```
-- Question 1
-- SQL query to get Unique Players in the Data
-- DISTINCT gives unique event_user IDs, removing the repetitive ones present in the rest of the table
-- The DISTINCT user ID's are then counted in order to get the total number of unique players

SELECT
    COUNT(DISTINCT event_user)
FROM
    game_data;
```

Simple query, does not have much additional explanation.

Answer:

From pandas,
Unique players in the data: **1275**

```sql
-- Question 2 A
-- SQL query to get average number of slot machines per session
-- Used subquery: Subquery groups the game sessions together by using session_id (meaning
from opening the game to closing) and groups them by session_token which corresponds to the
number of times a slot machine is entered
-- Considered using unique slotmachine_id instead but since the term "number of slot machines"
was vague, I took it to mean the number of times any slot machine is entered, hence the
session_token
-- For the outside query: averages the number of slot machines per session and returns the
value
SELECT
    AVG(number_of_slot_machines)
        AS average_number_of_slot_machines
FROM
    (SELECT
        session_id,
            COUNT(DISTINCT session_token)
                AS number_of_slot_machines
    FROM
        game_data
    GROUP BY
        session_id);
```

🟨 - subquery that groups session_id together to count the number of machines that were accessed by the player

(DISTINCT session_token) - DISTINCT was added in order to not mix spins in the same machine session

Answer:

From pandas,
Average number of slot machines a player plays in a session: **1.637902175934536**

```
– Question 2 B
-- SQL query to get the average number of spins per machine session
-- Similar structure to the question before
-- Used subquery to get the count of spins per machine session by grouping the session_tokens together and counting the total per token into a column
-- Outside query calculates the average of all the spins_count column and returns it
SELECT
    AVG(spins_count)
        AS average_spins_count
FROM
    (SELECT
        session_token, COUNT(*)
            AS spins_count
    FROM
        game_data
    GROUP BY
        session_token);
```

🟨 - subquery that groups by session_token and counts the rows that belong to each token in order to represent a machine session

Answer:

From pandas,
Average number of spins per machine session: **107.53668495597805**

```
-- Question 3
-- SQL query to get the probability of hitting the various win_types
-- Used inline query to count the total count of win_type which is the total number of rows
-- Divided count per win_type over total count to get probability
-- Can add "* 100" in order to make percentages
SELECT
    win_type,
    (COUNT(*) / (
        SELECT
            COUNT(*)
        FROM
            game_data))
                AS win_type_count
FROM
    game_data
GROUP BY
    win_type;
```

🟨 - inline query that counts the total number of win_types by counting the rows

```
-- Question 4
-- SQL query to get the retention rate
-- Has 1 external query and 2 subqueries
-- subquery 1 returns the total number of unique players who returned after 24 hours have passed since install date
-- subquery 2 returns the total number of unique players
-- combined the two queries together in order to get a table where the two count values are side by side
-- dividing the two counts together and multiplying by 100 gives the percentage of players who returned
SELECT
   count_more_than_24_hours / total_count * 100
      AS retention_rate
FROM
   (SELECT
      COUNT(DISTINCT event_user)
         AS count_more_than_24_hours
   FROM
      game_data
   WHERE
      event_time > install_date + INTERVAL 24 HOUR)
         AS subquery1
      CROSS JOIN
   (SELECT
      COUNT(DISTINCT event_user)
         AS total_count
   FROM
      game_data)
         AS subquery2;
```

🟨	- subquery 1 which counts the unique returning players
🟩	- subquery 2 which counts the total unique players
🟦	- CROSS JOIN was used to combine the two tables (2 1x1 tables) into one (1 1x2 table)
🟦	- calculates the retention rate percentage

```
-- Question 5
-- SQL query to get the average RTP
-- nested query
-- summary query is the innermost query which calculates RTP per spin
-- RTP is then averaged together according to slotmachine_id
SELECT
  slotmachine_id, AVG(rtp) AS avg_rtp
FROM
  (SELECT
    slotmachine_id,
      amount / total_bet_amount AS rtp
  FROM
    game_data
      ) AS summary
GROUP BY
      slotmachine_id
```

🟨 - subquery which calculates the rtp per row

Answer:

From pandas,
Average RTP for each slot machine: **12.448979591836734**