dname, specialization, consult_fee FROM DOCTOR WHERE (phone IS NULL OR specialization
LIKE '%ics') AND consult_fee <= 1000 ORDER BY consult_fee DESC, doctor_id;
```

5 row(s):

doctor_id	dname	specialization	consult_fee
103	Dr. Suresh Menon	Orthopedics	1000.00
107	Dr. Mohan Das	Orthopedics	800.00
104	Dr. Priya Nair	Pediatrics	700.00
106	Dr. Farah Khan	Dermatology	600.00
108	Dr. Kavita Joshi	Pediatrics	500.00

Task 2 - Summarise data (5 marks)

> Table DOCTOR. (a) ONE row with: doctors (count), with_phone (doctors having a phone), avg_fee (ROUND 2), senior (maximum experience_yrs), total_fee (sum of consult_fee). (b) For each specialization show specialization, doctors (count) and avg_fee (ROUND 2). Keep only specializations whose average fee is 800 or more. Sort by avg_fee (highest first), then specialization.

```
SELECT COUNT(*) AS doctors, COUNT(phone) AS with_phone, ROUND(AVG(consult_fee), 2) AS avg_fee,
MAX(experience_yrs) AS senior, SUM(consult_fee) AS total_fee FROM DOCTOR;
```

1 row(s):

doctors	with_phone	avg_fee	senior	total_fee
10	8	850.00	25	8500.00

```
SELECT specialization, COUNT(*) AS doctors, ROUND(AVG(consult_fee), 2) AS avg_fee FROM DOCTOR GROUP BY
specialization HAVING AVG(consult_fee) >= 800 ORDER BY avg_fee DESC, specialization;
```

3 row(s):

specialization	doctors	avg_fee
Neurology	1	1500.00
Cardiology	2	1050.00
Orthopedics	2	900.00

Task 3 - Design & define (5 marks)

> ER fragment: an APPOINTMENT can require many LAB_TESTs; each test belongs to one appointment. (a) Create table LAB_TEST: test_id INT primary key; appt_id INT NOT NULL, foreign key to APPOINTMENT(appt_id); test_name VARCHAR(30) NOT NULL; cost DECIMAL(8,2) with CHECK (cost >= 0); result VARCHAR(10) DEFAULT 'Pending'. (b) Add a column email VARCHAR(60) with a UNIQUE constraint to DOCTOR. (c) Insert (1, appt 4, 'X-Ray', 500) listing the columns and leaving result out. Then insert (2, appt 4, 'MRI', -1): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE LAB_TEST; and SELECT * FROM LAB_TEST ORDER BY test_id;

```
CREATE TABLE LAB_TEST (test_id INT PRIMARY KEY, appt_id INT NOT NULL, test_name VARCHAR(30) NOT NULL,
cost DECIMAL(8,2) CHECK (cost >= 0), result VARCHAR(10) DEFAULT 'Pending', FOREIGN KEY (appt_id)
REFERENCES APPOINTMENT(appt_id));
```

OK - 0 row(s) affected

```
ALTER TABLE DOCTOR ADD COLUMN email VARCHAR(60) UNIQUE;
```

OK - 0 row(s) affected

```
INSERT INTO LAB_TEST (test_id, appt_id, test_name, cost) VALUES (1, 4, 'X-Ray', 500);
```

OK - 1 row(s) affected

```
INSERT INTO LAB_TEST (test_id, appt_id, test_name, cost) VALUES (2, 4, 'MRI', -1);
```

Error number 3819: Check constraint 'lab_test_chk_1' is violated.

```
DESCRIBE LAB_TEST;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
test_id	int	NO	PRI	NULL	
appt_id	int	NO	MUL	NULL	
test_name	varchar(30)	NO		NULL	
cost	decimal(8,2)	YES		NULL	
result	varchar(10)	YES		Pending	

```
SELECT * FROM LAB_TEST ORDER BY test_id;
```

1 row(s):

test_id	appt_id	test_name	cost	result
1	4	X-Ray	500.00	Pending

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two doctors (list the columns): (111, 'Dr. Ravi Menon', 'Neurology', 8, 1300, NULL) and (112, 'Dr. Sita Rao', 'Pediatrics', 5, 650, '98100-11012'). (b) Insert another doctor with doctor_id 101: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Mark all Scheduled appointments of doctor 101 as Completed and set their fee_charged to 1200. (d) Delete doctor 111 from DOCTOR (this works). Then try to delete doctor 108: it must be rejected because appointments refer to the doctor. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (e) Verify with: SELECT doctor_id, dname FROM DOCTOR WHERE doctor_id > 108 ORDER BY doctor_id; and SELECT appt_id, status, fee_charged FROM APPOINTMENT WHERE doctor_id = 101 ORDER BY appt_id;

```
INSERT INTO DOCTOR (doctor_id, dname, specialization, experience_yrs, consult_fee, phone) VALUES (111,
'Dr. Ravi Menon', 'Neurology', 8, 1300, NULL), (112, 'Dr. Sita Rao', 'Pediatrics', 5, 650, '98100-
11012');
```

OK - 2 row(s) affected

```
INSERT INTO DOCTOR (doctor_id, dname, specialization, experience_yrs, consult_fee) VALUES (101, 'Dr.
Duplicate', 'Cardiology', 1, 500);
```

Error number 1062: Duplicate entry '101' for key 'doctor.PRIMARY'

```
UPDATE APPOINTMENT SET status = 'Completed', fee_charged = 1200 WHERE doctor_id = 101 AND status =
'Scheduled';
```

OK - 1 row(s) affected

```
DELETE FROM DOCTOR WHERE doctor_id = 111;
```

OK - 1 row(s) affected

```
DELETE FROM DOCTOR WHERE doctor_id = 108;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_hospital` . `appointment` , CONSTRAINT `appointment_ibfk_1` FOREIGN KEY (`doctor_id`) REFERENCES `doctor` (`doctor_id`))

```
SELECT doctor_id, dname FROM DOCTOR WHERE doctor_id > 108 ORDER BY doctor_id;
```

3 row(s):

doctor_id	dname
109	Dr. Amit Saxena
110	Dr. Leena Thomas
112	Dr. Sita Rao

```
SELECT appt_id, status, fee_charged FROM APPOINTMENT WHERE doctor_id = 101 ORDER BY appt_id;
```

3 row(s):

appt_id	status	fee_charged
1	Completed	1200.00
2	Completed	1200.00
22	Completed	1200.00

Set 09 - Order Desk

Retail Store (`eval_retail`) - tables: ORDERS, CUSTOMER

Task 1 - Retrieve data (5 marks)

> Table ORDERS. Show order_id, item, category, qty, status of orders whose status is Placed or Shipped (use IN) AND (category is 'Electronics' OR qty is 3 or more). Sort by order_id. (Mind the brackets.)

```
SELECT order_id, item, category, qty, status FROM ORDERS WHERE status IN ('Placed','Shipped') AND
(category = 'Electronics' OR qty >= 3) ORDER BY order_id;
```

5 row(s):

order_id	item	category	qty	status
1005	Leather Wallet	Accessories	3	Shipped
1010	Tablet	Electronics	1	Shipped
1018	Smartphone	Electronics	1	Placed
1023	Tablet	Electronics	2	Placed
1028	T-Shirt	Clothing	3	Placed

Task 2 - Summarise data (5 marks)

> Table ORDERS. (a) ONE row with: total_orders, units (sum of qty), max_price, min_price (unit_price), avg_price (ROUND 2). (b) For each cust_id show cust_id and orders (count). Keep only customers with 3 or more orders. Sort by cust_id.

```
SELECT COUNT(*) AS total_orders, SUM(qty) AS units, MAX(unit_price) AS max_price, MIN(unit_price) AS
min_price, ROUND(AVG(unit_price), 2) AS avg_price FROM ORDERS;
```

1 row(s):

total_orders	units	max_price	min_price	avg_price
28	54	55000.00	450.00	10117.86

```
SELECT cust_id, COUNT(*) AS orders FROM ORDERS GROUP BY cust_id HAVING COUNT(*) >= 3 ORDER BY cust_id;
```

4 row(s):

cust_id	orders
1	3
3	3
5	3
7	3

Task 3 - Design & define (5 marks)

> ER fragment: each ORDER has at most ONE SHIPMENT (one-to-one). (a) Create table SHIPMENT: ship_id INT primary key; order_id INT NOT NULL UNIQUE, foreign key to ORDERS(order_id); courier VARCHAR(20) NOT NULL; shipped_on DATE; delivered CHAR(1) DEFAULT 'N' with CHECK (delivered IN ('Y','N')). (b) Add a column remarks VARCHAR(100) to ORDERS. (c) Insert (1, order 1005, 'FastEx', '2026-06-08') listing the columns and leaving delivered out. Then insert (2, order 1005, 'QuickGo', '2026-06-09'): the same order again, it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE SHIPMENT; and SELECT * FROM SHIPMENT ORDER BY ship_id;

```
CREATE TABLE SHIPMENT (ship_id INT PRIMARY KEY, order_id INT NOT NULL UNIQUE, courier VARCHAR(20) NOT
NULL, shipped_on DATE, delivered CHAR(1) DEFAULT 'N' CHECK (delivered IN ('Y','N')), FOREIGN KEY
(order_id) REFERENCES ORDERS(order_id));
```

OK - 0 row(s) affected

```
ALTER TABLE ORDERS ADD COLUMN remarks VARCHAR(100);
```

OK - 0 row(s) affected

```
INSERT INTO SHIPMENT (ship_id, order_id, courier, shipped_on) VALUES (1, 1005, 'FastEx', '2026-06-08');
```

OK - 1 row(s) affected

```
INSERT INTO SHIPMENT (ship_id, order_id, courier, shipped_on) VALUES (2, 1005, 'QuickGo', '2026-06-09');
```

Error number 1062: Duplicate entry '1005' for key 'shipment.order_id'

```
DESCRIBE SHIPMENT;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
ship_id	int	NO	PRI	NULL	
order_id	int	NO	UNI	NULL	
courier	varchar(20)	NO		NULL	
shipped_on	date	YES		NULL	
delivered	char(1)	YES		N	

```
SELECT * FROM SHIPMENT ORDER BY ship_id;
```

1 row(s):

ship_id	order_id	courier	shipped_on	delivered
1	1005	FastEx	2026-06-08	N

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two customers (list the columns): (13, 'Mehul Doshi', 'Surat', 'Silver', '2026-09-01', 15000) and (14, 'Nisha Arora', 'Indore', NULL, '2026-09-05', 10000). (b) Insert another customer with cust_id 1: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set membership to 'Silver' for every customer who has NO membership and joined before '2025-04-01'. (d) Delete every Returned order. (e) Verify with: SELECT cust_id, membership FROM CUSTOMER WHERE cust_id IN (4, 8, 13, 14) ORDER BY cust_id; and SELECT COUNT(*) AS rows_left FROM ORDERS;

```
INSERT INTO CUSTOMER (cust_id, cname, city, membership, joined_on, credit_limit) VALUES (13, 'Mehul
Doshi', 'Surat', 'Silver', '2026-09-01', 15000), (14, 'Nisha Arora', 'Indore', NULL, '2026-09-05',
10000);
```

OK - 2 row(s) affected

```
INSERT INTO CUSTOMER (cust_id, cname) VALUES (1, 'Duplicate Person');
```

Error number 1062: Duplicate entry '1' for key 'customer.PRIMARY'

```
UPDATE CUSTOMER SET membership = 'Silver' WHERE membership IS NULL AND joined_on < '2025-04-01';
```

OK - 1 row(s) affected

```
DELETE FROM ORDERS WHERE status = 'Returned';
```

OK - 2 row(s) affected

```
SELECT cust_id, membership FROM CUSTOMER WHERE cust_id IN (4, 8, 13, 14) ORDER BY cust_id;
```

4 row(s):

cust_id	membership
4	Silver
8	NULL
13	Silver
14	NULL

```
SELECT COUNT(*) AS rows_left FROM ORDERS;
```

1 row(s):

rows_left
26

Set 10 - Transaction Audit

Bank (`eval_bank`) - tables: TXN, ACCOUNT

Task 1 - Retrieve data (5 marks)

> Table TXN. Show txn_id, acc_no, txn_type, amount, mode of transactions dated between '2026-07-10' and '2026-07-25' (inclusive) AND (made by 'Cheque' OR with an amount of 50000 or more). Sort by amount (highest first), then txn_id. (Mind the brackets.)

```
SELECT txn_id, acc_no, txn_type, amount, mode FROM TXN WHERE txn_date BETWEEN '2026-07-10' AND '2026-07-25' AND (mode = 'Cheque' OR amount >= 50000) ORDER BY amount DESC, txn_id;
```

6 row(s):

txn_id	acc_no	txn_type	amount	mode
14	5010	Deposit	500000.00	Transfer
16	5012	Deposit	250000.00	Transfer
23	5002	Deposit	90000.00	Transfer
11	5008	Deposit	60000.00	Cheque
22	5008	Withdraw	25000.00	Cheque
20	5014	Withdraw	15000.00	Cheque

Task 2 - Summarise data (5 marks)

> Table TXN. (a) ONE row with: txns (count), with_remarks (rows having remarks), accounts (number of DISTINCT acc_no), total_amount (sum), largest (max amount). (b) For each payment mode show mode, txns (count) and avg_amount (ROUND 2). Keep only modes with 6 or more transactions. Sort by mode.

```
SELECT COUNT(*) AS txns, COUNT(remarks) AS with_remarks, COUNT(DISTINCT acc_no) AS accounts, SUM(amount) AS total_amount, MAX(amount) AS largest FROM TXN;
```

1 row(s):

txns	with_remarks	accounts	total_amount	largest
28	13	12	1512200.00	500000.00

```
SELECT mode, COUNT(*) AS txns, ROUND(AVG(amount), 2) AS avg_amount FROM TXN GROUP BY mode HAVING COUNT(*) >= 6 ORDER BY mode;
```

3 row(s):

mode	txns	avg_amount
ATM	6	3700.00
Cheque	6	63333.33
Transfer	6	171666.67

Task 3 - Design & define (5 marks)

> ER fragment: an ACCOUNT can have many BENEFICIARY payees; every beneficiary account number is unique in the bank. (a) Create table BENEFICIARY: ben_id INT primary key; acc_no INT NOT NULL, foreign key to ACCOUNT(acc_no); ben_name VARCHAR(40) NOT NULL; ben_acc_no VARCHAR(20) NOT NULL UNIQUE; added_on DATE. (b) Add a column ref_no VARCHAR(20) to TXN. (c) Insert (1, account 5001, 'Rohan', '0099001') listing ben_id, acc_no, ben_name, ben_acc_no. Then insert (2, account 5002, 'Seema', '0099001'): same beneficiary number, it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE BENEFICIARY; and SELECT * FROM BENEFICIARY ORDER BY ben_id;

```
CREATE TABLE BENEFICIARY (ben_id INT PRIMARY KEY, acc_no INT NOT NULL, ben_name VARCHAR(40) NOT NULL, ben_acc_no VARCHAR(20) NOT NULL UNIQUE, added_on DATE, FOREIGN KEY (acc_no) REFERENCES ACCOUNT(acc_no));
```

OK - 0 row(s) affected

```
ALTER TABLE TXN ADD COLUMN ref_no VARCHAR(20);
```

OK - 0 row(s) affected

```
INSERT INTO BENEFICIARY (ben_id, acc_no, ben_name, ben_acc_no) VALUES (1, 5001, 'Rohan', '0099001');
```

OK - 1 row(s) affected

```
INSERT INTO BENEFICIARY (ben_id, acc_no, ben_name, ben_acc_no) VALUES (2, 5002, 'Seema', '0099001');
```

Error number 1062: Duplicate entry '0099001' for key 'beneficiary.ben_acc_no'

```
DESCRIBE BENEFICIARY;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
ben_id	int	NO	PRI	NULL	
acc_no	int	NO	MUL	NULL	
ben_name	varchar(40)	NO		NULL	
ben_acc_no	varchar(20)	NO	UNI	NULL	
added_on	date	YES		NULL	

```
SELECT * FROM BENEFICIARY ORDER BY ben_id;
```

1 row(s):

ben_id	acc_no	ben_name	ben_acc_no	added_on
1	5001	Rohan	0099001	NULL

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two transactions (list the columns): (29, 5001, 'Deposit', 8000, '2026-08-01', 'UPI', NULL) and (30, 5003, 'Withdraw', 1500, '2026-08-02', 'ATM', 'Cash needed'). (b) Insert a transaction (31, 5001, 'Deposit', NULL, '2026-08-03', 'Cash', NULL) with amount NULL: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set status to 'Active' for every Dormant account. (d) Delete every Closed account (it has no transactions, so this works). (e) Verify with: SELECT acc_no, status FROM ACCOUNT WHERE acc_no IN (5006, 5013) ORDER BY acc_no; and SELECT COUNT(*) AS txn_rows FROM TXN;

```
INSERT INTO TXN (txn_id, acc_no, txn_type, amount, txn_date, mode, remarks) VALUES (29, 5001, 'Deposit',
8000, '2026-08-01', 'UPI', NULL), (30, 5003, 'Withdraw', 1500, '2026-08-02', 'ATM', 'Cash needed');
```

OK - 2 row(s) affected

```
INSERT INTO TXN (txn_id, acc_no, txn_type, amount, txn_date, mode, remarks) VALUES (31, 5001, 'Deposit',
NULL, '2026-08-03', 'Cash', NULL);
```

Error number 1048: Column 'amount' cannot be null

```
UPDATE ACCOUNT SET status = 'Active' WHERE status = 'Dormant';
```

OK - 1 row(s) affected

```
DELETE FROM ACCOUNT WHERE status = 'Closed';
```

OK - 1 row(s) affected

```
SELECT acc_no, status FROM ACCOUNT WHERE acc_no IN (5006, 5013) ORDER BY acc_no;
```

1 row(s):

acc_no	status
5006	Active

```
SELECT COUNT(*) AS txn_rows FROM TXN;
```

1 row(s):

txn_rows
30

Set 11 - Passenger Manifest

Airline (`eval_airline`) - tables: BOOKING, FLIGHT

Task 1 - Retrieve data (5 marks)

> Table BOOKING. Show booking_id, passenger, age, seat_class of CONFIRMED bookings where (seat_class is 'Business' OR age is 60 or more OR age is below 12). Sort by age (oldest first), then booking_id. (Mind the brackets.)

```
SELECT booking_id, passenger, age, seat_class FROM BOOKING WHERE booking_status = 'Confirmed' AND
(seat_class = 'Business' OR age >= 60 OR age < 12) ORDER BY age DESC, booking_id;
```

10 row(s):

booking_id	passenger	age	seat_class
20	Omprakash Singh	71	Economy
15	Kiran Lal	66	Economy
5	Tushar Pandey	60	Economy
16	Abhijit Ghosh	55	Business
3	Vikas Oberoi	45	Business
19	Farida Shaikh	44	Business
7	Reena Dutta	36	Business
23	Deepa Nambiar	29	Business
26	Bunty Gill	9	Economy
8	Aarush Dutta	7	Economy

Task 2 - Summarise data (5 marks)

> Table BOOKING, CONFIRMED bookings only. (a) ONE row with: bookings (count), total_fare (sum), avg_bag (average baggage_kg, ROUND 2, NULLs ignored), avg_bag_all (ROUND 2, NULL counted as 0 - use IFNULL). (b) For ALL bookings (any status), for each flight_no show flight_no and bookings (count). Keep only flights with 3 or more bookings. Sort by bookings (highest first), then flight_no.

```
SELECT COUNT(*) AS bookings, SUM(fare_paid) AS total_fare, ROUND(AVG(baggage_kg), 2) AS avg_bag,
ROUND(AVG(IFNULL(baggage_kg, 0)), 2) AS avg_bag_all FROM BOOKING WHERE booking_status = 'Confirmed';
```

1 row(s):

bookings	total_fare	avg_bag	avg_bag_all
22	150660.00	19.41	15.00

```
SELECT flight_no, COUNT(*) AS bookings FROM BOOKING GROUP BY flight_no HAVING COUNT(*) >= 3 ORDER BY
bookings DESC, flight_no;
```

6 row(s):

flight_no	bookings
AN301	3
BW601	3
BW702	3
SJ101	3
SJ303	3
ZA902	3

Task 3 - Design & define (5 marks)

> ER fragment: each BOOKING has at most ONE MEAL_PREF (one-to-one). (a) Create table MEAL_PREF: pref_id INT primary key; booking_id INT NOT NULL UNIQUE, foreign key to BOOKING(booking_id); meal VARCHAR(10) DEFAULT 'Veg' with CHECK (meal IN ('Veg','Non-Veg','Vegan')); notes VARCHAR(50). (b) Change BOOKING.passenger to VARCHAR(60) NOT

NULL (MODIFY). (c) Insert (1, booking 1) listing pref_id, booking_id only. Then insert (2, booking 2, meal 'Jain'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE MEAL_PREF; and SELECT pref_id, booking_id, meal FROM MEAL_PREF ORDER BY pref_id;

```
CREATE TABLE MEAL_PREF (pref_id INT PRIMARY KEY, booking_id INT NOT NULL UNIQUE, meal VARCHAR(10)
DEFAULT 'Veg' CHECK (meal IN ('Veg','Non-Veg','Vegan')), notes VARCHAR(50), FOREIGN KEY (booking_id)
REFERENCES BOOKING(booking_id));
```

OK - 0 row(s) affected

```
ALTER TABLE BOOKING MODIFY passenger VARCHAR(60) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO MEAL_PREF (pref_id, booking_id) VALUES (1, 1);
```

OK - 1 row(s) affected

```
INSERT INTO MEAL_PREF (pref_id, booking_id, meal) VALUES (2, 2, 'Jain');
```

Error number 3819: Check constraint 'meal_pref_chk_1' is violated.

```
DESCRIBE MEAL_PREF;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
pref_id	int	NO	PRI	NULL	
booking_id	int	NO	UNI	NULL	
meal	varchar(10)	YES		Veg	
notes	varchar(50)	YES		NULL	

```
SELECT pref_id, booking_id, meal FROM MEAL_PREF ORDER BY pref_id;
```

1 row(s):

pref_id	booking_id	meal
1	1	Veg

Task 4 - Modify data (5 marks)

(a) Insert booking (27, 'AN402', 'Rekha Soni', 33, 'Economy', fare_paid NULL, 'Waitlisted', baggage NULL) listing all columns. (b) Insert another booking with booking_id 27: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Confirm waitlisted booking 10: set booking_status to 'Confirmed' and fare_paid to 3200. (d) Delete every Cancelled booking of flight 'AN503'. (e) Verify with: SELECT booking_id, booking_status, fare_paid FROM BOOKING WHERE booking_id IN (10, 27) ORDER BY booking_id; and SELECT COUNT(*) AS bookings_left FROM BOOKING;

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, fare_paid, booking_status,
baggage_kg) VALUES (27, 'AN402', 'Rekha Soni', 33, 'Economy', NULL, 'Waitlisted', NULL);
```

OK - 1 row(s) affected

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, booking_status) VALUES (27,
'AN402', 'Another Person', 40, 'Economy', 'Confirmed');
```

Error number 1062: Duplicate entry '27' for key 'booking.PRIMARY'

```
UPDATE BOOKING SET booking_status = 'Confirmed', fare_paid = 3200 WHERE booking_id = 10;
```

OK - 1 row(s) affected

```
DELETE FROM BOOKING WHERE booking_status = 'Cancelled' AND flight_no = 'AN503';
```

OK - 2 row(s) affected

```
SELECT booking_id, booking_status, fare_paid FROM BOOKING WHERE booking_id IN (10, 27) ORDER BY
booking_id;
```

2 row(s):

booking_id	booking_status	fare_paid
10	Confirmed	3200.00
27	Waitlisted	NULL

```
SELECT COUNT(*) AS bookings_left FROM BOOKING;
```

1 row(s):

bookings_left
25

Set 12 - Joiners & Cities

Human Resources (`eval_hr`) - tables: EMPLOYEE, DEPARTMENT

Task 1 - Retrieve data (5 marks)

> Table EMPLOYEE. Show emp_id, ename, city, hire_date of employees hired between '2020-01-01' and '2022-12-31' (inclusive) AND living in Jaipur or Pune (use IN). Sort by hire_date (earliest first), then emp_id.

```
SELECT emp_id, ename, city, hire_date FROM EMPLOYEE WHERE hire_date BETWEEN '2020-01-01' AND '2022-12-31' AND city IN ('Jaipur','Pune') ORDER BY hire_date, emp_id;
```

4 row(s):

emp_id	ename	city	hire_date
7012	Lata Pandey	Jaipur	2020-10-26
7003	Chirag Shah	Pune	2021-02-10
7010	Jaya Prakash	Pune	2021-07-07
7005	Eklavya Rana	Jaipur	2022-01-20

Task 2 - Summarise data (5 marks)

> Table EMPLOYEE. (a) ONE row with: total_emps (count), with_city (employees whose city is recorded), cities (number of DISTINCT cities), jobs (number of DISTINCT jobs). (b) For employees whose city is recorded, for each city show city and emps (count). Keep only cities with 3 or more employees. Sort by emps (highest first), then city.

```
SELECT COUNT(*) AS total_emps, COUNT(city) AS with_city, COUNT(DISTINCT city) AS cities, COUNT(DISTINCT job) AS jobs FROM EMPLOYEE;
```

1 row(s):

total_emps	with_city	cities	jobs
22	21	6	8

```
SELECT city, COUNT(*) AS emps FROM EMPLOYEE WHERE city IS NOT NULL GROUP BY city HAVING COUNT(*) >= 3 ORDER BY emps DESC, city;
```

5 row(s):

city	emps
Pune	6
Delhi	4
Jaipur	4
Bengaluru	3
Mumbai	3

Task 3 - Design & define (5 marks)

> ER fragment: an EMPLOYEE can raise many LEAVE_REQUESTs; each request belongs to one employee. (a) Create table LEAVE_REQUEST: leave_id INT primary key; emp_id INT NOT NULL, foreign key to EMPLOYEE(emp_id); from_date DATE NOT NULL; days INT with CHECK (days BETWEEN 1 AND 30); status VARCHAR(10) DEFAULT 'Pending'. (b) Add a column head_emp INT to DEPARTMENT. (c) Insert (1, employee 7001, '2026-10-10', 5 days) listing leave_id, emp_id, from_date, days. Then insert (2, employee 7002, '2026-10-12', 45 days): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE LEAVE_REQUEST; and SELECT * FROM LEAVE_REQUEST ORDER BY leave_id;

```
CREATE TABLE LEAVE_REQUEST (leave_id INT PRIMARY KEY, emp_id INT NOT NULL, from_date DATE NOT NULL, days INT CHECK (days BETWEEN 1 AND 30), status VARCHAR(10) DEFAULT 'Pending', FOREIGN KEY (emp_id) REFERENCES EMPLOYEE(emp_id));
```

OK - 0 row(s) affected

```
ALTER TABLE DEPARTMENT ADD COLUMN head_emp INT;
```

OK - 0 row(s) affected

```
INSERT INTO LEAVE_REQUEST (leave_id, emp_id, from_date, days) VALUES (1, 7001, '2026-10-10', 5);
```

OK - 1 row(s) affected

```
INSERT INTO LEAVE_REQUEST (leave_id, emp_id, from_date, days) VALUES (2, 7002, '2026-10-12', 45);
```

Error number 3819: Check constraint 'leave_request_chk_1' is violated.

```
DESCRIBE LEAVE_REQUEST;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
leave_id	int	NO	PRI	NULL	
emp_id	int	NO	MUL	NULL	
from_date	date	NO		NULL	
days	int	YES		NULL	
status	varchar(10)	YES		Pending	

```
SELECT * FROM LEAVE_REQUEST ORDER BY leave_id;
```

1 row(s):

leave_id	emp_id	from_date	days	status
1	7001	2026-10-10	5	Pending

Task 4 - Modify data (5 marks)

> (a) Insert employee (7023, 'Wasim Akhtar', 'Salesman', 20, 44000, '2026-09-01') listing emp_id, ename, job, dept_id, salary, hire_date only (commission and city stay NULL). (b) Insert another employee with emp_id 7001: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set city to 'Bengaluru' for every employee whose city is NULL. (d) Delete every Tester. (e) Verify with: SELECT emp_id, city FROM EMPLOYEE WHERE emp_id IN (7022, 7023) ORDER BY emp_id; and SELECT COUNT(*) AS emps_left FROM EMPLOYEE;

```
INSERT INTO EMPLOYEE (emp_id, ename, job, dept_id, salary, hire_date) VALUES (7023, 'Wasim Akhtar',
'Salesman', 20, 44000, '2026-09-01');
```

OK - 1 row(s) affected

```
INSERT INTO EMPLOYEE (emp_id, ename, job, salary, hire_date) VALUES (7001, 'Duplicate', 'Tester', 40000,
'2026-09-02');
```

Error number 1062: Duplicate entry '7001' for key 'employee.PRIMARY'

```
UPDATE EMPLOYEE SET city = 'Bengaluru' WHERE city IS NULL;
```

OK - 2 row(s) affected

```
DELETE FROM EMPLOYEE WHERE job = 'Tester';
```

OK - 1 row(s) affected

```
SELECT emp_id, city FROM EMPLOYEE WHERE emp_id IN (7022, 7023) ORDER BY emp_id;
```

2 row(s):

emp_id	city
7022	Bengaluru
7023	Bengaluru

```
SELECT COUNT(*) AS emps_left FROM EMPLOYEE;
```

1 row(s):

emps_left
22

Set 13 - Exam Analysis

University (`eval_university`) - tables: EXAM_RESULT, STUDENT

Task 1 - Retrieve data (5 marks)

> Table EXAM_RESULT. Show result_id, roll_no, subject, marks of End exams (exam_type = 'End') in the subjects Databases or Networks (use IN) where marks are between 60 and 90 (inclusive). Sort by marks (highest first), then result_id. Show only the first 5 rows (LIMIT).

```
SELECT result_id, roll_no, subject, marks FROM EXAM_RESULT WHERE exam_type = 'End' AND subject IN
('Databases','Networks') AND marks BETWEEN 60 AND 90 ORDER BY marks DESC, result_id LIMIT 5;
```

5 row(s):

result_id	roll_no	subject	marks
16	2110	Networks	90
1	2101	Databases	88
4	2102	Networks	85
18	2113	Databases	81
11	2106	Databases	79

Task 2 - Summarise data (5 marks)

> Table EXAM_RESULT, End exams only (exam_type = 'End'). (a) ONE row with: rows_total (all End rows), written (rows where marks is not NULL), avg_written (average marks ignoring NULL, ROUND 2), avg_all (average counting absentees as 0 - use IFNULL, ROUND 2), best (highest marks). (b) For each subject show subject, attempts (number of students who actually wrote it) and avg_marks (ROUND 1). Keep only subjects whose average is above 65. Sort by avg_marks (highest first).

```
SELECT COUNT(*) AS rows_total, COUNT(marks) AS written, ROUND(AVG(marks), 2) AS avg_written,
ROUND(AVG(IFNULL(marks, 0)), 2) AS avg_all, MAX(marks) AS best FROM EXAM_RESULT WHERE exam_type = 'End';
```

1 row(s):

rows_total	written	avg_written	avg_all	best
19	18	71.72	67.95	95

```
SELECT subject, COUNT(marks) AS attempts, ROUND(AVG(marks), 1) AS avg_marks FROM EXAM_RESULT WHERE
exam_type = 'End' GROUP BY subject HAVING AVG(marks) > 65 ORDER BY avg_marks DESC;
```

2 row(s):

subject	attempts	avg_marks
Databases	6	82.2
Networks	6	73.7

Task 3 - Design & define (5 marks)

> ER fragment: a COURSE is an independent entity (no relationship to the existing tables is needed yet). (a) Create table COURSE: course_code VARCHAR(8) primary key; title VARCHAR(40) NOT NULL UNIQUE; credits INT with DEFAULT 3 and CHECK (credits BETWEEN 1 AND 5); dept VARCHAR(10). (b) Add a column grade CHAR(2) to EXAM_RESULT. (c) Insert ('CS201', 'Databases') into COURSE listing course_code and title only. Then insert ('CS202', 'Networks') with credits 9: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE COURSE; and SELECT * FROM COURSE ORDER BY course_code;

```
CREATE TABLE COURSE (course_code VARCHAR(8) PRIMARY KEY, title VARCHAR(40) NOT NULL UNIQUE, credits INT
DEFAULT 3 CHECK (credits BETWEEN 1 AND 5), dept VARCHAR(10));
```

OK - 0 row(s) affected

```
ALTER TABLE EXAM_RESULT ADD COLUMN grade CHAR(2);
```

OK - 0 row(s) affected

```
INSERT INTO COURSE (course_code, title) VALUES ('CS201', 'Databases');
```

OK - 1 row(s) affected

```
INSERT INTO COURSE (course_code, title, credits) VALUES ('CS202', 'Networks', 9);
```

Error number 3819: Check constraint 'course_chk_1' is violated.

```
DESCRIBE COURSE;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
course_code	varchar(8)	NO	PRI	NULL	
title	varchar(40)	NO	UNI	NULL	
credits	int	YES		3	
dept	varchar(10)	YES		NULL	

```
SELECT * FROM COURSE ORDER BY course_code;
```

1 row(s):

course_code	title	credits	dept
CS201	Databases	3	NULL

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two rows to EXAM_RESULT (list the columns): (25, 2104, 'Databases', 'End', 67, 100, '2026-05-12') and (26, 2106, 'Networks', 'End', 72, 100, '2026-05-14'). (b) Insert another row with result_id 25: it must be rejected (duplicate primary key). Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Give 5 grace marks to every End exam of 'Mathematics' where marks are below 50. (d) Delete the 'Mid' exam rows where marks are below 30. (e) Verify with: SELECT result_id, roll_no, marks FROM EXAM_RESULT WHERE subject = 'Mathematics' AND exam_type = 'End' ORDER BY result_id; and SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;

```
INSERT INTO EXAM_RESULT (result_id, roll_no, subject, exam_type, marks, max_marks, exam_date) VALUES
(25, 2104, 'Databases', 'End', 67, 100, '2026-05-12'), (26, 2106, 'Networks', 'End', 72, 100, '2026-05-
14');
```

OK - 2 row(s) affected

```
INSERT INTO EXAM_RESULT (result_id, roll_no, subject, exam_type, marks, max_marks, exam_date) VALUES
(25, 2107, 'Databases', 'End', 50, 100, '2026-05-12');
```

Error number 1062: Duplicate entry '25' for key 'exam_result.PRIMARY'

```
UPDATE EXAM_RESULT SET marks = marks + 5 WHERE subject = 'Mathematics' AND exam_type = 'End' AND marks <
50;
```

OK - 2 row(s) affected

```
DELETE FROM EXAM_RESULT WHERE exam_type = 'Mid' AND marks < 30;
```

OK - 2 row(s) affected

```
SELECT result_id, roll_no, marks FROM EXAM_RESULT WHERE subject = 'Mathematics' AND exam_type = 'End'
ORDER BY result_id;
```

6 row(s):

result_id	roll_no	marks
6	2103	54
7	2104	71
9	2105	43
13	2107	74
14	2108	55
19	2113	69

```
SELECT COUNT(*) AS rows_left FROM EXAM_RESULT;
```

1 row(s):

rows_left
24

Set 14 - Patient Age Groups

Hospital (`eval_hospital`) - tables: APPOINTMENT, DOCTOR

Task 1 - Retrieve data (5 marks)

> Table APPOINTMENT. Show appt_id, patient_name, age, status of patients who (are children - the patient_name starts with 'Baby' - OR are older than 65) AND whose status is not 'Cancelled'. Sort by age (youngest first), then appt_id. (Mind the brackets.)

```
SELECT appt_id, patient_name, age, status FROM APPOINTMENT WHERE (patient_name LIKE 'Baby%' OR age > 65)
AND status <> 'Cancelled' ORDER BY age, appt_id;
```

7 row(s):

appt_id	patient_name	age	status
14	Baby Ishaan	2	Completed
6	Baby Aryan	3	Completed
26	Baby Zara	4	Completed
7	Baby Myra	5	Completed
15	Baby Anaya	7	Scheduled
9	Lata Verma	71	Scheduled
17	Kamla Bai	74	Completed

Task 2 - Summarise data (5 marks)

> Table APPOINTMENT. (a) For COMPLETED appointments, ONE row with: visits (count), min_fee, max_fee, avg_fee (ROUND 2), total_fee. (b) For ALL appointments, for each status show status, appts (count) and avg_age (ROUND 1). Keep only statuses with 3 or more appointments. Sort by appts (highest first), then status.

```
SELECT COUNT(*) AS visits, MIN(fee_charged) AS min_fee, MAX(fee_charged) AS max_fee,
ROUND(AVG(fee_charged), 2) AS avg_fee, SUM(fee_charged) AS total_fee FROM APPOINTMENT WHERE status =
'Completed';
```

1 row(s):

visits	min_fee	max_fee	avg_fee	total_fee
19	350.00	1500.00	776.32	14750.00

```
SELECT status, COUNT(*) AS appts, ROUND(AVG(age), 1) AS avg_age FROM APPOINTMENT GROUP BY status HAVING
COUNT(*) >= 3 ORDER BY appts DESC, status;
```

3 row(s):

status	appts	avg_age
Completed	19	35.6
Scheduled	4	46.8
Cancelled	3	48.0

Task 3 - Design & define (5 marks)

> ER fragment: each APPOINTMENT has at most ONE BILL (one-to-one). (a) Create table BILL: bill_id INT primary key; appt_id INT NOT NULL UNIQUE, foreign key to APPOINTMENT(appt_id); amount DECIMAL(8,2) NOT NULL with CHECK (amount > 0); paid CHAR(1) DEFAULT 'N'. (b) Change DOCTOR.dname to VARCHAR(60) NOT NULL (MODIFY). (c) Insert (1, appt 1, 1200) listing the columns and leaving paid out. Then insert (2, appt 1, 900): the same appointment again, it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE BILL; and SELECT * FROM BILL ORDER BY bill_id;

```
CREATE TABLE BILL (bill_id INT PRIMARY KEY, appt_id INT NOT NULL UNIQUE, amount DECIMAL(8,2) NOT NULL
CHECK (amount > 0), paid CHAR(1) DEFAULT 'N', FOREIGN KEY (appt_id) REFERENCES APPOINTMENT(appt_id));
```

OK - 0 row(s) affected

```
ALTER TABLE DOCTOR MODIFY dname VARCHAR(60) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO BILL (bill_id, appt_id, amount) VALUES (1, 1, 1200);
```

OK - 1 row(s) affected

```
INSERT INTO BILL (bill_id, appt_id, amount) VALUES (2, 1, 900);
```

Error number 1062: Duplicate entry '1' for key 'bill.appt_id'

```
DESCRIBE BILL;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
bill_id	int	NO	PRI	NULL	
appt_id	int	NO	UNI	NULL	
amount	decimal(8,2)	NO		NULL	
paid	char(1)	YES		N	

```
SELECT * FROM BILL ORDER BY bill_id;
```

1 row(s):

bill_id	appt_id	amount	paid
1	1	1200.00	N

Task 4 - Modify data (5 marks)

> (a) Insert appointment 27: doctor 109, 'Ravi Teja', age 36, 'M', date '2026-08-16', status 'Completed', fee_charged 400 (list all columns). (b) Insert one more appointment with appt_id 27: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Increase fee_charged by 10% for every Completed appointment of doctor 109. (d) Delete all Scheduled appointments dated before '2026-08-13'. (e) Verify with: SELECT appt_id, fee_charged FROM APPOINTMENT WHERE doctor_id = 109 ORDER BY appt_id; and SELECT appt_id, status FROM APPOINTMENT WHERE status = 'Scheduled' ORDER BY appt_id;

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date, status, fee_charged)
VALUES (27, 109, 'Ravi Teja', 36, 'M', '2026-08-16', 'Completed', 400);
```

OK - 1 row(s) affected

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date) VALUES (27, 110,
'Another Patient', 40, 'M', '2026-08-16');
```

Error number 1062: Duplicate entry '27' for key 'appointment.PRIMARY'

```
UPDATE APPOINTMENT SET fee_charged = fee_charged * 1.10 WHERE doctor_id = 109 AND status = 'Completed';
```

OK - 5 row(s) affected

```
DELETE FROM APPOINTMENT WHERE status = 'Scheduled' AND appt_date < '2026-08-13';
```

OK - 3 row(s) affected

```
SELECT appt_id, fee_charged FROM APPOINTMENT WHERE doctor_id = 109 ORDER BY appt_id;
```

5 row(s):

appt_id	fee_charged
16	440.00
17	440.00
18	440.00
19	385.00
27	440.00

```
SELECT appt_id, status FROM APPOINTMENT WHERE status = 'Scheduled' ORDER BY appt_id;
```

1 row(s):

appt_id	status
22	Scheduled

Set 15 - Category Performance

Retail Store (`eval_retail`) - tables: ORDERS, CUSTOMER

Task 1 - Retrieve data (5 marks)

> Table ORDERS. Show each DISTINCT category and item (two columns) among orders whose unit_price is between 1000 and 10000 (inclusive) and whose status is not 'Cancelled'. Sort by category, then item.

```
SELECT DISTINCT category, item FROM ORDERS WHERE unit_price BETWEEN 1000 AND 10000 AND status <>
'Cancelled' ORDER BY category, item;
```

9 row(s):

category	item
Accessories	Leather Wallet
Accessories	Sports Watch
Accessories	Sunglasses
Clothing	Denim Jeans
Clothing	Hoodie
Electronics	Bluetooth Speaker
Electronics	Headphones
Footwear	Formal Shoes
Footwear	Running Shoes

Task 2 - Summarise data (5 marks)

> Table CUSTOMER. (a) ONE row with: customers (count), with_membership (customers having a membership), avg_credit (ROUND 2), max_credit, total_credit (sum of credit_limit). (b) For each city show city and customers (count). Keep only cities with 2 or more customers. Sort by customers (highest first), then city.

```
SELECT COUNT(*) AS customers, COUNT(membership) AS with_membership, ROUND(AVG(credit_limit), 2) AS
avg_credit, MAX(credit_limit) AS max_credit, SUM(credit_limit) AS total_credit FROM CUSTOMER;
```

1 row(s):

customers	with_membership	avg_credit	max_credit	total_credit
12	10	42916.67	120000.00	515000.00

```
SELECT city, COUNT(*) AS customers FROM CUSTOMER GROUP BY city HAVING COUNT(*) >= 2 ORDER BY customers
DESC, city;
```

2 row(s):

city	customers
Delhi	3
Jaipur	2

Task 3 - Design & define (5 marks)

> ER fragment: COUPON is an independent entity. (a) Create table COUPON: code VARCHAR(10) primary key; discount_pct DECIMAL(4,1) NOT NULL with CHECK (discount_pct BETWEEN 0 AND 50); min_order DECIMAL(9,2) DEFAULT 0; expires_on DATE NOT NULL. (b) Change CUSTOMER.cname to VARCHAR(60) NOT NULL (MODIFY). (c) Insert ('NEW10', 10, '2026-12-31') listing code, discount_pct, expires_on. Then insert ('BIG99', 99, '2026-12-31'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE COUPON; and SELECT * FROM COUPON ORDER BY code;

```
CREATE TABLE COUPON (code VARCHAR(10) PRIMARY KEY, discount_pct DECIMAL(4,1) NOT NULL CHECK
(discount_pct BETWEEN 0 AND 50), min_order DECIMAL(9,2) DEFAULT 0, expires_on DATE NOT NULL);
```

OK - 0 row(s) affected

```
ALTER TABLE CUSTOMER MODIFY cname VARCHAR(60) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO COUPON (code, discount_pct, expires_on) VALUES ('NEW10', 10, '2026-12-31');
```

OK - 1 row(s) affected

```
INSERT INTO COUPON (code, discount_pct, expires_on) VALUES ('BIG99', 99, '2026-12-31');
```

Error number 3819: Check constraint 'coupon_chk_1' is violated.

```
DESCRIBE COUPON;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
code	varchar(10)	NO	PRI	NULL	
discount_pct	decimal(4,1)	NO		NULL	
min_order	decimal(9,2)	YES		0.00	
expires_on	date	NO		NULL	

```
SELECT * FROM COUPON ORDER BY code;
```

1 row(s):

code	discount_pct	min_order	expires_on
NEW10	10.0	0.00	2026-12-31

Task 4 - Modify data (5 marks)

> (a) Insert order (1029, customer 12, 'Hoodie', 'Clothing', 1, 2500, '2026-07-17', 'Placed') listing the columns and leaving discount_pct out (it becomes NULL). (b) Insert an order 1030 for customer 1 with qty NULL (other values of your choice): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set discount_pct to 5 for every Electronics order whose discount_pct is currently NULL. (d) Delete every Delivered order placed before '2026-06-10'. (e) Verify with: SELECT order_id, discount_pct FROM ORDERS WHERE category = 'Electronics' ORDER BY order_id; and SELECT COUNT(*) AS rows_left FROM ORDERS;

```
INSERT INTO ORDERS (order_id, cust_id, item, category, qty, unit_price, order_date, status) VALUES
(1029, 12, 'Hoodie', 'Clothing', 1, 2500, '2026-07-17', 'Placed');
```

OK - 1 row(s) affected

```
INSERT INTO ORDERS (order_id, cust_id, item, category, qty, unit_price, order_date, status) VALUES
(1030, 1, 'Mouse', 'Electronics', NULL, 800, '2026-07-17', 'Placed');
```

Error number 1048: Column 'qty' cannot be null

```
UPDATE ORDERS SET discount_pct = 5 WHERE category = 'Electronics' AND discount_pct IS NULL;
```

OK - 2 row(s) affected

```
DELETE FROM ORDERS WHERE status = 'Delivered' AND order_date < '2026-06-10';
```

OK - 4 row(s) affected

```
SELECT order_id, discount_pct FROM ORDERS WHERE category = 'Electronics' ORDER BY order_id;
```

8 row(s):

order_id	discount_pct
1007	5.0
1010	5.0
1014	8.0
1015	8.0
1018	5.0
1021	2.5
1023	10.0
1027	5.0

```
SELECT COUNT(*) AS rows_left FROM ORDERS;
```

1 row(s):

rows_left
25