# Task description

|  |  |
| --- | --- |
| Task | Set up and optimize a simple iGaming platforms database and its connectors |
| Description | Create a concise database for an iGaming platform including the application layer scripts to route endpoint calls to respective procedures.    The platform needs to manage the players, host tournaments, and distribute prizes. Tournaments are competitions where the players betting the most are ranked highest. |
| Requirements | * App layer: Simple application (language of your choice)   + Implement endpoint to trigger the distribute prize stored procedure   + Implement endpoint that returns the ranking list * MySQL DB: Create the necessary tables, write the stored procedure for prize distribution, implement CTE or window Functions to produce a ranking report. |
| Implementation details | When creating the tables please ensure that your implementation contains the following mandatory tables and procedures:   * [players]  - columns for: player id, player name, player email, account balance * [tournaments] – columns: tournament id, tournament name, prize pool, start date, end date * Create a Stored procedure for distributing the prizes based on player placement and tournament prize pool: 1 place 50%, 2 place 30%, 3 place 20%; * Use CTE or Window Function Query to generate a report that ranks the players based on their account balance in descending order. So we have the highest earnings at the top. |
| Deliverables | Share a GitHub or GitLab repo that contains the following:   * SQL scripts for creating the tables, the stored procedure and the ranking query * App code * A brief readme file explaining how to run the scripts and DB scripts and what each script does.   + What did you learn and if you encountered any challenges, how did you overcome them?   + What did you take in consideration to ensure the query and or stored procedure is efficient and handles edge cases?   + CTEs and Window function:     - If you used CTEs or Window Functions, what did you learn about their power and flexibility?     - How might you apply the technique in more complex scenarios,   + Optimization: How did you optimize the queries and stored procedures. |

# Some remarks

I recognize this task as a tournament settlement task. Tournament settlement (prize distribution) between players can be done in many ways, but according to the task we would need to distribute the prize according to requirement:

“1 place 50%, 2 place 30%, 3 place 20%;”

Every tournament has a prize pool which should be distributed according to the requirement.

I have created a solution which allows generic prize distribution. For every tournament there is default prize distribution (this one from request), but it is possible to create some other distribution with many places and with some desired distribution.

As described “Tournaments are competitions where the players betting the most are ranked highest.” I am aware of other types of tournaments as well but I will use this definition.

Usually, one game or more games are played in tournament. I will presume that points I am connecting to some player in some tournaments are already calculated by some engine (betting in games are generating points for that player in that specific tournament. I am not going to create that part of system where this generation is happening, but, basically, game played during tournament is generating points for that player, and it reports these bets to tournament server, which also can store these transactions (every round in tournament as one transaction).

So, I am presuming that I already have current points state for specific player for specific tournament calculated.

# Special cases

1. There are no players in the tournament
2. There is one player in the tournament
3. There are two players in the tournament
4. There are three players in the tournament (is this special case??)
5. Two or more players share the first prize
6. Two or more players share the second prize
7. Two or more player share the third prize.
8. Combination of these cases above
9. Other generic cases where we have random number of players (i.e. 10) sharing first place, and there are only 3 places and other generics.

I have created generic solution which does not do all these checks but it is generic. It will distribute the money fair if there are players which are tied in some position, regardless of their number. This solution also handles the cases where there are less players than prizes. It works for any number of players tied at any rank.

# Some presumptions

1. Tournament can be in several states: “not\_started”, “ongoing”, “finished”, “settlement\_started”, “settlement\_successful”, “settlement\_unsuccessful”, “cancelled”.
2. Only tournament which is in state “finished” can be settled (prizes to be given to players). Prize will be given in tournament (tournament\_settlement) and that money will be placed to player accounts as well.
3. Only tournament which is in “ongoing” state can accept some new player points (players bets). This is not developed, but it should be one of constraints for some new methods which are adding points to the player in some tournament.
4. If tournament is in state “settlement unsuccessful” it is a sign that whole process should be checked why settlements is unsuccessful, with logs inspection.
5. Only tournament which is settled successfully can have prize distribution.
6. During “ongoing” tournament state, ranking query should be used to return current standings. It will just return current standings (rank) not the prize distribution. So, idea is to settle the tournament once, when tournament is ended, and then prizes will be distributed.
7. Since there are players with the same balances or same points in leaderboard, rank function will be used (not dense\_rank) meaning that if two players have the same number of points, they will share first place, and there won’t be second place, just third place, as shown in this example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PlayerId | FirstName | LastName | Points | Ranked |
| 10 | Vojislav | Depalov | 100 | 1 |
| 15 | Ava | Harris | 100 | 1 |
| 3 | Daniel | Wright | 90 | 3 |
| 22 | Charlotte | Hall | 80 | 4 |
| ….. | ….. | ….. | ….. | ….. |

# Deliverables

1. report that ranks the players based on their account balance in player table descending order. So we have the highest earnings at the top.

Can be found in file ranking\_queries.sql

Can be seen through API.

1. leaderboard report for some specific tournament

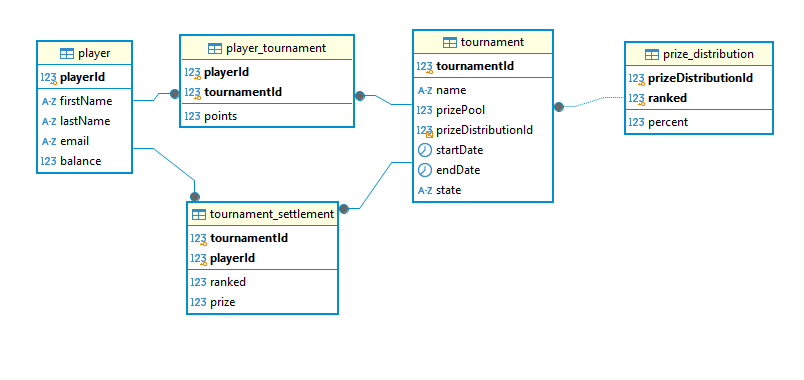
This report is used to show leaderboard in some ongoing tournament. It shows only players with their current standings, not with prize distribution.

Can be found in file ranking\_queries.sql

Can be seen through API.

1. SP for tournament settlement. In order to run this SP tournament must be in finished state. Calling it would trigger tournament settlement.

# Database model



# Tables description

**player** – contain players

**tournament** – contain tournaments. Every tournament has its **prize\_distribution** attached. Every tournament has its prize pool which is divided according to the formula defined in **prize\_distribution**. Formula can be any generic formula.

**player\_tournament** – every player can be in one or more tournaments. Player is earning points in tournaments. So, every player is accumulating points in that specific tournament.

**tournament\_settlement** – this table is showing, once the tournament is finished, how prizes for that tournament are distributed and how much money every player which participated in tournament earned. So, it contains data for finished and settled tournaments for all players which participated in tournaments even for this one which did not earn anything.

# How to call methods in app

## Settling the tournament

* Calling this API from bash starts tournament settling
* Parameter provided is tournamentId which we want to settle. This is that /1 <- tournamentId

curl -X PUT <http://localhost:8080/settleTournament/1>

## Getting the list of players with their rank made of their balance in player table

* Browser

<http://localhost:8080/playerRanks>

* or through bash script

curl -X GET <http://localhost:8080/playerRanks>

## Getting the list of players with their rank made of their points in some\current running tournament

* Browser

<http://localhost:8080/tournamentLeaderboardReport/1>

* or through bash script

curl -X GET <http://localhost:8080/tournamentLeaderboardReport/1>

That /1 is tournamentId we would like to get the results.