1. Assessment and Errors

The student's performance is highly inconsistent, with major errors in interpretation, motivation, and practical application.

Question	Where the Student Went Wrong	Impact/Severity
Q2.1 (Business Rules)	The student failed to correctly read the **Crow's Foot/UML multiplicities** (cardinality) on the Entity Relationship Diagram (ERD). For example, they stated "one and only one owner owns one and only one pet" and "One pet can have the many owners," which directly contradicts the visual representation and the primary key placement[2].	**Severe.** This demonstrates a fundamental inability to translate a visual database model (ERD) into real-world constraints (business rules), a core skill for a database analyst.
Q2.2.2 (Relationship)	The student misidentified a **Many-to-Many (*..* drives *..*) relationship** as "one to many"[33333]. While their *motivation* correctly described the M:N nature ("one driver can drive many trucks, one truck can be drive by many drivers"), the formal naming was incorrect, indicating a conceptual flaw.	**Moderate.** The student understood the *implication* but not the formal *naming*.
Q3.2 (Relationship Expl.)	Similar to Q2.1, the student provided **incorrect**	**Severe.** Reinforces the inability to correctly read

	business rules for the Doctor-Prescription-Medication-Patient relationship by misreading the 1..1 and 1..* multiplicities (e.g., claiming "A prescription belongs to many doctors" when the diagram showed 1..1 on the Doctor side)[4].	ERD multiplicities.
Q4.2 (Draw ERD)	**No attempt** was made on this 17-mark question[5].	**Severe.** Failing to attempt a high-value question significantly impacts the final mark.
Q5.1 (2NF vs 3NF)	The student's definitions of **Second Normal Form (2NF)** and **Third Normal Form (3NF)** were muddled and conceptually inaccurate. They failed to clearly state the core difference: **2NF resolves Partial Dependencies** and **3NF resolves Transitive Dependencies**[6].	**Severe.** Normalisation is a cornerstone of database design.
Q5.2 (Normal Form ID)	The student incorrectly identified an already decomposed/normalised ERD structure as **unnormalised** (Q5.2.1) and wrongly identified a relation with *both* partial and transitive dependencies as being in **Second Normal Form** (Q5.2.2)[7777].	**Severe.** This confirms a weak grasp of the normalisation process and how to identify violations.

Q5.3/Q5.4 (Normalisation)	**No attempt** was made on the major normalisation questions[8].	**Severe.** Failing to attempt high-value questions significantly impacts the final mark.
Q6.1 (SQL Queries)	**No SQL code was provided** for any of the practical database manipulation questions (Q6.1.1 to Q6.1.6)[9].	**Severe.** SQL is the industry standard for database interaction. Lack of practical application is a critical failure.
Q6.2/Q6.3 (SQL Theory)	The student provided **poor definitions** for **HAVING** (confusing it with checking for data types) and **Index** (stating its purpose is "to list data in categories")[10].	**Moderate.** Shows a theoretical misunderstanding of practical SQL components.

2. Guideline to Do Better

To move from a marginal understanding to a professional level, the student must focus on:

1. **Mastering ERD Notation:** Dedicate time to strictly practice reading **UML/Crow's Foot notation**, especially the **multiplicities** (cardinality). Remember to read the numbers *on the opposite side* of the relationship to understand the constraint.
   - *Example:* For the relationship A (1..1) --- (1..*) B, read the 1..1 to mean: "For every instance of **B**, there is *one and only one* **A**."
2. **Rigor in Normalisation:** You must be able to:
   - Clearly define the three Normal Forms (1NF, 2NF, 3NF).
   - Identify all **Functional Dependencies (FDs)** in a relation.
   - Distinguish between a **Partial Dependency** (resolved by 2NF) and a **Transitive Dependency** (resolved by 3NF).
   - Practice the **decomposition process** (breaking a large relation into smaller, more efficient relations) repeatedly.
3. **SQL Practice:** SQL is a practical language. You must move beyond theory and dedicate time to **writing and executing** CREATE TABLE, INSERT, SELECT (especially with JOIN,

GROUP BY, WHERE, and HAVING), and UPDATE/DELETE statements.
    - **Crucial Distinction: WHERE** filters **rows**; **HAVING** filters **groups** after aggregation.
4. **Key Terminology:** Ensure strict accuracy when defining key concepts (e.g., using "Candidate Key" instead of "secondary key" when the term is requested, and correctly identifying the purpose of an Index).

---

3. Student Mark Calculation

Question	Mark Achieved	Total Marks
Question 1 (Fundamentals)	8	10
Question 2 (ERD & Relationships)	5	10
Question 3 (Keys, Relationships, Data Analysis)	6	10
Question 4 (ERD Theory and Drawing)	3	20
Question 5 (Normalisation)	1	30
Question 6 (SQL)	8	40
TOTAL MARK	**31 / 120**	**120**

The student would receive a final mark of **31/120** (25.8%). This is a failing grade.

---

4. Full Answer Sheet (120/120)

Question 1: Database Fundamentals (10 Marks)

Q.1.1 What is the difference between data and information? (2)

- **Data** refers to raw, unprocessed facts, figures, or observations that lack context or meaning on their own[11].

- **Information** is data that has been processed, organised, and structured to provide context, meaning, and relevance, which can then be used to support decision-making[12].

Q.1.2 List five advantages of implementing a database management system (DBMS) in an organisation. (5)

1. **Improved Data Sharing:** Provides a central repository for controlled access by multiple users and applications[13].

2. **Minimized Data Redundancy:** Reduces data duplication by storing data in a single, consistent location.
3. **Increased Data Integrity:** Enforces rules and constraints to ensure the data is accurate and consistent.
4. **Enhanced Data Security:** Provides sophisticated mechanisms (user roles, permissions) to protect data from unauthorised access.
5. **Faster Data Access and Retrieval:** Utilises indexes and query optimisation techniques for efficient searching and reporting[14].

Q.1.3 Define in your own words the term database access language and provide one example of such a language. (3)
A Database Access Language is a specialised programming language or set of commands designed to communicate with a database management system (DBMS). Its primary function is to define the database structure and allow users to retrieve, insert, update, and delete data15.

- **Example:** Structured Query Language (**SQL**)[16].

Question 2: ERDs and Relationships (10 Marks)

Q.2.1 Write down any four business rules that are represented in the below Entity

Relationship Diagram (ERD). (4)

Based on the ERD (Competition, Entry, Pet, Owner, Species, Breed) [17]:

1. A **Pet** must be owned by one and only one **Owner** (1..1)[18].
2. An **Owner** may own one or more **Pets** (1..*)[19].
3. A **Competition** must have one or more **Entries** (1..*)[20].
4. An **Entry** must be submitted for one or more **Pets** (1..*) [21]and must be part of one and only one **Competition** (1..1)[22].

Q.2.2 Which type of relationship is shown in each case below? Motivate your answer. (6)

Q.2.2.1 Car and Person (2)

- **Relationship Type:** One-to-One (1:1).
- **Motivation:** The multiplicity shows **1..1** on both sides of the owns relationship, meaning one and only one **Person** owns one and only one **Car**, and one and only one **Car** is owned by one and only one **Person**[23].

Q.2.2.2 Driver and Truck (2)

- **Relationship Type:** Many-to-Many (M:N).
- **Motivation:** The multiplicity shows *** (many)** on both sides of the drives relationship[242424242424242424]. This means a **Driver** can drive many **Trucks**, and a **Truck** can be driven by many **Drivers**.

Q.2.2.3 Country and Province tables (2)

- **Relationship Type:** One-to-Many (1:M).
- **Motivation:** This is a one-to-many relationship because the CountryID (the Primary Key in Country [25]) is a **Foreign Key** in the Province table[26]. The data shows one CountryID (e.g., '1' for South Africa [27]) is referenced by multiple rows (provinces) in the Province table (e.g., Gauteng and KwaZulu-Natal [28]).

Question 3: Keys, Relationships, and Data Analysis (10 Marks)

Q.3.1 What is the name of the Foreign Key in the Character table? (1)

- **ClassID** (It links the Character entity to the Class entity)[29].

Q.3.2 Explain the relationship in the below Entity Relationship Diagram (ERD) in your own words. (4)

- The overall structure is a **Many-to-Many (M:N) relationship** between **Doctor** and **Medication** and **Patient**, which is resolved by the **Prescription** entity.
- **Doctor-Prescription:** A **Doctor** must write one or more **Prescriptions** (1..*). A **Prescription** must be written by one and only one **Doctor** (1..1)[30].

- **Medication-Prescription:** A **Medication** can be listed on one or more **Prescriptions** (1..*). A **Prescription** is for one and only one type of **Medication** (1..1)[31].

- **Patient-Prescription:** A **Patient** can receive one or more **Prescriptions** (1..*). A **Prescription** is written for one and only one **Patient** (1..1)[32].

Q.3.3 Answer the following questions by referring to the below sample data (Client table).

Q.3.3.1 Is the column Surname suitable to use as a primary key? Motivate your answer. (2)

- **No**[3333].

- **Motivation:** A Primary Key must uniquely identify every row. The surname **Modise** is repeated for ClientID 1 and ClientID 4[34343434].

Q.3.3.2 Is the column City suitable to use as a candidate key? Motivate your answer. (3)

- **No**[3535].

- **Motivation:** A Candidate Key must uniquely identify every row in the table. The city **Johannesburg** is repeated for ClientID 1 and ClientID 4[36363636].

Question 4: Advanced ERD (20 Marks)

Q.4.1 What is the difference between a Binary and Unary relationship diagram? (3)

- A **Binary Relationship** is the most common type, involving **two** distinct entities in the relationship[37].

- A **Unary Relationship** (or recursive relationship) involves a single entity relating to itself. For example, an Employee supervises another Employee[38].

Q.4.2 Draw an Entity Relationship Diagram (ERD) that represents the business scenario (17)
The business scenario for Q4.2 is not present in the provided documentation. A full-mark answer requires a complete ERD diagram based on a missing scenario.

Question 5: Normalisation (30 Marks)

Q.5.1 What is the difference between second normal form and third normal form? (2)

- **Second Normal Form (2NF):** A relation must be in 1NF **AND** all non-key attributes must be fully functionally dependent on the entire Primary Key. **2NF resolves Partial Dependencies**.
- **Third Normal Form (3NF):** A relation must be in 2NF **AND** no non-key attribute is dependent on another non-key attribute (i.e., no Transitive Dependencies). **3NF resolves Transitive Dependencies**.

Q.5.2 In which normal form is each of the below? Clearly motivate your answer. (9)

Q.5.2.1 Part / Category / Manufacturer ERD (3)

- **Normal Form: Third Normal Form (3NF)** (or higher).
- **Motivation:** This is an already decomposed database structure. Each entity (Part, Category, Manufacturer) has a simple primary key (PartID, CategoryID, ManufacturerID) and attributes that depend only on that key. The use of Foreign Keys (ManufacturerID {FK}, CategoryID {FK}) establishes the necessary relationships without data redundancy, indicating a normalised structure.

Q.5.2.2 Dependency Diagram showing Player/Spaceship (3)

- **Normal Form: First Normal Form (1NF)**.

- **Motivation:** The diagram shows a relation where the Primary Key is the composite of (Player Name, Spaceship Registration)[39]. The presence of **Partial Dependencies** (e.g., Player Name -> Player Date Joined and Spaceship Registration -> Spaceship Name, Spaceship Value) means it is **not in 2NF**. The presence of **Transitive Dependencies** means it is **not in 3NF**[40]. Since it satisfies the 1NF requirement (atomic attributes), it is in 1NF only.

Q.5.2.3 PartNumber Description Manufacturer Category IsRestricted table (3)

- **Normal Form: First Normal Form (1NF)**.
- **Motivation:** This represents a flat file or un-normalised table with repeating attributes that are not functionally dependent on the primary key, or where dependencies are not yet resolved. It meets the requirement of 1NF (atomic columns) but will contain Partial and/or Transitive Dependencies, placing it at the start of the normalisation process.

Q.5.3 Normalise the relation in Q.5.2.2 to Second Normal Form (2NF). (10)

1NF Relation: \$R\$ (<u>Player Name</u>, <u>Spaceship Registration</u>, Player Date Joined, Spaceship Name, Spaceship Value)

Functional Dependencies (FDs):

- (Player Name, Spaceship Registration) \$\rightarrow\$ All other attributes (Primary Key)
- Player Name \$\rightarrow\$ Player Date Joined (**Partial Dependency**)
- Spaceship Registration \$\rightarrow\$ Spaceship Name, Spaceship Value (**Partial Dependency**)

Decomposition to 2NF (Resolve Partial Dependencies):

1. **PLAYER** (<u>Player Name</u>, Player Date Joined)
2. **SPACESHIP** (<u>Spaceship Registration</u>, Spaceship Name, Spaceship Value)
3. **PLAYER_SPACESHIP** (<u>Player Name</u>, <u>Spaceship Registration</u>) (Link table, to preserve the relationship)

Q.5.4 Normalise the relations from Q.5.3 to Third Normal Form (3NF). (9)

- **Relations in 2NF:**
  - **PLAYER** (<u>Player Name</u>, Player Date Joined) - *Already in 3NF*
  - **PLAYER_SPACESHIP** (<u>Player Name</u>, <u>Spaceship Registration</u>) - *Already in 3NF*
  - **SPACESHIP** (<u>Spaceship Registration</u>, Spaceship Name, Spaceship Value) - *Check for Transitive Dependency*
- **Transitive Dependency (TD) Check in SPACESHIP:**
  - Assume a Spaceship Name determines the Spaceship Value (e.g., all 'Big Ships' are valued at '2 000 000').

    ○ **TD:** Spaceship Registration $\rightarrow$ Spaceship Name $\rightarrow$ Spaceship Value.

Decomposition to 3NF (Resolve Transitive Dependency):

1. **PLAYER** (<u>Player Name</u>, Player Date Joined)
2. **SPACESHIP_DETAIL** (<u>Spaceship Registration</u>, Spaceship Name)
3. **SPACESHIP_VALUE_LOOKUP** (<u>Spaceship Name</u>, Spaceship Value)
4. **PLAYER_SPACESHIP** (<u>Player Name</u>, <u>Spaceship Registration</u>)

Question 6: Structured Query Language (SQL) (40 Marks)

Q.6.1.1 Write a SQL statement to create the table President. (5)
Schema based on Q6_2020_DATA6211Ea_C19.sql:

SQL

```
CREATE TABLE President (
  PresidentID INT AUTO_INCREMENT NOT NULL,
  CountryID INT NOT NULL,
  Name VARCHAR(250) NOT NULL,
  Surname VARCHAR(250) NOT NULL,
  Year YEAR NOT NULL,
  PRIMARY KEY (PresidentID),
  FOREIGN KEY (CountryID)
    REFERENCES Country (CountryID)
);
```

Q.6.1.2 Write a SQL statement that will count the number of presidents that were inaugurated after 2009. (4)
Data: Ramaphosa (2018), Zuma (2009), Geingob (2015), Pohamba (2005)

SQL

```
SELECT COUNT(PresidentID)
FROM President
```

```
WHERE Year > 2009;
-- Result: 2 (Ramaphosa and Geingob)
```

Q.6.1.3 Write a SQL statement to insert the below row into table Country. (4)

```
"CountryID","Name","Abbreviation","CallingCode"
"5","Botswana","BW","267"
```

SQL

```
INSERT INTO Country (CountryID, Name, Abbreviation, CallingCode)
VALUES (5, 'Botswana', 'BW', '267');
```

Q.6.1.4 Write a SQL statement to get the list of all the countries from the database, in alphabetical order by country name. (3)

SQL

```
SELECT *
FROM Country
ORDER BY Name ASC;
```

Q.6.1.5 Write a SQL statement to get the list of all the presidents with a surname starting with the letter R. (3)

SQL

```
SELECT *
FROM President
WHERE Surname LIKE 'R%';
```

```
-- Result: Cyril Ramaphosa
```

Q.6.1.6 Write a SQL statement to get the list of all the presidents, showing only the name and surname of the president, and the name of their country. (5)

```SQL
SELECT
  P.Name,
  P.Surname,
  C.Name AS CountryName
FROM
  President P
JOIN
  Country C ON P.CountryID = C.CountryID;
```

Q.6.2 What is the difference between the WHERE and HAVING clauses in SQL statements? (4)

- **WHERE Clause:** Used to filter individual **rows** from the table(s) *before* any grouping or aggregation takes place. It cannot use aggregate functions (like SUM(), COUNT()).
- **HAVING Clause:** Used to filter **groups** of rows that have been created using the GROUP BY clause. It is applied *after* grouping and aggregation and can use aggregate functions[41].

Q.6.3 What is the purpose of an index in a SQL database? (3)
The purpose of an index in a SQL database is to significantly speed up data retrieval operations (SELECT statements), particularly for searching, sorting, and joining. It works by creating a structured lookup table on specified columns, similar to an index in a book. This comes at the cost of requiring more storage space and slowing down data modification operations (INSERT, UPDATE, DELETE).
Q.6.4. What will the result be of each of the below queries? (9)

Q.6.4.1 INSERT INTO Judge (Name) VALUES ('Abe'); (3)

- **Result:** A new row is added to the Judge table. The JudgeID column (Primary Key) will be automatically generated with the next available integer value (e.g., 1, 2, etc.), and the Name column will contain the value 'Abe'[42].

Q.6.4.2 SELECT J.*, COUNT(C.CaseID) FROM Judge J JOIN CourtCase C ON J.JudgeID

= C.JudgeID GROUP BY J.JudgeID HAVING COUNT(C.CaseID) >= 2; (4)

- **Result:** The query returns all columns (*) for every **Judge** who is associated with **two or more** records in the CourtCase table. It also returns the **count** of cases for each of those judges. The results are grouped by JudgeID (i.e., one row per qualifying judge)[43].

Q.6.4.3 DROP TABLE Judge; (2)

- **Result:** The entire Judge table, including its schema (structure) and all data contained within it, is **permanently deleted** from the database[44].