**Outage statistics:**

statistics(startTime in timestamp, endTime in timestamp)

*Algorithm:* Report for outage issues in the specified duration of time can be generated to find the statistics.

Number of outages which is recorded in the specified duration of time is calculated by comparing the start and end date in input to the dates in the ‘outage’ table.

The total count of houses affected will be calculated by comparing the zip-code of ‘outage’ table with the zip-code in ‘house\_address’ table.

Average number of houses affected will be computed by comparing the zip-code of ‘outage’ table with the zip-code in ‘house\_address’ table.

Average duration outage is computed by comparing start time and end time from ‘outage’ table.

*Input:*   1) startTime

2) endTime

*Output:* Show the number of outages in the range of dates specified.

Total count of houses affected.

Average count of houses affected.

Zip code with most outages reported

Average duration of outage

CREATE OR REPLACE PROCEDURE outageStats (startTime IN timestamp, endTime IN timestamp) IS

outage\_cnt int; --Holds the number of outages whose start time falls in the time periods specified by the parameters.

houses\_affected\_cnt int; --Holds the number of houses affected by the outage within the time period specified by the parameters.

avg\_number\_houses\_affected number; --Holds the average number of houses affected per outage.

avg\_outage\_duration varchar (30); --Holds the average duration of outage.

avg\_interval\_seconds number; --Holds the average number of seconds (for calculation purposes only).

zip\_code varchar (30); --Holds the zip code with the most number of outages.

BEGIN

SELECT count(\*) INTO outage\_cnt

FROM outage

WHERE start\_time >= startTime AND start\_time <= endTime; --Calculates total number of outages.

dbms\_output.put\_line('Total number of outages: ' || outage\_cnt);

SELECT count(distinct ha.hid) INTO houses\_affected\_cnt

FROM outage o

    INNER JOIN houses\_affected ha ON (o.oid = ha.oid)

WHERE start\_time >= startTime AND start\_time <= endTime;

--Calculates the total number of houses affected by all the outages in the specified time period.

dbms\_output.put\_line('Total number of houses affected: ' || houses\_affected\_cnt);

SELECT cast (avg (num\_houses) AS numeric (5,2)) AS avg\_num\_houses\_affected, avg (interval\_seconds), NUMTODSINTERVAL (avg (interval\_seconds), 'SECOND')

     INTO avg\_number\_houses\_affected, avg\_interval\_seconds, avg\_outage\_duration

FROM (

SELECT o.oid, count (ha.hid) AS num\_houses, (end\_time - start\_time) AS outage\_duration, (extract(second from end\_time - start\_time)

         + extract(minute from end\_time - start\_time) \* 60

         + extract(hour from end\_time - start\_time) \* 60 \* 60

         + extract(day from end\_time - start\_time) \* 60 \* 60 \* 24

           ) AS interval\_seconds

FROM outage o

    INNER JOIN houses\_affected ha ON (o.oid = ha.oid)

WHERE start\_time >= startTime AND start\_time <= endTime

GROUP BY o.oid, start\_time, end\_time

) t;

--We use a nested SQL query to determine the average number of houses affected per outage and the average duration of outage.

--The inner query creates an implicit table t which is queried by the outer query.

--This is necessary since we cannot use average a count function within a single query.

--Thus, we use the count function in the inner query and take those results and average them in the outer query.

--For instance, we count the total number of houses in the inner query (labeled as num\_houses) and average that count in the outer query.

--This gives us the average number of houses affected per outage. It must be noted that we used the cast function to limit the average to two decimal points.

--To calculate the average duration of outage, we subtracted the end time of the outage from the start time and converted it to seconds using the extract function.

--We summed up all the seconds in the inner query. We had to GROUP BY using outage id as well as start time and end time.

--We then averaged the sum of all the seconds. The NUMTODSINTERVAL function was used to convert the number into a timestamp.

dbms\_output.put\_line('Average number of houses affected per outage: ' || avg\_number\_houses\_affected);

dbms\_output.put\_line('Average duration of outage: ' || avg\_outage\_duration);

SELECT outage\_zip INTO zip\_code

FROM (

SELECT outage\_zip, count (\*) AS num\_outage, row\_number () OVER (ORDER BY count(\*) DESC) AS row\_num

FROM outage

WHERE start\_time >= startTime AND start\_time <= endTime

GROUP BY outage\_zip

ORDER BY outage\_zip) t

WHERE row\_num = 1;

--Here we determine the zip code with the most number of outages.

--For starters, we determine the number of outages in each zip code and assign each zip code with a row number in decreasing order.

--In essence, the zip code with the most number of outages is assigned a row number of 1.

--We then use an outer query to call upon the outage zip code where the assigned row number is 1.

--This provides us with the zip code that has the most frequent outages.

dbms\_output.put\_line('The zip code with the most number of outages is: ' || zip\_code);

/\*

SELECT \* FROM (

SELECT outage\_zip, count (\*) AS num\_outage

FROM outage

GROUP BY outage\_zip

ORDER BY outage\_zip) t

WHERE num\_outage = (SELECT max(count(\*)) FROM outage GROUP BY outage\_zip);

\*/

EXCEPTION

   WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('No outage data found.');

END;

EXEC outageStats (timestamp '2002-03-01 09:00:30.75', timestamp '2015-06-09 08:32:22.87');

SELECT \* FROM outage;

SELECT \* FROM houses\_affected;