Analysis of Gameplay Data from a Children's Education Games:
Part 1: 7 day Moving Average by User Type
Part 2: Measuring User Retention

Part 1: We'd like to look at gameplay data across user types for our games. There are three types of user types - teacher, home, or unknown.

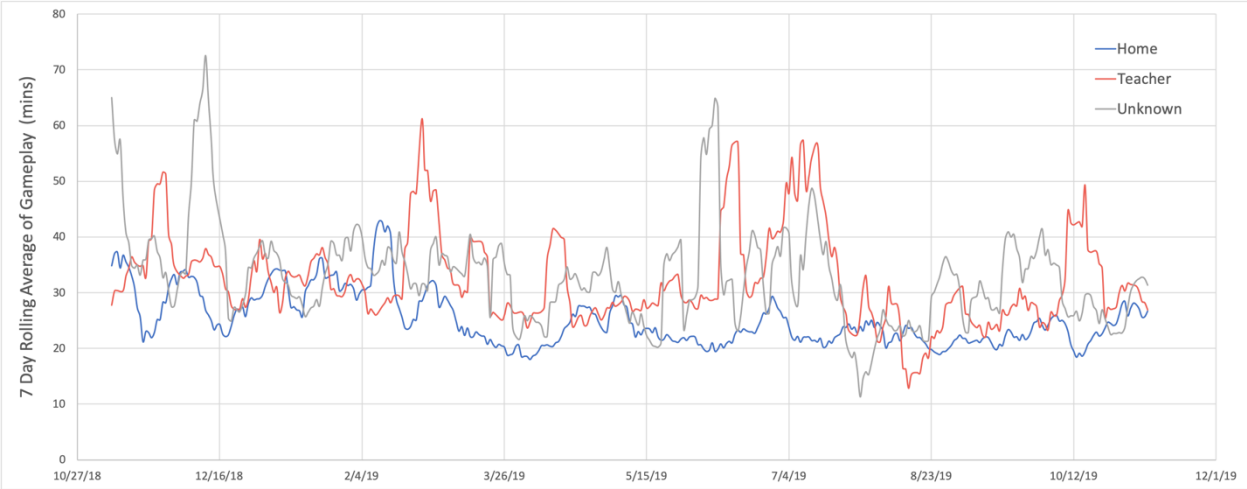
The data consists of the following:
device_id - unique device ID that identifies a device that has active game data.
date - date that the gameplay has occurred on.
is_teacher - nullable field - true if teacher, false if home data, unknown if null.
duration_ms - total milliseconds of active game session

The following SQL Query was used to calculate a 7 day moving averages for each user type. A moving average is helpful to identify trends beyond daily variation.

```
CREATE TABLE gameplay (
        device_id VARCHAR(20),
        date DATE,
         is_teacher TINYINT(1),
        duration_ms INT
        )
PARTITION BY LIST (is_teacher)(
        PARTITION unknown VALUES IN (NULL),
        PARTITION teacher VALUES IN (1),
        PARTITION home VALUES IN (0));

LOAD DATA LOCAL INFILE '…./gameplay_data_test.csv'
INTO TABLE gameplay
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

SELECT
        DISTINCT date,
        (AVG(duration_ms) OVER (
                ORDER BY date
                RANGE BETWEEN
                INTERVAL 6 DAY PRECEDING
                AND CURRENT ROW))/60000 as rolling_avg_teacher
FROM gameplay PARTITION (teacher);

SELECT
        DISTINCT date,
```

```
        (AVG(duration_ms) OVER (
            ORDER BY Date
            RANGE BETWEEN
             INTERVAL 6 DAY PRECEDING
             AND CURRENT ROW))/60000 as rolling_avg_unknown
FROM gameplay PARTITION (unknown);

SELECT
        DISTINCT date,
        (AVG(duration_ms) OVER (
              ORDER BY Date
              RANGE BETWEEN
             INTERVAL 6 DAY PRECEDING
             AND CURRENT ROW))/60000 as rolling_avg_home
FROM gameplay PARTITION (home);
```

The data produced was exported and graphed



Part 2 : Measuring User Retention

We'd like to look at user retention data across days. We have data on game sessions, measured in total milliseconds played during a game session. We define game day retention as follows - a given device will have another play session after the first day, no matter how far in the future that game day is. This is not typical to games that are played daily (think online games) but is more typical to family games such as board games, and is thus very useful. We'd like to measure falloff retention across multiple game days - how many users return and play a second time, third, etc.

The data is structured as follows:
device_id - unique device ID that identifies a device that has active game data.
date - date that the gameplay has occurred on.
duration_ms - total milliseconds of active game session.
source - ID of the game that was played in this session.

Data was partitioned by gametype. A recursive common table expression was used to generalize the query to find the percent of returning users and 'gameday' for game play sessions over 10 minutes. Indexes were added to the table columns (device_id, date and duration_ms) to improve performance.

*****Loading in data *****

```
CREATE TABLE game_retent (
        device_id VARCHAR(20),
        date DATE,
        duration_ms INT
        source VARCHAR(10))

   PARTITION BY LIST COLUMNS (source)(
        PARTITION game_a VALUES IN ("GAME_A"),
        PARTITION game_b VALUES IN ("GAME_B"),
        PARTITION game_c VALUES IN ("GAME_C"));

LOAD DATA LOCAL INFILE
        '…./gameretention_data.csv'
        INTO TABLE game_retent
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n'
         IGNORE 1 ROWS;
```

***** Adding Index *****

```
ALTER TABLE game_retent ADD INDEX new_index (device_id, date, duration_ms);
```

*****For game a *****

```
WITH RECURSIVE user AS
(SELECT
        device_id,
         date,
         1 AS counts
FROM game_retent PARTITION (game_a)
        WHERE duration_ms >= 600000

UNION ALL

SELECT
        gc.device_id,
        gc.date,
         u.counts + 1
FROM  game_retent PARTITION (game_a) AS gc,
        user AS u
                WHERE (gc.device_id = u.device_id
                        AND gc.date > u.date
                                AND duration_ms >= 600000 AND counts < 10)
                )
SELECT
        (COUNT(DISTINCT u.device_id))/ANY_VALUE(t.total)*100 AS game_a,
        u.counts AS day
FROM
        user u,
        (SELECT COUNT(DISTINCT device_id ) AS total
         FROM game_retent PARTITION (game_a)
                        WHERE duration_ms >= 600000 )  t
GROUP BY counts
```

*****For game_b*****

```
WITH RECURSIVE user AS
(SELECT
         device_id,
         date,
        1 AS counts
```

```
FROM game_retent PARTITION (game_b)
        WHERE duration_ms >= 600000

UNION ALL

SELECT
        gc.device_id,
        gc.date,
         u.counts + 1
FROM
        game_retent PARTITION (game_b) AS gc,
        user AS u
                WHERE (gc.device_id = u.device_id
                        AND gc.date > u.date
                                AND duration_ms >= 600000 AND counts < 10)
                )
SELECT
        (COUNT(DISTINCT u.device_id))/ANY_VALUE(t.total)*100 AS game_b,
        u.counts AS day
FROM
        user u,
        (SELECT
        count(DISTINCT device_id ) AS total
        FROM game_retent PARTITION (game_b)
                WHERE duration_ms >= 600000 ) t
GROUP BY counts
```

*****For game_c*****

```
WITH RECURSIVE user AS
(SELECT
         device_id,
         date,
        1 AS counts
FROM game_retent PARTITION (game_c)
        WHERE duration_ms >= 600000

UNION ALL

SELECT
        gc.device_id,
        gc.date,
         u.counts + 1
```

```
FROM
        game_retent PARTITION (game_c) AS gc,
        user AS u
                WHERE (gc.device_id = u.device_id
                        AND gc.date > u.date
                                AND duration_ms >= 600000)
                )
SELECT
        (COUNT(DISTINCT u.device_id))/ANY_VALUE(t.total)*100 AS percent_retained,
        u.counts AS day
FROM
        user u,
        (SELECT
                COUNT(DISTINCT device_id ) AS total
        FROM game_retent PARTITION (game_c)
                 WHERE duration_ms >= 600000 ) t
GROUP BY counts
```

The data produced was exported and graphed