placementpapers.app

Zomato Placement Papers 2026

28 MCQs with answer key · Free download

Exam pattern

CS MCQs (OS, DBMS, OOPs, CN, DSA) plus coding-logic themes on graphs, DP, and strings; later-round topics include rate limiters, payments idempotency, concurrency, caching, and SQL. Practice set — not an official Zomato paper.

Questions

Operating Systems

Q1. A Zomato order-worker process forks twice (no exec). Ignoring errors, how many total processes exist after both forks complete (including the original)?

A) 2
B) 3
C) 4
D) 1

Q2. Delivery microservices update shared rider-location state. Which best matches concurrency vs parallelism in this setting?

A) Concurrency is overlapping task progress; parallelism is simultaneous execution on multiple cores
B) They are identical terms for multi-threading
C) Parallelism only applies to I/O; concurrency only to CPU
D) Concurrency requires multiple CPUs; parallelism does not

Q3. Four riders hold restaurant locks and wait for each other in a cycle. Which Coffman condition is this?

A) Mutual exclusion only
B) Hold and wait only
C) Circular wait
D) No preemption only

DBMS

Q4. Payment capture for an order must either fully succeed or fully roll back. Which ACID property is primary?

A) Durability
B) Atomicity
C) Isolation
D) Availability

Q5. SQL for top restaurants by order count in last 30 days with at least 50 orders. Correct pattern?

A) WHERE COUNT(*) >= 50 AND AVG(rating)
B) GROUP BY restaurant_id HAVING COUNT(*) >= 50 ORDER BY COUNT(*) DESC
C) ORDER BY COUNT(*) without GROUP BY
D) SELECT DISTINCT restaurant_id only

Q6. Yearly revenue table (year, revenue). Best SQL idea to compare each year to the previous year?

A) Self-join on year = year+1 or window LAG(revenue) OVER (ORDER BY year)
B) GROUP BY without ordering
C) DELETE then INSERT
D) CROSS JOIN all years to themselves only

Q7. Hot restaurant pages are read-heavy. Which index choice is most appropriate for WHERE city_id = ? AND is_open = true?

A) No index; full table scan is fine at Zomato scale
B) Composite index on (city_id, is_open) matching the filter order
C) Only a bitmap index on the primary key
D) Hash index on the entire row blob

OOPs

Q8. In a food-delivery domain model, Order -> PaymentGatewayAdapter -> RazorpayClient. Which OOP idea is this primarily?

A) Multiple inheritance of payment classes only
B) Dependency inversion / adapter to decouple domain from a vendor SDK
C) Friend classes in C++ only
D) Operator overloading for order totals

Q9. Constructors run A then B then C; destructors should run in which order for stack objects C derived from B derived from A?

A) A, B, C
B) C, B, A
C) B, A, C
D) Random order

Computer Networks

Q10. Live rider GPS pings tolerate occasional loss but need low latency. Prefer:

A) TCP only, because UDP cannot carry IP
B) UDP-style datagrams or QUIC/UDP-based stacks for loss-tolerant telemetry; TCP for payment APIs
C) SMTP for location updates
D) FTP for every GPS point

Q11. Client resolves zomato.com before TLS. Which step is first in typical DNS resolution?

A) Ask the authoritative mail server for MX only
B) Check local stub resolver/OS cache, then recursive resolver toward root/TLD/authoritative
C) Open a raw TCP connection to port 80 first
D) Broadcast ARP for the domain name

DSA Patterns

Q12. Peak lunch demand over a day is modeled as max contiguous sum of minute-level order deltas (can be negative). Best classic approach?

A) Bubble sort the deltas
B) Kadane's algorithm (maximum subarray)
C) Dijkstra on a complete graph of minutes
D) Counting sort only

Q13. City map: restaurants as nodes, roads as non-negative weighted edges. Fastest ETA from warehouse W to restaurant R?

A) Floyd-Warshall always, even for one query on a sparse map
B) Dijkstra (or A*) from W; prefer heap-based Dijkstra on sparse graphs
C) BFS ignoring weights
D) DFS preorder only

Q14. Coupon values [1,2,5], target discount amount 11. Minimum coupons (unlimited supply) is a classic:

A) Greedy always taking largest only (always optimal for all sets)
B) Unbounded knapsack / coin-change DP
C) Binary search on the coupon array only
D) Prim's MST

Q15. Kitchen stations must finish dishes before plating; edges = prerequisites. Detect impossible menu = cycle. Algorithm?

A) Topological sort / Kahn BFS or DFS colors; cycle => no valid order
B) Binary search on station IDs
C) Kadane on the adjacency matrix
D) Two pointers on the edge list only

Q16. Promo codes along a street; cannot pick two adjacent houses (House Robber). Optimal DP idea?

A) dp[i] = max(dp[i-1], dp[i-2] + a[i])
B) dp[i] = dp[i-1] + a[i] always
C) Sort and take every element
D) Dijkstra from house 0

Q17. Implement LRU for restaurant menu cache: get/put in average O(1). Structure?

A) Array + linear scan only
B) HashMap + doubly linked list
C) Unbalanced BST only
D) Single queue without map

Q18. LCS of two menu-name strings is O(n*m) DP. If both strings have all unique characters, a faster approach can use:

A) Only bubble sort
B) Map positions and reduce to LIS-style / segment-tree accelerated methods discussed in interviews
C) Hashing that always runs in O(1)
D) Ignoring one string

System Scenarios

Q19. Design a rate limiter for Zomato APIs (per user/IP). Which statement is most accurate?

A) A single global counter without windows is enough at multi-DC scale
B) Token bucket / sliding window with Redis (or in-memory + sticky) are common; watch clock skew and memory
C) Rate limiting must be done only in the mobile app
D) DNS TTLs replace rate limits

Q20. Payments API: client retries after timeout; gateway may have already charged. Critical property?

A) Idempotency keys so duplicate requests do not double-charge
B) Disable all retries forever
C) Use UDP for payment confirmations
D) Store card CVV in application logs

Q21. Real-time order tracking: prefer which cache role?

A) Redis (or similar) for hot location/status; durable store for order source of truth
B) Only CSV files on one laptop
C) Email inbox as the cache
D) Blockchain for every GPS ping

Q22. Under network partition, live tracking must stay up even if replicas disagree briefly. CAP trade-off leaning toward:

A) CA only (drop partition tolerance)
B) AP with eventual consistency for location fan-out; stronger consistency for payment capture
C) CP for GPS pings and AP for money always
D) Ignore CAP; it does not apply to mobile apps

Q23. Thread-safe HashMap for in-memory promo catalogue. Wrong approach?

A) Coarse mutex around map operations
B) ConcurrentHashMap / striped locks
C) Unsynchronized HashMap shared across writers with no locks
D) Read-write lock if reads dominate

Q24. Golang order service: many outbound restaurant calls. Best concurrency primitive pair for fan-out with timeout?

A) Goroutines + context cancellation/timeouts (and often channels/errgroup)
B) Busy-wait loops on one OS thread only
C) Busy fork-bomb of processes per HTTP header
D) Disable timeouts to maximize latency

Quantitative

Q25. 15 partners deliver 450 orders in 6 hours. At same pace, orders by 25 partners in 10 hours?

A) 1000
B) 1250
C) 750
D) 1500

Q26. Mix rice Rs 40/kg and Rs 60/kg in ratio 3:2; sell at Rs 55/kg. Approx profit %?

A) 10%
B) 14.58%
C) 20%
D) 25%

Reasoning

Q27. Avg delivery time of 5 orders = 32 min; after 6th order avg = 35. Time of 6th order?

A) 35 min
B) 47 min
C) 50 min
D) 53 min

Q28. Batch sizes: 2, 6, 12, 20, 30, 42, ? Pattern n(n+1).

A) 56
B) 48
C) 54
D) 60

Answer Key & Explanations

Operating Systems

Q1. C
Each fork doubles the current count: 1 -> 2 -> 4. Classic OS MCQ style reported in Zomato technical screens (fork output / process count).

Q2. A
Interviewers (Zomato SDE rounds) probe concurrency vs parallelism with examples: overlapping vs truly simultaneous work

Q3. C
A wait-for cycle is circular wait - one of four necessary deadlock conditions. Prevention often imposes lock ordering on resource types.

DBMS

Q4. B
Atomicity = all-or-nothing. Medium SDE-1 reports stress payments backends where mid-failure must not leave half-updated order/payment rows.

Q5. B
Aggregates filter in HAVING after GROUP BY. Product/analyst Zomato tracks repeatedly test joins + GROUP BY + HAVING on orders/restaurants.

Q6. A
Product Analyst interviews (Medium) ask YoY revenue diffs; LAG or self-join on consecutive years is the expected pattern.

Q7. B
Composite B-tree indexes matching selective equality filters speed restaurant listing/search paths discussed in Zomato eng interviews.

OOPs

Q8. B
Zomato interviewers ask where the product uses OOPs; adapters/interfaces around payments/notifications are realistic answers.

Q9. B
Construction runs base-to-derived; destruction is reverse order.

Computer Networks

Q10. B
TCP is reliable ordered delivery; UDP is loss-tolerant for telemetry-style traffic.

Q11. B
DNS resolution flow (cache -> recursive -> hierarchy) appears in Zomato x Blinkit technical rounds alongside Docker/K8s/ Redis.

DSA Patterns

Q12. B
Kadane / max subarray is a standard Zomato coding-prep staple for contiguous peak windows.

Q13. B
LCS with a unique-character constraint often leads to LIS / segment-tree style approaches.

Q14. B
Coin change DP is repeatedly listed in Zomato coding question banks (min coins for amount).

Q15. A
Blinkit/Zomato DSA rounds include story problems reducing to topo sort + graph coloring/cycle checks.

Q16. A
House Robber / circular variants appear in Blinkit SDE-1 problem-solving rounds (with follow-ups).

Q17. B
LRU (modified) is a known Zomato on-campus interview problem; HashMap + DLL is the standard O(1) design.

Q18. B
Zomato SDE round: LCS then constraint 'unique characters' leading to improved approaches (LIS / segment tree ideas)

System Scenarios

Q19. B
Zomato x Blinkit Round 2 explicitly asked to design a rate limiter with edge cases and memory optimization.

Q20. A
Aug 2025 Medium SDE-1: payments backend round covered idempotency, retries, concurrency, and failure midway.

Q21. A
Concurrency is overlapping progress; parallelism is truly simultaneous execution.

Q22. B
Prep material maps Zomato tracking to AP/eventual consistency while payments need stronger consistency - a realistic split.

Q23. C
Zomato round: implement HashMap then make it thread-safe with mutex - unsynchronized shared mutation is incorrect

Q24. A
Zomato x Blinkit advanced round: implement request logic in Go with goroutines/channels plus timeout edge cases.

Quantitative

Q25. B
Rate = 450/(15*6) = 5 orders/partner-hour. 5*25*10 = 1250. Delivery-work aptitude pattern used in public Zomato paper collections.

Q26. B
CP = (3*40+2*60)/5 = 48; profit = 7; 7/48*100 ~ 14.58%.

Reasoning

Q27. C
Totals 160 then 210; sixth = 50 minutes.

Q28. A
7*8 = 56. Common series item in Zomato-style aptitude sets.