KEY

(a) = Relational Algebra
(b) = Domain Relational Calculus
(c) = SQL

(1)

(a) $\pi_{cID}(\sigma_{credits = 3}(course))$

(b) {<cID> | course(cID, clevel, cName, prerequisite, duration, credits) $\wedge$ credits = 3}

(c) SELECT cID FROM course WHERE credits = 3;

(2)

(a) $\pi_{cID}(\sigma_{dName = 'Computer\ Science'}(offers))$

(b) {<cID> | course(cID, clevel, cName, prerequisite, duration, credits) AND offers(dName, cID) AND dName = “computer science”}

(c) SELECT cID FROM offers WHERE dName = 'Computer Science';

(3)

(a) $\pi_{dName}(\sigma_{credits=3}(course \bowtie offers)) - \pi_{dName}(\sigma_{credits=4}(course \bowtie offers))$

(b) {<dName> | department(dName, location) AND (∃cID)(course(cID, clevel, cName, prerequisite, duration, credits) AND offers(dName, cID) AND credits = 3) AND (∀cID)(course(cID, clevel, cName, prerequisite, duration, credits) AND offers(dName, cID) → credits ≠ 4)}

(c)
```
SELECT offers.dname
FROM course, offers
WHERE course.cid = offers.cid
        AND course.credits = 3 AND course.credits != 4;
```

(4)

(a) $\pi_{dName}(\sigma_{max(count(sID))}(enrolled \bowtie offers))$

(b) {<dName> | department(dName, location) AND (∀d)(department(d, location) → (∃sID, cID)(enrolled(sID, cID) AND offers(dName, cID) AND (∀s, c)(enrolled(s, c) AND offers(d, c) → dName = d)))}

(c)
```
SELECT
        offers.dname,
        COUNT(enrolled.sid) AS enrolled_count
FROM offers, enrolled
WHERE offers.cid = enrolled.cid
GROUP BY offers.dname
ORDER BY COUNT(enrolled.sid) DESC
LIMIT 1;
```

(5)

(a) $\pi_{\text{instID, instName}}(\sigma_{\text{max(sum(credits))}}(\text{instructor} \bowtie \text{teaches} \bowtie \text{course}))))$

(b) {<instID, instName> | instructor(instID, instName, office) AND (∀i)(instructor(i, n, o) → (∃cID)(teaches(instID, cID) AND course(cID, clevel, cName, prerequisite, duration, credits) AND (∀c)(teaches(i, c) AND course(c, l, n, p, d, cr) → credits ≥ cr)))}

(c)
```
SELECT
        instructor.instid,
        instructor.instname,
        SUM(course.credits) AS credits_sum
FROM instructor, teaches, course
WHERE instructor.instid = teaches.instid
        AND teaches.cid = course.cid
GROUP BY instructor.instid
ORDER BY SUM(course.credits) DESC
LIMIT 1;
```

(6)

(a) $\pi_{\text{cID, cName}}(\sigma_{\text{count(dName)} > 1}(\pi_{\text{cID}}(\text{course} \bowtie \text{offers})))$

(b) {<cID, cName> | course(cID, clevel, cName, prerequisite, duration, credits) AND (∃d1, d2)(offers(d1, cID) AND offers(d2, cID) AND d1 ≠ d2)}

(c)
```
SELECT
        course.cid,
        course.cname
FROM course, offers
WHERE course.cid = offers.cid AND course.cid IN (
        SELECT course.cid
        FROM course, offers
        WHERE course.cid = offers.cid
        GROUP BY course.cid
        HAVING COUNT(offers.dname) > 1
);
```

(7)

(a) $\pi_{sID}(\sigma_{dName = \text{‘Computer science’}}(enrolled \bowtie offers)) \cap \pi_{sID}(\sigma_{dName = \text{‘Mathematics’}}(enrolled \bowtie offers))$

(b) {<sID> | student(sID, sName, dob) AND ($\exists$ c1, c2)(enrolled(sID, c1) AND enrolled(sID, c2) AND offers(“Computer science”, c1) AND offers(“Mathematics”, c2))}

(c)
```
SELECT enrolled.sid
FROM enrolled
WHERE enrolled.cid IN (
        SELECT offers.cid
        FROM offers
        WHERE offers.dname = 'Computer science'
) AND enrolled.cid IN (
        SELECT offers.cid
        FROM offers
        WHERE offers.dname = 'Mathematics'
);
```

(8)

(a) $\pi_{cID,\ cName}(\sigma_{clevel\ \text{LIKE ‘4\%’}}(course)) \cap \pi_{cID,\ cName}(\sigma_{clevel\ \text{LIKE ‘6\%’}}(course))$

(b) {<cID, cName> | course(cID, clevel, cName, prerequisite, duration, credits) AND ($\exists$ c)(course(c, clevel, cName, prerequisite, duration, credits) AND cID $\neq$ c AND clevel = 4% AND clevel = 6%)}

(c)
```
SELECT
        course.cid,
        course.cname
FROM course
WHERE course.clevel LIKE '4%' AND course.clevel IN (
        SELECT course.clevel
        FROM course
        WHERE course.clevel LIKE '6%'
);
```

SQL ONLY

(9)

```
SELECT
        offers.dname,
        course.clevel,
        COUNT(course.clevel) AS clevel_count
FROM course, offers
WHERE course.cid = offers.cid
GROUP BY offers.dname, course.clevel
ORDER BY offers.dname ASC, course.clevel ASC;
```

(10)

```
SELECT
        enrolled_count.dname,
        AVG(enrolled_count) AS avg_enrolled_count,
        MIN(enrolled_count) AS min_enrolled_count,
        MAX(enrolled_count) AS max_enrolled_count
FROM (
        SELECT
        offers.dname,
        COUNT(enrolled.sid) AS enrolled_count
        FROM offers, enrolled
        WHERE offers.cid = enrolled.cid
        GROUP BY offers.dname, enrolled.sid
) AS enrolled_count
GROUP BY enrolled_count.dname
ORDER BY avg_enrolled_count DESC;
```