

Project <Excelsior's Comic Database>

Student Name: Jiahao Liu

Student ID: 21212299

jiahao.liu@ucdconnect.ie

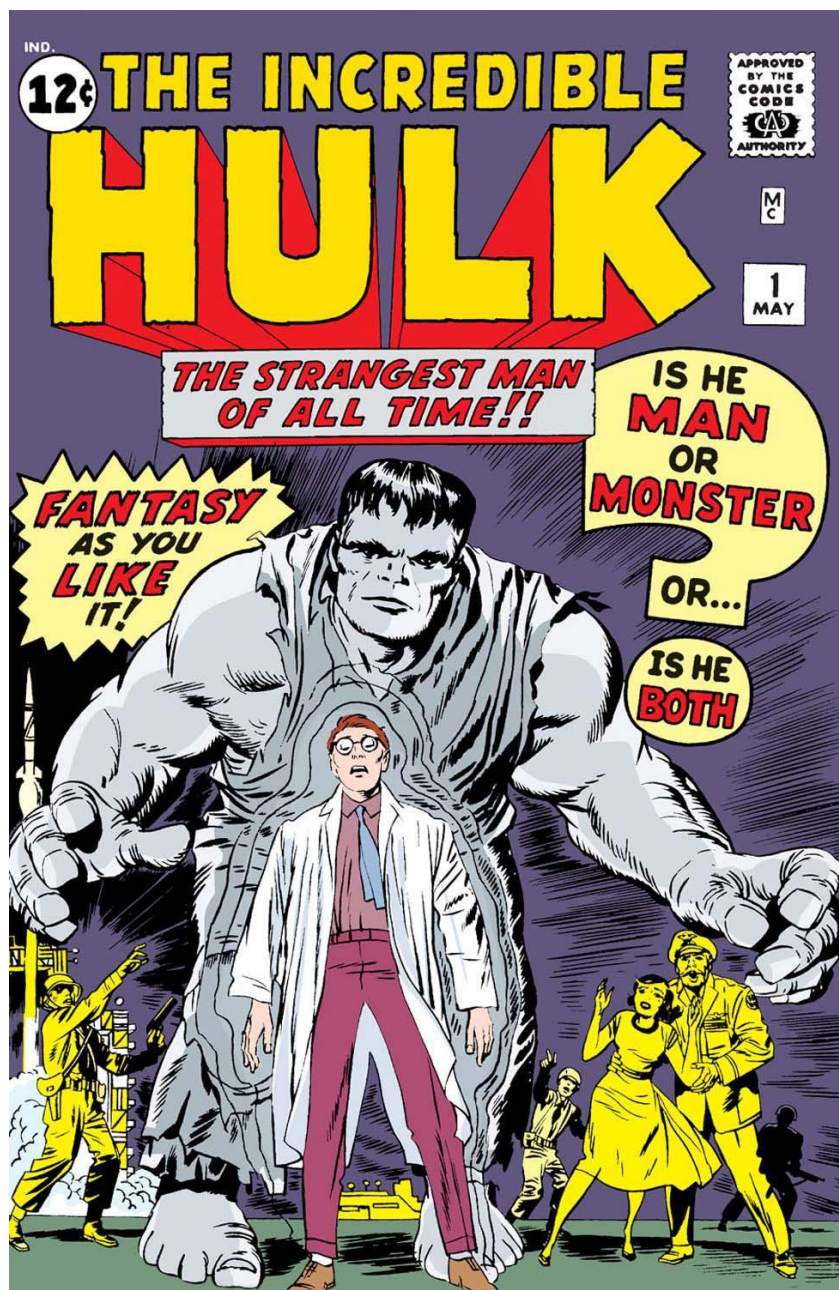


Table of Contents

Introduction.....	1
Database Plan: A Schematic View.....	2
Database Structure: A Normalized View.....	8
Database View:.....	13
Procedual Elements:	17
Example Queries:.....	26
Conclusion	31
Acknowledgements	33
Reference	33

List of Figures

Figure 1: E-R diagram Entities	7
Figure 2: E-R diagram	13
Figure 3: Comic_value_info view	14
Figure 4: Query for View_1	14
Figure 5: Results for View_1	14
Figure 6: Comics_availability_view	15
Figure 7: Query for View_2	15
Figure 8: Results for View_1	15
Figure 9: View_comic_details	16
Figure 10: Query for View_3	16
Figure 11: Results for View_1	16
Figure 12: Top_10_comics_by_sale_price	16
Figure 13: Query for View_4	17
Figure 14: Results for View_1	17
Figure 15: Insert comic data	18
Figure 16: GetSeriesIdByName's Effect	18
Figure 17: Call insert_publisher_if_not_exists	18
Figure 18: publisher data	18
Figure 19: Call insert_series_if_not_exists	19
Figure 20: Series data	19
Figure 21: CALL insert_character_if_not_exists	19
Figure 22: Part of character data	20
Figure 23: Call insert_if_not_exists_creators	20
Figure 24: Part of creators data	20
Figure 25: Call insert_comic_character_if_not_exists	21
Figure 26: Part of comic_character data	21
Figure 27: Call insert_comic_creator_if_not_exists	22
Figure 28: Part of omic_creator data	22
Figure 29: TEST PROCEDURE update_sale_price	23
Figure 30: The prices before the update	23
Figure 31: The updated prices.	23
Figure 32: TEST PROCEDURE update_sale_prices_in_range	23
Figure 33: Before update data	24
Figure 34: After update:	24
Figure 35: TEST PROCEDURE CalculateProfitLoss	24
Figure 36: Results for CalculateProfitLoss	25
Figure 37: TEST TRIGGER	25
Figure 38: Results for setting	25
Figure 39: Results for TRIGGER	26
Figure 40: Query1 for VIEW_1 comic_value_info	26
Figure 41: Results of Query1 for VIEW_1 comic_value_info	26
Figure 42: Query2 for VIEW_2 comics_Availability_View	27
Figure 43: Results of Query2 for VIEW_2 comics_Availability_View	27
Figure 44: Query3 for VIEW_3 view_comic_details	27
Figure 45: Results of Query3 for VIEW_3 view_comic_details	27
Figure 46: Query4 for VIEW_4 top_10_valuable_comics	27
Figure 47: Results of Query4 for VIEW_4 top_10_valuable_comics	27
Figure 48: Special Query 1	28
Figure 49: Results of Special Query 1	28
Figure 50 : Special Query 2:	28
Figure 51: The result of Special Query 2	28
Figure 52: Special Query 3	28
Figure 53: The result of Special Query 3	29
Figure 54: Special Query 4	29
Figure 55: The result of Special Query 4	29
Figure 56: Special Query 5	29
Figure 57: The result of Special Query 5	29
Figure 58: Special Query 6	30
Figure 59: Results of Special Query 6	30
Figure 60: Search comics by character:	30
Figure 61: The results for Search comics by character	30
Figure 62: Search comics by publisher	31
Figure 63: The results of Search comics by publisher	31

Introduction

The comic industry originated in the late 19th century and has continued to grow and develop over time. From the initial newspaper comic strips to today's digital comics, the industry has experienced many significant milestones. Classic American comics such as "Superman" "Batman" and "Spider-Man" have become representative masterpieces in the history of comics. These works, through exploring the depths of human imagination, have told captivating stories that transcend cultural and geographical boundaries, bringing endless joy to audiences worldwide.

The comic genre is diverse, ranging from traditional superheroes, fantasy, science fiction, and horror to more modern magical girl, school youth, and daily life themes. Various genres cater to readers of all age groups and interests. In addition to different genres, the comic industry is spread across various countries and regions, such as the United States, Japan, France, and Belgium, each having its unique comic style and tradition.

The development of the comic industry is closely related to the continuous innovation of publishers, creators, and characters. Some publishers, such as DC[1] and Marvel[2] in the United States, have become industry leaders, releasing many popular works. These publishers collaborate with numerous talented writers, artists, and editors to create unforgettable characters and stories.

Furthermore, with advances in technology and the popularization of the internet, the comic industry is continuously evolving[3]. The emergence of digital comics and webtoons allows readers to access comics anytime, anywhere. Moreover, comics are no longer limited to paper and digital forms. Many comic works have been adapted into animations, movies, TV shows, and games, further expanding the influence of the comic industry.

In this diverse and vibrant comic industry, collectors, store owners, investors, and enthusiasts require a robust database and application to better understand the comic market, manage their collections, analyze market trends and values, and interact with other fans. To meet these needs, a comprehensive comic database and advanced application can help them track newly released works, popular characters and authors, as well as rare collectibles in the market.

The primary purpose of this project is to design and develop a comprehensive and powerful database, Excelsior, as the cornerstone of an advanced application for comic book collectors, store owners, investors, and enthusiasts. The core objective of the comic database is to manage, query, and analyze data related to comic books, containing various detailed information, such as the publication date, author, artist, characters etc., for each work. Additionally, the database can track the market value and quality of comics, providing valuable reference information for investors and collectors. Given the large number of comic

issues published over the years, the numerous characters and creators involved, and the various factors influencing comic book values, the database will handle a considerable volume of data. For this project, comic data has been found on the leagueofcomicgeeks[4] website for experimentation. This data needs careful planning and structuring to ensure efficient storage, retrieval, and analysis.

The role of the database within the entire system is to serve as the foundation for the advanced application built upon it, providing a reliable and organized source of information for different user needs. The database itself is not responsible for directly implementing user interfaces or handling user interactions. Instead, it will focus on efficiently managing and organizing the necessary data, ensuring users can search, access, and analyze the information they need to make informed decisions regarding comic book collections, investments, or interests. Based on this, the advanced application can offer user-friendly features to increase user engagement and social elements, such as subscription functions, allowing users to follow their favorite works, authors, and characters to receive notifications when new works are released or important updates occur. Moreover, the application can recommend works related to users' interests, helping them discover new hobbies.

Therefore, to effectively support the advanced application, the design of the comic database must also consider the following key factors[5]:

- a. Scalability: Given the extensive nature of the comic field, the database must be able to handle large amounts of data and scale smoothly as the number of records grows over time.
- b. Flexibility: The database should accommodate various types of data, such as information about series, issues, characters, creators, and market values, as well as any new data requirements needed for future developments in the comic industry.
- c. Searchability: The database must support efficient search and filtering capabilities, enabling users to quickly and easily find comics that they are interested in.
- d. Reliability: The database should ensure data integrity, consistency, and accuracy, providing users with trustworthy information to support their decision-making process.
- e. Extensibility: The database should consider future developments and improvements, allowing for seamless integration of new features, data types, or external data sources.

By focusing on the above key factors, the database will be designed more comprehensively. In this way, it can better support the advanced application and continuously provide a comprehensive, user-friendly, and valuable experience for users in the comic field, addressing the ongoing development of the comic industry in the future.

Database Plan: A Schematic View

The Excelsior database have to be designed to address the growing needs of the comic book industry, with an emphasis on flexibility, ease of use, and scalability. In this section, I will present a schematic view of the database, showcasing the main entities, their attributes, and the relationships that connect them. Then I will provide an Entity-Relationship (E-R) diagram to visually represent these aspects, followed by a detailed explanation of the database schema design choices.

Main Entities

Publishers:

This entity represents the publishing companies responsible for producing and distributing comic books and graphic novels. Publishers play a crucial role in the comic book industry, as they provide the platform for creators to share their work with readers.

Attributes	Description
<u>publisher_id</u>	a unique identifier for each publisher.
name	the name of the publisher

Table 1: Publishers' Attributes

The "Publisher" entity is related to the "Series" entity through the `publisher_id` foreign key in the series table. This relationship signifies that each publisher can have multiple series associated with it, forming a one-to-many relationship between the Publisher and Series entities. The key relationship connecting the Publisher entity with other entities in the database is its connection to the Series entity through the `publisher_id` attribute.

Series:

A series refers to a collection of comic books or graphic novels that are part of a continuous storyline or share a common theme. Each series is typically associated with a specific publisher.

Attributes	Description
<u>series_id</u>	a unique identifier for each series
name	the name of the publisher
<u>inception_year</u>	the year when the series was started
<u>publisher_id</u>	a foreign key

Table 2: Series' Attributes

The "Series" entity is related to the "Publisher" entity through the `publisher_id` foreign key, forming a one-to-many relationship, as each publisher can have multiple series associated with it. The Series entity is also connected to the "Comics" entity through the `series_id` foreign key in the comics table. This relationship signifies that each series can have multiple comics associated with it, forming another one-to-many relationship between the Series and Comics entities.

The key relationships connecting the Series entity with other entities in the database are its connections to the Publisher entity through the `publisher_id` attribute and to the Comics entity through the `series_id` attribute.

Comics:

This entity encapsulates individual comic books or graphic novels. Comics are the core components of the database, as they contain the storylines and characters that define the industry.

Attributes	Description
<u>comic_id</u>	a unique identifier for each publisher.

title	the name of the publisher
issue_number	the issue number of the comic
release_date	the release date of the comic
series_id	a foreign key
type	representing the type of the comic
edition	the edition of the comic
signed_by_creator	indicating if the comic is signed
cover_date	the cover date of the comic

Table 3: Comics' Attributes

The "Comic" entity is related to the "Series" entity through the series_id foreign key, forming a one-to-many relationship, as each series can have multiple comics associated with it. The Comic entity is also connected to several other entities in the database, such as Quality Mapping, Characters, Creators, Prices, and Overstreet Price Guide, through their respective tables and foreign key relationships.

The key relationship connecting the Comic entity with other entities in the database is its connection to the Series entity through the series_id attribute. Additionally, the Comic entity serves as a central point connecting multiple other entities, such as Quality Mapping, Characters, Creators, Prices, and Overstreet Price Guide, through their respective foreign key relationships to the comic_id attribute.

quality_mapping:

quality_mapping associates the condition of a comic book with a condition number, allowing for easy comparison and evaluation of comic book conditions.

Attributes	Description
quality_mapping_id	a unique identifier for each quality mapping
comic_id	a foreign key
comics_condition	the condition of the comic as a string
condition_number	the condition of the comic as a decimal number

Table 4: quality_mapping's Attributes

The "Quality Mapping" entity is related to the "Comic" entity through the comic_id foreign key, forming a one-to-many relationship, as each comic can have multiple quality mappings associated with it. This relationship allows for the storage of different condition ratings for the same comic book, enabling better tracking of comic book quality.

The key relationship connecting the Quality Mapping entity with other entities in the database is its connection to the Comic entity through the comic_id attribute. This relationship establishes a link between the condition and quality information of a comic book and the comic book itself.

Creators:

Creators include artists, writers, and other professionals involved in the creation of comic books. Their roles and biographies provide insight into the creative process behind comics.

Attributes	Description
------------	-------------

creator_id	a unique identifier for each creator
name	the name of the creator
role	the role of the creator
biography	a text description of the creator

Table 5: Creators' Attributes

The "Creator" entity is related to the "Comic" entity through a many-to-many relationship facilitated by the comic_creators table. This relationship allows each comic to have multiple creators associated with it and each creator to be associated with multiple comics.

The key relationship connecting the Creator entity with other entities in the database is its connection to the Comic entity through the comic_creators table, which uses the creator_id attribute to link creators to comics. This relationship establishes a connection between the creators and the comic books they have worked on, enabling the tracking of creator contributions in the database.

Characters:

Characters are the fictional personalities featured in comic books. They are essential to the stories and often serve as the driving force behind a comic's popularity.

Attributes	Description
character_id	a unique identifier for each character
name	the name of the character

Table 6: Characters' Attributes

The "Character" entity is related to the "Comic" entity through a many-to-many relationship facilitated by the comic_characters table. This relationship allows each comic to have multiple characters associated with it and each character to appear in multiple comics.

The key relationship connecting the Character entity with other entities in the database is its connection to the Comic entity through the comic_characters table, which uses the character_id attribute to link characters to comics. This relationship establishes a connection between the characters and the comic books they appear in, enabling the tracking of character appearances in the database.

Comic_characters:

This entity serves as a bridge between comics and characters, defining the relationships between specific comic books and the characters featured in them.

Attributes	Description
comic_character_id	a unique identifier for each comic-character association
comic_id	a foreign key
character_id	a foreign key

Table 7: Comic_characters' Attributes

The key relationships connecting the comic_characters table with other entities in the database are its connections to the Comic and Character entities through the comic_id and character_id attributes. These relationships establish a many-to-many relationship between the Comic and Character entities, enabling the tracking of character appearances in the comic

books. The UNIQUE constraint on the combination of comic_id and character_id ensures that each comic-character association is unique and not duplicated in the table.

Comic_creators: Similar to Comic_characters, this entity connects comic books to their creators, highlighting the professionals responsible for each comic.

Attributes	Description
comic_creator_id	a unique identifier for each comic-creator association
comic_id	a foreign key
creator_id	a foreign key

Table 8: Comic_creators' Attributes

The key relationships connecting the comic_creators table with other entities in the database are its connections to the Comic and Creator entities through the comic_id and creator_id attributes. These relationships establish a many-to-many relationship between the Comic and Creator entities, enabling the tracking of creator contributions to the comic books. The UNIQUE constraint on the combination of comic_id and creator_id ensures that each comic-creator association is unique and not duplicated in the table.

Prices:

This entity stores the purchase and sale prices for individual comics, along with their associated quality.

Attributes	Description
price_id	a unique identifier for each publisher.
comic_id	a foreign key
purchase_price	the purchase price of the comic
sale_price	the sale price of the comic
quality	a text description of the comic
last_updated	the date when the pricing information was last updated

Table 9: Prices' Attributes

The key relationship connecting the Price entity with other entities in the database is its connection to the Comic entity through the comic_id attribute. This relationship establishes a connection between the comic books and their pricing information, enabling the tracking of prices for each comic in the database. The relationship between the Comic and Price entities is one-to-many, meaning that each comic can have multiple price records associated with it.

Overstreet_price_guide:

This entity holds the pricing data from the Overstreet Price Guide, a widely recognized authority on comic book valuations, providing industry-standard pricing information based on the condition of the comic.

Attributes	Description
overstreet_price_guide_id	a unique identifier for each Overstreet Price Guide entry
comic_id	a foreign key
year	the year of the Overstreet Price Guide entry
Fair_price	the price for the comic in "Fair" condition
Good_price	the price for the comic in "Good" condition
Very_good_price	the price for the comic in "Very Good" condition

Fine_price	the price for the comic in "Fine" condition
Very_fine_price	the price for the comic in "Very Fine" condition
Near_mint_price	the price for the comic in "Near Mint" condition
Mint_price	the price for the comic in "Mint" condition

Table 10: Overstreet_price_guide's Attributes

The key relationship connecting the Overstreet Price Guide entity with other entities in the database is its connection to the Comic entity through the comic_id attribute. This relationship establishes a connection between the comic books and their pricing information from the Overstreet Price Guide, enabling the tracking of prices for each comic across various quality grades. The relationship between the Comic and Overstreet Price Guide entities is one-to-many, meaning that each comic can have multiple Overstreet Price Guide records associated with it.

E-R diagram

Based on the analysis of the above ten tables and a comprehensive consideration of the relationships between the entities, I have drawn the following E-R diagram in Figure 1.

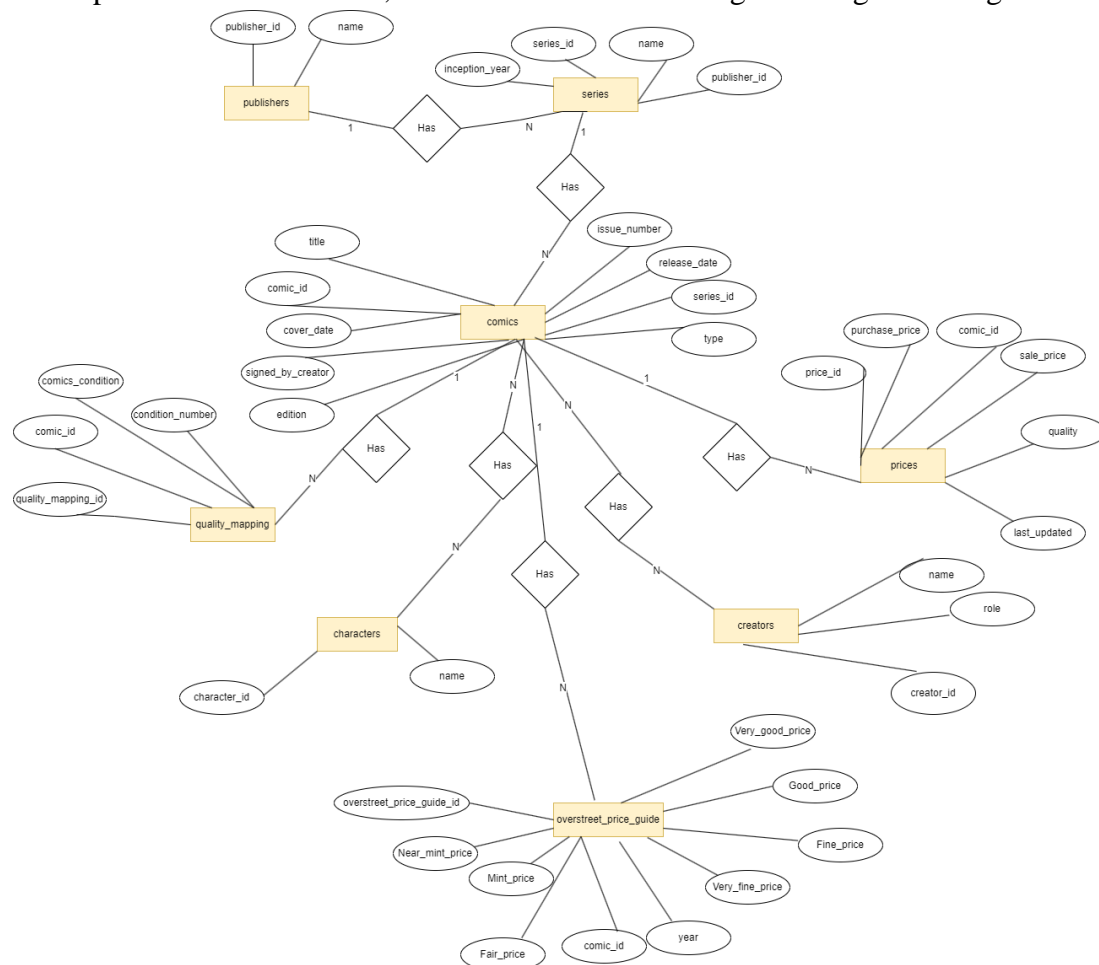


Figure 1: E-R diagram Entities

The E-R diagram depicted in the Excelsior database visually exemplifies the primary entities, attributes, and relationships integral to the schema. This design approach offers a coherent and succinct representation of the interconnections among the database's various entities.

Furthermore, the decision to segregate entities such as creators and characters from the central comic entities enhances adaptability and scalability in response to the ever-changing industry landscape. Consequently, this facilitates the efficient management and storage of data pertinent to comic books and their affiliated entities, including series, publishers, characters, creators, and pricing information. Employing one-to-many relationships in the majority of cases enables an accurate reflection of real-world associations between entities. For instance, while a publisher can have multiple series under its purview, each series is uniquely associated with a single publisher. In cases involving more intricate relationships between entities, such as the links between comics and characters or comics and creators, many-to-many relationships are employed, utilizing intermediary tables (comic characters and comic creators) to streamline data management and querying. Finally, by accurately modeling these relationships, the schema promotes efficient querying, data manipulation, and maintenance of database consistency, while mitigating redundancies.

Database Structure: A Normalized View

Normalization is the process of organizing a relational database in a way that reduces data redundancy and ensures data integrity. It involves the application of specific rules, or normal forms (NF), to the database schema design. In this section, I will describe the role of each table in the Excelsior database and demonstrate how the design meets 1NF, 2NF, and 3NF requirements. Additionally, I will discuss whether the database is in Boyce-Codd Normal Form (BCNF) and explain the reasoning.

Roles of Tables in the Excelsior Database:

Publishers: It stores information about publishing companies, including their unique identifier (publisher_id) and name.

Attributes	Data Type	Constraints	Key
publisher_id	INT	NOT NULL AI	PRIMARY KEY
name	VARCHAR(100)	NOT NULL	

Table 11: Publishers Table Details

The publishers table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria as it has unique column names and atomic values. The table is in 2NF since it is already in 1NF, and the non-key attribute name is fully functionally dependent on the primary key publisher_id. The table is also in 3NF because it is in 2NF, and there is no transitive dependency between non-key attributes. Lastly, the table is in BCNF as it is in 3NF, and the only functional dependency, publisher_id → name, involves a super key.

Series:

It contains information about comic book series, such as the unique identifier (series_id), name, inception year, and the associated publisher (publisher_id).

Attributes	Data Type	Constraints	Key
series_id	INT	NOT NULL AI	PRIMARY KEY
name	VARCHAR(100)	NOT NULL	
inception_year	INT	NOT NULL	

publisher_id	INT	NOT NULL	FOREIGN KEY
--------------	-----	----------	-------------

Table 12: Series Table Details

The series table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria as it has unique column names and atomic values. The table is in 2NF since it is already in 1NF, and all non-key attributes (name, inception_year, and publisher_id) are fully functionally dependent on the primary key series_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with publisher_id being a foreign key and the other two non-key attributes not being dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (series_id).

Comics:

It holds data about individual comic books, including the unique identifier (comic_id), title, issue number, release date, series identifier (series_id), type, edition, signed_by_creator status, and cover date.

Attributes	Data_Type	Constraints	Key
comic_id	INT	NOT NULL AI	PRIMARY KEY
title	VARCHAR(255)	NOT NULL	
issue_number	INT	NOT NULL	
release_date	DATE	NOT NULL	
series_id	INT	NOT NULL	FOREIGN KEY
type	ENUM	NOT NULL	
signed_by_creator	BOOLEAN	NOT NULL	
cover_date	DATE	NOT NULL	

Table 13: Comic Table Details

The comics table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria as it has unique column names and atomic values. The table is in 2NF since it is already in 1NF, and all non-key attributes are fully functionally dependent on the primary key comic_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with series_id being a foreign key and the other non-key attributes not being dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (comic_id).

quality_mapping:

It associates the condition of a comic book with a condition number, with attributes including the unique identifier (quality_mapping_id), comic identifier (comic_id), comics_condition, and condition_number.

Attributes	Data_Type	Constraints	Key
quality_mapping_id	INT	NOT NULL AI	PRIMARY KEY
comic_id	INT	NOT NULL	FOREIGN KEY
comics_condition	VARCHAR(10)	NOT NULL	
condition_number	DECIMAL(4,2)	NOT NULL	

Table 14: quality_mapping Table Details

The `quality_mapping` table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names, atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes (`comic_id`, `comics_condition`, and `condition_number`) are fully functionally dependent on the primary key `quality_mapping_id`. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with `comic_id` being a foreign key and the other non-key attributes not being dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (`quality_mapping_id`).

Creators: It contains information about creators, such as their unique identifier (`creator_id`), name, role, and biography.

Attributes	Data_Type	Constraints	Key
<code>creator_id</code>	INT	NOT NULL AI	PRIMARY KEY
<code>name</code>	VARCHAR(100)	NOT NULL	
<code>role</code>	VARCHAR(50)	NOT NULL	
<code>biography</code>	TEXT	NOT NULL	

Table 15: Creators Table Details

The `creators` table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names, atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes (`name`, `role`, and `biography`) are fully functionally dependent on the primary key `creator_id`. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, as none of them depend on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (`creator_id`).

Characters: It stores data about characters, including their unique identifier (`character_id`) and name.

Attributes	Data_Type	Constraints	Key
<code>character_id</code>	INT	NOT NULL AI	PRIMARY KEY
<code>name</code>	VARCHAR(100)	NOT NULL	

Table 16: Characters Table Details

The `characters` table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names, atomic values. The table is in 2NF since it is already in 1NF and the non-key attribute (`name`) is fully functionally dependent on the primary key `character_id`. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, given that there is only one non-key attribute (`name`). Lastly, the table is in BCNF as it is in 3NF, and the functional dependency involves a superkey (`character_id`).

Comic_characters: It serves as a bridge between comics and characters, with attributes such as the unique identifier (`comic_character_id`), comic identifier (`comic_id`), and character identifier (`character_id`).

Attributes	Data_Type	Constraints	Key
<code>comic_character_id</code>	INT	NOT NULL AI	PRIMARY KEY
<code>comic_id</code>	INT	NOT NULL	FOREIGN KEY

character_id	INT	NOT NULL	FOREIGN KEY
--------------	-----	----------	-------------

Table 17: Comic_characters Table Details

The comic_characters table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names, atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes (comic_id and character_id) are fully functionally dependent on the primary key comic_character_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with foreign key attributes directly depending on the primary key. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (comic_character_id).

Comic_creators: It connects comic books to their creators, with attributes including the unique identifier (comic_creator_id), comic identifier (comic_id), and creator identifier (creator_id).

Attributes	Data_Type	Constraints	Key
comic_creator_id	INT	NOT NULL AI	PRIMARY KEY
comic_id	INT	NOT NULL	FOREIGN KEY
creator_id	INT	NOT NULL	FOREIGN KEY

Table 18: Comic_creators Table Details

The comic_creators table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names and atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes (comic_id and creator_id) are fully functionally dependent on the primary key comic_creator_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with comic_id and creator_id being foreign keys and not dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (comic_creator_id).

Prices: It stores the purchase and sale prices for individual comics, as well as the associated quality, with attributes such as the unique identifier (price_id), comic identifier (comic_id), purchase_price, sale_price, quality, and last_updated date.

Attributes	Data_Type	Constraints	Key
price_id	INT	NOT NULL AI	PRIMARY KEY
comic_id	INT	NOT NULL	FOREIGN KEY
purchase_price	FLOAT	NOT NULL	
sale_price	FLOAT	NOT NULL	
quality	VARCHAR(10)	NOT NULL	
last_updated	DATE	NOT NULL	

Table 19: Prices Table Details

The prices table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names and atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes are fully functionally dependent on the primary key price_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with comic_id being a foreign key and the other non-key attributes not

being dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (price_id).

Overstreet_price_guide: It holds pricing data from the Overstreet Price Guide, with attributes including the unique identifier (overstreet_price_guide_id), comic identifier (comic_id), year, and prices for various conditions (Fair_price, Good_price, Very_good_price, Fine_price, Very_fine_price, Near_mint_price, Mint_price).

Attributes	Data Type	Constraints	Key
overstreet_price_guide_id	INT	NOT NULL AI	PRIMARY KEY
comic_id	INT	NOT NULL	FOREIGN KEY
year	INT	NOT NULL	
Fair_price	FLOAT	NOT NULL	
Good_price	FLOAT	NOT NULL	
Very_good_price	FLOAT	NOT NULL	
Fine_price	FLOAT	NOT NULL	
Very_fine_price	FLOAT	NOT NULL	
Near_mint_price	FLOAT	NOT NULL	
Mint_price	FLOAT	NOT NULL	

Table 20: Overstreet_price_guide Table Details

The overstreet_price_guide table adheres to 1NF, 2NF, 3NF, and BCNF normal forms. It meets 1NF criteria with unique column names and atomic values. The table is in 2NF since it is already in 1NF and all non-key attributes are fully functionally dependent on the primary key overstreet_price_guide_id. The table is also in 3NF because it is in 2NF, and there are no transitive dependencies between non-key attributes, with comic_id being a foreign key and the other non-key attributes not being dependent on each other. Lastly, the table is in BCNF as it is in 3NF, and all functional dependencies involve a superkey (overstreet_price_guide_id).

In conclusion, based on this analysis, the Excelsior database is in BCNF. The Excelsior database's normalized structure plays a crucial role in its efficiency and effectiveness, by meeting the requirements of 1NF, 2NF, 3NF, and BCNF, the database design eliminates data redundancy, ensures data integrity, and provides a solid foundation for managing and tracking information in the dynamic comic book industry. The careful separation of tables which can be seen in Figure 2, and adherence to normalization principles contribute to a robust and versatile database that can adapt to the evolving world of comics, serving as a valuable tool for collectors, enthusiasts, and industry professionals alike.

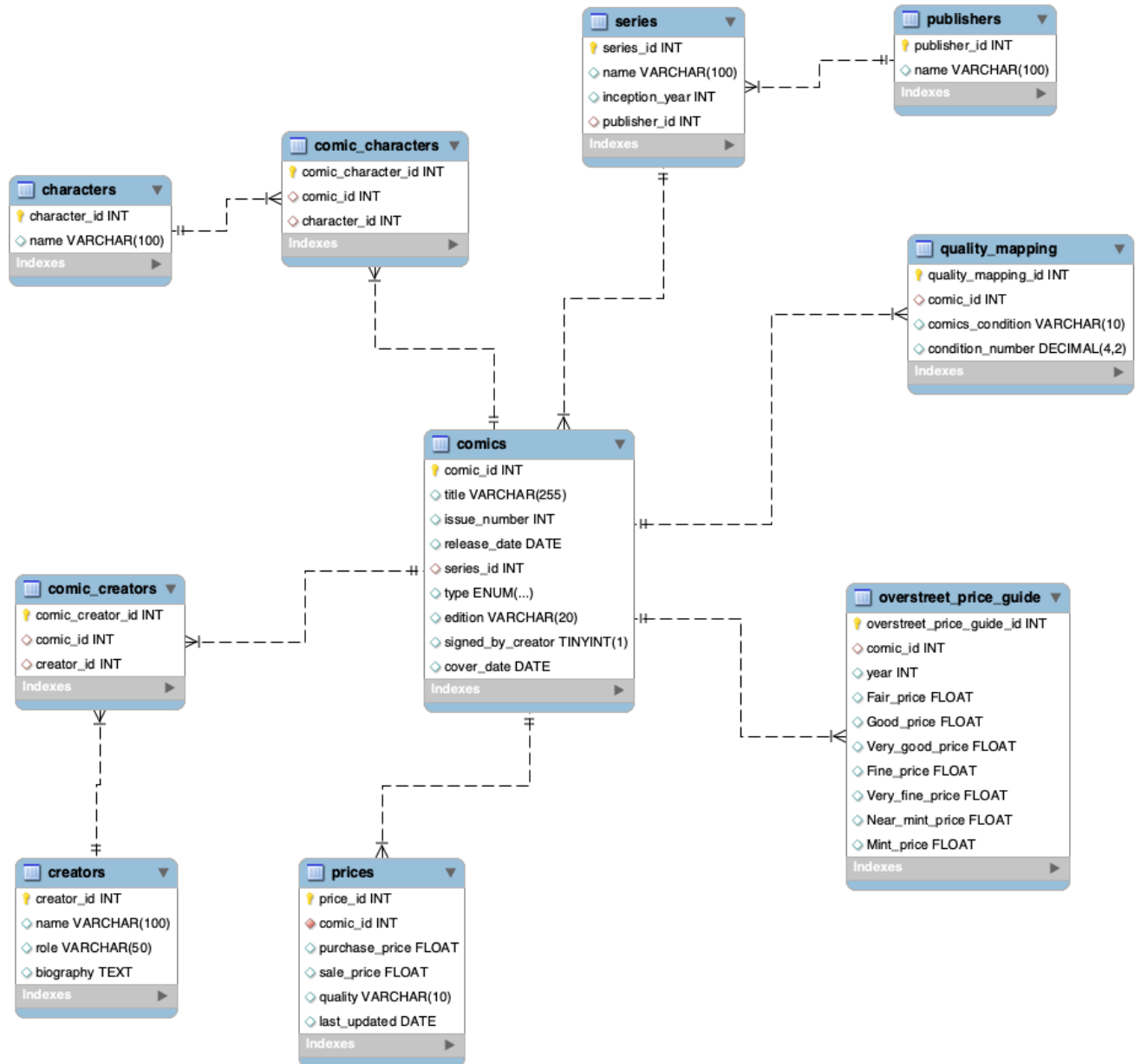


Figure 2: E-R diagram

Database View:

In this section, I have primarily created four views for the Excelsior database. These four views add significant value to the Excelsior database design, providing users with comprehensive and up-to-date information about various aspects of comics, such as their value, availability, and details. Defining these relationships as SQL views rather than tables ensures data integrity and simplifies database maintenance by avoiding redundancy and automatically updating changes in the underlying tables.

comic_value_info:

	Publisher	Title	Issue	Fair	Good	Very_Good	Fine	Very_Fine	Near_Mint	Signed
1	DC Comics	Batman (1939) (#1) (DC Comics)	1	30	40	50	60	70	80	None
2	DC Comics	Batman (1939) (#1) (DC Comics)	1	10	20	30	40	50	60	None
3	DC Comics	Batman (1939) (#16) (DC Comics)	16	20	40	60	80	100	120	VF (Signed)
4	DC Comics	Batman (1939) (#50) (DC Comics)	50	35	60	100	150	250	350	None
5	DC Comics	Batman (1939) (#85) (DC Comics)	85	35	60	100	150	250	350	None
6	Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	500	750	1000	2000	5000	10000	None
7	Marvel Comics	Daredevil (1964) (#158) (Marvel Comics)	158	20	30	50	75	90	100	None
8	Marvel Comics	Daredevil (1964) (#159) (Marvel Comics)	159	20	30	50	75	90	100	None
9	Marvel Comics	Daredevil (1964) (#160) (Marvel Comics)	160	20	30	50	75	90	100	None
10	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	30	40	50	60	70	80	None
11	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	100	150	200	300	500	750	None
12	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	75	100	150	200	250	350	None
13	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	20	30	40	60	80	120	None
14	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	30	40	50	60	70	80	None
15	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	300	400	500	600	800	1200	None

Figure 3: Comic_value_info view

Purpose and Value:

The comic_value_info view in Figure 3 provides an overview of the value of each comic in different conditions (Fair, Good, Very Good, Fine, Very Fine, Near Mint). It also indicates whether the comic has been signed by the creator. The view simplifies the retrieval of this information for users, ensuring data consistency, and providing a more efficient way to access the data. It can offer collectors, enthusiasts, and industry professionals a way to quickly assess the value of their comics, monitor market trends, and make informed decisions when purchasing new comics.

Here's an example of how the comic_value_info view can be useful:

Suppose we want to find the value of all comics from a specific publisher, say "Marvel Comics", and their value in "Near Mint" condition. Using the comic_value_info view, we can easily achieve this with a simple query in Figure 4 :

```
-- query for VIEW_1
SELECT Publisher, Title, Issue, Near_Mint
FROM comic_value_info
WHERE Publisher = 'Marvel Comics';
```

Figure 4: Query for View_1

This query will return a list of comics published by "Marvel Comics" along with their title, issue number, and value in "Near Mint" condition. This is more efficient and simpler than querying the individual tables and performing complex joins. The following Figure 5 shows the results of this query.

	Publisher	Title	Issue	Near_Mint
1	Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	10000
2	Marvel Comics	Daredevil (1964) (#158) (Marvel Comics)	158	100
3	Marvel Comics	Daredevil (1964) (#159) (Marvel Comics)	159	100
4	Marvel Comics	Daredevil (1964) (#160) (Marvel Comics)	160	100
5	Marvel Comics	Iron Man (1963) (#1) (Marvel Comics)	1	900
6	Marvel Comics	Iron Man (1963) (#2) (Marvel Comics)	2	150
7	Marvel Comics	Spider-Man (1963) (#1) (Marvel Comics)	1	20000
8	Marvel Comics	Spider-Man (1963) (#129) (Marvel Comics)	129	80
9	Marvel Comics	Spider-Man (1963) (#13) (Marvel Comics)	13	90
10	Marvel Comics	Spider-Man (1963) (#50) (Marvel Comics)	50	160
11	Marvel Comics	Spider-Man (1963) (#7) (Marvel Comics)	7	1800
12	Marvel Comics	The Amazing Spider-Man (1963) (#100) (Marvel Comics)	100	60
13	Marvel Comics	The Amazing Spider-Man (1963) (#150) (Marvel Comics)	150	75
14	Marvel Comics	The Amazing Spider-Man (1963) (#2) (Marvel Comics)	2	60
15	Marvel Comics	The Amazing Spider-Man (1963) (#3) (Marvel Comics)	3	75

Figure 5: Results for View_1

The view also ensures that the data retrieved is always up-to-date, as it is a virtual table that reflects the current state of the underlying tables. If any of the source data changes, the view will automatically reflect those changes. This is an advantage over generating a separate table that would require manual updates to maintain data consistency.

comics_availability_view:

	Issue	Release Date	Type	Edition	Cover Date	Quality	Sale Price	Status
1	1	1973-09-01	Comic	First	1973-09-01	VG	40	Available
2	1	1939-03-01	Comic	First	1939-03-01	GD	60	Available
3	10	1987-02-12	Graphic Novel	First	1987-02-12	VF	100	Available
4	50	1975-09-11	Comic	First	1975-09-11	VG	150	Available
5	85	1976-09-11	Comic	First	1976-09-11	VG	150	Available
6	1	1964-04-01	Comic	First	1964-04-01	FR	5000	Available
7	158	1980-05-01	Comic	First	1980-05-01	VG	25	Available
8	159	1980-06-01	Comic	First	1980-06-01	VG	25	Available
9	160	1980-07-01	Comic	First	1980-07-01	VG	25	Available
10	1	1994-03-23	Comic	First	1994-03-23	NM	300	Available
11	1	1993-08-01	Comic	First	1993-03-01	VF	80	Available
12	1	1994-04-23	Comic	First	1994-03-23	VF	500	Available
13	1	1994-03-01	Comic	First	1994-03-01	NM	120	Available
14	1	1994-03-01	Comic	First	1994-03-01	VF	200	Available
15	1	1994-03-09	Comic	First	1994-03-09	NM	800	Available

Figure 6: Comics_availability_view

Purpose and Value:

The comics_availability_view in Figure 6 displays availability, pricing, and other details about each comic in the database. The view is a convenient and efficient way to retrieve this information, making it easier for users to understand the availability of comics in the collection. It is very useful for collectors to manage their inventory, track the availability of desired comics in the market, and make informed decisions when buying and selling comics.

Here's an example of how the comics_availability_view can be useful:

Suppose we want to find all the comics that are available for sale and have a quality of "Very Fine" or better. We can use the comics_availability_view to achieve this with a simple query in Figure 7:

```
-- -- query for VIEW_2
SELECT Publisher, Title, Issue, Release_Date, Type, Edition, Cover_Date, Quality, Sale_Price, Status
FROM comics_availability_view
WHERE Status = 'Available' AND (Quality = 'VF' OR Quality = 'NM');
```

Figure 7: Query for View_2

This query will return a list of available comics with a quality of "Very Fine" or better, providing relevant information such as publisher, title, issue number, release date, type, edition, cover date, and sale price.

	Issue	Release Date	Type	Edition	Cover Date	Quality	Sale Price	Status
1	10	1987-02-12	Graphic Novel	First	1987-02-12	VF	100	Available
2	1	1994-03-01	Comic	First	1994-03-01	VF	200	Available
3	1	1994-03-23	Comic	First	1994-03-23	NM	300	Available
4	1	1993-08-01	Comic	First	1993-03-01	VF	80	Available
5	1	1994-03-09	Comic	First	1994-03-09	NM	800	Available
6	1	1994-03-01	Comic	First	1994-03-01	NM	120	Available
7	1	1994-04-23	Comic	First	1994-03-23	VF	500	Available
8	1	1994-05-01	Comic	First	1994-03-01	VF	300	Available
9	1	1979-10-10	Graphic Novel	First	1979-10-10	NM	350	Available
10	2	2000-03-29	Graphic Novel	First	2000-03-29	VF	80	Available
11	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available
12	1	2008-02-01	Comic	First	2008-02-01	NM	100	Available
13	1	2008-02-20	Comic	First	2008-02-20	NM	100	Available
14	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available
15	1	2014-04-09	Comic	First	2014-04-09	NM	50	Available

Figure 8: Results for View_1

Using the comics_availability_view view simplifies data retrieval and ensures data consistency across the underlying tables.

view_comic_details:

	Publisher	Title	Issue	series_name	character_name	creator_name
1	DC Comics	Batman (1939) (#1) (DC Comics)	1	Batman	Batman	Bob Kanes
2	DC Comics	Batman (1939) (#1) (DC Comics)	1	Batman	Ras al Ghul	Denny O'Neil, Neal Adams
3	DC Comics	Batman (1939) (#16) (DC Comics)	16	Batman	Superman	Jerry Siegel, Joe Shuster
4	DC Comics	Batman (1939) (#50) (DC Comics)	50	Batman	Batman	Denny O'Neil, Neal Adams
5	DC Comics	Batman (1939) (#85) (DC Comics)	85	Batman	Batman	Denny O'Neil, Neal Adams
6	Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	Daredevil	Daredevil, Foggy Nelson, Karen Page	Bill Everett, Jack Kirby, Stan Lee
7	Marvel Comics	Daredevil (1964) (#158) (Marvel Comics)	158	Daredevil	Daredevil	Frank Miller
8	Marvel Comics	Daredevil (1964) (#159) (Marvel Comics)	159	Daredevil	Daredevil	Gerry Conway
9	Marvel Comics	Daredevil (1964) (#160) (Marvel Comics)	160	Daredevil	Daredevil	Len Wein
10	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola
11	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola
12	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola
13	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola
14	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola
15	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	Hellboy	Mike Mignola

Figure 9: View_comic_details

Purpose and Value:

The view_comic_details view in Figure 9 provides a detailed overview of each comic, including information about its publisher, series, characters, and creators. This view consolidates data from multiple tables and makes it easier to retrieve comprehensive comic details in a single query. It is very useful for collectors, enthusiasts, and industry professionals to understand the background, artistic contributions, and characters covered by each comic. Here's an example of how the view_comic_details can be useful:

Suppose we want to find all the comics in which a specific character, say "Batman", appears.

We can use the view_comic_details to achieve this with a simple query in Figure 10:

```
-- -- query for VIEW_3
SELECT Publisher, Title, Issue, series_name, character_name, creator_name
FROM view_comic_details
WHERE character_name LIKE '%Batman%';
```

Figure 10: Query for View_3

This query will return a list of comics featuring Batman, along with their respective publisher, title, issue number, series name, character names, and creator names. The following Figure 11 shows the results of this query.

	Publisher	Title	Issue	series_name	character_name	creator_name
1	DC Comics	Batman (1939) (#1) (DC Comics)	1	Batman	Batman	Bob Kanes
2	DC Comics	Batman (1939) (#50) (DC Comics)	50	Batman	Batman	Denny O'Neil, Neal Adams
3	DC Comics	Batman (1939) (#85) (DC Comics)	85	Batman	Batman	Denny O'Neil, Neal Adams
4	DC Comics	Wonder Woman (1941) (#9) (DC Comics)	9	Wonder Woman	Batman	Bill Finger, Bob Kanes

Figure 11: Results for View_1

The view_comic_details view simplifies the process of retrieving this information and ensures data consistency across the underlying tables.

top_10_comics_by_sale_price:

	Publisher	Title	Issue	Series_Name	Sale_Price	Signed
1	Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	25000	Not Signed
2	Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	Daredevil	5000	Not Signed
3	Marvel Comics	Spider-Man (1963) (#1) (Marvel Comics)	1	Spider-Man	5000	Not Signed
4	Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	1000	Not Signed
5	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	800	Not Signed
6	Vertigo	Sandman (1989) (#1) (Vertigo)	1	Sandman	600	Not Signed
7	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	500	Not Signed
8	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	Watchmen	500	Not Signed
9	Marvel Comics	Spider-Man (1963) (#7) (Marvel Comics)	7	Spider-Man	500	Not Signed
10	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	Watchmen	400	Not Signed

Figure 12: Top_10_comics_by_sale_price

Purpose and Value:

The top_10_comics_by_sale_price view in Figure 12 displays the top ten most valuable comics based on sale price. This view helps collectors, enthusiasts, and industry professionals

identify high-value comics, monitor market trends, and make informed decisions when investing in rare or collectible comics.

Here's an example of how the `top_10_comics_by_sale_price` view can be useful:

Suppose we want to quickly identify the most valuable comics in your collection to showcase them in a special exhibition or advertise them for sale. we can use the `top_10_comics_by_sale_price` view to achieve this with a simple query in Figure 13:

```
-- query for VIEW_4
SELECT Publisher, Title, Issue, Series_Name, Sale_Price, Signed
FROM top_10_comics_by_sale_price;
```

Figure 13: Query for View_4

This query returns a list of the top 10 most valuable comics, along with their publisher, title, issue number, series name, sale price, and signed status. The following Figure 14 shows the results of this query.

	Publisher	Title	Issue	Series_Name	Sale_Price	Signed
1	Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	25000	Not Signed
2	Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	Daredevil	5000	Not Signed
3	Marvel Comics	Spider-Man (1963) (#1) (Marvel Comics)	1	Spider-Man	5000	Not Signed
4	Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	1000	Not Signed
5	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	800	Not Signed
6	Vertigo	Sandman (1989) (#1) (Vertigo)	1	Sandman	600	Not Signed
7	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	500	Not Signed
8	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	Watchmen	500	Not Signed
9	Marvel Comics	Spider-Man (1963) (#7) (Marvel Comics)	7	Spider-Man	500	Not Signed
10	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	Watchmen	400	Not Signed

Figure 14: Results for View_1

`top_10_comics_by_sale_price` view simplifies the process of identifying the most valuable comics and ensures data consistency across the underlying tables.

Overall, the use of these views significantly enhances the utility and efficiency of the Excelsior database, making it a powerful tool for collectors, enthusiasts, and industry professionals alike.

Procedural Elements:

In this design, we have implemented several PL/SQL procedures and a database trigger to enhance the functionality and maintain the integrity of the database. These procedural elements help automate various tasks, reduce code duplication, and ensure data consistency. Here are the detailed description of each procedural element:

Function:

GetSeriesIdByName: This function takes a series name as input and returns its corresponding `series_id` from the series table. It is useful when inserting a comic with its series name, which helps when inserting comics to convert series names into series IDs for insertion in Figure 15 into the comics table. This can help to reduce redundant data which we can see in Figure 16.


```
-- Insert comics
INSERT INTO comics (title, issue_number, release_date, series_id, type, edition, signed_by_creator, cover_date)
VALUES ('The Incredible Hulk', 1, '1962-05-01', GetSeriesIdByName('The Incredible Hulk'), 'Comic', 'First', FALSE, '1962-05-01'),
('Batman', 1, '1939-03-01', GetSeriesIdByName('Batman'), 'Comic', 'First', FALSE, '1939-03-01'),
('Spawn', 1, '1992-05-01', GetSeriesIdByName('Spawn'), 'Comic', 'First', FALSE, '1992-05-01'),
('Hellboy', 1, '1993-08-01', GetSeriesIdByName('Hellboy'), 'Comic', 'First', FALSE, '1993-03-01'),
('Batman', 1, '1973-09-01', GetSeriesIdByName('Batman'), 'Comic', 'First', FALSE, '1973-09-01');
```

Figure 15: Insert comic data

comic_id	title	issue_number	release_date	series_id	type	edition	signed_by_creator	cover_date
1	The Incredible Hulk	1	1962-05-01	1	Comic	First		0 1962-05-01
2	Batman	1	1939-03-01	2	Comic	First		0 1939-03-01
3	Spawn	1	1992-05-01	3	Comic	First		0 1992-05-01
4	Hellboy	1	1993-08-01	4	Comic	First		0 1993-03-01
5	Batman	1	1973-09-01	2	Comic	First		0 1973-09-01
6	Amazing Spider-Man	50	1967-07-01	5	Comic	First		0 1967-07-01
7	Teenage Mutant Ninja Turtles	1	2011-08-24	6	Comic	First		0 2011-08-24
8	Watchmen	1	1986-09-01	7	Comic	First		0 1986-09-01
9	Hellboy: Seed of Destruction	1	1994-03-01	4	Comic	First		0 1994-03-01
10	Daredevil	1	1964-04-01	8	Comic	First		0 1964-04-01

Figure 16: GetSeriesIdByName's Effect

Procedures:

insert_publisher_if_not_exists: This procedure checks if a publisher with the given name already exists in the publishers table. If not, it inserts a new publisher with the provided name in Figure 17. This reduces data duplication and ensures that a publisher is only inserted once. After I had inserted the data for all the comics, there were nine different publishers in Figure 18.

```
-- Insert publishers

-- Call the stored procedure for each publisher in the list
CALL insert_publisher_if_not_exists( publisher_name: 'Marvel Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'DC Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'Image Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'Dark Horse Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'DC Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'Marvel Comics');
CALL insert_publisher_if_not_exists( publisher_name: 'IDW Publishing');
CALL insert_publisher_if_not_exists( publisher_name: 'DC Comics');
```

Figure 17: Call insert_publisher_if_not_exists

This facilitates the redundancy of data when subsequently associating series and publishers in the later insert process.

publisher_id	name
1	Marvel Comics
2	DC Comics
3	Image Comics
4	Dark Horse Comics
5	IDW Publishing
6	Vertigo
7	Boom! Studios
8	Valiant Comics
9	Archie Comics

Figure 18: publisher data

insert_series_if_not_exists: Similar to the previous procedure in Figure 19, this one checks for the existence of a series with the given name, inception year, and publisher_id. If it does not exist, it inserts a new record into the series table. This procedure helps reduce data duplication and ensures that a series is only inserted once.

```
-- Insert series
CALL insert_series_if_not_exists( series_name: 'The Incredible Hulk', inception_year: 1962, publisher_name: 'Marvel Comics');
CALL insert_series_if_not_exists( series_name: 'Batman', inception_year: 1939, publisher_name: 'DC Comics');
CALL insert_series_if_not_exists( series_name: 'Spawn', inception_year: 1992, publisher_name: 'Image Comics');
CALL insert_series_if_not_exists( series_name: 'Hellboy', inception_year: 1993, publisher_name: 'Dark Horse Comics');
CALL insert_series_if_not_exists( series_name: 'Batman', inception_year: 1940, publisher_name: 'DC Comics');
CALL insert_series_if_not_exists( series_name: 'Spider-Man', inception_year: 1963, publisher_name: 'Marvel Comics');
CALL insert_series_if_not_exists( series_name: 'Teenage Mutant Ninja Turtles', inception_year: 1984, publisher_name: 'IDW Publishing');
CALL insert_series_if_not_exists( series_name: 'Watchmen', inception_year: 1986, publisher_name: 'DC Comics');
CALL insert_series_if_not_exists( series_name: 'Hellboy', inception_year: 1993, publisher_name: 'Dark Horse Comics');
```

Figure 19: Call insert_series_if_not_exists

The series corresponding to all the comic data I have inserted are 22 strips in total in Figure 20, and they all have corresponding publishers. Of the nine publishers mentioned above, they may have published more than one series each.

	series_id	name	inception_year	publisher_id
1	1	The Incredible Hulk	1962	1
2	2	Batman	1939	2
3	3	Spawn	1992	3
4	4	Hellboy	1993	4
5	5	Spider-Man	1963	1
6	6	Teenage Mutant Ninja Turtles	1984	5
7	7	Watchmen	1986	2
8	8	Daredevil	1964	1
9	9	Locke & Key	2008	5
10	10	X-Men	1963	1
11	11	Preacher	1995	6
12	12	Lumberjanes	2014	7
13	13	X-O Manowar	1992	8
14	14	Sabrina the Teenage Witch	2019	9
15	15	Sin City	1991	4
16	16	Sandman	1989	6
17	17	The Invincible Iron Man	1968	1
18	18	The Amazing Spider-Man	1963	1
19	19	Superman	1939	2
20	20	Wonder Woman	1941	2
21	21	Iron Man	1963	1
22	22	Thor	1966	1

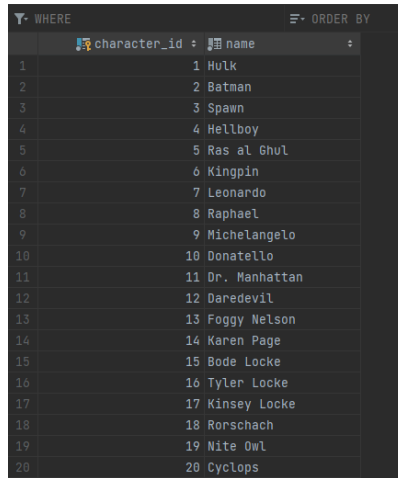
Figure 20: Series data

insert_character_if_not_exists: This procedure in Figure 21 inserts a new character into the characters table if it does not already exist. It checks for the existence of a character with the given name and inserts a new record if not found. This ensures that characters are only inserted once, reducing data duplication.

```
-- Insert characters
CALL insert_character_if_not_exists( character_name: 'Hulk');
CALL insert_character_if_not_exists( character_name: 'Batman');
CALL insert_character_if_not_exists( character_name: 'Spawn');
CALL insert_character_if_not_exists( character_name: 'Hellboy');
CALL insert_character_if_not_exists( character_name: 'Ras al Ghul');
CALL insert_character_if_not_exists( character_name: 'Kingpin');
CALL insert_character_if_not_exists( character_name: 'Leonardo');
CALL insert_character_if_not_exists( character_name: 'Raphael');
CALL insert_character_if_not_exists( character_name: 'Michelangelo');
CALL insert_character_if_not_exists( character_name: 'Donatello');
CALL insert_character_if_not_exists( character_name: 'Dr. Manhattan');
CALL insert_character_if_not_exists( character_name: 'Hellboy');
CALL insert_character_if_not_exists( character_name: 'Daredevil');
CALL insert_character_if_not_exists( character_name: 'Foggy Nelson');
CALL insert_character_if_not_exists( character_name: 'Karen Page');
CALL insert_character_if_not_exists( character_name: 'Bode Locke');
CALL insert_character_if_not_exists( character_name: 'Tyler Locke');
CALL insert_character_if_not_exists( character_name: 'Kinsey Locke');
CALL insert_character_if_not_exists( character_name: 'Rorschach');
CALL insert_character_if_not_exists( character_name: 'Dr. Manhattan');
```

Figure 21: CALL insert_character_if_not_exists

In my comic data, I have inserted a total of 43 characters from each comic's main characters and I only show some of them here. I actually inserted more comic data than character data. This is because during the insertion process, the characters in a series of comics may be the same character or more than one of the same characters. This is very effective in avoiding duplication of characters, especially when associating relationships characters with comics later on, which can be seen in the Figure 22.



	character_id	name
1	1	Hulk
2	2	Batman
3	3	Spawn
4	4	Hellboy
5	5	Ras al Ghul
6	6	Kingpin
7	7	Leonardo
8	8	Raphael
9	9	Michelangelo
10	10	Donatello
11	11	Dr. Manhattan
12	12	Daredevil
13	13	Foggy Nelson
14	14	Karen Page
15	15	Bode Locke
16	16	Tyler Locke
17	17	Kinsey Locke
18	18	Rorschach
19	19	Nite Owl
20	20	Cyclops

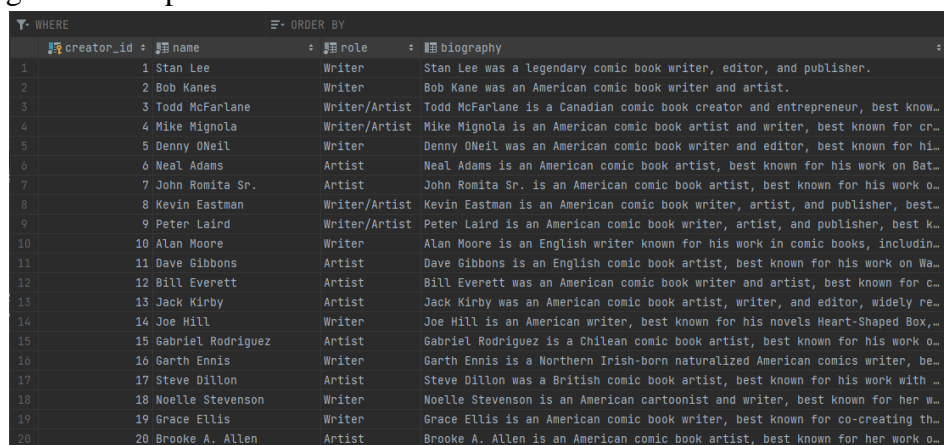
Figure 22: Part of character data

insert_if_not_exists_creators: This procedure in Figure 23 checks for the existence of a creator with the given name and role in the creators table. If not found, it inserts a new record with the provided name, role, and biography. This reduces data duplication and ensures that a creator is only inserted once.

```
-- Insert creators
CALL insert_if_not_exists_creators(creator_name: 'Stan Lee', creator_role: 'Writer', creator_bio: 'Stan Lee was a legendary comic book writer, editor, and publisher. ');
CALL insert_if_not_exists_creators(creator_name: 'Bob Kane', creator_role: 'Writer', creator_bio: 'Bob Kane was an American comic book writer and artist. ');
CALL insert_if_not_exists_creators(creator_name: 'Todd McFarlane', creator_role: 'Writer/Artist', creator_bio: 'Todd McFarlane is a Canadian comic book creator and entrepreneur, best known for his work on The Incredibles. ');
CALL insert_if_not_exists_creators(creator_name: 'Mike Mignola', creator_role: 'Writer/Artist', creator_bio: 'Mike Mignola is an American comic book artist and writer, best known for creating the dark fantasy comic book series Hellboy. ');
CALL insert_if_not_exists_creators(creator_name: 'Denny O'Neil', creator_role: 'Writer', creator_bio: 'Denny O'Neil was an American comic book writer and editor, best known for his work on Batman, Green Lantern, and Superman. ');
CALL insert_if_not_exists_creators(creator_name: 'Neal Adams', creator_role: 'Artist', creator_bio: 'Neal Adams is an American comic book artist, best known for his work on Batman, X-Men, and Green Lantern. ');
CALL insert_if_not_exists_creators(creator_name: 'John Romita Sr.', creator_role: 'Artist', creator_bio: 'John Romita Sr. is an American comic book artist, best known for his work on Marvel Comics characters like Spider-Man and Iron Man. ');
CALL insert_if_not_exists_creators(creator_name: 'Kevin Eastman', creator_role: 'Writer/Artist', creator_bio: 'Kevin Eastman is an American comic book writer, artist, and publisher, best known for co-creating the comic book series Teenage Mutant Ninja Turtles. ');
CALL insert_if_not_exists_creators(creator_name: 'Peter Laird', creator_role: 'Writer/Artist', creator_bio: 'Peter Laird is an American comic book writer, artist, and publisher, best known for co-creating the comic book series Teenage Mutant Ninja Turtles. ');
CALL insert_if_not_exists_creators(creator_name: 'Alan Moore', creator_role: 'Writer', creator_bio: 'Alan Moore is an English writer known for his work in comic books, including Watchmen, V for Vendetta, and Swamp Thing. ');
CALL insert_if_not_exists_creators(creator_name: 'Dave Gibbons', creator_role: 'Artist', creator_bio: 'Dave Gibbons is an English comic book artist, best known for his work on Watchmen and Green Lantern. ');
CALL insert_if_not_exists_creators(creator_name: 'Mike Mignola', creator_role: 'Writer/Artist', creator_bio: 'Mike Mignola is an American comic book artist and writer, best known for creating the character Hellboy. ');
CALL insert_if_not_exists_creators(creator_name: 'Stan Lee', creator_role: 'Writer', creator_bio: 'Stan Lee was an American comic book writer, editor, publisher, and producer, best known as the co-creator of the Marvel Comics universe. ');
```

Figure 23: Call insert_if_not_exists_creators

There are 41 authors in Figure 24 involved in my comic data, and I only show some of them here. Similar to the character insertion above, this effectively avoids duplicate data when associating relationships between comics and creators.



	creator_id	name	role	biography
1	1	Stan Lee	Writer	Stan Lee was a legendary comic book writer, editor, and publisher.
2	2	Bob Kane	Writer	Bob Kane was an American comic book writer and artist.
3	3	Todd McFarlane	Writer/Artist	Todd McFarlane is a Canadian comic book creator and entrepreneur, best known for his work on The Incredibles.
4	4	Mike Mignola	Writer/Artist	Mike Mignola is an American comic book artist and writer, best known for creating the dark fantasy comic book series Hellboy.
5	5	Denny O'Neil	Writer	Denny O'Neil was an American comic book writer and editor, best known for his work on Batman, Green Lantern, and Superman.
6	6	Neal Adams	Artist	Neal Adams is an American comic book artist, best known for his work on Batman, X-Men, and Green Lantern.
7	7	John Romita Sr.	Artist	John Romita Sr. is an American comic book artist, best known for his work on Marvel Comics characters like Spider-Man and Iron Man.
8	8	Kevin Eastman	Writer/Artist	Kevin Eastman is an American comic book writer, artist, and publisher, best known for co-creating the comic book series Teenage Mutant Ninja Turtles.
9	9	Peter Laird	Writer/Artist	Peter Laird is an American comic book writer, artist, and publisher, best known for co-creating the comic book series Teenage Mutant Ninja Turtles.
10	10	Alan Moore	Writer	Alan Moore is an English writer known for his work in comic books, including Watchmen, V for Vendetta, and Swamp Thing.
11	11	Dave Gibbons	Artist	Dave Gibbons is an English comic book artist, best known for his work on Watchmen and Green Lantern.
12	12	Bill Everett	Artist	Bill Everett was an American comic book writer and artist, best known for his work on Marvel Comics characters like the Hulk and Iron Man.
13	13	Jack Kirby	Artist	Jack Kirby was an American comic book artist, writer, and editor, widely regarded as one of the most influential comic book creators.
14	14	Joe Hill	Writer	Joe Hill is an American writer, best known for his novels Heart-Shaped Box and Locke & Key.
15	15	Gabriel Rodriguez	Artist	Gabriel Rodriguez is a Chilean comic book artist, best known for his work on Marvel Comics characters like Iron Man and the X-Men.
16	16	Garth Ennis	Writer	Garth Ennis is a Northern Irish-born naturalized American comics writer, best known for his work on Marvel Comics characters like Iron Man and the X-Men.
17	17	Steve Dillon	Artist	Steve Dillon was a British comic book artist, best known for his work with Marvel Comics on the X-Men and Iron Man.
18	18	Noelle Stevenson	Writer	Noelle Stevenson is an American cartoonist and writer, best known for her work on the comic book series Nimona.
19	19	Grace Ellis	Writer	Grace Ellis is an American comic book writer, best known for co-creating the comic book series Teenage Mutant Ninja Turtles.
20	20	Brooke A. Allen	Artist	Brooke A. Allen is an American comic book artist, best known for her work on Marvel Comics characters like Iron Man and the X-Men.

Figure 24: Part of creators data

insert_comic_character_if_not_exists: This procedure in Figure 25 inserts a new record into the comic_characters table, linking a comic with its character. It takes comic_id and character_name as inputs and inserts a new record, ensuring that the relationship between comics and characters is properly maintained.

```
-- Insert character_comics

call insert_comic_character_if_not_exists( comic_id: 1, character_name: 'Hulk');
call insert_comic_character_if_not_exists( comic_id: 2, character_name: 'Batman');
call insert_comic_character_if_not_exists( comic_id: 3, character_name: 'Spawn');
call insert_comic_character_if_not_exists( comic_id: 4, character_name: 'Hellboy');
call insert_comic_character_if_not_exists( comic_id: 5, character_name: 'Ras al Ghul');
call insert_comic_character_if_not_exists( comic_id: 6, character_name: 'Kingpin');
call insert_comic_character_if_not_exists( comic_id: 7, character_name: 'Leonardo');
call insert_comic_character_if_not_exists( comic_id: 7, character_name: 'Raphael');
call insert_comic_character_if_not_exists( comic_id: 7, character_name: 'Michelangelo');
call insert_comic_character_if_not_exists( comic_id: 7, character_name: 'Donatello');
call insert_comic_character_if_not_exists( comic_id: 8, character_name: 'Dr. Manhattan');
call insert_comic_character_if_not_exists( comic_id: 9, character_name: 'Hellboy');
call insert_comic_character_if_not_exists( comic_id: 10, character_name: 'Daredevil');
```

Figure 25: Call insert_comic_character_if_not_exists

In comics, multiple comics from the same series are associated to the same character or multiple characters in one comic. In the table above where characters are stored, duplicates have been removed so that when associating a comic with a character, the character's ID in the character table can be found by the character's name to avoid the insertion of duplicate association information which can be seen in Figure 26.

	comic_character_id	comic_id	character_id
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	7	8
9	9	7	9
10	10	7	10
11	11	8	11
12	12	9	4
13	13	10	12
14	14	10	13
15	15	10	14
16	16	11	15
17	17	11	16
18	18	11	17
19	19	12	18
20	20	12	11
21	21	12	19
22	22	13	20
23	23	13	21
24	24	13	22
25	25	14	4

Figure 26: Part of comic_character data

insert_comic_creator_if_not_exists: This procedure in Figure 27 performs a similar function as the previous one but links comics with their creators. It takes comic_id and creator_name as inputs and inserts a new record into the comic_creators table, maintaining the relationship between comics and creators.

```
-- Insert comic_creators

call insert_comic_creator_if_not_exists( comic_id: 1 , creator_name: 'Stan Lee');
call insert_comic_creator_if_not_exists( comic_id: 2 , creator_name: 'Bob Kanes');
call insert_comic_creator_if_not_exists( comic_id: 3 , creator_name: 'Todd McFarlane');
call insert_comic_creator_if_not_exists( comic_id: 4 , creator_name: 'Mike Mignola');
call insert_comic_creator_if_not_exists( comic_id: 5 , creator_name: 'Denny ONeil');
call insert_comic_creator_if_not_exists( comic_id: 5 , creator_name: 'Neal Adams');
call insert_comic_creator_if_not_exists( comic_id: 6 , creator_name: 'Stan Lee');
call insert_comic_creator_if_not_exists( comic_id: 6 , creator_name: 'John Romita Sr. ');
call insert_comic_creator_if_not_exists( comic_id: 7 , creator_name: 'Kevin Eastman');
call insert_comic_creator_if_not_exists( comic_id: 7 , creator_name: 'Peter Laird');
call insert_comic_creator_if_not_exists( comic_id: 8 , creator_name: 'Alan Moore');
call insert_comic_creator_if_not_exists( comic_id: 8 , creator_name: 'Dave Gibbons');
call insert_comic_creator_if_not_exists( comic_id: 9 , creator_name: 'Mike Mignola');
call insert_comic_creator_if_not_exists( comic_id: 10 , creator_name: 'Stan Lee');
call insert_comic_creator_if_not_exists( comic_id: 10 , creator_name: 'Bill Everett');
```

Figure 27: Call insert_comic_creator_if_not_exists

Similar to the aforementioned connection between the comic and the character. Doing so reduces the repetition of the insertion of information associated between the comic and the creator, which we can see in Figure 28 .

	comic_creator_id	comic_id	creator_id
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	5	6
7	7	6	1
8	8	6	7
9	9	7	8
10	10	7	9
11	11	8	10
12	12	8	11
13	13	9	4
14	14	10	1
15	15	10	12
16	16	10	13
17	17	11	14
18	18	11	15
19	19	12	10
20	20	12	11
21	21	13	1
22	22	13	13
23	23	14	4

Figure 28: Part of comic_creator data

Sale price update procedures:

update_sale_price: This procedure calculates the sale price of a comic based on its condition and whether it is signed. It takes the comic_id as input and updates the sale_price field in the prices table accordingly.

```
-- 1.TEST PROCEDURE update_sale_price and see the effect of the trigger
-- Before update
select * from prices where comic_id = 1;

call update_sale_price( comic_id_param: 1);

-- After update
select * from prices where comic_id = 1;
```

Figure 29: TEST PROCEDURE update_sale_price

Each comic has a price at the time of purchase, i.e. a purchase price, and also a recommended sale price. But the actual selling price of a comic is often determined by a number of factors. Only the quality of the comic and whether it is signed by the creator are taken into account here. The quality guide prices are detailed in the overstreet_price_guide table. If the comic is signed by the author then its value will be multiplied by 10.

I will update the sale price of comic books with ID 1 to follow the rules above in Figure 29. Let's take a look at the prices before the update in Figure 30.

price_id	comic_id	purchase_price	sale_price	quality	last_updated
1	1	20	25	VG	2023-05-01

Figure 30: The prices before the update

Let's take a look at the updated prices.

price_id	comic_id	purchase_price	sale_price	quality	last_updated
1	1	20	20	VG	2023-05-06

Figure 31: The updated prices.

Due to the quality of this comic, he may not be able to buy it at a very high price and has not been signed by the creator. Therefore the price here corresponds to the corresponding price according to its quality, even if the comic book is sold without profit.

update_sale_prices_in_range: This procedure updates the sale prices of all comics in the library by calling the update_sale_price procedure for each comic. It iterates through all the comic_ids and ensures that their sale prices are up-to-date.

```
-- 2.TEST PROCEDURE update_sale_prices_in_range

call update_sale_prices_all();
```

Figure 32: TEST PROCEDURE update_sale_prices_in_range

I will update the sale prices of all comics in Figure 32.

Before update In Figure 33:

IDW Comics		IDW - ORDER BY Publisher									
ID	Publisher	Title	Issue	Release Date	Type	Edition	Cover Date	Quality	Sale Price	Status	
1	Archie Comics	Sabrina the Teenage Witch (2019) (#1) (Archie Comics)	1	2019-03-27	Comic	First	2019-03-27	NM	20	Available	
2	Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	1	2014-04-09	Comic	First	2014-04-09	VF/NM	40	Available	
3	Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	1	2014-04-09	Comic	First	2014-04-09	NM	50	Available	
4	Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	1	2014-04-09	Comic	First	2014-04-09	NM	40	Available	
5	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-04-23	Comic	First	1994-03-23	VF	500	Available	
6	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-09	Comic	First	1994-03-09	NM	800	Available	
7	Dark Horse Comics	Sin City (1991) (#1) (Dark Horse Comics)	1	1991-04-01	Comic	First	1991-04-01	VF	200	Available	
8	Dark Horse Comics	Hellboy (1993) (#6) (Dark Horse Comics)	6	1994-05-01	Comic	First	1994-03-01	VF	300	Available	
9	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-01	Comic	First	1994-03-01	NM	120	Available	
10	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-01	Comic	First	1994-03-01	VF	200	Available	
11	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-23	Comic	First	1994-03-23	NM	300	Available	
12	Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1993-08-01	Comic	First	1993-03-01	VF	80	Available	
13	DC Comics	Superman (1939) (#15) (DC Comics)	15	2003-05-21	Graphic Novel	Reprint	2003-05-21	NM	300	Available	
14	DC Comics	Batman (1939) (#85) (DC Comics)	85	1976-09-11	Comic	First	1973-09-11	VG	150	Available	
15	DC Comics	Wonder Woman (1941) (#9) (DC Comics)	9	2016-04-06	Graphic Novel	First	2016-04-06	VF	200	Available	
16	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	VF	100	Available	
17	DC Comics	Batman (1939) (#50) (DC Comics)	50	1975-09-11	Comic	First	1973-09-11	VG	150	Available	
18	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	VF	500	Available	
19	DC Comics	Superman (1939) (#20) (DC Comics)	20	2003-09-17	Graphic Novel	First	2003-09-17	VG	80	Available	
20	DC Comics	Batman (1939) (#1) (DC Comics)	1	1973-09-01	Comic	First	1973-09-01	VG	40	Available	
21	DC Comics	Batman (1939) (#16) (DC Comics)	16	1987-02-12	Graphic Novel	First	1987-02-12	VF	100	Available	
22	DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	NM	400	Available	
23	DC Comics	Wonder Woman (1941) (#12) (DC Comics)	12	1987-02-01	Graphic Novel	First	1987-02-01	NM	400	Available	
24	DC Comics	Batman (1939) (#1) (DC Comics)	1	1939-03-01	Comic	First	1939-03-01	GO	60	Available	
25	IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-01	Comic	First	2008-02-01	NM	100	Available	
26	IDW Publishing	Teenage Mutant Ninja Turtles (1984) (#1) (IDW Publishing)	1	2011-08-24	Comic	First	2011-08-24	VF/NM	40	Available	
27	IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available	
28	IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	100	Available	
29	IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available	
30	IDW Publishing	Teenage Mutant Ninja Turtles (1984) (#1) (IDW Publishing)	1	2011-08-24	Comic	First	2011-08-24	VF/NM	25	Available	

Figure 33: Before update data

After update in Figure 34:

ORDER BY Publisher										
Publisher	Title	Issue	Release Date	Type	Edition	Cover Date	Quality	Sale Price	Status	
Archae Comics	Sabrina the Teenage Witch (2019) (#1) (Archae Comics)	1	2019-03-27	Comic	First	2019-03-27	NN	30 Available		
Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	1	2014-06-09	Comic	First	2014-06-09	VF/NN	100 Available		
Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	2	2014-06-09	Comic	First	2014-06-09	NN	75 Available		
Boom! Studios	Lumberjanes (2014) (#1) (Boom! Studios)	3	2014-06-09	Comic	First	2014-06-09	NN	75 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-23	Comic	First	1994-03-23	VF	500 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-09	Comic	First	1994-03-09	NN	1200 Available		
Dark Horse Comics	Sin City (1991) (#1) (Dark Horse Comics)	1	1991-04-01	Comic	First	1991-04-01	VF	200 Available		
Dark Horse Comics	Hellboy (1993) (#6) (Dark Horse Comics)	6	1994-05-01	Comic	First	1994-03-01	VF	450 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-01	Comic	First	1994-03-01	NN	80 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-01	Comic	First	1994-03-01	VF	80 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1994-03-23	Comic	First	1994-03-23	NN	350 Available		
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	1993-08-01	Comic	First	1993-03-01	VF	70 Available		
DC Comics	Superman (1939) (#15) (DC Comics)	15	2003-05-21	Graphic Novel	Reprint	2003-05-21	NN	720 Available		
DC Comics	Batman (1939) (#85) (DC Comics)	85	1976-09-11	Comic	First	1973-09-11	VG	100 Available		
DC Comics	Wonder Woman (1941) (#9) (DC Comics)	9	2010-04-06	Graphic Novel	First	2010-04-06	VF	2000 Available		
DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	VF	70 Available		
DC Comics	Batman (1939) (#50) (DC Comics)	50	1975-09-11	Comic	First	1973-09-11	VG	100 Available		
DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	VF	1200 Available		
DC Comics	Superman (1939) (#20) (DC Comics)	20	2003-09-17	Graphic Novel	First	2003-09-17	VF	600 Available		
DC Comics	Batman (1939) (#1) (DC Comics)	1	1973-09-01	Comic	First	1973-09-01	VG	30 Available		
DC Comics	Batman (1939) (#16) (DC Comics)	16	1987-02-12	Graphic Novel	First	1987-02-12	VF	1000 Available		
DC Comics	Watchmen (1986) (#1) (DC Comics)	1	1986-09-01	Comic	First	1986-09-01	NN	300 Available		
DC Comics	Wonder Woman (1941) (#12) (DC Comics)	12	1987-02-01	Graphic Novel	First	1987-02-01	NN	1080 Available		
DC Comics	Batman (1939) (#1) (DC Comics)	1	1939-03-01	Comic	First	1939-03-01	GD	40 Available		
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-01	Comic	First	2008-02-01	NN	75 Available		
IDW Publishing	Teenage Mutant Ninja Turtles (1984) (#1) (IDW Publishing)	1	2011-08-24	Comic	First	2011-08-24	VF/NN	35 Available		
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NN	200 Available		
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NN	150 Available		
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NN	120 Available		
IDW Publishing	Teenage Mutant Ninja Turtles (1984) (#1) (IDW Publishing)	1	2011-08-24	Comic	First	2011-08-24	VF/NN	50 Available		

Figure 34: After update:

We can see that the sale prices of all the comics have been changed. Then the next step is to make a profit calculation for the comics.

Profit and Loss calculation:

CalculateProfitLoss: This procedure calculates the profit or loss for each comic by subtracting its `purchase_price` from its `sale_price`. It displays the results in a tabular format with the `title`, `issue_number`, `purchase_price`, `sale_price`, and `profit_loss` columns.

```
-- 3.TEST PROCEDURE CalculateProfitLoss
call CalculateProfitLoss();
```

Figure 35: TEST PROCEDURE CalculateProfitLoss

In the process of the above, we have updated the selling price of all the comics. Below we will calculate the profit for each comic sold in Figure 35.

title	issue_number	purchase_price	sale_price	profit_loss
The Incredible Hulk	1	20	20	0
Batman	1	50	40	-10
Spawn	1	80	100	20
Hellboy	1	50	70	20
Batman	1	20	30	10
Amazing Spider-Man	50	100	80	-20
Teenage Mutant Ninja Turtles	1	20	35	15
Watchmen	1	50	70	20
Hellboy: Seed of Destruction	1	60	80	20
Daredevil	1	1000	500	-500
Locke & Key: Welcome to Lovecraft	1	40	120	80
Watchmen	1	250	1200	950
X-Men	1	500	2500	2000
Hellboy	6	150	450	300
Preacher	1	50	100	50
Locke & Key	1	40	200	160
Lumberjanes	1	20	100	80
Hellboy	1	200	350	150
X-O Manowar	1	80	200	120
Teenage Mutant Ninja Turtles	1	10	50	40
Hellboy: Seed of Destruction	1	500	1200	700
X-O Manowar	1	25	125	100
Lumberjanes	1	20	75	55
Sabrina the Teenage Witch	1	10	30	20
Locke & Key	1	50	150	100
Hellboy	1	200	500	300

Figure 36: Results for CalculateProfitLoss

Not every comic in Figure 36 is actually profitable when sold, but fortunately most of them are.

Trigger update_last_updated:

This trigger is executed before updating the prices table. It checks if the sale_price has changed, and if so, sets the last_updated field to the current system time. This trigger ensures that the last_updated field is always up-to-date and accurately reflects when the sale_price was last modified.

```
-- 4. TEST TRIGGER
UPDATE prices
SET last_updated = '2022-05-01'
WHERE comic_id = 1;

SELECT * FROM prices where comic_id = 1;

UPDATE prices
SET sale_price = 50
WHERE comic_id = 1;

SELECT * FROM prices where comic_id = 1;
```

Figure 37: TEST TRIGGER

First I set the update time to an arbitrary value in Figure 37. Look at the price information for a comic id equal to 1 in Figure 38 .

price_id	comic_id	purchase_price	sale_price	quality	last_updated
1	1	20	20	VG	2022-05-01

Figure 38: Results for setting

We were able to see that the time had been changed. Now let's change its sale price to 50 and we can see the corresponding change time will be updated in Figure 39.

	price_id	comic_id	purchase_price	sale_price	quality	last_updated
1	1	1	20	50	VG	2023-05-06

Figure 39: Results for TRIGGER

In conclusion, actually these procedural elements add value to the database design by automating tasks, reducing data duplication, maintaining data consistency, and improving overall data integrity which also can help ensure that the database stays up-to-date and can be easily managed and analyzed.

Example Queries:

In this section, I have created a series of Queries. As previously mentioned, the Excelsior database focuses on efficiently managing and organizing the necessary data, ensuring users can search, access, and analyze the information they need to make informed decisions in comic book collecting, investing, or interest. In this section, we will discuss the sample queries provided earlier in detail and their use cases in the context of the comic book database. These example queries showcase the functionality and versatility of the comic book database. They demonstrate various use cases that cater to the needs of comic book collectors, store owners, investors, and enthusiasts. Each query highlights the relationships between specific tables and how to extract information to provide meaningful results.

Query 1 - VIEW_1 comic_value_info:

```
-- Query1 for VIEW_1 comic_value_info:
-- Find all comics that have a Near Mint quality and are signed by the creator, sorted by publisher and title.
SELECT * FROM comic_value_info
WHERE Near_Mint IS NOT NULL AND Signed <> 'None'
ORDER BY Publisher, Title;
```

Figure 40: Query1 for VIEW_1 comic_value_info

The usage scenario: Comic book collectors or investors may be interested in finding high-value comics that are near mint quality and signed by the creator. This query in Figure 40 helps identify such comics, sorted by publisher and title for easy browsing. By using the comic_value_info view, we can retrieve the desired information from the comics, quality_mapping, and signed_by_creator tables. We can see the results in Figure 41.

	Publisher	Title	Issue	Fair	Good	Very_Good	Fine	Very_Fine	Near_Mint	Signed
1	DC Comics	Batman (1939) (#16) (DC Comics)	16	20	40	60	80	100	120	VF (Signed)
2	DC Comics	Superman (1939) (#20) (DC Comics)	20	20	40	60	80	100	120	VG (Signed)
3	DC Comics	Wonder Woman (1941) (#9) (DC Comics)	9	40	80	120	160	200	240	VF (Signed)
4	Marvel Comics	Iron Man (1963) (#2) (Marvel Comics)	2	25	50	75	100	125	150	VF (Signed)
5	Marvel Comics	Spider-Man (1963) (#13) (Marvel Comics)	13	15	30	45	60	75	90	VF (Signed)
6	Marvel Comics	The Amazing Spider-Man (1963) (#150) (Marvel Comics)	150	12.5	25	37.5	50	62.5	75	VG (Signed)
7	Marvel Comics	The Amazing Spider-Man (1963) (#2) (Marvel Comics)	2	10	20	30	40	50	60	VG (Signed)
8	Marvel Comics	The Amazing Spider-Man (1963) (#3) (Marvel Comics)	3	12.5	25	37.5	50	62.5	75	VG (Signed)
9	Marvel Comics	The Amazing Spider-Man (1963) (#4) (Marvel Comics)	4	15	30	45	60	75	90	VG (Signed)
10	Marvel Comics	The Amazing Spider-Man (1963) (#5) (Marvel Comics)	5	17.5	35	52.5	70	87.5	105	VG (Signed)
11	Marvel Comics	Thor (1966) (#3) (Marvel Comics)	3	20	40	60	80	100	120	VF (Signed)

Figure 41: Results of Query1 for VIEW_1 comic_value_info

Query 2 - VIEW_2 Comics_Availability_View:

```
-- Query2 for VIEW_2 Comics_Availability_View:
-- Find all available comics for sale with a release date between '2000-01-01' and '2010-12-31', sorted by title.
SELECT * FROM comics_availability_view
WHERE Status = 'Available' AND Release_Date BETWEEN '2000-01-01' AND '2010-12-31'
ORDER BY Title;
```

Figure 42: Query2 for VIEW_2 comics_Availability_View

The usage scenario: A comic book store owner may want to promote or display available comics published between 2000 and 2010. This query in Figure 42 retrieves all comics that meet these criteria, sorted by title. The comics_availability_view view retrieves data from the comics, prices, and availability tables. We can see the results in Figure 43.

Publisher	Title	Issue	Release_Date	Type	Edition	Cover_Date	Quality	Sale_Price	Status
Marvel Comics	Iron Man (1963) (#2) (Marvel Comics)	2	2006-03-29	Graphic Novel	First	2006-03-29	VF	80	Available
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-01	Comic	First	2008-02-01	NM	100	Available
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	100	Available
IDW Publishing	Locke & Key (2008) (#1) (IDW Publishing)	1	2008-02-20	Comic	First	2008-02-20	NM	80	Available
DC Comics	Superman (1939) (#15) (DC Comics)	15	2003-05-21	Graphic Novel	Reprint	2003-05-21	NM	300	Available
DC Comics	Superman (1939) (#20) (DC Comics)	20	2003-09-17	Graphic Novel	First	2003-09-17	VG	80	Available

Figure 43: Results of Query2 for VIEW_2 comics_Availability_View

Query 3 - VIEW_3 view_comic_details:

```
-- Query3 for VIEW_3 view_comic_details:
-- Find all comics that have the character 'Daredevil' and the creator 'Frank Miller', sorted by publisher and title.
SELECT * FROM view_comic_details
WHERE character_name LIKE '%Daredevil%' AND creator_name LIKE '%Frank Miller%'
ORDER BY Publisher, Title;
```

Figure 44: Query3 for VIEW_3 view_comic_details

The usage scenario: Fans of Daredevil and Frank Miller may want to find all Daredevil comics created by Frank Miller. This query in Figure 44 retrieves such comics, sorted by publisher and title. The view_comic_details view combines information from the comics, characters, creators, comic_characters, and comic_creators tables. We can see the results in Figure 45.

Publisher	Title	Issue	Series_Name	Character_Name	Creator_Name
Marvel Comics	Daredevil (1964) (#158) (Marvel Comics)	158	Daredevil	Daredevil	Frank Miller

Figure 45: Results of Query3 for VIEW_3 view_comic_details

Query 4 - VIEW_4 top_10_valuable_comics:

```
-- Query4 for VIEW_4 top_10_valuable_comics:
-- Find the top 5 most valuable signed comics, sorted by sale price in descending order.
SELECT * FROM top_10_comics_by_sale_price
WHERE Signed = 'Signed'
ORDER BY Sale_Price DESC
LIMIT 5;
```

Figure 46: Query4 for VIEW_4 top_10_valuable_comics

The usage scenario: Comic book investors or collectors may be interested in the top five most valuable signed comics. This query in Figure 46 retrieves the relevant comics, sorted by sale price in descending order. The top_10_comics_by_sale_price view retrieves data from the comics and prices tables. We can see the results in Figure 47.

Publisher	Title	Issue	Series_Name	Sale_Price	Signed
Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	25000	Not Signed
Marvel Comics	Daredevil (1964) (#1) (Marvel Comics)	1	Daredevil	5000	Not Signed
Marvel Comics	Spider-Man (1963) (#1) (Marvel Comics)	1	Spider-Man	5000	Not Signed
Marvel Comics	X-Men (1963) (#1) (Marvel Comics)	1	X-Men	1000	Not Signed
Dark Horse Comics	Hellboy (1993) (#1) (Dark Horse Comics)	1	Hellboy	800	Not Signed

Figure 47: Results of Query4 for VIEW_4 top_10_valuable_comics

Special Query 1:

```
# 1.Find the earliest copy of The Incredible Hulk in good or better condition for less than $25.
SELECT comics.title, comics.issue_number, quality_mapping.comics_condition, prices.sale_price
FROM comics
JOIN quality_mapping ON comics.comic_id = quality_mapping.comic_id
JOIN prices ON comics.comic_id = prices.comic_id
WHERE comics.title = 'The Incredible Hulk'
AND quality_mapping.condition_number >= 2.2
AND prices.sale_price <= 25
ORDER BY comics.issue_number ASC;
```

Figure 48: Special Query 1

The usage scenario: A comic collector wants to find the earliest issue of The Incredible Hulk in good or better condition and priced under \$25. This query in Figure 48 retrieves the desired comic by joining the comics, quality_mapping, and prices tables. We can see the results in Figure 49.

	title	issue_number	comics_condition	sale_price
1	The Incredible Hulk	1	VG	25
2	The Incredible Hulk	4	VG	25

Figure 49: Results of Special Query 1

Special Query 2:

```
# 2.Find how many issues of The Invincible Iron Man are in very good or better quality and can be purchased together for a total of $100 or less before issue 150.
SELECT COUNT(*) AS num_issues, SUM(prices.sale_price) AS total_price
FROM comics
JOIN quality_mapping ON comics.comic_id = quality_mapping.comic_id
JOIN prices ON comics.comic_id = prices.comic_id
WHERE comics.title = 'The Invincible Iron Man'
AND comics.issue_number < 150
AND quality_mapping.condition_number >= 4.0
HAVING total_price <= 100;
```

Figure 50 : Special Query 2:

The usage scenario: A collector wants to know how many issues of Iron Man in very good or better quality and priced at a total of \$100 or less can be purchased before issue 150. This query in Figure 50 retrieves the desired information by joining the comics, quality_mapping, and prices tables. We can see the result in Figure 51.

	num_issues	total_price
1	3	100

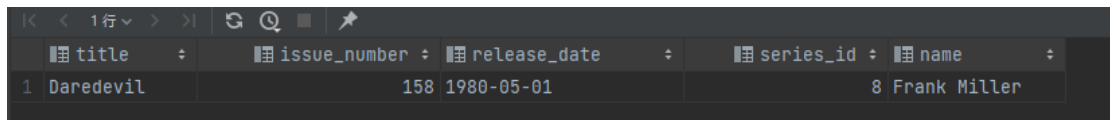
Figure 51: The result of Special Query 2

Special Query 3:

```
# 3.Find Daredevil issues written and drawn by Frank Miller in the 1980s.
SELECT c.title, c.issue_number, c.release_date, c.series_id, cs.name FROM comics c
JOIN comic_creators cc ON c.comic_id = cc.comic_id
JOIN creators cs ON cc.creator_id = cs.creator_id
WHERE cs.name = 'Frank Miller' AND title = 'Daredevil' AND YEAR(release_date) BETWEEN 1980 AND 1989;
```

Figure 52: Special Query 3

The usage scenario: A fan of Daredevil and Frank Miller wants to find all Daredevil issues created by Frank Miller in the 1980s. This query in Figure 52 retrieves the relevant comics by joining the comics, comic_creators, and creators tables. We can see the result in Figure 53.



	title	issue_number	release_date	series_id	name
1	Daredevil	158	1980-05-01	8	Frank Miller

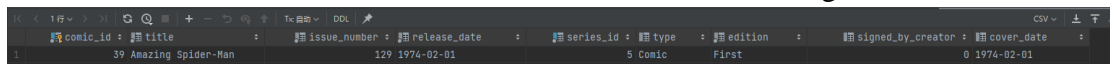
Figure 53: The result of Special Query 3

Special Query 4:

```
# 4. Find the comic where The Punisher first appeared.
SELECT c.*
FROM comics c
JOIN comic_characters cc ON c.comic_id = cc.comic_id
JOIN characters ch ON cc.character_id = ch.character_id
WHERE ch.name = 'The Punisher'
ORDER BY c.cover_date;
```

Figure 54: Special Query 4

The usage scenario: A comic enthusiast wants to find the comic in which The Punisher first appears. This query in Figure 54 retrieves the desired comic by joining the comics, comic_characters, and characters tables. We can see the result in Figure 55



comic_id	title	issue_number	release_date	series_id	type	edition	signed_by_creator	cover_date
39	Amazing Spider-Man	129	1974-02-01	5	Comic	First	0	1974-02-01

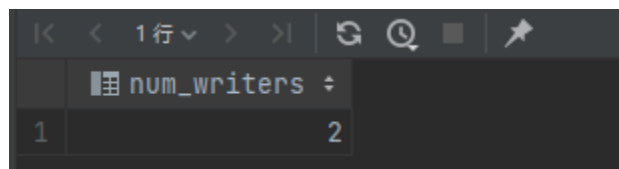
Figure 55: The result of Special Query 4

Special Query 5:

```
# 5. Find how many different writers worked on The Amazing Spider-Man in the 1970s.
SELECT COUNT(DISTINCT cr.creator_id) AS num_writers
FROM comics c
JOIN series s ON c.series_id = s.series_id
JOIN comic_creators cc ON c.comic_id = cc.comic_id
JOIN creators cr ON cc.creator_id = cr.creator_id
WHERE s.name = 'The Amazing Spider-Man'
AND cr.role = 'Writer'
AND YEAR(c.release_date) BETWEEN 1970 AND 1979;
```

Figure 56: Special Query 5

The usage scenario: A fan wanting to find out how many different writers worked on The Amazing Spider-Man in the 1970s. This query in Figure 56 retrieves the desired information by joining the comics, series, comic_creators, and creators tables. We can see the result in Figure 57.



	num_writers
1	2

Figure 57: The result of Special Query 5

Special Query 6:


```
# 6.Find the person who created Daredevil.

SELECT DISTINCT cr.*
FROM creators cr
JOIN comic_creators cc ON cr.creator_id = cc.creator_id
JOIN comics c ON cc.comic_id = c.comic_id
JOIN comic_characters ch_c ON c.comic_id = ch_c.comic_id
JOIN characters ch ON ch_c.character_id = ch.character_id
WHERE ch.name = 'Daredevil' or ch.name = 'The Punisher';
```

Figure 58: Special Query 6

The usage scenario: A comic enthusiast wants to find the creator of Daredevil. This query in Figure 58 retrieves the required creator information by joining the creators, comic_creators, comics, comic_characters, and characters tables. We can see the results in Figure 59.

	creator_id	name	role	biography
1	1	Stan Lee	Writer	Stan Lee was a legendary comic book writer, editor, and publisher.
2	12	Bill Everett	Artist	Bill Everett was an American comic book writer and artist, best known for...
3	13	Jack Kirby	Artist	Jack Kirby was an American comic book artist, writer, and editor, widely ...
4	7	John Romita Sr.	Artist	John Romita Sr. is an American comic book artist, best known for his work...
5	29	Frank Miller	Writer/Artist	Frank Miller is an American comic book writer, artist, and filmmaker, bes...
6	35	Gerry Conway	Writer	Gerry Conway is an American comic book writer known for co-creating the M...
7	36	Len Wein	Writer	Len Wein was an American comic book writer and editor best known for co-C...

Figure 59: Results of Special Query 6

Search comics by character:

```
-- To search for comics by character, you can use the characters and comic_characters tables:

SELECT c.*
FROM comics c
JOIN comic_characters cc ON c.comic_id = cc.comic_id
JOIN characters ch ON cc.character_id = ch.character_id
WHERE ch.name = 'Batman';
```

Figure 60: Search comics by character:

The usage scenario: A Batman fan wants to find all comics featuring Batman as the main character. This query in Figure 60 retrieves the desired comics by joining the comics, comic_characters, and characters tables. We can see the results in Figure 61.

comic_id	title	issue_number	release_date	series_id	type	edition	signed_by_creator	cover_date
2	Batman	1	1939-03-01	2	Comic	First		0 1939-03-01
28	Batman	58	1975-09-11	2	Comic	First		0 1975-09-11
48	Batman	85	1976-09-11	2	Comic	First		0 1975-09-11
56	Batman: The Dark Knight Returns	2	1986-02-01	46	Graphic Novel	First		0 1986-02-01
57	Wonder Woman: Earth One	9	2016-04-06	20	Graphic Novel	First		1 2016-04-06

Figure 61: The results for Search comics by character

Search comics by publisher:

```
-- To search by publisher, you can use the publishers table:
SELECT c.*
FROM comics c
JOIN series s ON c.series_id = s.series_id
JOIN publishers p ON s.publisher_id = p.publisher_id
WHERE p.name = 'DC Comics';
```

Figure 62: Search comics by publisher

The usage scenario: A fan of DC Comics wants to find all comics published by DC Comics. This query in Figure 62 retrieves the desired comics by joining the comics, series, and publishers tables. We can see the results in Figure 63.

comic_id	title	issue_number	release_date	series_id	type	edition	signed_by_creator	cover_date
2	Batman	1	1939-03-01	2	Comic	First		0 1939-03-01
5	Batman	1	1973-09-01	2	Comic	First		0 1973-09-01
28	Batman	50	1975-09-11	2	Comic	First		0 1973-09-11
40	Batman	85	1976-09-11	2	Comic	First		0 1973-09-11
55	Batman: Year One	16	1987-02-12	2	Graphic Novel	First		1 1987-02-12
8	Watchmen	1	1986-09-01	7	Comic	First		0 1986-09-01
12	Watchmen	1	1986-09-01	7	Comic	First		0 1986-09-01
32	Watchmen	1	1986-09-01	7	Comic	First		0 1986-09-01
53	Superman: Birthright	20	2003-09-17	19	Graphic Novel	First		1 2003-09-17
54	Superman: Red Son	15	2003-05-21	19	Graphic Novel	Reprint		0 2003-05-21
57	Wonder Woman: Earth One	9	2016-04-06	20	Graphic Novel	First		1 2016-04-06
58	Wonder Woman: Gods and Mortals	12	1987-02-01	20	Graphic Novel	First		0 1987-02-01

Figure 63: The results of Search comics by publisher

In the results of the aforementioned queries, the relevant tables and established views are mentioned, showcasing the data sources and their connections. Additionally, these tables have been populated with appropriate data to ensure the queries return illustrative results. By analyzing these queries and understanding their usage scenario, we can better comprehend the structure and practicality of the Excelsior database.

The design of the Excelsior database aims to cater to a wide range of user needs. Aptly, these example queries demonstrate the database's ability to handle various types of data and extract valuable information, showcasing its powerful functionality and flexibility. By understanding the context and usage scenario of each query, we can also clearly see that the Excelsior database can indeed lay a solid foundation for advanced applications, better assisting users in learning about comic book-related information.

Conclusion

In this project, the Excelsior database schema provides an efficient and well-structured design for managing comic book data, including series, publishers, characters, creators, and pricing information. The design prioritizes normalizing the database structure to reduce redundancy and ensure data integrity. Separating entities like Creators and Characters from the central Comics entity allows for increased flexibility and scalability as the industry evolves. Relationships between entities are clearly established through primary and foreign key constraints, maintaining data consistency. Entities like `quality_mapping` and `overstreet_price_guide` enable users to assess and compare comic books based on their condition, providing valuable insights for industry stakeholders. Combined with customised views, it allows users to access customised subsets of data, improving data access and

security. Procedural elements are then added, including stored procedures, triggers and functions, to automate tasks, enforce business rules and improve performance, making the database flexible, maintainable and efficient. Finally, sample queries demonstrate the practical applications and functionality of the database, showing its capabilities for data retrieval, manipulation and analysis.

However, enhancements and expansions for the Excelsior database will be required in the future to respond to changing user needs and industry trends.

One area for future development is the incorporation of a more sophisticated pricing model that takes into account various factors that affect market value, such as scarcity, historical significance and market trends. Currently The Excelsior database takes into account the fact that comics are only priced in terms of the quality of a single comic to determine the selling price. Therefore, by integrating additional pricing data from external sources such as auction results, price guides or comic price APIs from other online sales platforms, the database will be able to provide users with more accurate valuations in the future.

The database could also be enhanced by including more detailed information about the comic and its creators, such as awards, affiliations and collaborations, and would meet users' needs to explore more thoroughly the creative networks of creators' careers and industries in the future.

Finally, it would also enable the integration of the database with social media platforms or online communities, fostering a sense of community among comics enthusiasts and facilitating knowledge sharing and experience exchange. Building on this updated database and providing a user-friendly, intuitive interface and advanced search functions on top of these advanced applications will facilitate user interaction with the database and provide a better user experience.

Therefore, The Excelsior database must be optimised by constantly adapting it to the future development of the comics industry, constantly improving it as an invaluable tool for managing and tracking information about the world of comics.

Acknowledgements

After completing a semester of the COMP40725 course at UCD, I have finally accomplished this project.

First and foremost, I would like to extend my special gratitude to my professor, Tony Veale. I am particularly grateful for his dedication and efforts throughout the course this semester. I appreciate the guidance and answers to my questions he provided during the entire project process. His invaluable advice played a critical role in the successful completion of this project.

Additionally, I would like to express my profound thanks to my fellow classmates with whom I exchanged ideas and insights. Their input on how to improve the project's functionality was truly invaluable.

Finally, I am grateful to myself for continuously challenging and persevering in the face of obstacles of the project.

Reference

- [1]: Wikipedia Contributors (2023). 'DC Comics', Wikipedia. Available at: https://en.wikipedia.org/wiki/DC_Comics (Accessed: 1 May 2023).
- [2]: 'Marvel Comics: The Untold Story': full of action, not so many heroes 2013, , Tribune Content Agency.
- [3]: Kleefeld, S. 2020, Webcomics, First edn, Bloomsbury, London [England].
- [4]: League of Comic Geeks (2023). Comic Books. Available at: <https://leagueofcomicgeeks.com/comics> (Accessed: 1 May 2023).
- [5]: Harrington, J.L. & Harrington, J.L. 2016, Relational database design and implementation, Fourth edn, Morgan Kaufmann/Elsevier, Amsterdam.