RDBMS Lab Evaluation

Batch 2 — Sets 16–30 · 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 16 – Deposit Insights
Set 17 – Fare Analysis
Set 18 – Commission Report
Set 19 – Fee Defaulters & Cities
Set 20 – Revenue Desk
Set 21 – Customer Loyalty
Set 22 – Branch Performance
Set 23 – Class & Baggage
Set 24 – Department Budgets
Set 25 – Department Scoreboard
Set 26 – Specialty Overview
Set 27 – Pricing Review
Set 28 – Cash Flow
Set 29 – Operations Control
Set 30 – Seniority & Tenure

Set 16 - Deposit Insights

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

Task 1 - Retrieve data (5 marks)

> Table ACCOUNT. Show acc_no, holder_name, branch_city, opened_on of accounts that have a nominee registered, are held in Jaipur, Pune or Delhi (use IN) and were opened on or after '2020-01-01'. Sort by opened_on (oldest first), then acc_no.

```
SELECT acc_no, holder_name, branch_city, opened_on FROM ACCOUNT WHERE nominee IS NOT NULL AND
branch_city IN ('Jaipur','Pune','Delhi') AND opened_on >= '2020-01-01' ORDER BY opened_on, acc_no;
```

4 row(s):

acc_no	holder_name	branch_city	opened_on
5004	Divya Narang	Delhi	2020-10-18
5007	Gaurav Anand	Delhi	2021-09-09
5003	Chetan Pawar	Pune	2022-01-05
5009	Imran Sheikh	Pune	2023-06-25

Task 2 - Summarise data (5 marks)

> Table TXN. (a) For Deposits only, ONE row with: deposits (count), total_deposited, avg_deposit (ROUND 2), largest, smallest. (b) For ALL transactions, for each acc_no show acc_no, txns (count) and total_amount (sum). Keep only accounts whose total_amount is more than 100000. Sort by total_amount (highest first).

```
SELECT COUNT(*) AS deposits, SUM(amount) AS total_deposited, ROUND(AVG(amount), 2) AS avg_deposit,
MAX(amount) AS largest, MIN(amount) AS smallest FROM TXN WHERE txn_type = 'Deposit';
```

1 row(s):

deposits	total_deposited	avg_deposit	largest	smallest
17	1317500.00	77500.00	500000.00	1000.00

```
SELECT acc_no, COUNT(*) AS txns, SUM(amount) AS total_amount FROM TXN GROUP BY acc_no HAVING SUM(amount)
> 100000 ORDER BY total_amount DESC;
```

4 row(s):

acc_no	txns	total_amount
5010	1	500000.00
5002	3	320000.00
5012	1	250000.00
5005	2	170000.00

Task 3 - Design & define (5 marks)

> ER fragment: each ACCOUNT has at most ONE ATM_CARD (one-to-one). (a) Create table ATM_CARD: card_no CHAR(16) primary key; acc_no INT NOT NULL UNIQUE, foreign key to ACCOUNT(acc_no); pin_set CHAR(1) DEFAULT 'N'; daily_limit DECIMAL(8,2) NOT NULL with CHECK (daily_limit BETWEEN 1000 AND 100000); issued_on DATE NOT NULL. (b) Change ACCOUNT.holder_name to VARCHAR(60) NOT NULL (MODIFY). (c) Insert ('4000000000000001', account 5001, daily_limit 25000, issued_on '2026-09-01') listing the columns and leaving pin_set out. Then insert ('4000000000000002', account 5002, daily_limit 500, issued_on '2026-09-01'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE ATM_CARD; and SELECT card_no, acc_no, pin_set, daily_limit FROM ATM_CARD ORDER BY card_no;

```
CREATE TABLE ATM_CARD (card_no CHAR(16) PRIMARY KEY, acc_no INT NOT NULL UNIQUE, pin_set CHAR(1) DEFAULT
'N', daily_limit DECIMAL(8,2) NOT NULL CHECK (daily_limit BETWEEN 1000 AND 100000), issued_on DATE NOT
NULL, FOREIGN KEY (acc_no) REFERENCES ACCOUNT(acc_no));
```

OK - 0 row(s) affected

```
ALTER TABLE ACCOUNT MODIFY holder_name VARCHAR(60) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO ATM_CARD (card_no, acc_no, daily_limit, issued_on) VALUES ('4000000000000001', 5001, 25000,
'2026-09-01');
```

OK - 1 row(s) affected

```
INSERT INTO ATM_CARD (card_no, acc_no, daily_limit, issued_on) VALUES ('4000000000000002', 5002, 500,
'2026-09-01');
```

Error number 3819: Check constraint 'atm_card_chk_1' is violated.

```
DESCRIBE ATM_CARD;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
card_no	char(16)	NO	PRI	NULL	
acc_no	int	NO	UNI	NULL	
pin_set	char(1)	YES		N	
daily_limit	decimal(8,2)	NO		NULL	
issued_on	date	NO		NULL	

```
SELECT card_no, acc_no, pin_set, daily_limit FROM ATM_CARD ORDER BY card_no;
```

1 row(s):

card_no	acc_no	pin_set	daily_limit
4000000000000001	5001	N	25000.00

Task 4 - Modify data (5 marks)

> (a) Insert account (5015, 'Omkar Pillai', 'Chennai', 'Savings', 5000, '2026-10-01', 'Active', 'Meena Pillai') listing all columns. (b) Insert another account with acc_no 5001: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set remarks to 'Verified' for every transaction that has NO remarks and an amount of 20000 or more. (d) Delete all transactions of account 5011, and then delete account 5011 itself. (e) Verify with: SELECT txn_id, remarks FROM TXN WHERE remarks = 'Verified' ORDER BY txn_id; and SELECT COUNT(*) AS accounts_left FROM ACCOUNT;

```
INSERT INTO ACCOUNT (acc_no, holder_name, branch_city, acc_type, balance, opened_on, status, nominee)
VALUES (5015, 'Omkar Pillai', 'Chennai', 'Savings', 5000, '2026-10-01', 'Active', 'Meena Pillai');
```

OK - 1 row(s) affected

```
INSERT INTO ACCOUNT (acc_no, holder_name, branch_city, acc_type, opened_on) VALUES (5001, 'Duplicate',
'Delhi', 'Savings', '2026-10-01');
```

Error number 1062: Duplicate entry '5001' for key 'account.PRIMARY'

```
UPDATE TXN SET remarks = 'Verified' WHERE remarks IS NULL AND amount >= 20000;
```

OK - 1 row(s) affected

```
DELETE FROM TXN WHERE acc_no = 5011;
```

OK - 2 row(s) affected

```
DELETE FROM ACCOUNT WHERE acc_no = 5011;
```

OK - 1 row(s) affected

```
SELECT txn_id, remarks FROM TXN WHERE remarks = 'Verified' ORDER BY txn_id;
```

1 row(s):

txn_id	remarks
1	Verified

```
SELECT COUNT(*) AS accounts_left FROM ACCOUNT;
```

1 row(s):

accounts_left
14

Set 17 - Fare Analysis

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

Task 1 - Retrieve data (5 marks)

> Table FLIGHT. Show flight_no, source, destination, dep_date of flights whose destination ends with the letter 'i' (LIKE '%i') OR whose source is 'Kolkata'. Sort by dep_date, then flight_no.

```
SELECT flight_no, source, destination, dep_date FROM FLIGHT WHERE destination LIKE '%i' OR source =
'Kolkata' ORDER BY dep_date, flight_no;
```

6 row(s):

flight_no	source	destination	dep_date
SJ101	Delhi	Mumbai	2026-11-01
SJ202	Mumbai	Chennai	2026-11-01
AN402	Jaipur	Delhi	2026-11-02
BW702	Kolkata	Delhi	2026-11-04
ZA801	Mumbai	Delhi	2026-11-04
ZA902	Delhi	Chennai	2026-11-05

Task 2 - Summarise data (5 marks)

> Table BOOKING. (a) ONE row with: total_bookings, paid (rows having a fare), with_bag (rows having baggage_kg), total_fare (sum of fare_paid). (b) For each booking_status show booking_status, bookings (count) and total_fare (sum of fare_paid; show 0 instead of NULL - use IFNULL). Sort by bookings (highest first), then booking_status.

```
SELECT COUNT(*) AS total_bookings, COUNT(fare_paid) AS paid, COUNT(baggage_kg) AS with_bag,
SUM(fare_paid) AS total_fare FROM BOOKING;
```

1 row(s):

total_bookings	paid	with_bag	total_fare
26	24	18	156460.00

```
SELECT booking_status, COUNT(*) AS bookings, IFNULL(SUM(fare_paid), 0) AS total_fare FROM BOOKING GROUP
BY booking_status ORDER BY bookings DESC, booking_status;
```

3 row(s):

booking_status	bookings	total_fare
Confirmed	22	150660.00
Cancelled	2	5800.00
Waitlisted	2	0.00

Task 3 - Design & define (5 marks)

> ER fragment: AIRPORT is an independent entity. (a) Create table AIRPORT: code CHAR(3) primary key; city VARCHAR(30) NOT NULL UNIQUE; terminals INT DEFAULT 1 with CHECK (terminals BETWEEN 1 AND 5). (b) Rename FLIGHT.source to origin (RENAME COLUMN). (c) Insert ('DEL', 'Delhi') listing code and city. Then insert ('DLI', 'Delhi'): same city, it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE AIRPORT; and SELECT * FROM AIRPORT ORDER BY code;

```
CREATE TABLE AIRPORT (code CHAR(3) PRIMARY KEY, city VARCHAR(30) NOT NULL UNIQUE, terminals INT DEFAULT
1 CHECK (terminals BETWEEN 1 AND 5));
```

OK - 0 row(s) affected

```
ALTER TABLE FLIGHT RENAME COLUMN source TO origin;
```

OK - 0 row(s) affected

```
INSERT INTO AIRPORT (code, city) VALUES ('DEL', 'Delhi');
```

OK - 1 row(s) affected

```
INSERT INTO AIRPORT (code, city) VALUES ('DLI', 'Delhi');
```

Error number 1062: Duplicate entry 'Delhi' for key 'airport.city'

```
DESCRIBE AIRPORT;
```

3 row(s):

Field	Type	Null	Key	Default	Extra
code	char(3)	NO	PRI	NULL	
city	varchar(30)	NO	UNI	NULL	
terminals	int	YES		1	

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

1 row(s):

code	city	terminals
DEL	Delhi	1

Task 4 - Modify data (5 marks)

> (a) Insert flight ('BW803', 'BlueWing', 'Jaipur', 'Mumbai', '2026-11-06', 140, 4200) listing the columns and leaving status out (it must default to 'On Time'). Note: after Task 3(b) the column formerly called source is now called origin. (b) Insert booking (27, flight 'ZZ000', ...) 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) Reduce base_fare by 10% for every Delayed flight. (d) Delete every Waitlisted booking. (e) Verify with: SELECT flight_no, status FROM FLIGHT WHERE flight_no = 'BW803'; SELECT flight_no, base_fare FROM FLIGHT WHERE status = 'Delayed' ORDER BY flight_no; and SELECT COUNT(*) AS bookings_left FROM BOOKING;

```
INSERT INTO FLIGHT (flight_no, airline, origin, destination, dep_date, total_seats, base_fare) VALUES
('BW803', 'BlueWing', 'Jaipur', 'Mumbai', '2026-11-06', 140, 4200);
```

OK - 1 row(s) affected

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, booking_status) VALUES (27,
'ZZ000', '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 * 0.90 WHERE status = 'Delayed';
```

OK - 2 row(s) affected

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

OK - 2 row(s) affected

```
SELECT flight_no, status FROM FLIGHT WHERE flight_no = 'BW803';
```

1 row(s):

flight_no	status
BW803	On Time

```
SELECT flight_no, base_fare FROM FLIGHT WHERE status = 'Delayed' ORDER BY flight_no;
```

2 row(s):

flight_no	base_fare
BW702	4950.00
SJ202	4680.00

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

1 row(s):

bookings_left
24

Set 18 - Commission Report

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

Task 1 - Retrieve data (5 marks)

> Table EMPLOYEE. Show emp_id, ename, job, salary, commission of employees who (earn a commission OR are Managers) AND have a salary of 45000 or more. Sort by salary (highest first), then emp_id. (Mind the brackets.)

```
SELECT emp_id, ename, job, salary, commission FROM EMPLOYEE WHERE (commission IS NOT NULL OR job =
'Manager') AND salary >= 45000 ORDER BY salary DESC, emp_id;
```

8 row(s):

emp_id	ename	job	salary	commission
7001	Arvind Kulkarni	Manager	120000.00	NULL
7006	Fatima Siddiqui	Manager	105000.00	NULL
7014	Nalini Bhat	Manager	98000.00	NULL
7011	Kishore Banerjee	Manager	95000.00	NULL
7018	Radhika Menon	Manager	90000.00	NULL
7008	Harini Subbu	Salesman	47000.00	12000.00
7010	Jaya Prakash	Salesman	46000.00	5000.00
7007	Gopal Naidu	Salesman	45000.00	8000.00

Task 2 - Summarise data (5 marks)

> Table EMPLOYEE. (a) ONE row with: earners (employees with a commission), total_comm (sum of commission), avg_comm (average among earners, ROUND 2), avg_comm_all (average over ALL employees counting NULL as 0 - IFNULL, ROUND 2). (b) For each job show job, emps (count) and avg_salary (ROUND 2). Keep only jobs whose average salary is above 60000. Sort by avg_salary (highest first).

```
SELECT COUNT(commission) AS earners, SUM(commission) AS total_comm, ROUND(AVG(commission), 2) AS
avg_comm, ROUND(AVG(IFNULL(commission, 0)), 2) AS avg_comm_all FROM EMPLOYEE;
```

1 row(s):

earners	total_comm	avg_comm	avg_comm_all
4	26000.00	6500.00	1181.82

```
SELECT job, COUNT(*) AS emps, ROUND(AVG(salary), 2) AS avg_salary FROM EMPLOYEE GROUP BY job HAVING
AVG(salary) > 60000 ORDER BY avg_salary DESC;
```

3 row(s):

job	emps	avg_salary
Manager	5	101600.00
Developer	4	77750.00
Analyst	1	62000.00

Task 3 - Design & define (5 marks)

> ER fragment: EMPLOYEE and SKILL are in a many-to-many relationship; model it with a relationship table. (a) Create table EMP_SKILL: emp_id INT NOT NULL, foreign key to EMPLOYEE(emp_id); skill VARCHAR(30) NOT NULL; level_no INT DEFAULT 1 with CHECK (level_no BETWEEN 1 AND 5); composite PRIMARY KEY (emp_id, skill). (b) Change DEPARTMENT.location to VARCHAR(50) (MODIFY). (c) Insert (7002, 'SQL') listing emp_id and skill. Insert the SAME (7002, 'SQL') again: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE EMP_SKILL; and SELECT * FROM EMP_SKILL ORDER BY emp_id, skill;

```
CREATE TABLE EMP_SKILL (emp_id INT NOT NULL, skill VARCHAR(30) NOT NULL, level_no INT DEFAULT 1 CHECK (level_no BETWEEN 1 AND 5), PRIMARY KEY (emp_id, skill), FOREIGN KEY (emp_id) REFERENCES EMPLOYEE(emp_id));
```

OK - 0 row(s) affected

```
ALTER TABLE DEPARTMENT MODIFY location VARCHAR(50);
```

OK - 0 row(s) affected

```
INSERT INTO EMP_SKILL (emp_id, skill) VALUES (7002, 'SQL');
```

OK - 1 row(s) affected

```
INSERT INTO EMP_SKILL (emp_id, skill) VALUES (7002, 'SQL');
```

Error number 1062: Duplicate entry '7002-SQL' for key 'emp_skill.PRIMARY'

```
DESCRIBE EMP_SKILL;
```

3 row(s):

Field	Type	Null	Key	Default	Extra
emp_id	int	NO	PRI	NULL	
skill	varchar(30)	NO	PRI	NULL	
level_no	int	YES		1	

```
SELECT * FROM EMP_SKILL ORDER BY emp_id, skill;
```

1 row(s):

emp_id	skill	level_no
7002	SQL	1

Task 4 - Modify data (5 marks)

> (a) Insert employee (7023, 'Wasim Akhtar', 'Salesman', 20, 44000, NULL, '2026-09-01', 'Mumbai') listing all columns. (b) Set commission to 2000 for every Salesman whose commission is currently NULL. (c) Try to delete department 50: it must be rejected because employees belong to it. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Delete every Manager hired before '2017-01-01'. (e) Verify with: SELECT emp_id, commission FROM EMPLOYEE WHERE job = 'Salesman' ORDER BY emp_id; and SELECT emp_id, ename FROM EMPLOYEE WHERE job = 'Manager' ORDER BY emp_id;

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

OK - 1 row(s) affected

```
UPDATE EMPLOYEE SET commission = 2000 WHERE job = 'Salesman' AND commission IS NULL;
```

OK - 2 row(s) affected

```
DELETE FROM DEPARTMENT WHERE dept_id = 50;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_hr` . `employee` , CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`dept_id`) REFERENCES `department` (`dept_id`))

```
DELETE FROM EMPLOYEE WHERE hire_date < '2017-01-01' AND job = 'Manager';
```

OK - 3 row(s) affected

```
SELECT emp_id, commission FROM EMPLOYEE WHERE job = 'Salesman' ORDER BY emp_id;
```

5 row(s):

emp_id	commission
7007	8000.00
7008	12000.00
7009	2000.00
7010	5000.00
7023	2000.00

```
SELECT emp_id, ename FROM EMPLOYEE WHERE job = 'Manager' ORDER BY emp_id;
```

2 row(s):

emp_id	ename
7011	Kishore Banerjee
7018	Radhika Menon

Set 19 - Fee Defaulters & Cities

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

Task 1 - Retrieve data (5 marks)

> Table STUDENT. Show roll_no, sname, city, cgpa of students who do NOT live in Jaipur (<>), belong to CSE, ECE or ME (use IN) and have a CGPA of 7 or more. Sort by city (A to Z), then cgpa (highest first), then roll_no.

```
SELECT roll_no, sname, city, cgpa FROM STUDENT WHERE city <> 'Jaipur' AND dept IN ('CSE','ECE','ME') AND cgpa >= 7 ORDER BY city ASC, cgpa DESC, roll_no ASC;
```

8 row(s):

roll_no	sname	city	cgpa
2111	Yash Patel	Ahmedabad	7.60
2104	Sneha Iyer	Chennai	8.05
2110	Tanya Kapoor	Delhi	9.35
2103	Rohan Gupta	Delhi	7.45
2117	Aditya Bose	Kolkata	8.05
2120	Zoya Khan	Lucknow	8.72
2102	Diya Mehta	Mumbai	9.10
2113	Karan Malhotra	Mumbai	8.30

Task 2 - Summarise data (5 marks)

> Table STUDENT. (a) For students who owe money (fee_due > 0), ONE row with: defaulters (count), total_due, max_due, avg_due (ROUND 2). (b) For ALL students, for each city show city and students (count). Keep only cities with 2 or more students. Sort by students (highest first), then city.

```
SELECT COUNT(*) AS defaulters, SUM(fee_due) AS total_due, MAX(fee_due) AS max_due, ROUND(AVG(fee_due), 2) AS avg_due FROM STUDENT WHERE fee_due > 0;
```

1 row(s):

defaulters	total_due	max_due	avg_due
7	80000.00	25000.00	11428.57

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

5 row(s):

city	students
Jaipur	5
Delhi	3
Kochi	2
Kolkata	2
Mumbai	2

Task 3 - Design & define (5 marks)

> ER fragment: a STUDENT makes many FEE_PAYMENTs; each payment is made by exactly one student. (a) Create table FEE_PAYMENT: pay_id INT primary key; roll_no INT NOT NULL, foreign key to STUDENT(roll_no); amount DECIMAL(8,2) NOT NULL with CHECK (amount > 0); paid_on DATE NOT NULL; mode VARCHAR(10) DEFAULT 'UPI'. (b) Change EXAM_RESULT.subject to VARCHAR(50) NOT NULL (MODIFY). (c) Insert (1, 2103, 5000, '2026-09-01') listing the columns and leaving mode out. Then insert (2, 9999, 1000, '2026-09-02'): student 9999 does not exist, so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE FEE_PAYMENT; and SELECT * FROM FEE_PAYMENT ORDER BY pay_id;

```
CREATE TABLE FEE_PAYMENT (pay_id INT PRIMARY KEY, roll_no INT NOT NULL, amount DECIMAL(8,2) NOT NULL
CHECK (amount > 0), paid_on DATE NOT NULL, mode VARCHAR(10) DEFAULT 'UPI', FOREIGN KEY (roll_no)
REFERENCES STUDENT(roll_no));
```

OK - 0 row(s) affected

```
ALTER TABLE EXAM_RESULT MODIFY subject VARCHAR(50) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO FEE_PAYMENT (pay_id, roll_no, amount, paid_on) VALUES (1, 2103, 5000, '2026-09-01');
```

OK - 1 row(s) affected

```
INSERT INTO FEE_PAYMENT (pay_id, roll_no, amount, paid_on) VALUES (2, 9999, 1000, '2026-09-02');
```

Error number 1452: Cannot add or update a child row: a foreign key constraint fails (`eval_university` . `fee_payment` , CONSTRAINT `fee_payment_ibfk_1` FOREIGN KEY (`roll_no`) REFERENCES `student` (`roll_no`))

```
DESCRIBE FEE_PAYMENT;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
pay_id	int	NO	PRI	NULL	
roll_no	int	NO	MUL	NULL	
amount	decimal(8,2)	NO		NULL	
paid_on	date	NO		NULL	
mode	varchar(10)	YES		UPI	

```
SELECT * FROM FEE_PAYMENT ORDER BY pay_id;
```

1 row(s):

pay_id	roll_no	amount	paid_on	mode
1	2103	5000.00	2026-09-01	UPI

Task 4 - Modify data (5 marks)

> (a) Insert into EXAM_RESULT (result_id 25, roll_no 2108, 'Databases', 'End', marks 60) listing the columns and leaving max_marks and exam_date out. (b) Insert a student 2121 with sname NULL (any other values): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set fee_due to 0 for every student who lives in 'Delhi' and owes money. (d) Delete the student(s) whose CGPA has not been assigned (cgpa IS NULL). (e) Verify with: SELECT result_id, max_marks, exam_date FROM EXAM_RESULT WHERE result_id = 25; SELECT roll_no, fee_due FROM STUDENT WHERE city = 'Delhi' ORDER BY roll_no; and SELECT COUNT(*) AS students_left FROM STUDENT;

```
INSERT INTO EXAM_RESULT (result_id, roll_no, subject, exam_type, marks) VALUES (25, 2108, 'Databases',
'End', 60);
```

OK - 1 row(s) affected

```
INSERT INTO STUDENT (roll_no, sname, dept, semester) VALUES (2121, NULL, 'CSE', 3);
```

Error number 1048: Column 'sname' cannot be null

```
UPDATE STUDENT SET fee_due = 0 WHERE city = 'Delhi' AND fee_due > 0;
```

OK - 1 row(s) affected

```
DELETE FROM STUDENT WHERE cgpa IS NULL;
```

OK - 1 row(s) affected

```
SELECT result_id, max_marks, exam_date FROM EXAM_RESULT WHERE result_id = 25;
```

1 row(s):

result_id	max_marks	exam_date
25	100	NULL

```
SELECT roll_no, fee_due FROM STUDENT WHERE city = 'Delhi' ORDER BY roll_no;
```

3 row(s):

roll_no	fee_due
2103	0.00
2110	0.00
2115	0.00

```
SELECT COUNT(*) AS students_left FROM STUDENT;
```

1 row(s):

students_left
19

Set 20 - Revenue Desk

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

Task 1 - Retrieve data (5 marks)

> Table APPOINTMENT. Show appt_id, doctor_id, patient_name, fee_charged of appointments of doctors 102, 103, 104, 105 or 110 (use IN) where fee_charged is between 700 and 1000 (inclusive). Sort by fee_charged (highest first), then appt_id. Show only the first 5 rows (LIMIT).

```
SELECT appt_id, doctor_id, patient_name, fee_charged FROM APPOINTMENT WHERE doctor_id IN
(102,103,104,105,110) AND fee_charged BETWEEN 700 AND 1000 ORDER BY fee_charged DESC, appt_id LIMIT 5;
```

5 row(s):

appt_id	doctor_id	patient_name	fee_charged
4	103	Pankaj Sethi	1000.00
25	103	Jaspreet Singh	1000.00
3	102	Alok Mishra	900.00
20	110	Nandini Roy	900.00
23	102	Shalini Gupta	900.00

Task 2 - Summarise data (5 marks)

> Table APPOINTMENT. (a) For COMPLETED appointments, ONE row with: visits (count), revenue (sum), avg_fee (ROUND 2), lowest, highest (fee_charged). (b) For ALL appointments, for each doctor_id show doctor_id and appts (count). Keep only doctors with 3 or more appointments. Sort by appts (highest first), then doctor_id.

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

1 row(s):

visits	revenue	avg_fee	lowest	highest
19	14750.00	776.32	350.00	1500.00

```
SELECT doctor_id, COUNT(*) AS appts FROM APPOINTMENT GROUP BY doctor_id HAVING COUNT(*) >= 3 ORDER BY
appts DESC, doctor_id;
```

5 row(s):

doctor_id	appts
109	4
101	3
103	3
104	3
105	3

Task 3 - Design & define (5 marks)

> ER fragment: INSURANCE policies are stored independently (no foreign key needed). (a) Create table INSURANCE: policy_no VARCHAR(10) primary key; patient_name VARCHAR(40) NOT NULL; provider VARCHAR(30) NOT NULL; cover_amt DECIMAL(10,2) with CHECK (cover_amt > 0); valid_till DATE. (b) Add a column followup DATE to APPOINTMENT. (c) Insert ('P001', 'Ramesh Kumar', 'CareFirst', 200000, '2027-03-31') listing the columns. Then insert ('P002', 'Sunita Devi', 'CareFirst', 0, '2027-03-31'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE INSURANCE; and SELECT * FROM INSURANCE ORDER BY policy_no;

```
CREATE TABLE INSURANCE (policy_no VARCHAR(10) PRIMARY KEY, patient_name VARCHAR(40) NOT NULL, provider
VARCHAR(30) NOT NULL, cover_amt DECIMAL(10,2) CHECK (cover_amt > 0), valid_till DATE);
```

OK - 0 row(s) affected

```
ALTER TABLE APPOINTMENT ADD COLUMN followup DATE;
```

OK - 0 row(s) affected

```
INSERT INTO INSURANCE (policy_no, patient_name, provider, cover_amt, valid_till) VALUES ('P001', 'Ramesh Kumar', 'CareFirst', 200000, '2027-03-31');
```

OK - 1 row(s) affected

```
INSERT INTO INSURANCE (policy_no, patient_name, provider, cover_amt, valid_till) VALUES ('P002', 'Sunita Devi', 'CareFirst', 0, '2027-03-31');
```

Error number 3819: Check constraint 'insurance_chk_1' is violated.

```
DESCRIBE INSURANCE;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
policy_no	varchar(10)	NO	PRI	NULL	
patient_name	varchar(40)	NO		NULL	
provider	varchar(30)	NO		NULL	
cover_amt	decimal(10,2)	YES		NULL	
valid_till	date	YES		NULL	

```
SELECT * FROM INSURANCE ORDER BY policy_no;
```

1 row(s):

policy_no	patient_name	provider	cover_amt	valid_till
P001	Ramesh Kumar	CareFirst	200000.00	2027-03-31

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two appointments (list the columns, leave status and fee_charged out): (27, 104, 'Baby Rhea', 6, 'F', '2026-08-16') and (28, 105, 'Kunal Shah', 44, 'M', '2026-08-16'). (b) Insert appointment 29 with patient_name NULL: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Cancel (status = 'Cancelled') every Scheduled appointment of doctors 105 and 108. (d) Delete every appointment whose fee_charged is below 500 (rows with no fee must stay). (e) Verify with: SELECT appt_id, doctor_id, status FROM APPOINTMENT WHERE status IN ('Scheduled','Cancelled') ORDER BY appt_id; and SELECT COUNT(*) AS rows_left FROM APPOINTMENT;

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date) VALUES (27, 104, 'Baby Rhea', 6, 'F', '2026-08-16'), (28, 105, 'Kunal Shah', 44, 'M', '2026-08-16');
```

OK - 2 row(s) affected

```
INSERT INTO APPOINTMENT (appt_id, doctor_id, patient_name, age, gender, appt_date) VALUES (29, 105, NULL, 30, 'M', '2026-08-16');
```

Error number 1048: Column 'patient_name' cannot be null

```
UPDATE APPOINTMENT SET status = 'Cancelled' WHERE status = 'Scheduled' AND doctor_id IN (105, 108);
```

OK - 3 row(s) affected

```
DELETE FROM APPOINTMENT WHERE fee_charged < 500;
```

OK - 4 row(s) affected

```
SELECT appt_id, doctor_id, status FROM APPOINTMENT WHERE status IN ('Scheduled','Cancelled') ORDER BY appt_id;
```

9 row(s):

appt_id	doctor_id	status
5	103	Cancelled
9	105	Cancelled
13	107	Cancelled
15	108	Cancelled
21	110	Scheduled
22	101	Scheduled
24	105	Cancelled
27	104	Scheduled
28	105	Cancelled

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

1 row(s):

rows_left
24

Set 21 - Customer Loyalty

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

Task 1 - Retrieve data (5 marks)

> Table CUSTOMER. Show cust_id, cname, joined_on, membership, credit_limit of customers who have NO membership OR a credit_limit of 20000 or less. Sort by joined_on (latest first), then cust_id.

```
SELECT cust_id, cname, joined_on, membership, credit_limit FROM CUSTOMER WHERE membership IS NULL OR credit_limit <= 20000 ORDER BY joined_on DESC, cust_id;
```

5 row(s):

cust_id	cname	joined_on	membership	credit_limit
8	Himanshu Rathore	2025-05-30	NULL	10000.00
4	Deepak Verma	2025-02-20	NULL	10000.00
12	Lakshay Sethi	2025-01-08	Silver	15000.00
9	Isha Kulkarni	2024-09-09	Silver	20000.00
2	Bharat Jain	2024-01-15	Silver	20000.00

Task 2 - Summarise data (5 marks)

> Table ORDERS. (a) For Delivered orders, ONE row with: orders (count), units (sum of qty), revenue (sum of qty *unit_price), biggest (largest single qty* unit_price). (b) For ALL orders, for each status show status, orders (count) and revenue (sum of qty * unit_price). Keep only statuses with 3 or more orders. Sort by orders (highest first), then status.

```
SELECT COUNT(*) AS orders, SUM(qty) AS units, SUM(qty * unit_price) AS revenue, MAX(qty * unit_price) AS biggest FROM ORDERS WHERE status = 'Delivered';
```

1 row(s):

orders	units	revenue	biggest
17	36	277200.00	110000.00

```
SELECT status, COUNT(*) AS orders, SUM(qty * unit_price) AS revenue FROM ORDERS GROUP BY status HAVING COUNT(*) >= 3 ORDER BY orders DESC, status;
```

3 row(s):

status	orders	revenue
Delivered	17	277200.00
Placed	4	95850.00
Shipped	3	42100.00

Task 3 - Design & define (5 marks)

> ER fragment: CUSTOMER and item are in a many-to-many wish-list relationship; one customer can wish for many items. (a) Create table WISHLIST: cust_id INT NOT NULL, foreign key to CUSTOMER(cust_id); item VARCHAR(40) NOT NULL; priority INT DEFAULT 3 with CHECK (priority BETWEEN 1 AND 5); composite PRIMARY KEY (cust_id, item). (b) Add a column gift CHAR(1) with DEFAULT 'N' to ORDERS. (c) Insert (1, 'Camera') listing cust_id and item. Insert the SAME (1, 'Camera') again: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE WISHLIST; and SELECT * FROM WISHLIST ORDER BY cust_id, item;

```
CREATE TABLE WISHLIST (cust_id INT NOT NULL, item VARCHAR(40) NOT NULL, priority INT DEFAULT 3 CHECK (priority BETWEEN 1 AND 5), PRIMARY KEY (cust_id, item), FOREIGN KEY (cust_id) REFERENCES CUSTOMER(cust_id));
```

OK - 0 row(s) affected

```
ALTER TABLE ORDERS ADD COLUMN gift CHAR(1) DEFAULT 'N';
```

OK - 0 row(s) affected

```
INSERT INTO WISHLIST (cust_id, item) VALUES (1, 'Camera');
```

OK - 1 row(s) affected

```
INSERT INTO WISHLIST (cust_id, item) VALUES (1, 'Camera');
```

Error number 1062: Duplicate entry '1-Camera' for key 'wishlist.PRIMARY'

```
DESCRIBE WISHLIST;
```

3 row(s):

Field	Type	Null	Key	Default	Extra
cust_id	int	NO	PRI	NULL	
item	varchar(40)	NO	PRI	NULL	
priority	int	YES		3	

```
SELECT * FROM WISHLIST ORDER BY cust_id, item;
```

1 row(s):

cust_id	item	priority
1	Camera	3

Task 4 - Modify data (5 marks)

> (a) Insert customer (13, 'Mehul Doshi', 'Surat', 'Gold', '2026-09-01', 30000) listing all columns. (b) Try to delete customer 1: it must be rejected because orders refer to the customer. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Raise credit_limit by 20% for every Silver customer. (d) Delete customer 13 (this works). (e) Verify with: SELECT cust_id, credit_limit FROM CUSTOMER WHERE membership = 'Silver' ORDER BY cust_id; and SELECT COUNT(*) AS customers_left FROM CUSTOMER;

```
INSERT INTO CUSTOMER (cust_id, cname, city, membership, joined_on, credit_limit) VALUES (13, 'Mehul
Doshi', 'Surat', 'Gold', '2026-09-01', 30000);
```

OK - 1 row(s) affected

```
DELETE FROM CUSTOMER WHERE cust_id = 1;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_retail` . `orders` , CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`cust_id`) REFERENCES `customer` (`cust_id`))

```
UPDATE CUSTOMER SET credit_limit = credit_limit * 1.20 WHERE membership = 'Silver';
```

OK - 4 row(s) affected

```
DELETE FROM CUSTOMER WHERE cust_id = 13;
```

OK - 1 row(s) affected

```
SELECT cust_id, credit_limit FROM CUSTOMER WHERE membership = 'Silver' ORDER BY cust_id;
```

4 row(s):

cust_id	credit_limit
2	24000.00
6	30000.00
9	24000.00
12	18000.00

```
SELECT COUNT(*) AS customers_left FROM CUSTOMER;
```

1 row(s):

customers_left
12

Set 22 - Branch Performance

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

Task 1 - Retrieve data (5 marks)

> Table ACCOUNT. Show each DISTINCT combination of branch_city and acc_type (two columns) among accounts whose balance is more than 50000 and whose status is not 'Closed'. Sort by branch_city, then acc_type.

```
SELECT DISTINCT branch_city, acc_type FROM ACCOUNT WHERE balance > 50000 AND status <> 'Closed' ORDER BY branch_city, acc_type;
```

8 row(s):

branch_city	acc_type
Chennai	Fixed
Delhi	Current
Delhi	Fixed
Delhi	Savings
Hyderabad	Current
Jaipur	Current
Jaipur	Savings
Mumbai	Current

Task 2 - Summarise data (5 marks)

> Table ACCOUNT. (a) For the Delhi branch only, ONE row with: accounts (count), max_balance, min_balance, total_balance. (b) For ACTIVE accounts, for each branch_city show branch_city, accounts (count) and total_balance. Keep only branches with 2 or more active accounts. Sort by total_balance (highest first), then branch_city.

```
SELECT COUNT(*) AS accounts, MAX(balance) AS max_balance, MIN(balance) AS min_balance, SUM(balance) AS total_balance FROM ACCOUNT WHERE branch_city = 'Delhi';
```

1 row(s):

accounts	max_balance	min_balance	total_balance
4	500000.00	45250.50	719250.50

```
SELECT branch_city, COUNT(*) AS accounts, SUM(balance) AS total_balance FROM ACCOUNT WHERE status = 'Active' GROUP BY branch_city HAVING COUNT(*) >= 2 ORDER BY total_balance DESC, branch_city;
```

3 row(s):

branch_city	accounts	total_balance
Delhi	4	719250.50
Jaipur	3	342300.75
Pune	2	23500.00

Task 3 - Design & define (5 marks)

> ER fragment: an ACCOUNT can be issued many CHEQUE_BOOKs; each belongs to one account. (a) Create table CHEQUE_BOOK: cb_id INT primary key; acc_no INT NOT NULL, foreign key to ACCOUNT(acc_no); leaves INT DEFAULT 25 with CHECK (leaves IN (25, 50, 100)); issued_on DATE NOT NULL. (b) Change TXN.remarks to VARCHAR(100) (MODIFY). (c) Insert (1, account 5002, issued_on '2026-09-01') listing the columns and leaving leaves out. Then insert (2, account 5002, leaves 30, issued_on '2026-09-02'): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE CHEQUE_BOOK; and SELECT * FROM CHEQUE_BOOK ORDER BY cb_id;

```
CREATE TABLE CHEQUE_BOOK (cb_id INT PRIMARY KEY, acc_no INT NOT NULL, leaves INT DEFAULT 25 CHECK (leaves IN (25, 50, 100)), issued_on DATE NOT NULL, FOREIGN KEY (acc_no) REFERENCES ACCOUNT(acc_no));
```

OK - 0 row(s) affected

```
ALTER TABLE TXN MODIFY remarks VARCHAR(100);
```

OK - 28 row(s) affected

```
INSERT INTO CHEQUE_BOOK (cb_id, acc_no, issued_on) VALUES (1, 5002, '2026-09-01');
```

OK - 1 row(s) affected

```
INSERT INTO CHEQUE_BOOK (cb_id, acc_no, leaves, issued_on) VALUES (2, 5002, 30, '2026-09-02');
```

Error number 3819: Check constraint 'cheque_book_chk_1' is violated.

```
DESCRIBE CHEQUE_BOOK;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
cb_id	int	NO	PRI	NULL	
acc_no	int	NO	MUL	NULL	
leaves	int	YES		25	
issued_on	date	NO		NULL	

```
SELECT * FROM CHEQUE_BOOK ORDER BY cb_id;
```

1 row(s):

cb_id	acc_no	leaves	issued_on
1	5002	25	2026-09-01

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two accounts (list the columns): (5015, 'Omkar Pillai', 'Chennai', 'Savings', 5000, '2026-10-01', 'Active', NULL) and (5016, 'Preeti Nair', 'Pune', 'Savings', 12000, '2026-10-01', 'Active', 'Anil Nair'). (b) Try to delete account 5001: it must be rejected because transactions refer to it. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Set nominee to 'Not Provided' for every Active account that has no nominee. (d) Delete every Dormant account with a balance of 0 (it has no transactions). (e) Verify with: SELECT acc_no, nominee FROM ACCOUNT WHERE nominee = 'Not Provided' ORDER BY acc_no; and SELECT COUNT(*) AS accounts_left FROM ACCOUNT;

```
INSERT INTO ACCOUNT (acc_no, holder_name, branch_city, acc_type, balance, opened_on, status, nominee)
VALUES (5015, 'Omkar Pillai', 'Chennai', 'Savings', 5000, '2026-10-01', 'Active', NULL), (5016, 'Preeti
Nair', 'Pune', 'Savings', 12000, '2026-10-01', 'Active', 'Anil Nair');
```

OK - 2 row(s) affected

```
DELETE FROM ACCOUNT WHERE acc_no = 5001;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_bank` . `txn` , CONSTRAINT `txn_ibfk_1` FOREIGN KEY (`acc_no`) REFERENCES `account` (`acc_no`))

```
UPDATE ACCOUNT SET nominee = 'Not Provided' WHERE nominee IS NULL AND status = 'Active';
```

OK - 6 row(s) affected

```
DELETE FROM ACCOUNT WHERE status = 'Dormant' AND balance = 0;
```

OK - 1 row(s) affected

```
SELECT acc_no, nominee FROM ACCOUNT WHERE nominee = 'Not Provided' ORDER BY acc_no;
```

6 row(s):

acc_no	nominee
5002	Not Provided
5005	Not Provided
5008	Not Provided
5011	Not Provided
5014	Not Provided
5015	Not Provided

```
SELECT COUNT(*) AS accounts_left FROM ACCOUNT;
```

1 row(s):

accounts_left
15

Set 23 - Class & Baggage

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

Task 1 - Retrieve data (5 marks)

> Table BOOKING. Show booking_id, passenger, age, fare_paid of Economy passengers who have NO check-in baggage recorded (baggage_kg IS NULL) and are aged between 20 and 60 (inclusive). Sort by age (oldest first), then booking_id.

```
SELECT booking_id, passenger, age, fare_paid FROM BOOKING WHERE seat_class = 'Economy' AND baggage_kg IS NULL AND age BETWEEN 20 AND 60 ORDER BY age DESC, booking_id;
```

4 row(s):

booking_id	passenger	age	fare_paid
5	Tushar Pandey	60	4680.00
10	Hema Joshi	49	NULL
12	Sunil Gavande	38	2900.00
18	Tanmay Dey	22	NULL

Task 2 - Summarise data (5 marks)

> Table BOOKING. (a) ONE row with: oldest (maximum age), youngest (minimum age), avg_age (ROUND 1), classes (number of DISTINCT seat_class values). (b) For each seat_class show seat_class, bookings (count) and total_bag (sum of baggage_kg, show 0 instead of NULL - IFNULL). Keep only classes with more than 5 bookings. Sort by seat_class.

```
SELECT MAX(age) AS oldest, MIN(age) AS youngest, ROUND(AVG(age), 1) AS avg_age, COUNT(DISTINCT seat_class) AS classes FROM BOOKING;
```

1 row(s):

oldest	youngest	avg_age	classes
71	7	38.7	2

```
SELECT seat_class, COUNT(*) AS bookings, IFNULL(SUM(baggage_kg), 0) AS total_bag FROM BOOKING GROUP BY seat_class HAVING COUNT(*) > 5 ORDER BY seat_class;
```

1 row(s):

seat_class	bookings	total_bag
Economy	21	205

Task 3 - Design & define (5 marks)

> ER fragment: a BOOKING can include many LOUNGE_PASSes; each pass belongs to one booking. (a) Create table LOUNGE_PASS: pass_id INT primary key; booking_id INT NOT NULL, foreign key to BOOKING(booking_id); lounge VARCHAR(20) NOT NULL; valid_hours INT DEFAULT 3 with CHECK (valid_hours BETWEEN 1 AND 12). (b) Add a column meal_pref VARCHAR(10) to BOOKING. (c) Insert (1, booking 3, 'Maharaja') listing pass_id, booking_id, lounge. Then insert (2, booking 7, 'Maharaja', valid_hours 24): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE LOUNGE_PASS; and SELECT * FROM LOUNGE_PASS ORDER BY pass_id;

```
CREATE TABLE LOUNGE_PASS (pass_id INT PRIMARY KEY, booking_id INT NOT NULL, lounge VARCHAR(20) NOT NULL, valid_hours INT DEFAULT 3 CHECK (valid_hours BETWEEN 1 AND 12), FOREIGN KEY (booking_id) REFERENCES BOOKING(booking_id));
```

OK - 0 row(s) affected

```
ALTER TABLE BOOKING ADD COLUMN meal_pref VARCHAR(10);
```

OK - 0 row(s) affected

```
INSERT INTO LOUNGE_PASS (pass_id, booking_id, lounge) VALUES (1, 3, 'Maharaja');
```

OK - 1 row(s) affected

```
INSERT INTO LOUNGE_PASS (pass_id, booking_id, lounge, valid_hours) VALUES (2, 7, 'Maharaja', 24);
```

Error number 3819: Check constraint 'lounge_pass_chk_1' is violated.

```
DESCRIBE LOUNGE_PASS;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
pass_id	int	NO	PRI	NULL	
booking_id	int	NO	MUL	NULL	
lounge	varchar(20)	NO		NULL	
valid_hours	int	YES		3	

```
SELECT * FROM LOUNGE_PASS ORDER BY pass_id;
```

1 row(s):

pass_id	booking_id	lounge	valid_hours
1	3	Maharaja	3

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two bookings (list the columns): (27, 'SJ303', 'Dev Sharma', 29, 'Economy', 3400, 'Confirmed', 10) and (28, 'SJ303', 'Ira Sharma', 5, 'Economy', 1700, 'Confirmed', NULL). (b) Insert a booking 29 with passenger 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) Give every passenger aged 60 or above a 10% discount on fare_paid (fare_paid *0.90). (d) Delete every Confirmed booking with fare_paid below 2800. (e) Verify with: SELECT booking_id, age, fare_paid FROM BOOKING WHERE age >= 60 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, 'SJ303', 'Dev Sharma', 29, 'Economy', 3400, 'Confirmed', 10), (28, 'SJ303', 'Ira
Sharma', 5, 'Economy', 1700, 'Confirmed', NULL);
```

OK - 2 row(s) affected

```
INSERT INTO BOOKING (booking_id, flight_no, passenger, age, seat_class, booking_status) VALUES (29,
'SJ303', NULL, 30, 'Economy', 'Confirmed');
```

Error number 1048: Column 'passenger' cannot be null

```
UPDATE BOOKING SET fare_paid = fare_paid * 0.90 WHERE age >= 60;
```

OK - 3 row(s) affected

```
DELETE FROM BOOKING WHERE booking_status = 'Confirmed' AND fare_paid < 2800;
```

OK - 3 row(s) affected

```
SELECT booking_id, age, fare_paid FROM BOOKING WHERE age >= 60 ORDER BY booking_id;
```

2 row(s):

booking_id	age	fare_paid
5	60	4212.00
20	71	5832.00

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

1 row(s):

bookings_left
25

Set 24 - Department Budgets

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

Task 1 - Retrieve data (5 marks)

> Table DEPARTMENT. Show dept_id, dept_name, location, budget of departments whose budget is between 900000 and 2500000 (inclusive) and whose name contains the letter 'a' (LIKE '%a%'). Sort by budget (highest first), then dept_id.

```
SELECT dept_id, dept_name, location, budget FROM DEPARTMENT WHERE budget BETWEEN 900000 AND 2500000 AND dept_name LIKE '%a%' ORDER BY budget DESC, dept_id;
```

3 row(s):

dept_id	dept_name	location	budget
20	Sales	Mumbai	2500000.00
40	Finance	Delhi	1800000.00
30	Human Resources	Jaipur	900000.00

Task 2 - Summarise data (5 marks)

> (a) Table DEPARTMENT: ONE row with: depts (count), total_budget, avg_budget (ROUND 2), max_budget. (b) Table EMPLOYEE: for each dept_id (where dept_id is not NULL) show dept_id, emps (count) and total_pay (sum of salary + commission, counting a missing commission as 0 - IFNULL). Keep only departments whose total_pay is more than 200000. Sort by total_pay (highest first), then dept_id.

```
SELECT COUNT(*) AS depts, SUM(budget) AS total_budget, ROUND(AVG(budget), 2) AS avg_budget, MAX(budget) AS max_budget FROM DEPARTMENT;
```

1 row(s):

depts	total_budget	avg_budget	max_budget
5	11400000.00	2280000.00	5000000.00

```
SELECT dept_id, COUNT(*) AS emps, SUM(salary + IFNULL(commission, 0)) AS total_pay FROM EMPLOYEE WHERE dept_id IS NOT NULL GROUP BY dept_id HAVING SUM(salary + IFNULL(commission, 0)) > 200000 ORDER BY total_pay DESC, dept_id;
```

3 row(s):

dept_id	emps	total_pay
10	5	413000.00
20	5	311000.00
40	4	270000.00

Task 3 - Design & define (5 marks)

> ER fragment: an EMPLOYEE can be issued many ASSETs; each asset is issued to one employee. (a) Create table ASSET: asset_id INT primary key; emp_id INT NOT NULL, foreign key to EMPLOYEE(emp_id); asset_name VARCHAR(30) NOT NULL; cost DECIMAL(9,2) with CHECK (cost > 0); returned CHAR(1) DEFAULT 'N'. (b) Change DEPARTMENT.dept_name to VARCHAR(40) NOT NULL (MODIFY). (c) Insert (1, employee 7002, 'Laptop', 85000) listing asset_id, emp_id, asset_name, cost. Then insert (2, employee 7002, 'Monitor', 0): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE ASSET; and SELECT * FROM ASSET ORDER BY asset_id;

```
CREATE TABLE ASSET (asset_id INT PRIMARY KEY, emp_id INT NOT NULL, asset_name VARCHAR(30) NOT NULL, cost DECIMAL(9,2) CHECK (cost > 0), returned CHAR(1) DEFAULT 'N', FOREIGN KEY (emp_id) REFERENCES EMPLOYEE(emp_id));
```

OK - 0 row(s) affected

```
ALTER TABLE DEPARTMENT MODIFY dept_name VARCHAR(40) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO ASSET (asset_id, emp_id, asset_name, cost) VALUES (1, 7002, 'Laptop', 85000);
```

OK - 1 row(s) affected

```
INSERT INTO ASSET (asset_id, emp_id, asset_name, cost) VALUES (2, 7002, 'Monitor', 0);
```

Error number 3819: Check constraint 'asset_chk_1' is violated.

```
DESCRIBE ASSET;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
asset_id	int	NO	PRI	NULL	
emp_id	int	NO	MUL	NULL	
asset_name	varchar(30)	NO		NULL	
cost	decimal(9,2)	YES		NULL	
returned	char(1)	YES		N	

```
SELECT * FROM ASSET ORDER BY asset_id;
```

1 row(s):

asset_id	emp_id	asset_name	cost	returned
1	7002	Laptop	85000.00	N

Task 4 - Modify data (5 marks)

> (a) Insert department (60, 'Research', 'Hyderabad', 1500000) listing all columns. (b) Insert department (70, 'Sales', 'Kolkata', 100000): the department name 'Sales' already exists (UNIQUE), so it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Increase budget by 10% for every department located in 'Pune' or 'Jaipur'. (d) Delete department 60 (this works because it has no employees). (e) Verify with: SELECT dept_id, budget FROM DEPARTMENT ORDER BY dept_id;

```
INSERT INTO DEPARTMENT (dept_id, dept_name, location, budget) VALUES (60, 'Research', 'Hyderabad',
1500000);
```

OK - 1 row(s) affected

```
INSERT INTO DEPARTMENT (dept_id, dept_name, location, budget) VALUES (70, 'Sales', 'Kolkata', 100000);
```

Error number 1062: Duplicate entry 'Sales' for key 'department.dept_name'

```
UPDATE DEPARTMENT SET budget = budget * 1.10 WHERE location IN ('Pune', 'Jaipur');
```

OK - 2 row(s) affected

```
DELETE FROM DEPARTMENT WHERE dept_id = 60;
```

OK - 1 row(s) affected

```
SELECT dept_id, budget FROM DEPARTMENT ORDER BY dept_id;
```

5 row(s):

dept_id	budget
10	5000000.00
20	2500000.00
30	990000.00
40	1800000.00
50	1320000.00

Set 25 - Department Scoreboard

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

Task 1 - Retrieve data (5 marks)

> Table STUDENT. Show sname, dept, cgpa of the 5 best students among those whose email is recorded, whose CGPA is assigned and whose dept is not 'ME' (<>). Sort by cgpa (highest first), then roll_no; show only the first 5 rows (LIMIT).

```
SELECT sname, dept, cgpa FROM STUDENT WHERE email IS NOT NULL AND cgpa IS NOT NULL AND dept <> 'ME'
ORDER BY cgpa DESC, roll_no LIMIT 5;
```

5 row(s):

sname	dept	cgpa
Tanya Kapoor	CSE	9.35
Diya Mehta	CSE	9.10
Lakshmi Menon	IT	9.00
Aarav Sharma	CSE	8.72
Meera Nair	IT	8.72

Task 2 - Summarise data (5 marks)

> Table STUDENT. (a) For semester 7 only, ONE row with: students (count), avg_cgpa (ROUND 2, NULLs ignored), unassigned (students in semester 7 whose CGPA is NULL - hint: COUNT(*) minus COUNT(cgpa)). (b) For each dept show dept, max_cgpa and total_due (sum of fee_due). Keep only departments whose max_cgpa is 9 or more. Sort by dept.

```
SELECT COUNT(*) AS students, ROUND(AVG(cgpa), 2) AS avg_cgpa, COUNT(*) - COUNT(cgpa) AS unassigned FROM
STUDENT WHERE semester = 7;
```

1 row(s):

students	avg_cgpa	unassigned
5	7.95	1

```
SELECT dept, MAX(cgpa) AS max_cgpa, SUM(fee_due) AS total_due FROM STUDENT GROUP BY dept HAVING
MAX(cgpa) >= 9 ORDER BY dept;
```

2 row(s):

dept	max_cgpa	total_due
CSE	9.35	12500.00
IT	9.00	5000.00

Task 3 - Design & define (5 marks)

> ER fragment: STUDENT and CLUB are in a many-to-many relationship (a student joins many clubs, a club has many students). Model it with a relationship table. (a) Create table CLUB_MEMBER: roll_no INT NOT NULL, foreign key to STUDENT(roll_no); club_name VARCHAR(30) NOT NULL; role VARCHAR(15) DEFAULT 'Member'; composite PRIMARY KEY (roll_no, club_name). (b) Rename EXAM_RESULT.exam_date to held_on (RENAME COLUMN). (c) Insert (2101, 'Coding Club') listing the columns and leaving role out. Insert the SAME (2101, 'Coding Club') again: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE CLUB_MEMBER; and SELECT * FROM CLUB_MEMBER ORDER BY roll_no, club_name;

```
CREATE TABLE CLUB_MEMBER (roll_no INT NOT NULL, club_name VARCHAR(30) NOT NULL, role VARCHAR(15) DEFAULT
'Member', PRIMARY KEY (roll_no, club_name), FOREIGN KEY (roll_no) REFERENCES STUDENT(roll_no));
```

OK - 0 row(s) affected

```
ALTER TABLE EXAM_RESULT RENAME COLUMN exam_date TO held_on;
```

OK - 0 row(s) affected

```
INSERT INTO CLUB_MEMBER (roll_no, club_name) VALUES (2101, 'Coding Club');
```

OK - 1 row(s) affected

```
INSERT INTO CLUB_MEMBER (roll_no, club_name) VALUES (2101, 'Coding Club');
```

Error number 1062: Duplicate entry '2101-Coding Club' for key 'club_member.PRIMARY'

```
DESCRIBE CLUB_MEMBER;
```

3 row(s):

Field	Type	Null	Key	Default	Extra
roll_no	int	NO	PRI	NULL	
club_name	varchar(30)	NO	PRI	NULL	
role	varchar(15)	YES		Member	

```
SELECT * FROM CLUB_MEMBER ORDER BY roll_no, club_name;
```

1 row(s):

roll_no	club_name	role
2101	Coding Club	Member

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two students (list the columns): (2121, 'Aman Roy', 'IT', 3, 8.10, 'Pune', 'aman.roy@uni.edu') and (2122, 'Bina Paul', 'IT', 3, 7.20, 'Kochi', 'bina.paul@uni.edu'). (b) Try to delete student 2101 from STUDENT: it must be rejected because exam results refer to the student. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Add 1 to the semester of every CE student whose semester is below 7. (d) Delete all EXAM_RESULT rows of student 2108, and then delete student 2108 from STUDENT. (e) Verify with: SELECT roll_no, dept, semester FROM STUDENT WHERE dept IN ('CE','IT') ORDER BY roll_no; and SELECT COUNT(*) AS students_left FROM STUDENT;

```
INSERT INTO STUDENT (roll_no, sname, dept, semester, cgpa, city, email) VALUES (2121, 'Aman Roy', 'IT',
3, 8.10, 'Pune', 'aman.roy@uni.edu'), (2122, 'Bina Paul', 'IT', 3, 7.20, 'Kochi', 'bina.paul@uni.edu');
```

OK - 2 row(s) affected

```
DELETE FROM STUDENT WHERE roll_no = 2101;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_university`.`club_member`, CONSTRAINT `club_member_ibfk_1` FOREIGN KEY (`roll_no`) REFERENCES `student` (`roll_no`))

```
UPDATE STUDENT SET semester = semester + 1 WHERE dept = 'CE' AND semester < 7;
```

OK - 2 row(s) affected

```
DELETE FROM EXAM_RESULT WHERE roll_no = 2108;
```

OK - 1 row(s) affected

```
DELETE FROM STUDENT WHERE roll_no = 2108;
```

OK - 1 row(s) affected

```
SELECT roll_no, dept, semester FROM STUDENT WHERE dept IN ('CE','IT') ORDER BY roll_no;
```

9 row(s):

roll_no	dept	semester
2106	IT	3
2107	IT	5
2112	IT	7
2114	CE	4
2115	CE	6
2116	CE	7
2118	IT	5
2121	IT	3
2122	IT	3

```
SELECT COUNT(*) AS students_left FROM STUDENT;
```

1 row(s):

students_left
21

Set 26 - Specialty Overview

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

Task 1 - Retrieve data (5 marks)

> Table DOCTOR. Show each DISTINCT specialization of doctors who have 10 or more years of experience OR a consult_fee of 1000 or more. Show one column, specialization. Sort A to Z.

```
SELECT DISTINCT specialization FROM DOCTOR WHERE experience_yrs >= 10 OR consult_fee >= 1000 ORDER BY specialization;
```

6 row(s):

specialization
Cardiology
Dermatology
General Medicine
Neurology
Orthopedics
Pediatrics

Task 2 - Summarise data (5 marks)

> (a) Table APPOINTMENT: ONE row with: first_day (earliest appt_date), last_day (latest appt_date), busy_days (number of DISTINCT dates), patients (number of DISTINCT patient_name). (b) Table DOCTOR: for each specialization show specialization, doctors (count) and total_exp (sum of experience_yrs). Keep only specializations whose total_exp is 25 or more. Sort by total_exp (highest first), then specialization.

```
SELECT MIN(appt_date) AS first_day, MAX(appt_date) AS last_day, COUNT(DISTINCT appt_date) AS busy_days, COUNT(DISTINCT patient_name) AS patients FROM APPOINTMENT;
```

1 row(s):

first_day	last_day	busy_days	patients
2026-08-03	2026-08-15	13	26

```
SELECT specialization, COUNT(*) AS doctors, SUM(experience_yrs) AS total_exp FROM DOCTOR GROUP BY specialization HAVING SUM(experience_yrs) >= 25 ORDER BY total_exp DESC, specialization;
```

3 row(s):

specialization	doctors	total_exp
Orthopedics	2	29
Cardiology	2	27
General Medicine	1	25

Task 3 - Design & define (5 marks)

> ER fragment: WARD is an independent entity. (a) Create table WARD: ward_id INT primary key; ward_name VARCHAR(20) NOT NULL UNIQUE; beds INT with CHECK (beds > 0); floor_no INT DEFAULT 1. (b) Add a column joined_on DATE to DOCTOR. (c) Insert (1, 'ICU', 10 beds) listing the columns and leaving floor_no out. Then insert (2, 'General', -5 beds): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE WARD; and SELECT * FROM WARD ORDER BY ward_id;

```
CREATE TABLE WARD (ward_id INT PRIMARY KEY, ward_name VARCHAR(20) NOT NULL UNIQUE, beds INT CHECK (beds > 0), floor_no INT DEFAULT 1);
```

OK - 0 row(s) affected

```
ALTER TABLE DOCTOR ADD COLUMN joined_on DATE;
```

OK - 0 row(s) affected

```
INSERT INTO WARD (ward_id, ward_name, beds) VALUES (1, 'ICU', 10);
```

OK - 1 row(s) affected

```
INSERT INTO WARD (ward_id, ward_name, beds) VALUES (2, 'General', -5);
```

Error number 3819: Check constraint 'ward_chk_1' is violated.

```
DESCRIBE WARD;
```

4 row(s):

Field	Type	Null	Key	Default	Extra
ward_id	int	NO	PRI	NULL	
ward_name	varchar(20)	NO	UNI	NULL	
beds	int	YES		NULL	
floor_no	int	YES		1	

```
SELECT * FROM WARD ORDER BY ward_id;
```

1 row(s):

ward_id	ward_name	beds	floor_no
1	ICU	10	1

Task 4 - Modify data (5 marks)

> (a) Insert doctor (111, 'Dr. Anita Paul', 'Cardiology', 3, 800, NULL) listing all columns. (b) Add 1 to experience_yrs of every Pediatrics doctor. (c) Delete doctor 111. Then try to delete doctor 102: it must be rejected because appointments refer to the doctor. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Delete every Cancelled appointment of a patient older than 60. (e) Verify with: SELECT doctor_id, experience_yrs FROM DOCTOR WHERE specialization = 'Pediatrics' ORDER BY doctor_id; SELECT COUNT*() AS doctors_left FROM DOCTOR; and SELECT COUNT*()* AS appts_left FROM APPOINTMENT;

```
INSERT INTO DOCTOR (doctor_id, dname, specialization, experience_yrs, consult_fee, phone) VALUES (111,
'Dr. Anita Paul', 'Cardiology', 3, 800, NULL);
```

OK - 1 row(s) affected

```
UPDATE DOCTOR SET experience_yrs = experience_yrs + 1 WHERE specialization = 'Pediatrics';
```

OK - 2 row(s) affected

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

OK - 1 row(s) affected

```
DELETE FROM DOCTOR WHERE doctor_id = 102;
```

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`))

```
DELETE FROM APPOINTMENT WHERE status = 'Cancelled' AND age > 60;
```

OK - 1 row(s) affected

```
SELECT doctor_id, experience_yrs FROM DOCTOR WHERE specialization = 'Pediatrics' ORDER BY doctor_id;
```

2 row(s):

doctor_id	experience_yrs
104	13
108	5

```
SELECT COUNT(*) AS doctors_left FROM DOCTOR;
```

1 row(s):

doctors_left
10

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

1 row(s):

appts_left
25

Set 27 - Pricing Review

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

Task 1 - Retrieve data (5 marks)

> Table ORDERS. Show order_id, item, category, unit_price of orders that are not Cancelled AND (are Electronics with unit_price 3000 or more OR are Accessories with unit_price 5000 or more). Sort by unit_price (highest first), then order_id. Show only the first 6 rows (LIMIT). (Mind the brackets.)

```
SELECT order_id, item, category, unit_price FROM ORDERS WHERE status <> 'Cancelled' AND ((category = 'Electronics' AND unit_price >= 3000) OR (category = 'Accessories' AND unit_price >= 5000)) ORDER BY unit_price DESC, order_id LIMIT 6;
```

6 row(s):

order_id	item	category	unit_price
1001	Laptop	Electronics	55000.00
1014	Laptop	Electronics	55000.00
1010	Tablet	Electronics	32000.00
1023	Tablet	Electronics	32000.00
1004	Smartphone	Electronics	28000.00
1018	Smartphone	Electronics	28000.00

Task 2 - Summarise data (5 marks)

> Table ORDERS. (a) ONE row with: total_orders, with_discount (orders having discount_pct), avg_discount (ROUND 2, NULLs ignored), avg_discount_all (ROUND 2, NULL counted as 0 - IFNULL). (b) For each category show category, orders (count) and avg_price (average unit_price, ROUND 2). Keep only categories whose average unit_price is below 5000. Sort by avg_price (highest first).

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

1 row(s):

total_orders	with_discount	avg_discount	avg_discount_all
28	12	5.92	2.54

```
SELECT category, COUNT(*) AS orders, ROUND(AVG(unit_price), 2) AS avg_price FROM ORDERS GROUP BY category HAVING AVG(unit_price) < 5000 ORDER BY avg_price DESC;
```

3 row(s):

category	orders	avg_price
Accessories	5	3440.00
Footwear	5	2480.00
Clothing	7	1371.43

Task 3 - Design & define (5 marks)

> ER fragment: PRODUCT_MASTER is an independent entity. (a) Create table PRODUCT_MASTER: product_id INT primary key; item_name VARCHAR(40) NOT NULL UNIQUE; category VARCHAR(20) NOT NULL; mrp DECIMAL(9,2) NOT NULL with CHECK (mrp > 0); stock INT DEFAULT 0 with CHECK (stock >= 0). (b) Change ORDERS.item to VARCHAR(80) NOT NULL (MODIFY). (c) Insert (1, 'Laptop', 'Electronics', 60000) listing product_id, item_name, category, mrp. Then insert (2, 'Mouse', 'Electronics', 900, stock -5): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE PRODUCT_MASTER; and SELECT * FROM PRODUCT_MASTER ORDER BY product_id;

```
CREATE TABLE PRODUCT_MASTER (product_id INT PRIMARY KEY, item_name VARCHAR(40) NOT NULL UNIQUE, category VARCHAR(20) NOT NULL, mrp DECIMAL(9,2) NOT NULL CHECK (mrp > 0), stock INT DEFAULT 0 CHECK (stock >= 0));
```

OK - 0 row(s) affected

```
ALTER TABLE ORDERS MODIFY item VARCHAR(80) NOT NULL;
```

OK - 28 row(s) affected

```
INSERT INTO PRODUCT_MASTER (product_id, item_name, category, mrp) VALUES (1, 'Laptop', 'Electronics',
60000);
```

OK - 1 row(s) affected

```
INSERT INTO PRODUCT_MASTER (product_id, item_name, category, mrp, stock) VALUES (2, 'Mouse',
'Electronics', 900, -5);
```

Error number 3819: Check constraint 'product_master_chk_2' is violated.

```
DESCRIBE PRODUCT_MASTER;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
item_name	varchar(40)	NO	UNI	NULL	
category	varchar(20)	NO		NULL	
mrp	decimal(9,2)	NO		NULL	
stock	int	YES		0	

```
SELECT * FROM PRODUCT_MASTER ORDER BY product_id;
```

1 row(s):

product_id	item_name	category	mrp	stock
1	Laptop	Electronics	60000.00	0

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two orders (list the columns, leave discount_pct out): (1029, 4, 'Sandals', 'Footwear', 2, 950, '2026-07-17', 'Placed') and (1030, 9, 'Hoodie', 'Clothing', 1, 2500, '2026-07-18', 'Placed'). (b) Insert an order 1031 for customer 99: it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Change the status to 'Shipped' for every 'Placed' order dated before '2026-07-05'. (d) Delete every Cancelled order with qty = 1. (e) Verify with: SELECT order_id, status FROM ORDERS WHERE order_id >= 1017 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, 4, 'Sandals', 'Footwear', 2, 950, '2026-07-17', 'Placed'), (1030, 9, 'Hoodie', 'Clothing', 1,
2500, '2026-07-18', 'Placed');
```

OK - 2 row(s) affected

```
INSERT INTO ORDERS (order_id, cust_id, item, category, qty, unit_price, order_date, status) VALUES
(1031, 99, 'Mouse', 'Electronics', 1, 800, '2026-07-18', '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 status = 'Shipped' WHERE status = 'Placed' AND order_date < '2026-07-05';
```

OK - 2 row(s) affected

```
DELETE FROM ORDERS WHERE status = 'Cancelled' AND qty = 1;
```

OK - 1 row(s) affected

```
SELECT order_id, status FROM ORDERS WHERE order_id >= 1017 ORDER BY order_id;
```

14 row(s):

order_id	status
1017	Shipped
1018	Shipped
1019	Cancelled
1020	Delivered
1021	Delivered
1022	Delivered
1023	Placed
1024	Delivered
1025	Shipped
1026	Returned
1027	Delivered
1028	Placed
1029	Placed
1030	Placed

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

1 row(s):

rows_left
29

Set 28 - Cash Flow

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

Task 1 - Retrieve data (5 marks)

> Table TXN. Show txn_id, acc_no, amount, txn_date of Withdraw transactions that have NO remarks and an amount of 1500 or more. Sort by txn_date (earliest first), then txn_id. Show only the first 6 rows (LIMIT).

```
SELECT txn_id, acc_no, amount, txn_date FROM TXN WHERE txn_type = 'Withdraw' AND remarks IS NULL AND
amount >= 1500 ORDER BY txn_date, txn_id LIMIT 6;
```

5 row(s):

txn_id	acc_no	amount	txn_date
2	5001	5000.00	2026-07-03
7	5004	2000.00	2026-07-08
10	5007	1500.00	2026-07-11
13	5009	2500.00	2026-07-14
20	5014	15000.00	2026-07-21

Task 2 - Summarise data (5 marks)

> Table TXN. (a) For Withdraw transactions dated between '2026-07-10' and '2026-07-25' (inclusive), ONE row with: withdrawals (count), total_withdrawn, avg_withdrawn (ROUND 2). (b) For ALL transactions, for each acc_no show acc_no and txns (count). Keep only accounts with 3 or more transactions. Sort by txns (highest first), then acc_no.

```
SELECT COUNT(*) AS withdrawals, SUM(amount) AS total_withdrawn, ROUND(AVG(amount), 2) AS avg_withdrawn
FROM TXN WHERE txn_type = 'Withdraw' AND txn_date BETWEEN '2026-07-10' AND '2026-07-25';
```

1 row(s):

withdrawals	total_withdrawn	avg_withdrawn
5	45200.00	9040.00

```
SELECT acc_no, COUNT(*) AS txns FROM TXN GROUP BY acc_no HAVING COUNT(*) >= 3 ORDER BY txns DESC,
acc_no;
```

5 row(s):

acc_no	txns
5001	4
5002	3
5004	3
5007	3
5009	3

Task 3 - Design & define (5 marks)

> ER fragment: an ACCOUNT can set up many STANDING_ORDERs; each belongs to one account. (a) Create table STANDING_ORDER: so_id INT primary key; acc_no INT NOT NULL, foreign key to ACCOUNT(acc_no); payee VARCHAR(40) NOT NULL; amount DECIMAL(10,2) with CHECK (amount > 0); frequency VARCHAR(10) DEFAULT 'Monthly'; next_run DATE. (b) Add a column pan CHAR(10) with a UNIQUE constraint to ACCOUNT. (c) Insert (1, account 5001, 'Landlord', 15000) listing so_id, acc_no, payee, amount. Then insert (2, account 5001, 'Gym', -1): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE STANDING_ORDER; and SELECT so_id, acc_no, payee, amount, frequency FROM STANDING_ORDER ORDER BY so_id;

```
CREATE TABLE STANDING_ORDER (so_id INT PRIMARY KEY, acc_no INT NOT NULL, payee VARCHAR(40) NOT NULL,
amount DECIMAL(10,2) CHECK (amount > 0), frequency VARCHAR(10) DEFAULT 'Monthly', next_run DATE, FOREIGN
KEY (acc_no) REFERENCES ACCOUNT(acc_no));
```

OK - 0 row(s) affected

```
ALTER TABLE ACCOUNT ADD COLUMN pan CHAR(10) UNIQUE;
```

OK - 0 row(s) affected

```
INSERT INTO STANDING_ORDER (so_id, acc_no, payee, amount) VALUES (1, 5001, 'Landlord', 15000);
```

OK - 1 row(s) affected

```
INSERT INTO STANDING_ORDER (so_id, acc_no, payee, amount) VALUES (2, 5001, 'Gym', -1);
```

Error number 3819: Check constraint 'standing_order_chk_1' is violated.

```
DESCRIBE STANDING_ORDER;
```

6 row(s):

Field	Type	Null	Key	Default	Extra
so_id	int	NO	PRI	NULL	
acc_no	int	NO	MUL	NULL	
payee	varchar(40)	NO		NULL	
amount	decimal(10,2)	YES		NULL	
frequency	varchar(10)	YES		Monthly	
next_run	date	YES		NULL	

```
SELECT so_id, acc_no, payee, amount, frequency FROM STANDING_ORDER ORDER BY so_id;
```

1 row(s):

so_id	acc_no	payee	amount	frequency
1	5001	Landlord	15000.00	Monthly

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two transactions (list the columns): (29, 5003, 'Deposit', 2500, '2026-08-01', 'Cash', NULL) and (30, 5004, 'Withdraw', 1800, '2026-08-02', 'ATM', NULL). (b) Insert a transaction with txn_id 1 (other values of your choice): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Add 5000 to the balance of account 5003. (d) Delete every Cash transaction of 3000 or less. (e) Verify with: SELECT acc_no, balance FROM ACCOUNT WHERE acc_no = 5003; and SELECT txn_id, mode, amount FROM TXN WHERE mode = 'Cash' ORDER BY txn_id;

```
INSERT INTO TXN (txn_id, acc_no, txn_type, amount, txn_date, mode, remarks) VALUES (29, 5003, 'Deposit',
2500, '2026-08-01', 'Cash', NULL), (30, 5004, 'Withdraw', 1800, '2026-08-02', 'ATM', NULL);
```

OK - 2 row(s) affected

```
INSERT INTO TXN (txn_id, acc_no, txn_type, amount, txn_date) VALUES (1, 5001, 'Deposit', 100, '2026-08-
03');
```

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

```
UPDATE ACCOUNT SET balance = balance + 5000 WHERE acc_no = 5003;
```

OK - 1 row(s) affected

```
DELETE FROM TXN WHERE mode = 'Cash' AND amount <= 3000;
```

OK - 3 row(s) affected

```
SELECT acc_no, balance FROM ACCOUNT WHERE acc_no = 5003;
```

1 row(s):

acc_no	balance
5003	13500.00

```
SELECT txn_id, mode, amount FROM TXN WHERE mode = 'Cash' ORDER BY txn_id;
```

3 row(s):

txn_id	mode	amount
1	Cash	25000.00
12	Cash	5000.00
24	Cash	4000.00

Set 29 - Operations Control

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

Task 1 - Retrieve data (5 marks)

> Table FLIGHT. Show flight_no, airline, status, total_seats of flights that are 'Delayed' or 'Cancelled' (use IN) OR have fewer than 150 total_seats. Sort by flight_no.

```
SELECT flight_no, airline, status, total_seats FROM FLIGHT WHERE status IN ('Delayed','Cancelled') OR
total_seats < 150 ORDER BY flight_no;
```

5 row(s):

flight_no	airline	status	total_seats
AN503	AirNova	Cancelled	186
BW601	BlueWing	On Time	140
BW702	BlueWing	Delayed	140
SJ202	SkyJet	Delayed	160
SJ303	SkyJet	On Time	120

Task 2 - Summarise data (5 marks)

> (a) Table FLIGHT: for flights that are 'On Time', ONE row with: flights (count), seats (sum of total_seats), avg_fare (ROUND 2), cheapest (lowest base_fare). (b) Table BOOKING: for each flight_no show flight_no, bookings (count) and total_fare (sum of fare_paid, 0 instead of NULL - IFNULL). Keep only flights whose total_fare is more than 15000. Sort by total_fare (highest first), then flight_no.

```
SELECT COUNT(*) AS flights, SUM(total_seats) AS seats, ROUND(AVG(base_fare), 2) AS avg_fare,
MIN(base_fare) AS cheapest FROM FLIGHT WHERE status = 'On Time';
```

1 row(s):

flights	seats	avg_fare	cheapest
7	1106	5314.29	3000.00

```
SELECT flight_no, COUNT(*) AS bookings, IFNULL(SUM(fare_paid), 0) AS total_fare FROM BOOKING GROUP BY
flight_no HAVING IFNULL(SUM(fare_paid), 0) > 15000 ORDER BY total_fare DESC, flight_no;
```

5 row(s):

flight_no	bookings	total_fare
ZA902	3	33700.00
SJ101	3	27000.00
ZA801	2	21980.00
AN301	3	21200.00
BW702	3	16500.00

Task 3 - Design & define (5 marks)

> ER fragment: a FLIGHT can have many MAINTENANCE_LOG entries; each entry belongs to one flight. (a) Create table MAINTENANCE_LOG: log_id INT primary key; flight_no VARCHAR(8) NOT NULL, foreign key to FLIGHT(flight_no); work_done VARCHAR(60) NOT NULL; cost DECIMAL(9,2) with CHECK (cost >= 0); checked CHAR(1) DEFAULT 'N'. (b) Change FLIGHT.airline to VARCHAR(30) NOT NULL (MODIFY). (c) Insert (1, 'SJ101', 'Engine check', 25000) listing log_id, flight_no, work_done, cost. Then insert (2, 'SJ101', 'Tyre change', -10): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE MAINTENANCE_LOG; and SELECT * FROM MAINTENANCE_LOG ORDER BY log_id;

```
CREATE TABLE MAINTENANCE_LOG (log_id INT PRIMARY KEY, flight_no VARCHAR(8) NOT NULL, work_done
VARCHAR(60) NOT NULL, cost DECIMAL(9,2) CHECK (cost >= 0), checked CHAR(1) DEFAULT 'N', FOREIGN KEY
(flight_no) REFERENCES FLIGHT(flight_no));
```

OK - 0 row(s) affected

```
ALTER TABLE FLIGHT MODIFY airline VARCHAR(30) NOT NULL;
```

OK - 0 row(s) affected

```
INSERT INTO MAINTENANCE_LOG (log_id, flight_no, work_done, cost) VALUES (1, 'SJ101', 'Engine check',
25000);
```

OK - 1 row(s) affected

```
INSERT INTO MAINTENANCE_LOG (log_id, flight_no, work_done, cost) VALUES (2, 'SJ101', 'Tyre change',
-10);
```

Error number 3819: Check constraint 'maintenance_log_chk_1' is violated.

```
DESCRIBE MAINTENANCE_LOG;
```

5 row(s):

Field	Type	Null	Key	Default	Extra
log_id	int	NO	PRI	NULL	
flight_no	varchar(8)	NO	MUL	NULL	
work_done	varchar(60)	NO		NULL	
cost	decimal(9,2)	YES		NULL	
checked	char(1)	YES		N	

```
SELECT * FROM MAINTENANCE_LOG ORDER BY log_id;
```

1 row(s):

log_id	flight_no	work_done	cost	checked
1	SJ101	Engine check	25000.00	N

Task 4 - Modify data (5 marks)

> Flight 'SJ202' has been cancelled. (a) Set its status to 'Cancelled' in FLIGHT. Then set booking_status to 'Cancelled' for all its bookings in BOOKING. (b) Try to delete flight 'SJ202' right now: it must be rejected because bookings refer to it. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (c) Delete every Cancelled booking. Then delete every Cancelled flight (this now works). (d) Insert a flight ('ZA903', 'Zephyr', 'Delhi', 'Goa', '2026-11-06', 150, 6000) listing the columns (status defaults). (e) Verify with: SELECT flight_no, status FROM FLIGHT ORDER BY flight_no; and SELECT COUNT(*) AS bookings_left FROM BOOKING;

```
UPDATE FLIGHT SET status = 'Cancelled' WHERE flight_no = 'SJ202';
```

OK - 1 row(s) affected

```
UPDATE BOOKING SET booking_status = 'Cancelled' WHERE flight_no = 'SJ202';
```

OK - 2 row(s) affected

```
DELETE FROM FLIGHT WHERE flight_no = 'SJ202';
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_airline` . `booking` , CONSTRAINT `booking_ibfk_1` FOREIGN KEY (`flight_no`) REFERENCES `flight` (`flight_no`))

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

OK - 4 row(s) affected

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

OK - 2 row(s) affected

```
INSERT INTO FLIGHT (flight_no, airline, source, destination, dep_date, total_seats, base_fare) VALUES
('ZA903', 'Zephyr', 'Delhi', 'Goa', '2026-11-06', 150, 6000);
```

OK - 1 row(s) affected

```
SELECT flight_no, status FROM FLIGHT ORDER BY flight_no;
```

9 row(s):

flight_no	status
AN301	On Time
AN402	On Time
BW601	On Time
BW702	Delayed
SJ101	On Time
SJ303	On Time
ZA801	On Time
ZA902	On Time
ZA903	On Time

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

1 row(s):

bookings_left
22

Set 30 - Seniority & Tenure

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

Task 1 - Retrieve data (5 marks)

> Table EMPLOYEE. Show emp_id, ename, job, hire_date, salary of employees hired BEFORE '2019-01-01' who are Managers or Accountants (use IN). Sort by hire_date (earliest first), then emp_id. Show only the first 4 rows (LIMIT).

```
SELECT emp_id, ename, job, hire_date, salary FROM EMPLOYEE WHERE hire_date < '2019-01-01' AND job IN ('Manager','Accountant') ORDER BY hire_date, emp_id LIMIT 4;
```

4 row(s):

emp_id	ename	job	hire_date	salary
7001	Arvind Kulkarni	Manager	2015-04-01	120000.00
7014	Nalini Bhat	Manager	2016-03-21	98000.00
7006	Fatima Siddiqui	Manager	2016-09-05	105000.00
7011	Kishore Banerjee	Manager	2017-01-16	95000.00

Task 2 - Summarise data (5 marks)

> Table EMPLOYEE. (a) For Managers only, ONE row with: managers (count), min_sal, max_sal, avg_sal (ROUND 2), total_sal. (b) For each job show job and emps (count). Keep only jobs with 2 or more employees. Sort by emps (highest first), then job.

```
SELECT COUNT(*) AS managers, MIN(salary) AS min_sal, MAX(salary) AS max_sal, ROUND(AVG(salary), 2) AS avg_sal, SUM(salary) AS total_sal FROM EMPLOYEE WHERE job = 'Manager';
```

1 row(s):

managers	min_sal	max_sal	avg_sal	total_sal
5	90000.00	120000.00	101600.00	508000.00

```
SELECT job, COUNT(*) AS emps FROM EMPLOYEE GROUP BY job HAVING COUNT(*) >= 2 ORDER BY emps DESC, job;
```

6 row(s):

job	emps
Manager	5
Developer	4
Salesman	4
Support Agent	3
Accountant	2
HR Executive	2

Task 3 - Design & define (5 marks)

> ER fragment: a TRAINING course is attended by many EMPLOYEEs and one employee attends many courses - model it with a relationship table. (a) Create table TRAINING_ATTENDANCE: emp_id INT NOT NULL, foreign key to EMPLOYEE(emp_id); course VARCHAR(30) NOT NULL; hours INT DEFAULT 8 with CHECK (hours > 0); composite PRIMARY KEY (emp_id, course). (b) Rename EMPLOYEE.city to location (RENAME COLUMN). (c) Insert (7003, 'Leadership') listing emp_id and course. Then insert (7003, 'Safety', hours 0): it must be rejected. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Finish with: DESCRIBE TRAINING_ATTENDANCE; and SELECT * FROM TRAINING_ATTENDANCE ORDER BY emp_id, course;

```
CREATE TABLE TRAINING_ATTENDANCE (emp_id INT NOT NULL, course VARCHAR(30) NOT NULL, hours INT DEFAULT 8 CHECK (hours > 0), PRIMARY KEY (emp_id, course), FOREIGN KEY (emp_id) REFERENCES EMPLOYEE(emp_id));
```

OK - 0 row(s) affected

```
ALTER TABLE EMPLOYEE RENAME COLUMN city TO location;
```

OK - 0 row(s) affected

```
INSERT INTO TRAINING_ATTENDANCE (emp_id, course) VALUES (7003, 'Leadership');
```

OK - 1 row(s) affected

```
INSERT INTO TRAINING_ATTENDANCE (emp_id, course, hours) VALUES (7003, 'Safety', 0);
```

Error number 3819: Check constraint 'training_attendance_chk_1' is violated.

```
DESCRIBE TRAINING_ATTENDANCE;
```

3 row(s):

Field	Type	Null	Key	Default	Extra
emp_id	int	NO	PRI	NULL	
course	varchar(30)	NO	PRI	NULL	
hours	int	YES		8	

```
SELECT * FROM TRAINING_ATTENDANCE ORDER BY emp_id, course;
```

1 row(s):

emp_id	course	hours
7003	Leadership	8

Task 4 - Modify data (5 marks)

> (a) With ONE multi-row INSERT add two employees (list the columns; the column formerly called city is now called location): (7023, 'Wasim Akhtar', 'Salesman', 20, 44000, NULL, '2026-09-01', 'Mumbai') and (7024, 'Yamini Rao', 'HR Executive', 30, 47000, NULL, '2026-09-15', 'Jaipur'). (b) Give every employee of department 40 a raise of 3000. (c) Try to delete department 10: it must be rejected because employees belong to it. Write the MySQL error number as a comment under it, e.g. -- Error 1062 (d) Delete the two employees you added in (a). (e) Verify with: SELECT emp_id, salary FROM EMPLOYEE WHERE dept_id = 40 ORDER BY emp_id; and SELECT COUNT(*) AS emps_left FROM EMPLOYEE;

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

OK - 2 row(s) affected

```
UPDATE EMPLOYEE SET salary = salary + 3000 WHERE dept_id = 40;
```

OK - 4 row(s) affected

```
DELETE FROM DEPARTMENT WHERE dept_id = 10;
```

Error number 1451: Cannot delete or update a parent row: a foreign key constraint fails (`eval_hr` . `employee` , CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`dept_id`) REFERENCES `department` (`dept_id`))

```
DELETE FROM EMPLOYEE WHERE emp_id IN (7023, 7024);
```

OK - 2 row(s) affected

```
SELECT emp_id, salary FROM EMPLOYEE WHERE dept_id = 40 ORDER BY emp_id;
```

4 row(s):

emp_id	salary
7014	101000.00
7015	59000.00
7016	57000.00
7017	65000.00

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

1 row(s):

emps_left
22