

L5 - DATABASES AND APPLICATION DEVELOPMENT (DB - DIVYA)

ACCA5026

AdventureWorks
Cycles' Database
Design
& Implementation

Contents

Introduction.....	2
Entity Relationship Diagram (ERD)	2
Justification for Design Choices	3
Database Design and Implementation	3
SQL Code for Table Structures	3
SQL Code for Referential Integrity Constraints	5
SQL Code for Check Constraints.....	6
SQL Code for Populating Tables	6
Testing Constraints	8
Transactional Scenario: Rollback and Commit Demonstration	9
Performance Tuning Enhancements	12
Screenshots of Completed Work.....	14
Conclusion	19
References	4

Introduction

AdventureWorks Cycles, a prominent bicycle manufacturer, aims to modernize its operations by developing a robust online marketplace. This requires a well-structured database solution that supports product management, customer orders, manufacturer registration, and customer feedback.

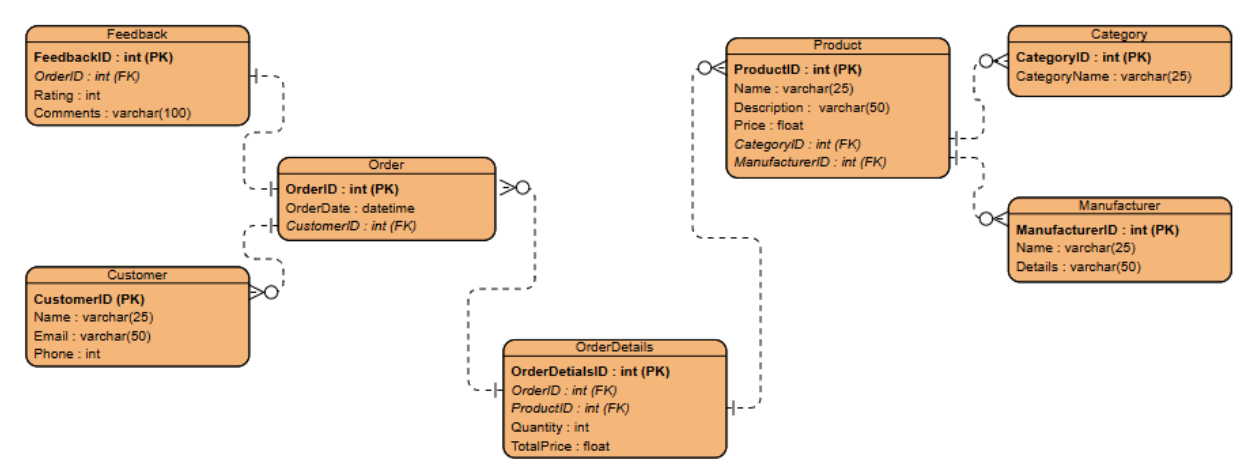
This report outlines the proposed database design and implementation, focusing on:

- An Entity Relationship Diagram (ERD)
- SQL statements for database creation and integrity enforcement
- Transactions to demonstrate rollback and commit
- Performance tuning through indexing

The goal is to ensure AdventureWorks Cycles has a scalable and efficient database system that supports their online platform.

Entity Relationship Diagram (ERD)

- **Manufacturer:** Stores details about manufacturers.
- **Product:** Stores details about products and associates products with manufacturers and categories.
- **Category:** Classifies products into categories.
- **Customer:** Stores customer details.
- **OrderData:** Represents a purchase transaction.
- **OrderDetails:** Links orders and products for many-to-many relationships.
- **Feedback:** Stores customer feedback.



Entity 1	Entity 2	Cardinality	Description
----------	----------	-------------	-------------

Manufacturer	Product	One-to-Many	A manufacturer can produce many products.
Category	Product	One-to-Many	A category can include many products.
Customer	OrderData	One-to-Many	A customer can place multiple orders.
Order	OrderDetails	One-to-Many	An order contains multiple products.
Product	OrderDetails	One-to-Many	A product can appear in multiple orders.
OrderData	Feedback	One-to-One	An order can have one feedback entry.

Justification for Design Choices

The database schema is normalized to Third Normal Form (3NF) to eliminate any redundancy and maintain data integrity [1] for AdventureWorks. For example, product information is stored in the Product table, and its manufacturer is referenced via a foreign key to the Manufacturer table.

Each table has a Primary Key to uniquely identify rows for example FeedBackID, CustomerID, ProductID and so on. It also includes foreign keys that are used to link related tables, ensuring referential integrity. I have also used CHECK constraints, such as ensuring product prices are positive and ratings are between 1 and 5 to reduce any errors when entering data. The design also supports many-to-many relationships using bridge tables like OrderDetails which allows multiple products in an order.

Database Design and Implementation

SQL Code for Table Structures

-- Table Structure

-- Manufacturer

```
CREATE TABLE Manufacturer (
    ManufacturerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactEmail VARCHAR(100) UNIQUE,
    Country VARCHAR(50)
);
```

-- Category

```
CREATE TABLE Category (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(50) NOT NULL UNIQUE
);
```

```

-- Customer
CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Address VARCHAR(255)
);

-- Product
CREATE TABLE Product (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Description VARCHAR(255),
    Price DECIMAL(10, 2) CHECK (Price > 0),
    ManufacturerID INT,
    CategoryID INT,
    FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID), -- Integrity
Constraint
    FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID), -- Integrity Constraint
    CHECK (Price > 0) -- Check Constraint
);

-- OrderData
CREATE TABLE OrderData (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    OrderDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    CustomerID INT,
    TotalAmount DECIMAL(10,2) CHECK (TotalAmount > 0),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), -- Integrity
Constraint
    CHECK (TotalAmount > 0) -- Check Constraint
);

-- OrderDetails
CREATE TABLE OrderDetails (
    OrderDetailID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    FOREIGN KEY (OrderID) REFERENCES OrderData(OrderID), -- Integrity Constraint
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID), -- Integrity Constraint
    CHECK (Quantity > 0) -- Check Constraint
);

```

```
-- Feedback
CREATE TABLE Feedback (
    FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    ProductID INT,
    Review VARCHAR(255),
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), -- Integrity
Constraint
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID), -- Integrity Constraint
    CHECK (Rating BETWEEN 1 AND 5) -- Check Constraint
);
```

SQL Code for Referential Integrity Constraints

The referential integrity constraints in this schema are the FOREIGN KEY relationships that ensure data consistency between related tables [2].

ProductTable:

ManufacturerID is a foreign key referencing Manufacturer (ManufacturerID). This ensures that every product must have a valid manufacturer listed in the Manufacturer table.

CategoryID is a foreign key referencing Category (CategoryID). This ensures that every product belongs to a valid category in the Category table.

OrderData Table:

CustomerID is a foreign key referencing Customer (CustomerID). This ensures that every order is connected with a valid customer in the Customer table.

OrderDetails Table:

OrderID is a foreign key referencing OrderData(OrderID). This ensures that every order detail belongs to a valid order in the OrderData table.

ProductID is a foreign key referencing Product (ProductID). This ensures that every order detail refers to a valid product in the Product table.

Feedback Table:

CustomerID is a foreign key referencing Customer (CustomerID). This ensures that every feedback entry is connected with a valid customer in the Customer table.

ProductID is a foreign key referencing Product (ProductID). This ensures that every feedback entry refers to a valid product in the Product table.

These constraints maintain the integrity of the relationships between tables by preventing actions like inserting invalid foreign keys, updating referenced primary keys without cascading changes, or deleting referenced rows without addressing dependencies.

SQL Code for Check Constraints

-- Check Constraints

-- Manufacturer Table

ContactEmail VARCHAR(100) UNIQUE, -- Makes sure that contact email is unique

-- Product Table

CHECK (Price > 0) -- Makes sure that the price of a product is positive.

-- OrderData Table

CHECK (TotalAmount > 0) -- Makes sure that the total amount for an order is positive.

-- OrderDetails Table

CHECK (Quantity > 0) -- Makes sure that the quantity in an order is positive.

-- Feedback Table

CHECK (Rating BETWEEN 1 AND 5) -- Makes sure that the rating is within the range of 1 to 5.

SQL Code for Populating Tables

-- Sample Data

-- Manufacturer Records

```
INSERT INTO Manufacturer (Name, ContactEmail, Country) VALUES
('Bike Co', 'contact@bikeco.com', 'USA'),
('Trail Masters', 'info@trailmasters.com', 'Canada'),
('Urban Cycles', 'support@urbancycles.com', 'UK'),
('Cyclone Parts', 'sales@cycloneparts.com', 'Germany'),
('Velotech', 'contact@velotech.com', 'France');
```

-- Category Records

```
INSERT INTO Category (CategoryName) VALUES
('Mountain Bike'),
('Road Bike'),
('Accessories'),
('Electric Bike'),
('Helmets');
```

-- Product Records

```
INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID) VALUES
('TrailBlazer MTB', 'Mountain bike for rough terrains', 599.99, 1, 1),
('Speedster Road Bike', 'Lightweight road bike', 799.99, 2, 2),
('Xtreme E-Bike', 'Electric bike with powerful motor', 1299.99, 4, 4),
('Helix Helmet', 'High-quality cycling helmet', 49.99, 5, 5),
('Gear Up Accessories', 'Bike repair tools and gear', 29.99, 3, 3);
```

-- Customer Records

```
INSERT INTO Customer (FullName, Email, Address) VALUES
('John Doe', 'john@example.com', '123 Main St, USA'),
('Jane Smith', 'jane@example.com', '456 Elm St, Canada'),
('Mike Johnson', 'mike@example.com', '789 Pine St, UK'),
('Sara Lee', 'sara@example.com', '101 Maple Ave, Germany'),
('Tom Hardy', 'tom@example.com', '202 Oak Dr, France');
```

-- OrderData Records

```
INSERT INTO OrderData (CustomerID, TotalAmount) VALUES
(1, 599.99),
(2, 799.99),
(3, 1299.99),
(4, 49.99),
(5, 29.99);
```

-- OrderDetails Records

```
INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES
(1, 1, 1),
(2, 2, 1),
(3, 3, 1),
```

```
(4, 4, 1),  
(5, 5, 1);
```

-- Feedback Records

```
INSERT INTO Feedback (CustomerID, ProductID, Review, Rating) VALUES  
(1, 1, 'Excellent bike!', 5),  
(2, 2, 'Smooth and fast ride', 4),  
(3, 3, 'Incredible e-bike experience!', 5),  
(4, 4, 'Great helmet, comfortable and safe.', 4),  
(5, 5, 'Useful gear set for bike maintenance.', 5);
```

Testing Constraints

-- Testing Constraints

-- Test Check Constraint: Price > 0 (Product Table)

-- Attempt to insert a product with a negative price

```
INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID)  
VALUES ('Test Product', 'Negative Price Test', -10, NULL, NULL);
```

The goal of this test is to add a product to the Product table that has a negative price. This insertion will fail and throw an error, protecting the integrity of pricing data because the CHECK constraint requires that the Price be greater than 0.

-- Test Check Constraint: TotalAmount > 0 (OrderData Table)

-- Attempt to insert an order with a negative total amount

```
INSERT INTO OrderData (CustomerID, TotalAmount)  
VALUES (NULL, -50);
```

This test tries to insert an order with a negative TotalAmount into the OrderData table. The database should reject this invalid record since the CHECK constraint guarantees that TotalAmount is always a positive value.

-- Test Check Constraint: Quantity > 0 (OrderDetails Table)

-- Attempt to insert an order detail with a zero quantity

```
INSERT INTO OrderDetails (OrderID, ProductID, Quantity)  
VALUES (NULL, NULL, 0);
```

This test aims to insert an order detail with a Quantity of zero. The CHECK constraint on the OrderDetails table requires Quantity to be greater than zero, so this operation will fail, safeguarding the validity of order quantities.

```
-- Test Check Constraint: Rating BETWEEN 1 AND 5 (Feedback Table)
-- Attempt to insert feedback with an invalid rating
INSERT INTO Feedback (CustomerID, ProductID, Review, Rating)
VALUES (NULL, NULL, 'Invalid Rating Test', 6);
```

This test attempts to insert feedback with a Rating value of 6, which is outside the valid range of 1 to 5 as enforced by the CHECK constraint. The insertion will be rejected, ensuring that ratings are always within the specified range.

```
-- Test Unique Constraint: ContactEmail in Manufacturer Table
-- Insert two manufacturers with the same email
INSERT INTO Manufacturer (Name, ContactEmail, Country)
VALUES ('Manufacturer 1', 'duplicate@example.com', 'USA');
```

```
INSERT INTO Manufacturer (Name, ContactEmail, Country)
VALUES ('Manufacturer 2', 'duplicate@example.com', 'USA');
```

The goal of this test is to add two manufacturers to the Manufacturer table that have the same ContactEmail. The second insertion will fail because ContactEmail is a UNIQUE column, keeping the manufacturer's contact information's uniqueness.

Rollback and Commit

Using Rollback and Commit in SQL means that your database operations are reliable, consistent, and secure. These commands are essential for transaction management, as they allow you to execute a series of operations as a single logical unit [3]. Here are some examples used for AdventureWorks schema

```
-- Transaction 1
```

```
START TRANSACTION;
```

```
-- Insert a new product
```

```
INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID)
VALUES ('New Product', 'A brand-new product description', 49.99, 1, 1);
```

-- Add feedback for the product by a customer

```
INSERT INTO Feedback (CustomerID, ProductID, Review, Rating)
```

```
VALUES (1, LAST_INSERT_ID(), 'Great product!', 5);
```

-- Commit the transaction if both operations are correct

```
COMMIT;
```

-- Rollback if any issue happen during the transaction

```
ROLLBACK;
```

This transaction aims to immediately record customer feedback for a new product and add it to the database. A new product with the following details is first added to the Product table: Name, Description, Price, ManufacturerID, and CategoryID. Once the product has been successfully added, the ProductID of the newly created product is retrieved using the LAST_INSERT_ID() function. Once the feedback has been linked to the appropriate product, it is inserted into the Feedback table using this ProductID. The transaction is committed, making the modifications permanent, if both operations are successfully finished. The ROLLBACK feature, on the other hand, makes sure that no partial changes are saved in the event that an issue occurs during the insertion (such as constraint violations or invalid data), preserving database integrity.

-- Transaction 2

```
START TRANSACTION;
```

-- Insert a new order

```
INSERT INTO OrderData (OrderDate, CustomerID, TotalAmount)
```

```
VALUES (NOW(), 1, 200.00);
```

-- Check the output or error manually

-- Set OrderID to latest CustomerID

SET @OrderID = LAST_INSERT_ID();

-- Add the order to order details

INSERT INTO OrderDetails (OrderID, ProductID, Quantity)

VALUES (@OrderID, 1, 2);

-- Commit the transaction if both operations are correct

COMMIT;

-- Rollback if any issue happen during the transaction

ROLLBACK;

This transaction manages the process of documenting a customer's new order, including the specifics of the goods they bought. First, a new order is entered into the OrderData table, which records the order's OrderDate, CustomerID, and TotalAmount. The OrderID of the newly created order is retrieved by the LAST_INSERT_ID() function after the order has been added. In order to link the order to particular products and specify their quantities, this OrderID is then used to insert the necessary information into the OrderDetails table. The transaction is committed to save the data if the order creation and order details insertion processes are successful. A ROLLBACK is carried out to undo all modifications in the event that an error arises during the procedure (for example, a problem with foreign key constraints or invalid values).

-- Transaction 3

START TRANSACTION;

-- Add a new manufacturer

INSERT INTO Manufacturer (Name, ContactEmail, Country)

VALUES ('Bike Inc', 'info@bikeinc.com', 'USA');

-- Assign products to the new manufacturer

UPDATE Product

SET ManufacturerID = LAST_INSERT_ID()

WHERE ProductID IN (1, 2);

-- Commit the transaction if both operations are correct

COMMIT;

-- Rollback if any issue happen during the transaction

ROLLBACK;

The goal of this transaction is to add a new manufacturer and assign specific products to them. The transaction starts by inserting a new manufacturer into the Manufacturer table with details such as Name, ContactEmail, and Country. After successfully adding the manufacturer, the LAST_INSERT_ID() function retrieves the newly generated ManufacturerID. This ID is then used to update the ManufacturerID column for specific products in the Product table, linking the products to the new manufacturer. If both the manufacturer addition and product updates are successful, the transaction is committed, permanently saving the changes. If any issue arises during the process (for example invalid data or constraint violations), the ROLLBACK ensures that no changes are applied, preserving the consistency of the database.

Performance Tuning Enhancements

In SQL, indexes improve query performance by helping the database to rapidly locate the data requested by a query instead of scanning the whole table to retrieve a few records [4].

Product Table Indexing

CREATE INDEX idx_manufacturer_id ON Product(ManufacturerID);

CREATE INDEX idx_category_id ON Product(CategoryID);

ManufacturerID and CategoryID indexes are foreign keys that are frequently used in joins to extract product information in addition to manufacturer or category data. Join performance will be enhanced by adding indexes on ManufacturerID and CategoryID.

CREATE INDEX idx_product_name ON Product(Name);

Searching for products by name is a common query. Indexing the Name column will speed up searches for specific products.

OrderData Table Indexing

CREATE INDEX idx_customer_id ON OrderData(CustomerID);

An index on the CustomerID column will be useful for queries that retrieve orders by a particular customer. When clients are retrieving their order history, this is quite helpful.

CREATE INDEX idx_order_date ON OrderData(OrderDate);

Adding an index on OrderDate will help maximise performance if queries frequently filter or sort orders by OrderDate (for example, retrieving orders within a given date period).

OrderDetails Table Indexing

CREATE INDEX idx_order_id ON OrderDetails(OrderID);

Adding an index on OrderID will enhance query performance when retrieving details for a particular order, as this column is commonly used to connect with the OrderData table.

CREATE INDEX idx_product_id ON OrderDetails(ProductID);

Index on ProductID: In the same way, queries that use ProductID for filtering or joining will gain from having an index on this column, particularly when examining order information related to a product.

FeedBack Table Indexing

CREATE INDEX idx_feedback_customer_id ON Feedback(CustomerID);

CREATE INDEX idx_feedback_product_id ON Feedback(ProductID);

Adding indexes to CustomerID and ProductID can speed up user searches that are regularly based on customer or product feedback. This is especially useful for getting reviews for a particular product or from a single customer.

Screenshots of Completed Work

SCHEMAS

adv

Tables

- category
- customer
- feedback
- manufacturer
- orderdata
- orderdetails
- product

Views

Stored Procedures

Functions

sys

```

28 Name VARCHAR(100) NOT NULL,
29 Description VARCHAR(100),
30 Price DECIMAL(10, 2) CHECK (Price > 0),
31 ManufacturerID INT,
32 CategoryID INT,
33 FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID), -- Integrity Constraint
34 FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID), -- Integrity Constraint
35 CHECK (Price > 0) -- Check Constraint
36 }
37
38 -- OrderData
39 * CREATE TABLE OrderData (
40 OrderID INT AUTO_INCREMENT PRIMARY KEY,
41 OrderDate DATETIME DEFAULT CURRENT_TIMESTAMP,
42 CustomerID INT,
43 TotalAmount DECIMAL(10, 2) CHECK (TotalAmount > 0),
44 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), -- Integrity Constraint
45 CHECK (TotalAmount > 0) -- Check Constraint
46 }
47
48 -- OrderDetails
49 * CREATE TABLE OrderDetails (
50 OrderDetailID INT AUTO_INCREMENT PRIMARY KEY,
51 OrderID INT,
52 ProductID INT,
53 Quantity INT,
54 FOREIGN KEY (OrderID) REFERENCES OrderData(OrderID), -- Integrity Constraint
55 FOREIGN KEY (ProductID) REFERENCES Product(ProductID), -- Integrity Constraint
56 CHECK (Quantity > 0) -- Check Constraint
57 }
58
59 -- Feedback
60 * CREATE TABLE Feedback (
61 FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
62 CustomerID INT,
63 ProductID INT,
64 Review VARCHAR(100),
65 Rating INT CHECK (Rating BETWEEN 1 AND 5),
66 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), -- Integrity Constraint
67 FOREIGN KEY (ProductID) REFERENCES Product(ProductID), -- Integrity Constraint
68 CHECK (Rating BETWEEN 1 AND 5) -- Check Constraint
69 }
70
71

```

Adminstration Schema

Information

Schema: adv

Output

Time	Action	Message
1 21:02:04	CREATE TABLE Manufacturer (ManufacturerID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, ContactInfo VARCHAR(100) UNIQUE, Country VARCHAR(50))	Successfully affected
2 21:02:04	CREATE TABLE Category (CategoryID INT AUTO_INCREMENT PRIMARY KEY, CategoryName VARCHAR(100) NOT NULL UNIQUE, Email VARCHAR(100) UNIQUE NOT NULL, Address VARCHAR(255))	Successfully affected
3 21:02:04	CREATE TABLE Customer (CustomerID INT AUTO_INCREMENT PRIMARY KEY, FullName VARCHAR(100) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, Address VARCHAR(255))	Successfully affected
4 21:02:04	CREATE TABLE Product (ProductID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Description VARCHAR(255), Price DECIMAL(10, 2) CHECK (Price > 0), ManufacturerID INT)	Successfully affected
5 21:02:04	CREATE TABLE OrderData (OrderID INT AUTO_INCREMENT PRIMARY KEY, OrderDate DATETIME DEFAULT CURRENT_TIMESTAMP, CustomerID INT, TotalAmount DECIMAL(10, 2) CHECK (TotalAmount > 0))	Successfully affected
6 21:02:04	CREATE TABLE OrderDetails (OrderDetailID INT AUTO_INCREMENT PRIMARY KEY, OrderID INT, ProductID INT, Quantity INT, FOREIGN KEY (OrderID) REFERENCES OrderData(OrderID), FOREIGN KEY (ProductID) REFERENCES Product(ProductID), CHECK (Quantity > 0))	Successfully affected
7 21:02:04	CREATE TABLE Feedback (FeedbackID INT AUTO_INCREMENT PRIMARY KEY, CustomerID INT, ProductID INT, Review VARCHAR(100), Rating INT CHECK (Rating BETWEEN 1 AND 5), FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), FOREIGN KEY (ProductID) REFERENCES Product(ProductID), CHECK (Rating BETWEEN 1 AND 5))	Successfully affected

Close Info Session

Navigator

Filter objects

adv

Tables

- category
- customer
- feedback
- manufacturer
- orderdata
- orderdetails
- product

Views

Stored Procedures

Functions

sys

TableStructure

IntegrityConstraintUsed

CheckConstraintsUsed

SampleData

TestingConstraints

Limit to 1000 rows

```

1 -- Check Constraints
2 -- (These are in the TableStructure but placed here for clear view)
3 -- Product Table
4 FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID),
5 FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID),
6
7 -- OrderData Table
8 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
9
10 -- OrderDetails Table
11 FOREIGN KEY (OrderID) REFERENCES OrderData(OrderID),
12 FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
13
14 -- Feedback Table
15 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
16 FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
17

```

Navigator: SCHEMAS Filter objects adv Tables category customer feedback manufacturer orderdata orderdetails product Views Stored Procedures Functions sys

TableStructure IntegrityConstraintUsed CheckConstraintsUsed SampleData TestingConstraints TransactionRollback

```
1 -- Check Constraints
2 -- (These are in the TableStructure but placed here for clear view)
3 -- Manufacturer Table
4 ContactEmail VARCHAR(100) UNIQUE, -- Makes sure that contact email is unique
5
6 -- Product Table
7 CHECK (Price > 0) -- Makes sure that the price of a product is positive.
8
9 -- OrderData Table
10 CHECK (TotalAmount > 0) -- Makes sure that the total amount for an order is positive.
11
12 -- OrderDetails Table
13 CHECK (Quantity > 0) -- Makes sure that the quantity in an order is positive.
14
15 -- Feedback Table
16 CHECK (Rating BETWEEN 1 AND 5) -- Makes sure that the rating is within the range of 1 to 5.
```

Navigator: SCHEMAS Filter objects adv Tables category customer feedback manufacturer orderdata orderdetails product Views Stored Procedures Functions sys

TableStructure IntegrityConstraintUsed CheckConstraintsUsed SampleData TestingConstraints TransactionRollbackCommitOpen PerformanceTuning

```
16 ('Electric Bike'),
17 ('Helmets');
18
19 -- Product Records
20 INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID) VALUES
21 ('Trailblazer MTB', 'Mountain bike for rough terrains', 599.99, 1, 1),
22 ('Speedster Road Bike', 'Lightweight road bike', 799.99, 2, 2),
23 ('Xtreme E-Bike', 'Electric bike with powerful motor', 1299.99, 4, 4),
24 ('Helix Helmet', 'High-quality cycling helmet', 49.99, 5, 5),
25 ('Gear Up Accessories', 'Bike repair tools and gear', 29.99, 3, 3);
26
27 -- Customer Records
28 INSERT INTO Customer (FullName, Email, Address) VALUES
29 ('John Doe', 'john@example.com', '123 Main St, USA'),
30 ('Jane Smith', 'jane@example.com', '456 Elm St, Canada'),
31 ('Mike Johnson', 'mike@example.com', '789 Pine St, UK'),
32 ('Sara Lee', 'sara@example.com', '101 Maple Ave, Germany'),
33 ('Tom Hardy', 'tom@example.com', '202 Oak Dr, France');
34
35 -- OrderData Records
36 INSERT INTO OrderData (CustomerID, TotalAmount) VALUES
37 (1, 599.99),
38 (2, 799.99),
39 (3, 1299.99),
40 (4, 49.99),
41 (5, 29.99);
42
43 -- OrderDetails Records
44 INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES
45 (1, 1, 1),
46 (2, 2, 1),
47 (3, 3, 1),
48 (4, 4, 1),
49 (5, 5, 1);
50
51 -- Feedback Records
52 INSERT INTO Feedback (CustomerID, ProductID, Review, Rating) VALUES
53 (1, 1, 'Excellent bike!', 5),
54 (2, 2, 'Smooth and fast ride', 4),
55 (3, 3, 'Incredible e-bike experience!', 5),
56 (4, 4, 'Great helmet, comfortable and safe.', 4),
57 (5, 5, 'Useful gear set for bike maintenance.', 5);
58
59
```

Administration Schemas Information

Schema: adv

Output

#	Time	Action	Message
1	23.08.21	INSERT INTO Manufacturer (Name, ContactEmail, Country) VALUES ('Bike Co', 'contact@bikeco.com', 'USA'), ('Trail Masters', 'info@trailmasters.com', 'Canada'), ('Urban Cycles', 'support@urbancycles.com', 'UK'), ('Cycle	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
2	23.08.21	INSERT INTO Category (CategoryName) VALUES ('Mountain Bike'), ('Road Bike'), ('Accessories'), ('Electric Bike'), ('Helmets')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
3	23.08.21	INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID) VALUES ('TrailBlazer MTB', 'Mountain bike for rough terrains', 599.99, 1, 1), ('Speedster Road Bike', 'Lightweight road bike', 799.99, 2, 2), ('X	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
4	23.08.21	INSERT INTO Customer (FullName, Email, Address) VALUES ('John Doe', 'john@example.com', '123 Main St, USA'), ('Jane Smith', 'jane@example.com', '456 Elm St, Canada'), ('Mike Johnson', 'mike@example.com', '789	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
5	23.08.21	INSERT INTO OrderData (CustomerID, TotalAmount) VALUES (1, 599.99), (2, 799.99), (3, 1299.99), (4, 49.99), (5, 29.99)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
6	23.08.21	INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES (1, 1, 1), (2, 2, 1), (3, 3, 1), (4, 4, 1), (5, 5, 1)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
7	23.08.21	INSERT INTO Feedback (CustomerID, ProductID, Review, Rating) VALUES (1, 1, 'Excellent bike!', 5), (2, 2, 'Smooth and fast ride', 4), (3, 3, 'Incredible e-bike experience!', 5), (4, 4, 'Great helmet, comfortable and safe',	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

Navigation

Schemas

adv

Tables

category

customer

feedback

manufacturer

orderdata

orderdetails

product

Views

Stored Procedures

Functions

sys

TableStructure

IntegrityConstraintUsed

CheckConstraintsUsed

SampleData

Testing_Constraints

TransactionRollbackCommitOpe...

Performance Tuning

1

-- Testing Constraints

2

3

-- Test Check Constraint: Price > 0 (Product Table)

4

-- Attempt to insert a product with a negative price

5

INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID)

6

VALUES ('Test Product', 'Negative Price Test', -10, NULL, NULL);

7

8

-- Test Check Constraint: TotalAmount > 0 (OrderData Table)

9

-- Attempt to insert an order with a negative total amount

10

INSERT INTO OrderData (CustomerID, TotalAmount)

11

VALUES (NULL, -50);

12

13

-- Test Check Constraint: Quantity > 0 (OrderDetails Table)

14

-- Attempt to insert an order detail with a zero quantity

15

INSERT INTO OrderDetails (OrderID, ProductID, Quantity)

16

VALUES (NULL, NULL, 0);

17

18

-- Test Check Constraint: Rating BETWEEN 1 AND 5 (Feedback Table)

19

-- Attempt to insert feedback with an invalid rating

20

INSERT INTO Feedback (CustomerID, ProductID, Review, Rating)

21

VALUES (NULL, NULL, 'Invalid Rating Test', 6);

22

23

-- Test Unique Constraint: ContactEmail in Manufacturer Table

24

-- Insert two manufacturers with the same email

25

INSERT INTO Manufacturer (Name, ContactEmail, Country)

26

VALUES ('Manufacturer 1', 'duplicate@example.com', 'USA');

27

28

INSERT INTO Manufacturer (Name, ContactEmail, Country)

29

VALUES ('Manufacturer 2', 'duplicate@example.com', 'USA');

30

31

32

Administration

Schemas

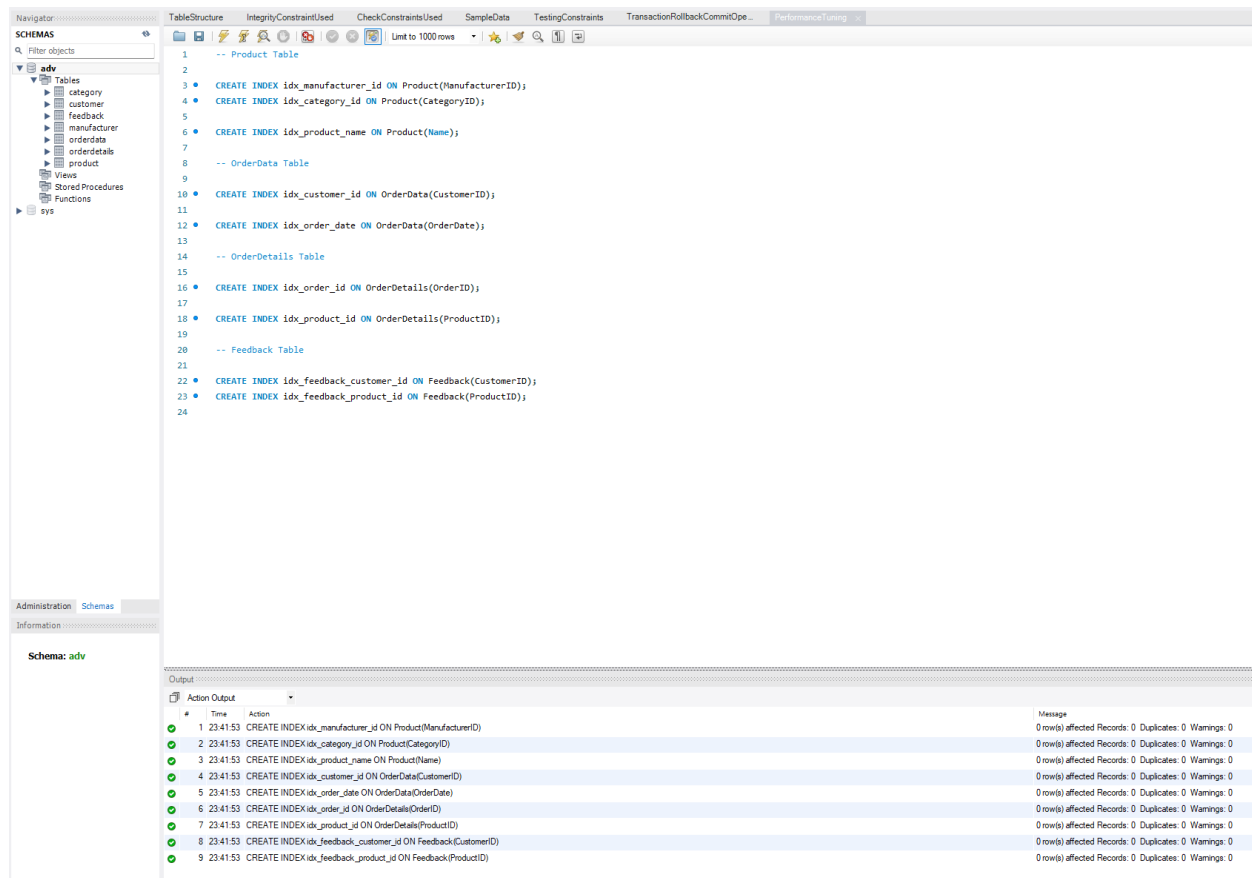
Information

Schema: adv

Output

Action Output

#	Time	Action	Message
1	23:37:11	INSERT INTO Product (Name, Description, Price, ManufacturerID, CategoryID) VALUES ('Test Product', 'Negative Price Test', -10, NULL, NULL)	Error Code: 3819 Check constraint 'product_chk_1' is violated.
2	23:37:20	INSERT INTO OrderData (CustomerID, TotalAmount) VALUES (NULL, -50)	Error Code: 3819 Check constraint 'orderdata_chk_1' is violated.
3	23:37:25	INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES (NULL, NULL, 0)	Error Code: 3819 Check constraint 'orderdetails_chk_1' is violated.
4	23:37:29	INSERT INTO Feedback (CustomerID, ProductID, Review, Rating) VALUES (NULL, NULL, 'Invalid Rating Test', 6)	Error Code: 3819 Check constraint 'feedback_chk_1' is violated.
5	23:37:32	INSERT INTO Manufacturer (Name, ContactEmail, Country) VALUES ('Manufacturer 1', 'duplicate@example.com', 'USA')	1 row(s) affected
6	23:37:36	INSERT INTO Manufacturer (Name, ContactEmail, Country) VALUES ('Manufacturer 2', 'duplicate@example.com', 'USA')	Error Code: 1062 Duplicate entry 'duplicate@example.com' for key 'Manufacturer.ContactEmail'



Conclusion

In Conclusion, the database schema created for AdventureWorks Cycles provides a solid base for the platform's operations, allowing for efficient data management, rapid transaction processing, and scalability for future expansion. The method of my schema not only fulfils the current business needs but also allows for future expansions and additional features.

References

[1] GeeksforGeeks, "Third normal form (3NF)," GeeksforGeeks, Dec. 27, 2024.

<https://www.geeksforgeeks.org/third-normal-form-3nf/>

[2] "What is Referential Integrity and Why do You Need it?" <https://intelligent-ds.com/blog/what-is-referential-integrity>

[3] GeeksforGeeks, "Difference between COMMIT and ROLLBACK in SQL,"

GeeksforGeeks, Dec. 19, 2024. <https://www.geeksforgeeks.org/difference-between-commit-and-rollback-in-sql/>

[4] V. Barvaliya, "Understanding Indexing in SQL: Improve Query Performance with Data Structures," Medium, Sep. 08, 2024. [Online]. Available: <https://medium.com/towards-data-engineering/understanding-indexing-in-sql-improve-query-performance-with-data-structures-99e275f29c7d>

Appendix

Ai Prompt – Generate me dummy data for my schema for the following tables:

{TableStructure code here}

ERD Tool - <https://online.visual-paradigm.com/diagrams/features/erd-tool/>