Exam Report

12 august 2025
5:15pm

Daud Eid Mouse
Application administrator
exam evaluation

Nice — great work powering through all that. I graded everything one-by-one like you asked. I focused on *understanding* over wording, corrected mistakes, and gave clear, short explanations and examples you can use to study.

✅ Section 1 — Multiple Choice (30)

You answered **27 / 30 correct (90%)**. Below I mark each question as **Correct** or **Wrong**, and explain any wrong answers.

1. **(b) — Correct.** Application Admin = maintain/configure/troubleshoot apps.
2. **(d) — Correct.** Designing logos is not part of app maintenance.
3. **(c) — Correct.** Staging is for testing before production.
4. **(b) — Correct.** Redis is a key-value store used for caching.
5. **(c) — Correct.** git pull downloads changes from remote.
6. **(b) — Correct.** First check application logs for 500 errors.
7. **(b) — Correct.** Patches improve security & stability.
8. **(a) — Correct.** Grafana is used for real-time monitoring/visualization.
9. **(a) — Correct.** Bash is widely used for Linux automation.
10. **(c) — Wrong. Correct answer: (a) PostgreSQL.**
    Explanation: PostgreSQL **is** a relational database. You chose (c) which was MongoDB — MongoDB is a NoSQL document store, not relational.
11. **(a) — Correct.** Staging = testing pre-release.
12. **(b) — Correct.** Investigate running processes / metrics first on CPU spike.
13. **(b) — Correct.** Git = version control.
14. **(b) — Correct.** Escalate when vendor code changes are required.
15. **(c) — Wrong. Correct answer: (a) .sql.**
    Explanation: Database backups are typically .sql dumps (e.g., mysqldump → .sql). CSVs are used for data exports but not the typical full backup format.
16. **(a) — Correct.** Test in staging before production.
17. **(b) — Correct.** Active monitoring sends alerts on problems.
18. **(a) — Correct.** SELECT * FROM table_name; is correct SQL.
19. **(a) — Correct.** Backups are a typical Bash automation task.
20. **(c) — Correct.** Excel is not a DB (so it's the NOT-database choice).
21. **(d) — Correct.** "App running too fast" is not a cause of slowness.
22. **(a) — Correct.** API = Application Programming Interface.
23. **(a) — Correct.** Golang is used for backend/API work.
24. **(a) — Correct.** Monitor disk usage to avoid running out of storage.
25. **(a) — Correct.** Apache is web server software.
26. **(a) — Correct.** Backups before updates in case rollback is needed.
27. **(b) — Correct.** Redis = caching to speed requests.
28. **(d) — Wrong. Correct answer: (a) git clone.**
    Explanation: git clone creates a local copy of a remote repo. git merge does not clone.
29. **(b) — Correct.** Follow incident response procedures.
30. **(b) — Correct.** Hotfix = urgent fix applied to production.

📝 Section 2 — Structured (20)

You scored **10.5 / 20 (52.5%)** — I marked each item and gave a corrected/clear answer you can study.

1. Separation of production and staging — *Your answer:* good.

- **Evaluation: Correct.** Add: staging should mirror production (data masked) for realistic tests and to avoid surprises on release.

2. Four databases — *You gave:* mysql, postgresql, redis, mangoDB

- **Evaluation: Mostly correct (typo).** Should be **MySQL, PostgreSQL, Redis, MongoDB**. Note Redis is in-memory key-value store; MongoDB is document DB; MySQL/Postgres are relational.

3. Steps for troubleshooting a web app error — *You wrote:* find problem, clear cache, run queries.

- **Evaluation: Partial (0.5).** Good start, but missing key diagnostic steps.
  Recommended step-by-step:
  1. Reproduce the error (if possible) and note exact message/time.
  2. Check monitoring (response time, error rates).
  3. Inspect logs (web server, application, database) for stack traces/timeouts.
  4. Check recent deployments or config changes.
  5. Check upstream services (DB, cache, external APIs) & connectivity.
  6. Try safe remediation (restart app process/service) if appropriate.
  7. Escalate to dev/vendor with logs and reproduction steps if code fix needed.

4. Why system monitoring is essential — *You:* good explanation.

- **Evaluation: Correct.** Add: monitoring gives alerts, historical trends, and capacity planning ability.

5. SQL: all customers from "Hargaysa" — *You wrote:* SELECT * FROM customers

- **Evaluation: Incorrect.** Correct example (assuming column city):

sql
CopyEdit
SELECT * FROM customers WHERE city = 'Hargaysa';
If the column is location or address, use that column name.

6. Two Python automation tasks — *You:* vague.

- **Evaluation: Partial (0.5).** Better examples:
  - Automated daily backups & upload to secure storage (S3).
  - Script to parse error logs and create daily summary emails / tickets.
  - API integration job that syncs data between services.

7. Difference: application uptime vs performance — *You:* reversed/incorrect.

- **Evaluation: Incorrect.** Correct:
  - **Uptime** = availability (% of time service is reachable).
  - **Performance** = how fast/responsive (latency, throughput, error rate). Example: uptime 99.9% but slow performance means app is available but slow.

8. Three benefits of Git — *You:* good (collaboration, merge, commit).

- **Evaluation: Correct.** Add benefits: version history, branching, rollback, collaboration.

9. Why companies use vendors — *You:* unclear.

- **Evaluation: Partial (0.5).** Correct reasons: vendors provide specialized expertise, support & SLAs, faster time-to-market, lower maintenance overhead, and often features you'd otherwise have to build.

10. Bash compress /var/log into logs.tar.gz — *You:* just shebang/echo.

- **Evaluation: Incorrect.** Correct command:

bash
CopyEdit
sudo tar -czf logs.tar.gz /var/log
Or from within script:
bash
CopyEdit
#!/bin/bash
tar -czf /path/to/logs_$(date +%F).tar.gz /var/log

11. How caching improves performance — *You:* partly right but mixed.

- **Evaluation: Partial (0.5).** Clear answer: caching stores frequently requested data (in memory) so the app avoids expensive DB calls; reduces latency & DB load. Examples: page cache, Redis caching query results, CDN for static assets.

12. When to apply security patch immediately — *You:* gave some scenarios.

- **Evaluation: Partial (0.5).** High-priority examples:
  - Active exploit is in the wild for that CVE (public exploit).
  - Patch fixes remote code execution or privilege escalation.
  - Patch fixes authentication bypass or data leak vulnerability.
  - If system compromise is suspected, apply/mitigate immediately.

13. What "least privilege" means — *You:* vague.

- **Evaluation: Incorrect.** Correct: **Grant users, services, and processes only the minimum permissions they need to perform their tasks** (reduce attack surface). Example: DB user for reporting should not have DDL or admin privileges.

14. Why document system changes — *You:* reasonable.

- **Evaluation: Correct.** Add: documentation supports audits, rollback, knowledge transfer, and troubleshooting.

15. Three causes for slow DB performance — *You:* incorrect list.

- **Evaluation: Incorrect.** Correct common causes:
  - Slow/inefficient queries (missing indexes, bad joins).
  - Lock contention or long transactions.
  - Insufficient memory/CPU/disk I/O bottlenecks.
  - Large table scans, lack of indexing, network latency.

16. Example to escalate to Level 2 — *You:* input validation bug + ticket — good.

- **Evaluation: Correct.** Also escalate if the fix requires vendor code changes or access you don't have.

17. Why backup before deploying — *You:* "to not lose if changes wrong" — correct.

- **Evaluation: Correct.** Add: validate backup integrity, have rollback plan & test restore procedure.

18. One daily routine check — *You:* monitor performance — ok but generic.

- **Evaluation: Partial (0.5).** Example concrete daily checks: confirm last backup completed, check APdex/response times, check error rate/alerts, verify queue lengths.

19. SQL to delete all rows from temp_data — *You:* DELETE TEMP_DATA — **Wrong**.

- **Correct:**

sql
CopyEdit
DELETE FROM temp_data;
-- or faster if you want truncation and don't need transaction logs:
TRUNCATE TABLE temp_data;

20. How a monitoring alert system works — *You:* general idea — partial.

- **Evaluation: Partial (0.5).** Clear flow: monitoring agents collect metrics → metrics stored → alert rules define thresholds → when threshold breached alerting system sends notifications (email/SMS/Slack) with runbook/escalation steps → on-call follows incident process.

🔁 Section 3 — True / False (15)

You got **11 / 15 correct (73.3%)**. Mistakes flagged with short explanation.

1. **False** — *Correct.* (You: false)
2. **True** — *Correct.* (You: true)
3. **False** — *Correct.* Redis is not relational. (You: false)
4. **False** — *Wrong*. You answered **True**.
    - **Why:** Staging servers are **not** for production work — they are for testing before production. Statement is false.
5. **True** — *Correct.* (You: true)
6. **True** — *Correct.* (You: true)
7. **False** — *Wrong*. You answered **True**.
    - **Why:** 99.99% uptime means *very little* downtime (about ~4.3 minutes monthly), not many hours. So the statement "means system was down for many hours per month" is **false**.
8. **True** — *Wrong*. You answered **False**.
    - **Why:** High CPU alert *can* indicate inefficient code, memory leaks, infinite loops, or heavy queries — so the statement is true.
9. **False** — *Correct.* (You: false) — vendors **are** contacted for bug fixes.
10. **True** — *Correct.* (You: true)
11. **True** — *Correct.* (You: true)
12. **False** — *Wrong*. You answered **True**.

- **Why:** Production outages should **not** be solved by immediately restarting without investigation. Follow incident response — capture logs, diagnose, then remediate and document. Immediate restart may lose evidence.

13. **True** — *Correct.* (You: true)
14. **True** — *Correct.* (You: true)
15. **False** — *Correct.* (You: false) — Admins **do** interact with end users (support, gather requirements).

📊 **Scores Summary**

- Multiple Choice: **27 / 30 → 90%**
- Structured: **10.5 / 20 → 52.5%**
- True/False: **11 / 15 → 73.3%**

Overall: 48.5 / 65 → 74.6%

Nice job — solid on conceptual MCQs. Structured answers are where you lost most points (need clearer, concrete commands/procedures and correct SQL/Bash syntax).