Theoretical Questions

Theory 1 :

One-to-One - One PKey, One FKey. For every one instance of an ID key, there is only one other instance where it's used in a OtO connected table, respectively. Can be used to back up player statistics (One player, One player statistics).
One-to-Many - Table stores FKey with every PKey; One FKey, Many PKey. For every one instance of an ID key, there can be several instances where it's used in a OtM connected table, respectively. Can be used to initialize servers (One server, Several players), or Leaderboards (One leaderboard, Several players).
Many-to-Many - Additional table that stores combinations of FKeys of all Many-to-Many connected tables exists. All PKeys have an FKey that is in a OtO relationship with the additional table. For every instance of an ID key, there can be multiple instances of it being used in a MtM connected table. Can be used to initialize friend groups (Several players, Several friend groups).

Theory 2 :

Depending on what exactly is needed from the NoSQL for the game in question, main options would be a Document-based database like JSON (for the purposes of quick saving and retrieving of data, coming at a drawback of easy accessibility and modifiability by the players on the user's end), or an In-memory database like Redis (same advantages as above - high-speed read and write operations - without being easily modifiable), or a flexible document model of mongoDB (enables advanced data management and detailed recording)

Between the CAP types of NoSQL databases, for online multiplayer gaming, Consistency should be a priority. CP type of databases is generally prioritized for the sake of server hubs remaining accessible for all users independent of each other's connection failures, but a CA database could be implemented instead for local CO-OP multiplayer games.

Theory 3 :

Icantbefuckingbothered.png

Symmetric encryption : same key is used to both encrypt and decrypt.
Asymmetric encryption : one key is used to encrypt, a combination of two keys is used to decrypt.

Theory 4 :

NoSQL strengths:

- Greatly improve the performance and scalability
- Data retrieval is much faster and easier

NoSQL weaknesses:

- Data input and storage is much more expensive
- Tend to have less features and complexity compared to the Relational databases
- Tend to have unreliable view of the results

Labs

A : Data Query Language

Task 1 :

```
SELECT 1 [Player_ID], [Player_Name], [Player_DateOfCreation]
FROM [mock_exam2024db].[dbo].[Players]
ORDER BY [Player_DateOfCreation]  ASC;
```

Task 2 :

```
SELECT GS.[GameS_ID], P.Player_Name, PS.PlayersInSession FROM
[mock_exam2024db].[dbo].[GameSession] AS GS
LEFT JOIN PlayerPosition AS PP ON PP.GameS_ID = GS.GameS_ID
LEFT JOIN TreasorPosition AS TP ON TP.GameS_ID = GS.GameS_ID
LEFT JOIN Players AS P ON P.Player_ID = PP.Player_ID
LEFT JOIN (
            SELECT GS.GameS_ID, COUNT(*) AS PlayersInSession
            FROM GameSession AS GS
            LEFT JOIN PlayerPosition AS PS
            ON PS.GameS_ID = GS.GameS_ID
            GROUP BY GS.GameS_ID
                ) AS PS ON PS.GameS_ID = GS.GameS_ID
 WHERE (PP.Position_X = TP.Position_X AND PP.Position_Y = TP.Position_Y);
```

```
SELECT GS.GameS_ID, P.Player_Name, SP.PlayersInSession FROM GameSession
AS GS
LEFT JOIN PlayerPosition AS PP ON PP.GameS_ID = GS.GameS_ID
LEFT JOIN TreasorPosition AS TP ON TP.GameS_ID = GS.GameS_ID
LEFT JOIN Players AS P ON PP.Player_ID = P.Player_ID
LEFT JOIN (
    SELECT GS.GameS_ID, COUNT(*) AS PlayersInSession FROM GameSession
AS GS
    LEFT JOIN PlayerPosition AS PS ON PS.GameS_ID = GS.GameS_ID
```

```
    GROUP BY GS.GameS_ID
) AS SP ON SP.GameS_ID = GS.GameS_ID
WHERE (PP.Position_X = TP.Position_X AND PP.Position_Y = TP.Position_Y)


select GS.[GameS_ID],

 (select count(*) from [dbo].[PlayerPosition] PP where PP.GameS_ID =
GS.GameS_ID) 'Number of players',
 (select Pl.Player_Name from PlayerPosition PP2
 inner join TreasorPosition TP on TP.GameS_ID = PP2.GameS_ID
 inner join Players Pl on Pl.Player_ID = PP2.Player_ID
 where PP2.Position_X = TP.Position_X AND PP2.Position_Y =
TP.Position_Y AND GS.GameS_ID = PP2.GameS_ID) 'The winner'
 from [dbo].[GameSession] GS
```

Task 3 :

```
SELECT DISTINCT P.Player_ID 'ID', Player_Name 'Player name', NumberOfGames 'Games
played' FROM Players P
LEFT JOIN PlayerPosition AS PST ON PST.Player_ID = P.Player_ID
LEFT JOIN (
            SELECT PP.Player_ID, COUNT(DISTINCT PP.GameS_ID)
                  AS NumberOfGames FROM PlayerPosition AS PP
            RIGHT JOIN PlayerPosition AS PS ON PS.Player_ID = PP.Player_ID
            GROUP BY PP.Player_ID
                  ) AS PS ON PS.Player_ID = PST.Player_ID;
```

Task 4 :

```
SELECT P.Player_Name 'Nickname', COUNT(PP.GameS_ID) AS GamesWon
FROM [mock_exam2024db].[dbo].[Players] AS P
        JOIN PlayerPosition PP ON P.Player_ID = PP.Player_ID
        JOIN GameSession GS ON PP.GameS_ID = GS.GameS_ID
        JOIN TreasorPosition TP ON GS.GameS_ID = TP.GameS_ID
WHERE
        PP.Position_X = TP.Position_X AND PP.Position_Y = TP.Position_Y

GROUP BY P.Player_Name;
```

Task 5 :

```
SELECT GS.GameS_ID 'Game ID', TimeStart 'Session started', TimeEnd 'Session ended'
FROM [mock_exam2024db].[dbo].[GameSession] AS GS
LEFT JOIN PlayerPosition AS PP ON PP.GameS_ID = GS.GameS_ID
LEFT JOIN TreasorPosition AS TP ON TP.GameS_ID = GS.GameS_ID
LEFT JOIN (
            SELECT TP.[Timestamp] AS TimeStart FROM TreasorPosition AS TP
                ) AS TS ON GS.GameS_ID = TP.GameS_ID
LEFT JOIN (
        SELECT PP.GameS_ID, PP.[Timestamp] AS TimeEnd FROM PlayerPosition AS PP
                JOIN TreasorPosition AS TP ON TP.GameS_ID = PP.GameS_ID
                WHERE PP.Position_X = TP.Position_X AND PP.Position_Y = TP.Position_Y
                ) AS TE ON GS.GameS_ID = TE.GameS_ID
                WHERE (PP.Position_X = TP.Position_X AND PP.Position_Y =
TP.Position_Y);
```

B. SQL Transactional

Task 1 :

```
CREATE PROCEDURE GenerateMapSize @MaxPlayers int, @min_X int, @max_X int,
@min_Y int, @max_Y int
 AS
DECLARE @GameS_ID int;
DECLARE @Tresor_ID int;
  BEGIN TRAN GenerateRect
    SAVE TRAN SavepointA
          -- create a transaction which adds a row to game sessions.
              INSERT GameSession (GameS_MaxPlayers, Rect_min_x, Rect_max_x,
Rect_min_y, Rect_max_y)
              VALUES (@MaxPlayers, @min_X, @max_X, @min_Y, @max_Y)
              SET @GameS_ID = (SELECT TOP 1 (GameS_ID) FROM GameSession
ORDER BY GameS_ID DESC);
              --SET @GameS_ID = SELECT GameS_ID FROM INSERTED;
              -- select top 1 row ordered by gameid descending and set that value as your
games ID for treasure position.
          SAVE TRAN SavepointA

              DECLARE @num INT = ROUND(RAND()*4,0) + 1
              INSERT Treasor (Treasor_Name) VALUES (CHOOSE(@num, 'Ancient
Scroll', 'Golden Chalice', 'Healing Potion', 'Mystic Gem', 'Silver Sword'));
```

```
		SET @Tresor_ID = (SELECT TOP 1 (Tresor_ID) FROM Treasor ORDER BY
Tresor_ID DESC);

    SAVE TRAN SavepointA

		INSERT TreasorPosition (GameS_ID, Tresor_ID, Position_X, Position_Y,
Timestamp)
		VALUES (@GameS_ID, @Tresor_ID,
FLOOR(RAND()*(@max_X-@min_X+1)+@min_X),
FLOOR(RAND()*(@max_Y-@min_Y+1)+@min_Y), GETDATE())

   COMMIT TRAN GenerateRect
	ROLLBACK TRAN SavepointA
 GO --NO SEMICOLON! no ; this guy
```

Task 2 :

Most likely, create a separate DB for that and just log in PlayerID Position_X Position_Y via trigger on every update.

Task 3 :

```
ALTER TRIGGER OutOfBounds ON PlayerPosition AFTER UPDATE
AS
BEGIN
INSERT GameSessions AS GS
	UPDATE PlayerPosition SET Position_X = (SELECT GS.Rect_min_x FROM
GameSession GS WHERE GS.GameS_ID = GameS_ID)
	WHERE Position_X < (SELECT GS.Rect_min_x FROM GameSession GS WHERE
GS.GameS_ID = GameS_ID)
	UPDATE PlayerPosition SET Position_X = (SELECT GS.Rect_max_x FROM
GameSession GS WHERE GS.GameS_ID = GameS_ID)
	WHERE Position_X > (SELECT GS.Rect_max_x FROM GameSession GS WHERE
GS.GameS_ID = GameS_ID)
	UPDATE PlayerPosition SET Position_Y = (SELECT GS.Rect_min_y FROM
GameSession GS WHERE GS.GameS_ID = GameS_ID)
	WHERE Position_Y < (SELECT GS.Rect_min_y FROM GameSession GS WHERE
GS.GameS_ID = GameS_ID)
	UPDATE PlayerPosition SET Position_Y = (SELECT GS.Rect_max_y FROM
GameSession GS WHERE GS.GameS_ID = GameS_ID)
	WHERE Position_Y > (SELECT GS.Rect_max_y FROM GameSession GS WHERE
GS.GameS_ID = GameS_ID)
END;
```