Oracle SQL Case Study Online Store Inventory)

Problem Statement: Online Store Inventory

Background

An emerging e-commerce business, "TechTrend Innovations," operates an online store selling electronics and accessories, such as laptops, headphones, and chargers. To streamline operations, the company requires a database to manage its product inventory, customer information, and order processing. The database should ensure data integrity, enforce business rules, and support basic reporting to track sales and inventory levels. As a database developer, your task is to design and implement a Oracle database that meets these needs while incorporating fundamental database concepts, tables, columns, and constraints.

Objectives

The goal is to create a robust database system that:

1. Stores information about products, customers, orders, and order details.
2. Enforces business rules using appropriate constraints (e.g., primary key, foreign key, unique, not null, check, default).
3. Supports basic queries to retrieve and update data, such as listing low-stock products, calculating customer spending, and adjusting inventory after orders.
4. Provides a foundation for beginner-to-intermediate SQL learners to practice database creation, data manipulation, and querying.

Requirements

1. Database Structure

- **Tables**:
    - **Products**: Stores details about items sold in the store.
    - **Customers**: Stores customer information for order processing.
    - **Orders**: Tracks customer orders with order date and total amount.
    - **OrderDetails**: Links orders to products, specifying quantities purchased.
- Each table must include appropriate columns with correct data types and constraints to enforce business rules.

2. Business Rules

The database must enforce the following rules using Oracle constraints:

- **Products**:
    - Each product must have a unique identifier.
    - Product names and categories are mandatory.

  - Prices must be positive values (greater than 0).
  - Stock quantities default to 0 if not specified.
- **Customers**:
  - Each customer must have a unique identifier.
  - First and last names are mandatory.
  - Email addresses must be unique to prevent duplicate customer accounts.
  - Phone numbers are optional.
- **Orders**:
  - Each order must have a unique identifier.
  - Orders must be linked to a valid customer.
  - Order dates default to the current date if not provided.
  - Total order amounts must be non-negative.
- **OrderDetails**:
  - Each order detail record must have a unique identifier.
  - Must link to a valid order and product.
  - Quantities ordered must be positive (greater than 0).

3. Functional Requirements

- **Data Insertion**: Populate the database with sample data for at least:
  - 2 products (e.g., a laptop and headphones).
  - 2 customers with contact details.
  - 2 orders, each associated with a customer.
  - Order details linking products to orders with quantities.
- **Queries**:
  - Retrieve products with low stock (e.g., less than 20 units).
  - Calculate the total amount spent by each customer.
  - Update product stock quantities after orders are placed to reflect purchased items.
- **Constraints**:
  - Use **primary keys** to uniquely identify records in each table.
  - Use **foreign keys** to enforce relationships between tables (e.g., Orders to Customers, OrderDetails to Orders and Products).
  - Apply **not null** constraints for mandatory fields.

  - Use **unique** constraints to prevent duplicate emails and other unique fields.
  - Implement **check** constraints to ensure valid data (e.g., positive prices and quantities).
  - Use **default** constraints to set default values where applicable.

4. Technical Requirements

- **Platform**: Oracle (any recent version, e.g., Oracle 19c, Later).
- **SQL Features**:
  - Use basic DDL (Data Definition Language) commands like CREATE DATABASE, CREATE TABLE.
  - Use DML (Data Manipulation Language) commands like INSERT, SELECT, UPDATE.
  - Use aggregation (SUM, GROUP BY) for reporting.
- **Constraints**:
  - Primary Key: Ensure unique identification of records.
  - Foreign Key: Maintain referential integrity.
  - Not Null: Enforce mandatory fields.
  - Unique: Prevent duplicate entries in specific columns.
  - Check: Restrict data to valid ranges or values.
  - Default: Provide fallback values for optional fields.

5. Deliverables

- SQL scripts to:
  - Define tables (Products, Customers, Orders, OrderDetails) with all specified columns and constraints.
  - Insert sample data to test the database.
  - Execute queries to demonstrate functionality (e.g., low-stock report, customer spending, stock updates).