DUMPSARENA

WGU Data Management – Foundations Exam

WGU Data-Management-Foundations

Version Demo

Total Demo Questions: 7

Total Premium Questions: 73

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

QUESTION NO: 1

Which phase of entity-relationship modeling refers to the maxima and minima of relationships and attributes?

A. Cardinality

B. Physical design

C. Attribute minimum

D. Partition

ANSWER: A

Explanation:

Cardinality is the ER-modeling concept that specifies the allowed minimum and maximum number of entity occurrences that can participate in a relationship. These limits communicate both whether participation is optional or mandatory and how many related instances are permitted. For example, a customer may be associated with zero orders before making a purchase, while an order must be associated with one customer. This can be represented with minimum/maximum notation such as (0, many) for the customer-to-order side and (1, 1) for the order-to-customer side.

Cardinality therefore captures business rules in the logical data model before database tables are implemented. It helps modelers ensure that the structure reflects real-world constraints, such as whether a relationship is one-to-one, one-to-many, or many-to-many. In ER notation, the maximum is commonly shown as one or many, while the minimum is commonly zero or one. Accurate cardinality supports appropriate foreign-key design and data-integrity rules when the model is later translated into a relational schema. See Oracle's discussion of data modeling and entity relationships and IBM's overview of entity-relationship diagrams.

QUESTION NO: 2

What does the aggregate function do?

A. It computes values over a set of rows.

B. It selects rows that appear in one table but not another.

C. It eliminates one or more columns of a table.

D. It lists combinations of rows in two tables.

ANSWER: A

Explanation:

It computes values over a set of rows. In SQL, an aggregate function takes values from multiple rows and produces a summarized result, such as a total, average, count, minimum, or maximum. For example, `COUNT(*)` returns the number of rows selected, `SUM(amount)` returns the total of the `amount` values, and `AVG(score)` returns the mean score. This is useful when a database query needs a concise summary of a larger collection of records rather than each individual record.

Aggregate functions can summarize all rows returned by a query or summarize separate groups when used with a `GROUP BY` clause. For instance, a query can calculate total sales for the entire company, or it can calculate a separate total for each sales region. The defining behavior is that the function performs a calculation across a set of input rows and returns a result for that set or group. PostgreSQL's documentation describes aggregate functions as functions that compute a single result from a set of input values, and its tutorial demonstrates common functions such as `COUNT`, `SUM`, `MAX`, `MIN`, and `AVG`. See PostgreSQL Aggregate Functions and PostgreSQL Aggregate Functions Tutorial.

QUESTION NO: 3

Which statement uses valid syntax for the DELETE statement in SQL?

A. DELETE table_name WHERE condition;

B. DELETE FROM table_name WHERE condition;

C. DELETE FROM table_name;

D. DELETE * FROM table_name WHERE condition;

ANSWER: B

Explanation:

`DELETE FROM table_name WHERE condition;` uses the standard DELETE structure for removing selected rows from a table. The `DELETE FROM` clause identifies the target table, while the `WHERE` clause supplies a predicate that determines exactly which rows are removed. For example, a condition such as `WHERE customer_id = 42` limits the operation to records whose customer identifier is 42. This is the form generally used when a database task requires deletion of specific data rather than an unrestricted operation, and it is the essential pattern to recognize in SQL syntax questions: `DELETE FROM`, followed by a table name, followed optionally by a row-selection condition. The SQL Server DELETE reference documents this syntax as `DELETE [ FROM ] { table_alias | <object> }` with an optional `WHERE` clause, and PostgreSQL likewise documents `DELETE FROM [ ONLY ] table_name [ WHERE condition ]`. These references show that the statement correctly combines the target-table clause and conditional filter. See Microsoft Learn: DELETE (Transact-SQL) and PostgreSQL documentation: DELETE.

QUESTION NO: 4

A database must prevent an order from referencing a customer that does not exist. Which constraint should be defined on Orders.CustomerID?

A. A PRIMARY KEY constraint

B. A FOREIGN KEY constraint

C. A UNIQUE constraint

D. A CHECK constraint

ANSWER: B

Explanation:

A foreign-key constraint referencing the Customers table enforces referential integrity. When a non-null value is inserted into `Orders.CustomerID`, the database verifies that the corresponding referenced customer key exists.

A primary key would identify order rows rather than validate the related customer. A UNIQUE constraint would prevent repeated customer identifiers in Orders, which would incorrectly limit a customer to one order. A CHECK constraint can validate a local condition but does not ordinarily enforce existence of a row in another table.

QUESTION NO: 5

An order can contain many products, and a product can appear on many orders. The database must also record the quantity ordered for each product on each order. Which table design best implements this relationship?

A. Create an OrderItems table containing OrderID, ProductID, and Quantity.

B. Add ProductID and Quantity columns to the Orders table.

C. Add OrderID and Quantity columns to the Products table.

D. Store all ProductID values for an order in one text column in Orders.

ANSWER: A

Explanation:

An OrderItems table with foreign keys to both Orders and Products correctly implements the many-to-many relationship. Each row represents one product on one order, and `Quantity` is an attribute of that relationship because its value depends on the particular order-product combination.

Placing ProductID directly in Orders would allow only one product per order. Placing OrderID in Products would incorrectly make a product belong to one order. Storing a comma-separated product list violates relational design because a column should store atomic values.

QUESTION NO: 6

What is shown on the "many" side of a relationship between two tables?

A. Reflexive relationship

B. Binary relationship

C. Weak entity

D. Foreign key

ANSWER: D

Explanation:

Foreign key is correct because, in a one-to-many relationship, the table representing the many occurrences stores a foreign-key column that references the primary key (or another candidate key) in the table representing the one occurrence. This column records which parent row each child row belongs to. For example, a department can be associated with many employees, so the employee table contains a department identifier as a foreign key that points to the department table's primary key. The foreign key enforces referential integrity by ensuring that a child-table value corresponds to an existing parent-table key, subject to the database system's configured handling of null values and update or delete actions. This placement supports efficient joins and accurately represents the business rule that multiple child records may be related to a single parent record. Microsoft's documentation describes a foreign key as a column or combination of columns whose values correspond to a primary or unique key in another table, establishing the relationship between the tables. See Microsoft Learn: Create foreign key relationships and IBM Db2: Foreign key constraints.

QUESTION NO: 7

Which object relates an entity to itself in an entity-relationship model?

A. Repository

B. Relationship instance

C. Reflexive relationship

D. Attribute

ANSWER: C

Explanation:

Reflexive relationship is correct because it models an association between occurrences of the same entity type. Also called a recursive or self-referencing relationship, it is used when one instance of an entity can be meaningfully related to another instance of that same entity. For example, an Employee entity can have a relationship in which one employee supervises another employee. Both participants are employees, but they have distinct roles in the relationship, such as supervisor and subordinate. Similarly, a Part entity may contain other parts, or a Person entity may be married to another person. In an ER

diagram, this relationship is drawn as a relationship line that begins and ends at the same entity, with role names and cardinalities used to clarify how the entity instances participate. When implemented in a relational database, a reflexive relationship is commonly represented by a foreign key in a table that references the primary key of that same table. For example, an Employee table may include a ManagerID column that refers to EmployeeID. Microsoft's documentation illustrates this pattern as a self-referencing relationship: Self-referencing relationships. Additional ER-modeling background is available from Lucidchart's ER diagram guide.