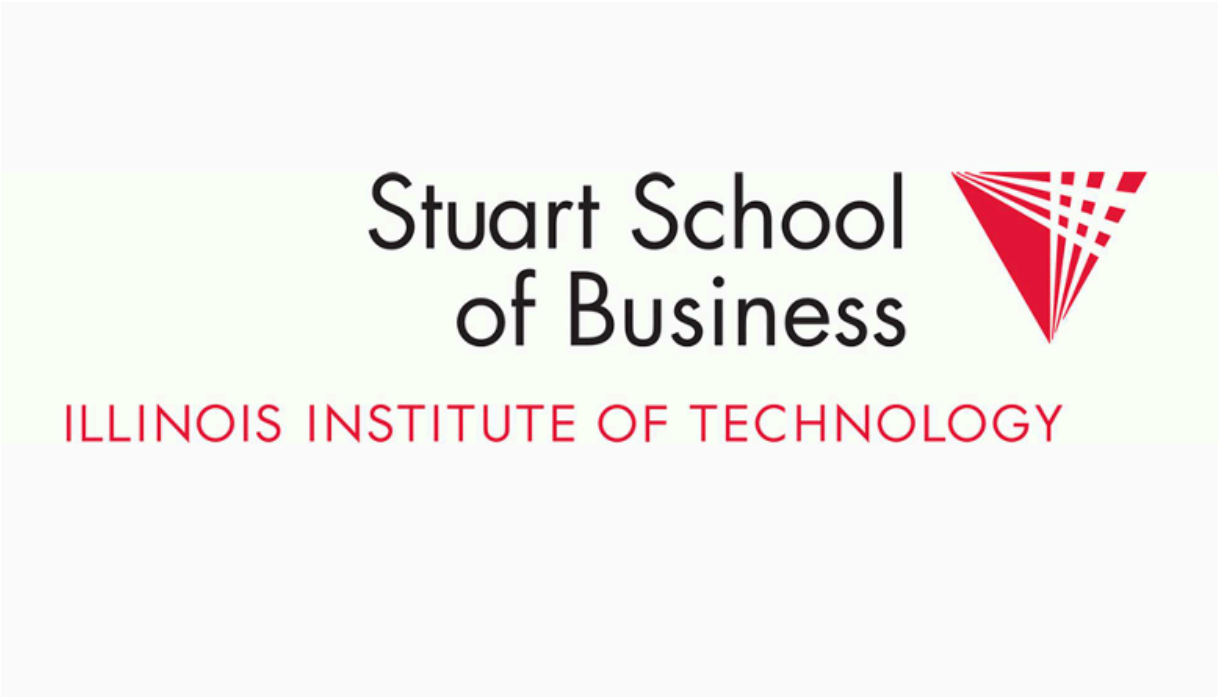


Project Report

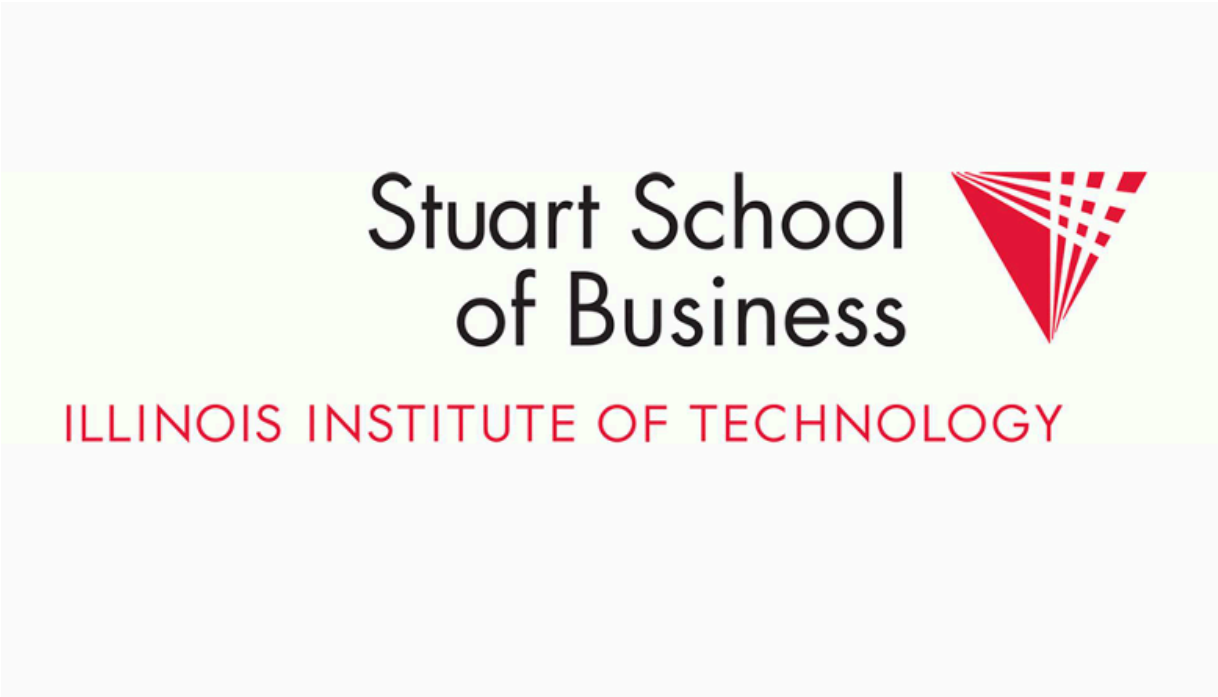
Foxcore Retail - Designing A Database

Database Design and SQL

Dr. Dinakar Jayarajan

Submitted by - Group 6

Ishitha Nanda Kumar

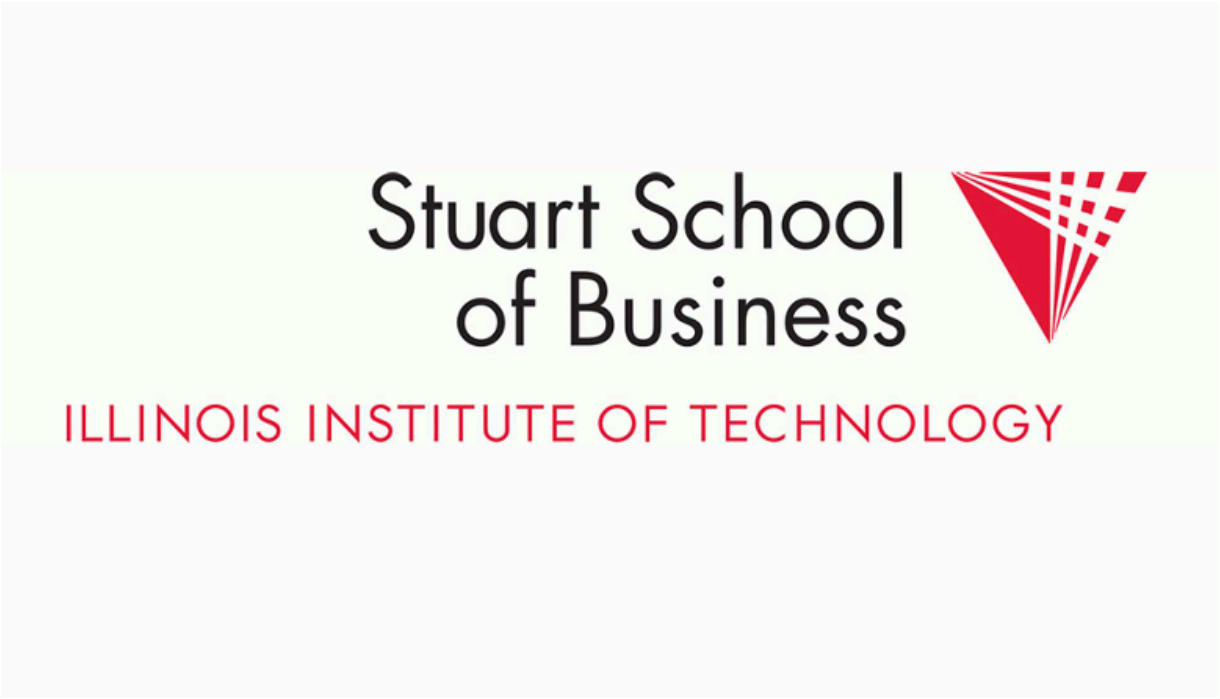


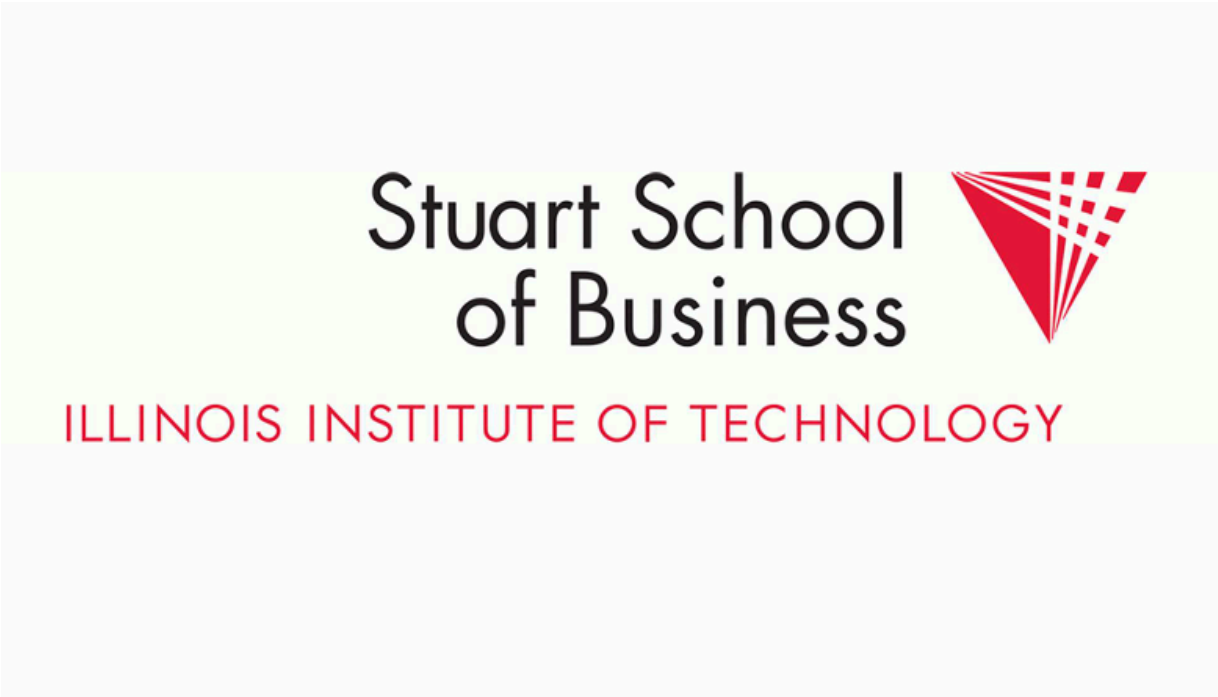
Acknowledgment

We convey our earnest appreciation to the academic staff at the Stuart School of Business for their insightful decision to include the Database Management System (DBMS) course within the curriculum of our Master's program. This educational strategy has equipped us with the intellectual tools to confront and resolve contemporary industrial challenges.

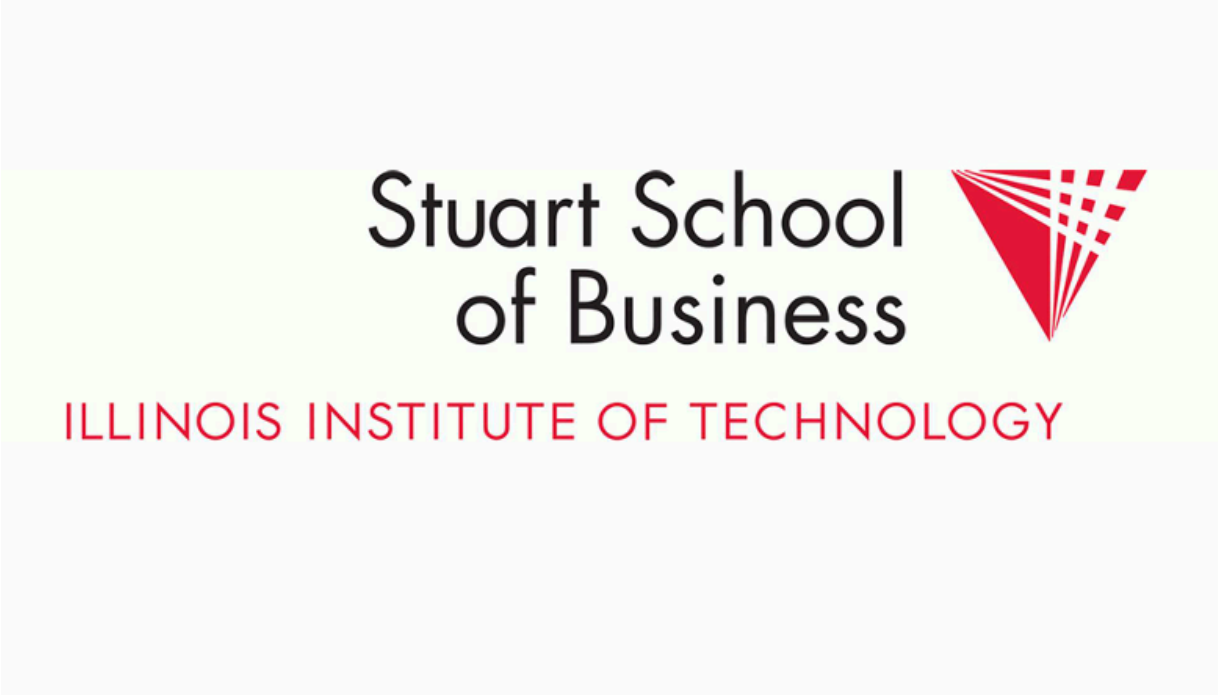
Our heartfelt thanks go to Professor Dinakar Jayarajan, who entrusted us with this project. His consistent mentorship and valuable advice were crucial in our ability to adhere to the project timeline and objectives.

In closing, we acknowledge the collective efforts of our peers in this endeavor. Our shared commitment and collaborative work ethic were the cornerstones of turning this venture into a triumphant achievement.


Stuart School
of Business
ILLINOIS INSTITUTE OF TECHNOLOGY



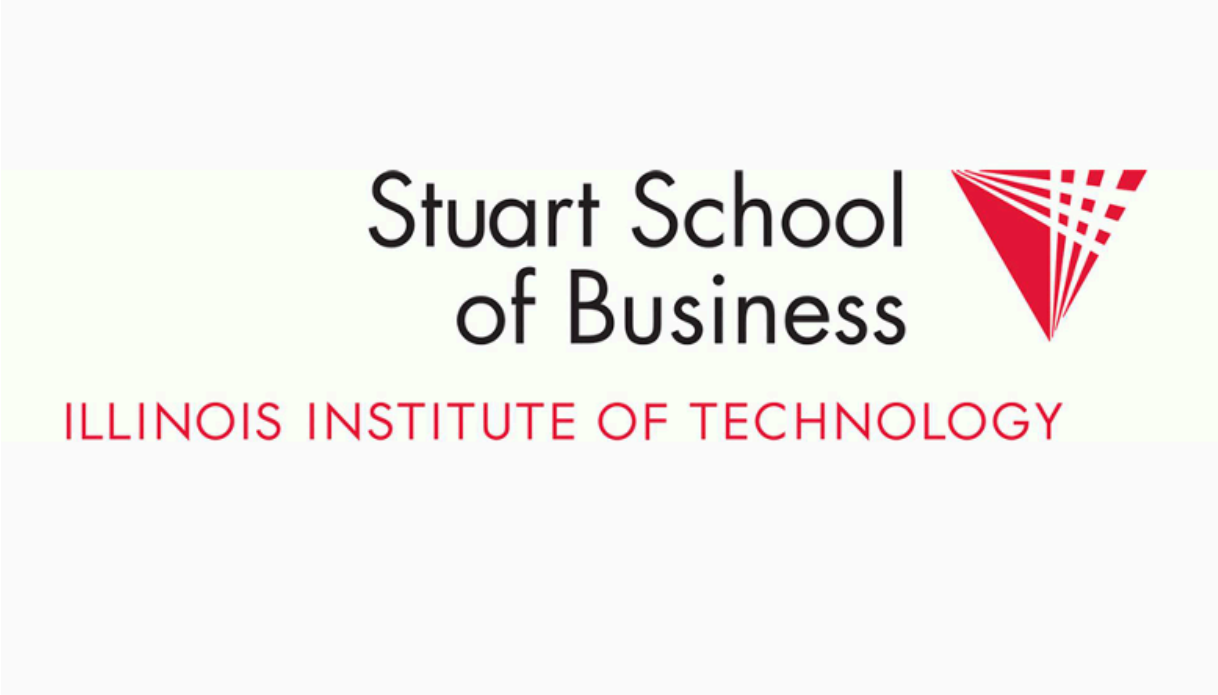
Contents



Executive Summary

Foxcore Retail, a burgeoning retail enterprise renowned for its novelty items sold at music festivals and trade shows, is poised to undergo a transformational shift in its operational management. The impetus for the Foxcore Retail database project is the design and deployment of a cutting-edge, scalable database system, crafted to refine and enrich the company's burgeoning operational demands efficiently. This strategic initiative arises from the critical need to supersede the current rudimentary manual and spreadsheet-dependent data management practices—methods that have fallen short in supporting the expanding scale of operations, from an increase in event engagements to a diversified product range and an amplified workforce.

The envisioned database will be an amalgamation of key features, including a sophisticated entity-relationship model that ensures structured data normalization, automated processing of transactions, immediate data access in real-time, and bolstered measures for data integrity and security. The introduction of this database is a deliberate technological investment to bolster Foxcore Retail's ongoing commitment to



operational superiority and to carve a competitive niche in the retail sector.

Upon implementation, the database is anticipated to catalyze significant enhancements in data precision, facilitating meticulous monitoring of sales, inventory control, and employee performance metrics. It will serve as the bedrock for strategic decision-making, yielding prompt and actionable insights into the heartbeat of business operations and thus refining resource distribution while augmenting customer service quality. Designed with scalability at its core, the system is ready to accommodate future business growth and integration seamlessly. The amalgamation of these aspects heralds not only an elevation in day-to-day operational efficiency but also fortifies Foxcore Retail's capacity to flourish and maintain a competitive edge in a dynamic market landscape.

Business Problem

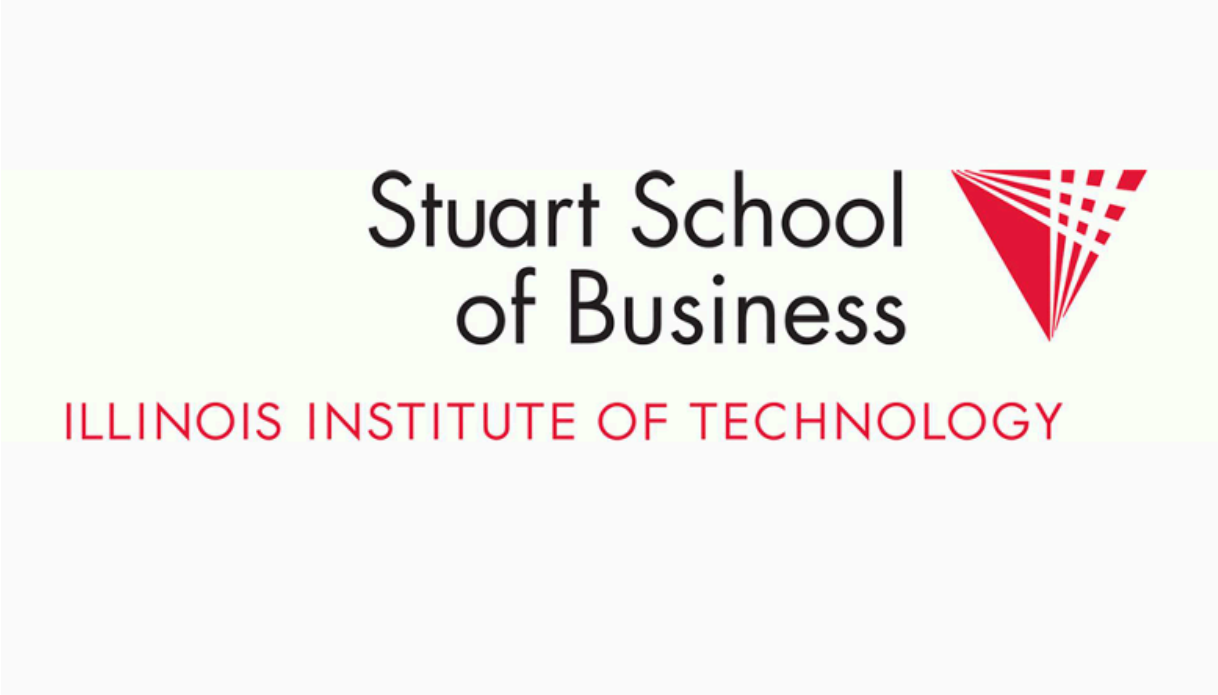
Foxcore Retail is confronted with a critical inefficiency in its data management processes. The absence of a unified system for data collection has precipitated

suboptimal decision-making practices within the company. Current reliance on fragmented spreadsheets and manual record-keeping has engendered considerable issues, including the forfeiture of sales commissions and the generation of unreliable inventory assessments. As the festival season looms, the necessity for a bespoke database becomes ever more pressing to meticulously track product-specific sales data, ensuring the integrity and accuracy of business operations.

Objectives

The paramount objective for this project is the creation of a tailor-made database system designed with the express purpose of enabling Foxcore Retail to meticulously monitor sales and inventory in real-time. This system will serve as a cornerstone for:

- Enhancing the precision of sales tracking and inventory management, thereby curtailing the loss of revenues and ensuring accurate stock evaluations.
- Providing a robust framework upon which Foxcore can anchor data-driven decision-making, laying the groundwork for strategic planning and operational



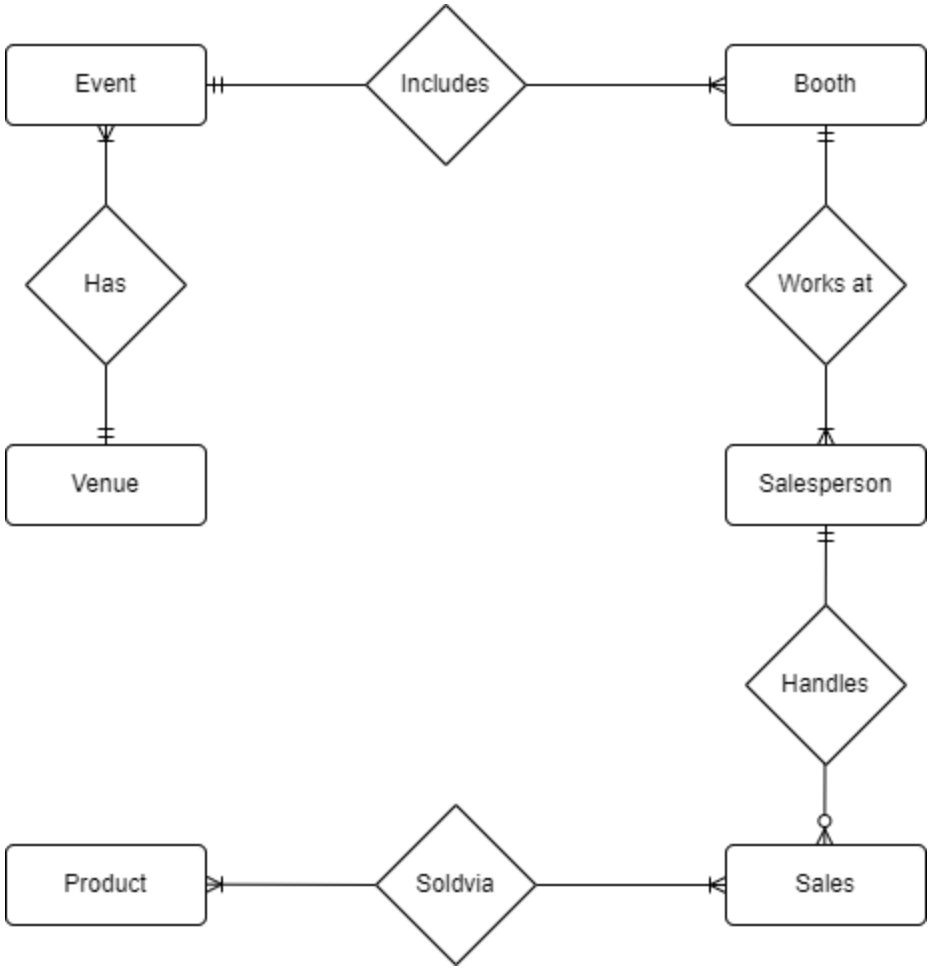
excellence.

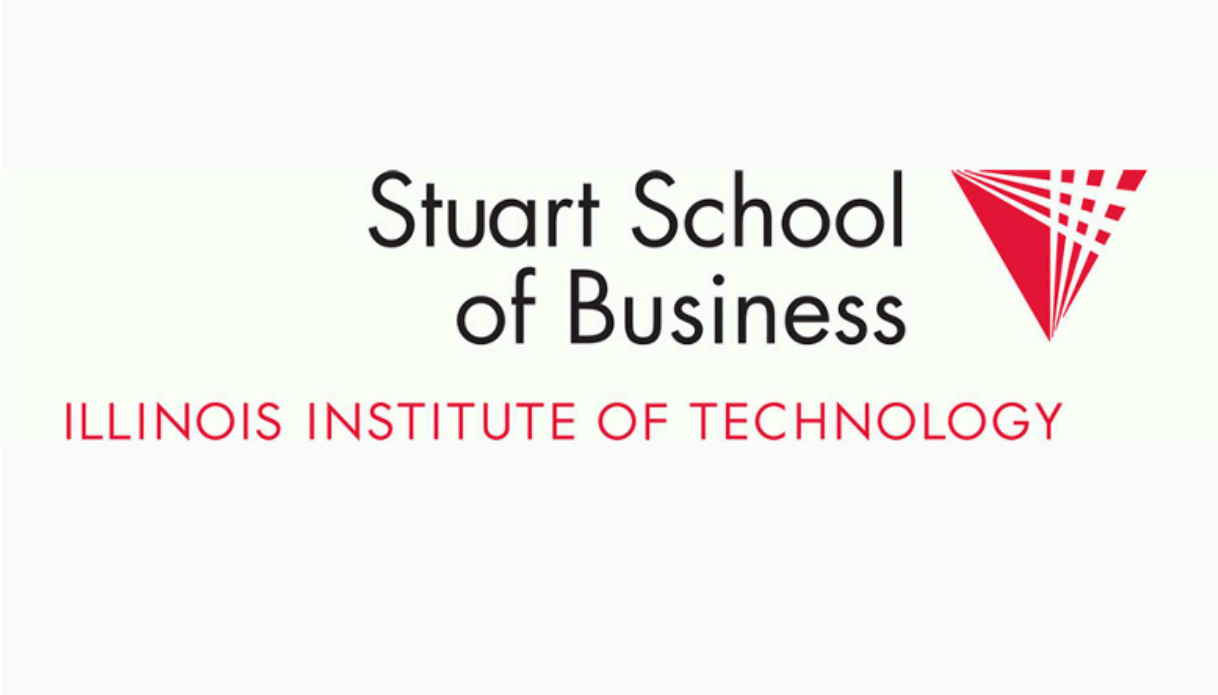
ER Diagram

The ER diagram, crafted in Barker's notation, provides a clear blueprint of the Foxcore Retail database, detailing entities and their relationships to support operational needs efficiently.

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY





Event Entity:

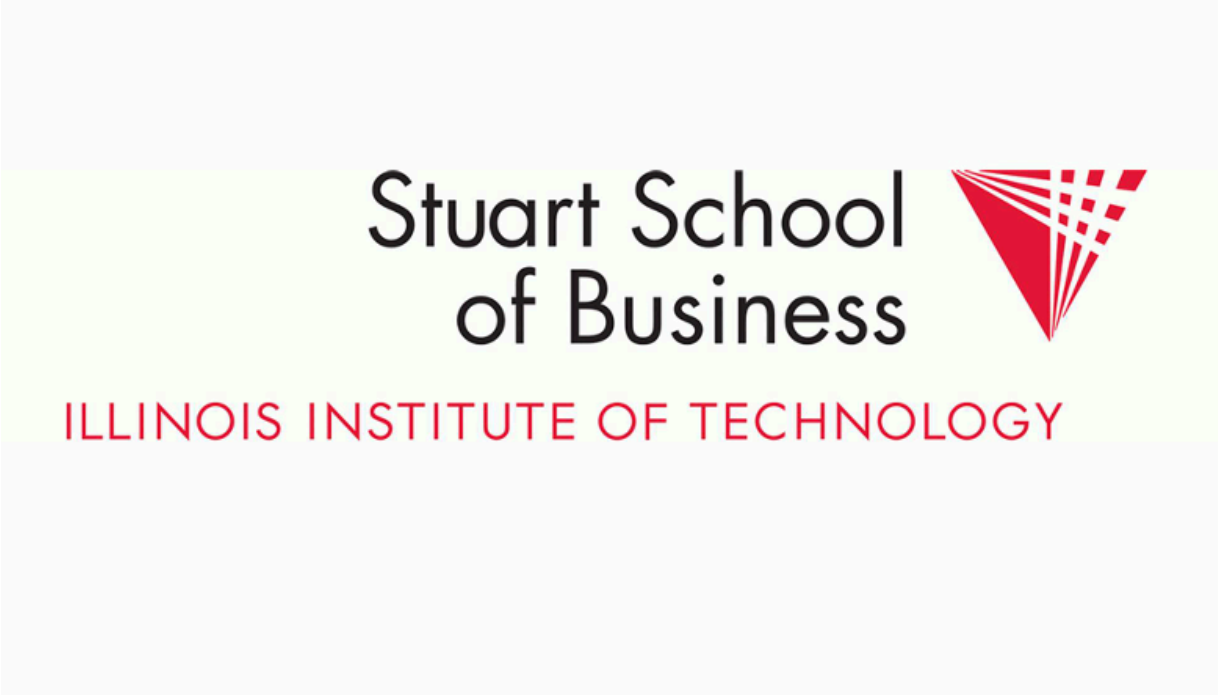
- Captures the details of different events where Foxcore Retail participates, such as music festivals and beer festivals.
- Each event has a unique identifier and is associated with specific venues where it takes place.

Venue Entity:

- Represents the physical location of each event.
- This entity includes details about the venue which are crucial for logistical and operational planning for events.

Booth Entity:

- Each event includes one or more booths manned by Foxcore Retail.
- The booth is the sales point at events and is key to tracking sales and inventory at specific events.



Salesperson Entity:

- Details about Foxcore Retail employees working at the booths.
- This entity is vital for managing staff schedules, tracking sales performance, and calculating commissions.

Sales Entity:

- This is a transactional entity that records all sales activities.
- It links products, booths, and salespersons, thereby tracking which products are sold, where, and by whom.

Product Entity:

- Contains all information about the products sold by Foxcore Retail.

- It is essential for inventory management and sales tracking.

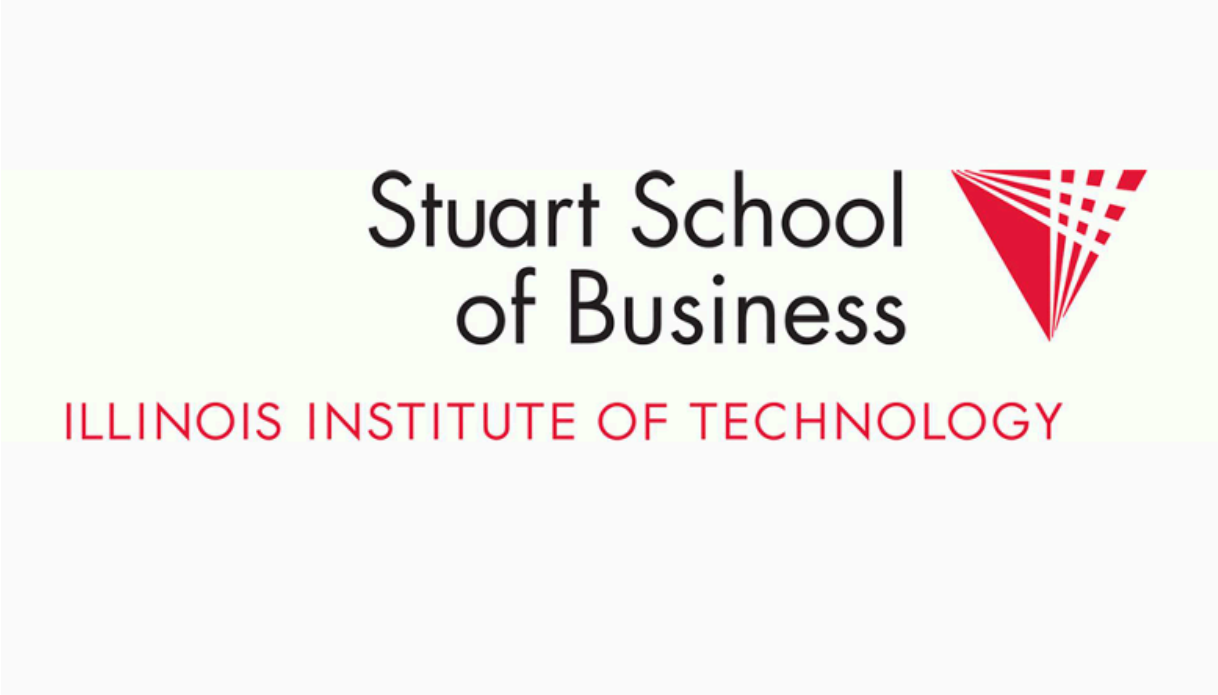
Relationships:

The ER diagram defines the relationships between these entities, including:

1. Events "include" Booths.
2. Each Booth "works at" by a Salesperson.
3. Salespersons "handles" Sales.
4. Products are "sold via" Sales.

No Assumptions:

- The ER diagram is developed based on the explicit requirements and information provided in the case study.
- There have been no assumptions made beyond the scope defined in the case, ensuring that the ER diagram accurately reflects the actual data relationships for Foxcore Retail.



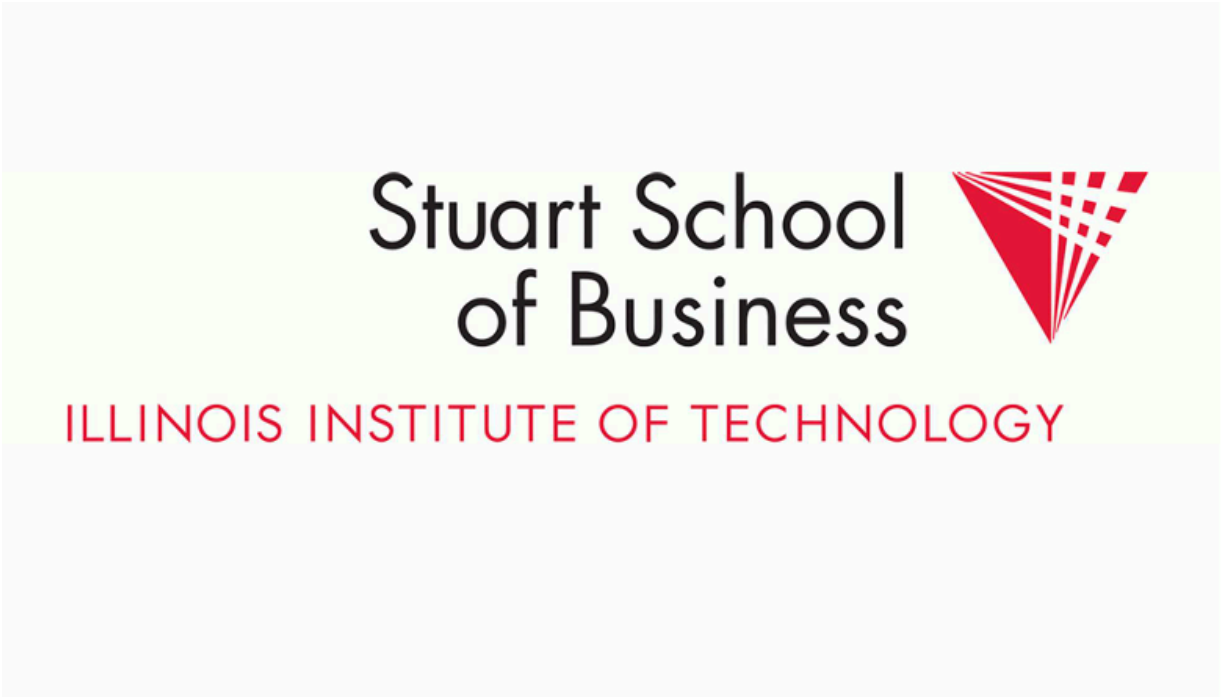
Normalization

Normalization in database design is the process of organizing data to minimize redundancy and improve data integrity. It involves structuring a database in a way that each data element is stored in the appropriate place and is accessible in the most efficient and unambiguous manner. The goal is to reduce and even eliminate data anomalies, ease data management, and enhance the scalability and flexibility of the database structure.

Second Normal Form (2NF):

The Second Normal Form (2NF) is a step in the database normalization process that builds upon the first normal form (1NF). A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the entire primary key. This means that there are no partial dependencies of any column on the primary key. Ensuring a table is in 2NF eliminates redundancy that can occur when there are partial dependencies on composite primary keys.

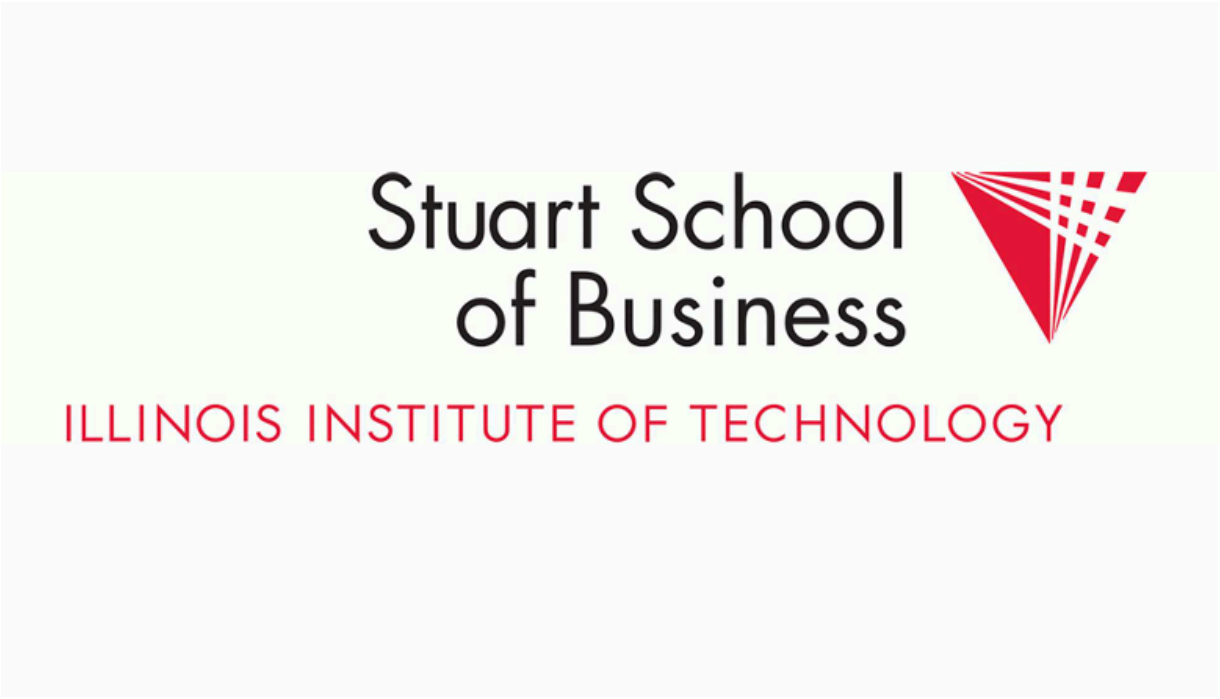
In the development of the normalization table for the Foxcore Retail database, our team has efficiently advanced the data structure to the Second Normal Form (2NF). Through this progression, we've achieved a layout where each non-prime attribute is fully functionally dependent on the entirety of its primary key, eliminating partial

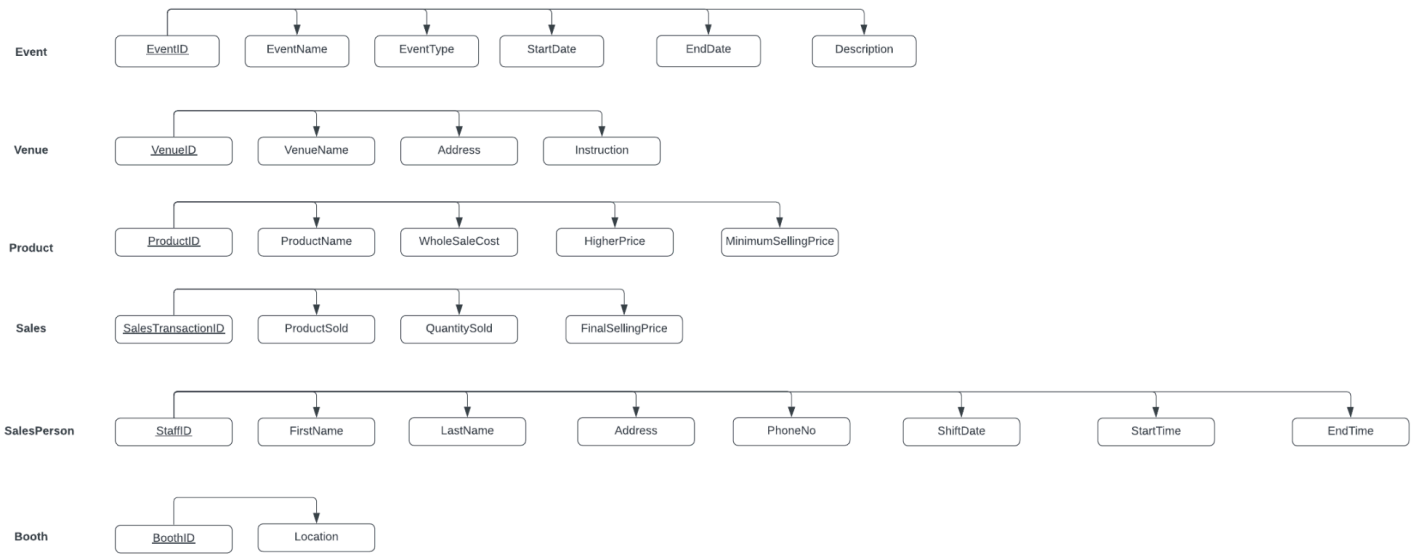


dependencies.

One noteworthy advancement in this normalization phase was the creation of the 'Shift' entity. This new entity, though not elaborated upon in detail within the scope of this report, is an essential construct for potential future analysis. It will enable the Foxcore Retail management team to associate time frames with sales activity, which may prove beneficial for various operational and strategic applications.

By instituting the 'Shift' entity, we lay a foundation that could serve future explorations into employee scheduling, sales trends, and operational efficiency, without complicating the current structure and analysis requirements.
This proactive approach ensures our database architecture is not only optimized for today's needs but is also well-prepared for tomorrow's growth and exploratory data analysis.





Third Normal Form (3NF):

The Third Normal Form (3NF) is achieved when a table is in 2NF and all its columns are not only fully functionally dependent on the primary key but are also non-transitively dependent. That is, there should be no dependency of one non-prime attribute on any other non-prime attribute. A table is in 3NF if all the data in the table is directly related to the primary key, and non-key information is dependent only on primary key information. This form further eliminates redundancy and potential for anomalies in data

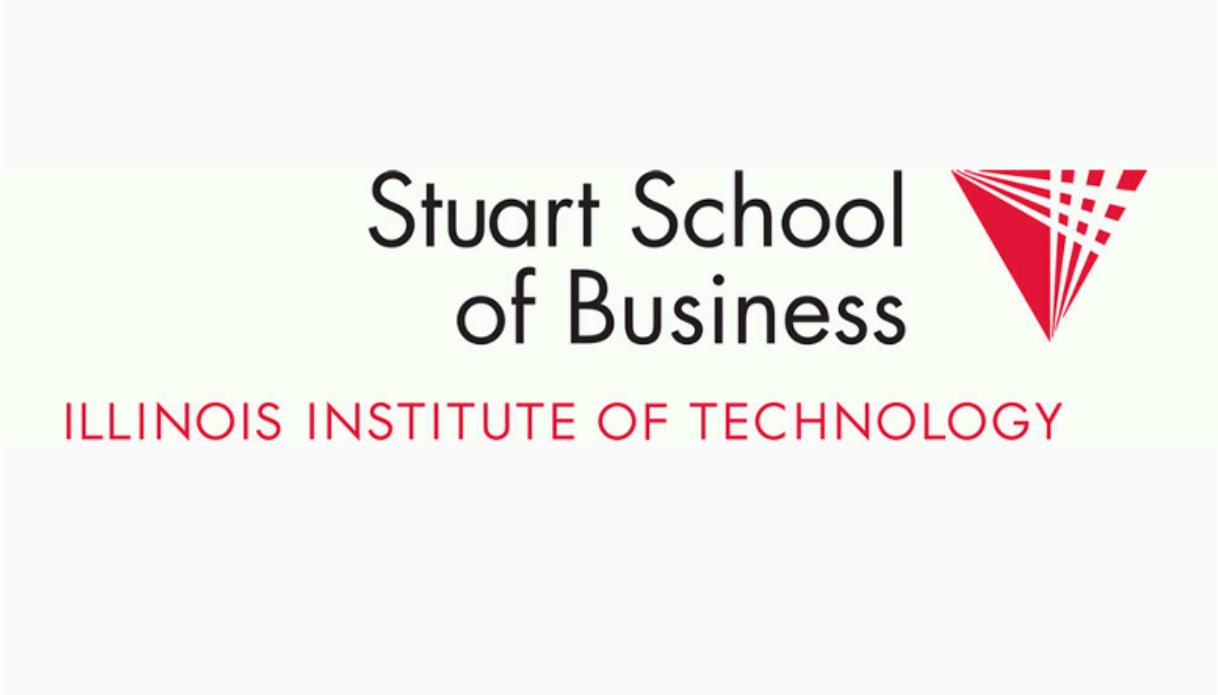
manipulation.
Each table within our database has been designed to be in 3NF, ensuring that:

- It already meets all the criteria of the Second Normal Form (2NF).
- All non-key attributes are non-transitively dependent on the primary key, thereby eliminating any indirect dependencies within our tables.

This level of normalization helps us prevent data duplication and maintain consistency across the database. For instance, in the 'SalesPerson' table, the relationship to the 'Shift' entity through 'ShiftID' indicates a design where shift patterns are independent of other non-prime attributes, allowing for clear, direct reporting on salesperson shifts without redundant or unnecessary linkage to other attributes.

Adhering to 3NF, our database tables now ensure that every non-key attribute is directly dependent on only the key. This design philosophy enables Foxcore Retail to maintain streamlined and uncluttered data management practices, paving the way for reliable data retrieval and updates which are critical for real-time operations.

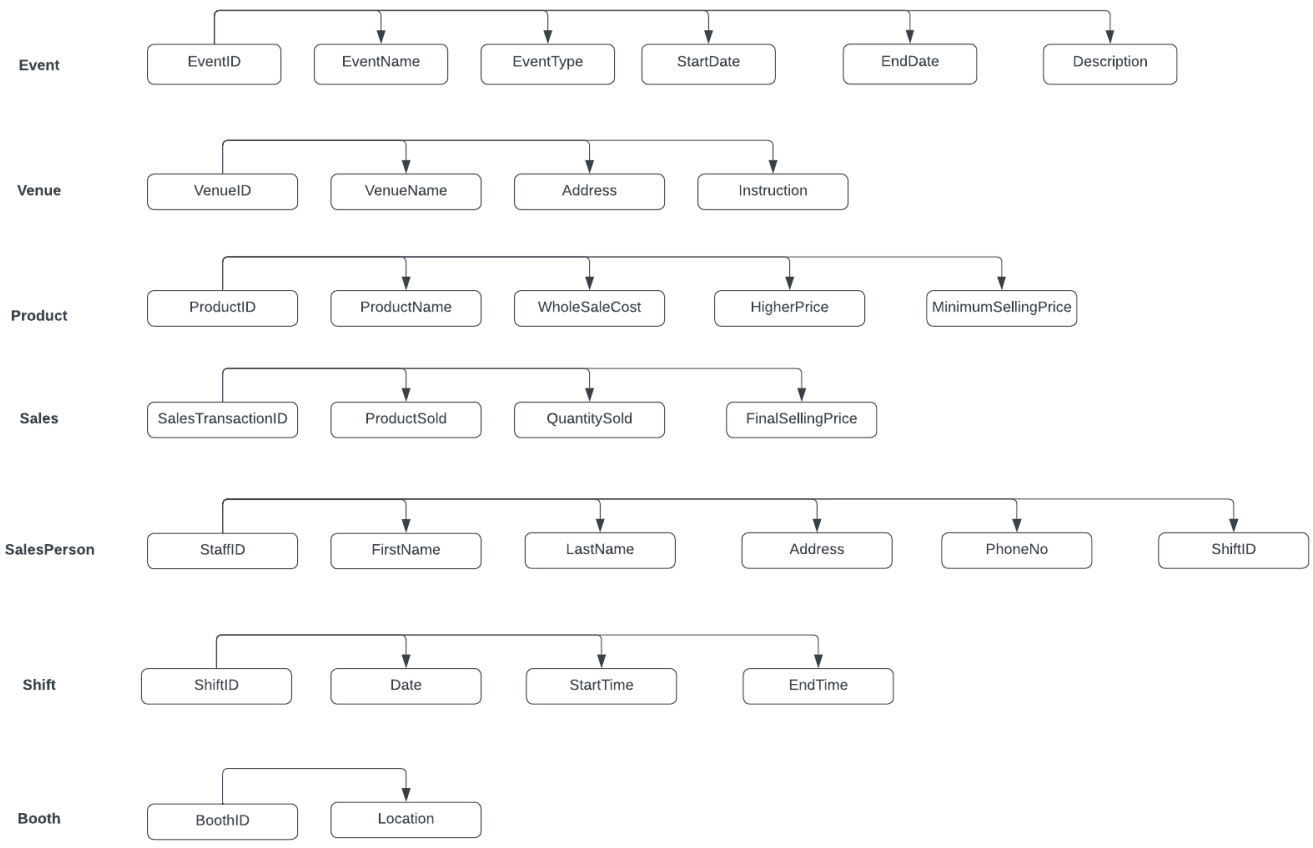
The structured approach of 3NF in our database will not only improve the performance



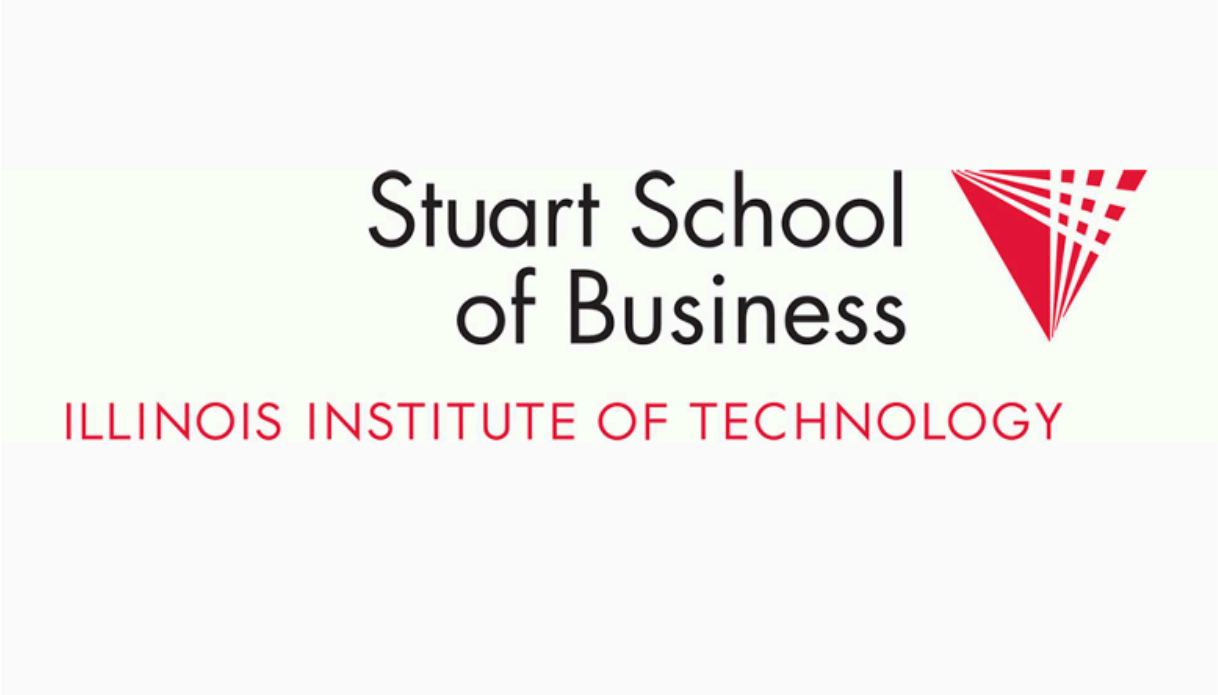
but will also simplify maintenance, making it easier to extend the database structure in the future without compromising data quality. The 3NF structure thus positions Foxcore Retail's database as a robust and agile asset for both current and future operational challenges.

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY



Third Normal Form (3NF)

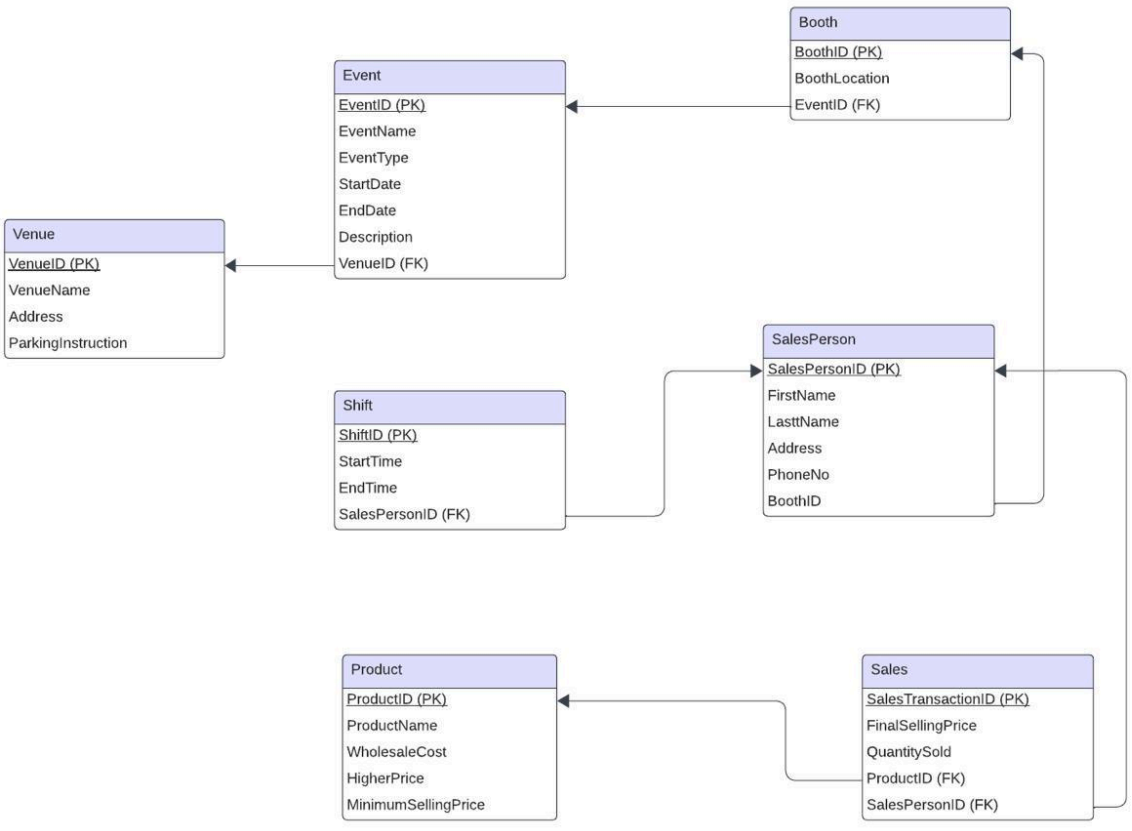


Relational Schema

A relational schema is the blueprint for a relational database, defining its tables, fields, and the relationships between them. It serves as the framework for how data is organized and interlinked, ensuring data integrity and reflecting the connections between real-world entities.

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY



The relational schema depicted provides a comprehensive view of the database

architecture for Foxcore Retail. It illustrates the methodical organization of data across various tables, each representing a unique entity such as Venue, Event, Booth, SalesPerson, Shift, Product, and Sales. These entities are linked through clearly defined relationships, signified by primary and foreign keys.

- The Venue entity is uniquely identified by a VenueID and includes pertinent details such as name, address, and parking instructions.
- Events are tied to venues through a foreign key relationship, with each event having its distinct characteristics like type, start and end dates, and a descriptive overview.
- Booths are associated with events, detailing their location within the event's scope.
- The SalesPerson entity is cross-referenced with both Booth and Shift entities, providing a relational thread that tracks where and when salespersons are working.
- The Shift entity is delineated by its own ID, capturing the times salespersons are active.

- Products are cataloged with information on cost and pricing.
- Finally, the Sales entity captures the transactions, linking products and salespersons, thereby allowing for a detailed account of sales activities. P

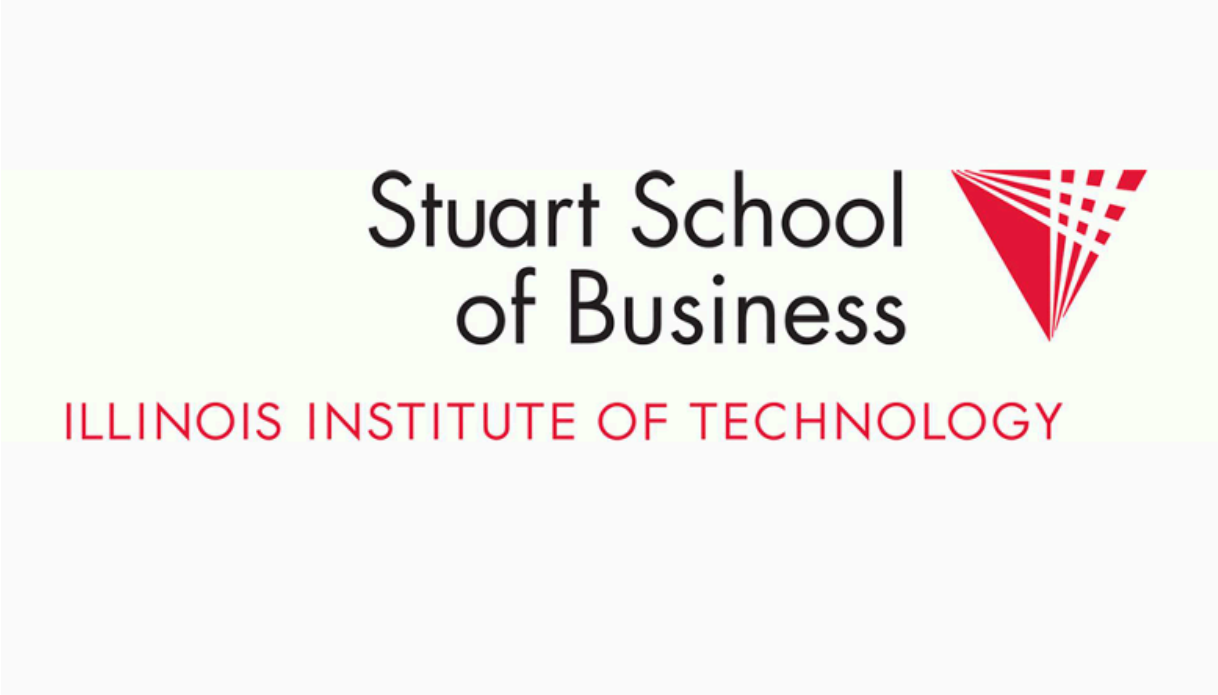
This schema enables Foxcore Retail to maintain an organized and efficient database, with clear pathways for querying, reporting, and analyzing data across the company's operational spectrum. The design facilitates precise tracking of sales performance, inventory management, and employee allocation, setting a strong foundation for data-driven decision-making.

Entity Name	Description	Occurrence
Venue	Represents physical locations where events are held.	One record per venue.
Event	An occurrence where Foxcore Retail participates, such as festivals or shows.	One record per event.

Booth	A sales point within an event managed by Foxcore Retail.	Multiple booths may occur per event.
SalesPerson	An employee who works at a booth and handles sales.	One record per salesperson.
Shift	The working time period of a salesperson.	Multiple shifts can occur per salesperson, one per shift period.
Product	An item sold by Foxcore Retail.	One record per unique product.
Sales	A record of transaction details of products sold.	One record per sales transaction.

DDL and DML commands

In database management, Data Definition Language (DDL) and Data Manipulation Language (DML) are two categories of SQL commands crucial for database structuring and data handling. DDL commands structure database objects through:

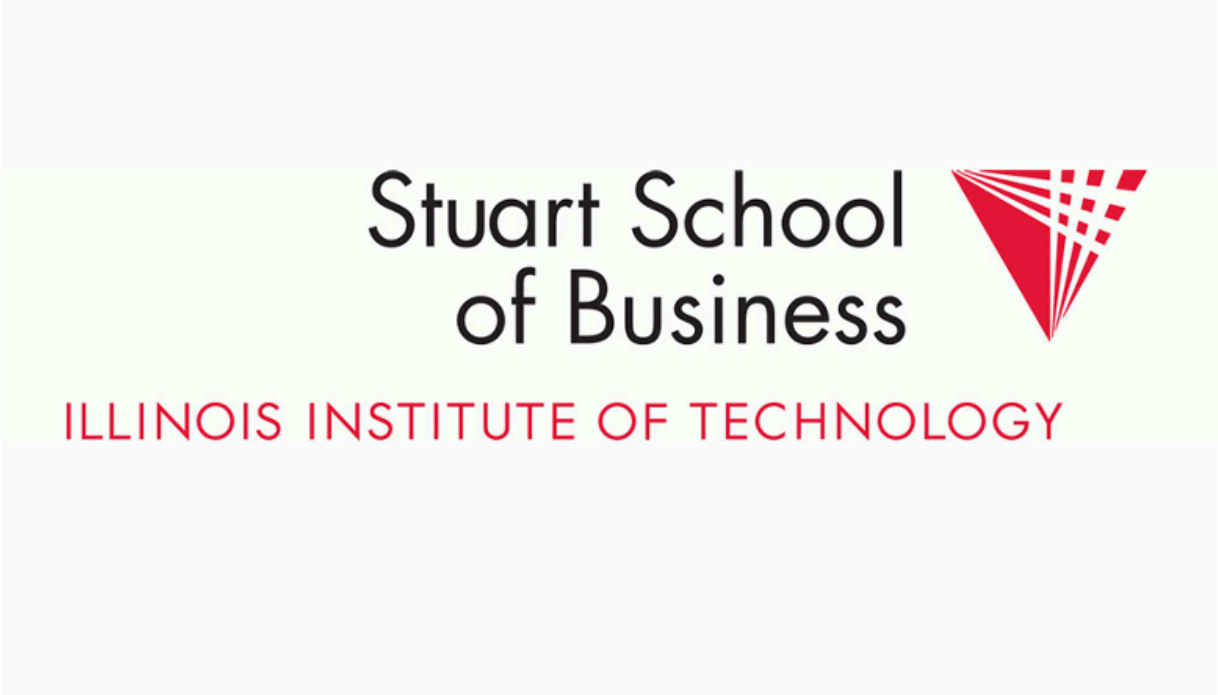


- CREATE TABLE table_name (column1 datatype, column2 datatype, ...); to define new tables.
- ALTER TABLE table_name ADD column_name datatype; to modify tables.
- DROP TABLE table_name; to remove tables.
- DML commands manage the data within these objects with:
- INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...); for data insertion.
- UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition; for updating data.
- DELETE FROM table_name WHERE condition; for data deletion.
- SELECT column1, column2, ... FROM table_name WHERE condition; for data retrieval.

We used the MySQL Workbench to run our commands to build our database and perform the required actions.

1. **Venue table DDL**

```
CREATE TABLE Venue (
```



VenueID INTEGER NOT NULL ,

VenueName VARCHAR(25) NOT NULL,

Address VARCHAR(30) NOT NULL ,

ParkingInstruction TEXT,

CONSTRAINT Venue_PK PRIMARY KEY (VenueID));

Venue table DML

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)

VALUES (124, 'Maple Leaf Banquet Hall', '100 King St W, Toronto, ON', 'Underground parking available; entrance on King St.');

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)

VALUES (125, 'Pacific Coast Conference Center', '200 Granville St, Vancouver, BC', 'Valet parking available at the entrance.');

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)

VALUES (126, 'Mount Royal Grand Hall', '300 Rue Sherbrooke O, Montreal, QC', 'Limited parking available onsite; street parking may be available nearby.');

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)

VALUES (127, 'Rocky Mountain Event Center', '400 4 Ave SW, Calgary, AB', 'Paid parking available in the building\'s underground garage; entrance on 4th Ave.');

2. Event table DDL

CREATE TABLE Event (

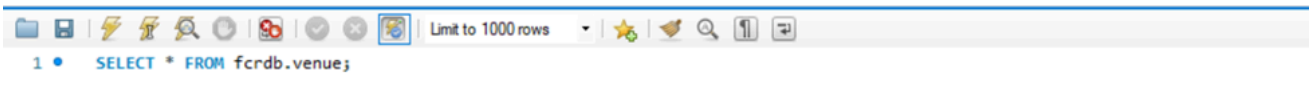
EventID INTEGER NOT NULL PRIMARY KEY,

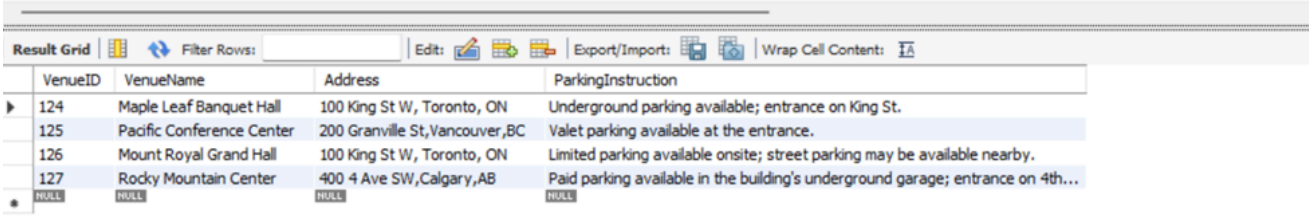
EventName VARCHAR(40) NOT NULL,

EventType VARCHAR(30) NOT NULL ,

StartDate DATE,

EndDate DATE,





VenueID	VenueName	Address	ParkingInstruction
124	Maple Leaf Banquet Hall	100 King St W, Toronto, ON	Underground parking available; entrance on King St.
125	Pacific Conference Center	200 Granville St,Vancouver,BC	Valet parking available at the entrance.
126	Mount Royal Grand Hall	100 King St W, Toronto, ON	Limited parking available onsite; street parking may be available nearby.
127	Rocky Mountain Center	400 4 Ave SW,Calgary,AB	Paid parking available in the building's underground garage; entrance on 4th...
NULL	NULL	NULL	NULL

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

```
USE fcrdb;
CREATE TABLE Venue (
                                VenueID INTEGER NOT NULL ,
                               VenueName VARCHAR(25) NOT NULL,
                               Address VARCHAR(30) NOT NULL ,
                               ParkingInstruction TEXT,
                          CONSTRAINT Venue_PK PRIMARY KEY (VenueID));

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)
VALUES ('124', 'Maple Leaf Banquet Hall', '100 King St W, Toronto, ON', 'Underground parking available; entrance on King St.');

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)
VALUES ('125', 'Pacific Conference Center', '200 Granville St,Vancouver,BC', 'Valet parking available at the entrance.');

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)
VALUES ('126', 'Mount Royal Grand Hall', '100 King St W, Toronto, ON', 'Limited parking available onsite; street parking may be available nearby.

INSERT INTO venue (venueID, VenueName, Address, ParkingInstruction)
VALUES ('127', 'Rocky Mountain Center', '400 4 Ave SW,Calgary,AB', 'Paid parking available in the building\'s underground garage; entrance on 4th
```

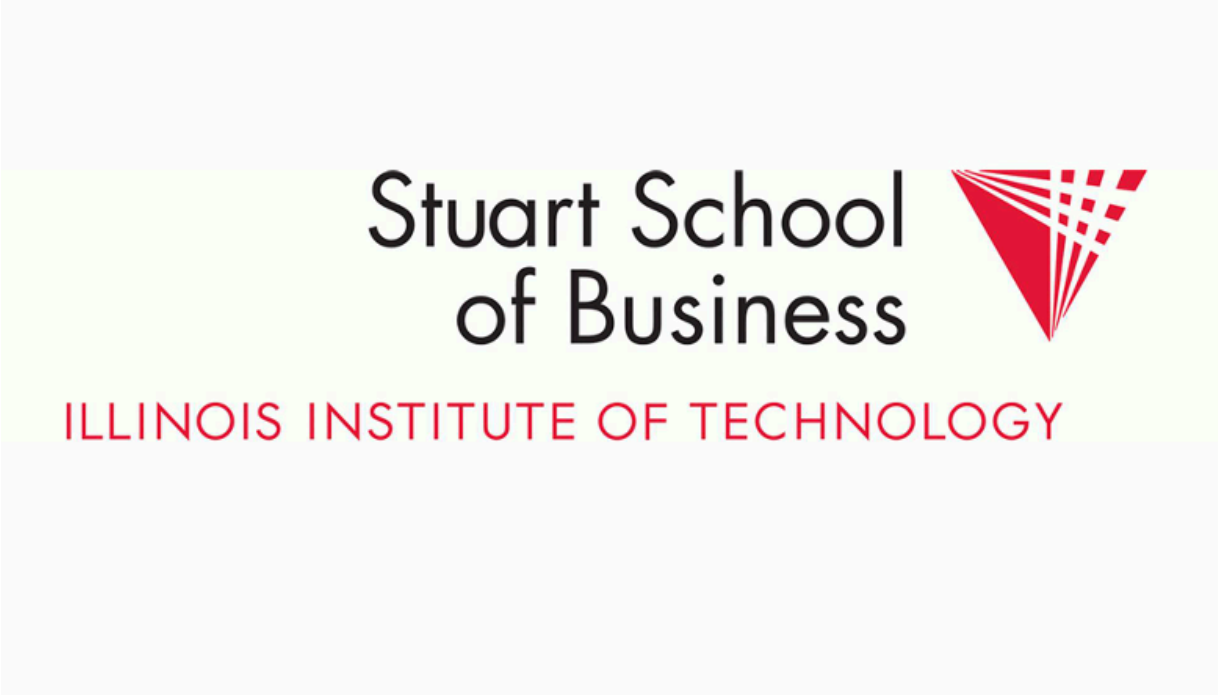
Description TEXT,

VenueID INTEGER NOT NULL,

CONSTRAINT Event_FK FOREIGN KEY (VenueID) REFERENCES Venue(VenueID));

Event table DML

INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)



VALUES (1, 'Tech Expo', 'Trade Show', '2024-06-15', '2024-06-17', 'Annual technology exhibition showcasing the latest innovations in the industry.', 125);

INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)

VALUES (2, 'Summer Rib Festival', 'Rib Fest', '2024-07-20', '2024-07-22', 'Celebration of BBQ ribs with live music, vendors, and family-friendly activities.', 126);

INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)

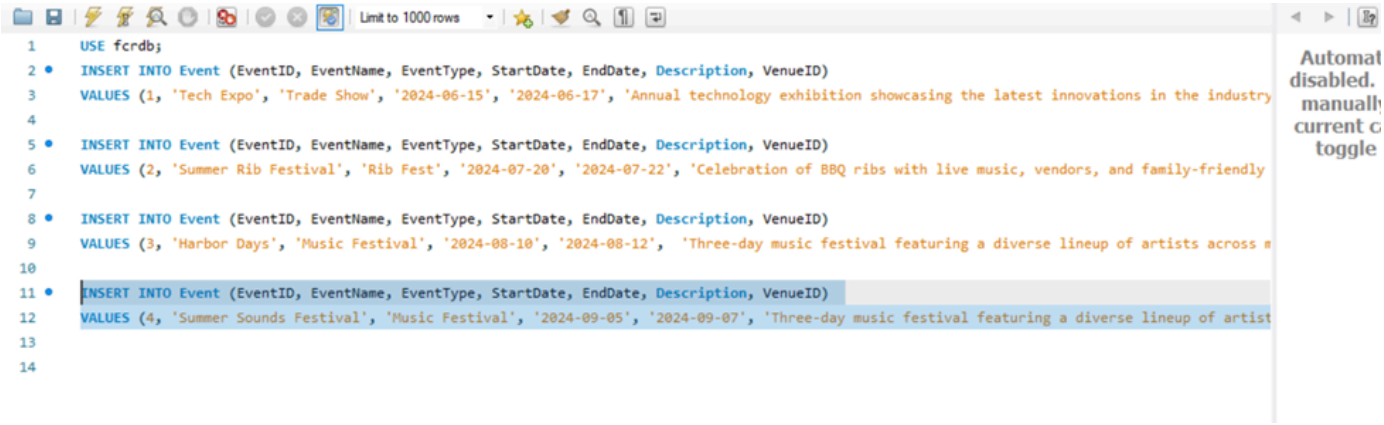
VALUES (3, 'Harbor Days', 'Music Festival', '2024-08-10', '2024-08-12', 'Three-day music festival featuring a diverse lineup of artists across multiple stages.', 127);

INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)

VALUES (4, 'Summer Sounds Festival', 'Music Festival', '2024-09-05', '2024-09-07', 'Three-day music festival featuring a diverse lineup of artists across multiple stages.', 127);

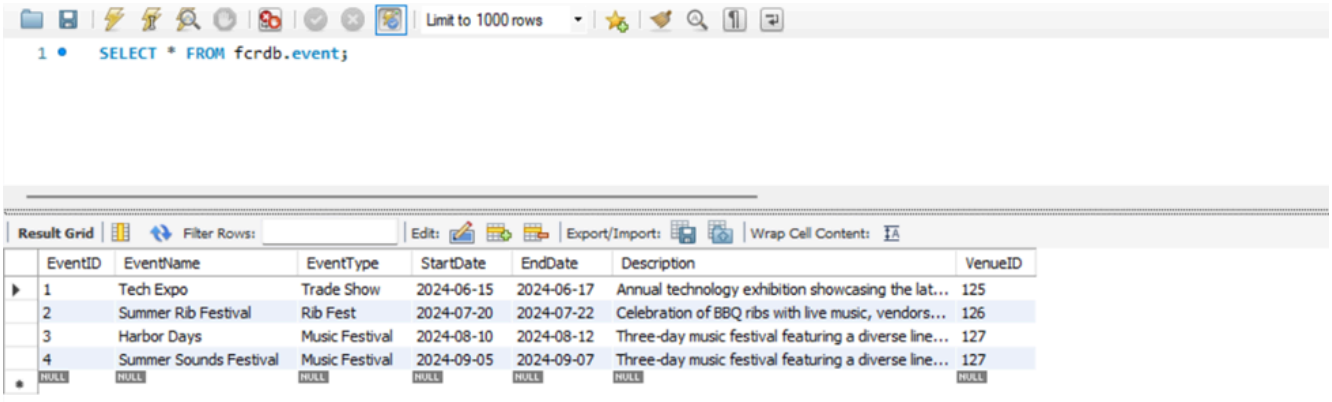
Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY



```
1  USE fcrdb;
2  INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)
3  VALUES (1, 'Tech Expo', 'Trade Show', '2024-06-15', '2024-06-17', 'Annual technology exhibition showcasing the latest innovations in the industry
4
5  INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)
6  VALUES (2, 'Summer Rib Festival', 'Rib Fest', '2024-07-20', '2024-07-22', 'Celebration of BBQ ribs with live music, vendors, and family-friendly
7
8  INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)
9  VALUES (3, 'Harbor Days', 'Music Festival', '2024-08-10', '2024-08-12',  'Three-day music festival featuring a diverse lineup of artists across m
10
11 INSERT INTO Event (EventID, EventName, EventType, StartDate, EndDate, Description, VenueID)
12 VALUES (4, 'Summer Sounds Festival', 'Music Festival', '2024-09-05', '2024-09-07', 'Three-day music festival featuring a diverse lineup of artist
13
14
```

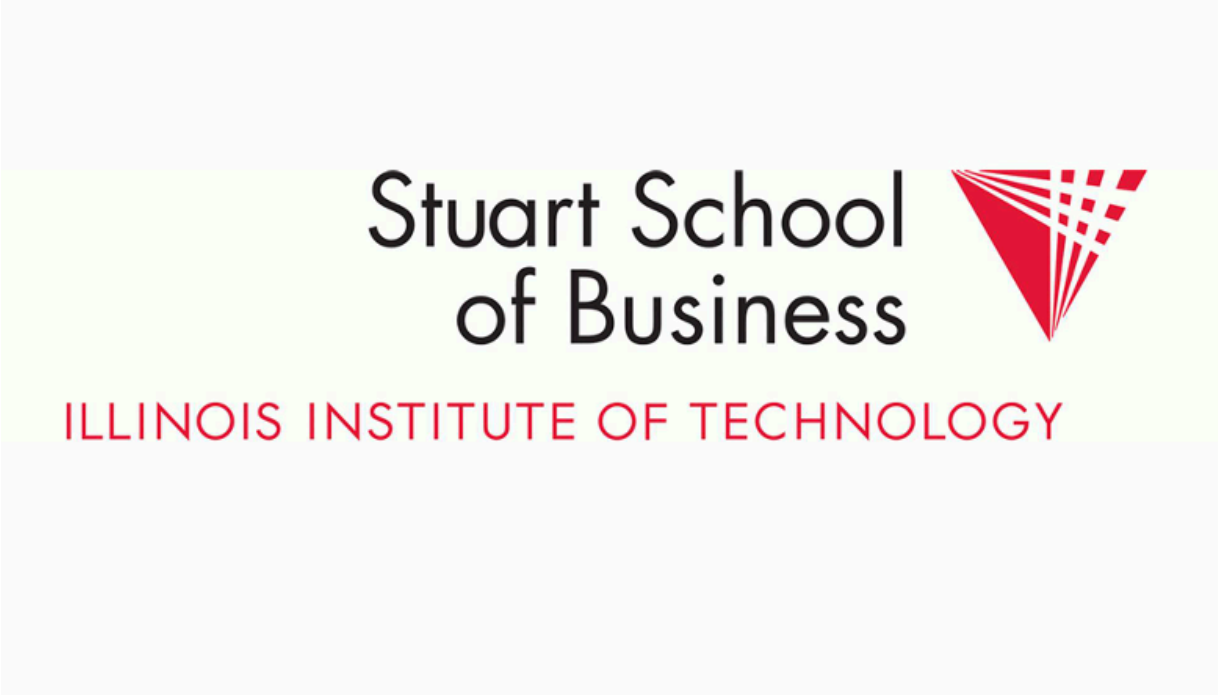
Automat
disabled.
manuall
current c
toggle



```
1  SELECT * FROM fcrdb.event;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EventID	EventName	EventType	StartDate	EndDate	Description	VenueID
1	Tech Expo	Trade Show	2024-06-15	2024-06-17	Annual technology exhibition showcasing the lat...	125
2	Summer Rib Festival	Rib Fest	2024-07-20	2024-07-22	Celebration of BBQ ribs with live music, vendors...	126
3	Harbor Days	Music Festival	2024-08-10	2024-08-12	Three-day music festival featuring a diverse line...	127
4	Summer Sounds Festival	Music Festival	2024-09-05	2024-09-07	Three-day music festival featuring a diverse line...	127
NULL	NULL	NULL	NULL	NULL	NULL	NULL



3. Booth table DDL

CREATE TABLE Booth (

BoothID INTEGER NOT NULL,

BoothLocation VARCHAR(50),

EventID INTEGER,

CONSTRAINT Booth_PK PRIMARY KEY(BoothID),

CONSTRAINT Booth_FK1 FOREIGN KEY (EventID) REFERENCES Event(EventID));

Booth table DML

INSERT INTO Booth (BoothID, BoothLocation, EventID)

VALUES (3001, 'J1', 1);

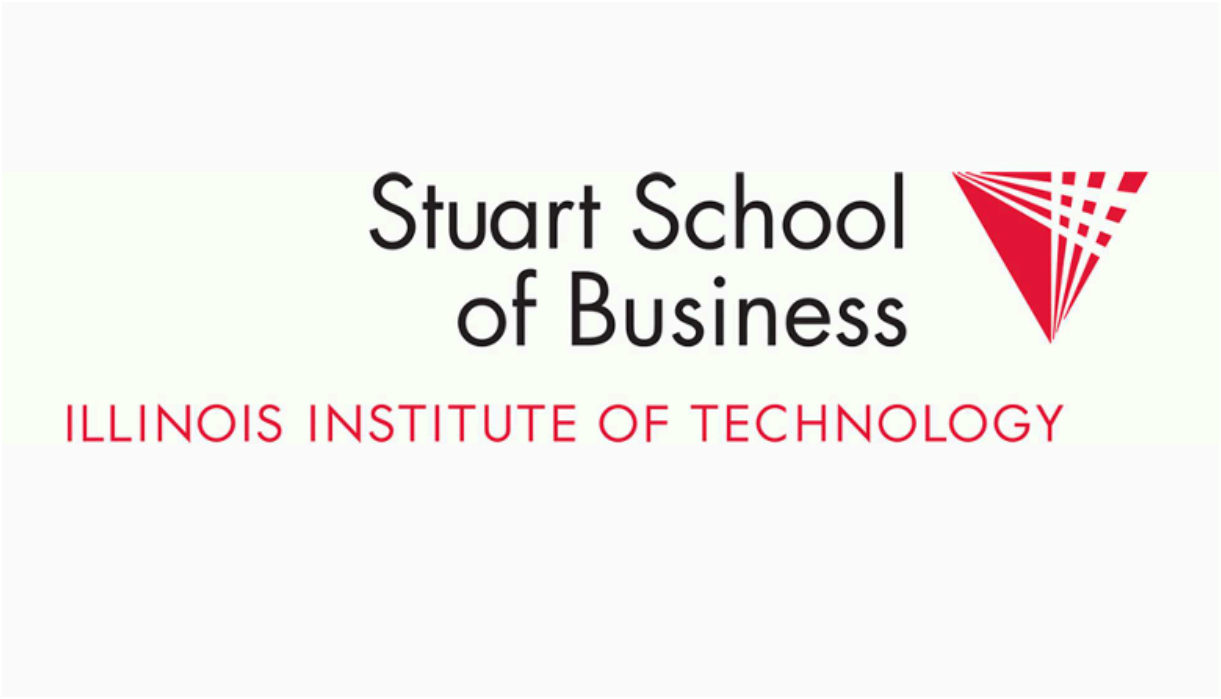
INSERT INTO Booth (BoothID, BoothLocation, EventID)

VALUES (3002, 'K2', 2);

INSERT INTO Booth (BoothID, BoothLocation, EventID)

VALUES (3003, 'S11', 3);

INSERT INTO Booth (BoothID, BoothLocation, EventID)



VALUES (3004, 'T14', 4);

Stuart School of Business

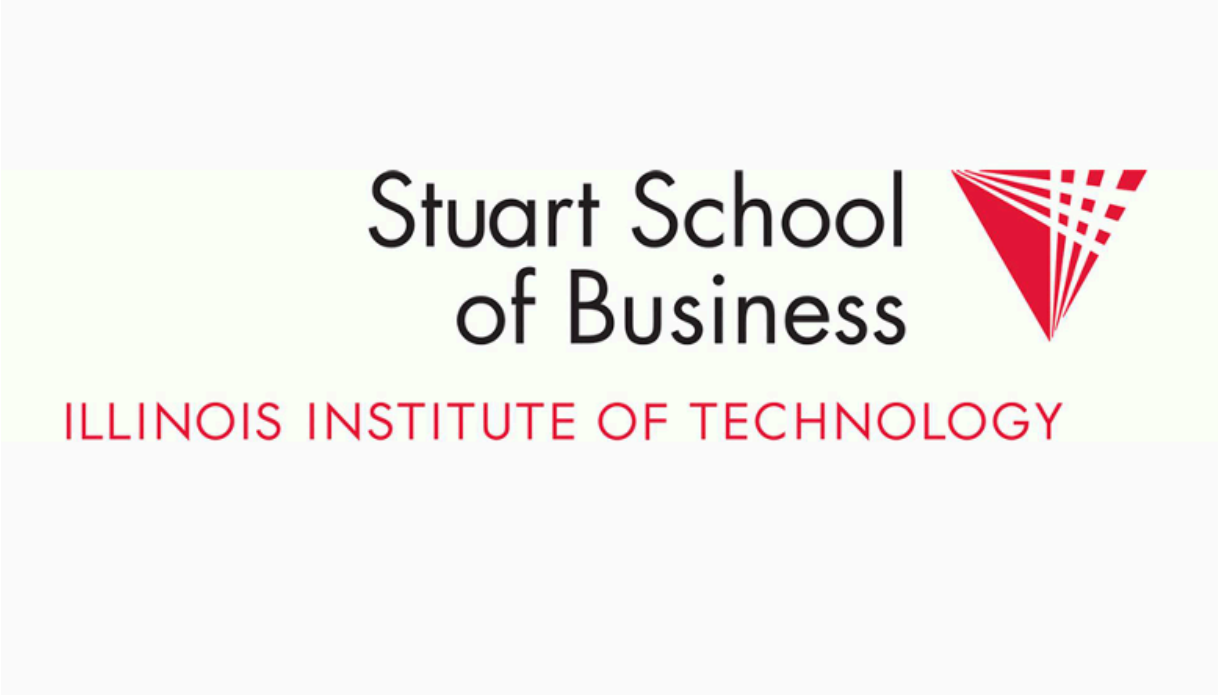
ILLINOIS INSTITUTE OF TECHNOLOGY

```sql
USE fcrdb;
CREATE TABLE Booth (
    BoothID INTEGER NOT NULL,
    BoothLocation VARCHAR(50),
    EventID INTEGER,
     CONSTRAINT Booth_PK PRIMARY KEY(BoothID),
     CONSTRAINT Booth_FK1 FOREIGN KEY (EventID) REFERENCES Event(EventID));

INSERT INTO Booth (BoothID, BoothLocation, EventID)
VALUES (3001, 'J1', 1);

INSERT INTO Booth (BoothID, BoothLocation, EventID)
VALUES (3002, 'K2', 2);

INSERT INTO Booth (BoothID, BoothLocation, EventID)
VALUES (3003, 'S11', 3);

INSERT INTO Booth (BoothID, BoothLocation, EventID)
VALUES (3004, 'T14', 4);
```

```sql
SELECT * FROM fcrdb.booth;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

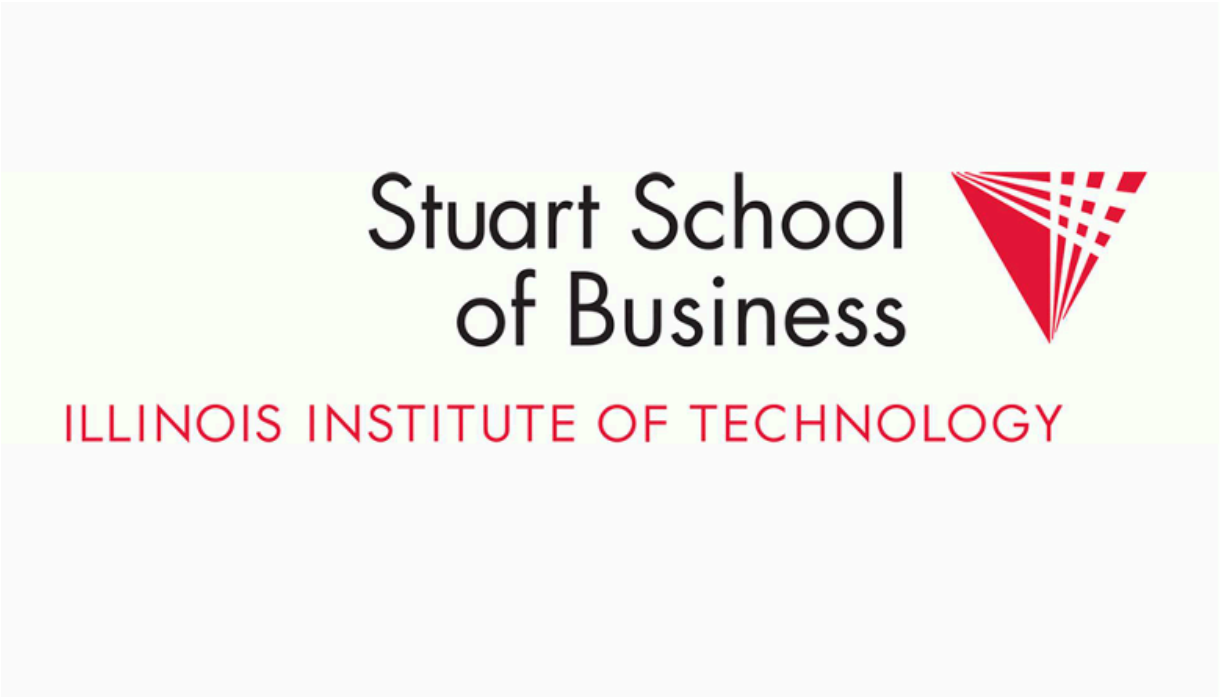
BoothID	BoothLocation	EventID
3001	J1	1
3002	K2	2
3003	S11	3



4. SalesPerson table DDL

```
CREATE TABLE SalesPerson (

SalesPersonID INTEGER NOT NULL ,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Address VARCHAR(100),

PhoneNo VARCHAR(10),

BoothID INTEGER NOT NULL,

CONSTRAINT SalesPerson_PK PRIMARY KEY (SalesPersonID)

CONSTRAINT SalesPerson_FK1 FOREIGN KEY (BoothID) REFERENCES booth(BoothID)

);
```

SalesPerson table DML



INSERT INTO SalesPerson (SalesPersonID, FirstName, LastName, Address, PhoneNo, BoothID)

VALUES (401, 'John', 'Doe', '123 Main St, Cityville, ON', 1234567890, 3001);

INSERT INTO SalesPerson (SalesPersonID, FirstName, LastName, Address, PhoneNo, BoothID)

VALUES (402, 'Jane', 'Smith', '456 Elm St, Townsville, BC', 9876543210, 3002);

INSERT INTO SalesPerson (SalesPersonID, FirstName, LastName, Address, PhoneNo, BoothID)

VALUES (403, 'Mike', 'Johnson', '789 Oak St, Hilltop, QC', 4567890123, 3003);

INSERT INTO SalesPerson (SalesPersonID, FirstName, LastName, Address, PhoneNo, BoothID)

VALUES (404, 'Sarah', 'Williams', '101 Pine St, Seaside, AB', 9870123456, 3004);

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

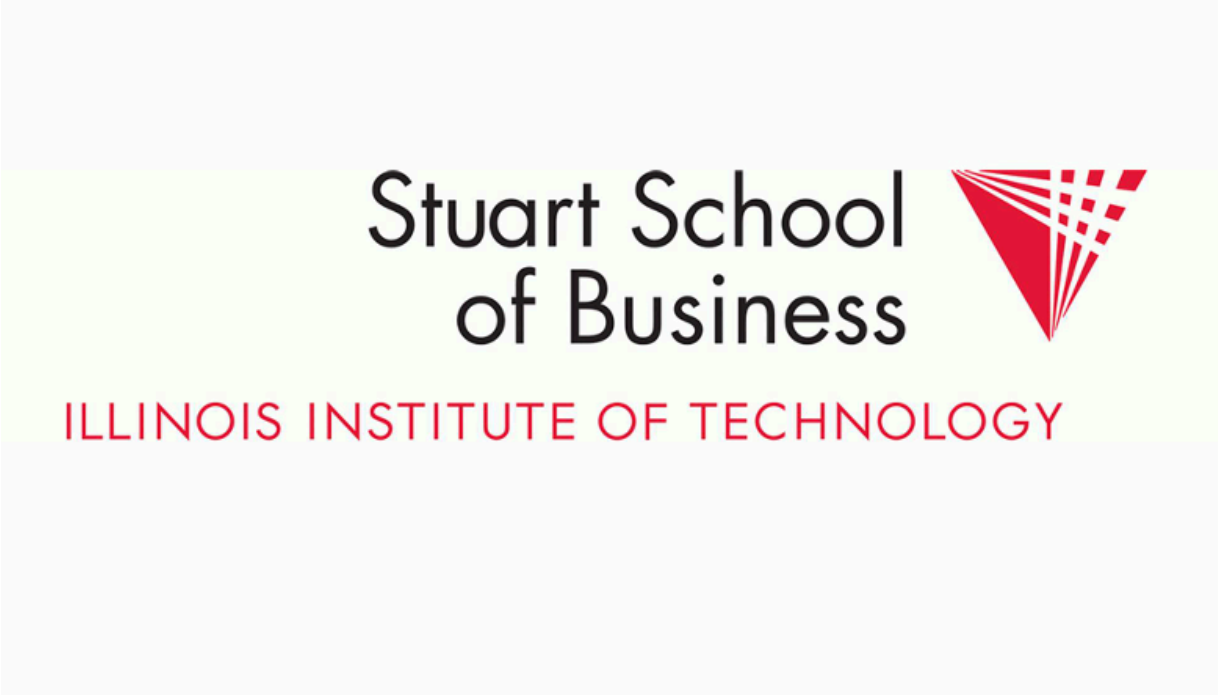
```sql
USE fcrdb;
CREATE TABLE SalesPerson (
                                        SalesPersonID INTEGER NOT NULL,
                                        FirstName VARCHAR(50),
                                        LastName VARCHAR(50),
                                        Address VARCHAR(100),
                                        PhoneNo VARCHAR(10),
                                        BoothID INTEGER NOT NULL,
    CONSTRAINT SalesPerson_PK PRIMARY KEY (SalesPersonID),
    CONSTRAINT SalesPerson_FK1 FOREIGN KEY (BoothID) REFERENCES booth(BoothID));
```

SQL File 14* | venue | venue | venue | venue | event | venue | DDSFSQL | event | event

```sql
SELECT * FROM fcrdb.salesperson;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

SalesPersonID	FirstName	LastName	Address	PhoneNo	BoothID
401	John	Doe	123 Main St, Cityville, ON	1234567890	3001
402	Jane	Smith	456 Elm St, Townsville, BC	9876543210	3002
403	Mike	Johnson	789 Oak St, Hilltop, QC	4567890123	3003
404	Sarah	Williams	101 Pine St, Seaside, AB	9870123456	3004
NULL	NULL	NULL	NULL	NULL	NULL

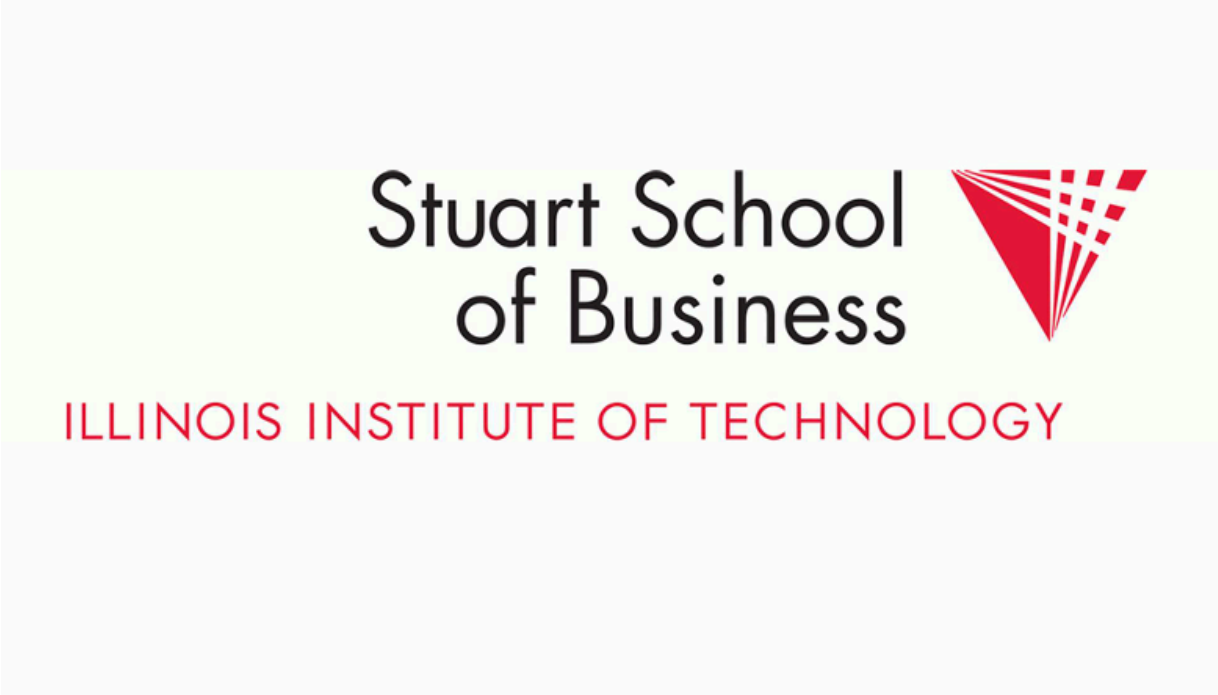


5. Product table DDL

```
CREATE TABLE Product (

ProductID INTEGER NOT NULL,

ProductName VARCHAR(30),

WholesaleCost DECIMAL(6, 2),

HigherPrice DECIMAL(6, 2),

MinimumSellingPrice DECIMAL(6, 2),

CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

Product table DML

```
INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice,
MinimumSellingPrice)

VALUES (7, 'Bubble Guns', 50.00, 79.99, 69.99);

INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice,
MinimumSellingPrice)

VALUES (5, 'Mini Frisbees', 20.00, 49.99, 29.99);

INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice,
```



MinimumSellingPrice)

VALUES (9, 'Glow Sticks', 15.00, 25.00, 20.00);

INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice, MinimumSellingPrice)

VALUES (6, 'Cooling Towel', 70.00, 90.00, 79.99);

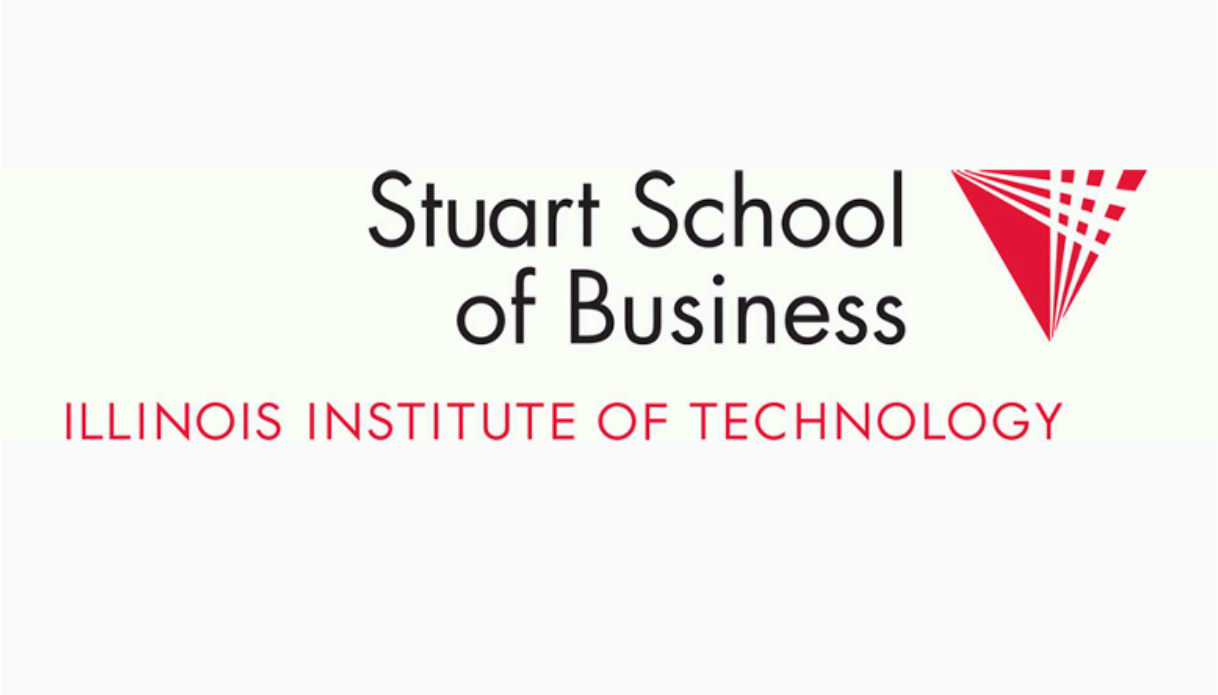
Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

```
1   USE fcrdb;
2   CREATE TABLE Product (
3                                   ProductID INTEGER NOT NULL,
4                                   ProductName VARCHAR(30),
5                                   WholesaleCost DECIMAL(6, 2),
6                                   HigherPrice DECIMAL(6, 2),
7                                   MinimumSellingPrice DECIMAL(6, 2),
8                                   CONSTRAINT Product_PK PRIMARY KEY (ProductID));
9   INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice, MinimumSellingPrice)
10  VALUES (7, 'Bubble Guns', 50.00, 79.99, 69.99);
11
12  INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice, MinimumSellingPrice)
13  VALUES (5, 'Mini Frisbees', 20.00, 49.99, 29.99);
14
15  INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice, MinimumSellingPrice)
16  VALUES (9, 'Glow Sticks', 15.00, 25.00, 20.00);
17
18  INSERT INTO Product (ProductID, ProductName, WholesaleCost, HigherPrice, MinimumSellingPrice)
19  VALUES (6, 'Cooling Towel', 70.00, 90.00, 79.99);
20  SELECT * FROM PRODUCT;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ProductID	ProductName	WholesaleCost	HigherPrice	MinimumSellingPrice
5	Mini Frisbees	20.00	49.99	29.99
6	Cooling Towel	70.00	90.00	79.99
7	Bubble Guns	50.00	79.99	69.99
9	Glow Sticks	15.00	25.00	20.00
NULL	NULL	NULL	NULL	NULL



6. Sales table DDL

CREATE TABLE Sales (

SalesTransactionID INTEGER NOT NULL,

FinalSellingPrice DECIMAL(10, 2),

QuantitySold INTEGER,

ProductID INTEGER,

SalesPersonID INTEGER

CONSTRAINT SalesPerson_PK PRIMARY KEY (SalesTrasactionID)

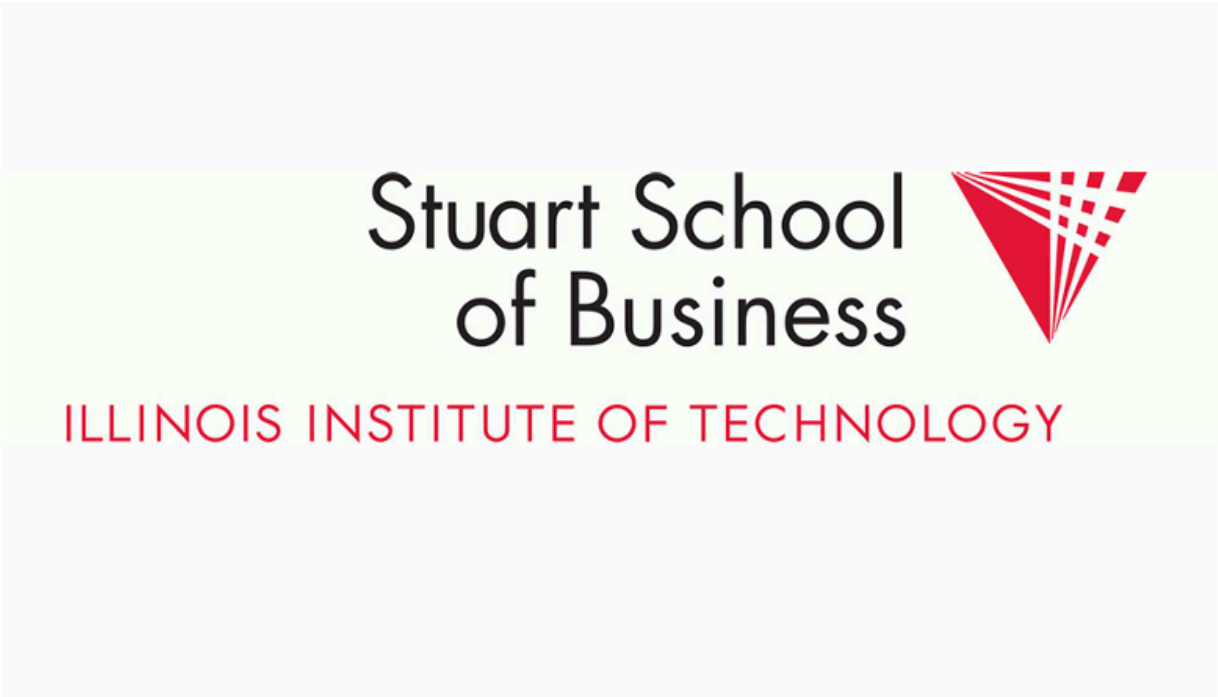
CONSTRAINT SalePerson_FK1 FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

CONSTRAINT SalesPerson_FK2 FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID));

Sales table DML

INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)

VALUES (501, 76.00, 10, 7, 402);



INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)

VALUES (502, 34.99, 20, 5, 404);

INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)

VALUES (503, 36.99, 15, 5, 403);

INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)

VALUES (504, 85.00, 5, 6, 401);

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

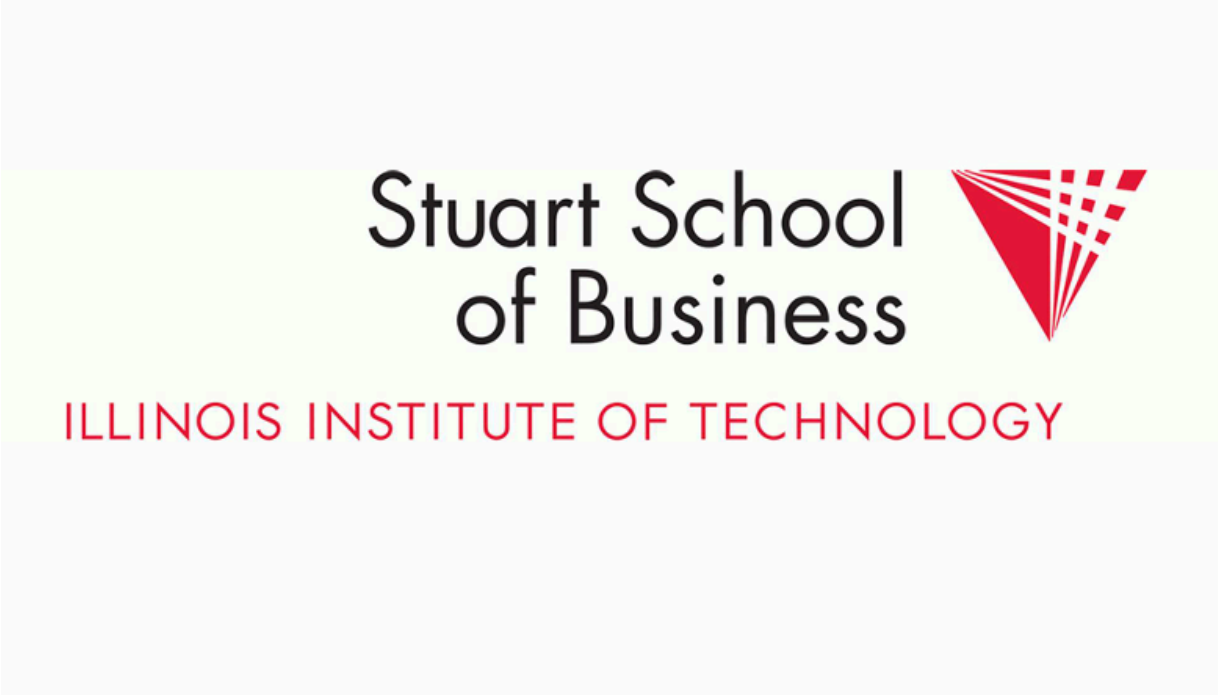
```
1  USE fcrdb;
2  CREATE TABLE Sales (
3                                  SalesTransactionID INTEGER NOT NULL,
4                                  FinalSellingPrice DECIMAL(10, 2),
5                                  QuantitySold INTEGER,
6                                 ProductID INTEGER,
7                                 SalesPersonID INTEGER,
8    CONSTRAINT SalesPerson_PK PRIMARY KEY (SalesTransactionID),
9    CONSTRAINT SalePerson_FK1 FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
10     CONSTRAINT SalesPerson_FK2 FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID));
11     INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)
12 VALUES (501, 76.00, 10, 7, 402);
13
14 INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)
15 VALUES (502, 34.99, 20, 5, 404);
16
17 INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)
18 VALUES (503, 36.99, 15, 5, 403);
19
20 INSERT INTO Sales (SalesTransactionID, FinalSellingPrice, QuantitySold, ProductID, SalesPersonID)
21 VALUES (504, 85.00, 5, 6, 401);
22 SELECT * FROM Sales;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

SalesTransactionID	FinalSellingPrice	QuantitySold	ProductID	SalesPersonID
501	76.00	10	7	402
502	34.99	20	5	404
503	36.99	15	5	403
504	85.00	5	6	401
NULL	NULL	NULL	NULL	NULL

7. Shift table DDL

CREATE TABLE Shift (



ShiftID INTEGER NOT NULL,

StartTime TIME,

EndTime TIME,

SalesPersonID INTEGER,

CONSTRAINT Shift_PK PRIMARY KEY(ShiftID),

CONSTRAINT Shift_FK2 FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID));

Shift table DML

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)

VALUES (001, '08:00:00', '12:00:00', 401);

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)

VALUES (002, '12:00:00', '16:00:00',402);

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)

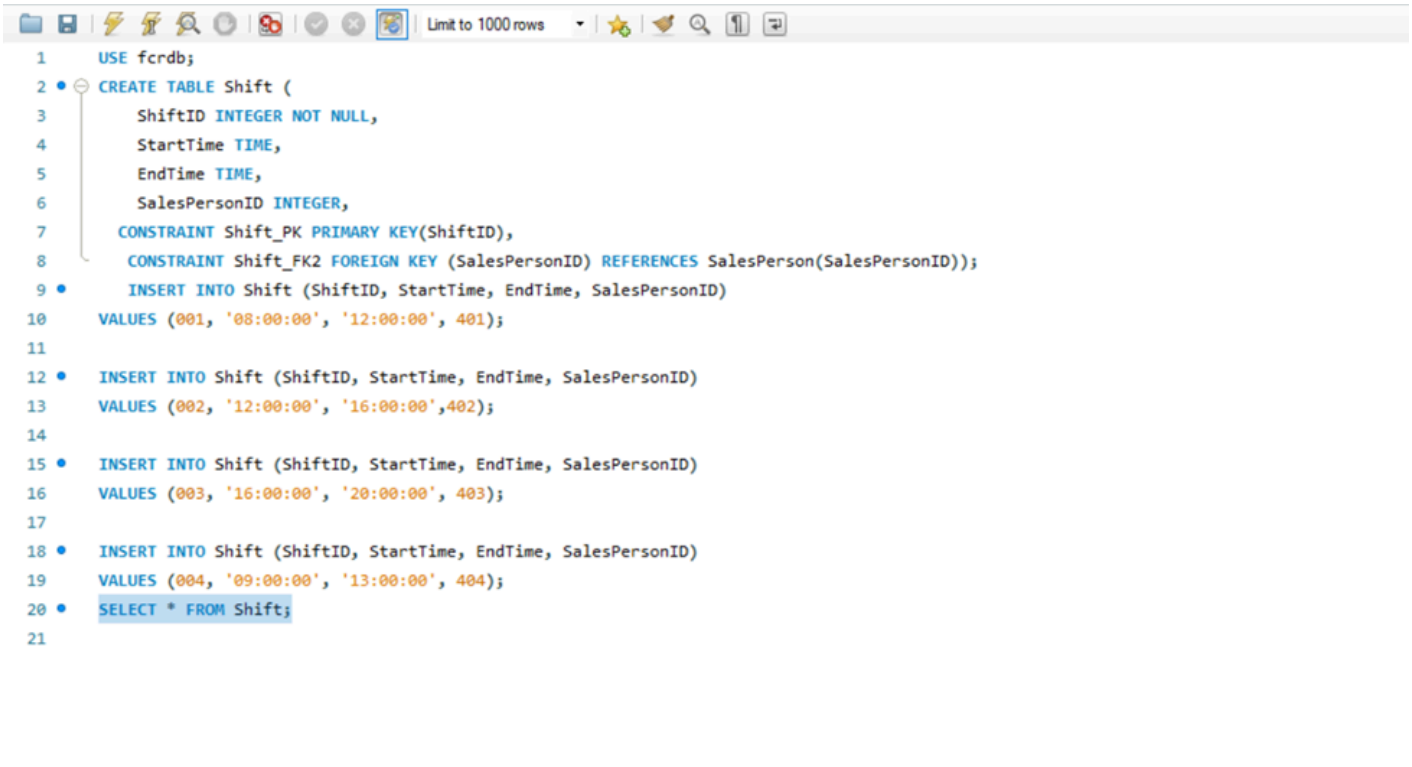
VALUES (003, '16:00:00', '20:00:00', 403);

Stuart School of Business

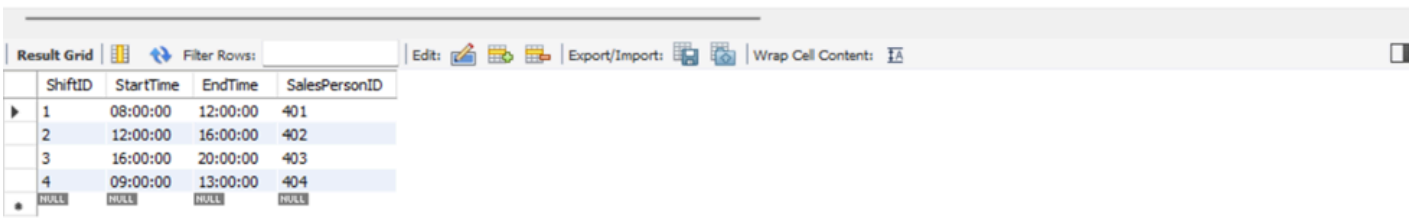
ILLINOIS INSTITUTE OF TECHNOLOGY

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)

VALUES (004, '09:00:00', '13:00:00', 404);



```
USE fcrdb;
CREATE TABLE Shift (
    ShiftID INTEGER NOT NULL,
    StartTime TIME,
    EndTime TIME,
    SalesPersonID INTEGER,
  CONSTRAINT Shift_PK PRIMARY KEY(ShiftID),
   CONSTRAINT Shift_FK2 FOREIGN KEY (SalesPersonID) REFERENCES SalesPerson(SalesPersonID));
   INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)
VALUES (001, '08:00:00', '12:00:00', 401);

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)
VALUES (002, '12:00:00', '16:00:00',402);

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)
VALUES (003, '16:00:00', '20:00:00', 403);

INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)
VALUES (004, '09:00:00', '13:00:00', 404);
SELECT * FROM Shift;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ShiftID	StartTime	EndTime	SalesPersonID
1	08:00:00	12:00:00	401
2	12:00:00	16:00:00	402
3	16:00:00	20:00:00	403
4	09:00:00	13:00:00	404
NULL	NULL	NULL	NULL

Stuart School of Business

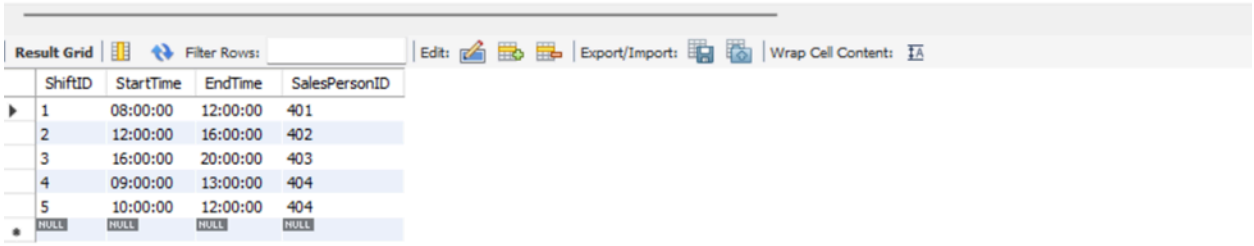
ILLINOIS INSTITUTE OF TECHNOLOGY

Update and Delete query

We can utilize the UPDATE and DELETE queries to effortlessly make the required changes within the tables of our database.

Before

```
21 •  INSERT INTO Shift (ShiftID, StartTime, EndTime, SalesPersonID)
22    VALUES (005, '10:00:00', '12:00:00', 404);
23
```



ShiftID	StartTime	EndTime	SalesPersonID
1	08:00:00	12:00:00	401
2	12:00:00	16:00:00	402
3	16:00:00	20:00:00	403
4	09:00:00	13:00:00	404
5	10:00:00	12:00:00	404
NULL	NULL	NULL	NULL

UPDATE Shift

SET EndTime = '17:00:00'

WHERE ShiftID = 005;

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

```
23 •  UPDATE Shift
24    SET EndTime = '17:00:00'
25    WHERE ShiftID = 005;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ShiftID	StartTime	EndTime	SalesPersonID
1	08:00:00	12:00:00	401
2	12:00:00	16:00:00	402
3	16:00:00	20:00:00	403
4	09:00:00	13:00:00	404
5	10:00:00	17:00:00	404
NULL	NULL	NULL	NULL

DELETE FROM Shift

WHERE ShiftID = 005;

```
26 •  DELETE FROM Shift
27    WHERE ShiftID = 005;
28
29
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ShiftID	StartTime	EndTime	SalesPersonID
1	08:00:00	12:00:00	401
2	12:00:00	16:00:00	402
3	16:00:00	20:00:00	403
4	09:00:00	13:00:00	404
NULL	NULL	NULL	NULL

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

SQL commands to generate reports from the database:

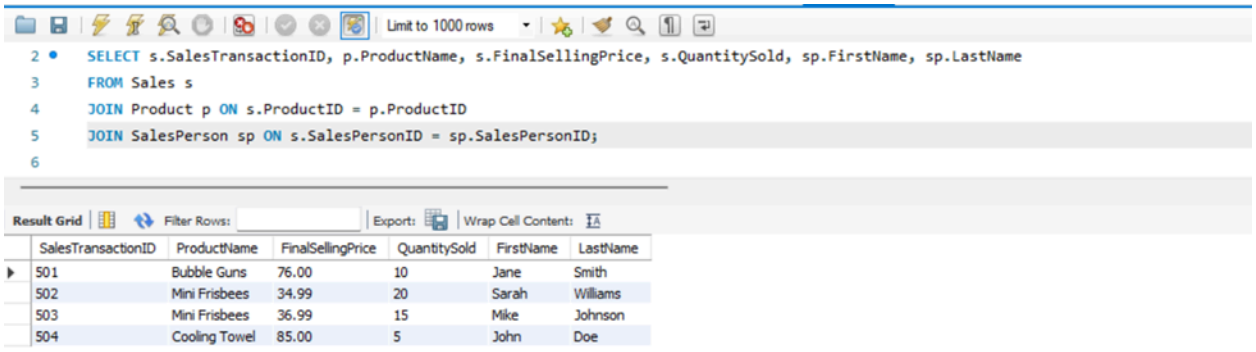
1)Retrieve all sales transactions along with the corresponding salesperson's first name, last name, and product type sold.

SELECT s.SalesTransactionID, p.ProductName, s.FinalSellingPrice, s.QuantitySold, sp.FirstName, sp.LastName

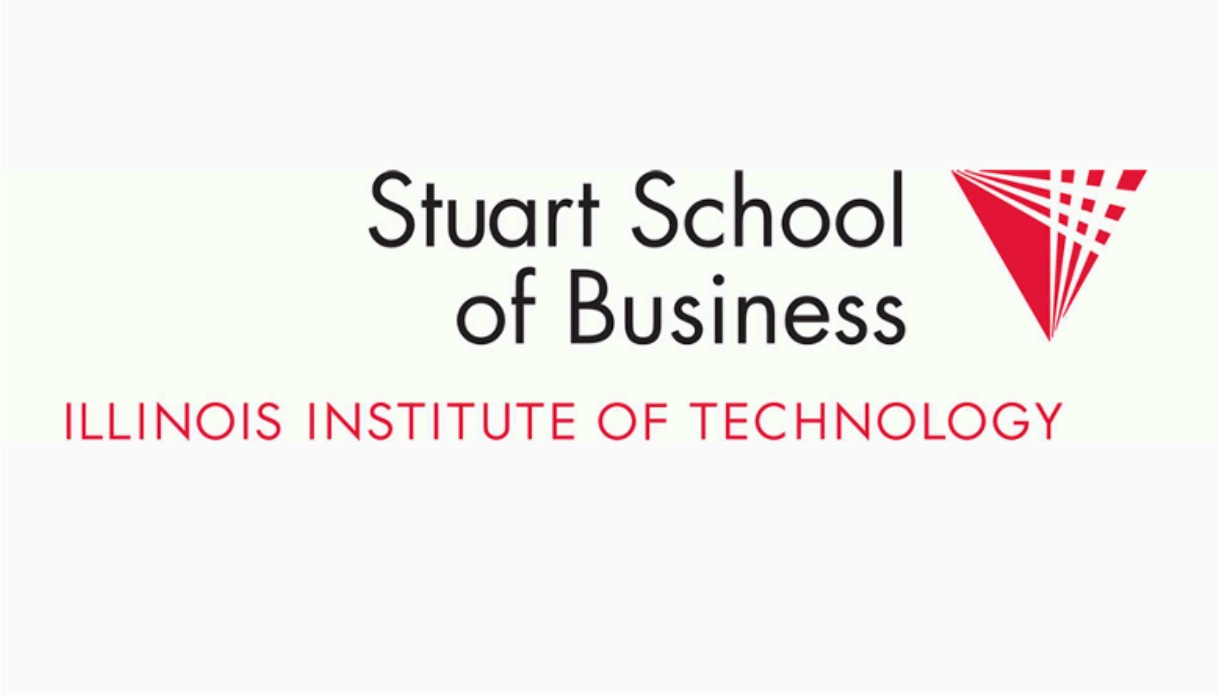
FROM Sales s

JOIN Product p ON s.ProductID = p.ProductID

JOIN SalesPerson sp ON s.SalesPersonID = sp.SalesPersonID;



SalesTransactionID	ProductName	FinalSellingPrice	QuantitySold	FirstName	LastName
501	Bubble Guns	76.00	10	Jane	Smith
502	Mini Frisbees	34.99	20	Sarah	Williams
503	Mini Frisbees	36.99	15	Mike	Johnson
504	Cooling Towel	85.00	5	John	Doe



2: Find the total sales amount for each event along with the event name and venue name.

```
SELECT e.EventName, v.VenueName, SUM(s.FinalSellingPrice * s.QuantitySold) AS
TotalSalesAmount

FROM Event e

JOIN Venue v ON e.VenueID = v.VenueID

LEFT JOIN Booth b ON e.EventID = b.EventID

LEFT JOIN SalesPerson sp ON b.BoothID = sp.BoothID

LEFT JOIN Sales s ON sp.SalesPersonID = s.SalesPersonID

GROUP BY e.EventID, e.EventName, v.VenueName;
```

Stuart School of Business

ILLINOIS INSTITUTE OF TECHNOLOGY

```
1   USE fcrdb;
2 • SELECT e.EventName, v.VenueName, SUM(s.FinalSellingPrice * s.QuantitySold) AS TotalSalesAmount
3   FROM Event e
4   JOIN Venue v ON e.VenueID = v.VenueID
5   LEFT JOIN Booth b ON e.EventID = b.EventID
6   LEFT JOIN SalesPerson sp ON b.BoothID = sp.BoothID
7   LEFT JOIN Sales s ON sp.SalesPersonID = s.SalesPersonID
8   GROUP BY e.EventID, e.EventName, v.VenueName;
9
10
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EventName	VenueName	TotalSalesAmount
Tech Expo	Pacific Conference Center	425.00
Summer Rib Festival	Mount Royal Grand Hall	760.00
Harbor Days	Rocky Mountain Center	554.85
Summer Sounds Festival	Rocky Mountain Center	699.80

CONCLUSION

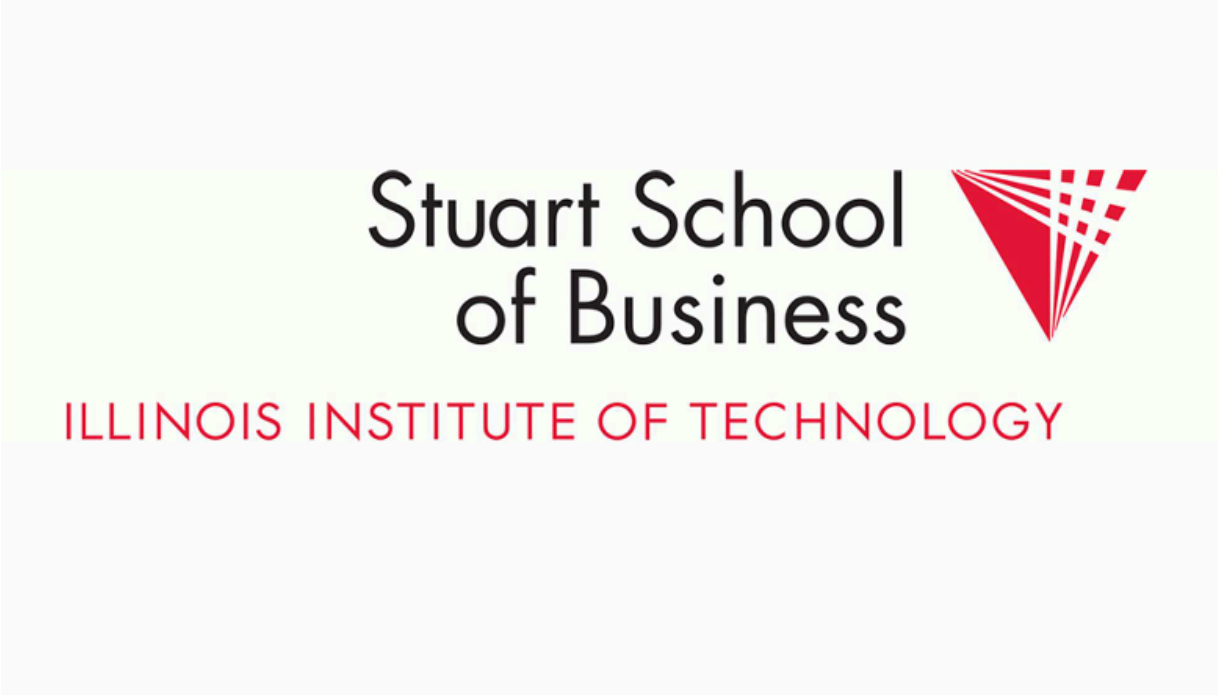
In conclusion, the comprehensive design and implementation of Foxcore Retail's

database, underscored by a sophisticated entity-relationship model and adherence to normalization principles, has established a robust and scalable infrastructure. This technological leap, facilitated by Barker's notation and the strategic use of DDL and DML operations, has equipped Foxcore Retail with a powerful tool to enhance their operational management significantly.

The implementation of the database is reflected in meticulously crafted tables and relationships that mirror the company's workflow and data management requirements. Through the use of SQL commands to manipulate and define the database structure, we have ensured that the data environment is not only optimized for current operations but is also poised for future expansion and adaptability.

As Foxcore Retail continues to navigate the dynamic landscape of retail sales at music festivals and trade shows, this database system stands as a testament to the company's commitment to data integrity, operational efficiency, and strategic growth. With the ability to easily update and modify data, the database paves the way for continuous improvement and agility in decision-making processes.

Ultimately, the database is more than just a collection of tables and data points; it is the cornerstone of a data-driven approach that will drive Foxcore Retail's success in a competitive market. With a solid foundation now in place, Foxcore Retail can look forward to not just sustaining but also enhancing their market position through informed strategies and exceptional customer service.



RESOURCES

1. MySQL Workbench
2. SQL Complier
3. Lucid chart