Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

DBMS PROJECT

FOXCORE RETAIL DESIGNING A DATABASE

MAX 506 – DATABASE DESIGN AND SQL

PROFESSOR DR. DINAKAR JAYARAJAN

GROUP 2

ALMAS SAMALBAYEV

TATHAGAT PRATAP SINGH

MUTHUSELVI HARIHARAN

LOHIT REDDY ARRABOTHU

RAMPRAKASH REDDY GUDURU

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our instructor of this course, Dr. Dinakar Jayarajan who provided us with suggestions and for his guidance in both the classroom and in completing this project. We would like to thank him for making us do this project to sharpen our knowledge and skills on this material. Lastly, we would like to thank everyone who was involved directly or indirectly to help us prepare this project.

Sincerely,

Group 2.

TABLE OF CONTENT

1. COMPANY OVERVIEW:

Foxcore Retail is a small retail business started by Liam Corrigan and Mitchell Fox after graduating from the University of Western Ontario. They sell inexpensive novelty items like bubble guns, cooling towels, emoji pillows, heating pads, and remote-controlled drones. Their business model involves setting up pop-up shops at summer music festivals, tradeshows, rib fests, beer fests and other events across Ontario.

1.1 HISTORY:

Corrigan and Fox started the business inspired by Fox's previous job selling children's toys at events like the Calgary Stampede. They began by selling just two products - bubble guns and Arctic Skin cooling towels that Fox had relationships with suppliers for.

They chose the company name "Foxcore" combining their last names.

Initially they operated the booths themselves, later hiring sales consultants and paying them commissions. By the second year, they had expanded to managing up to 3 shows per weekend with multiple booths at some venues. The business proved more successful than they imagined initially.

1.2 OPERATIONS

They travel around Ontario attending various events and festivals to sell their products through pop-up booth setups. They hire sales consultants (17 full-time for 3rd season) who work shifts at the different booths to make sales. They had 28 different products available for sale by the 3rd season. Sales consultants tracked sales manually on paper which was used to calculate their commissions. So in summary, Foxcore Retail is a young, rapidly growing novelty retail business operating pop-up shops at events across Ontario, founded and managed by two entrepreneurial partners.

1.3 BUSINESS CHALLENGES

1. Inefficient Sales Tracking: Initially, sales were recorded manually on paper, which later needed to be transposed into Excel spreadsheets. This method proved inefficient and error-prone as the business scaled up.

2. Data Fragmentation: Over time, various disparate spreadsheets were created to store different types of business data such as event information, sales, employee commissions, and product details. Managing these spreadsheets became an administrative burden and led to data inconsistencies.

3. Lack of Strategic Insight: Despite having data on sales and operations, the inability to effectively access and analyze this data hindered strategic decision-making. This limitation affected both short-term operations and long-term planning.

4. Unreliable Inventory and Commission Calculations: The manual and disjointed data recording methods led to incomplete tracking of sales transactions. As a result, there were inaccuracies in inventory management and sales commissions, which demotivated staff and potentially led to financial discrepancies.

5. Urgency and Time Constraint: As the festival season approached, there was an urgent need to organize the data management processes. The lack of an integrated system meant that the partners were rapidly running out of time to implement a solution before the start of the season.

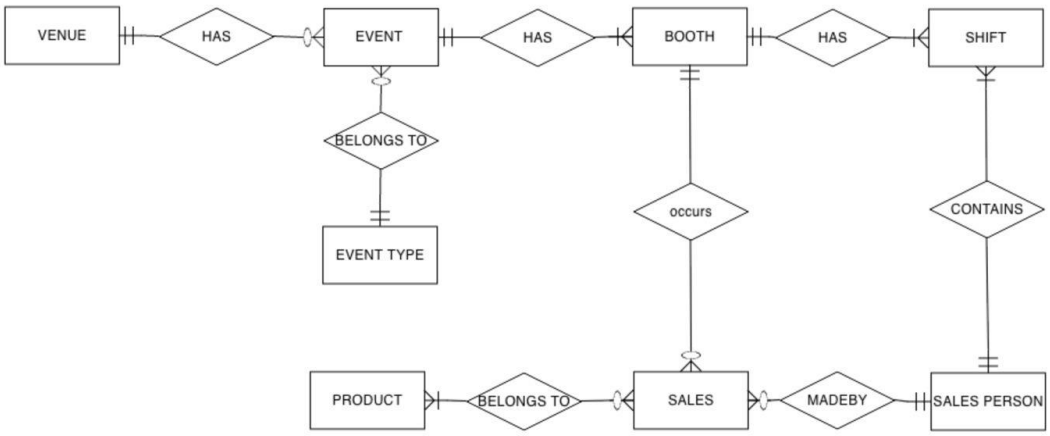
6. Technical Challenge in Database Design: Designing a new database system that could accommodate all necessary business functions was a major technical challenge, particularly under time constraints. The decision between quickly "hacking out" a database solution and taking the time to develop a well-thought-out conceptual data model posed a significant dilemma.

1.4 GOAL

To resolve the inefficiencies and inaccuracies stemming from Foxcore Retail's manual approach to sales tracking, a custom relational database system will be designed. This database system will facilitate efficient tracking of events, sales consultants, and individual product sales. By accessing and analyzing the data stored in this database, valuable insights can be obtained to drive improved decision-making for both short-term and long-term planning. With this database solution, Foxcore Retail will accurately record their sales data, identify top-selling products, and optimize their operations accordingly.

2. ER DIAGRAM

An Entity-Relationship (ER) diagram is a graphical representation used in database design to model the relationships between entities (objects or concepts) within a system. It is a conceptual modeling technique that helps visualize the structure of a database.



2.1 RELATIONSHIP CARDINALITY

This defines the numerical relationship between two entities in a relationship. It specifies the minimum and maximum number of instances of one entity that can be associated with instances of another entity.

The relationships that we defined from the ER Diagram are as follows:

Venue and Event:

A venue can host multiple events, but each event can only take place at one venue.

Event and Event Type:

An event can have only one event type, but each event type can be assigned to multiple events.

Event and Booth:

Each event should have at least one booth or can have multiple booths, but each booth can belong to only one event.

Booth and Shift:

Each booth should have at least one shift or can have multiple shifts, but each shift belongs to a single booth.

Booth and Sales:

A booth can have zero to many sales but sales can take place at one and only one booth.

Shift and Salesperson:

A shift can have one only one salesperson at a time, but each salesperson can have at least one to multiple shifts.

Sales and Product:

Each sales is associated to one to many product, but each product can have zero to multiple sales

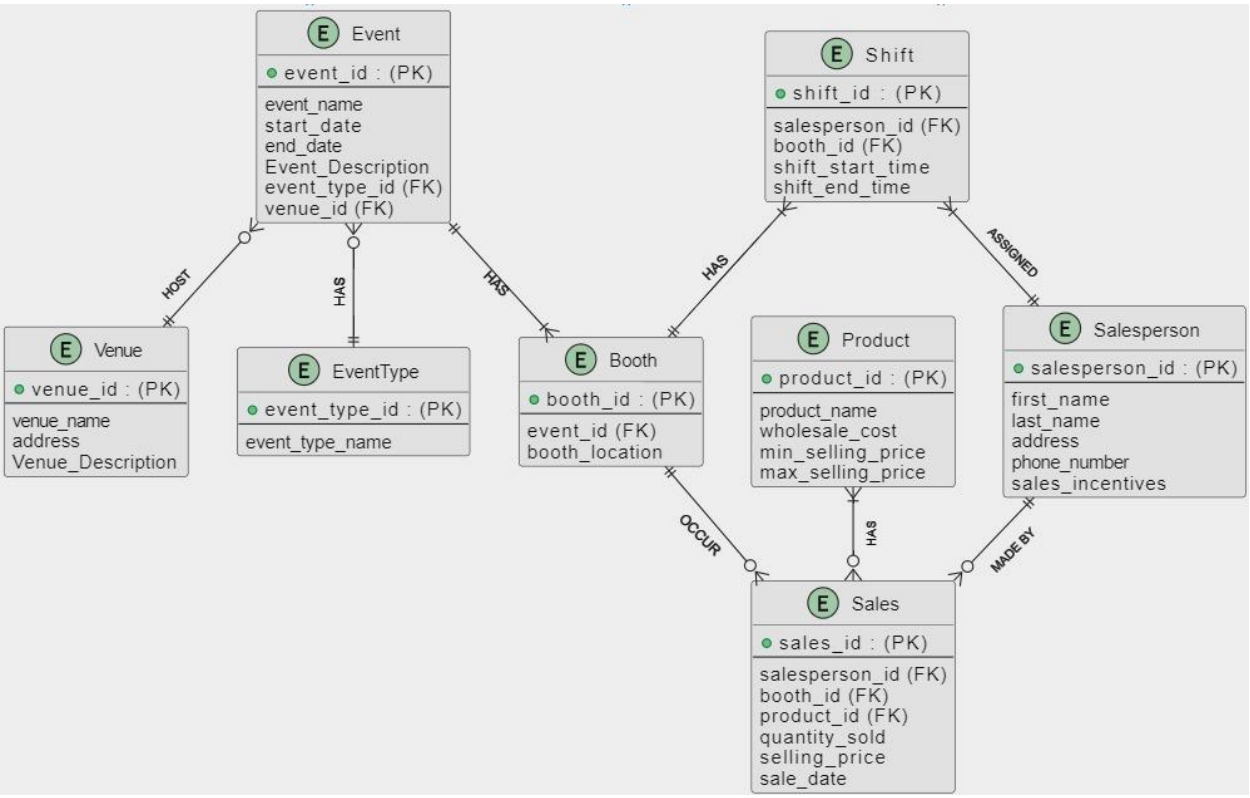
Sales and Salesperson:

Each sale is made by one salesperson, but each sales person can make zero to multiple sales.

3.FOXCORE RETAIL DB RELATIONAL SCHEMA

A relational schema is a set of relational tables and associated items that are related to one another. All of the base tables, views, indexes, domains, user roles, stored modules, and other items that a user creates to full fill the data needs of a particular enterprise or set of applications belong to one schema.

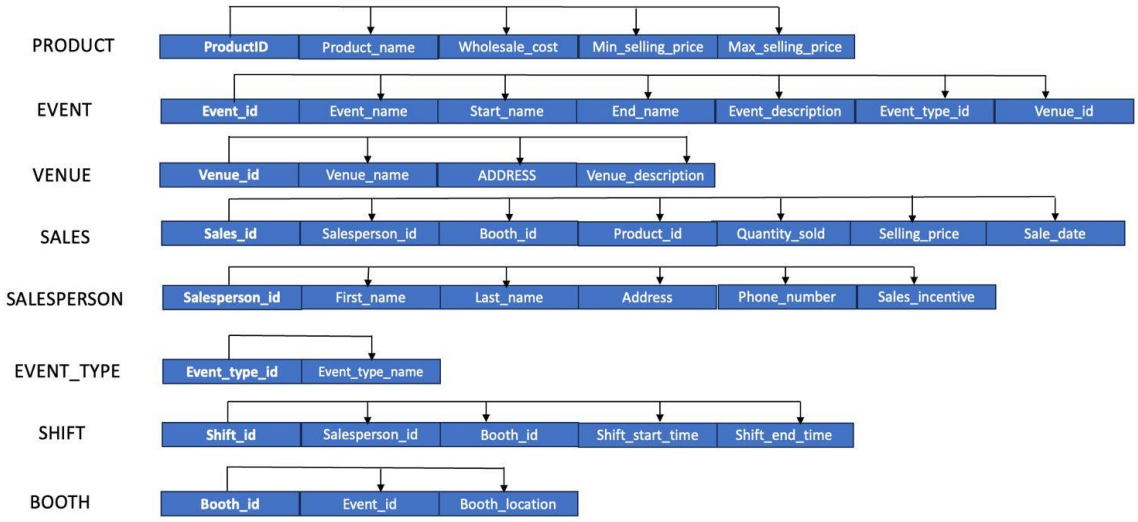
The schema plays a crucial role in ensuring data integrity, consistency, and efficiency in data storage and retrieval. It defines the rules and constraints that govern the data, such as primary keys, foreign keys, and other constraints like uniqueness or null values. The schema also establishes the relationships between tables, such as one-to-one, one-to-many, or many-to-many relationships, which are essential for representing real-world entities and their associations. By adhering to the schema, applications can store and retrieve data accurately and consistently, facilitating data analysis, reporting, and overall data management.



4. NORMALIZATION

Normalization is a systematic process of organizing data in a database to reduce redundancy (duplicate data) and improve data integrity. It involves breaking down a database into smaller tables and defining relationships between them according to certain rules or "normal forms".

The primary objective of normalization is to eliminate redundant data, which can lead to inconsistencies and update anomalies. By following the normal forms (1NF, 2NF, 3NF, and beyond), data is organized into logical, well-structured tables that represent entities and their relationships accurately. This process helps to prevent insertion, update, and deletion anomalies, which can compromise data integrity. Normalization also facilitates data maintenance, reduces storage requirements, and improves overall database performance by optimizing queries and minimizing data redundancy.



5. DATA ATTRIBUTES AND DATA TYPES

Entity	Attributes	Data Type
Event	EventID(PK)	Integer
	Event_Name	VARCHAR(50)
	Start_Date	DATE
	End_Date	DATE
	Event_Description	VARCHAR(100)
	EventTypeID(FK)	Integer
Event Type	EventTypeID(PK)	Integer
	TypeName	VARCHAR(50)
Venue	VenueID(PK)	Integer
	Venue Name	VARCHAR(50)
	Address	VARCHAR(100)
	Venue_Description	VARCHAR(100)
Booth	Booth_id(PK)	Integer
	Event_id(FK)	Integer
	Booth_Location	VARCHAR(100)
Shift	Shift_id(PK)	Integer
	Salesperson_id(FK)	Integer
	Booth_id(FK)	Integer
	Shift_Start_Time	DATETIME
	Shift_end_Time	DATETIME
Product	Product_id(PK)	Integer
	Product_Name	VARCHAR(50)
	Wholesale_cost	DECIMAL(10,2)
	Min_Selling_Price	DECIMAL(10,2)
	Max_Selling_Price	DECIMAL(10,2)
Sales	Sales_id(PK)	Integer
	Salesperson_id(FK)	Integer
	Booth_id(FK)	Integer
	Product_id(FK)	Integer
	Quantity_sold	Integer
	Selling_price	DECIMAL(10,2)
	Sale_Date	DATE
SalesPerson	Salesperson_id(PK)	Integer
	First_Name	VARCHAR(50)
	Last_Name	VARCHAR(50)
	Address	VARCHAR(100)
	Phone_Number	VARCHAR(15)
	Sales_Incentive	DECIMAL(10,2)

6. DATABASE CREATION: DDL-SQL COMMAND

DDL (Data Definition Language) commands are used to define, modify, and delete the structure of database objects like tables, indexes, and views. SQL (Structured Query Language) is the language used to communicate with relational databases

6.1 CREATE STATEMENTS

CREATE SCHEMA [FoxcoreDatabase| AUTHORIZATION FoxcoreOwner]

```
CREATE TABLE EventType (
    event_type_id INT PRIMARY KEY NOT NULL,
    event_type_name VARCHAR(50) NOT NULL
);
```

Field	Type
event_type_id	int
event_type_name	varchar(50)

```
CREATE TABLE Venue (
    venue_id INT PRIMARY KEY NOT NULL,
    venue_name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    Venue_Description VARCHAR(100) NOT NULL
);
```

Field	Type
venue_id	int
venue_name	varchar(50)
address	varchar(100)
Venue_Description	varchar(100)

```
CREATE TABLE Event (
    event_id INT PRIMARY KEY NOT NULL,
    event_name VARCHAR(50) NOT NULL,
```

```
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    Event_Description VARCHAR(100) NOT NULL,
    event_type_id INT NOT NULL,
    venue_id INT NOT NULL,
    FOREIGN KEY (event_type_id) REFERENCES EventType(event_type_id),
    FOREIGN KEY (venue_id) REFERENCES Venue(venue_id)
);
```

Field	Type
event_id	int
event_name	varchar(50)
start_date	date
end_date	date
Event_Description	varchar(100)
event_type_id	int
venue_id	int

```
CREATE TABLE Booth (
    booth_id INT PRIMARY KEY NOT NULL,
    event_id INT NOT NULL,
    booth_location VARCHAR(100) NOT NULL,
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);
```

Field	Type
booth_id	int
event_id	int
booth_location	varchar(100)

```
CREATE TABLE Product (
    product_id INT PRIMARY KEY NOT NULL,
    product_name VARCHAR(50) NOT NULL,
    wholesale_cost DECIMAL(10,2) NOT NULL,
    min_selling_price DECIMAL(10,2) NOT NULL,
    max_selling_price DECIMAL(10,2) NOT NULL
```

);

Field	Type
product_id	int
product_name	varchar(50)
wholesale_cost	decimal(10,2)
min_selling_price	decimal(10,2)
max_selling_price	decimal(10,2)

```
CREATE TABLE Salesperson (
    salesperson_id INT PRIMARY KEY NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50),
    address VARCHAR(100) NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    sales_incentives DECIMAL(10,2) NOT NULL
);
```

Field	Type
salesperson_id	int
first_name	varchar(50)
last_name	varchar(50)
address	varchar(100)
phone_number	varchar(15)
sales_incentives	decimal(10,2)

```
CREATE TABLE Shift (
    shift_id INT PRIMARY KEY NOT NULL,
    salesperson_id INT NOT NULL,
    booth_id INT NOT NULL,
    shift_start_time DATETIME NOT NULL,
    shift_end_time DATETIME NOT NULL,
    FOREIGN KEY (salesperson_id) REFERENCES Salesperson(salesperson_id),
    FOREIGN KEY (booth_id) REFERENCES Booth(booth_id)
);
```

Field	Type
shift_id	int
salesperson_id	int
booth_id	int
shift_start_time	datetime
shift_end_time	datetime

```
CREATE TABLE Sales (
    sales_id INT PRIMARY KEY NOT NULL,
    salesperson_id INT NOT NULL,
    booth_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity_sold INT NOT NULL,
    selling_price DECIMAL(10,2) NOT NULL,
    sale_date DATE NOT NULL,
    FOREIGN KEY (salesperson_id) REFERENCES Salesperson(salesperson_id),
    FOREIGN KEY (booth_id) REFERENCES Booth(booth_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
```

Field	Type
sales_id	int
salesperson_id	int
booth_id	int
product_id	int
quantity_sold	int
selling_price	decimal(10,2)
sale_date	date

7. DCL COMMANDS

DCL (Data Control Language) commands are a subset of SQL statements used to manage access privileges and control over objects in a database management system (DBMS).

1. Grant
2. Revoke
3. Commit
4. Roll back
5. Privileges

These DCL commands are essential for managing access control, ensuring data security, and implementing role-based access control in a database environment. They allow database administrators to grant, revoke, and manage privileges effectively, ensuring that only authorized users can perform specific operations on database objects.

//**Table Name: Event**

GRANT ALL PRIVILEGES

ON Event

TO Fox;

GRANT ALL PRIVILEGES

ON Event

TO Corrigan;

//**Table Name: Venue**

GRANT ALL PRIVILEGES

ON Venue

TO Fox;

GRANT ALL PRIVILEGES

ON Venue

TO Corrigan;

//**Table Name: EventType**

GRANT ALL PRIVILEGES

ON EventType

TO Fox;

GRANT ALL PRIVILEGES

ON EventType

TO Corrigan;

//**Table Name: Booth**

GRANT ALL PRIVILEGES

ON Booth

TO Fox;

GRANT ALL PRIVILEGES

ON Booth

TO Corrigan;

//**Table Name: Shift**

GRANT ALL PRIVILEGES

ON Shift

TO Fox;

GRANT ALL PRIVILEGES

ON Shift

TO Corrigan;

//**Table Name: Product**

GRANT ALL PRIVILEGES

ON Product

TO Fox;

GRANT ALL PRIVILEGES

ON Product

TO Corrigan;

//**Table Name: Sales**

GRANT ALL PRIVILEGES

ON Sales

TO Fox;

GRANT ALL PRIVILEGES

ON Sales

TO Corrigan;

//**Table Name: Salesperson**

GRANT ALL PRIVILEGES

ON Salesperson

TO Fox;

GRANT ALL PRIVILEGES

ON Salesperson

TO Corrigan;

The main important functions of the DCL Commands are the security of the database.
Security Purpose:
(1) The database should be password protected.
(2) Each member of staff should be assigned database access privileges appropriate to a particular user view, namely Director, Manager, Supervisor, or Assistant.
(3) Staff should only see the data necessary to do his or her job in a form that suits what they are doing.
(4) Use the principle of least privilege, granting users only the minimum permissions required for their tasks.

8. DML

We have used the SQL INSERT, UPDATE, and DELETE commands to populate and modify the current database. The following are the sample statements that could be used to help Foxcore Retail ensure its database is up-to-date and kept clear of unnecessary or outdated information.

The following are the sample statements that could be used to help Foxcore Retail ensure its database is up-to-date and kept clear of unnecessary or outdated information**.**

8.1. INSERT STATEMENTS TO POPULATE THE DATABASE

1. How to insert the new data into EventType table?

INSERT INTO EventType (event_type_id, event_type_name) VALUES

(1, 'Trade Show'),

(2, 'Rib Fest'),

(3, 'Waterfront Festival'),

(4, 'Music Festival'),

(5, 'Beer Festival'),

(6, 'Sporting Event'),

(7, 'Street Festival');

event_type_...	event_type_name
1	Trade Show
2	Rib Fest
3	Waterfront Festival
4	Music Festival
5	Beer Festival
6	Sporting Event
7	Street Festival

2.How to insert new data into Venue table?

INSERT INTO Venue (venue_id, venue_name, address, Venue_Description) VALUES

(1, 'Venue 1', 'Address 1', 'Description 1'),

(2, 'Venue 2', 'Address 2', 'Description 2'),

(3, 'Venue 3', 'Address 3', 'Description 3');

venue_id	venue_name	address	Venue_Descripti...
1	Venue 1	Address 1	Description 1
2	Venue 2	Address 2	Description 2
3	Venue 3	Address 3	Description 3

3.How to add new events into Event table?

INSERT INTO Event (event_id, event_name, start_date, end_date, Event_Description, event_type_id, venue_id) VALUES

(1, 'Event 1', '2024-05-01', '2024-05-05', 'Event Description 1', 1, 1),

(2, 'Event 2', '2024-06-10', '2024-06-15', 'Event Description 2', 4, 2),

(3, 'Event 3', '2024-07-20', '2024-07-25', 'Event Description 3', 7, 3);

event_id	event_name	start_date	end_date	Event_Description	event_type_...	venue_id
1	Event 1	2024-05-01	2024-05-05	Event Description 1	1	1
2	Event 2	2024-06-10	2024-06-15	Event Description 2	4	2
3	Event 3	2024-07-20	2024-07-25	Event Description 3	7	3

4. How to add new booths in Booth table?

INSERT INTO Booth (booth_id, event_id, booth_location) VALUES

(1, 1, 'Location 1'),

(2, 2, 'Location 2'),

(3, 3, 'Location 3');

booth_id	event_id	booth_locati...
1	1	Location 1
2	2	Location 2
3	3	Location 3

5. To insert new rows in a Product table?

INSERT INTO Product (product_id, product_name, wholesale_cost, min_selling_price, max_selling_price) VALUES

(1, 'Product 1', 10.00, 15.00, 25.00),

(2, 'Product 2', 12.00, 18.00, 30.00),

(3, 'Product 3', 8.00, 12.00, 20.00);

product_...	product_na...	wholesale_cost	min_selling_pri...	max_selling_pri...
1	Product 1	10.00	15.00	25.00
2	Product 2	12.00	18.00	30.00
3	Product 3	8.00	12.00	20.00

6. How to new data of salesperson ?

INSERT INTO Salesperson (salesperson_id, first_name, last_name, address, phone_number, sales_incentives) VALUES

(1, 'John', 'Doe', 'Address 1', '123-456-7890', 0.05),

(2, 'Jane', 'Smith', 'Address 2', '456-789-0123', 0.07),

(3, 'Alice', 'Johnson', 'Address 3', '789-012-3456', 0.08);

salesperson_...	first_name	last_name	address	phone_number	sales_incentiv...
1	John	Doe	Address 1	123-456-7890	0.05
2	Jane	Smith	Address 2	456-789-0123	0.07
3	Alice	Johnson	Address 3	789-012-3456	0.08

7. How to insert new information into shifttable?

INSERT INTO Shift (shift_id, salesperson_id, booth_id, shift_start_time, shift_end_time) VALUES

(1, 1, 1, '2024-05-01 08:00:00', '2024-05-01 17:00:00'),

(2, 2, 2, '2024-06-10 09:00:00', '2024-06-10 18:00:00'),

(3, 3, 3, '2024-07-20 10:00:00', '2024-07-20 20:00:00');

shift_id	salesperson_...	booth_id	shift_start_time	shift_end_time
1	1	1	2024-05-01 08:00:00	2024-05-01 17:00:00
2	2	2	2024-06-10 09:00:00	2024-06-10 18:00:00
3	3	3	2024-07-20 10:00:00	2024-07-20 20:00:00

8. How to add new sales into the sales table?

INSERT INTO Sales (sales_id, salesperson_id, booth_id, product_id, quantity_sold, selling_price, sale_date) VALUES

(1, 1, 1, 1, 20, 20.00, '2024-05-01'),

(2, 2, 2, 2, 15, 25.00, '2024-06-10'),

(3, 3, 3, 3, 10, 15.00, '2024-07-20');

sales_id	salesperson_...	booth_id	product_...	quantity_so...	selling_pri...	sale_date
1	1	1	1	20	20.00	2024-05-01
2	2	2	2	15	25.00	2024-06-10
3	3	3	3	10	15.00	2024-07-20

8.2. UPDATE STATEMENTS TO MODIFY EXISTING DATA:

1.Update the information of Event_Descrpition of first event?

UPDATE Event SET Event_Description = 'New Event Description' WHERE event_id = 1;

event_id	event_name	start_date	end_date	Event_Description	event_type_...	venue_id
1	Event 1	2024-05-01	2024-05-05	New Event Description	1	1
2	Event 2	2024-06-10	2024-06-15	Event Description 2	4	2
3	Event 3	2024-07-20	2024-07-25	Event Description 3	7	3

2.How to update the min_selling_price for Product as 20 where product_id=2?

UPDATE Product SET min_selling_price = 20.00 WHERE product_id = 2

product_...	product_na...	wholesale_cost	min_selling_pri...	max_selling_pri...
1	Product 1	10.00	15.00	25.00
2	Product 2	12.00	20.00	30.00
3	Product 3	8.00	12.00	20.00

3. How to update the sales_incentives for Salesperson with ID=3?

UPDATE Salesperson SET sales_incentives = 0.10 WHERE salesperson_id = 3;

salesperson_...	first_name	last_name	address	phone_number	sales_incentiv...
1	John	Doe	Address 1	123-456-7890	0.05
2	Jane	Smith	Address 2	456-789-0123	0.07
3	Alice	Johnson	Address 3	789-012-3456	0.10

8.3. DELETE STATEMENTS TO REMOVE DATA FROM THE DATABASE

1. How to delete the shift id=3?

SET FOREIGN_KEY_CHECKS=1;

DELETE FROM Shift WHERE shift_id = 3;

SET FOREIGN_KEY_CHECKS=0;

shift_id	salesperson_...	booth_id	shift_start_time	shift_end_time
1	1	1	2024-05-01 08:00:00	2024-05-01 17:00:00
2	2	2	2024-06-10 09:00:00	2024-06-10 18:00:00

2.How to delete the salesperson with salesperson_id=2?

SET FOREIGN_KEY_CHECKS=1;

DELETE FROM Salesperson WHERE salesperson_id = 2;

SET FOREIGN_KEY_CHECKS=0;

salesperson_...	first_name	last_name	address	phone_number	sales_incentiv...
1	John	Doe	Address 1	123-456-7890	0.05
3	Alice	Johnson	Address 3	789-012-3456	0.10

3. How to delete the sales records for event_id=1?

SET FOREIGN_KEY_CHECKS=1;

DELETE FROM Sales WHERE booth_id IN (SELECT booth_id FROM Booth WHERE event_id = 1);

SET FOREIGN_KEY_CHECKS=0;

sales_id	salesperson_...	booth_id	product_...	quantity_so...	selling_pri...	sale_date
2	2	2	2	15	25.00	2024-06-10
3	3	3	3	10	15.00	2024-07-20

8.4. SQL STATEMENTS REQUIRED TO PREPARE THE REPORTS DESCRIBED IN THE CASE

1.Total Sales by Event Type and Date Range:

SELECT et.event_type_name, e.event_name, b.booth_id, SUM(s.selling_price * s.quantity_sold) AS Total_sales

FROM event e

JOIN booth b ON b.event_id = e.event_id

JOIN sales s ON s.booth_id = b.booth_id

JOIN eventtype et ON et.event_type_id = e.event_type_id

WHERE e.start_date >= '2024-01-01' AND e.end_date <= '2024-12-12'

GROUP BY et.event_type_name, e.event_name, b.booth_id;

event_type_name	event_name	booth_id	Total_sales
Music Festival	Event 2	2	375.00
Street Festival	Event 3	3	150.00

2: Report Sales of each Salesperson?

SELECTconcat(sp.first_name,",sp.last_name)AS Salesperson_name,sum(s.quantity_sold*s.selling_price) as total_sales from sales s

JOIN salesperson sp on sp.salesperson_id = s.salesperson_id

GROUP BY Salesperson_name;

Salesperson_name	total_sales
Alice Johnson	150.00

3: Show the Sales by each Product.

SELECT Product.product_name, SUM(Sales.quantity_sold * Sales.selling_price) AS total_sales

FROM Product

JOIN Sales ON Product.product_id = Sales.product_id

GROUP BY Product.product_name;

product_na...	total_sales
Product 2	375.00
Product 3	150.00

4.Report the Sales by Event

SELECT Event.event_name, SUM(Sales.quantity_sold * Sales.selling_price) AS total_sales

FROM Event

JOIN Booth ON Event.event_id = Booth.event_id

JOIN Sales ON Booth.booth_id = Sales.booth_id

GROUP BY Event.event_name;

event_name	total_sales
Event 2	375.00
Event 3	150.00

9. CONCLUSION

The implementation of a robust database system at Foxcore Retail has revolutionized its operational capabilities. Transitioning from manual sales tracking to an automated, structured database has significantly enhanced data accuracy and operational efficiency. Moreover, the system has empowered the company with valuable business intelligence, enabling in-depth analysis of sales performance, inventory turnover, and customer preferences.

This insight is essential for informed decision-making and forecasting future trends. Furthermore, the scalability and adaptability of the database ensure seamless management of growing data volumes and complex business operations as the company expands. In essence, the database system has played a crucial role in streamlining operations, facilitating sophisticated data analysis, and supporting Foxcore Retail's strategic initiatives for growth and market adaptability.

SOFTWARE USED

1. ERD Diagram: ERD Plus
2. DB RELATIONAL SCHEMA: Draw.io
3. DBMS: MYSQL WORKBOOK
4. Data types and Data Attributes: Excel