Video Game Store Database

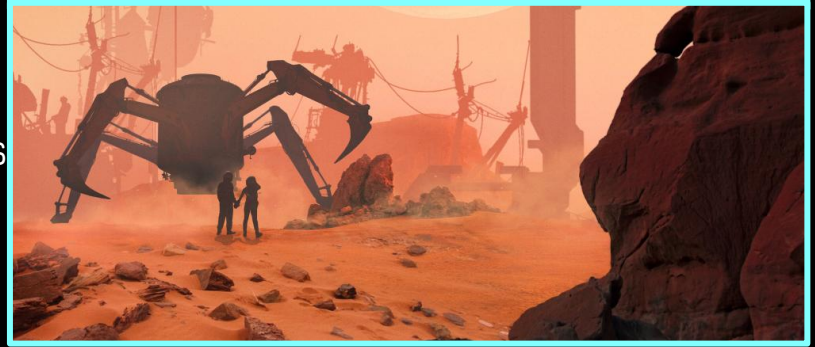
Anqi Guan, Chuhan Xu,
Lin Ye, Xinyu Qiu, Yuwen Mai

Table of contents

01

Database Design Document

Purpose oriented database design to address business problems with consideration of business rules which optimizes the design decisions



Database Purposes

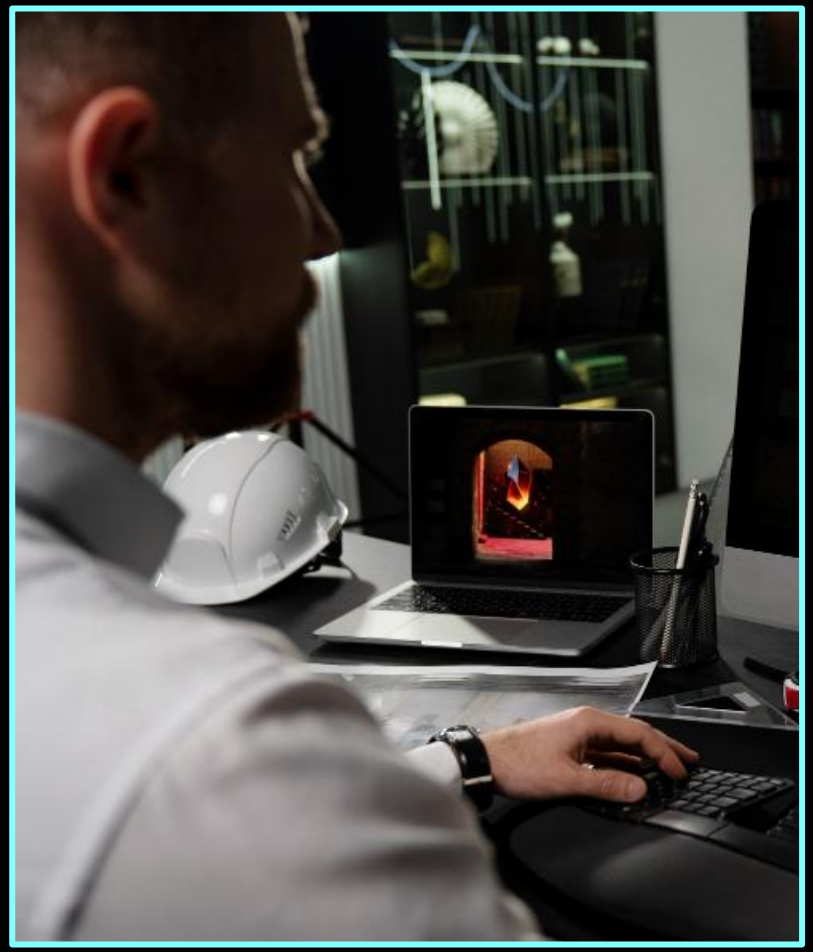
To maintain the data used to support customer game purchases, and analyze store operational performance. It will be used by store employees only.

Business Problems

- Enable the video game store to create detailed sales reports, breaking down sales trends by game, platform and publisher to optimize sales strategies.
- Provide data for inventory management to ensure that supply meets demand across different games and platforms, preventing overstocking or stockouts.
- Offer insights for targeted promotions by analyzing customer preferences and purchase histories, allowing for personalized marketing campaigns.
- Enhance customer service by using sales and preference data to provide recommendations, improving customer satisfaction and loyalty

Business Rules

- Each Order is placed by one Customer.
- Each Order is processed by one Employee.
- Each Order will have one Payment record.
- Each Order may have one or more Games.
- A Customer may have one or more Orders.
- An Employee may process one or more Orders.
- A Payment record is related to one Order.
- A Game may appear in one or more Orders.
- Each Game is developed by one Developer.
- Each Game is published by one Publisher.
- Each Game is categorized under one Genre.
- Each Game is available on one or more Platforms.
- Each Game can have one Inventory record.
- A Developer can develop one or more Games.
- A Publisher can publish one or more Games.
- A Genre can encompass one or more Games.
- A Platform can support one or more Games.
- An Inventory record will have one Game.
- A Customer will have one Preference record.
- Each Preference record will have one Customer.
- Each Preference record will have one Genre and one Publisher.



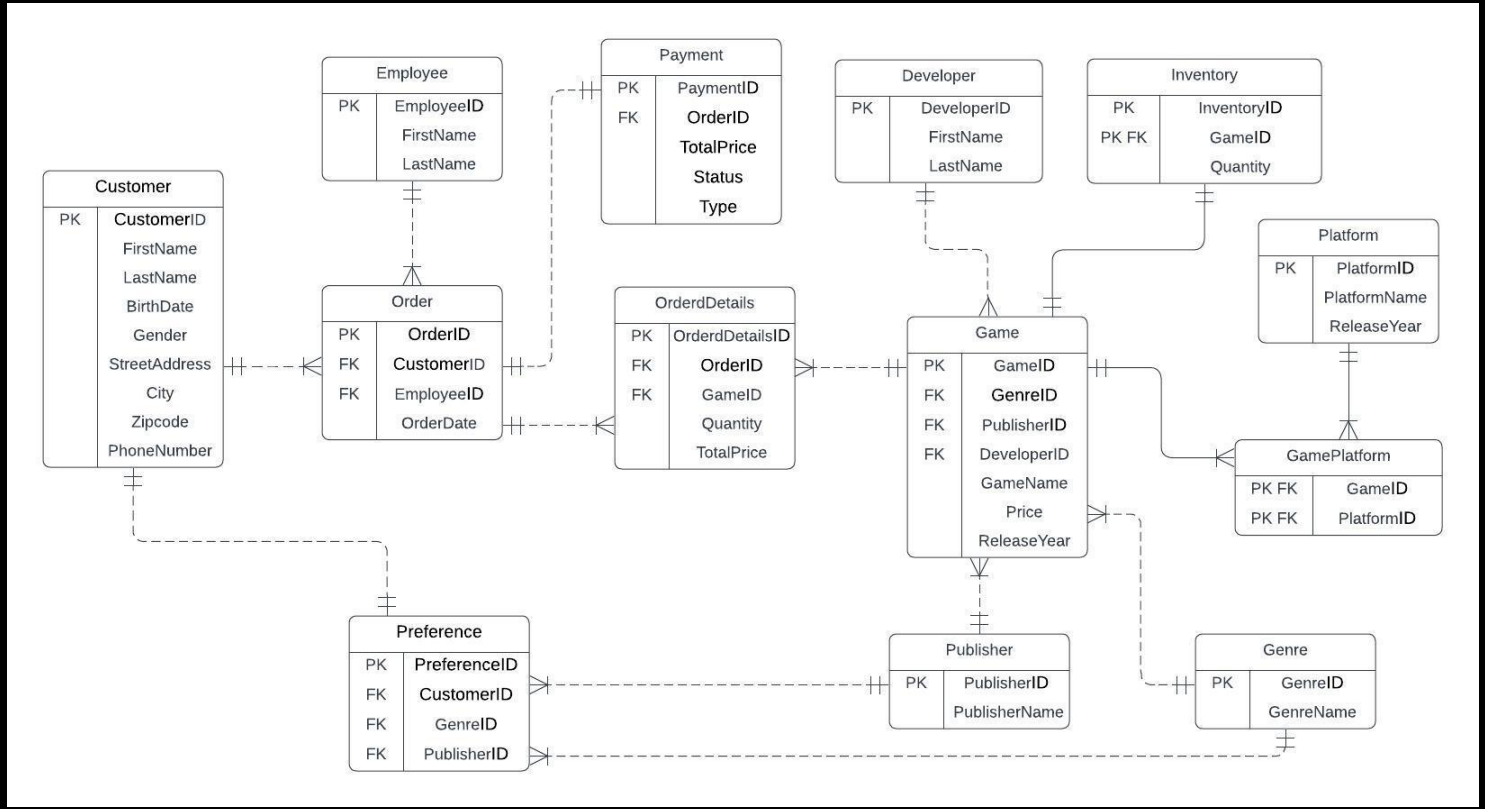
02

Entity Relationship Diagram

Design Decisions

Entity Name	Why Entity Included	How Entity is Related to Other Entities
Customer	In the Video Games Store Management Database, the customer entity is included to collect and manage comprehensive information about the individuals who purchase video games. This entity is crucial for tracking customer preferences, purchase history, and interactions with the store. By maintaining detailed records on customers, such as age, gender, street address, the database facilitates personalized marketing, enhances customer service, and enables analysis of buying patterns to optimize inventory management.	As the main entity in the database, the customer's primary key, Customer ID, relates it to Order and Preference so that important information about personalized marketing and customer service may be gained.
Order	The Order entity is crucial in the Video Game Store Management Database for recording and managing the details of purchases made by customers. It captures essential information such as the identification of the customer making the purchase,	The Order entity is directly related to the Game entity through an associative entity OrderedDetails due to the many-to-many relationships. Many orders may
	the employee managing the sale, and the linkage to the payment details. This entity enables the tracking of sales transactions, providing insights into customer buying patterns and store performance	be created and could contain multiple games.The Order entity also relates to the Employee entity and Customer entity with many to one relationship, as one customer and one employee can create many orders while one order only belongs to one customer and employee.

ERD


Employee
PK EmployeeID
FirstName
LastName
Payment
PK PaymentID
FK OrderID
TotalPrice
Status
Type
Developer
PK DeveloperID
FirstName
LastName
Inventory
PK InventoryID
PK FK GameID
Quantity
Customer
PK CustomerID
FirstName
LastName
BirthDate
Gender
StreetAddress
City
Zipcode
PhoneNumber
Order
PK OrderID
FK CustomerID
FK EmployeeID
OrderDate
OrderdDetails
PK OrderdDetailsID
FK OrderID
FK GameID
Quantity
TotalPrice
Game
PK GameID
FK GenreID
FK PublisherID
FK DeveloperID
GameName
Price
ReleaseYear
Platform
PK PlatformID
PlatformName
ReleaseYear
GamePlatform
PK FK GameID
PK FK PlatformID
Preference
PK PreferenceID
FK CustomerID
FK GenreID
FK PublisherID
Publisher
PK PublisherID
PublisherName
Genre
PK GenreID
GenreName

03

SQL DDL Statements

Table Creation

Customer, Order, Game

Use of IDENTITY for auto-incrementing primary keys and relationships between tables through foreign keys

```sql
-- PART1 Create tables
CREATE TABLE Customer (
  CustomerID INT IDENTITY NOT NULL PRIMARY KEY,
  FirstName VARCHAR(26) NOT NULL,
  LastName VARCHAR(26) NOT NULL,
  BirthDate DATE NOT NULL,
  Gender VARCHAR(10) NOT NULL CONSTRAINT CHK_GENDER CHECK (Gender IN ('Male', 'Female')),
  StreetAddress VARCHAR(255) NOT NULL,
  City VARCHAR(255) NOT NULL,
  Zipcode VARCHAR(10) NOT NULL,
  PhoneNumber VARCHAR(20) NOT NULL
);
```

```sql
-- Order
CREATE TABLE [Order] (
  OrderID INT IDENTITY NOT NULL PRIMARY KEY,
  CustomerID INT NOT NULL
    REFERENCES Customer(CustomerID),
  EmployeeID INT NOT NULL
    REFERENCES Employee(EmployeeID),
  OrderDate DATE NOT NULL
);
```

```sql
-- Game
CREATE TABLE Game (
  GameID INT IDENTITY NOT NULL PRIMARY KEY,
  GenreID INT NOT NULL
    REFERENCES Genre(GenreID),
  PublisherID INT NOT NULL
     REFERENCES Publisher(PublisherID),
  DeveloperID INT NOT NULL
    REFERENCES Developer(DeveloperID),
  GameName VARCHAR(255) NOT NULL,
  Price DECIMAL(10, 2) NOT NULL,
  ReleaseYear INT NOT NULL,
);
```

Data Insertion

SQL INSERT Script

Inserting data populates tables with initial values essential for the database's operation.

```sql
-- Inserting into Game
INSERT INTO Game (GenreID, PublisherID, DeveloperID, GameName, Price, ReleaseYear) VALUES
(1, 1, 1, 'Mass Effect', 59.99, 2007),
(2, 2, 2, 'Halo 5', 49.99, 2015),
(3, 3, 3, 'The Witcher 3', 39.99, 2015),
(4, 4, 4, 'Splatoon 2', 59.99, 2017),
(5, 5, 5, 'Civilization VI', 49.99, 2016),
(6, 6, 6, 'Tetris 99', 0.00, 2019),
(7, 7, 7, 'Sonic Mania', 19.99, 2017),
(8, 8, 8, 'The Sims 4', 39.99, 2014),
(9, 9, 9, 'Resident Evil 7', 29.99, 2017),
(10, 10, 10, 'FIFA 21', 59.99, 2020),
(11, 11, 11, 'Half-Life 3', 60.00, 2021),
(12, 12, 12, 'Fortnite', 0.00, 2017),
(13, 13, 13, 'GTA VI', 70.00, 2023),
(14, 14, 14, 'FIFA 22', 60.00, 2021),
(15, 15, 15, 'World of Warcraft', 15.00, 2004),
(16, 16, 16, 'Mobile Legends', 0.00, 2016),
(17, 17, 17, 'League of Legends', 0.00, 2009),
(18, 18, 18, 'Clash of Clans', 0.00, 2012),
(19, 19, 19, 'Rules of Survival', 0.00, 2017),
(20, 20, 20, 'Genshin Impact', 0.00, 2020);
```

```sql
serting into Customer
T INTO Customer (FirstName, LastName, BirthDate, Gender, StreetAddress, City, Zipcode, PhoneNumber) VALUE
n', 'Doe', '1992-04-01', 'Male', '123 Maple Street', 'Springfield', '12345', '555-0101'),
e', 'Smith', '1988-11-24', 'Female', '456 Oak Avenue', 'Greenville', '23456', '555-0102'),
ce', 'Johnson', '1995-07-18', 'Female', '789 Pine Road', 'Franklin', '34567', '555-0103'),
', 'Brown', '1985-02-11', 'Male', '1012 Cedar Lane', 'Centerville', '45678', '555-0104'),
ol', 'Jones', '1990-12-05', 'Female', '1314 Elm St', 'Lakeview', '56789', '555-0105'),
id', 'Wilson', '1993-03-22', 'Male', '1516 Spruce Dr', 'Hill Valley', '67890', '555-0106'),
', 'Davis', '1987-08-30', 'Female', '1718 Oak St', 'Shelbyville', '78901', '555-0107'),
nk', 'Miller', '1989-06-15', 'Male', '1920 Birch Blvd', 'Sandford', '89012', '555-0108'),
ce', 'White', '1994-09-07', 'Female', '2122 Palm Road', 'Springfield', '90123', '555-0109'),
ry', 'Black', '1991-10-29', 'Male', '2324 Pine Ave', 'Greenville', '01234', '555-0110'),
ra', 'Craft', '1983-05-14', 'Female', '45 Adventurer Rd', 'Explorer City', '23456', '555-0111'),
cus', 'Fenix', '1979-04-12', 'Male', '123 Sera St', 'Jacinto', '34567', '555-0112'),
han', 'Drake', '1981-11-10', 'Male', '789 Treasure Ave', 'Libertalia', '45678', '555-0113'),
be', 'Frazer', '1984-02-21', 'Female', '321 Artifact Ln', 'Uncharted Isle', '56789', '555-0114'),
a', 'Croft', '1985-02-14', 'Female', '132 Explorer Tr', 'Tomb City', '67890', '555-0115'),
n', 'Marston', '1973-05-20', 'Male', '456 Outlaw Rd', 'Armadillo', '78901', '555-0116'),
ie', 'Williams', '2001-09-26', 'Female', '789 Quarantine Zone', 'Boston', '89012', '555-0117'),
hur', 'Morgan', '1863-04-22', 'Male', '1012 Western Way', 'Valentine', '90123', '555-0118'),
l', 'Valentine', '1974-11-03', 'Female', '1314 Raccoon St', 'Raccoon City', '01234', '555-0119'),
n', 'Kennedy', '1977-10-10', 'Male', '1516 Survivor Ln', 'Raccoon City', '12345', '555-0120');
```

Triggers

trg_UpdateOrderdetailsTotoalPrice

```sql
-- 3.1.Trigger to automatically update the total price in orderdetails
CREATE TRIGGER trg_UpdateOrderdetailsTotalPrice
ON OrderDetails
AFTER INSERT, UPDATE
AS
BEGIN
   DECLARE @TotalPrice DECIMAL(10, 2), @GameID INT;

   SELECT @TotalPrice = i.Quantity * g.Price,
          @GameID = i.GameID
   FROM Inserted i
   JOIN Game g
        ON i.GameID = i.GameID

   UPDATE OrderDetails
   SET TotalPrice = @TotalPrice
   WHERE GameID = @GameID;
END;
-- DROP TRIGGER trg_UpdateOrderdetailsTotalPrice;
```

trg_UpdateInventoryQuantity

```sql
-- 3.2.Trigger to automatically update the inventory quantity when an OrderDetails record is changed.
CREATE TRIGGER trg_UpdateInventoryQuantity
ON OrderDetails
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   DECLARE @GameID INT, @Adjustment INT
   SELECT @GameID = COALESCE(i.GameID, d.GameID),
          @Adjustment = ISNULL(i.Quantity, 0) - ISNULL(d.Quantity, 0)
   FROM Inserted i
   FULL JOIN Deleted d ON i.GameID = d.GameID

   UPDATE Inventory
   SET Quantity = Quantity - @Adjustment
   WHERE GameID = @GameID;
END;
-- DROP TRIGGER trg_UpdateInventoryQuantity;
```

CREATE TRIGGER

AFTER INSERT, UPDATE

Automate the updating of total prices in order details and to adjust inventory quantities automatically when orders are placed or modified.

Table-Level Check Constraints

CHECK CONSTRAINT

dbo.CheckInventory(GameID) >=0

To prevent the processing of orders when the ordered quantity exceeds the available inventory for a game

```
REATE FUNCTION CheckInventory (@GameID INT)
ETURNS smallint
S
EGIN
  DECLARE @InventoryQuantity smallint, @SoldQuantity smallint, @GameRemain smallint;

  SELECT @InventoryQuantity = Quantity
  FROM Inventory
  WHERE GameID = @GameID;

  SELECT @SoldQuantity = Quantity
  FROM OrderDetails
  WHERE GameID = @GameID;

  IF @InventoryQuantity IS NULL OR @SoldQuantity IS NULL
   RETURN -1;

  SET @GameRemain = @InventoryQuantity - @SoldQuantity;
  RETURN @GameRemain
ND;
- Add table-level CHECK constraint
LTER TABLE OrderDetails ADD CONSTRAINT BanInventoryUnavailable CHECK (dbo.CheckInventory(GameID) >= 0);
- Clean up
- ALTER TABLE OrderDetails DROP CONSTRAINT BanInventoryUnavailable;
- DROP FUNCTION CheckInventory
```

Computed Column

ALTER TABLE, ADD

Calculating a customer's age in the Customer table from their birthdate

```sql
-- 5.2.Computed Columns based on a function: Create a function to calculate the age of a customer
CREATE FUNCTION fn_CalculateAge(@CustID INT)
RETURNS INT
AS
BEGIN
DECLARE @Age INT, @DateOfBirth DATE=
    (SELECT BirthDate
    FROM Customer
    WHERE CustomerID = @CustID);
    SET @Age = DATEDIFF(hour, @DateOfBirth, GETDATE()) / 8766;
    RETURN @Age;
END
-- Add a function-based computed column to the Customer
ALTER TABLE Customer
ADD Age AS (dbo.fn_CalculateAge(CustomerID));
-- See what the computed column looks like
SELECT CustomerID,
       FirstName,
       LastName,
       Age
FROM Customer
WHERE Age < 30
ORDER BY Age DESC;
-- Clean up
--ALTER TABLE Customer DROP COLUMN Age;
--DROP FUNCTION fn_CalculateAge;
```

Output

CustomerID	FirstName	LastName	Age
9	Grace	White	29
3	Alice	Johnson	28
17	Ellie	Williams	22

Column Data Encryption

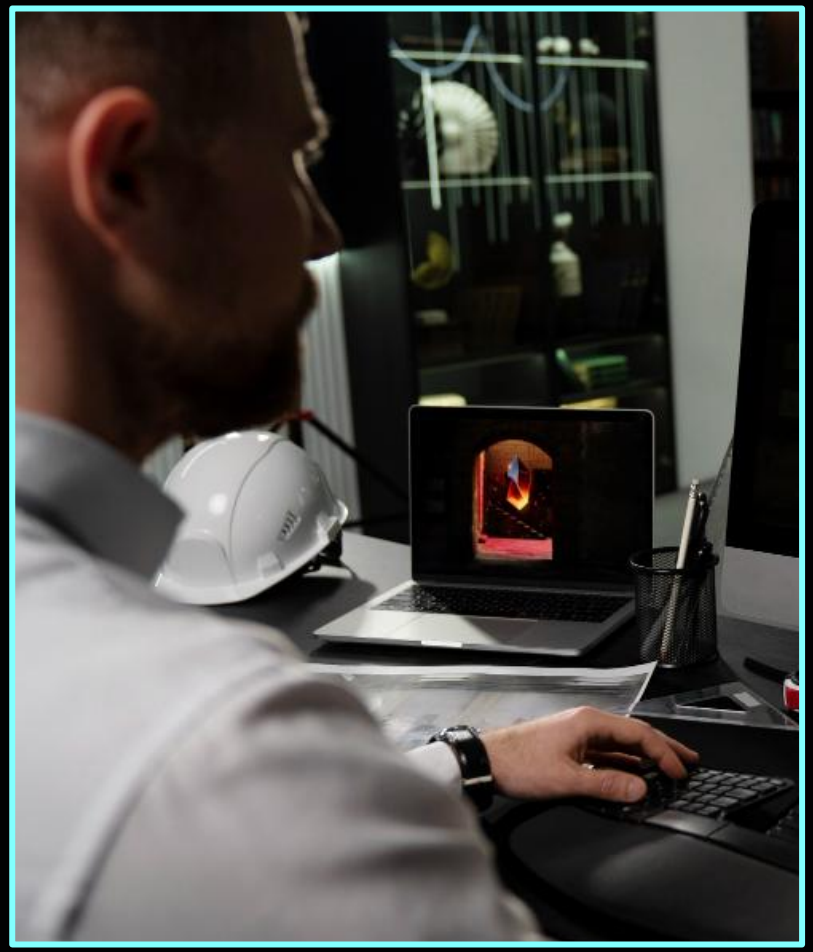
Symmetric Key

AES_256

Encrypting the PhoneNumber column in the Customer table to protect customer privacy

```sql
-- 5.3.Column Data Encryption: Encrypt PhoneNumber
-- Check if the EncryptedPhoneNumber column already exists
IF NOT EXISTS (
SELECT 1
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Customer'
AND COLUMN_NAME = 'EncryptedPhoneNumber'
)
BEGIN

-- Add a new column for encrypted phone number
ALTER TABLE Customer
ADD EncryptedPhoneNumber VARBINARY(256) NULL;
END
GO

-- Create master key
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Group8';
GO

-- Create certificate
CREATE CERTIFICATE CustomerPhoneCert
WITH SUBJECT = 'Customer Phone Number Encryption',
EXPIRY_DATE = '2026-10-31';
GO

-- Create symmetric key
CREATE SYMMETRIC KEY CustomerPhoneKey
WITH ALGORITHM = AES_128
ENCRYPTION BY CERTIFICATE CustomerPhoneCert;
GO
```

PhoneNumber	EncryptedPhoneNumber
555-0101	, Õõ2O Þ «åL Ë #á´½·7¥Ö±... [68]
555-0102	, Õõ2O Þ «åL #P æbcù æ¤ß... [68]
555-0103	, Õõ2O Þ «åL S>M O¬ ÷ HV... [68]
555-0104	, Õõ2O Þ «åL ± F# ü ðx ... [68]
555-0105	, Õõ2O Þ «åL ËÖ»ý°@²'J©Ì ... [68]
555-0106	, Õõ2O Þ «åL 9<£¥Éa»ÉõâLc... [68]
555-0107	, Õõ2O Þ «åL ~² ¯þr æ D"... [68]
555-0108	, Õõ2O Þ «åL I¬/ D{øZ ê ... [68]
555-0109	, Õõ2O Þ «åL `|»_4 ÁxÏ]®... [68]
555-0110	, Õõ2O Þ «åL iÖ Ìë KC tC... [68]
555-0111	, Õõ2O Þ «åL e±§? C 2ån ... [68]
555-0112	, Õõ2O Þ «åL K°¾\wÁ2H & F... [68]
555-0113	, Õõ2O Þ «åL ÿ ¶ |? ~ Fã ... [68]
555-0114	, Õõ2O Þ «åL iw <ç£Öz l... [68]
555-0115	, Õõ2O Þ «åL 0Æ· aý 0Ô &... [68]
555-0116	, Õõ2O Þ «åL Üïy «êmí ×§i... [68]
555-0117	, Õõ2O Þ «åL µ dà à;· æ... [68]
555-0118	, Õõ2O Þ «åL gÞ ~|)) ... [68]
555-0119	, Õõ2O Þ «åL = ¥é ' e ±í... [68]
555-0120	, Õõ2O Þ «åL ý ±/áaÊ2U¿ ... [68]



04

SQL Views

View 1 - Remain Game Quantity

```sql
CREATE VIEW RemainingGameQuantity AS
SELECT
   G.GameID,
   G.GameName,
   I.Quantity AS TotalInventory,
   ISNULL(SUM(OD.Quantity), 0) AS TotalOrdered,
   I.Quantity - ISNULL(SUM(OD.Quantity), 0) AS RemainingQuantity
FROM
   Game G
JOIN
   Inventory I ON G.GameID = I.GameID
LEFT JOIN
   OrderDetails OD ON G.GameID = OD.GameID
GROUP BY
   G.GameID,
   G.GameName,
   I.Quantity;
-- use RemainingGameQuantity view
SELECT * FROM RemainingGameQuantity
ORDER BY RemainingQuantity;
-- DROP VIEW RemainingQuantity;
```

	123 GameID	ABC GameName	123 TotalInventory	123 TotalOrdered	123 RemainingQuantity
1	5	Civilization VI	10	2	8
2	4	Splatoon 2	15	4	11
3	3	The Witcher 3	20	6	14
4	2	Halo 5	25	2	23
5	1	Mass Effect	30	4	26
6	6	Tetris 99	35	4	31
7	7	Sonic Mania	40	6	34
8	8	The Sims 4	45	4	41
9	9	Resident Evil 7	50	2	48
10	11	Half-Life 3	50	0	50
11	10	FIFA 21	55	4	51
12	13	GTA VI	60	0	60
13	14	FIFA 22	80	0	80
14	12	Fortnite	100	0	100
15	16	Mobile Legends	150	0	150
16	15	World of Warcraft	200	0	200
17	18	Clash of Clans	250	0	250
18	17	League of Legends	300	0	300
19	19	Rules of Survival	400	0	400
20	20	Genshin Impact	500	0	500

View 2 -
Top 3 Popular games Quarterly

```sql
CREATE VIEW TopThreePopularGamesYearQuarterly AS
SELECT *
FROM (
  SELECT
    G.GameID,
    G.GameName,
    GN.GenreName AS Genre,
    P.PublisherName AS Publisher,
    D.FirstName + ' ' + D.LastName AS Developer,
    CAST(YEAR(O.OrderDate) AS VARCHAR) + '-' + CAST(DATEPART(qq, O.OrderDate) AS VARCHAR) AS YearQuarter,
    SUM(OD.Quantity) AS TotalOrderAmount,
    RANK() OVER (PARTITION BY CAST(YEAR(O.OrderDate) AS VARCHAR) + '-' + CAST(DATEPART(qq, O.OrderDate) AS
     ORDER BY SUM(OD.Quantity) DESC) AS Ranking
  FROM
    Game G
    JOIN Genre GN ON G.GenreID = GN.GenreID
    JOIN Publisher P ON G.PublisherID = P.PublisherID
    JOIN Developer D ON G.DeveloperID = D.DeveloperID
    JOIN OrderDetails OD ON G.GameID = OD.GameID
    JOIN [Order] O ON OD.OrderID = O.OrderID
  GROUP BY
    G.GameID,
    G.GameName,
    GN.GenreName,
    P.PublisherName,
    D.FirstName,
    D.LastName,
    O.OrderDate
) AS SubRankedGames
WHERE
  Ranking <= 3;
```

	YearQuarter	TopThreeGames
1	2023-1	3-The Witcher 3, 7-Sonic Mania, 6-Tetris 99, 1-Mass Effect, 3-The Witcher 3
2	2023-2	10-FIFA 21, 8-The Sims 4, 7-Sonic Mania, 6-Tetris 99, 4-Splatoon 2, 5-Civilization VI
3	2023-3	8-The Sims 4, 7-Sonic Mania, 3-The Witcher 3, 1-Mass Effect, 2-Halo 5, 6-Tetris 99, 8-The Sims 4, 9-Resident Evil 7
4	2023-4	3-The Witcher 3, 7-Sonic Mania, 8-The Sims 4, 10-FIFA 21, 10-FIFA 21, 1-Mass Effect, 1-Mass Effect, 4-Splatoon 2, 2-Halo 5, 3-The Witcher 3

	GameID	GameName	Genre	Publisher	Developer	YearQuarter	TotalOrderAmount	Ra
1	3	The Witcher 3	RPG	Ubisoft	Todd Howard	2023-1	6	
2	7	Sonic Mania	Arcade	Sega	Ted Price	2023-1	6	
3	6	Tetris 99	Puzzle	Square Enix	Neil Druckmann	2023-1	4	
4	1	Mass Effect	Action	Electronic Arts	Casey Hudson	2023-1	4	
5	3	The Witcher 3	RPG	Ubisoft	Todd Howard	2023-1	3	
6	10	FIFA 21	Sports	Capcom	Sam Houser	2023-2	4	
7	8	The Sims 4	Simulation	Bandai Namco Entertainment	Kenzo Tsujimoto	2023-2	4	
8	7	Sonic Mania	Arcade	Sega	Ted Price	2023-2	3	
9	6	Tetris 99	Puzzle	Square Enix	Neil Druckmann	2023-2	2	
10	4	Splatoon 2	Shooter	Nintendo	Hidetaka Miyazaki	2023-2	2	
11	5	Civilization VI	Strategy	Sony Interactive Entertainment	Marcin Iwinski	2023-2	2	
12	8	The Sims 4	Simulation	Bandai Namco Entertainment	Kenzo Tsujimoto	2023-3	6	
13	7	Sonic Mania	Arcade	Sega	Ted Price	2023-3	6	
14	3	The Witcher 3	RPG	Ubisoft	Todd Howard	2023-3	6	
15	1	Mass Effect	Action	Electronic Arts	Casey Hudson	2023-3	4	
16	2	Halo 5	Adventure	Activision	Jason Jones	2023-3	2	
17	6	Tetris 99	Puzzle	Square Enix	Neil Druckmann	2023-3	2	
18	8	The Sims 4	Simulation	Bandai Namco Entertainment	Kenzo Tsujimoto	2023-3	2	
19	9	Resident Evil 7	Horror	Bethesda Softworks	Yves Guillemot	2023-3	2	
20	3	The Witcher 3	RPG	Ubisoft	Todd Howard	2023-4	3	
21	7	Sonic Mania	Arcade	Sega	Ted Price	2023-4	3	
22	8	The Sims 4	Simulation	Bandai Namco Entertainment	Kenzo Tsujimoto	2023-4	2	
23	10	FIFA 21	Sports	Capcom	Sam Houser	2023-4	2	
24	10	FIFA 21	Sports	Capcom	Sam Houser	2023-4		
25	1	Mass Effect	Action	Electronic Arts	Casey Hudson	2023-4		
26	1	Mass Effect	Action	Electronic Arts	Casey Hudson	2023-4	2	
27	4	Splatoon 2	Shooter	Nintendo	Hidetaka Miyazaki	2023-4	2	
28	2	Halo 5	Adventure	Activision	Jason Jones	2023-4	1	
29	3	The Witcher 3	RPG	Ubisoft	Todd Howard	2023-4	1	

TopThreeGames: varchar(M
corresponding table colum

View 3 - Customer Preference

```sql
-- 4.3.Create view to analyze customer preference
CREATE VIEW CustomerPreferenceSummary AS
SELECT c.CustomerID,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    GenreName,
    PublisherName

FROM Preference pre
   JOIN Genre g ON pre.GenreID = g.GenreID
   JOIN Publisher pub ON pre.PublisherID = pub.PublisherID
   JOIN Customer c  ON c.CustomerID = pre.CustomerID

-- use view to get the most popular Genre
SELECT GenreName,
    COUNT(GenreName) AS GenreCount,
    DENSE_RANK() OVER
    (ORDER BY COUNT(GenreName) DESC) AS RankingGenreCount
FROM CustomerPreferenceSummary
GROUP BY GenreName
ORDER BY GenreCount DESC

-- use view to get the most popular Publisher
SELECT PublisherName,
       COUNT(PublisherName) AS PublisherCount,
       DENSE_RANK() OVER
    (ORDER BY COUNT(PublisherName) DESC) AS RankingPublisherCount
FROM CustomerPreferenceSummary
GROUP BY PublisherName
ORDER BY PublisherCount DESC;
--clean up
-- DROP VIEW CustomerPreferenceSummary
```

Most popular Genre

	GenreName	GenreCount	RankingGenreCount
1	RPG	6	1
2	Adventure	4	2
3	Horror	4	2
4	Sports Simulation	2	3
5	Strategy	1	4
6	Visual Novel	1	4
7	Puzzle	1	4
8	Shooter	1	4

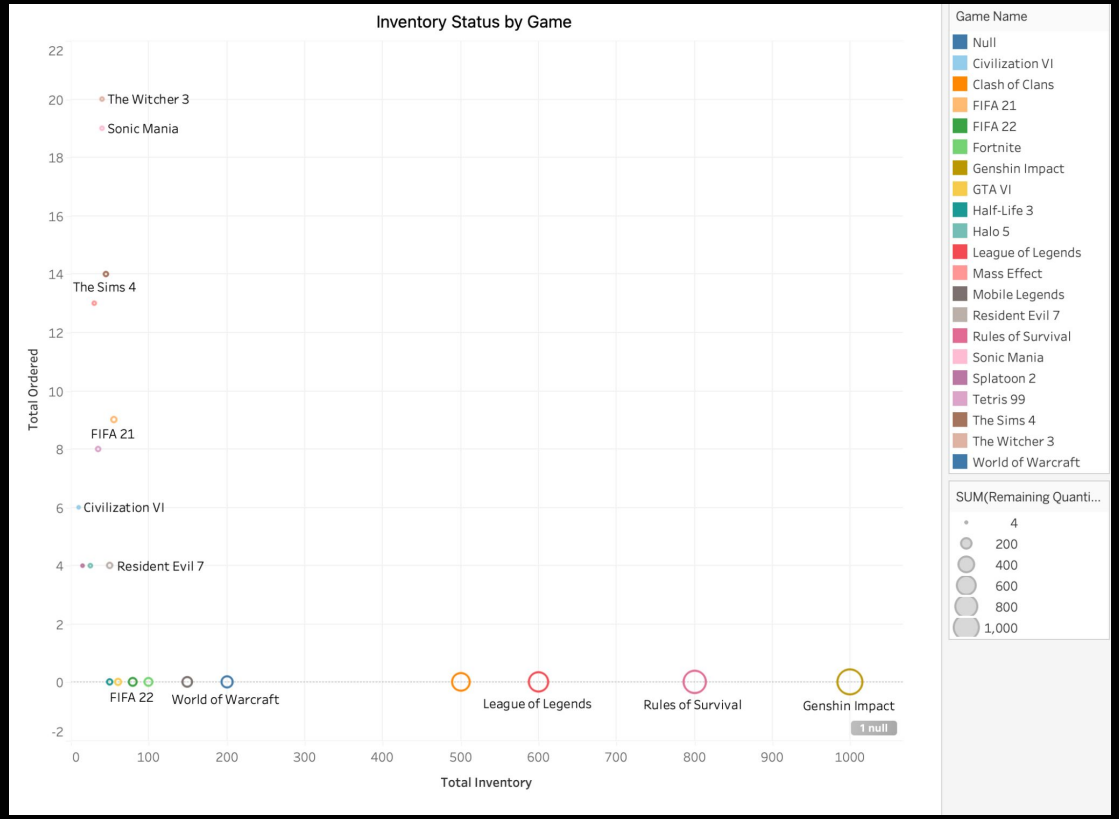
Most popular Publisher

	PublisherName	PublisherCount	RankingPublisherCount
1	Nintendo	7	1
2	Capcom	4	2
3	Activision	3	3
4	miHoYo	2	4
5	Epic Games	1	5
6	Rockstar Games	1	5
7	Sega	1	5
8	Ubisoft	1	5

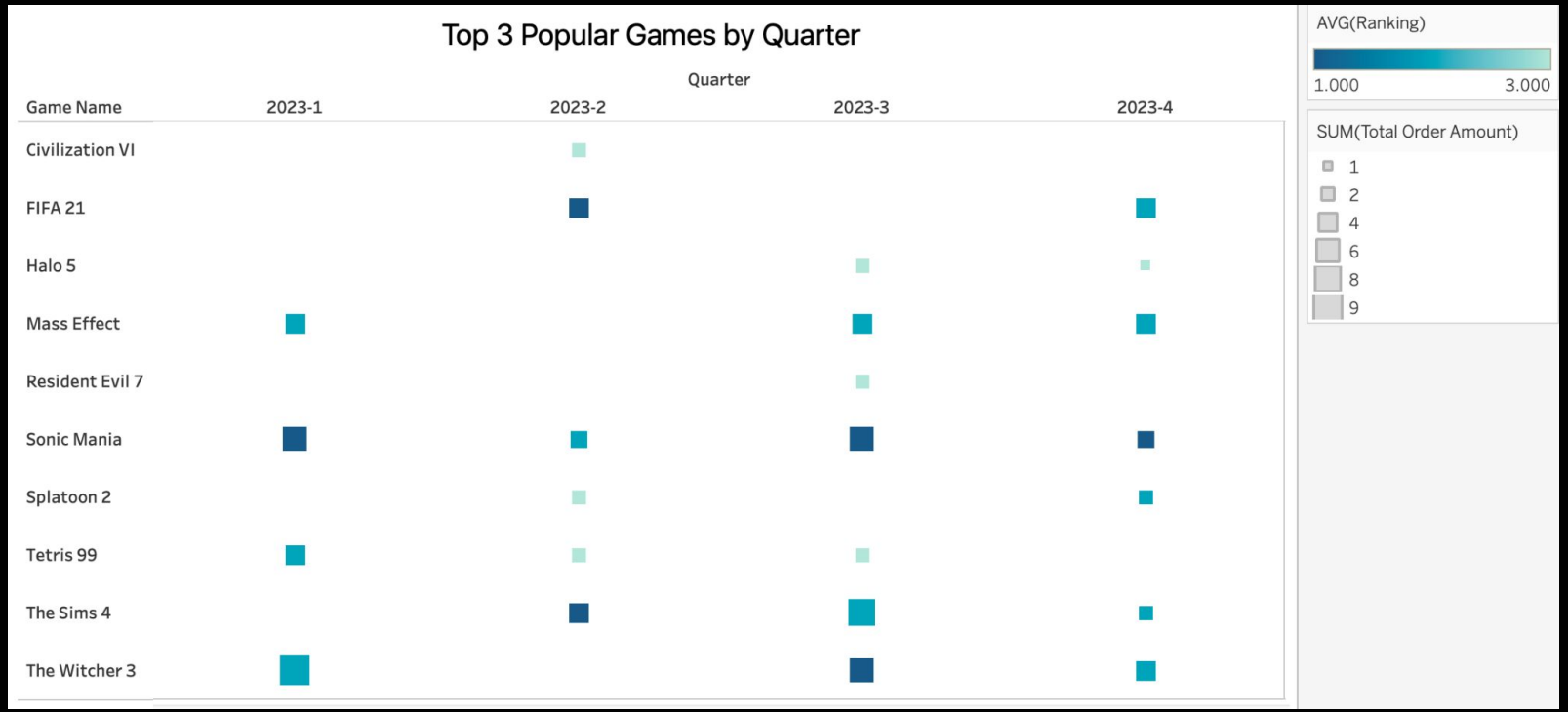
05

Visual Reports

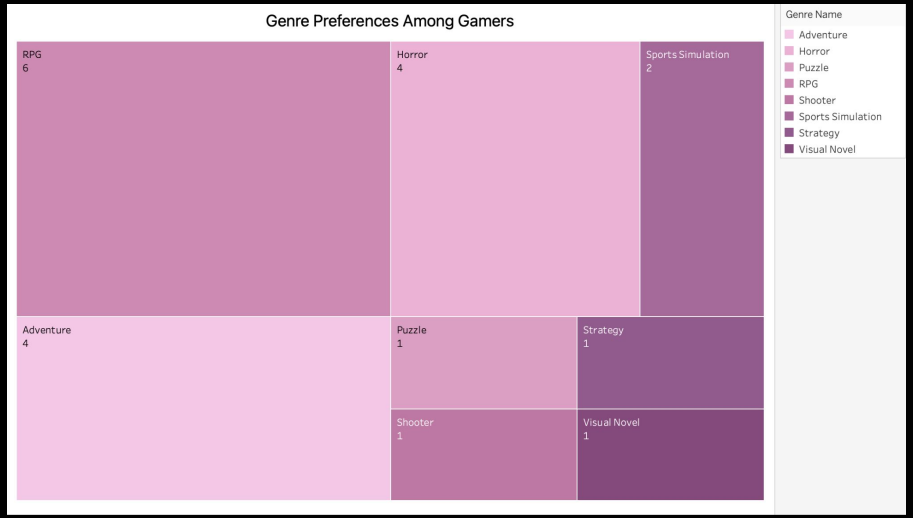
Visual Reports 1
Inventory Status by Game

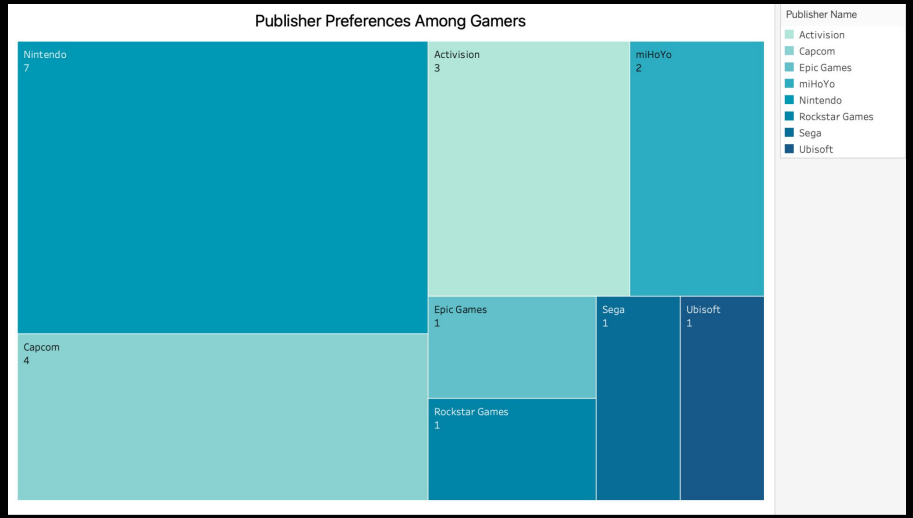


Visual Reports 2
Top 3 Popular games Quarterly



Visual Reports 3
Customer Preference





Thanks