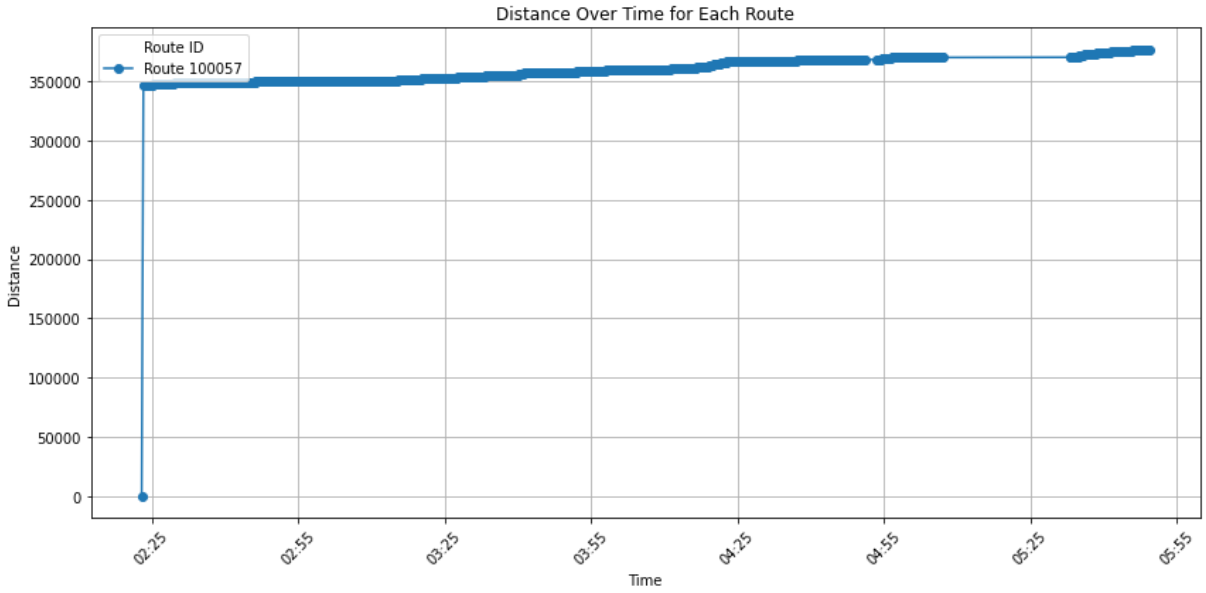
Target Case 1

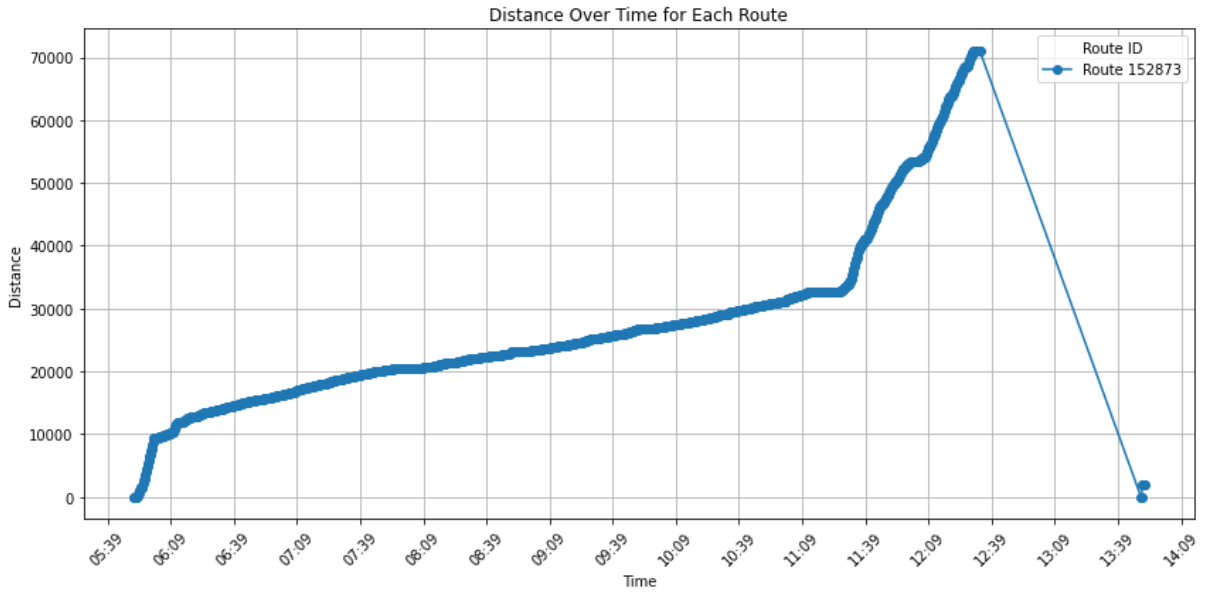
For each route, graphs of distance over time have been generated. The identified problems and insights specific to these graphs are explained below.

Problems and Insights

1. In the first graph, due to a device-related issue, the initial value recorded was 0, followed by a sudden jump in the distance. Additionally, no data was received for a while, which could be attributed to a network issue.



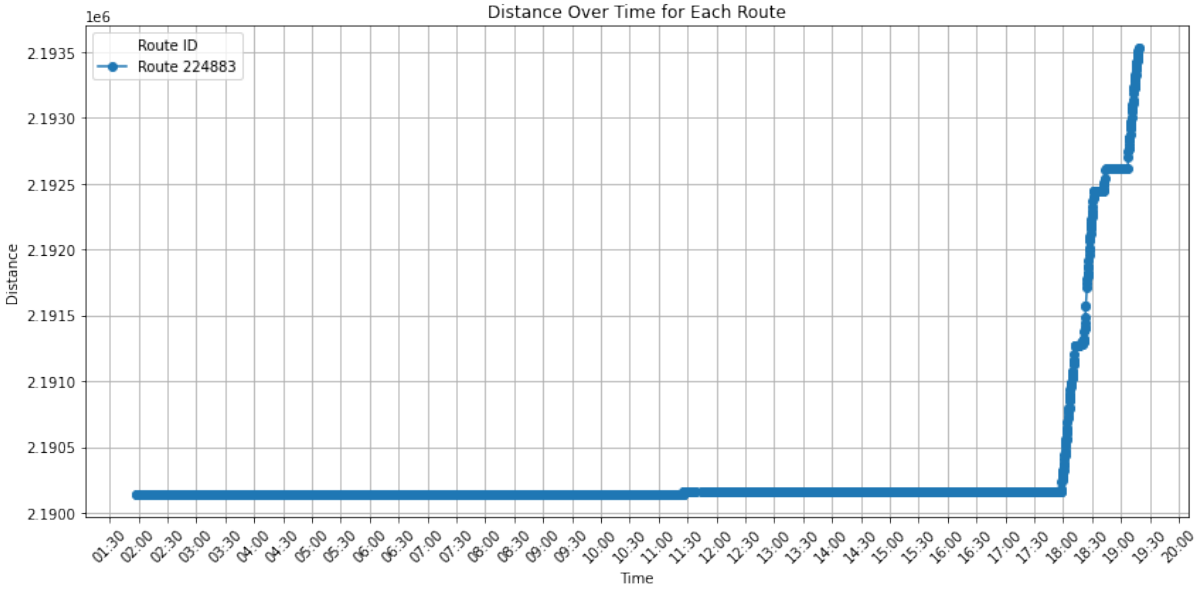
2. In the following graph, a reset occurred, and some data was recorded afterward. Similar situations have been observed in several other routes.



3. In another graph, the initial value is unexpectedly large instead of being 0. This might be due to a device-related error or because the device had not been reset for a long time.

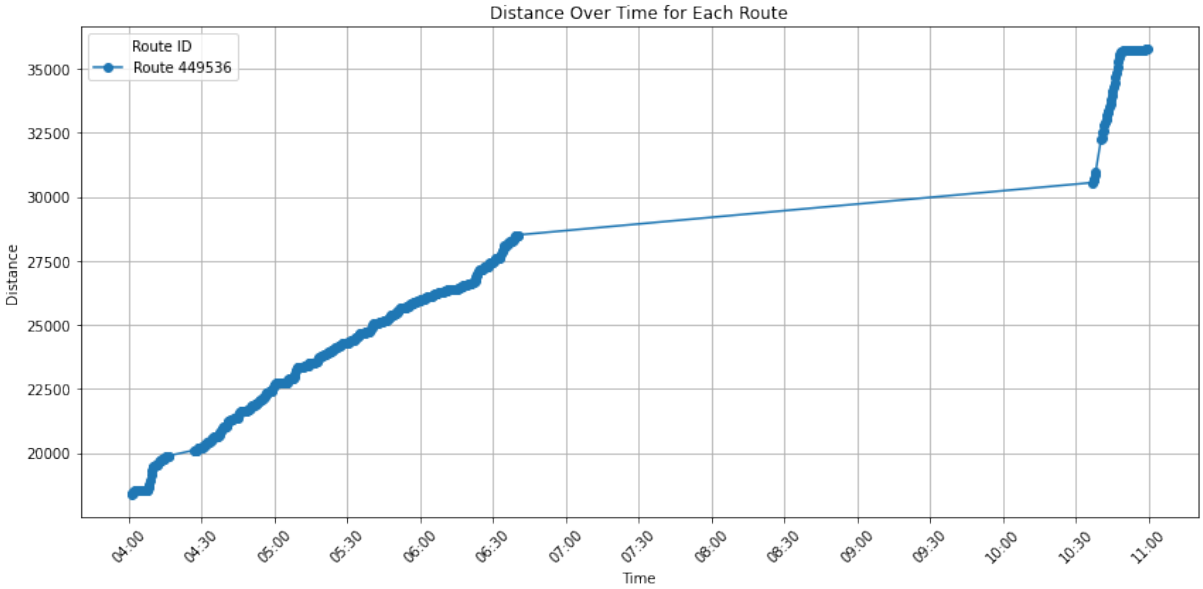
Afterward, the device appears to have remained idle for an extended period. This situation can be explained with two potential scenarios:

a. **Scenario 1:** The device might have encountered errors and stopped working for a long time. When it resumed, it uploaded the previously collected data, causing jumps in the graph.
b. **Scenario 2**: The device might have stayed on for a long time, but the vehicle remained stationary. Around 6 AM, the vehicle started moving again, and the distance values were added cumulatively as expected.



c.

4. In another graph, no data was received for an extended period due to network failures. However, the data appears to be consistent after the connection was restored, with no data loss observed. If similar patterns appear in other routes where no data is collected for a long time, but the values remain consistent, this could indicate a problem.



Possible Solution:

Since the distance values are cumulative, problems like jumps, non-zero initial values, and resets can affect the calculation of the total distance.
To address these issues:

- Calculate the difference between consecutive distance values to capture the incremental distance traveled for each record.
- The incremental distances are summed to get the total distance traveled for each route.

In ideal cases, the last recorded value would represent the total distance. However, due to resets and jumps, calculating the incremental distances and summing them is the most reliable way to determine the total distance for each route.

This step processes the raw data by calculating the previous distance and timestamp for each record. The LAG() function is used to retrieve the distance and recorded_at values from the previous entry for each row.

```
WITH valid_segments AS (
    SELECT
        route_id,
        distance,
        recorded_at,
        LAG(distance) OVER (PARTITION BY route_id ORDER BY recorded_at) AS prev_distance,
        LAG(recorded_at) OVER (PARTITION BY route_id ORDER BY recorded_at) AS prev_time
    FROM events.navigation_records
),
```

In this step, the distance and time differences are calculated. Additionally, the speed between two consecutive records (distance/time) is computed:

- If there is progress compared to the previous distance, the distance difference is calculated as (distance - prev_distance).
- The time difference is measured in seconds using TIMESTAMP_DIFF.
- Speed is calculated based on the distance and time differences.

This step enables the detection of abnormal speeds (e.g., sudden jumps in distance) based on the calculated distance and time differences.

```
distance_diff AS (
    SELECT
        route_id,
        recorded_at,
        IFNULL(TIMESTAMP_DIFF(recorded_at, prev_time, SECOND), 1) AS time_diff_sec,
        CASE
            WHEN prev_distance IS NULL OR distance >= prev_distance THEN distance - prev_distance
            ELSE 0
        END AS distance_travelled,
        CASE
            WHEN prev_distance IS NOT NULL AND distance >= prev_distance
                THEN (distance - prev_distance) / IFNULL(TIMESTAMP_DIFF(recorded_at, prev_time, SECOND), 1)
            ELSE 0
        END AS speed,
        prev_time
    FROM valid_segments
    WHERE prev_time IS NULL
        OR TIMESTAMP_DIFF(recorded_at, prev_time, SECOND) > 0
),
```

This section filters out abnormally high speeds (> 100 units/second). As a result, distances associated with these speeds are treated as zero.

```
filtered_distance AS (
    SELECT
        route_id,
        recorded_at,
        CASE
            WHEN speed > 100 THEN 0
            ELSE distance_travelled
        END AS valid_distance_travelled,
        prev_time,
        time_diff_sec
    FROM distance_diff
),
```

In this step, the durations where valid_distance_travelled is zero are accumulated. If the distance remains unchanged for a long period (e.g., 10 hours or 36,000 seconds), this situation is recorded. This step is intended to monitor how long the distance remains constant.

```
zero_streaks AS (
    SELECT
        route_id,
        recorded_at,
        valid_distance_travelled,
        SUM(CASE WHEN valid_distance_travelled = 0 THEN time_diff_sec ELSE 0 END)
        OVER (PARTITION BY route_id ORDER BY recorded_at ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS zero_duration
    FROM filtered_distance
),
```

In this step, if the distance remains unchanged for 10 hours (36,000 seconds) in zero_streaks and a change occurs afterward, the time of this first change is selected as the adjusted_start_time. This part is used to reset the start time when the distance remains constant for an extended period.

```
adjusted_times AS (
    -- Identify the first change after a 10-hour (36000 seconds) streak of zeros
    SELECT
        route_id,
        MIN(CASE WHEN zero_duration >= 36000 AND valid_distance_travelled > 0 THEN recorded_at ELSE NULL END) AS
adjusted_start_time
    FROM zero_streaks
    GROUP BY route_id
),
```

This step calculates the total distance, the start time (if there is a 10-hour gap, it uses the adjusted_start_time; otherwise, the original start time), and the end time for each route.

```
route_summary AS (
    SELECT
        fd.route_id,
        SUM(fd.valid_distance_travelled) AS total_distance,
        -- Use the adjusted start time if there's a gap, otherwise the original min time
        COALESCE(ast.adjusted_start_time, MIN(fd.recorded_at)) AS start_time,
        MAX(fd.recorded_at) AS end_time
    FROM filtered_distance fd
    LEFT JOIN adjusted_times ast ON fd.route_id = ast.route_id
    GROUP BY fd.route_id, ast.adjusted_start_time
)
```

In the final step, the total distance and total duration are calculated for each route.

```
SELECT
    route_id,
    total_distance,
    TIMESTAMP_DIFF(end_time, start_time, MINUTE) AS total_duration -- Duration in minutes
FROM route_summary;
```