# ISYS2120 Final Exam - Harder Practice Exam: Answer Key

This document provides answers to the practice exam questions. Remember, this is just a guide, and your own understanding of the concepts is crucial.

**Section A: SQL Queries**

**Q1.**

```
SELECT CId
FROM Shipment
GROUP BY CId
HAVING COUNT(DISTINCT SId) > 1
AND SUM(CASE WHEN SPQty > 0 THEN 1 ELSE 0 END) > 0;
```

**Q2.**

```
SELECT DISTINCT C.CName, C.CAddr
FROM Customer C
JOIN Shipment S ON C.CId = S.CId
JOIN Contains CO ON S.SId = CO.SId
JOIN Part P ON CO.PId = P.PId
WHERE P.PDesc LIKE 'Small%';
```

**Q3.**

```
SELECT S.SId,
       SUM(P.PPrice * CO.SPQty) * (1 - C.CLoyaltyDiscount / 100) AS Tot
FROM Shipment S
JOIN Contains CO ON S.SId = CO.SId
JOIN Part P ON CO.PId = P.PId
JOIN Customer C ON S.CId = C.CId
GROUP BY S.SId;
```

**Q4.**

```
SELECT AVG(P.PPrice)
FROM Part P
JOIN Contains CO ON P.PId = CO.PId
JOIN Shipment S ON CO.SId = S.SId
WHERE S.SDate < '2022-01-20';
```

**Q5.**

```
SELECT P.PId, P.PDesc
FROM Part P
JOIN Contains CO ON P.PId = CO.PId
JOIN Shipment S ON CO.SId = S.SId
JOIN Customer C ON S.CId = C.CId
WHERE C.CLoyaltyDiscount > 10
GROUP BY P.PId, P.PDesc
HAVING COUNT(DISTINCT S.SId) >= (SELECT COUNT(DISTINCT SId) FROM Shipme
```

**Section B: Relational Concepts**

**Q6.**

- **Purpose of Primary Keys:** Primary keys ensure data uniqueness and integrity in relational databases. They act as identifiers for each row, preventing duplicate entries and facilitating efficient data retrieval.
- **Benefits of Composite Keys:**
  - **More specific identification:** Composite keys can represent relationships between entities more accurately than single-attribute keys.
  - **Flexibility:** They allow for a combination of attributes to uniquely identify a record, which can be useful in situations where a single attribute is not sufficient.
  - **Data integrity:** They enforce relationships between entities and ensure data consistency.

**Q7.**

```
π SId (σ (PPrice > 10) (Contains) ∩ σ (CLoyaltyDiscount >= 20) (Custome
```

**Q8.**

- **Index Concept:** An index is a data structure that allows for fast retrieval of records based on specific attribute values. It works like a lookup table, mapping attribute values to their corresponding row locations.
- **Efficiency Improvement:** An index on `SDate` would significantly improve the query performance. Instead of scanning the entire `Shipment` table, the database could directly access the relevant records based on the date range specified in the WHERE clause. This avoids unnecessary table scans and speeds up the query execution.

**Q9.**

- **ER Diagram:**
  - **Entities:** Part, Customer, Shipment, Contains
  - **Attributes:**
    - **Part:** PId, PDesc, PPrice
    - **Customer:** CId, CName, CAddr, CLoyaltyDiscount
    - **Shipment:** SId, SDate, CId
    - **Contains:** PId, SId, SPQty
  - **Relationships:**
    - **Shipment to Customer:** One-to-one (One shipment is placed by one customer).
    - **Contains to Part:** Many-to-one (Many shipments can contain one part).
    - **Contains to Shipment:** Many-to-one (Many parts can be in one shipment).
- **Diagram Symbolism:**
  - Use standard ER diagram symbols like rectangles for entities, ovals for attributes, and diamonds for relationships.

**Section C: Conceptual Model**

**Q10.**

- **ER Diagram:**
  - **Entities:** Book, Member, Loan
  - **Attributes:**
    - **Book:** ISBN, Title, Author, PublicationYear
    - **Member:** MemberID, Name, Address
    - **Loan:** LoanID, BookISBN, MemberID, LoanDate, DueDate
  - **Relationships:**
    - **Loan to Book:** One-to-one (One loan is associated with one book).
    - **Loan to Member:** One-to-one (One loan is taken by one member).
- **CREATE TABLE Statements:**

```
CREATE TABLE Book (
  ISBN VARCHAR(20) PRIMARY KEY,
  Title VARCHAR(100),
  Author VARCHAR(50),
  PublicationYear INTEGER
);

CREATE TABLE Member (
  MemberID INTEGER PRIMARY KEY,
  Name VARCHAR(50),
  Address VARCHAR(100)
);

CREATE TABLE Loan (
  LoanID INTEGER PRIMARY KEY,
  BookISBN VARCHAR(20),
  MemberID INTEGER,
  LoanDate DATE,
  DueDate DATE,
  FOREIGN KEY (BookISBN) REFERENCES Book(ISBN),
  FOREIGN KEY (MemberID) REFERENCES Member(MemberID)
);
```

**Section D: Relational Design Theory**

**Q11.**

- **Invalid Dependency:** `DiseaseName ® DrugName` is not valid because a disease can have multiple drugs prescribed for treatment. This means that knowing the disease name does not uniquely determine the drug name.

**Q12.**

- **Attribute Closure:**
  - `(HospitalName, DiseaseName, SizeofDose)+`
  - Step 1: Start with the initial set of attributes: `(HospitalName, DiseaseName,`

`SizeofDose)`

- Step 2: Apply the functional dependencies:
  - `HospitalName, DiseaseName ® DrugName` (from the FD set)
  - `HospitalName, DrugName ® SizeofDose` (from the FD set)
  - Now the closure includes: `(HospitalName, DiseaseName, SizeofDose, DrugName)`
- Step 3: Apply the dependencies again, but we do not get any new attributes, so the closure is: `(HospitalName, DiseaseName, SizeofDose, DrugName)`

**Q13.**

- **DrugUsage not in 3NF:** The relation DrugUsage is not in 3NF because it contains a transitive dependency. The dependency `HospitalName, DiseaseName ® DrugName` violates the 3NF requirement, as `DrugName` is not a part of any candidate key, and it is functionally dependent on `HospitalName, DiseaseName` which is not a superkey.

**Q14.**

- **Lossless-Join Decomposition:**

  1. **Relation 1:** `HospitalDrug(HospitalName, DrugName)`
     - **PK:** `HospitalName, DrugName`
     - **FDs:** `HospitalName, DiseaseName ® DrugName`
     - **BCNF:** Yes, because the determinant of every FD is a superkey.

  5. **Relation 2:** `DiseaseDrug(DiseaseName, DrugName)`
     - **PK:** `DiseaseName, DrugName`
     - **FDs:** None
     - **BCNF:** Yes, because there are no non-trivial FDs.

  9. **Relation 3:** `HospitalDosage(HospitalName, DiseaseName, SizeofDose)`
     - **PK:** `HospitalName, DiseaseName, SizeofDose`
     - **FDs:** `HospitalName, DrugName ® SizeofDose`
     - **BCNF:** Yes, because the determinant of every FD is a superkey.

- **Lossless-Join Property:** This decomposition has the lossless-join property because every tuple in the original DrugUsage relation can be reconstructed by joining the three decomposed relations using the common attributes (HospitalName, DiseaseName, DrugName, SizeofDose). This is because the join operation will produce all possible combinations based on the shared attributes, and since the FDs were preserved, all the original tuples will be included in the result.

## Section E: Database-backed Applications

**Q15.**

- **SQL Injection Attacks:** These occur when malicious code is injected into data inputs to manipulate database queries. Hackers can gain access to sensitive data, modify data, or even take over a system.

- **Mitigation:**
    - **Input Validation:** Sanitize user inputs by removing or escaping special characters that can be used for malicious code injection.
    - **Prepared Statements:** Use parameterized queries where data is treated as distinct values rather than part of the SQL command itself.
    - **Stored Procedures:** Utilize stored procedures that enforce stricter security checks on database operations.

**Q16.**

- **Hashing Importance:** Hashing is essential for securing user passwords because it converts plain-text passwords into a one-way hash value. This means that the original password cannot be recovered from the hash, preventing attackers from obtaining plain-text passwords.
- **Protection:**
    - **Password Storage:** Hashing allows storing a hashed version of the password instead of the plain text, protecting it from unauthorized access.
    - **Verification:** When a user attempts to log in, the entered password is hashed and compared to the stored hash. If the hashes match, the user is authenticated.

**Q17.**

- **Data Retrieval, Processing, and Display:**
    1. **Data Retrieval:** The web application sends an SQL query to the database, specifying the data to be retrieved.
    2. **Database Processing:** The database executes the query and returns the requested data in a structured format (e.g., JSON or XML).
    3. **Application Processing:** The web application receives the data, processes it as required (e.g., filtering, sorting, calculations), and converts it into a suitable format for display.
    4. **Display:** The application renders the processed data on the user interface (e.g., HTML, JavaScript) for user interaction.
- **Key Components:** Database server, web server, web application, database client library (to interact with the database), user interface (to display information).

**Important Note:** This answer key provides a concise overview of the solutions. It's essential to understand the reasoning behind each answer and explore related concepts to solidify your knowledge. Good luck with your studies!