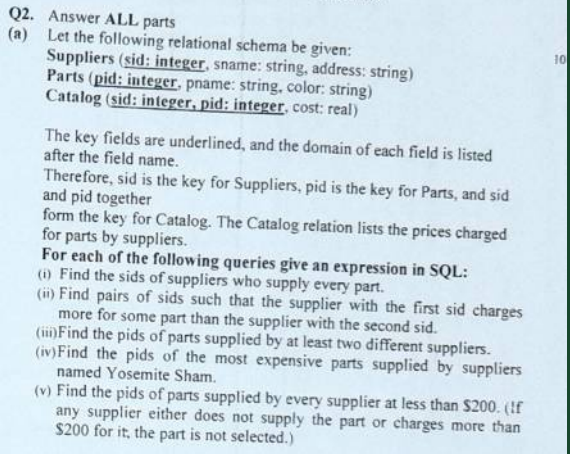
Simple SQL command



Q2. Answer **ALL** parts

(a) Let the following relational schema be given: 10

Suppliers (**sid: integer**, sname: string, address: string)
Parts (**pid: integer**, pname: string, color: string)
Catalog (**sid: integer, pid: integer**, cost: real)

The key fields are underlined, and the domain of each field is listed after the field name.
Therefore, sid is the key for Suppliers, pid is the key for Parts, and sid and pid together
form the key for Catalog. The Catalog relation lists the prices charged for parts by suppliers.

For each of the following queries give an expression in SQL:

(i) Find the sids of suppliers who supply every part.
(ii) Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid.
(iii) Find the pids of parts supplied by at least two different suppliers.
(iv) Find the pids of the most expensive parts supplied by suppliers named Yosemite Sham.
(v) Find the pids of parts supplied by every supplier at less than $200. (If any supplier either does not supply the part or charges more than $200 for it, the part is not selected.)

give simple sql command

Here are **simple SQL queries** for each part, using the given schema.

(i) Find the sids of suppliers who supply every part

```sql
SELECT c.sid
FROM Catalog c
GROUP BY c.sid
HAVING COUNT(DISTINCT c.pid) = (SELECT COUNT(*) FROM Parts);
```

(ii) Find pairs of sids such that first supplier charges more than second for some part

```sql
SELECT c1.sid, c2.sid
FROM Catalog c1, Catalog c2
WHERE c1.pid = c2.pid
  AND c1.sid <> c2.sid
  AND c1.cost > c2.cost;
```

(iii) Find the pids of parts supplied by at least two different suppliers

```sql
SELECT pid
FROM Catalog
GROUP BY pid
HAVING COUNT(DISTINCT sid) >= 2;
```

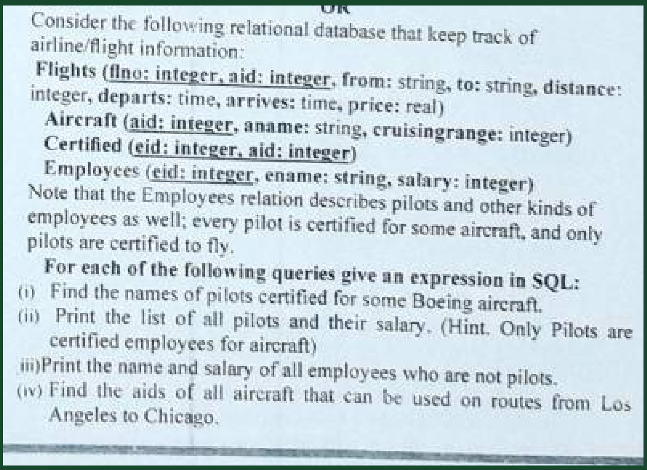
(iv) Find the pids of the most expensive parts supplied by supplier named 'Yosemite Sham'

```sql
SELECT c.pid
FROM Catalog c, Suppliers s
WHERE c.sid = s.sid
  AND s.sname = 'Yosemite Sham'
  AND c.cost = (
      SELECT MAX(c2.cost)
      FROM Catalog c2, Suppliers s2
      WHERE c2.sid = s2.sid
        AND s2.sname = 'Yosemite Sham'
  );
```

(v) Find the pids of parts supplied by every supplier at less than $200

```sql
SELECT p.pid
FROM Parts p
WHERE NOT EXISTS (
    SELECT s.sid
    FROM Suppliers s
    WHERE NOT EXISTS (
        SELECT *
        FROM Catalog c
        WHERE c.pid = p.pid
          AND c.sid = s.sid
          AND c.cost < 200
    )
);
```

If you want, I can also give **JOIN-based versions** or **very short exam-style answers**.
1/2



OR

Consider the following relational database that keep track of airline/flight information:

Flights (**flno: integer, aid: integer**, **from:** string, **to:** string, **distance:** integer, **departs:** time, **arrives:** time, **price:** real)

Aircraft (**aid: integer**, **aname:** string, **cruisingrange:** integer)

Certified (**eid: integer, aid: integer**)

Employees (**eid: integer**, **ename: string, salary: integer**)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly.

For each of the following queries give an expression in SQL:

(i) Find the names of pilots certified for some Boeing aircraft.

(ii) Print the list of all pilots and their salary. (Hint. Only Pilots are certified employees for aircraft)

iii)Print the name and salary of all employees who are not pilots.

(iv) Find the aids of all aircraft that can be used on routes from Los Angeles to Chicago.

this too

Here are **simple SQL queries** for each part (exam-friendly):

(i) Names of pilots certified for some Boeing aircraft

```sql
SELECT DISTINCT e.ename
FROM Employees e, Certified c, Aircraft a
WHERE e.eid = c.eid
  AND c.aid = a.aid
  AND a.aname LIKE '%Boeing%';
```

(ii) All pilots and their salary (only pilots)

```sql
SELECT DISTINCT e.ename, e.salary
FROM Employees e, Certified c
WHERE e.eid = c.eid;
```

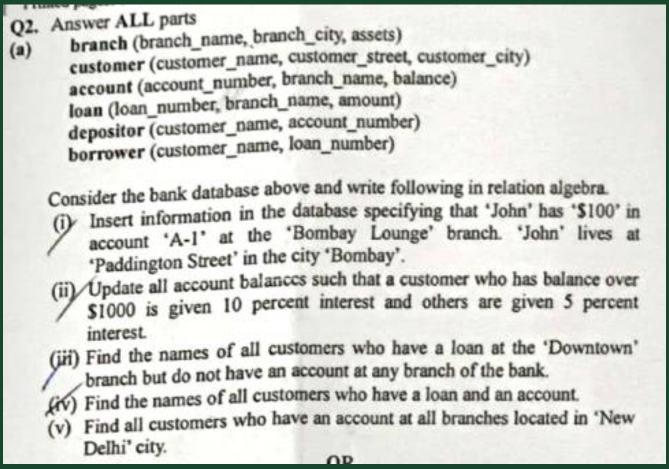
(iii) Name and salary of employees who are not pilots

```sql
SELECT e.ename, e.salary
FROM Employees e
WHERE e.eid NOT IN (
    SELECT eid FROM Certified
);
```

(iv) Aids of aircraft used on routes from Los Angeles to Chicago

```sql
SELECT DISTINCT c.aid
FROM Certified c, Flights f
WHERE f.from = 'Los Angeles'
  AND f.to = 'Chicago'
  AND f.distance <= (
      SELECT a.cruisingrange
      FROM Aircraft a
      WHERE a.aid = c.aid
  );
```

If you want **JOIN-based versions** or **one-line answers**, tell me 👍



Q2. Answer ALL parts

(a) **branch** (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

Consider the bank database above and write following in relation algebra.

(i) Insert information in the database specifying that 'John' has '$100' in account 'A-1' at the 'Bombay Lounge' branch. 'John' lives at 'Paddington Street' in the city 'Bombay'.

(ii) Update all account balancces such that a customer who has balance over $1000 is given 10 percent interest and others are given 5 percent interest.

(iii) Find the names of all customers who have a loan at the 'Downtown' branch but do not have an account at any branch of the bank.

(iv) Find the names of all customers who have a loan and an account.

(v) Find all customers who have an account at all branches located in 'New Delhi' city.

this one

Here are **simple SQL commands** for each part based on the given **bank database**:

(i) Insert information for John

```sql
INSERT INTO customer
VALUES ('John', 'Padding Street', 'Bombay');

INSERT INTO account
VALUES ('A-1', 'Bombay Lounge', 5000);

INSERT INTO depositor
VALUES ('John', 'A-1');
```

(ii) Update balances: > $1000 → 10% interest, others → 5%

```sql
UPDATE account
SET balance = balance * 1.10
WHERE balance > 1000;

UPDATE account
SET balance = balance * 1.05
WHERE balance <= 1000;
```

(iii) Names of customers who have a loan at 'Downtown' branch but no account

```sql
SELECT DISTINCT b.customer_name
FROM borrower b, loan l
WHERE b.loan_number = l.loan_number
  AND l.branch_name = 'Downtown'
  AND b.customer_name NOT IN (
      SELECT customer_name FROM depositor
  );
```

(iv) Names of customers who have both a loan and an account

sql

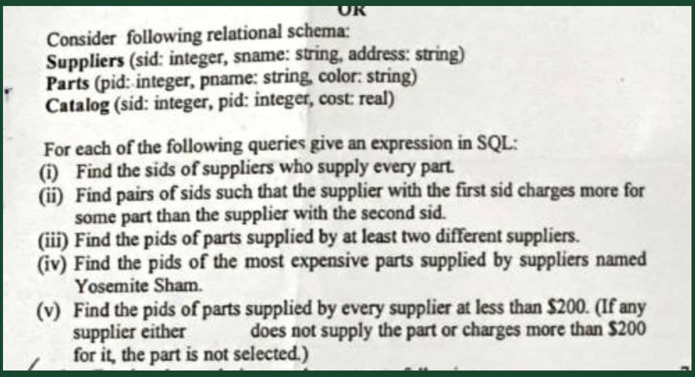
```
SELECT DISTINCT d.customer_name
FROM depositor d
WHERE d.customer_name IN (
    SELECT customer_name FROM borrower
);
```

(v) Customers who have an account at all branches in 'New Delhi'

sql

```
SELECT d.customer_name
FROM depositor d, account a, branch b
WHERE d.account_number = a.account_number
  AND a.branch_name = b.branch_name
  AND b.branch_city = 'New Delhi'
GROUP BY d.customer_name
HAVING COUNT(DISTINCT b.branch_name) = (
    SELECT COUNT(DISTINCT branch_name)
    FROM branch
    WHERE branch_city = 'New Delhi'
);
```

If you want **very short exam-style answers** or **JOIN-only queries**, tell me.
2/2



OR

Consider following relational schema:
Suppliers (sid: integer, sname: string, address: string)
Parts (pid: integer, pname: string, color: string)
Catalog (sid: integer, pid: integer, cost: real)

For each of the following queries give an expression in SQL:

(i) Find the sids of suppliers who supply every part.
(ii) Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid.
(iii) Find the pids of parts supplied by at least two different suppliers.
(iv) Find the pids of the most expensive parts supplied by suppliers named Yosemite Sham.
(v) Find the pids of parts supplied by every supplier at less than $200. (If any supplier either does not supply the part or charges more than $200 for it, the part is not selected.)

give sql command for this

Here are **clear, simple SQL commands** for each query based on the given schema:

(i) SIDs of suppliers who supply every part

sql

```sql
SELECT c.sid
FROM Catalog c
GROUP BY c.sid
HAVING COUNT(DISTINCT c.pid) = (SELECT COUNT(*) FROM Parts);
```

(ii) Pairs of sids where first supplier charges more than second for the same part

sql

```sql
SELECT DISTINCT c1.sid, c2.sid
FROM Catalog c1, Catalog c2
WHERE c1.pid = c2.pid
  AND c1.sid <> c2.sid
  AND c1.cost > c2.cost;
```

(iii) PIDs of parts supplied by at least two different suppliers

sql

```sql
SELECT pid
FROM Catalog
GROUP BY pid
HAVING COUNT(DISTINCT sid) >= 2;
```

(iv) PIDs of the most expensive parts supplied by supplier named 'Yosemite Sham'

sql

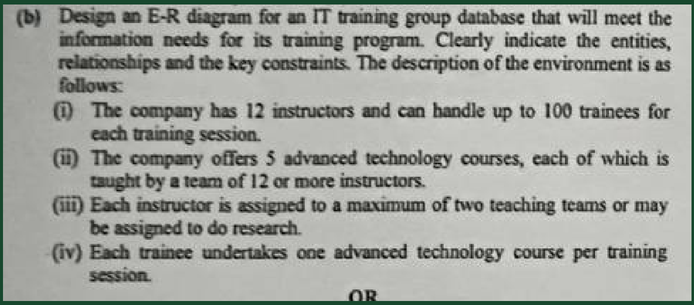
```sql
SELECT c.pid
FROM Catalog c, Suppliers s
WHERE c.sid = s.sid
  AND s.sname = 'Yosemite Sham'
  AND c.cost = (
      SELECT MAX(c2.cost)
      FROM Catalog c2, Suppliers s2
      WHERE c2.sid = s2.sid
        AND s2.sname = 'Yosemite Sham'
  );
```

(v) PIDs of parts supplied by every supplier at less than $200

sql

```sql
SELECT p.pid
FROM Parts p
WHERE NOT EXISTS (
    SELECT s.sid
    FROM Suppliers s
    WHERE NOT EXISTS (
        SELECT *
        FROM Catalog c
        WHERE c.pid = p.pid
          AND c.sid = s.sid
          AND c.cost < 200
    )
);
```

If you want **JOIN-based versions** or **shorter exam answers**, tell me.



(b) Design an E-R diagram for an IT training group database that will meet the information needs for its training program. Clearly indicate the entities, relationships and the key constraints. The description of the environment is as follows:

(i) The company has 12 instructors and can handle up to 100 trainees for each training session.

(ii) The company offers 5 advanced technology courses, each of which is taught by a team of 12 or more instructors.

(iii) Each instructor is assigned to a maximum of two teaching teams or may be assigned to do research.

(iv) Each trainee undertakes one advanced technology course per training session.

OR

explain the make er diagram

Below is a **clear, exam-oriented explanation** of how to **design the E-R diagram** for the given IT training group database.

---

1. Identify Entities and Keys

Instructor

- **Attributes**: `Instructor_ID (PK)`, Name, Expertise
- Constraint: Total instructors = **12**

Trainee

- **Attributes**: `Trainee_ID (PK)`, Name, Qualification
- Constraint: Max **100 trainees per training session**

Course

- **Attributes**: `Course_ID (PK)`, Course_Name, Technology
- Constraint: Total courses = **5 advanced technology courses**

Teaching_Team

- **Attributes**: `Team_ID (PK)`
- Each team consists of **2 or more instructors**

Training_Session

- **Attributes**: `Session_ID (PK)`, Session_Date

---

2. Identify Relationships with Cardinalities

Instructor – Teaching_Team

- Relationship: **ASSIGNED_TO**
- Cardinality:
  - Instructor → Teaching_Team : **0 to 2**
  - Teaching_Team → Instructor : **2 or more**
- Explanation:
  - An instructor can be part of **maximum two teams** or may be assigned to **research** (optional participation).

Teaching_Team – Course

- Relationship: **TEACHES**
- Cardinality:
  - Teaching_Team → Course : **1**
  - Course → Teaching_Team : **1**
- Explanation:
  - Each course is taught by **exactly one team**.

Trainee – Course

- Relationship: **ENROLLS**
- Cardinality:
  - Trainee → Course : **1**
  - Course → Trainee : **many**
- Explanation:
  - Each trainee undertakes **one course per training session**.

Trainee – Training_Session

- Relationship: **ATTENDS**
- Cardinality:
  - Training_Session → Trainee : **up to 100**

- Trainee → Training_Session : **1**

---

3. Key Constraints Summary

Entity	Primary Key
Instructor	Instructor_ID
Trainee	Trainee_ID
Course	Course_ID
Teaching_Team	Team_ID
Training_Session	Session_ID

---

4. Special Constraints (write beside diagram in exam)

- Each **teaching team has at least 2 instructors**
- Each **instructor participates in at most 2 teams**
- Each **course is taught by exactly one team**
- Each **trainee takes exactly one course per session**
- Max **100 trainees per session**

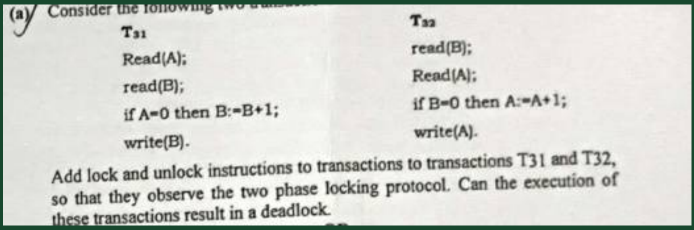
---

5. How to Draw in Exam (Step-wise)

1. Draw **rectangles** for entities
2. Draw **diamonds** for relationships
3. Write **cardinalities (1, M, N)** on each edge
4. Underline **primary keys**
5. Mention constraints near relationships

If you want, I can also:

- Draw a **text-based ER diagram**
- Convert this ER into **relational tables**
- Give a **perfect exam answer (short version)**



(a) Consider the following

T_{31}	T_{32}
Read(A);	read(B);
read(B);	Read(A);
if A=0 then B:=B+1;	if B=0 then A:=A+1;
write(B).	write(A).

Add lock and unlock instructions to transactions to transactions T31 and T32, so that they observe the two phase locking protocol. Can the execution of these transactions result in a deadlock.

solve this

Let the two transactions be **T31** and **T32**.

Given Transactions

T31

```
mathematica

Read(A);
Read(B);
if A = 0 then B = B + 1;
Write(B);
```

T32

```
mathematica

Read(B);
Read(A);
if B = 0 then A = A + 1;
Write(A);
```

Add Locks following Two-Phase Locking (2PL)

Assume:

- **Shared lock (S)** for Read
- **Exclusive lock (X)** for Write
- All locks are acquired before any lock is released

Transaction T31 with Locks

```
scss

Lock-S(A)
Read(A)

Lock-S(B)
Read(B)
```

```
Lock-X(B)
Write(B)

Unlock(B)
Unlock(A)
```

Transaction T32 with Locks

```scss
Lock-S(B)
Read(B)

Lock-S(A)
Read(A)

Lock-X(A)
Write(A)

Unlock(A)
Unlock(B)
```

Possible Execution Leading to Deadlock

Step	T31	T32
1	Lock-S(A)	
2		Lock-S(B)
3	Lock-S(B)	
4		Lock-S(A)
5	Lock-X(B) (waits)	
6		Lock-X(A) (waits)

Deadlock Explanation

- **T31** holds **S(A)** and waits for **X(B)**
- **T32** holds **S(B)** and waits for **X(A)**
- Circular wait condition satisfied → **Deadlock occurs**

Final Answer

✓ **Two-Phase Locking followed**

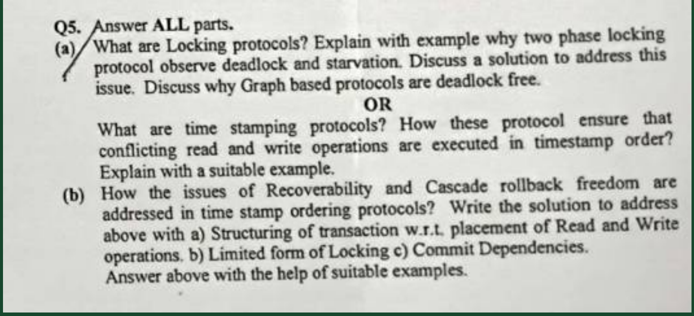
✓ **Deadlock CAN occur**

Conclusion (Exam Line)

> Yes, the execution of T31 and T32 under two-phase locking can result in a deadlock due to circular wait on data items A and B.

If you want, I can also give:

- **Wait-for graph**
- **Deadlock-free schedule**
- **Short 5-mark answer format**



Q5. Answer **ALL** parts.

(a) What are Locking protocols? Explain with example why two phase locking protocol observe deadlock and starvation. Discuss a solution to address this issue. Discuss why Graph based protocols are deadlock free.

OR

What are time stamping protocols? How these protocol ensure that conflicting read and write operations are executed in timestamp order? Explain with a suitable example.

(b) How the issues of Recoverability and Cascade rollback freedom are addressed in time stamp ordering protocols? Write the solution to address above with a) Structuring of transaction w.r.t. placement of Read and Write operations. b) Limited form of Locking c) Commit Dependencies. Answer above with the help of suitable examples.

explain in detail

Below is a **detailed, exam-ready explanation** covering **ALL parts** of Q5 in a structured way.

Q5 (a)

1. Locking Protocols

A **locking protocol** is a set of rules that determines **how transactions acquire and release locks** on database items to ensure **serializability and consistency**.

Types of Locks

- **Shared Lock (S)** → Read only
- **Exclusive Lock (X)** → Read and Write

---

2. Two-Phase Locking (2PL) Protocol

A transaction must follow **two phases**:

(i) Growing Phase

- Transaction can **acquire locks**
- Cannot release any lock

(ii) Shrinking Phase

- Transaction **releases locks**
- Cannot acquire new locks

✓ Ensures **conflict serializability**

---

3. Deadlock in 2PL (with Example)

Example Transactions

```css
T1: Read(A); Write(B)
T2: Read(B); Write(A)
```

Execution

- T1 locks A
- T2 locks B
- T1 waits for B
- T2 waits for A

✓ **Circular wait → Deadlock**

---

4. Starvation in 2PL

Starvation occurs when a transaction **never gets the required lock** because other transactions repeatedly acquire it.

Example

- T1 repeatedly requests X(A)
- Short transactions continuously acquire S(A)
- T1 keeps waiting indefinitely

---

5. Solutions to Deadlock & Starvation

Deadlock Solutions

1. **Deadlock Prevention**

- Wait-Die scheme
- Wound-Wait scheme

2. **Deadlock Detection**
    - Wait-for graph
3. **Deadlock Recovery**
    - Rollback one transaction

Starvation Solution

- Use **aging**
- Assign **priorities**

---

6. Why Graph-Based Protocols are Deadlock-Free

Graph-Based (Tree) Protocol

- Database items are organized as a **directed acyclic graph**
- A transaction can lock a node **only after locking its parent**

✔️ **No cycles** → **No circular wait** → **No deadlock**

OR

7. Time-Stamping Protocols

Each transaction gets a **unique timestamp** when it starts.

Rules

- Older transaction → **smaller timestamp**
- Conflicting operations are ordered by timestamp

---

8. How Conflicting Operations are Executed

Let:

- TS(T1) < TS(T2)

Read Rule

- If T2 already wrote X → abort T1

Write Rule

- If T2 read or wrote X → abort T1

Example

```scss
T1 (TS=5): Write(A)
T2 (TS=10): Read(A)
```

✔️ Allowed (timestamp order preserved)

```scss
T1 (TS=10): Write(A)
T2 (TS=5): Read(A)
```

❌ T1 aborted

Q5 (b)

1. Recoverability in Timestamp Protocol

A schedule is **recoverable** if a transaction **commits only after** the transactions whose data it read have committed.

Problem

- Timestamp protocol allows **dirty reads**
- Can cause **cascading rollback**

---

2. Cascading Rollback

Occurs when:

- T2 reads data written by T1
- T1 aborts
- T2 must also abort

---

3. Solutions

(i) Structuring Transactions (Read/Write Placement)

- Delay reads until writes are committed
- Avoid reading uncommitted data

✓ Reduces cascading rollback

(ii) Limited Form of Locking

- Use **Strict Timestamp Ordering**
- Delay writes until commit

✓ Ensures **recoverability + cascadeless schedules**

(iii) Commit Dependencies

- Track dependencies between transactions
- A transaction can commit only if all transactions it depends on have committed

✓ Guarantees recoverability

Summary Table

Concept	Ensures
2PL	Serializability
Timestamp Protocol	Serial order execution
Graph-based Protocol	Deadlock-free
Strict TO	Cascadeless & recoverable
Commit Dependency	Safe commit order

Exam Tip (Final Line)

> Locking protocols ensure serializability but may cause deadlock and starvation, while timestamp-based and graph-based protocols avoid deadlocks and improve recoverability with proper constraints.

If you want:

- **Short 10-mark version**
- **Diagrams**
- **Numerical examples**
- **Answer in bullet-only format**

Just tell me 👍