PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
/* ===== BLOCK: Block 1 ===== */

-- creating a table of players
-- join with date of birth table
CREATE OR REPLACE TABLE PLAYER AS
SELECT
    P."playerid"::int AS P_PLAYER_ID,
    "NameFirst" AS P_NAME_FIRST,
    "NameLast" AS P_NAME_LAST,
    "CurrentHandle" AS P_NICK_NAME,
    "DateOfBirth"::date AS P_DATE_OF_BIRTH,
    "Year"::int AS P_BIRTH_YEAR,
    UPPER("CountryCode") AS P_COUNTRY2CODE,
    "WorldRanking"::int AS P_WORLD_RANK,
    "CountryRanking"::int AS P_COUNTRY_RANK,
    "TotalUSDPrize"::float AS P_PRIZE_USD,
    "TotalTournaments"::int AS P_CNT_TOURNAMENTS
FROM API01PLAYERBYID_COMPLETE_DATASET AS P
FULL JOIN PLAYERIDDOBLINKS AS D ON P."playerid"=D."playerid"
ORDER BY P_PLAYER_ID;

-- the code for kosovo ‘xk’ is replaced by ‘ko’ in the PLAYER table, so that it can be correctly linked to the COUNTRY table
UPDATE PLAYER
SET P_COUNTRY2CODE = 'KO'
WHERE P_COUNTRY2CODE = 'XK';

/* ===== BLOCK: Block 2 ===== */
-- creating a table of games
CREATE OR REPLACE TABLE GAME AS
-- add RANK to the table according to three indicators - number of players, number of tournaments and total prize
WITH CTE_GAME AS (
SELECT
    "gameid"::int AS G_GAME_ID,
    "GameName" AS G_GAME_NAME,
    "TotalUSDPrize"::float AS G_PRIZE_USD,
```

```
    "TotalTournaments"::int AS G_CNT_TOURNAMENTS,
    "TotalPlayers"::int AS G_CNT_PLAYERS,
    RANK() OVER(ORDER BY G_PRIZE_USD DESC NULLS LAST) AS G_RANK_PRIZE_USD,
    RANK() OVER(ORDER BY G_CNT_TOURNAMENTS DESC NULLS LAST) AS G_RANK_CNT_TOURNAMENTS,
    RANK() OVER(ORDER BY G_CNT_PLAYERS DESC NULLS LAST) AS G_RANK_CNT_PLAYERS
FROM API04GAMEBYID_COMPLETE_DATASET
ORDER BY G_GAME_ID)

-- add the total RANK for the game to the table (created by adding all three basic ranks and ordering them from the lowest one)
SELECT
    G_GAME_ID,
    G_GAME_NAME,
    G_PRIZE_USD,
    G_CNT_TOURNAMENTS,
    G_CNT_PLAYERS,
    G_RANK_PRIZE_USD,
    G_RANK_CNT_TOURNAMENTS,
    G_RANK_CNT_PLAYERS,
    RANK() OVER(ORDER BY (G_RANK_PRIZE_USD + G_RANK_CNT_TOURNAMENTS + G_RANK_CNT_PLAYERS) ASC NULLS LAST) AS G_RANK_ALL
FROM CTE_GAME
ORDER BY G_GAME_ID ASC NULLS LAST;

/* ===== BLOCK: Block 3 ===== */

-- creation of a country codebook
CREATE OR REPLACE TABLE COUNTRY AS
SELECT
    "CountryLAT" AS C_COUNTRY,
    "English_short_name" AS C_COUNTRY_UTF8,
    "Alpha2code" AS C_COUNTRY2CODE,
    "Alpha3code" AS C_COUNTRY3CODE,
    "Continent" AS C_CONTINENT
FROM COUNTRYCODES_IVETA
ORDER BY C_COUNTRY;
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```
/* ===== BLOCK: Block 4 ===== */

-- creation of the GDP PER CAPITA table
CREATE OR REPLACE TEMPORARY TABLE TEMP_GDP_PER_CAPITA AS
SELECT
    "Code" AS GDP_COUNTRY3CODE,
    "Year"::int AS GDP_YEAR,
    "GDP_per_capita_PPP_constant_2017_international"::float AS GDP_GDP_PER_CAPITA_USD
FROM "gdp-per-capita-worldbank"
ORDER BY GDP_COUNTRY3CODE, GDP_YEAR;

-- change the Kosovo code OWID_KOS to KOS so that it can be paired with the COUNTRY table
UPDATE TEMP_GDP_PER_CAPITA
SET GDP_COUNTRY3CODE = 'KOS'
WHERE GDP_COUNTRY3CODE = 'OWID_KOS';

CREATE OR REPLACE TABLE GDP_PER_CAPITA AS
SELECT
    C_COUNTRY2CODE AS GDP_COUNTRY2CODE,
    GDP_YEAR,
    GDP_GDP_PER_CAPITA_USD
FROM TEMP_GDP_PER_CAPITA
LEFT JOIN COUNTRY ON GDP_COUNTRY3CODE= C_COUNTRY3CODE
WHERE GDP_COUNTRY3CODE is not null AND GDP_COUNTRY3CODE != 'OWID_WRL'
ORDER BY GDP_COUNTRY2CODE, GDP_YEAR;

/* ===== BLOCK: Block 5 ===== */

-- creation population table
CREATE OR REPLACE TEMPORARY TABLE TEMP_POPULATION AS
SELECT
    "Country_name" AS POP_COUNTRY,
    "Year"::int AS POP_YEAR,
    "Population"::int AS POP_POPULATION,
    "Population_aged_15_to_19_years"::int AS POP_POPULATION_AGED15_19,
```

```
    "Population_aged_20_to_29_years"::int AS POP_POPULATION_AGED20_29
FROM "population-and-demography"
-- omitting rows containing summaries for certain areas (rows that are not for individual countries)
WHERE
    NOT POP_COUNTRY LIKE '%UN%'
    AND NOT POP_COUNTRY ILIKE '%developed%'
    AND NOT POP_COUNTRY ILIKE '%income%'
    AND NOT POP_COUNTRY ILIKE '%countries%'
    AND NOT POP_COUNTRY ILIKE '%developing%'
    AND NOT POP_COUNTRY = 'World'
ORDER BY POP_COUNTRY, POP_YEAR;

-- edit country names to be linked to the COUNTRY table
UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Congo-Kinshasa'
WHERE POP_COUNTRY = 'Congo';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Bonaire, Sint Eustatius and Saba'
WHERE POP_COUNTRY = 'Bonaire Sint Eustatius and Saba';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Virgin Islands (British)'
WHERE POP_COUNTRY = 'British Virgin Islands';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Brunei Darussalam'
WHERE POP_COUNTRY = 'Brunei';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Cabo Verde'
WHERE POP_COUNTRY = 'Cape Verde';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Cote dIvoire'
```

```
WHERE POP_COUNTRY = $$Cote d'Ivoire$$;

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Congo-Kinshasa'
WHERE POP_COUNTRY = 'Democratic Republic of Congo';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Timor-Leste'
WHERE POP_COUNTRY = 'East Timor';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Lao Peoples Democratic Republic'
WHERE POP_COUNTRY = 'Laos';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Micronesia'
WHERE POP_COUNTRY = 'Micronesia (country)';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Russian Federation'
WHERE POP_COUNTRY = 'Russia';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Saint Martin'
WHERE POP_COUNTRY = 'Saint Martin (French part)';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Sint Maarten'
WHERE POP_COUNTRY = 'Sint Maarten (Dutch part)';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'Syrian Arab Republic'
WHERE POP_COUNTRY = 'Syria';

UPDATE TEMP_POPULATION
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
SET POP_COUNTRY = 'Turkiye'
WHERE POP_COUNTRY = 'Turkey';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'United Kingdom of Great Britain and Northern Ireland'
WHERE POP_COUNTRY = 'United Kingdom';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'United States of America'
WHERE POP_COUNTRY = 'United States';

UPDATE TEMP_POPULATION
SET POP_COUNTRY = 'United States Minor Outlying Islands'
WHERE POP_COUNTRY = 'United States Virgin Islands';

CREATE OR REPLACE TABLE POPULATION AS
SELECT
    C_COUNTRY2CODE AS POP_COUNTRY2CODE,
    POP_YEAR,
    POP_POPULATION,
    POP_POPULATION_AGED15_19,
    POP_POPULATION_AGED20_29
FROM TEMP_POPULATION
LEFT JOIN COUNTRY ON POP_COUNTRY = C_COUNTRY
ORDER BY POP_COUNTRY, POP_YEAR;

/* ===== BLOCK: Block 6 ===== */

-- creating an Internet table
CREATE OR REPLACE TEMPORARY TABLE TEPM_INTERNET AS
SELECT
    "Entity" AS I_COUNTRY,
    "Code" AS I_COUNTRY3CODE,
    "Year"::int AS I_YEAR,
    "Individuals_using_the_Internet_of_population"::float AS I_SHARE_OF_INDIVID_USING_INTERNET
```

```
FROM "share-of-individuals-using-the-internet"
WHERE I_COUNTRY3CODE is not null AND I_COUNTRY3CODE != 'OWID_WRL'
ORDER BY I_COUNTRY, I_YEAR;

-- change the Kosovo code OWID_KOS to KOS so that it can be paired with the COUNTRY table
UPDATE TEPM_INTERNET
SET I_COUNTRY3CODE = 'KOS'
WHERE I_COUNTRY3CODE = 'OWID_KOS';

-- edit country names if someone decides to pair them to the COUNTRY table

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Virgin Islands (British)'
WHERE I_COUNTRY = 'British Virgin Islands';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Brunei Darussalam'
WHERE I_COUNTRY = 'Brunei';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Congo-Kinshasa'
WHERE I_COUNTRY = 'Congo';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Cabo Verde'
WHERE I_COUNTRY = 'Cape Verde';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Cote dIvoire'
WHERE I_COUNTRY = $$Cote d'Ivoire$$;

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Congo-Kinshasa'
WHERE I_COUNTRY = 'Democratic Republic of Congo';
```

```sql
UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Timor-Leste'
WHERE I_COUNTRY = 'East Timor';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Lao Peoples Democratic Republic'
WHERE I_COUNTRY = 'Laos';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Micronesia'
WHERE I_COUNTRY = 'Micronesia (country)';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Russian Federation'
WHERE I_COUNTRY = 'Russia';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Syrian Arab Republic'
WHERE I_COUNTRY = 'Syria';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'Turkiye'
WHERE I_COUNTRY = 'Turkey';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'United Kingdom of Great Britain and Northern Ireland'
WHERE I_COUNTRY = 'United Kingdom';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'United States of America'
WHERE I_COUNTRY = 'United States';

UPDATE TEPM_INTERNET
SET I_COUNTRY = 'United States Minor Outlying Islands'
WHERE I_COUNTRY = 'United States Virgin Islands';
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
CREATE OR REPLACE TABLE INTERNET AS
SELECT
    C_COUNTRY2CODE AS I_COUNTRY2CODE,
    I_YEAR,
    I_SHARE_OF_INDIVID_USING_INTERNET
FROM TEPM_INTERNET
LEFT JOIN COUNTRY ON I_COUNTRY3CODE=C_COUNTRY3CODE
WHERE I_COUNTRY3CODE is not null AND I_COUNTRY3CODE != 'OWID_WRL'
ORDER BY I_COUNTRY, I_YEAR;

/* ===== BLOCK: Block 7 ===== */

-- create Tournament table
CREATE OR REPLACE TABLE TOURNAMENT AS
SELECT
    "tournamentid"::int AS T_TOURNAMENT_ID,
    "TournamentName" AS T_TOURNAMENT_NAME,
    "StartDate"::date AS T_START_DATE,
    "EndDate"::date AS T_END_DATE,
    "Location" AS T_LOCATION,
    UPDATEDLOCATION AS T_COUNTRY,
    C_COUNTRY2CODE AS T_COUNTRY2CODE,
    "ONLINETOURNAMENTS" AS T_ONLINE,
    "GameId"::int AS T_GAME_ID,
    "TotalUSDPrize"::float AS T_PRIZE_USD,
    "Teamplay"::boolean AS T_WAS_IT_TEAM_PLAY
FROM API07TOURNAMENTBYID_LOCATONS_UPD
LEFT JOIN COUNTRY ON T_COUNTRY=C_COUNTRY
ORDER BY T_TOURNAMENT_ID, T_GAME_ID;

/* ===== BLOCK: Block 8 ===== */

-- creating results table for individual player tournaments
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
CREATE OR REPLACE TABLE TOURNAMENT_RESULTS_INDIVIDUAL AS
SELECT
    "TournamentId"::int AS TRI_TOURNAMENT_ID,
    "PlayerId"::int AS TRI_PLAYER_ID,
    "Ranking"::int AS TRI_RANKING,
    "RankText" AS TRI_RANK_TEXT,
    "PrizeUSD"::float AS TRI_PRIZE_USD
FROM API08_COMPLETEDATASET
ORDER BY TRI_TOURNAMENT_ID,TRI_PLAYER_ID;

/* ===== BLOCK: Block 10 ===== */

-- creating TOURNAMENT_TEAM table
CREATE OR REPLACE TABLE TOURNAMENT_TEAM AS
SELECT
    DISTINCT "TournamentTeamId"::int AS TT_TOURNAMENT_TEAM_ID,
    "TournamentTeamName" AS TT_TOURNAMENT_TEAM_NAME
FROM API09TOURNAMENTTEAMRESULTBYTOURNAMENTID_COMPLETE_DATASET
ORDER BY TT_TOURNAMENT_TEAM_ID;

/* ===== BLOCK: Block 11 ===== */

-- creation of results table for team tournaments

CREATE OR REPLACE TABLE TOURNAMENT_RESULTS_TEAM AS
WITH CTE_CNTPLAYERS AS (
SELECT
    "TournamentId"::int AS TOURNAMENT_ID,
    "TournamentTeamId"::int AS TOURNAMENT_TEAM_ID,
    IFNULL(COUNT(DISTINCT "PlayerId"),0)::int AS CNT_PLAYER_ID
FROM "API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET"
GROUP BY TOURNAMENT_ID, TOURNAMENT_TEAM_ID
ORDER BY TOURNAMENT_ID, TOURNAMENT_TEAM_ID, CNT_PLAYER_ID
)
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```
SELECT
    dev."TournamentId"::int AS TRT_TOURNAMENT_ID,
    dev."TeamId"::int AS TRT_TEAM_ID,
    dev."TeamName" AS TRT_TEAM_NAME,
    dev."TournamentTeamId"::int AS TRT_TOURNAMENT_TEAM_ID,
    dev."Ranking"::int AS TRT_TEAM_RANKING,
    dev."RankText" AS TRT_TEAM_RANK_TEXT,
    dev."PrizeUSD"::float AS TRT_TEAM_PRIZE_USD,
    dev."UnknownPlayerCount"::int AS TRT_CNT_UNKNOWN_PLAYERS,
    c.CNT_PLAYER_ID::int AS TRT_CNT_KNOWN_PLAYERS
FROM API09TOURNAMENTTEAMRESULTBYTOURNAMENTID_COMPLETE_DATASET dev
LEFT JOIN CTE_CNTPLAYERS c ON TRT_TOURNAMENT_TEAM_ID=c.TOURNAMENT_TEAM_ID
ORDER BY TRT_TOURNAMENT_ID, TRT_TEAM_ID, TRT_TOURNAMENT_TEAM_ID;

/* ===== BLOCK: Block 12 ===== */

-- creating a table of results of individual players in teams
CREATE OR REPLACE TABLE TOURNAMENT_RESULTS_PLAYER_IN_TEAM AS
WITH CTE_CNTPLAYERS AS (
       SELECT
           "TournamentId"::int AS TOURNAMENT_ID,
           "TournamentTeamId"::int AS TOURNAMENT_TEAM_ID,
           IFNULL(COUNT(DISTINCT "PlayerId"),0)::int AS CNT_PLAYER_ID
       FROM "API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET"
       GROUP BY TOURNAMENT_ID, TOURNAMENT_TEAM_ID
       ORDER BY TOURNAMENT_ID, TOURNAMENT_TEAM_ID, CNT_PLAYER_ID
       )

SELECT
    des."TournamentId"::int AS TRP_TOURNAMENT_ID,
    des."TournamentTeamId"::int AS TRP_TOURNAMENT_TEAM_ID,
    des."PlayerId"::int AS TRP_PLAYER_ID,
    (dev."PrizeUSD"::float / (dev."UnknownPlayerCount"::int + c.CNT_PLAYER_ID)) AS TRP_PRIZE_USD_FOR_PLAYER
FROM "API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET" des
LEFT JOIN "API09TOURNAMENTTEAMRESULTBYTOURNAMENTID_COMPLETE_DATASET" dev ON dev."TournamentTeamId"=des."TournamentTeamId"
```

```
JOIN CTE_CNTPLAYERS c ON c.TOURNAMENT_TEAM_ID=des."TournamentTeamId"
ORDER BY TRP_TOURNAMENT_ID, TRP_TOURNAMENT_TEAM_ID, TRP_PLAYER_ID;

/* ===== BLOCK: Block 13 ===== */

-- adding records about players who are in API08_COMPLETEDATASET (from 08 table is created TOURNAMENT_RESULTS_INDIVIDUAL, but without
player details)
-- and API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET (from 10 table is created TOURNAMENT_RESULTS_PLAYER_IN_TEAM, but
without player details)
-- tables but not in PLAYER table
INSERT INTO PLAYER
    (P_PLAYER_ID,
    P_NAME_FIRST,
    P_NAME_LAST,
    P_NICK_NAME,
    P_COUNTRY2CODE,
    P_PRIZE_USD)
-- select player detail data from table API08_COMPLETEDATASET
SELECT
    DISTINCT "PlayerId"::int AS PLAYER_ID,
    "NameFirst" AS NAME_FIRST,
    "NameLast" AS NAME_LAST,
    "CurrentHandle" AS NICK_NAME,
    "CountryCode" AS COUNTRY2CODE,
    SUM("PrizeUSD")::float AS PRIZE_USD
    FROM API08_COMPLETEDATASET
    -- select id of players who are in TOURNAMENT_RESULTS_INDIVIDUAL table but not in PLAYERS table
        WHERE PLAYER_ID IN
        (SELECT * FROM
            (SELECT DISTINCT TRI_PLAYER_ID AS PLAYERID
            FROM TOURNAMENT_RESULTS_INDIVIDUAL
            EXCEPT
            (SELECT DISTINCT P_PLAYER_ID AS PLAYERID
            FROM PLAYER)
            )
```

```
        )
        -- PLAYER_IDs greater than 900,000 are not real IDs,
        -- they only serve as a placeholder to keep track of the number of players and their winnings when the player is unknown.
        AND PLAYER_ID < 900000
    GROUP BY PLAYER_ID,NAME_FIRST,NAME_LAST,NICK_NAME,COUNTRY2CODE
UNION
    -- select player detail data from table API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET
    SELECT
        DISTINCT "PlayerId"::int AS PLAYER_ID,
        "NameFirst" AS NAME_FIRST,
        "NameLast" AS NAME_LAST,
        "CurrentHandle" AS NICK_NAME,
        "CountryCode" AS COUNTRY2CODE,
        SUM(TRP_PRIZE_USD_FOR_PLAYER)::float AS PRIZE_USD
    FROM "API10TOURNAMENTTEAMPLAYERByTOURNAMENTID_COMPLETE_DATASET"
    LEFT JOIN TOURNAMENT_RESULTS_PLAYER_IN_TEAM ON PLAYER_ID=TRP_PLAYER_ID
    -- select id of players who are in  TOURNAMENT_RESULTS_PLAYER_IN_TEAM table but not in PLAYERS table
        WHERE PLAYER_ID IN (
        SELECT * FROM
        (SELECT DISTINCT TRP_PLAYER_ID AS PLAYERID
        FROM TOURNAMENT_RESULTS_PLAYER_IN_TEAM)
        EXCEPT
        (SELECT DISTINCT P_PLAYER_ID AS PLAYERID
        FROM PLAYER)
        )
    GROUP BY PLAYER_ID, NAME_FIRST, NAME_LAST, NICK_NAME, COUNTRY2CODE
ORDER BY PLAYER_ID, NAME_FIRST, NAME_LAST, NICK_NAME, COUNTRY2CODE;

/* ===== BLOCK: Block 16 ===== */

-- UPDATE TOURNAMENT: add the tournaments that are in the TOURNAMENT_RESULTS_INDIVIDUAL and TOURNAMENT_RESULTS_TEAM tables,
-- BUT THEY'RE NOT IN THE TOURNAMENT TABLE
-- to the TOURNAMENTS table
INSERT INTO TOURNAMENT
       (T_TOURNAMENT_ID,
```

```
      T_PRIZE_USD)
-- select available data from TOURNAMENT RESULTS
SELECT * FROM
    (
    SELECT TRI_TOURNAMENT_ID::int AS T_TOURNAMENT_ID, SUM(TRI_PRIZE_USD)::float AS T_PRIZE_USD
    FROM TOURNAMENT_RESULTS_INDIVIDUAL
    GROUP BY T_TOURNAMENT_ID
    UNION ALL
    SELECT TRT_TOURNAMENT_ID::int AS T_TOURNAMENT_ID, SUM(TRT_TEAM_PRIZE_USD)::float AS T_PRIZE_USD
    FROM TOURNAMENT_RESULTS_TEAM
    GROUP BY T_TOURNAMENT_ID
    )
WHERE T_TOURNAMENT_ID IN
    (
    SELECT DISTINCT T_TOURNAMENT_ID FROM
        (
        SELECT TRI_TOURNAMENT_ID::int AS T_TOURNAMENT_ID
        FROM TOURNAMENT_RESULTS_INDIVIDUAL
        GROUP BY T_TOURNAMENT_ID
        UNION ALL
        SELECT TRT_TOURNAMENT_ID::int AS T_TOURNAMENT_ID
        FROM TOURNAMENT_RESULTS_TEAM
        GROUP BY T_TOURNAMENT_ID
        )
        EXCEPT
        SELECT T_TOURNAMENT_ID
        FROM TOURNAMENT
    );
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```
/* ===== BLOCK: Block 14 ===== */
--  AGGREGATED DENORMALIZED TABLE FOR THE STATISTICS OF PLAYER CORRELATIONS IN PYTHON AND TABLEAU
CREATE OR REPLACE TABLE PLAYER_GDP_POP_INT AS
SELECT
    C_COUNTRY AS GPI_COUNTRY,
    C_CONTINENT AS GPI_CONTINENT,
    COUNT(DISTINCT P_PLAYER_ID)::int as GPI_CNT_PLAYER_ID,
    AVG(GDP_GDP_PER_CAPITA_USD)::float as GPI_AVG_GDP_PER_CAPITA,
    AVG(POP_POPULATION)::float as GPI_AVG_POPULATION,
    AVG(POP_POPULATION_AGED15_19+POP_POPULATION_AGED20_29)::float as GPI_AVG_POPULATION_AGED_15TO29,
    ((COUNT(DISTINCT P_PLAYER_ID)*1000000)/AVG(POP_POPULATION))::float as GPI_CNT_PLAYERS_ON_1MIL_POP,
    AVG(I_SHARE_OF_INDIVID_USING_INTERNET)::float as GPI_AVG_SHARE_OF_INDIVIDUALS_USING_INTERNET
FROM PLAYER a
LEFT JOIN COUNTRY c ON upper(P_COUNTRY2CODE)=C_COUNTRY2CODE
LEFT JOIN GDP_PER_CAPITA g ON C_COUNTRY2CODE=GDP_COUNTRY2CODE
LEFT JOIN POPULATION p ON POP_COUNTRY2CODE=C_COUNTRY2CODE
LEFT JOIN INTERNET s ON I_COUNTRY2CODE=C_COUNTRY2CODE
WHERE
    GDP_GDP_PER_CAPITA_USD is not null
    -- ALL TABLES NEED TO BE UNIFIED FOR THE SAME PERIOD 1997-2021 (TABLES GDP, POP, INT ARE UP TO 2021)
    AND GDP_YEAR >= 1997
    AND POP_YEAR >= 1997
    AND I_YEAR >= 1997
    AND P_PLAYER_ID IN
        -- SELECT A PLAYER_ID OF EACH TOURNAMENT TO BE HELD BETWEEN 1997 and 2021
        (SELECT DISTINCT TRI_PLAYER_ID AS PLAYERID
        FROM TOURNAMENT_RESULTS_INDIVIDUAL
        WHERE TRI_TOURNAMENT_ID IN
               (SELECT DISTINCT T_TOURNAMENT_ID FROM TOURNAMENT WHERE YEAR(T_END_DATE) BETWEEN 1997 AND 2021)
               AND TRI_PLAYER_ID < 900000
               UNION
               SELECT DISTINCT TRP_PLAYER_ID AS PLAYERID
               FROM TOURNAMENT_RESULTS_PLAYER_IN_TEAM
               WHERE TRP_TOURNAMENT_ID IN
                     (SELECT DISTINCT T_TOURNAMENT_ID FROM TOURNAMENT WHERE YEAR(T_END_DATE) BETWEEN 1997 AND 2021))
```

```
GROUP BY C_COUNTRY, C_CONTINENT;
/* ===== BLOCK: Block 15 ===== */

-- AGGREGATED DENORMALIZED TABLES FOR PLAYERS RETIREMENT STATISTICS IN TABLEAU
CREATE OR REPLACE TEMPORARY TABLE TEMP_PLA_GAME_YEAR AS
-- join of the select below with the TOURNAMENT table
SELECT
       TOURNAMENT_ID::int AS TOURNAMENT_ID,
       PLAYER_ID::int AS PLAYER_ID,
       SUM(SUM_PRIZE::float) AS SUM_PRIZE,
       YEAR(T_END_DATE)::int as YEAR_OF_TOURNAMENT,
       T_GAME_ID::int AS GAME_ID
FROM
    (
    -- union of TOURNAMENT_RESULTS_INDIVIDUAL and TOURNAMENT_RESULTS_PLAYER_IN_TEAM tables
    SELECT
        TRI_PLAYER_ID::int AS PLAYER_ID,
        SUM(TRI_PRIZE_USD)::float AS SUM_PRIZE,
        TRI_TOURNAMENT_ID::int AS TOURNAMENT_ID,
        NULL AS TOURNAMENT_TEAM_ID
    FROM TOURNAMENT_RESULTS_INDIVIDUAL
    WHERE TRI_PLAYER_ID < 900000
    GROUP BY TRI_PLAYER_ID,TRI_TOURNAMENT_ID, TOURNAMENT_TEAM_ID
    UNION ALL
    SELECT
        TRP_PLAYER_ID::int AS PLAYER_ID,
        SUM(TRP_PRIZE_USD_FOR_PLAYER)::float AS SUM_PRIZE,
        TRP_TOURNAMENT_ID::int AS TOURNAMENT_ID,
        TRP_TOURNAMENT_TEAM_ID AS TOURNAMENT_TEAM_ID
    FROM TOURNAMENT_RESULTS_PLAYER_IN_TEAM
    GROUP BY PLAYER_ID, TOURNAMENT_ID, TOURNAMENT_TEAM_ID
    ) AS AAA
JOIN TOURNAMENT ON AAA.TOURNAMENT_ID=T_TOURNAMENT_ID
GROUP BY TOURNAMENT_ID, PLAYER_ID, YEAR_OF_TOURNAMENT, GAME_ID;
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
-- WHAT IS THE AVERAGE, MEDIAN AND QUANTILE VALUE FOR A GAME?
CREATE OR REPLACE TEMPORARY TABLE TEMP_AVG_MED_KVA_PERGAME AS
WITH cte AS (
       SELECT PLAYER_ID, GAME_ID, SUM(SUM_PRIZE) as SUM2_PRIZE
       FROM TEMP_PLA_GAME_YEAR
       GROUP BY PLAYER_ID, GAME_ID)

SELECT
       GAME_ID,
       AVG(SUM2_PRIZE) as AVG_PRIZE_FOR_GAME,
       MEDIAN(SUM2_PRIZE) as MED_PRIZE_FOR_GAME,
       PERCENTILE_CONT( 0.9 ) WITHIN GROUP (ORDER BY SUM2_PRIZE) as KVA09_PRIZE_FOR_GAME
FROM cte
GROUP BY GAME_ID
ORDER BY GAME_ID;


-- Merge TABLE with TEMP_PLA_GAME_YEAR and TEMP_AVG_MED_KVA_PERGAME
-- add columns if PRIZE >= AVG, MED, KVA PER GAME

CREATE OR REPLACE TEMPORARY TABLE TEMP_PLAYER_GAME_AVG_MED_KVA AS
WITH CTE_UNIK AS (
       SELECT pp.GAME_ID, pp.PLAYER_ID, SUM(pp.SUM_PRIZE) AS SUM2_PRIZE
       FROM TEMP_PLA_GAME_YEAR pp
       GROUP BY pp.GAME_ID, pp.PLAYER_ID)

SELECT c.GAME_ID, PLAYER_ID, SUM2_PRIZE,
    CASE WHEN SUM2_PRIZE >= KVA09_PRIZE_FOR_GAME THEN 1
        ELSE 0
        END as ISMORETHAN_KVA,
    CASE WHEN SUM2_PRIZE >= AVG_PRIZE_FOR_GAME THEN 1
        ELSE 0
        END as ISMORETHAN_AVG,
    CASE WHEN SUM2_PRIZE >= MED_PRIZE_FOR_GAME THEN 1
```

```
        ELSE 0
        END as ISMORETHAN_MED
FROM CTE_UNIK c
JOIN TEMP_AVG_MED_KVA_PERGAME a ON a.GAME_ID = c.GAME_ID
WHERE SUM2_PRIZE is not null
ORDER BY PLAYER_ID, c.GAME_ID, SUM2_PRIZE DESC;

-- the TEMP_PLAYER_GAME_AVG_MED_KVA table shows that MEDIAN is not suitable for our purposes because it divides the players ca into
two halves (48.6% and 51.4%).
-- The mean and quantile are relatively similar, the quantile is more in line with our idea, so we only work with the quantile below
-- The average divides the players into 83.6% and 16.7%
-- The quantile divides the players into 89.6% and 10.4%
-- Percentages should be viewed with the understanding that this is a rough preview that does not take into account fact,
-- that the players are always in the source table broken down by a game.

-- QUANTILE
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_KVA = 1; -- 7 149 = 10,4 %
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_KVA = 0; -- 61 849 = 89,6 %

-- AVERAGE
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_AVG = 1; -- 11 396 = 16,7 %
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_AVG = 0; -- 58 220 = 83,6 %

-- MEDIAN
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_MED = 1; -- 34 057 = 48,6 %
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
-- SELECT COUNT(DISTINCT PLAYER_ID)
-- FROM TEMP_PLAYER_GAME_AVG_MED_KVA
-- WHERE ISMORETHAN_MED = 0; -- 36 029 = 51,4 %


-- creating final tables

CREATE OR REPLACE TEMPORARY TABLE TEMP_WHOSEARLYERRETIRED_KVA AS

-- counting Players and Average count of years they played and summing Prizes for each game, adding Game names
SELECT
       x.GAME_ID::int AS W_GAME_ID,
       G_GAME_NAME AS W_GAME_NAME,
       x.ISMORETHAN_KVA::int AS W_ISMORETHAN_KVA,
       COUNT(DISTINCT x.PLAYER_ID)::int as W_CTN_PLAYERS,
       AVG(x.CNT_YEARS)::float as W_AVG_CNT_YEARS,
       SUM(x.SUM3_PRIZE)::float as W_SUM_PRIZE_USD
FROM
    -- adding Prize and Quantile by joining table TEMP_PLAYER_GAME_AVG_MED_KVA
    (
    SELECT
         tityp.GAME_ID,
         tityp.PLAYER_ID,
         tityp.CNT_YEARS,
         SUM(tg.SUM2_PRIZE) AS SUM3_PRIZE,
         ISMORETHAN_KVA
    FROM
        -- counting years of tournaments and grouping other collums
        (
        SELECT
            GAME_ID,
            PLAYER_ID,
            COUNT(DISTINCT YEAR_OF_TOURNAMENT) AS CNT_YEARS
        FROM
            -- adding year of tournament from Tournament table
```

```
            (
            SELECT GAME_ID, PLAYER_ID, YEAR(T_END_DATE) AS YEAR_OF_TOURNAMENT
            FROM
                -- core data: game, player, tournament ID is obtained by merging two tables
                (
                SELECT
                    GAME_ID,
                    PLAYER_ID,
                    TRI_TOURNAMENT_ID AS TOURNAMENT_ID
                FROM TEMP_PLAYER_GAME_AVG_MED_KVA
                LEFT JOIN TOURNAMENT_RESULTS_INDIVIDUAL ti ON PLAYER_ID=TRI_PLAYER_ID
                GROUP BY GAME_ID, PLAYER_ID, TRI_TOURNAMENT_ID
                UNION ALL
                SELECT
                    GAME_ID,
                    PLAYER_ID,
                    TRP_TOURNAMENT_ID AS TOURNAMENT_ID
                FROM TEMP_PLAYER_GAME_AVG_MED_KVA
                LEFT JOIN TOURNAMENT_RESULTS_PLAYER_IN_TEAM tt ON PLAYER_ID=TRP_PLAYER_ID
                GROUP BY GAME_ID, PLAYER_ID, TRP_TOURNAMENT_ID
                ORDER BY GAME_ID, PLAYER_ID
               ) tit
            LEFT JOIN TOURNAMENT t ON TOURNAMENT_ID=T_TOURNAMENT_ID
             ) tity
          GROUP BY GAME_ID, PLAYER_ID
          ) tityp
      JOIN TEMP_PLAYER_GAME_AVG_MED_KVA tg ON tg.GAME_ID=tityp.GAME_ID AND tg.PLAYER_ID=tityp.PLAYER_ID
      GROUP BY tityp.GAME_ID, tityp.PLAYER_ID, tityp.CNT_YEARS, ISMORETHAN_KVA
      -- ORDER BY tityp.PLAYER_ID,tityp.GAME_ID
   ) x
JOIN GAME g ON G_GAME_ID=x.GAME_ID
GROUP BY x.GAME_ID, G_GAME_NAME, x.ISMORETHAN_KVA
ORDER BY x.GAME_ID, G_GAME_NAME, x.ISMORETHAN_KVA DESC;
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```sql
-- creating final table, where sample of players for a game is more than 100 players (Because smaller sample is not statisticaly representative for us)
CREATE OR REPLACE TABLE WHOSEARLYERRETIRED_KVA AS
SELECT
       W_GAME_ID,
       W_GAME_NAME,
       W_ISMORETHAN_KVA,
       W_CTN_PLAYERS,
       W_AVG_CNT_YEARS,
       W_SUM_PRIZE_USD
FROM TEMP_WHOSEARLYERRETIRED_KVA
WHERE W_GAME_ID IN
        (SELECT DISTINCT W_GAME_ID FROM
            (
            SELECT W_GAME_ID, SUM(W_CTN_PLAYERS) AS SUM_PLAYERS
            FROM TEMP_WHOSEARLYERRETIRED_KVA
            GROUP BY W_GAME_ID
            HAVING SUM_PLAYERS >=100
            )
        );

-- it is necessary (for the desired visualizations in Tableau) to transform the table so that each column is split into two, depending on whether W_ISMORETHAN_KVA = 1 or 0.
CREATE OR REPLACE TABLE WHOSEARLYERRETIRED_KVA_ROWS AS
SELECT
    W_GAME_ID,
    W_GAME_NAME,
    SUM(IFNULL(CTN_PLAYERS_0,0)) AS CTN_PLAYERS_0,
    SUM(IFNULL(CTN_PLAYERS_1,0)) AS CTN_PLAYERS_1,
    SUM(IFNULL(SUM_PRIZE_USD_1,0)) AS SUM_PRIZE_USD_1,
    SUM(IFNULL(W_AVG_CNT_YEARS_0,0)) AS W_AVG_CNT_YEARS_0,
    SUM(IFNULL(W_AVG_CNT_YEARS_1,0)) AS W_AVG_CNT_YEARS_1
FROM
    (SELECT
        W_GAME_ID,
```

```
        W_GAME_NAME,
        0 AS CTN_PLAYERS_0,
        W_CTN_PLAYERS AS CTN_PLAYERS_1,
        0 AS SUM_PRIZE_USD_0,
        W_SUM_PRIZE_USD AS SUM_PRIZE_USD_1,
        0 AS W_AVG_CNT_YEARS_0,
        W_AVG_CNT_YEARS AS W_AVG_CNT_YEARS_1
    FROM TEMP_WHOSEARLYERRETIRED_KVA
    WHERE W_ISMORETHAN_KVA = 1
    UNION
    SELECT
        W_GAME_ID,
        W_GAME_NAME,
        W_CTN_PLAYERS AS CTN_PLAYERS_0,
        0 AS CTN_PLAYERS_1,
        W_SUM_PRIZE_USD AS SUM_PRIZE_USD_0,
        0 AS SUM_PRIZE_USD_1,
        W_AVG_CNT_YEARS AS W_AVG_CNT_YEARS_0,
        0 AS W_AVG_CNT_YEARS_1
    FROM TEMP_WHOSEARLYERRETIRED_KVA
    WHERE W_ISMORETHAN_KVA = 0)
GROUP BY W_GAME_ID, W_GAME_NAME
ORDER BY W_GAME_ID;


-- table for maximum age of players

CREATE OR REPLACE TABLE WHOSEARLYERRETIRED_MAXAGE AS
WITH CTE_PDAK AS (
   SELECT
       pp.TOURNAMENT_ID,
       pp.YEAR_OF_TOURNAMENT,
       pp.GAME_ID, pp.PLAYER_ID,
       P_BIRTH_YEAR AS YEAR_OF_BIRTH,
       pp.YEAR_OF_TOURNAMENT - P_BIRTH_YEAR AS AGE_ON_TOURNAMENT_YEAR,
```

```
       pp.SUM_PRIZE,
       CASE WHEN pp.SUM_PRIZE >= KVA09_PRIZE_FOR_GAME THEN 1
       ELSE 0
       END as ISMORETHAN_KVA
   FROM TEMP_PLA_GAME_YEAR pp
   JOIN PLAYER ON pp.PLAYER_ID = P_PLAYER_ID
   JOIN TEMP_AVG_MED_KVA_PERGAME a ON a.GAME_ID = pp.GAME_ID
   WHERE P_BIRTH_YEAR is not null
   )

SELECT
    x.GAME_ID::int AS W_GAME_ID,
    G_GAME_NAME AS W_GAME_NAME,
    x.ISMORETHAN_KVA::int AS W_ISMORETHAN_KVA,
    COUNT(DISTINCT x.PLAYER_ID)::int as W_CTN_PLAYERS,
    AVG(CNT_AGE)::float as W_AVG_CNT_YEARS,
    MAX(MAX_AGE)::int as W_MAX_AGE
FROM
    (
    SELECT
        GAME_ID,
        PLAYER_ID,
        COUNT(DISTINCT AGE_ON_TOURNAMENT_YEAR) as CNT_AGE,
        MAX(AGE_ON_TOURNAMENT_YEAR) as MAX_AGE,
        SUM(SUM_PRIZE) as SUM2_PRIZE,
        ISMORETHAN_KVA
    FROM CTE_PDAK
    GROUP BY GAME_ID, PLAYER_ID, ISMORETHAN_KVA
    ) x
JOIN GAME g ON G_GAME_ID=x.GAME_ID
GROUP BY x.GAME_ID, G_GAME_NAME, x.ISMORETHAN_KVA
ORDER BY x.GAME_ID, G_GAME_NAME, x.ISMORETHAN_KVA DESC;

-- just a control table for a distribution of players in their years played
CREATE OR REPLACE TABLE WHOSEARLIERRETIRED_PLAYERSBYYERS AS
```

PROJECT ESPORT – SQL: 02 CREATING TABLES

```
SELECT
    x.ISMORETHAN_KVA::int AS ISMORETHAN_KVA,
    CNT_YEARS,
    COUNT(DISTINCT x.PLAYER_ID)::int as CTN_PLAYERS
FROM
    (
    SELECT
        tityp.GAME_ID,
        tityp.PLAYER_ID,
        tityp.CNT_YEARS,
        SUM(SUM2_PRIZE) AS SUM3_PRIZE,
        ISMORETHAN_KVA,
        tityp.CNT_TOURNAMENTS
    FROM
        (
        SELECT
            GAME_ID,
            PLAYER_ID,
            COUNT(DISTINCT YEAR_OF_TOURNAMENT) AS CNT_YEARS,
            COUNT(DISTINCT TOURNAMENT_ID) AS CNT_TOURNAMENTS
        FROM
            (
            SELECT
                GAME_ID,
                PLAYER_ID,
                TOURNAMENT_ID,
                YEAR(T_END_DATE) AS YEAR_OF_TOURNAMENT
            FROM
                (
                SELECT
                    GAME_ID, PLAYER_ID,
                    TRI_TOURNAMENT_ID AS TOURNAMENT_ID
                FROM TEMP_PLAYER_GAME_AVG_MED_KVA
                LEFT JOIN TOURNAMENT_RESULTS_INDIVIDUAL ti ON PLAYER_ID=TRI_PLAYER_ID
                GROUP BY GAME_ID, PLAYER_ID, TRI_TOURNAMENT_ID
```

```
                UNION ALL
                SELECT
                    GAME_ID,
                    PLAYER_ID,
                    TRP_TOURNAMENT_ID AS TOURNAMENT_ID
                FROM TEMP_PLAYER_GAME_AVG_MED_KVA
                LEFT JOIN TOURNAMENT_RESULTS_PLAYER_IN_TEAM tt ON PLAYER_ID=TRP_PLAYER_ID
                GROUP BY GAME_ID, PLAYER_ID, TRP_TOURNAMENT_ID
                ORDER BY GAME_ID, PLAYER_ID
               ) tit
            LEFT JOIN TOURNAMENT t ON TOURNAMENT_ID=T_TOURNAMENT_ID
             ) tity
         GROUP BY GAME_ID, PLAYER_ID
         ) tityp
     JOIN TEMP_PLAYER_GAME_AVG_MED_KVA tg ON tg.GAME_ID=tityp.GAME_ID AND tg.PLAYER_ID=tityp.PLAYER_ID
     GROUP BY tityp.GAME_ID, tityp.PLAYER_ID, tityp.CNT_YEARS, tityp.CNT_TOURNAMENTS, ISMORETHAN_KVA
   ) x
GROUP BY ISMORETHAN_KVA, CNT_YEARS
ORDER BY ISMORETHAN_KVA, CNT_YEARS DESC;

/* ===== BLOCK: Block 15 ===== */
-- drop redundant columns
ALTER TABLE COUNTRY
DROP COLUMN C_COUNTRY3CODE, C_COUNTRY_UTF8;

ALTER TABLE TOURNAMENT
DROP COLUMN T_COUNTRY;
```