**Processing GPS Data with Temporal Aggregation and Movement Path Construction**

By completing this lab, you will:

* Understand **date/time manipulation** in SQL.
* Learn to **group GPS points** into **fixed time intervals**.
* Use **PostGIS functions** to generate **movement paths**.
* Analyze the **distance traveled and movement duration**.

Total points: **9pts**

**Please submit the lab in word or pdf format, with commands and screenshots.**

## Data Introduction

aussie\_track\_points

A map of a city

AI-generated content may be incorrect.

Figure Track Points visualized using PostGIS

A map of a track point

AI-generated content may be incorrect.

Figure Track Points visualized in python

Assue\_track

A map of a city

AI-generated content may be incorrect.

**Figure 3 Aussie Track**

## Linear path from point observations

* Examples of applications: GPS tracking (e.g., vehicle movement, animal migration).
* Route reconstruction from timestamped observations.
* Mapping historical travel routes.

## Workflow

A brown rectangle with white text

AI-generated content may be incorrect.

## Step 1: Understanding Time Manipulation in SQL

* 1. Using DATE\_TRUNC to Round Timestamps (DATE\_TRUNC() function is used to round timestamps to a specific unit);
* The ‘-4’ at the end of the timestamp represents the time zone offset from Coordinated Universal Time (UTC), which mean UTC -4 hours

SELECT DATE\_TRUNC('minute', TIMESTAMP '2009-07-18 04:30:20-04');

Practice 1: what is the result of '2019-07-13 15:30:56-04' when round it to second?

SELECT DATE\_TRUNC(second, TIMESTAMP ''2019-07-13 15:30:56-04');

1.2 Extracting Date Parts (DATE\_PART)

The DATE\_PART function extracts **a specific part** of a timestamp, such as the minute or hour.

SELECT DATE\_PART('minute', TIMESTAMP '2009-07-18 04:31:20-04');

Practice 2: change ‘minute’ to ‘hour’ or ‘year’, and what is the result?

## Step 2: Normalizing Time Intervals

2.1 Convert Minutes to an Integer

To group GPS points into **15-minute intervals**, we adjust timestamps.

* CAST(expression AS target\_data\_type), this function converts data from one type to another.

SELECT CAST(DATE\_PART('minute', TIMESTAMP '2009-07-18 04:31:20-04') AS integer);

Practice 3: How about converting Minutes in column ‘time’ from ch11.aussie\_track\_points to integers and print it?

**2.2 Find the Offset Using MOD**

We determine how far each timestamp is from the **nearest 15-minute mark**.

* MOD(dividend, divisor); dividend is the number to be divided, divisor is the number by which the dividend is divided, and it returns the remainder after division. For example, 10 / 3 = 3 … 1

SELECT mod(CAST(DATE\_PART('minute', TIMESTAMP '2009-07-18 04:30:20-04') AS integer), 15);

Practice 4: How about replacing TIMESTAMP '2009-07-18 04:30:20-04' with column ‘time’ from ch11.aussie\_track\_points?

**2.3 Adjusting Time to Fit 15-Minute Bins**

* To normalize timestamps to the nearest 15-min interval, use the following query to ensure timestamps align to 0, 15, 30, and 45 minutes.
* When working with INTERVAL, you can specify the unit you need.

SELECT

DATE\_TRUNC('minute', time) -

CAST(

mod(

CAST(DATE\_PART('minute', time) AS integer), 15

) || ' minutes' AS interval

) AS track\_period

FROM ch11.aussie\_track\_points;

## Step 3: Finding Start and End Times in Each Interval

3.1 Compute Minimum (t\_start) and Maximum (t\_end) Timestamps for each 15-min window

* MIN(time): returns earliest recorded timestamp in the interval.
* MAX(time): returns latest recorded timestamp in the interval.

SELECT

DATE\_TRUNC('minute', time) -

CAST(

mod(CAST(DATE\_PART('minute', time) AS integer), 15) || ' minutes' AS interval

) AS track\_period,

*MIN(time) AS t\_start,*

*MAX(time) AS t\_end*

FROM ch11.aussie\_track\_points

GROUP BY track\_period;

## Step 4: Constructing Movement Paths

## 4.1 Creating a Line from Ordered GPS Points

We use ST\_MakeLine() to generate a movement path (ensure geom is ordered by time).

SELECT

DATE\_TRUNC('minute', time) -

CAST(

mod(CAST(DATE\_PART('minute', time) AS integer), 15) || ' minutes' AS interval

) AS track\_period,

MIN(time) AS t\_start,

MAX(time) AS t\_end,

*ST\_MakeLine(geom ORDER BY time) AS geom*

FROM ch11.aussie\_track\_points

GROUP BY track\_period;

## Step 5: Storing Results in a New Table

SELECT

DATE\_TRUNC('minute', time) -

CAST(

mod(CAST(DATE\_PART('minute', time) AS integer), 15) || ' minutes' AS interval

) AS track\_period,

MIN(time) AS t\_start,

MAX(time) AS t\_end,

ST\_MakeLine(geom ORDER BY time) AS geom

*INTO ch11.aussie\_run*

*FROM ch11.aussie\_track\_points*

GROUP BY track\_period;

**Exercise 01 (2pts):**

You are working with GPS movement data stored in the ch11.aussie\_run table. Your task is to **write an SQL query** that retrieves key movement statistics for each time period.

The query should:

1. **Create a table-like result** with the following columns:
   1. track\_period (Timestamp) – The normalized time period for grouping movements.
   2. t\_start (Timestamp) – The earliest recorded timestamp in the interval.
   3. t\_end (Timestamp) – The latest recorded timestamp in the interval.
   4. np (Integer) – The number of GPS points recorded in the interval.
   5. dist\_m (Integer) – The total distance traveled in meters.
   6. dur (Interval) – The duration of movement in the interval.

SELECT

CAST(track\_period AS timestamp),

CAST(t\_start AS timestamp) AS t\_start,

CAST(t\_end AS timestamp) AS t\_end,

ST\_NPoints(geom) AS np,

CAST(ST\_Length(geom::geography) AS integer) AS dist\_m,

(t\_end - t\_start) AS dur

FROM ch11.aussie\_run;

A screenshot of a computer

AI-generated content may be incorrect.