



Question 1: Data Concepts (Theory)

Q.1.1: Differentiate between Data Governance and Data Quality. (4 Marks)

Step	Action	Perfect Answer Component
1 (Governance)	Define Data Governance by focusing on <i>control and policy</i> .	Data Governance is the system of decision rights and accountabilities for ensuring the appropriate behavior (policies, standards, procedures) in the valuation, creation, use, and control of an organization's data assets.
2 (Quality)	Define Data Quality by focusing on the <i>state</i> of the data.	Data Quality refers to the state of data that is fit for its intended use in operations, decision-making, and planning, measured across dimensions like accuracy, completeness, consistency, and timeliness.
3 (Differentiation)	Summarize the difference (System vs. State).	Differentiation: Governance is the framework (the rules) by which quality is achieved and maintained; Quality is the outcome (the state of the data) of an effective

		governance framework.
--	--	-----------------------

Q.1.2: Summarise any three levels through which data quality can be examined. (6 Marks)

Step	Action	Perfect Answer Component
1 (Level 1)	Discuss the smallest level (Field).	Data Field/Attribute Level: Examination focuses on single fields for correctness (e.g., ensuring data type, format, and range constraints like Age > 18 are met).
2 (Level 2)	Discuss the middle level (Record/Row).	Record/Row Level: Examination ensures all data within a single record is consistent and valid (e.g., checking that a customer's City matches their AreaCode).
3 (Level 3)	Discuss the largest level (Table/Relationship).	Relationship/Table Level (Integrity): Examination checks data relationships across tables, primarily through referential integrity (Primary Key/Foreign Key constraints), ensuring all links are valid.



Question 2: Business Rules and Independence

(Theory)

Q.2.1: Discuss "business rules" and give three reasons why they are essential. (10 Marks)

Step	Action	Perfect Answer Component
1 (Definition)	Define Business Rules as <i>policies that guide behavior.</i>	Business rules are precise, unambiguous statements derived from business policies that describe and restrict the various aspects of the business. They define the policies, procedures, or principles that guide and restrict organizational behavior and data usage.
2 (Reason 1)	Focus on Consistency/Integrity.	1. Enforcement of Constraints: They are the foundation for defining all database constraints (PKs, FKs, CHECK constraints), ensuring data integrity and consistency across the entire database.
3 (Reason 2)	Focus on Design/Structure.	2. Guiding Database Design: They determine the necessary entities, attributes, and relationships , acting as the blueprint for translating real-world operational

		requirements into a logical data model.
4 (Reason 3)	Focus on Communication/Documentation .	3. Facilitating Communication: They serve as crucial documentation that clarifies business policies for all stakeholders (users, analysts, and developers), ensuring a shared understanding of the data's meaning and use.

Q.2.2: Provide two business rules examples: One-to-one (1:1) and One-to-many (1:M). (5 Marks)

Step	Action	Perfect Answer Component
1 (1:1 Rule)	Select two entities where one instance of A relates to <i>at most one</i> instance of B.	One-to-One (1:1): "One employee is assigned to one, and only one, company laptop. That laptop can only be assigned to one, and only one, employee."
2 (1:M Rule)	Select two entities where one instance of A relates to <i>many</i> instances of B.	One-to-Many (1:M): "A single department can employ many employees. However, each employee can only be assigned to one department."

Q.2.3: Briefly discuss the importance of software independence and its benefits. (5 Marks)

Step	Action	Perfect Answer Component
1 (Discussion)	Define Software Independence (or Program-Data Independence).	Software Independence is the ability to change the database's physical or logical structure (the data) without forcing changes to the applications (the software) that access the data, and vice-versa.
2 (Benefits)	Discuss its key benefits.	Importance/Benefits: It is crucial because it promotes flexibility and reduced maintenance cost . Changes to the database or application can occur independently, extending the life of both components and reducing the time and effort required for updates.



Question 3: Data Modelling (ERD)

Q.3.1: Discuss why data models are important in database modelling. (10 Marks)

Step	Action	Perfect Answer Component
1 (Definition)	Define a Data Model as a blueprint.	Data models are blueprints that graphically represent the structure of data, including entities, attributes, and relationships.
2 (Reasons)	List and explain key reasons (Communication, Structure, Policy).	<p>1. Facilitate Communication: Provides a clear, standardized map of data that allows developers, analysts, and end-users to agree on the business data requirements.</p> <p>2. Guide Implementation: Serves as the precise plan for creating the physical database structure (tables, keys, indexes) in the DBMS.</p> <p>3. Enforce Business Policy: Ensures all business rules and constraints are logically and consistently captured before any code is written, reducing errors.</p>

Q.3.2: Create an Entity Relationship Diagram (ERD) using UML notation... (30 Marks)

Step	Action	Perfect Result
1 (Entities)	Identify core entities from the business rules.	Entities (5): Student, Subject, Lecturer, Parent, and Registration (as the linking entity). (Do not include an unnecessary Course entity).
2 (Attributes & Keys)	Add necessary attributes and define Primary Keys (PKs) .	Use attributes from Q.3.1 (e.g., StudentNumber (PK), SubjectCode (PK), LecturerCode (PK)). Registration uses RegistrationNumber as its PK.
3 (M:N Resolution)	Resolve the M:N between Student and Subject using Registration .	Link Student to Registration (1:M) and Subject to Registration (1:M). Add StudentNumber (FK) and SubjectCode (FK) to the Registration entity.
4 (1:M Relationships)	Define the two 1:M relationships (Lecturer and Parent).	Lecturer to Subject (1:M): One lecturer lectures many subjects. Add LecturerCode (FK) to the Subject entity. Parent to Registration (1:M): One parent pays for many registrations. Add ParentID (FK) to the Registration entity.
5 (Cardinality)	Ensure mandatory participation is correctly shown.	Use 1..* (One-to-Many, mandatory) for all relationships based on the rules (e.g., a Subject must

		be lectured by at least one Lecturer).
--	--	--

Question 4: Normalization

Use the provided 1NF relation: (<u>COURSE_CODE, LECTURER_ID, SUBJECT_CODE, SEMESTER_DATE</u>, COURSE_NAME, LECTURER_NAME, SUBJECT_NAME, HOURS).

Q.4.1: Normalise to 2NF (10 Marks)

Step	Action	Perfect Result
1 (Identify Dependencies)	Find non-key attributes that depend on only a part of the composite key (Partial Dependency).	Partial Dependencies: \$COURSE_CODE \to COURSE_NAME\$; \$LECTURER_ID \to LECTURER_NAME\$; \$SUBJECT_CODE \to SUBJECT_NAME, HOURS\$.
2 (Decompose)	Create new tables for each partial dependency, moving the non-key attributes out of the main table.	1. COURSE (<u>COURSE_CODE</u>,&br/>COURSE_NAME) 2. LECTURER (<u>LECTURER_ID</u>,&br/>LECTURER_NAME) 3. SUBJECT (<u>SUBJECT_CODE</u>,&br/>SUBJECT_NAME, HOURS) 4. COURSE_OFFERING (<u>COURSE_CODE,&br/>LECTURER_ID,&br/>SUBJECT_CODE,&br/>SEMESTER_DATE</u>)

		\$\rightarrow\$ (The main remaining table/original key)
--	--	---

Q.4.2: Normalise to 3NF (15 Marks)

Step	Action	Perfect Result
1 (Identify Dependencies)	Check the 2NF tables for Transitive Dependencies (where a non-key attribute depends on another non-key attribute).	Check: In the resulting 2NF tables, there are NO transitive dependencies. All non-key attributes depend only on the Primary Key.
2 (Final 3NF)	State the final 3NF tables (since 2NF = 3NF in this case).	Resulting 3NF Tables: 1. COURSE (<u>COURSE_CODE</u>, COURSE_NAME) 2. LECTURER (<u>LECTURER_ID</u>, LECTURER_NAME) 3. SUBJECT (<u>SUBJECT_CODE</u>, SUBJECT_NAME, HOURS) 4. COURSE_OFFERING (<u>COURSE_CODE, LECTURER_ID, SUBJECT_CODE, SEMESTER_DATE</u>)



Question 5: SQL Queries

Assume the Student table PK is StudentNumber (or Student#) and the Subject table PK is

Subject (or SubjectCode).

Q.	Task	A+ SQL Code	Explanation
Q.5.1	Create Results table with FKs.	CREATE TABLE Results (Student# VARCHAR(20) NOT NULL, Subject VARCHAR(50) NOT NULL, Grade_Status VARCHAR(20), PRIMARY KEY (Student#, Subject), FOREIGN KEY (Student#) REFERENCES Student (Student#), FOREIGN KEY (Subject) REFERENCES Subject (Subject));	Defines a composite PK and uses FOREIGN KEYS for referential integrity.
Q.5.2	Show all columns, sort by Student# descending.	SELECT * FROM Results ORDER BY Student# DESC;	Uses * for all columns and the crucial DESC keyword.
Q.5.3	Show subjects costing between 2300 and 2500.	SELECT Subject, Subject_Cost FROM Subject WHERE Subject_Cost BETWEEN 2300 AND 2500;	Uses the efficient and inclusive BETWEEN operator.
Q.5.4	Add an Address column to Student.	ALTER TABLE Student ADD COLUMN Address VARCHAR(100);	Uses the ALTER TABLE ADD COLUMN command. Best to avoid NOT NULL

			unless a default is specified.
Q.5.5	Count all subjects.	SELECT COUNT(*) AS TotalSubjects FROM Subject;	Uses the COUNT(*) aggregate function and an alias for clarity.
Q.5.6	Select student details with emails ending in @rosebank.co.za.	SELECT Student#, First_Name, Last_Name, Age FROM Student WHERE Email_Address LIKE '%@rosebank.co.za';	Uses the LIKE operator and the % wildcard for pattern matching.
Q.5.7	Insert a new record into Results.	INSERT INTO Results (Student#, Subject, Grade_Status) VALUES ('201011224', 'Accounting', 'Pass');	Uses the INSERT INTO command, explicitly listing columns and values.
Q.5.8	Display the youngest student's details.	SELECT Student#, First_Name, Last_Name, Age FROM Student ORDER BY Age ASC LIMIT 1;	Dynamically finds the minimum age using ORDER BY Age ASC and LIMIT 1 .
Q.5.9	Determine the average subject cost.	SELECT AVG(Subject_Cost) AS AverageCost FROM Subject;	Uses the AVG() aggregate function on the correct column (Subject_Cost).