

DATABASE DESIGN AND PROGRAMING

by Sohaan Sureshkumar

Submission date: 09-Jan-2025 12:49AM (UTC+0000)

Submission ID: 247718039

File name: 51186_Sohaansureshkumar_DATABASE_DESIGN_AND_PROGRAMING_621315_541032222.docx
(7.99M)

Word count: 2353

Character count: 13095



DATABASE DESIGN AND PROGRAMING

PART A

Submitted By:

Sohaam Sureshkumar

2221048

Module: Databases and Application Development



Table of Contents

INTRODUCTION	 1	3
1.1 AdventureWorks Cycles Case Study		3
ENTITY RELATIONSHIP DIAGRAM (ERD)		4
2.1 Conceptual ERD		4
2.2 Schema Diagram		5
DATABASE IMPLEMENTATION		6
3.1 Table Creation and Relationships		6
3.1.1 Manufacturer Table.....		6
3.1.2 Category Table.....		6
3.1.3 Product Table.....		7
3.1.4 Customer Table.....		7
3.1.5 OrderTable Table		7
3.1.6 OrderDetails Table.....		8
3.1.7 Feedback Table.....		8
3.2 Data Population.....		9
3.2.1 Manufacture Table		10
3.2.2 Category Table.....		10
3.2.3 Product Table.....		11
3.2.4 Customer Table.....		11
3.2.5 OrderTable Table		12
3.2.6 OrderDetails Table.....		13
3.2.7 Feedback Table.....		13
3.2.8 All Action Outputs For Population:		14
3.3 Testing Constraints.....		14
3.3.1 Primary Key Constraints		14
Test Case 1: Attempt Duplicate Primary Key.....		14
Test Case 2: Attempt NULL Primary Key		15
3.3.2 Foreign Key Constraints.....		15
Test Case 1: Insert with Non-Existent Foreign Key		15
Test Case 2: Attempt to Delete a Record with Existing Dependencies		16
3.3.3 Check Constraints.....		16
Test Case 1: Inserting an Invalid Price.....		16
Test Case 2: Inserting an Invalid Rating		16
TRANSACTIONAL SCENARIOS		17
4.1 Successful Transaction (COMMIT).....		17
4.2 Failed Transaction (ROLLBACK).....		20
PERFORMANCE TUNING ENHANCEMENTS		22
5.1 Important Query Requirements		22
5.2 Data Queries without Indexing		22
5.2.1 Fetching All Orders for a Given Customer		22

5.2.2 Checking Product Availability	23
5.3 Data Queries Indexing	23
5.3.1 Get order details for a particular order	24
5.3.2 Retrieving Product based on Category.....	24
5.3.3 Search Feedback by its Rating	24
5.4 Secondary Indexing Rationale	25
REFERENCES.....	26

INTRODUCTION

1.1 AdventureWorks Cycles Case Study

AdventureWorks Cycles is a leading bicycle manufacturing company located in the UK dealing in road bikes, mountain bikes, hybrid bikes, and cycling accessories. The company has been in operation for many years as one of the leading suppliers of cycling products and services and is currently planning to enhance its business by developing a new online marketplace. This new platform will help the cycling enthusiasts to search, select, and assemble products from various manufacturers across the globe [1].



In order to meet these requirements, AdventureWorks Cycles needs a solid data infrastructure to support a variety of business processes such as product listing, registration of customers, ordering, and feedback [2]. Also, the database must handle high concurrency, maintain data integrity and provide high performance for the frequently used queries.

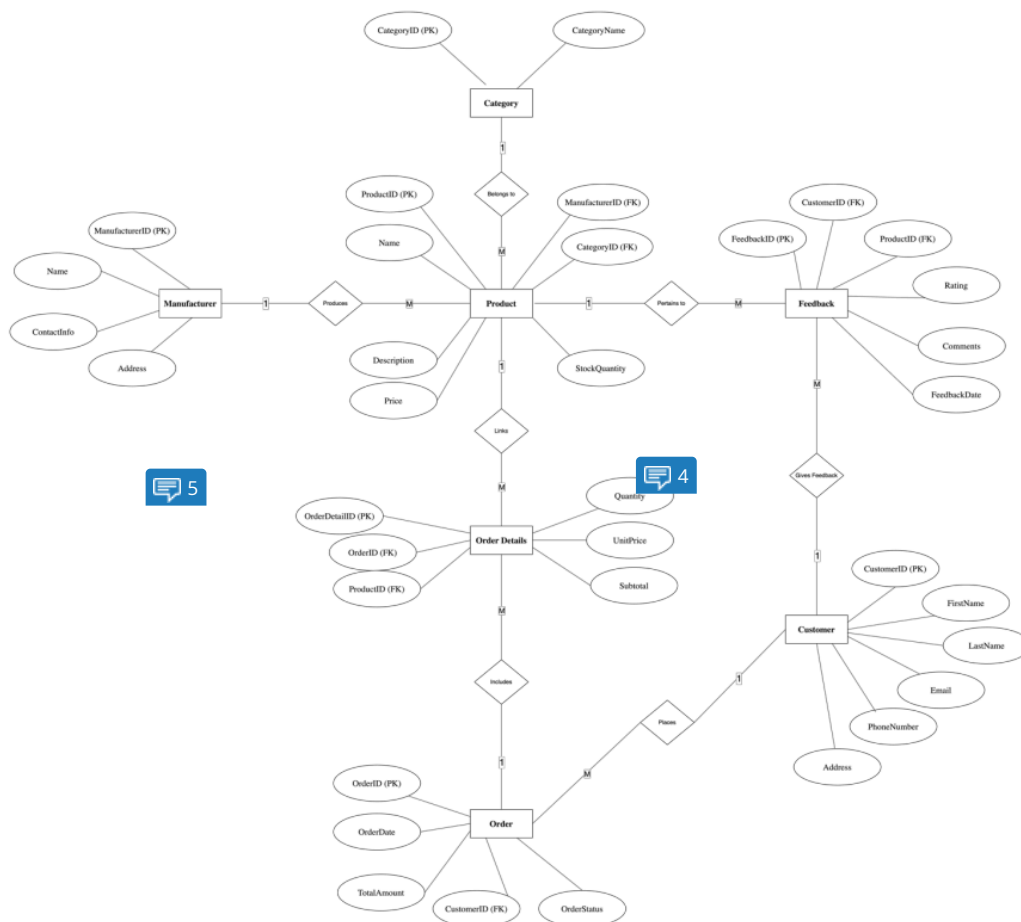


The output of this paper is to Identify a suitable database for AdventureWorks Cycles and to design and execute a database that meets their business needs. This involves designing an ERD, designing a normalized database, checking data consistencies using transactions, and improving query performance using indexing.

ENTITY RELATIONSHIP DIAGRAM (ERD)

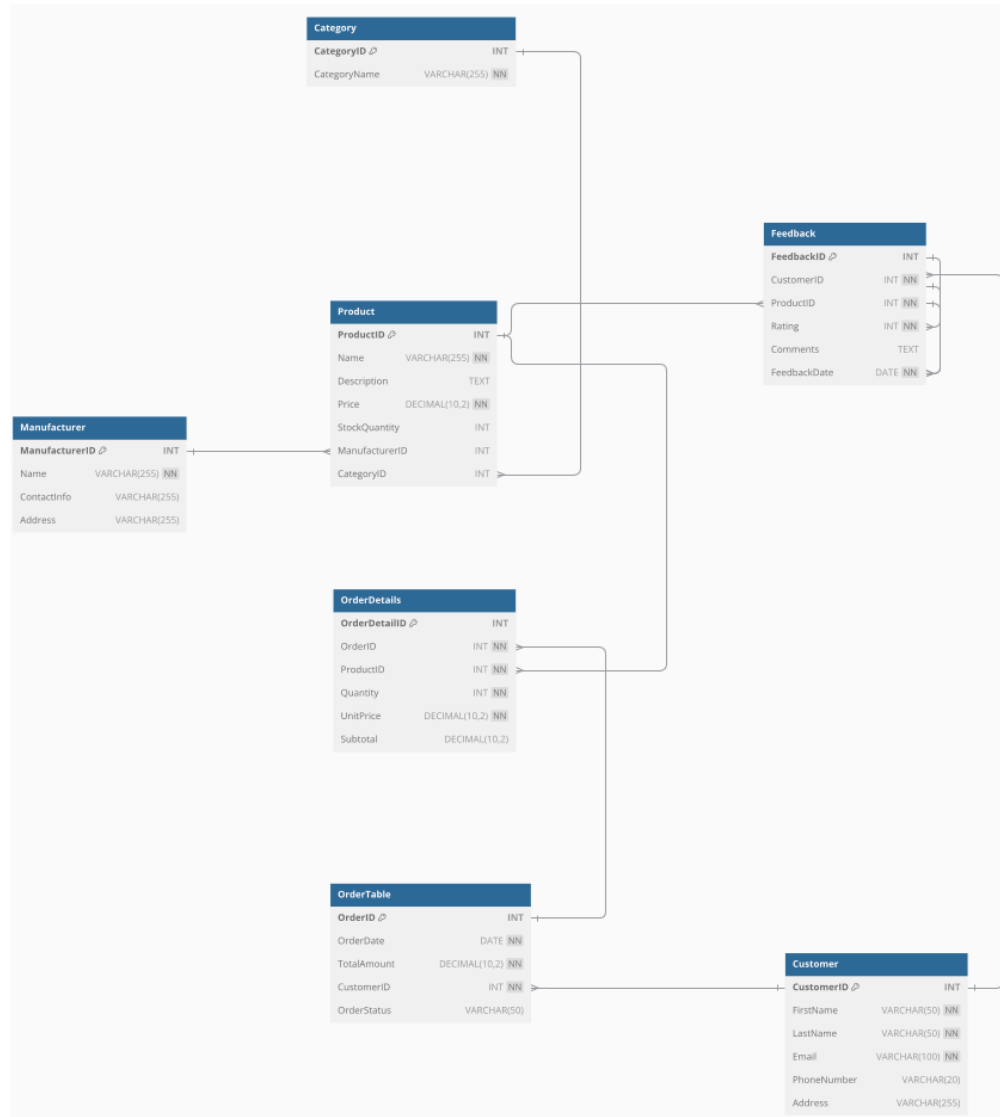
2.1 Conceptual ERD

The conceptual ERD for AdventureWorks Cycles reflects the most basic level of database structure that is required to support the operations of the online marketplace [3]. This diagram emphasizes the main entities and their relationships to ensure that data is moved around the system in the most efficient manner possible in order to support business processes like listing of products, processing of customer orders, and managing feedback.



2.2 Schema Diagram

The schema diagram below is a detailed representation of the database structure which includes all the tables, fields and the relationship between the tables as well [4].

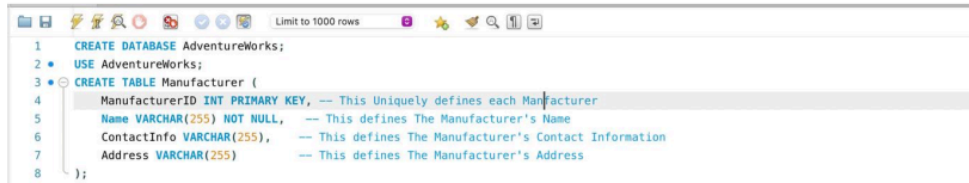


DATABASE IMPLEMENTATION

3.1 Table Creation and Relationships

3.1.1 Manufacturer Table

The Manufacturer table contains information about the companies that produce the products for AdventureWorks Cycles. Every manufacturer has a unique identification number called ManufacturerID which is set as the PRIMARY KEY.



```
1 CREATE DATABASE AdventureWorks;
2 USE AdventureWorks;
3 CREATE TABLE Manufacturer (
4     ManufacturerID INT PRIMARY KEY, -- This Uniquely defines each Manufacturer
5     Name VARCHAR(255) NOT NULL, -- This defines The Manufacturer's Name
6     ContactInfo VARCHAR(255), -- This defines The Manufacturer's Contact Information
7     Address VARCHAR(255) -- This defines The Manufacturer's Address
8 );
```

Action Output:

10	12:36:10	CREATE DATABASE AdventureWorks	1 row(s) affected	0.0030 sec
11	12:36:10	USE AdventureWorks	0 row(s) affected	0.00061 sec
12	12:36:10	CREATE TABLE Manufacturer (ManufacturerID INT PRIMARY KEY, -- This Uniquely defines each Manufacturer Name V...	0 row(s) affected	0.010 sec

3.1.2 Category Table

To improve product navigation, the Category table is used to divide products into general groups (i.e., road bikes, accessories).



```
10 CREATE TABLE Category {
11     CategoryID INT PRIMARY KEY, -- This Uniquely defines each Category
12     CategoryName VARCHAR(255) NOT NULL -- This defines The Category's Name
13 };
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:23:16	CREATE TABLE Category (CategoryID INT PRIMARY KEY, -- This Uniquely defines each Category CategoryName VA...	0 row(s) affected	0.072 sec

3.1.3 Product Table

The Product table contains information on products available in the market, such as price, description, and the quantity available in stock. It has Foreign keys to Manufacturer and Category tables [5].

```
15 CREATE TABLE Product (  
16     ProductID INT PRIMARY KEY,           -- This Uniquely defines each Product  
17     Name VARCHAR(255) NOT NULL,          -- This defines The Name of the product  
18     Description TEXT,                    -- This defines The Description of the product  
19     Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0), -- This defines The Price of the product which allows positive value  
20     StockQuantity INT,                  -- This defines The Available Product stock which cannot be negative  
21     ManufacturerID INT,                 -- Foreign key to The Manufacturer table  
22     CategoryID INT,                    -- Foreign key to The Category table  
23     FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID),  
24     FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)  
25 );
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:28:42	CREATE TABLE Product (ProductID INT PRIMARY KEY, -- This Uniquely defines each Product Name VARCHAR(255) NOT NULL, -- This defines The Name of the product Description TEXT, -- This defines The Description of the product Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0), -- This defines The Price of the product which allows positive value StockQuantity INT, -- This defines The Available Product stock which cannot be negative ManufacturerID INT, -- Foreign key to The Manufacturer table CategoryID INT, -- Foreign key to The Category table FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID), FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)	0 row(s) affected	0.018 sec

3.1.4 Customer Table

The Customer table contains data about customers who make orders on the given platform.

```
26 CREATE TABLE Customer (  
27     CustomerID INT PRIMARY KEY,           -- This Uniquely defines each Customer  
28     FirstName VARCHAR(50) NOT NULL,       -- This defines The First Name of the Customer  
29     LastName VARCHAR(50) NOT NULL,       -- This defines The Last Name of the Customer  
30     Email VARCHAR(100) UNIQUE NOT NULL,  -- This defines Customer's Unique email  
31     PhoneNumber VARCHAR(20),             -- This defines Customer's Phone Number  
32     Address VARCHAR(255)                 -- This defines Customer's Home Address  
33 );
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:31:54	CREATE TABLE Customer (CustomerID INT PRIMARY KEY, -- This Uniquely defines each Customer FirstName VARCHAR(50) NOT NULL, -- This defines The First Name of the Customer LastName VARCHAR(50) NOT NULL, -- This defines The Last Name of the Customer Email VARCHAR(100) UNIQUE NOT NULL, -- This defines Customer's Unique email PhoneNumber VARCHAR(20), -- This defines Customer's Phone Number Address VARCHAR(255) -- This defines Customer's Home Address	0 row(s) affected	0.031 sec

3.1.5 OrderTable Table

The OrderTable contains information about the customer's purchases including order date, the total amount and the status of the order.

```
34 CREATE TABLE OrderTable (  
35     OrderID INT PRIMARY KEY,           -- This Uniquely defines each Order Placed  
36     OrderDate DATE NOT NULL,           -- This defines The Order Date  
37     TotalAmount DECIMAL(10, 2) NOT NULL, -- This defines The Total cost of the order  
38     CustomerID INT NOT NULL,           -- Foreign key to Customer table  
39     OrderStatus VARCHAR(50),           -- This defines The Status of the order (e.g., Pending, Shipped)  
40     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)  
41 );
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:35:51	CREATE TABLE OrderTable (OrderID INT PRIMARY KEY, -- This Uniquely defines each Order Placed OrderDate D...	0 row(s) affected	0.074 sec

3.1.6 OrderDetails Table

The OrderDetails table contains the details of every order made which include the quantity of the product, the price of the unit, and the sub-total [5].

```
42 CREATE TABLE OrderDetails (  
43     OrderDetailID INT PRIMARY KEY,      -- This Uniquely defines each Order Placed  
44     OrderID INT NOT NULL,               -- Foreign key to OrderTable  
45     ProductID INT NOT NULL,            -- Foreign key to Product  
46     Quantity INT NOT NULL CHECK (Quantity > 0), -- This defines The Number of products ordered and checks if its greater than 0  
47     UnitPrice DECIMAL(10, 2) NOT NULL CHECK (UnitPrice > 0), -- This defined Price of a product and checks if its greater than 0  
48     Subtotal DECIMAL(10, 2) GENERATED ALWAYS AS (Quantity * UnitPrice) STORED, -- This defines The Calculated subtotal  
49     FOREIGN KEY (OrderID) REFERENCES OrderTable(OrderID),  
50     FOREIGN KEY (ProductID) REFERENCES Product(ProductID)  
51 );
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:37:09	CREATE TABLE OrderDetails (OrderDetailID INT PRIMARY KEY, -- This Uniquely defines each Order Placed Order...	0 row(s) affected	0.047 sec

3.1.7 Feedback Table

The Feedback table turn helps helps in the enhancing customer the in quality providing of feedback the and products reviews as along well with as ratings the for experience various of products the which customers. in

```
52 CREATE TABLE Feedback (
53     FeedbackID INT PRIMARY KEY,           -- This Uniquely defines each Feedback submitted
54     CustomerID INT NOT NULL,              -- Foreign key to Customer table
55     ProductID INT NOT NULL,               -- Foreign key to Product table
56     Rating INT NOT NULL CHECK (Rating BETWEEN 1 AND 5), -- This defines Rating recieved (1 to 5) and checks if its between 1 and 5
57     Comments TEXT,                        -- This defines The comments of each Feedback
58     FeedbackDate DATE NOT NULL,           -- This defines The Date the feedback was submitted
59     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
60     FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
61 );
```

Action Output:

	Time	Action	Response	Duration / Fetch Time
1	14:40:27	CREATE TABLE Feedback (FeedbackID INT PRIMARY KEY, -- This Uniquely defines each Feedback submitted	C... 0 row(s) affected	0.076 sec

3.2 Data Population

3.2.1 Manufacture Table

We are inserting sample manufacturers into the Manufacturer table for companies that produce products for AdventureWorks Cycles. The manufacturer is linked to a unique ManufacturerID and has details such as the manufacturer's name, contact information and address. This data is crucial to products being linked to their manufacturers.

```
62 • INSERT INTO Manufacturer (ManufacturerID, Name, ContactInfo, Address)
63 VALUES
64 (1, 'UKCycleWorks', 'support@ukcycleworks.co.uk', '101 Oxford Street, London, W1D 1LL'),
65 (2, 'TrailMasters Ltd', 'info@trailmasters.co.uk', '202 High Street, Manchester, M4 1HQ'),
66 (3, 'UrbanCyclists Co', 'contact@urbancyclists.co.uk', '303 Princes Street, Edinburgh, EH2 4AD'),
67 (4, 'PeakCycles UK', 'info@peakcycles.co.uk', '404 Castle Street, Cardiff, CF10 1SZ');
```

Output:

Manufacturer...	Name	ContactInfo	Address
1	UKCycleWorks	support@ukcycleworks.co.uk	101 Oxford Street, London, W1D 1LL
2	TrailMasters Ltd	info@trailmasters.co.uk	202 High Street, Manchester, M4 1HQ
3	UrbanCyclists Co	contact@urbancyclists.co.uk	303 Princes Street, Edinburgh, EH2 4AD
4	PeakCycles UK	info@peakcycles.co.uk	404 Castle Street, Cardiff, CF10 1SZ
NULL	NULL	NULL	NULL

3.2.2 Category Table

We are populating the Category table with various categories of products accessories. i.e. Each bikes category and is cycling assigned a unique CategoryID and a descriptive name. This data helps classify the products for easier navigation and management [6].

```
69 • INSERT INTO Category (CategoryID, CategoryName)
70 VALUES
71 (1, 'Road Bikes'),
72 (2, 'Mountain Bikes'),
73 (3, 'Hybrid Bikes'),
74 (4, 'Electric Bikes'),
75 (5, 'Cycling Accessories');
```

Output:

CategoryID	CategoryName	
1	Road Bikes	
2	Mountain Bikes	
3	Hybrid Bikes	
4	Electric Bikes	
5	Cycling Accessories	
NULL	NULL	

3.2.3 Product Table

In this, we are entering certain products into the Product table, containing information such as the product name, the product description, the category price, to which quantity it belongs in.

```

77 * INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID)
78 VALUES
79 (1, 'RoadMaster 500', 'Lightweight road bike for UK streets.', 1100.00, 5, 1, 1),
80 (2, 'TrailRider Pro', 'Durable mountain bike for UK trails.', 1500.00, 10, 2, 2),
81 (3, 'CityExplorer', 'Versatile hybrid bike for urban rides.', 950.00, 8, 3, 3),
82 (4, 'EcoRide E-Bike', 'Eco-friendly electric bike for commuters.', 2700.00, 7, 1, 4),
83 (5, 'CycleSafe Helmet', 'High-quality helmet for cycling safety.', 80.00, 4, 2, 5),
84 (6, 'TrailPack Accessories', 'Essential accessories for mountain biking.', 150.00, 3, 4, 5);
85

```

Output:

ProductID	Name	Description	Price	StockQuanti...	Manufacturer...	CategoryID
1	RoadMaster 500	Lightweight road bike for UK streets.	1100.00	5	1	1
2	TrailRider Pro	Durable mountain bike for UK trails.	1500.00	10	2	2
3	CityExplorer	Versatile hybrid bike for urban rides.	950.00	8	3	3
4	EcoRide E-Bike	Eco-friendly electric bike for commuters.	2700.00	7	1	4
5	CycleSafe Helmet	High-quality helmet for cycling safety.	80.00	4	2	5
6	TrailPack Accessories	Essential accessories for mountain biking.	150.00	3	4	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.2.4 Customer Table

In this, we are entering information about customers into the Customer table which includes the customer's name, phone number, and address. Every customer has a CustomerID through which he or she is identified and which will be useful when it comes to processing orders and feedback.

```
86 * INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Address)
87 VALUES
88 (1, 'Emily', 'Thompson', 'emily.thompson@example.co.uk', '07123456789', '10 Downing Street, London, SW1A 2AA'),
89 (2, 'James', 'Harris', 'james.harris@example.co.uk', '07987654321', '25 King Street, Manchester, M2 6AG'),
90 (3, 'Oliver', 'Walker', 'oliver.walker@example.co.uk', '07711223344', '50 Queen Street, Glasgow, G1 3DN'),
91 (4, 'Charlotte', 'Evans', 'charlotte.evans@example.co.uk', '07567890123', '75 Victoria Street, Bristol, BS1 6AA');
92
```

Output :

	CustomerID	FirstName	LastName	Email	PhoneNumber	Address
	1	Emily	Thompson	emily.thompson@example.co.uk	07123456789	10 Downing Street, London, SW1A 2AA
	2	James	Harris	james.harris@example.co.uk	07987654321	25 King Street, Manchester, M2 6AG
	3	Oliver	Walker	oliver.walker@example.co.uk	07711223344	50 Queen Street, Glasgow, G1 3DN
	4	Charlotte	Evans	charlotte.evans@example.co.uk	07567890123	75 Victoria Street, Bristol, BS1 6AA
	NULL	NULL	NULL	NULL	NULL	NULL

3.2.5 OrderTable Table

We are creating sample orders in the OrderTable which includes all the Orders made by customers

```
93 * INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus)
94 VALUES
95 (1, '2024-12-01', 1900.00, 1, 'Pending'), -- Customer Emily ORDERS RoadMaster 500 and CycleSafe Helmet
96 (2, '2024-12-02', 1500.00, 2, 'Completed'), -- Customer James ORDERS TrailRider Pro
97 (3, '2024-12-03', 2700.00, 3, 'Shipped'), -- Customer Oliver ORDERS EcoRide E-Bike
98 (4, '2024-12-04', 200.00, 4, 'Pending'); -- Customer Charlotte ORDERS CycleSafe Helmets
99
```

Output:

	OrderID	OrderDate	TotalAmount	CustomerID	OrderStatus
	1	2024-12-01	1900.00	1	Pending
	2	2024-12-02	1500.00	2	Completed
	3	2024-12-03	2700.00	3	Shipped
	4	2024-12-04	200.00	4	Pending
	NULL	NULL	NULL	NULL	NULL

3.2.6 OrderDetails Table

We are using sample information to populate the OrderDetails table with details about the products picked in each order. This table is used to hold the quantity and unit price of the items to calculate the subtotal of each order.

```
100 * INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice)
101 VALUES
102 (1, 1, 1, 1, 1100.00), -- Customer Emily ORDERS 1 RoadMaster 500
103 (2, 1, 5, 1, 80.00), -- Customer Emily also ORDERS 1 CycleSafe Helmet
104 (3, 2, 2, 1, 1500.00), -- Customer James ORDERS 1 TrailRider Pro
105 (4, 3, 4, 1, 2700.00), -- Customer Oliver ORDERS 1 EcoRide E-Bike
106 (5, 4, 5, 2, 80.00); -- Customer Charlotte ORDERS 2 CycleSafe Helmets
107
```

Output:

OrderDetailID	OrderID	ProductID	Quantity	UnitPrice	Subtotal
1	1	1	1	1100.00	1100.00
2	1	5	1	80.00	80.00
3	2	2	1	1500.00	1500.00
4	3	4	1	2700.00	2700.00
5	4	5	2	80.00	160.00
NULL	NULL	NULL	NULL	NULL	NULL

3.2.7 Feedback Table

We are using this to populate the Feedback table with ratings and reviews of products that customers have purchased. This table is used to demonstrate customer feedback data for product evaluation and improvement [7].

```
108 * INSERT INTO Feedback (FeedbackID, CustomerID, ProductID, Rating, Comments, FeedbackDate)
109 VALUES
110 (1, 1, 1, 5, 'Fantastic bike for city rides!', '2024-12-05'), -- Customer Emily's FEEDBACK for RoadMaster 500
111 (2, 2, 2, 4, 'TrailRider Pro handles rough terrains well.', '2024-12-06'), -- Customer James's FEEDBACK for TrailRider Pro
112 (3, 3, 4, 5, 'EcoRide E-Bike is perfect for long commutes.', '2024-12-07'), -- Customer Oliver's FEEDBACK for EcoRide E-Bike
113 (4, 4, 5, 4, 'Comfortable and secure helmet.', '2024-12-08'); -- Customer Charlotte's FEEDBACK for CycleSafe Helmet
114
```

Output:

FeedbackID	CustomerID	ProductID	Rating	Comments	FeedbackDate
1	1	1	5	Fantastic bike for city rides!	2024-12-05
2	2	2	4	TrailRider Pro handles rough terrains well.	2024-12-06
3	3	4	5	EcoRide E-Bike is perfect for long commutes.	2024-12-07
4	4	5	4	Comfortable and secure helmet.	2024-12-08
NULL	NULL	NULL	NULL	NULL	NULL

3.2.8 All Action Outputs For Population:

Action Output				
	Time	Action	Response	Duration / Fetch Time
1	15:25:11	INSERT INTO Manufacturer (ManufacturerID, Name, ContactInfo, Address) VALUES (1, 'UKCycleWorks', 'support@ukcycle...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.061 sec
2	15:25:12	INSERT INTO Category (CategoryID, CategoryName) VALUES (1, 'Road Bikes'), (2, 'Mountain Bikes'), (3, 'Hybrid Bikes'), (4, 'E-Bikes')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.0080 sec
3	15:25:12	INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID) VALUES (1, 'Road Bike', 'A sleek road bike with a lightweight frame and high-speed tires.', 1200, 10, 1, 1)	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.0041 sec
4	15:25:12	INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Address) VALUES (1, 'Emily', 'Thompson', 'emily.thompson@example.com', '555-123-4567', '123 Main St, New York, NY 10001')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.0017 sec
5	15:25:12	INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus) VALUES (1, '2024-12-01', 1900.00, 1, 'Completed')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.0012 sec
6	15:25:12	INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUES (1, 1, 1, 1, 1100.00), (2, 1, 2, 1, 800.00)	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.0018 sec
7	15:25:12	INSERT INTO Feedback (FeedbackID, CustomerID, ProductID, Rating, Comments, FeedbackDate) VALUES (1, 1, 1, 5, 'Fantastic bike, great customer service!', '2024-12-01')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.0011 sec

3.3 Testing Constraints

3.3.1 Primary Key Constraints

Test Case 1: Attempt Duplicate Primary Key

To verify that the database does not accept duplicate primary keys, we deliberately attempt to insert a duplicate CustomerID into the Customer table.

```
121 -- Primary Key Constraints
122
123 -- 1. Test Case 1: Attempt Duplicate Primary Key
124
125 * INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Address)
126 VALUES (1, 'Duplicate', 'Customer', 'duplicate.customer@example.co.uk', '07800000000', '123 Test Street, London'); -- This should fail because there is already a "CustomerID=1"
127
```

Test Case 2: Attempt NULL Primary Key

We try to insert a record into the Product table with a NULL value for ProductID, which is the primary key. This scenario shows that the database enforces the rule that primary keys cannot be null.

```
128 -- 2. Test Case 2: Attempt NULL Primary Key
129
130 * INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID)
131 VALUES (NULL, 'Null ID Product', 'Testing Constraint.', 500.00, 10, 1, 1); -- This should fail because "ProductID" cant be NULL
132
```

Action Outputs for both Tests

Action Output				
	Time	Action	Response	Duration / Fetch Time
1	16:27:03	INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Address) VALUES (1, 'Duplicate', 'Customer...	Error Code: 1062. Duplicate entry '1' for key 'custome...	0.00051 sec
2	16:27:08	INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID) VALUES (NULL, 'Nul...	Error Code: 1048. Column 'ProductID' cannot be null	0.0030 sec

3.3.2 Foreign Key Constraints

Test Case 1: Insert with Non-Existent Foreign Key

This is a test that tries to insert a record into the Product table with a non-existent ManufacturerID. This test ensures that the database does not allow a record to reference an invalid foreign key.

```

135      -- Test Case 1: Insert with Non-Existent Foreign Key
136
137 •   INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID)
138     VALUES (6, 'Invalid Manufacturer', 'Testing Foreign Key Constraint.', 700.00, 5, 99, 1); -- "ManufacturerID = 99" is non-existent
139

```

Test Case 2: Attempt to Delete a Record with Existing Dependencies

This shows what happens when we try to delete a record from the Manufacturer table which is being referenced by Product table. This test verifies that the database is preventing the deletion of a parent record when there are child records, thus ensuring the integrity of the relationships between the tables.

```

140      -- Test Case 2: Attempt to Delete a Record with Existing Dependencies
141
142 •   DELETE FROM Manufacturer WHERE ManufacturerID = 1; -- This should fail because products depend on "ManufacturerID = 1".
143
144

```

Action Outputs for both Tests:

Action Output				
	Time	Action	Response	Duration / Fetch Time
1	16:28:49	INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID) VALUES (6, 'Invalid...	Error Code: 1062. Duplicate entry '6' for key 'product...	0.0027 sec
2	16:28:52	DELETE FROM Manufacturer WHERE ManufacturerID = 1	Error Code: 1451. Cannot delete or update a parent re...	0.016 sec

3.3.3 Check Constraints

Test Case 1: Inserting an Invalid Price

When we try to insert a product into the Product table with a negative price, this is contrary to the CHECK constraint which states that the Price column must contain positive value. This scenario is therefore valid to show that the database stops the insertion of invalid values that are not within the specified conditions.

```

147      -- Test Case 1: Inserting an Invalid Price
148
149 •   INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID)
150     VALUES (7, 'Invalid Price ', 'Product with negative price.', -100.00, 10, 1, 1); -- This should fail as Price Value is not positive
151

```

Test Case 2: Inserting an Invalid Rating

In this test, we are entering a rating in the Feedback table outside of the 1 to 5 allowed range. This test guarantees that the database enforces the rule that ratings

must be between 1 and 5, making sure that the feedback data is accurate and reliable.

```
152      -- Test Case 2: Inserting an Invalid Rating
153
154      INSERT INTO Feedback (FeedbackID, CustomerID, ProductID, Rating, Comments, FeedbackDate)
155      VALUES (4, 1, 1, 6, 'Invalid rating 5.', '2024-12-08'); -- This should fail as Rating entered is greater than the Max value
```

Action Outputs for both Tests:



Action Output			
	Time	Action	Response
1	16:29:35	INSERT INTO Product (ProductID, Name, Description, Price, StockQuantity, ManufacturerID, CategoryID) VALUES (7, 'invalid...	Error Code: 3819. Check constraint 'product_chk_1' is... 0.0034 sec
2	16:29:39	INSERT INTO Feedback (FeedbackID, CustomerID, ProductID, Rating, Comments, FeedbackDate) VALUES (4, 1, 1, 6, 'invalid r...	Error Code: 3819. Check constraint 'feedback_chk_1'... 0.0018 sec

TRANSACTIONAL SCENARIOS

4.1 Successful Transaction (COMMIT)

This is a successful order placement scenario when there is sufficient stock for the products ordered. This transaction involves adding a new order along with its details and decreasing the stock of the products purchased [8]. All operations being successful, the changes are committed, rendering the modifications irreversible.

```

160 ● START TRANSACTION;
161
162 -- 1. New order being inserted into OrderTable
163 ● INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus)
164 VALUES (6, '2024-12-27', 2200.00, 1, 'Pending');
165
166 -- 2. Updating OrderDetails table
167 ● INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice)
168 VALUES
169     (8, 6, 1, 2, 1100.00), -- 2 units of "ProductID = 1"
170     (9, 6, 5, 1, 80.00); -- 1 unit of "ProductID = 5"
171
172 -- 3: Deduct Stock Quantity Values
173 ● UPDATE Product
174 SET StockQuantity = StockQuantity - 2
175 WHERE ProductID = 1;
176
177 ● UPDATE Product
178 SET StockQuantity = StockQuantity - 1
179 WHERE ProductID = 5;
180
181 -- Step 4: Check if the stock quantity is positive
182 ● SELECT ProductID, StockQuantity
183 FROM Product
184 WHERE ProductID IN (1, 5);
185 -- In this Case , It is Positive
186
187 -- Step 5: COMMIT the transaction to save all changes as stock levels are sufficient
188 ● COMMIT;
189 -- Successful Transaction made and changes are now permanent. Rollback is no longer possible

```

10

Action Output			
	Time	Action	Response
1	20:17:38	START TRANSACTION	0 row(s) affected
2	20:17:38	INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus) VALUES (7, '2024-12-27', 5000.00, 1, ...)	1 row(s) affected
3	20:17:38	INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUES (10, 7, 1, 10, 1100.00), -- 10 un...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
4	20:17:38	UPDATE Product SET StockQuantity = StockQuantity - 10 WHERE ProductID = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
5	20:17:38	UPDATE Product SET StockQuantity = StockQuantity - 1 WHERE ProductID = 5	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

Now we let us check if there is available stock left

ProductID	StockQuantity
1	3
5	3
NULL	NULL

As we could see that there is availability after Deduction of the number of ordered units by customer for both products with Product ID 1 and 5, we can COMMIT the transaction to save the changed in our database.

Action Output			
	Time	Action	Response
1	20:27:37	COMMIT	0 row(s) affected

Now if we check the Stock quantity and other table we are able to see that changes have been saved to the database

```
191 SELECT * FROM OrderDetails WHERE OrderID = 6; -- Should return related order details
192 SELECT ProductID, StockQuantity FROM Product WHERE ProductID IN (1, 5);
193 -- Stock for ProductID = 1 should be reduced by 2, and ProductID = 5 by 1.
```

Result Grid						
Filter Rows: Search						
	OrderDetailID	OrderID	ProductID	Quantity	UnitPrice	Subtotal
	8	6	1	2	1100.00	2200.00
	9	6	5	1	80.00	80.00
	NULL	NULL	NULL	NULL	NULL	NULL
OrderDetails 21						

Result Grid	
Filter Rows: Search	
ProductID	StockQuantity
1	3
5	3
NULL	NULL
Product 22	

The transactional scenarios are started by the START TRANSACTION command that puts all related operations in a single logical unit. In the COMMIT scenario, a new order is inserted into the OrderTable, corresponding products and their quantities are inserted to the OrderDetails table, and then stock levels in the Product

table are updated for the ordered quantities. Then, the stock levels are verified to be sufficient. When all operations are successful and the stock is positive, the COMMIT command is executed to make the changes permanent.

4.2 Failed Transaction (ROLLBACK)

This is an example of an order placement being failed due to insufficient stock. When a customer tries to buy more quantities than what are available in the market, then the transaction is rolled back in order to maintain consistency so as to avoid saving any changes [8].

In the ROLLBACK scenario, a new order is similarly inserted, order details are added, but the quantities requested are greater than the available stock. When the stock levels are updated and verified, it finds that there is not sufficient stock for one or more products. Therefore, the ROLLBACK command is used to reverse all changes, so nothing incomplete or invalid is stored in the database.

```
198 ● START TRANSACTION;
199
200 -- Tries to add a new order and order details when quantities are more than available stock.
201
202 ● INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus)
203 VALUES (7, '2024-12-27', 5000.00, 1, 'Pending');
204
205
206 ● INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice)
207 VALUES
208     (10, 7, 1, 10, 1100.00), -- 10 units of ProductID 1
209     (11, 7, 5, 7, 80.00); -- 7 units of ProductID 5
210
211 -- Both Exceed Stock Quantity
212
213 -- This will describe the negative stock values that result from the simulation of stock deduction.
214 ● UPDATE Product
215 SET StockQuantity = StockQuantity - 10
216 WHERE ProductID = 1;
217
218 ● UPDATE Product
219 SET StockQuantity = StockQuantity - 7
220 WHERE ProductID = 5;
221
222 -- From the Product table, insufficient stock is identified.
223 ● SELECT ProductID, StockQuantity
224 FROM Product
225 WHERE StockQuantity < 0;
226
227 ● ROLLBACK;
```

Action Output			
	Time	Action	Response / Duration / Fetch Time
1	20:34:58	START TRANSACTION	0 row(s) affected 0.00090 sec
2	20:34:58	INSERT INTO OrderTable (OrderID, OrderDate, TotalAmount, CustomerID, OrderStatus) VALUES (7, '2024-12-27', 5000.00, 1, 'Pending')	1 row(s) affected 0.0011 sec
3	20:34:58	INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUES (10, 7, 1, 10, 1100.00), (11, 7, 5, 7, 80.00);	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0 0.00057 sec
4	20:34:58	UPDATE Product SET StockQuantity = StockQuantity - 10 WHERE ProductID = 1;	1 row(s) affected Rows matched: 1 Changed: 1 Warni... 0.00045 sec
5	20:34:58	UPDATE Product SET StockQuantity = StockQuantity - 7 WHERE ProductID = 5;	1 row(s) affected Rows matched: 1 Changed: 1 Warni... 0.00066 sec

Let us check If we have stock left in this scenario.

Result Grid		Filter Rows:	Search	Edit:
ProductID	StockQuanti...			
1	-5			
5	-3			
HULL	HULL			
Product 23				

As the stock appears to be in negative value, the changes should not be saved as there is not enough stock and, in this case, we would have to rollback.

Action Output		Time	Action	Response	Duration / Fetch Time
1	20:38:32	ROLLBACK		0 row(s) affected	0.00035 sec

Now if we check the Stock quantity and other tables, we are able to see that no changes were saved in this transaction

ProductID	StockQuanti...			
1	5			
5	4			
HULL	HULL			
OrderTable 24		OrderDetails 25	Product 26	

OrderDetailID	OrderID	ProductID	Quantity	UnitPrice	Subtotal	
HULL	HULL	HULL	HULL	HULL	HULL	
OrderTable 24		OrderDetails 25	Product 26			

OrderID	OrderDate	TotalAmount	CustomerID	OrderStatus	
NULL	NULL	NULL	NULL	NULL	
OrderTable 24		OrderDetails 25		Product 26	

PERFORMANCE TUNING ENHANCEMENTS

5.1 Important Query Requirements

From the business processes of AdventureWorks Cycles, several important queries that are likely to be executed frequently are evident. For example, customers have a need to look at their past orders, therefore, queries that fetch orders for specific customers should be highly prioritized. This is like stock availability, which is an important condition for timely order execution and requires checking product availability efficiently. Detail information for particular orders is also retrieved for order processing. Finally, customers and administrators are both able to browse products by category and analyse customer feedback based on ratings.

5.2 Data Queries without Indexing

The initial queries were executed without any secondary indexes to establish a baseline performance. However, due to full table scans, execution times were longer, most notably with larger tables.

5.2.1 Fetching All Orders for a Given Customer

This query is essential as customers often need to view their past orders.

```
237      -- PERFORMANCE QUERIES --
238
239      -- Data Queries without Indexing
240
241      -- Query 1: Get all orders for specific customer without indexing.
242
243 • SELECT *
244   FROM OrderTable
245  WHERE CustomerID = 1;
```

5.2.2 Checking Product Availability

It is important to have enough stock on hand before customers place their orders.

```
247      -- Query 2: Check product availability
248
249 • SELECT Name, StockQuantity
250   FROM Product
251  WHERE StockQuantity > 0;
```

5.3 Data Queries Indexing

After applying indexes, the execution times of queries were drastically reduced. This was because indexes enabled the database engine to quickly locate the relevant rows, thus preventing it from having to perform unnecessary scans [9].

```
253      -- Data Queries with Indexing
254
255 • CREATE INDEX idx_CustomerID ON OrderTable (CustomerID);
256 • CREATE INDEX idx_StockQuantity ON Product (StockQuantity);
257 • CREATE INDEX idx_OrderID ON OrderDetails (OrderID);
258 • CREATE INDEX idx_CategoryID ON Product (CategoryID);
259 • CREATE INDEX idx_Rating ON Feedback (Rating);
260
```

	Time	Action	Response	Duration / Fetch Time
1	22:45:54	CREATE INDEX idx_CustomerID ON OrderTable (CustomerID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.022 sec
2	22:45:54	CREATE INDEX idx_StockQuantity ON Product (StockQuantity)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.019 sec
3	22:45:54	CREATE INDEX idx_OrderID ON OrderDetails (OrderID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.011 sec
4	22:45:54	CREATE INDEX idx_CategoryID ON Product (CategoryID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.014 sec
5	22:45:54	CREATE INDEX idx_Rating ON Feedback (Rating)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.014 sec



5.3.1 Get order details for a particular order

This query is used to get detailed information about each order.

```

262  -- Query 3: Get order details for a particular order with indexing.
263
264  • SELECT OrderDetailID, ProductID, Quantity, Subtotal
265     FROM OrderDetails
266     WHERE OrderID = 4;
--

```

	OrderDetailID	ProductID	Quantity	Subtotal
	5	5	2	160.00
	NULL	NULL	NULL	NULL

	Time	Action	Response	Duration / Fetch Time
1	22:47:53	SELECT OrderDetailID, ProductID, Quantity, Subtotal FROM OrderDetails WHERE OrderID = 4 LIMIT 0, 1000	1 row(s) returned	0.0014 sec / 0.00002...

5.3.2 Retrieving Product based on Category

Allows filtering as Products are often browsed by customers and administrators by their categories.

```

268  -- Query 4: Retrieving Product based on Category with indexing.
269
270  • SELECT *
271     FROM Product
272     WHERE CategoryID = 2;

```

ProductID	Name	Description	Price	StockQuant...	Manufacturer...	CategoryID
2	TrailRider Pro	Durable mountain bike for UK trails.	1500.00	10	2	2

	Time	Action	Response	Duration / Fetch Time
1	22:48:42	SELECT * FROM Product WHERE CategoryID = 2 LIMIT 0, 1000	1 row(s) returned	0.00084 sec / 0.000...

5.3.3 Search Feedback by its Rating

This query is useful for finding highly rated products and understanding customer satisfaction.

```

269  -- Query 5: Search Feedback by Rating Without Indexing
270  • SELECT FeedbackID, CustomerID, ProductID, Rating, Comments
271  FROM Feedback
272  WHERE Rating = 5;
---
```

	Time	Action	Response	Duration / Fetch Time
1	22:49:07	SELECT FeedbackID, CustomerID, ProductID, Rating, Comments FROM Feedback WHERE Rating = 5 LIMIT 0, 1000	2 row(s) returned	0.0016 sec / 0.00000...

FeedbackID	CustomerID	ProductID	Rating	Comments
1	1	1	5	Fantastic bike for city rides!
3	3	4	5	EcoRide E-Bike is perfect for long commutes.
HULL	HULL	HULL	HULL	HULL

	Time	Action	Response	Duration / Fetch Time
1	22:49:07	SELECT FeedbackID, CustomerID, ProductID, Rating, Comments FROM Feedback WHERE Rating = 5 LIMIT 0, 1000	2 row(s) returned	0.0016 sec / 0.00000...

5.4 Secondary Indexing Rationale

To enhance query performance, certain columns in the tables were identified to be candidates for secondary indexing in view of how often they are used in queries and in filtering data. The CustomerID in the OrderTable was indexed to enhance the speed of retrieving orders by customer [9].

In checking product availability, indexing the StockQuantity column in the Product table was useful.

Because getting order details by OrderID needed an OrderID index in the OrderDetails table. This optimization helped in product browsing by category, indexing the CategoryID column in the Product table, and in filtering feedback by ratings, indexing the Rating column in the Feedback table.

These indexes improved data retrieval efficiency by preventing full table scans and decreasing query execution times.



REFERENCES

[1]

"Learn some best practices and tips for developing a database that meets the specific needs and expectations of your business stakeholders. Discover how to choose the right database model, design the database schema, implement the database, test the d," *Linkedin.com*, Dec. 25, 2023.
<https://www.linkedin.com/advice/0/what-best-ways-ensure-your-database-meets-1bgpc> (accessed Jan. 08, 2025).

[2]

A. Manning, *Databases for small business : essentials of database management, data analysis, and staff training for entrepreneurs and professionals*. [Berkeley, CA]: Apress, 2015.

[3]

LucidChart, "What is an Entity Relationship Diagram (ERD)?," *Lucidchart*, 2024.
<https://www.lucidchart.com/pages/er-diagrams>

[4]

A. Dearmer, "Database Schema Design Guide: Examples & Best Practices," *Integrate.io*, Jul. 16, 2021. <https://www.integrate.io/blog/complete-guide-to-database-schema-design-guide/>

[5]

C. Coronel and S. Morris, *Database systems: Design, implementation, and management*, 13th ed. Australia ; United States: Cengage Learning, 2019.

[6]

R. Peterson, "What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples," *Guru99.com*, Mar. 18, 2019. <https://www.guru99.com/database-normalization.html>

[7]



A. G. Taylor, *SQL for Dummies*. Hoboken, Nj: John Wiley & Sons, Inc, 2019.

[8]

E. Erkec, "Transactions in SQL Server for beginners," *SQL Shack - articles about database auditing, server performance, data recovery, and more*, Feb. 10, 2021. <https://www.sqlshack.com/transactions-in-sql-server-for-beginners/>

[9]



"How to optimize SQL Queries using Indexes," *Educative. Interactive Courses for Software Developers*. <https://www.educative.io/answers/how-to-optimize-sql-queries-using-indexes>

DATABASE DESIGN AND PROGRAMING

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

81 /100

PAGE 1

PAGE 2

 **Comment 1** |  Transferable Skills
Good start with clear ToC

PAGE 3

PAGE 4

 **Comment 2** |  Transferable Skills
introduction provides a clear explanation of the business problem, outlining AdventureWorks Cycles' need for a robust database system to support its online marketplace operations.

 **Comment 3** |  Transferable Skills
Insufficient Focus on Challenges

PAGE 5

 **Comment 4** |  Knowledge&Understanding Skills
The ERD includes most key entities, which are essential for the database design.

 **Comment 5** |  Knowledge&Understanding Skills
Underlining the primary key is the appropriate way to denote it in an ERD



Comment 6 | Knowledge&Understanding Skills

Eventhough most of entities and attributes adheres to 3rd Normal Form (3NF), relevant explanation seem to be missing

PAGE 6



Comment 7 | Knowledge&Understanding Skills

well designed DB schema

PAGE 7

PAGE 8



Comment 8 | Cognitive &Intellectual Skills

screenshots are neatly organized (e.g., each step clearly separated and labelled), they help maintain the report's readability and professionalism.

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18



Comment 9 | Practical Application Skills

Including examples where data is rejected due to constraint violations ensures data quality.

PAGE 19



Comment 10 | Practical Application Skills

Proper Transaction Handling



Comment 11 | **Practical Application Skills**

by demonstrating performance checks before and after indexing, you provide evidence that the indexing improves query performance



Comment 12 | **Transferable Skills**

conclusion will effectively summarize the key points of the report and is essential in every report



Comment 13 | **Literature Skills**

variety of relevant sources, such as books, academic papers, and online resources, will support the research and provide depth to the report



Comment 14 | **Literature Skills**

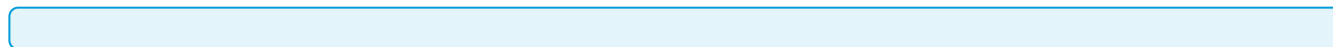
reference section well-organized with accurate citations

GRADING FORM: DAD_2024

SOHAAN SURESHKUMAR

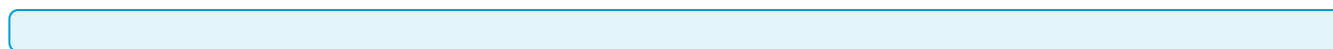
81

LITERATURE SKILLS



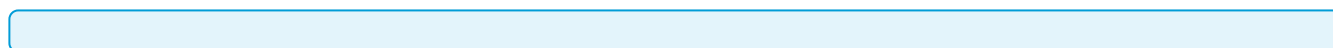
7

KNOWLEDGE&UNDERSTANDING SKILLS



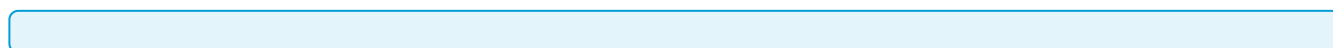
13

COGNITIVE &INTELLECTUAL SKILLS



17

PRACTICAL APPLICATION SKILLS



28.5

TRANSFERABLE SKILLS



15.5