# Rentr

> *Vehicle rental service*

**Group 10**: *Francis Piché, Joe Bashour, Laetitia Fesselier, Nuri Amiraslan*

## Application Report

### Stored Procedure

A plan fine was added. We decided to incur a penalty, if a bike was not returned after borrowing it for a trip.

**Implementation:**

```
DROP FUNCTION IF EXISTS surcharge(p_date DATE);
CREATE OR REPLACE FUNCTION surcharge(p_date DATE)
RETURNS VOID AS $$

DECLARE
last_proc INT;
trip RECORD;

-- if bike is not returned after a day
incomplete_trips CURSOR
FOR SELECT t.vtype, t.userid, t.s_time, t.station, p.name, p.price
FROM
(SELECT t.vtype, userid, s_time, s.name AS station
FROM trips AS t
JOIN stations AS s ON s.sid = t.startsat
WHERE t.vtype = 'electric-bike' OR t.vtype = 'bike'
AND endsat IS NULL AND DATE_PART('day', p_date - s_time) >= 1
AND  DATE_PART('day', p_date - s_time) < 2) AS t

JOIN (SELECT p.name, p.price, f.vtype
FROM plans AS p
NATURAL JOIN surcharges AS s
NATURAL JOIN isFor As f
WHERE stype = 'not-returned' ) AS p
ON p.vtype = t.vtype;

BEGIN

last_proc := (SELECT DATE_PART('day', p_date - t.date)
```

```
FROM transactions as t
NATURAL JOIN surcharges as s
WHERE s.stype = 'not-returned'
ORDER BY t.date DESC
LIMIT 1);

IF last_proc < 1 THEN
    RETURN;
END IF;

-- Open the cursor
OPEN incomplete_trips;

LOOP
    -- fetch row
      FETCH incomplete_trips INTO trip;

    -- exit when no more row to fetch
      EXIT WHEN NOT FOUND;

    -- build the output
        INSERT INTO transactions (userid, name, status, date, subtotal, comment)
        VALUES (trip.userid, trip.name , 1, NOW(), trip.price, 'Bike not returned for trip made on ' || trip.s_time || ' from ' || trip.station);
END LOOP;

-- Close the cursor
CLOSE incomplete_trips;

END; $$

LANGUAGE plpgsql;
```

**Demonstration:**

**Before:**

In trips, we have two rides for which the bike has not been returned for more than a day.

```
    s_time           |   e_time   | userid | startsat | endsat | serialnb |  vtype
-----------------------------+------------------------------+--------+----------+--------
 2019-03-22 13:20:00 |            |      3 |     6049 |        |       8  | bike
 2019-03-22 12:30:00 |            |      1 |     6049 |        |       11 | bike
```

**Running it:**

```
-- Add transaction surcharges
SELECT surcharge(CURRENT_DATE);
-- Current date being: 24 march
```

**After:**

2 new transactions have been created (tranid #396 & 397).

```
tranid | userid | name |status|         date        | subtotal |  comment
------+--------+--------------------------------------+--------+----------------------
395    |    6   | 1-trip biking|   1   | 2019-03-20 00:00:00 |  23.00   |

396    |    3   | Fine - Vehicle not returned - Bikes |    1   |    2019-03-24 19:50:00
    |  150.00  | Bike not returned for trip made on 2019-03-22 13:20:00 from Queen /
Wellington

397    |    1   | Fine - Vehicle not returned - Bikes |    1   |    2019-03-24 19:50:00
    |  150.00  | Bike not returned for trip made on 2019-03-22 12:30:00 from Queen /
Wellington
```

---

## User Interface

Will be demonstrated during the TA demo.

---

## Indexing

We created 2 indexes. Here they are, along with the motivation behind them:

| Index Name | Table | Motivation |
|---|---|---|
| idx_email | users | Used a lot when user is checking their transactions, personal info, and/or favorite station. |
| idx_vehicles_state | vehicles | Used a lot when displaying available vehicles for users. |

---

## Data Visualization

For data analysis, we are interested in knowing the sales for each month of the year, for ALL the years between 2013 & 2018 (*inclusive*).

Moreover, we are also interested to know how many active users there are within each age bracket. We grouped the ages from 12 to 89 into age brackets of 6 years each. Finally, by *active users*, we mean users who actually went on trips during those years.

*Note: CSV files contain the data from the queries. Some manipulations were done when creating the pivot tables used in creating the charts. CSV files could be found in the submission folder under:*

```
"Data Visualisation" -> "sales" -> "sales.csv"
&
"Data Visualisation" -> "age distribution" -> "age.csv"
```

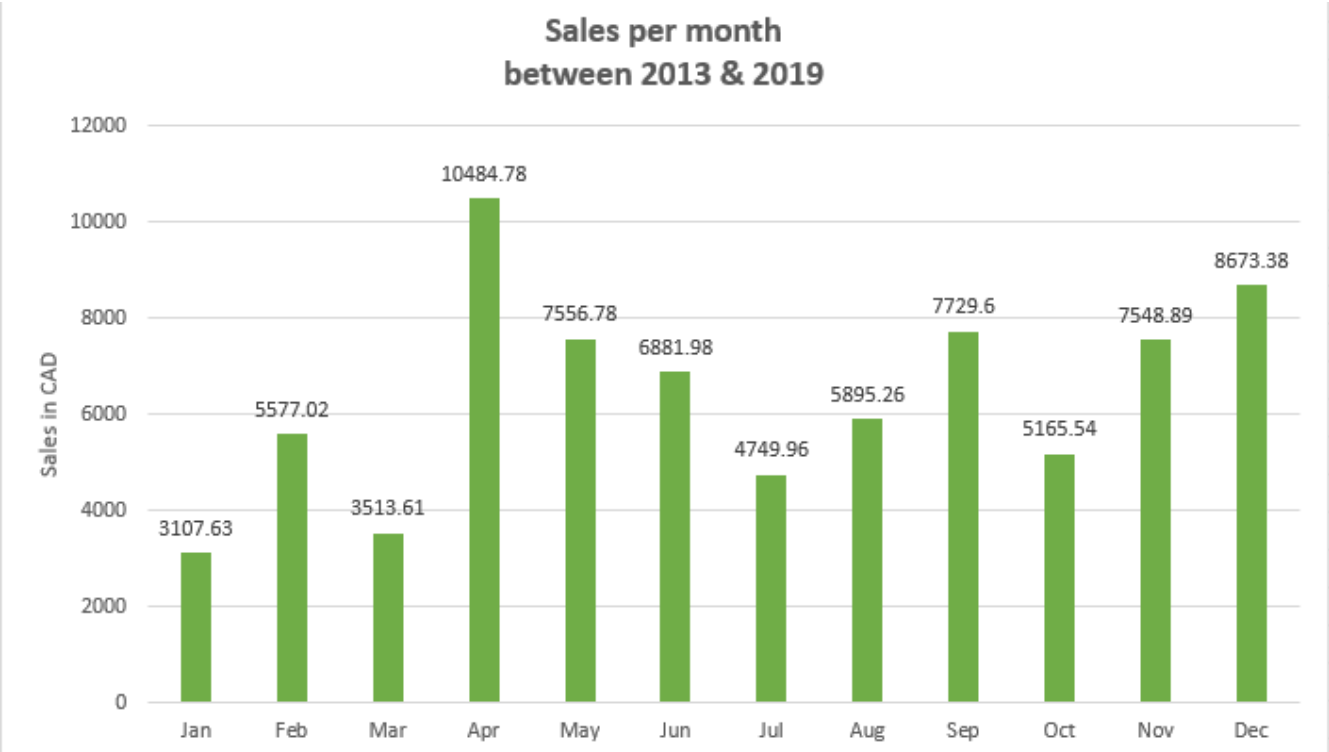
**Sales Analysis**

**1. Query:**

CSV data was generated based on results from following query.

```
SELECT * FROM transactions WHERE date >= '2013-01-01' AND date < '2019-01-01';
```

**2. Data Chart:**




***: between 2013 & 2019 in chart's title actually means from 2013 and up until end of 2018*

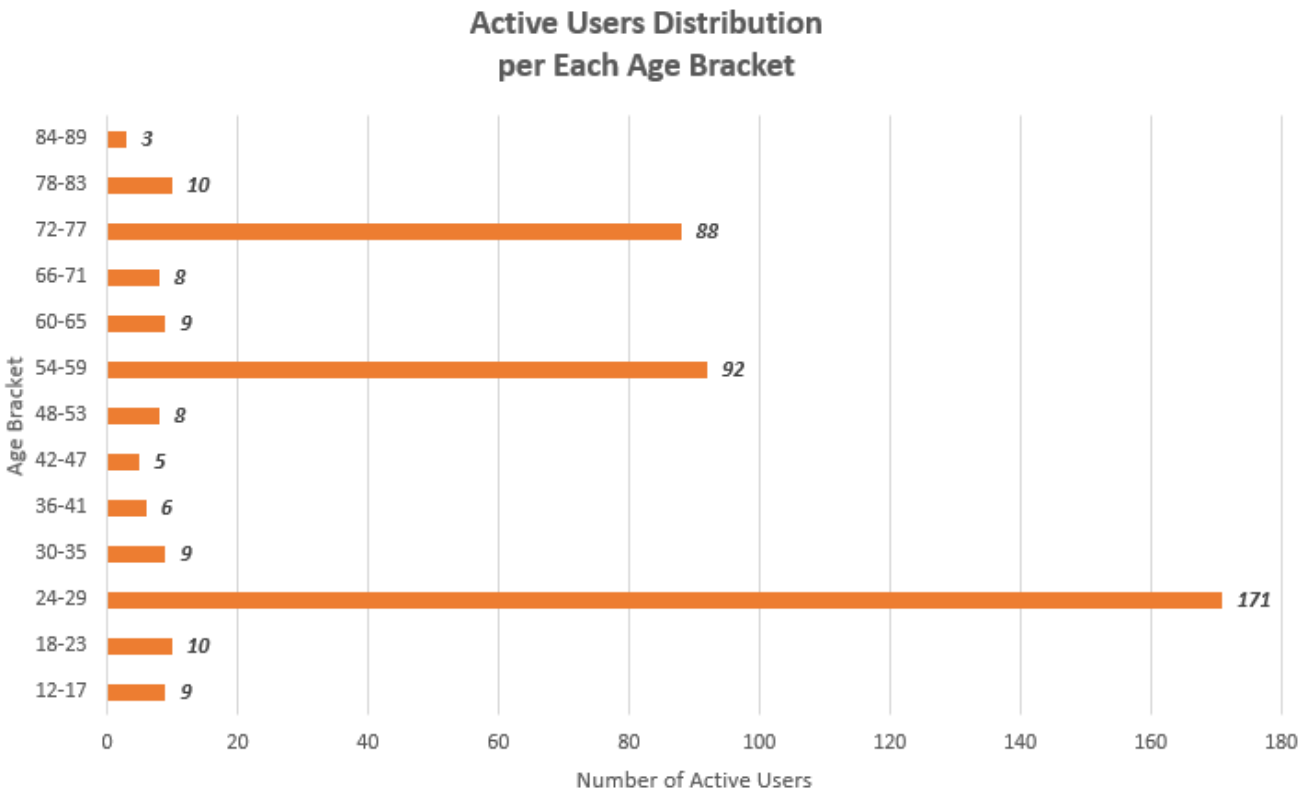
**Age Analysis**

**1. Query:**

CSV data was generated based on results from following query.

```
select users.age, count(users.userid) as Members from users join trips on users.userid
= trips.userid group by users.age order by users.age asc;
```

**2. Data Chart:**




---

## Creativity

In addition to having implemented a complex user interface, we also created a trigger that updates a given vehicle's location when the vehicle is returned to a certain station at the end of a trip.

**Implementation:**

```
CREATE OR REPLACE FUNCTION update_vehicle_location()
    RETURNS trigger AS
```

```
$BODY$
BEGIN
    UPDATE vehicles
    SET station = NEW.endsat
    WHERE serialnb = NEW.serialnb;
    RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql;

CREATE TRIGGER update_vehicle_location_trigger
  BEFORE INSERT OR UPDATE OF endsat ON trips
  FOR EACH ROW
  EXECUTE PROCEDURE update_vehicle_location();
```

**Demonstration:**

```
UPDATE trips
    SET endsat = 7030
    where s_time = '2019-03-24 13:20:00' and userid = 3;
```

**Results:**

```
 serialnb |     make      |    model   | veh_state  | capacity |  vtype   | station
----------+---------------+------------+------------+----------+----------+---------
    1     |     Jeep      | Grand Cherokee| good shape |  7 | regular car   |
    2     | Mercedes-Benz | CL-Class      | good shape |  2 | regular car   |
    3     | Volkswagen    | Golf          | good shape |  4 | regular car   |
    4     | Honda         | Prelude       | good shape |  2 | regular car   |
    5     | Chevrolet     | Tahoe         | good shape |  7 | regular car   |
    7     | GMC           | Savana        | good shape |  7 | regular car   |
    11    | Ford          | Mustang       | good_shape |  2 | luxury car    |    7030
```

**Thank you!**