📝 DATA6211/p/d Database (Introduction) Answer Key

This complete solution addresses all questions from the exam paper, ensuring technical accuracy and adherence to best practices in database design and SQL.

Question 1: Database Management System Function (5 Marks)

Data Dictionary Management

The **Data Dictionary Management** function of a Database Management System (DBMS) involves storing the definitions of the data and its relationships[1]. This metadata (data about data) is stored in the **data dictionary** or **system catalog**[2].

The data dictionary contains crucial information, including:

- The names and data types of every data element (attribute)[3].
- The relationship between the data elements.
- The source of the data.
- Which users can access or modify the data.

Importance of the Function

This function is **critically important** for the following reasons:

1. **Data Integrity and Quality:** It enforces data types, constraints (like primary keys and foreign keys), and validation rules, ensuring the data in the database is accurate and consistent[4].

2. **Centralized Control:** It provides a central source of metadata, allowing the DBMS to manage the internal structure and external views of the data effectively.
3. **Security:** It controls user access and permissions to different parts of the database.
4. **DBMS Operations:** The DBMS uses the dictionary to look up the necessary structures to execute queries, optimize performance, and manage storage[5].

Question 2: Entity Relationship Diagram (ERD) Analysis (5 Marks)

Q.2.1 Notation (1 Mark)

The notation used in the ER diagram is **Crow's Foot notation**.

Q.2.2 Relationship (4 Marks)

- **Relationship Name:** The relationship is between student and seat.
- **Relationship Type and Description:** The relationship is **Many-to-Many ($M:N$)**[6].
    - **Description:** A single student can potentially be assigned to many different seats (over time or in different classes), and a single seat can be occupied by many different students (over time, such as in different class periods). The cardinality shows $1..*$ (one or many) on both sides of the relationship line, which is characteristic of a Many-to-Many relationship[7].

Question 3: Normalization (40 Marks)

Q.3.1 Objective of Normalization (6 Marks)

The objective of the **normalization process** is to design a database schema that is free from data anomalies and redundancies, which are common issues in poorly structured tables (relations)[8].

Key objectives include:

- **Eliminating Data Redundancy:** Storing the same information multiple times is minimized, saving space and improving data consistency[9].

- **Eliminating Data Anomalies:** Normalization removes the problems associated with:
    - **Update Anomalies:** Changing data in one row doesn't require changing the same data in many other rows.
    - **Insertion Anomalies:** Allows for the insertion of data without requiring the existence of other unrelated data.
    - **Deletion Anomalies:** Prevents the loss of important data when a row is deleted.
- **Simplifying Data Structure:** Breaking large tables into smaller, well-structured tables based on dependencies[10].

- **Ensuring Data Dependencies:** Ensuring that all non-key attributes in a table are dependent on the **entire** primary key, the **whole** primary key, and **nothing but** the primary key.

Q.3.2 Second Normal Form (2NF) (2 Marks)

A table is in **Second Normal Form (2NF)** if and only if:

1. It is already in **First Normal Form (1NF)**.
2. All non-key attributes are **fully functionally dependent** on the entire primary key[11]. That is, there are no **partial dependencies**.

Q.3.3 Third Normal Form (3NF) (2 Marks)

A table is in **Third Normal Form (3NF)** if and only if:

1. It is already in **Second Normal Form (2NF)**.
2. There are **no transitive dependencies**[12]. That is, no non-key attribute is dependent on another non-key attribute.

Q.3.4 Normalise to Second Normal Form (2NF) (15 Marks)

The starting table is DATA_ORG_1NF.
Composite Primary Key: $\text{PHOTOGRAPHER ID} + \text{PHOTO ID}$ (as a photo description is unique per PHOTO ID, and a photographer name is unique per PHOTOGRAPHER ID)

1NF Dependency Diagram

$$\text{PHOTOGRAPHER ID}, \text{PHOTO ID}, \text{GENRE ID} \to \text{PHOTOGRAPHER NAME \& SURNAME}, \text{PHOTO DESCRIPTION}, \text{GENRE DESCRIPTION}$$

Step: Identify and Remove Partial Dependencies

A **partial dependency** exists when a non-key attribute is dependent on only *part* of the composite primary key.

1. **Partial Dependency 1:**
   - $\text{PHOTOGRAPHER ID} \to \text{PHOTOGRAPHER NAME \& SURNAME}$ (The photographer's name depends only on their ID, not the photo ID).
2. **Partial Dependency 2:**
   - $\text{PHOTO ID} \to \text{PHOTO DESCRIPTION}$ (The photo's description depends only on its ID, not the photographer ID).
3. **Partial Dependency 3:**
   - GENRE ID is not part of the primary key, but it determines GENRE DESCRIPTION. This is a **transitive dependency**, which is removed in 3NF. However, we must first address the partial dependencies to achieve 2NF.

To achieve 2NF, we decompose the table into three separate tables based on the partial dependencies.

2NF Dependency Diagrams

1. PHOTOGRAPHER Table: (Removes PD 1)

   $$\text{PHOTOGRAPHER ID} \to \text{PHOTOGRAPHER NAME \& SURNAME}$$

   (Primary Key: PHOTOGRAPHER ID)
2. PHOTO Table: (Removes PD 2)

   $$\text{PHOTO ID} \to \text{PHOTO DESCRIPTION}$$

   (Primary Key: PHOTO ID)
3. PHOTO_GENRE_LINK Table (The original table):

   $$\text{PHOTOGRAPHER ID}, \text{PHOTO ID}, \text{GENRE ID} \to \text{GENRE DESCRIPTION}$$

   (Primary Key: $\text{PHOTOGRAPHER ID}, \text{PHOTO ID}$)
   - *Note: GENRE ID is a non-key attribute that determines GENRE DESCRIPTION, which is a transitive dependency we address in 3NF.*

The table is now in 2NF because all non-key attributes ($\text{PHOTOGRAPHER NAME \& SURNAME}$, PHOTO DESCRIPTION, GENRE DESCRIPTION) are fully functionally dependent on their respective primary keys.

Q.3.5 Normalise to Third Normal Form (3NF) (15 Marks)

We start from the 2NF tables:

PHOTO_GENRE_LINK Table (from Q.3.4)

$$\text{PHOTOGRAPHER ID}, \text{PHOTO ID}, \text{GENRE ID} \to \text{GENRE DESCRIPTION}$$

Step: Identify and Remove Transitive Dependencies

A **transitive dependency** exists when a non-key attribute determines another non-key attribute.

- Transitive Dependency:

  $$\text{GENRE ID} \to \text{GENRE DESCRIPTION}$$

  (GENRE ID is a non-key attribute in the PHOTO_GENRE_LINK table, and it determines another non-key attribute, GENRE DESCRIPTION.)

To achieve 3NF, we move the transitive dependency into a new table and leave only the foreign key in the original table.

1. GENRE Table: (Removes the transitive dependency)

   $$\text{GENRE ID} \to \text{GENRE DESCRIPTION}$$

   (Primary Key: GENRE ID)
2. PHOTO_GENRE_LINK Table (The final link table):

   $$\text{PHOTOGRAPHER ID}, \text{PHOTO ID} \to \text{GENRE ID}$$

   (Primary Key: $\text{PHOTOGRAPHER ID}, \text{PHOTO ID}$ / Foreign Keys: PHOTOGRAPHER ID, PHOTO ID)

3NF Dependency Diagrams (Final Answer)

The final 3NF decomposition results in four tables (relations):

1. PHOTOGRAPHER:

   $$\text{PHOTOGRAPHER ID} \to \text{PHOTOGRAPHER NAME \& SURNAME}$$

   (PK: PHOTOGRAPHER ID)
2. PHOTO:

   $$\text{PHOTO ID} \to \text{PHOTO DESCRIPTION}$$

   (PK: PHOTO ID)
3. GENRE:

$$\text{GENRE ID} \to \text{GENRE DESCRIPTION}$$

(PK: GENRE ID)

4. PHOTO_LINK (Junction Table): (Links all three entities)

   $$\text{PHOTOGRAPHER ID}, \text{PHOTO ID}, \text{GENRE ID}$$

   (Composite PK: $\text{PHOTOGRAPHER ID}, \text{PHOTO ID}$) (FKs: PHOTOGRAPHER ID, PHOTO ID, GENRE ID)

Question 4: Entity Relationship Diagram (ERD) (25 Marks)

The ERD will be drawn using **UML Notation** at the **Logical Level** (including Primary and Foreign Keys) and resolve the $M:N$ relationship into an associative entity.

Business Rules & Entities

1. **Owner** owns **Cat** ($1:M$)
2. **Cat** belongs to **Breed** ($M:1$)
3. **Cat** enters **Competition** ($M:N$) $\to$ resolved by **Entry** (Associative Entity)

Logical ERD using UML Notation

The corrected ERD structure is as follows:

Entity	**Primary Key (PK)**	**Other Attributes**	**Foreign Keys (FK)**	**Relationship**	**Multiplicity**
Owner	OwnerID (PK)	Name, Surname	None	Owns	$\mathbf{1..1}$ owns

					$\mathbf{1..*}$ Cats
Cat	CatID (PK)	Name	OwnerID (FK), BreedID (FK)	Is owned by $\mathbf{1..1}$ Owner; Belongs to $\mathbf{1..1}$ Breed	$\mathbf{1..*}$ is owned by $\mathbf{1..1}$ Owner; $\mathbf{1..*}$ belongs to $\mathbf{1..1}$ Breed
Breed	BreedID (PK)	Description	None	Has $\mathbf{1..*}$ Cats	$\mathbf{1..*}$ has $\mathbf{1..1}$ Cat
Competition	CompetitionID (PK)	Description	None	Has $\mathbf{1..*}$ Entries	$\mathbf{1..*}$ has $\mathbf{1..*}$ Entry
Entry (Associative)	CatID (PK, FK), CompetitionID (PK, FK)	*None*	CatID (FK), CompetitionID (FK)	Links $\mathbf{1..1}$ Cat and $\mathbf{1..1}$ Competition	$\mathbf{1..*}$ Cat is in $\mathbf{1..1}$ Entry; $\mathbf{1..*}$ Competition has $\mathbf{1..1}$ Entry

Question 5: SQL Implementation (45 Marks)

Based on the correct logical ERD from Question 4, the correct SQL code is provided below. We assume an OwnerID is required in the Cat table to enforce the $1:M$ relationship between Owner and Cat. Similarly, BreedID is required in Cat. We will assign IDs with an INT data type, assuming they are auto-incrementing or assigned keys.

Q.5.1 Create the Cat Table Structure (10 Marks)

The Cat table must include a primary key (CatID), the Name attribute, and foreign keys to Owner and Breed.

SQL

```sql
CREATE TABLE Cat (
    CatID INT NOT NULL PRIMARY KEY,
    Name VARCHAR(250) NOT NULL,
    OwnerID INT NOT NULL, -- Foreign key to Owner
    BreedID INT NOT NULL, -- Foreign key to Breed
    FOREIGN KEY (OwnerID) REFERENCES Owner(OwnerID),
    FOREIGN KEY (BreedID) REFERENCES Breed(BreedID)
);
```

Q.5.2 Populate Breed Table & Display (10 Marks)

The Breed table needs to be populated with the five specified breeds. We will assume BreedIDs are assigned sequentially starting from 1.

SQL

```sql
-- Assuming Breed table was created as: CREATE TABLE Breed (BreedID INT NOT NULL PRIMARY KEY,
Description VARCHAR(250) NOT NULL);

-- 1. Populate the Breed table
INSERT INTO Breed (BreedID, Description)
VALUES (1, 'Siamese'),
       (2, 'Burmese'),
       (3, 'Sphynx'),
       (4, 'Persian'),
       (5, 'Maine Coon');

-- 2. Display the contents of the table
SELECT
  BreedID,
  Description
FROM
  Breed;
```

Q.5.3 List Cats by Owner (10 Marks)

This requires joining the Cat and Owner tables, filtering by the owner's name, and ordering the results by the cat's name.

- **Assumption:** Owner table exists with columns OwnerID, Name, Surname. We will search by $\text{Name} = 'Thabo'$ and $\text{Surname} = 'Ndlovu'$.

SQL

```sql
SELECT
  C.Name AS Cat_Name
FROM
  Cat C
JOIN
  Owner O ON C.OwnerID = O.OwnerID
WHERE
  O.Name = 'Thabo' AND O.Surname = 'Ndlovu'
ORDER BY
  Cat_Name ASC;
```

Q.5.4 Generate Breed Report (15 Marks)

This report requires joining the Cat and Breed tables, grouping the results by breed description, counting the number of cats in each group, and ordering the output alphabetically by breed description.

SQL

```
SELECT
    B.Description AS BREED,
    COUNT(C.CatID) AS "NUMBER OF CATS"
FROM
    Breed B
JOIN
    Cat C ON B.BreedID = C.BreedID
GROUP BY
    B.Description
ORDER BY
    BREED ASC;
```

Would you like me to review any specific section in more detail, or perhaps generate the complete Owner, Competition, and Entry table creation SQL code as well?