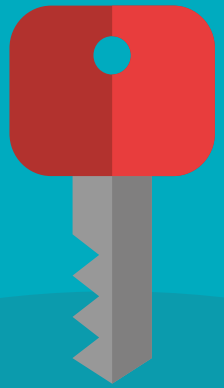
Traffic Detection

Camila Barbagallo, Paula García,
Juan Gil, Rocío González,
Oriol Vall, Valeria Zaldivar

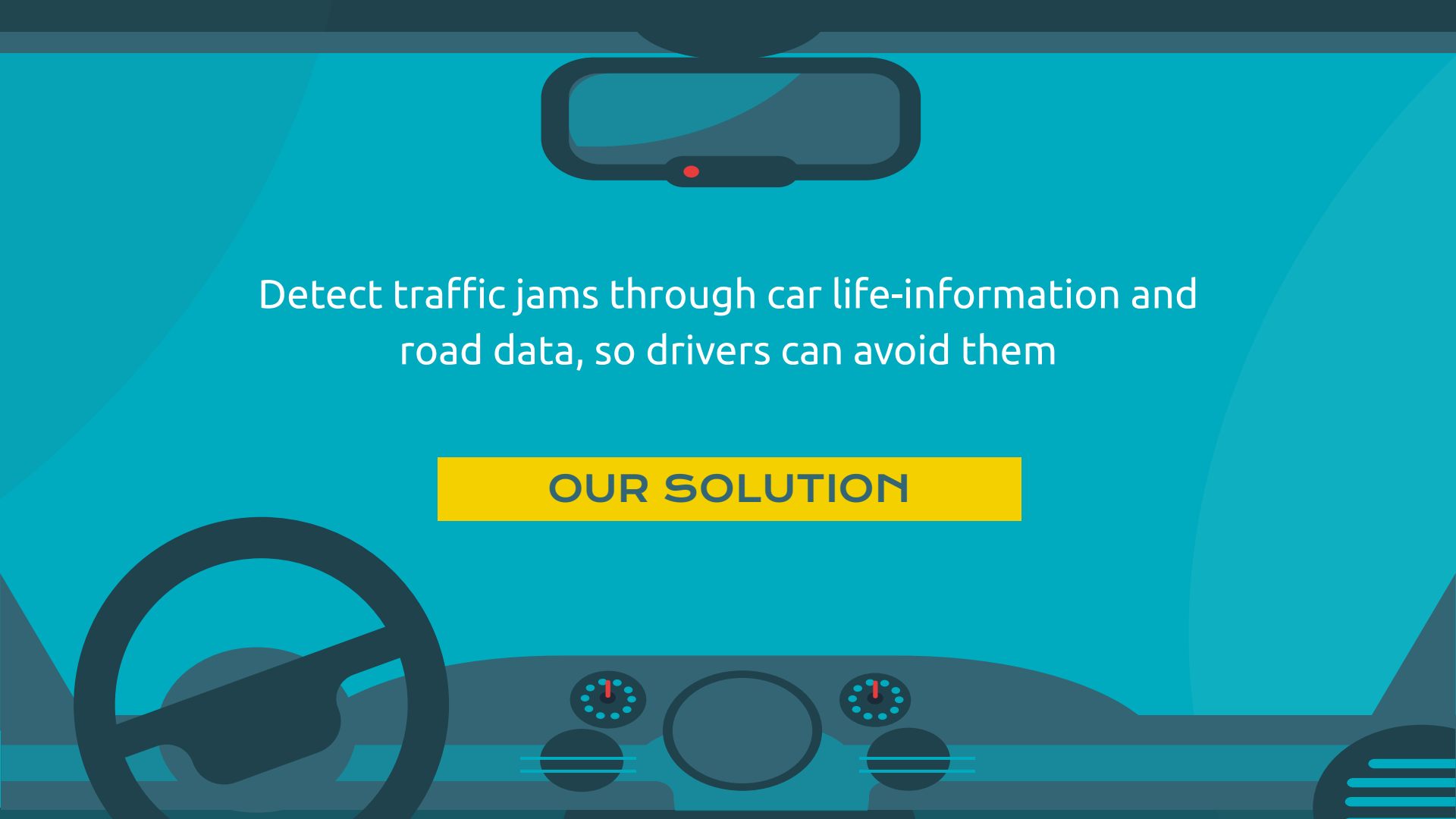
Table of Contents

01

Business



https://www.lavanguardia.com/motor/rankings/20180330/441822247588/cuantas-horas-ano-atascado-trafico.html


Detect traffic jams through car life-information and road data, so drivers can avoid them
OUR SOLUTION

Our Customer

- Government
- VTC
- Self-driving vehicle companies
- Web Mapping Companies

DIRECT CUSTOMERS

- Emergency Services
- Average driver

INDIRECT CUSTOMERS

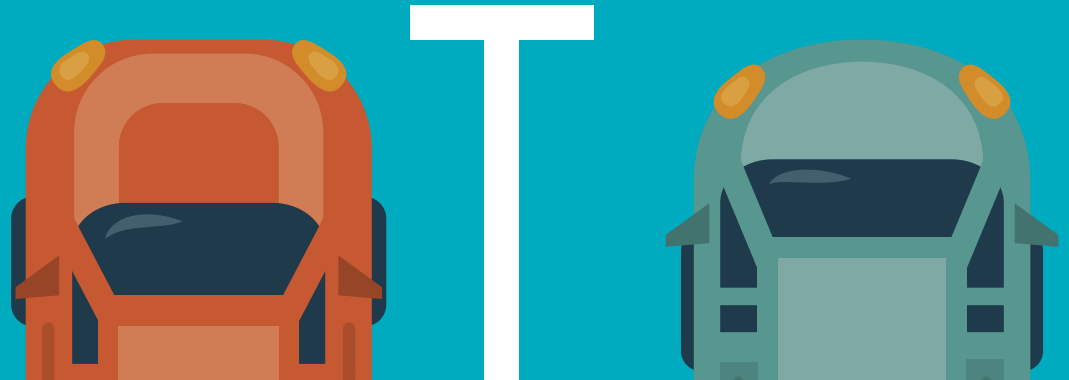
Revenue Model

Revenue

- Personalised Ads
- Pay for service
  (Platforms implementing it)

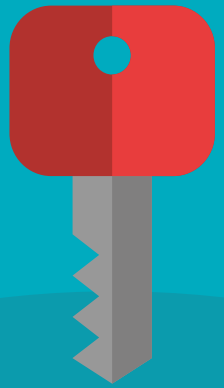
Funding

- Government
- Established platforms (exclusivity)
- Investors



02

Data Pipeline



What are our inputs?

Streaming

Slow Changing

Vehicle:

- vehicle_id (string)
- datetime_utc (int)
- latitude (float)
- longitude (float)
- accelerometer (float)
- vehicle_type (string)

Roads:

- road_name (string)
- number_lanes (int)
- min_speed (float)
- has_bus_lane (binary)

Assumptions

Data Flow

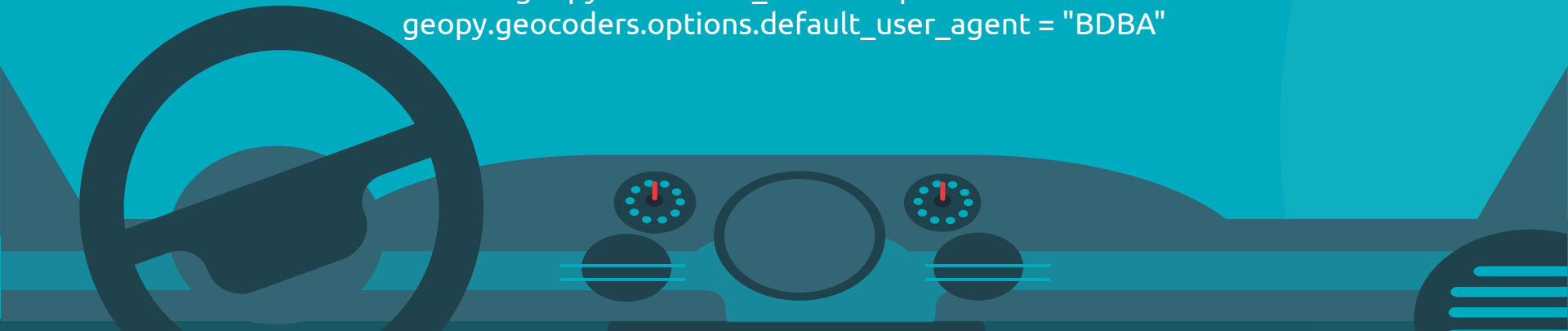
We will only receive the data once the motor is on

Traffic (300m)

More than 50 cars per lane going at a speed lower than the roads' minimum

Libraries

```
from pyspark import SparkContext
from pyspark.sql import SparkSession, Row
from pyspark.sql.types import StructType
from pyspark.sql.functions import monotonically_increasing_id, row_number
from pyspark.sql.window import Window

import geopy
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter
geopy.geocoders.options.default_user_agent = "BDBA"
```



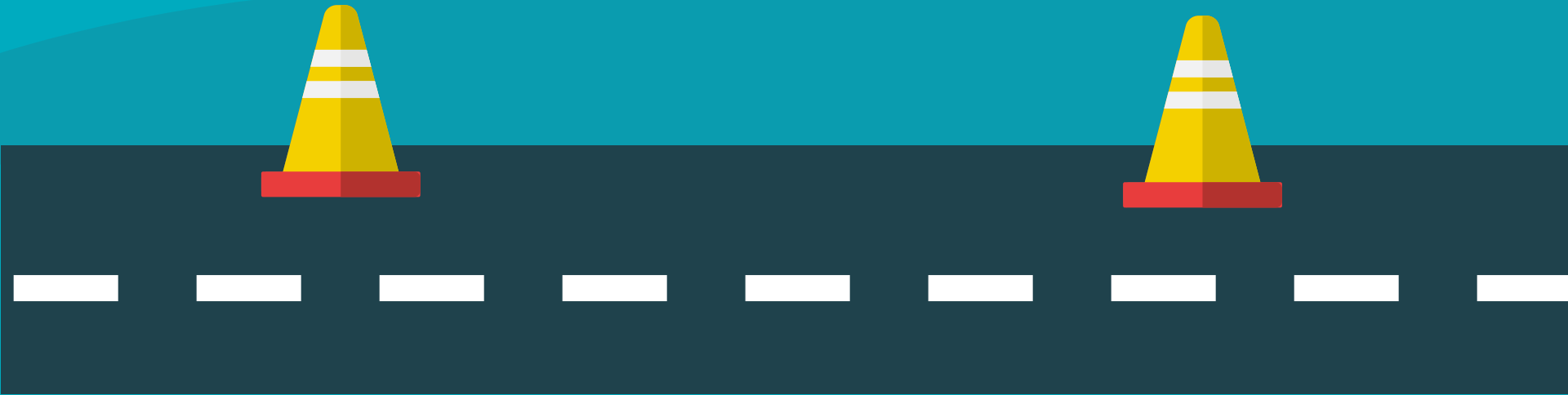
Read Data

Vehicles

Roads

```
vehic= spark.read.json('vehicles.json')
```

```
roads= spark.read.json('roads.json')
```

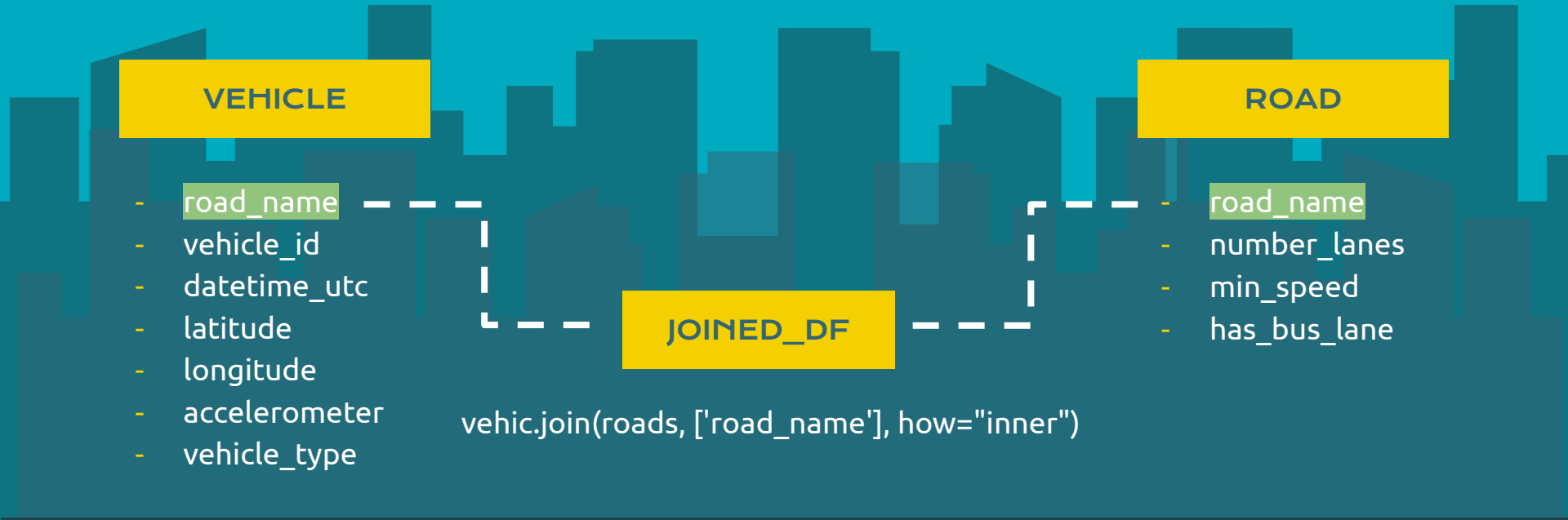


```
def road_from_coord(lat, lon):

    coordinates= str(lat)+ ',' +str(lon)
    locator = Nominatim(timeout=10)
    rgeocode = RateLimiter(locator.reverse, min_delay_seconds=0.001)
    location = rgeocode(coordinates)

    return(location.raw['address']['road'])
```

Road names

```
road_names = []
for i in tqdm(range(vehic.count())):
        road_names.append(road_from_coord(vehic.collect()[i]['latitude'],vehic.collect()[i]['longitude']))

row = Row("road_name")
rdd = spark.sparkContext.parallelize(road_names)
rf=rdd.map(row).toDF()

vehic=vehic.withColumn('row_index',
row_number().over(Window.orderBy(monotonically_increasing_id())))
rf=rf.withColumn('row_index', row_number().over(Window.orderBy(monotonically_increasing_id())))

vehic = vehic.join(rf, ["row_index"],how="inner").drop("row_index")
```

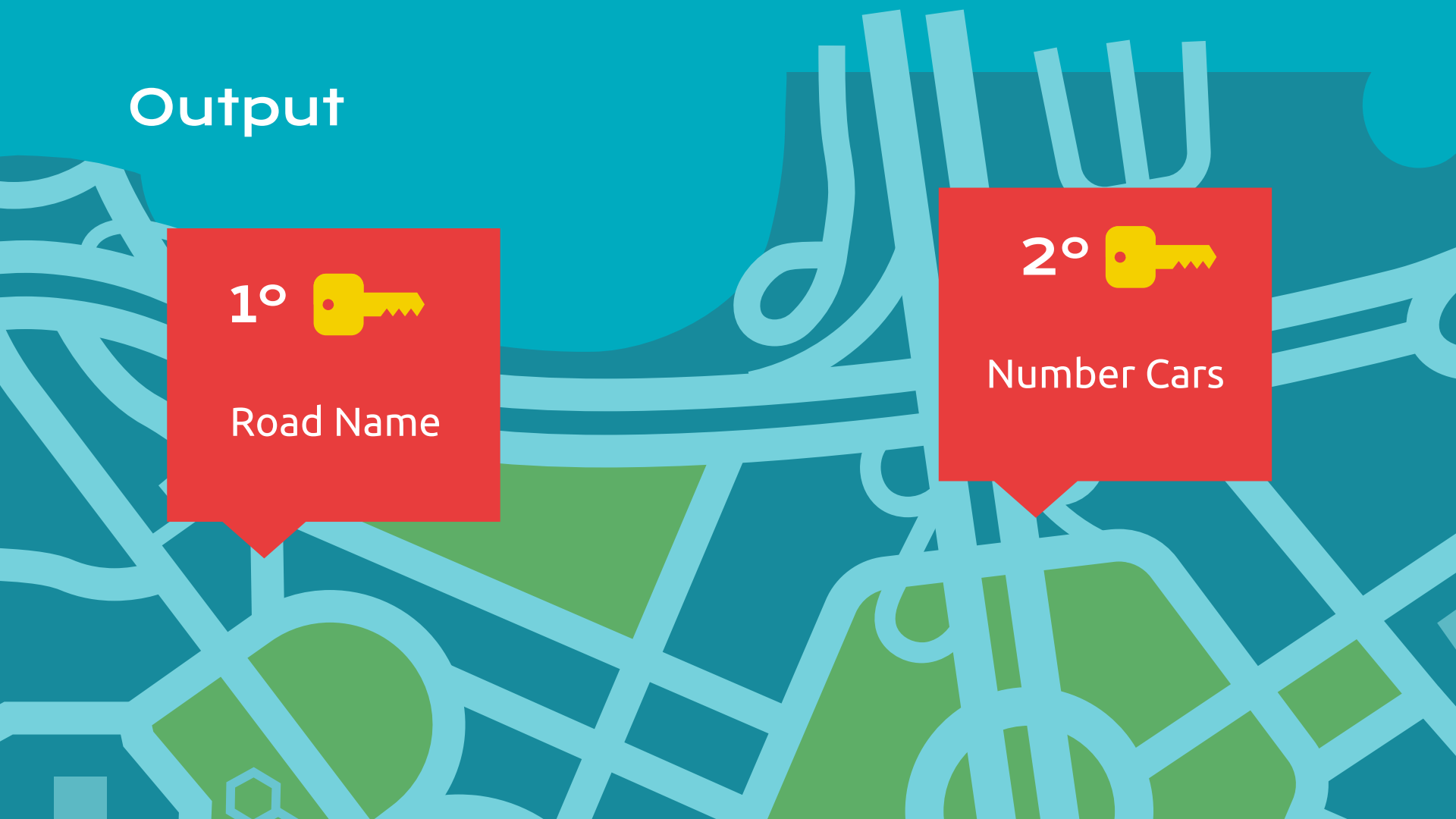
Join of Information



vehic.join(roads, ['road_name'], how="inner")

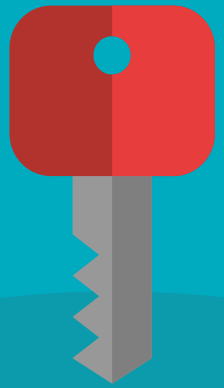

```
joined_df.createOrReplaceTempView("joined_df")
results = spark.sql("select road_name, count(vehicle_id) from joined_df\
            where accelerometer < min_speed \
            group by road_name, number_lanes, has_bus_lane\
            having count(vehicle_id) > (50*(number_lanes-has_bus_lane))")

results.write.csv("td_results_6000.csv")
```

Output



03

Proof of Concept



Use Case

Vehicles Set

4646

Faker

datetime_utc: fake.date_time_between_dates(datetime_start, datetime_end=).timestamp()
vehicle_id - digits: str(fake.pyint(min_value=0, max_value=9, step=1))
accelerometer: fake.pyint(min_value=0, max_value=180, step=1)

Random

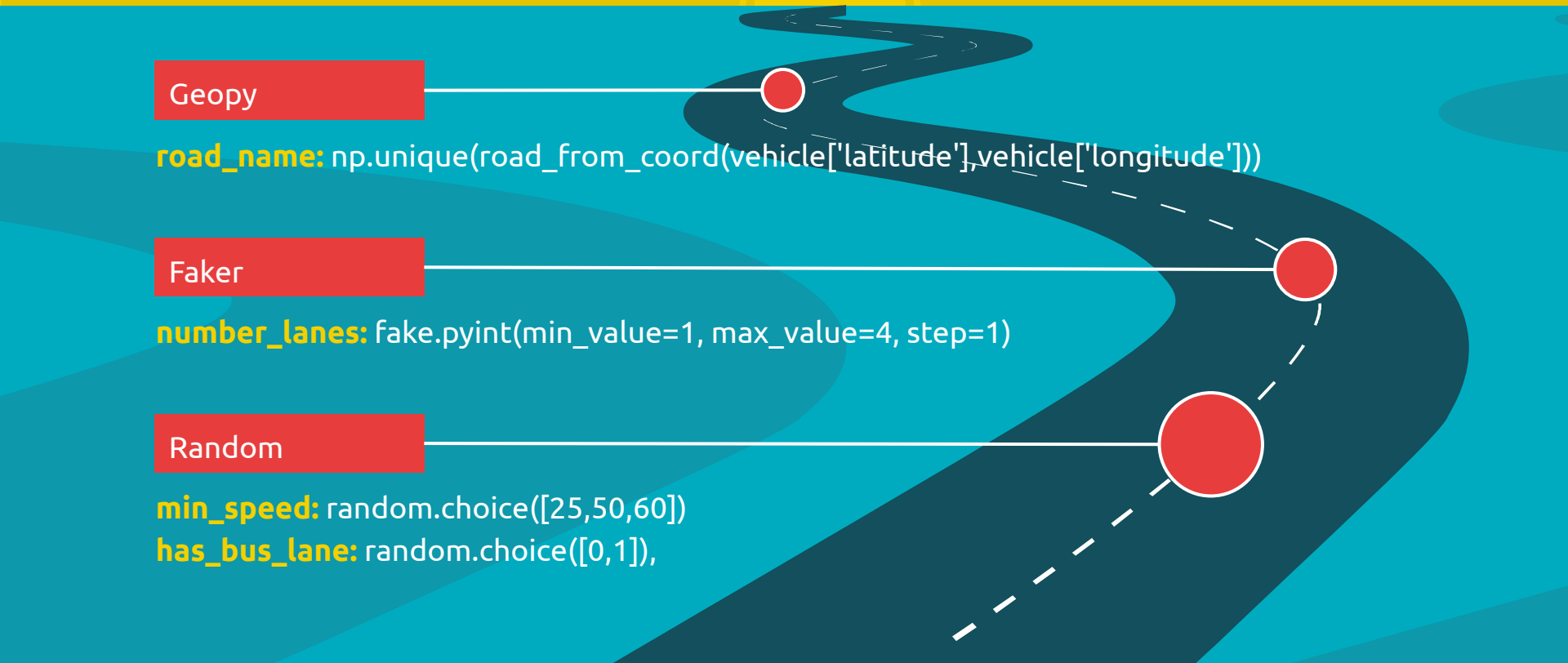
vehicle_type: random.choice(["truck","taxi","bus","private_vehicle"])
vehicle_id - letters: random.choice(CONSONANTS)

- CONSONANTS = list(set(string.ascii_uppercase) - set(list("AEIOU")))

latitude: random.uniform(40.227240, 40.644740)
longitude: random.uniform(-3.944317, -3.426800)

Roads Set

1979



Geopy

road_name: np.unique(road_from_coord(vehicle['latitude'],vehicle['longitude']))

Faker

number_lanes: fake.pyint(min_value=1, max_value=4, step=1)

Random

min_speed: random.choice([25,50,60])
has_bus_lane: random.choice([0,1]),

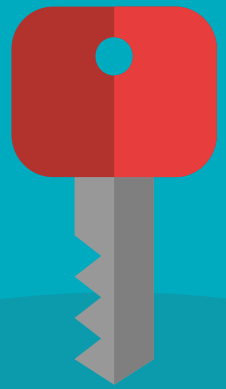
Result

	road_name	number_vehicles
0	Camino del Espinar	3
1	Carretera Particular de la Zarzuela	2
2	Avenida de la Pesadilla	1
3	Calle Enrique Casas	1
4	Calle de Valdemorillo	1
...	...	...
84	enlace con M-40	1
85	Calle XX	1
86	Camino del Esparragal	1
87	Calle de los Morales	1
88	Calle de las Acacias	1

89 rows × 2 columns

04

Environment and Infrastructure



Resources

Microsoft Azure

Platform to host the application

Real-time vehicle information

World-road database

Production Pipeline

New Vehicles

Ingestion of new stream of data

Algorithm

Execute Analysis Query Stream Analytics Job

Alert

Traffic Jam!

Notification

Send results to consumer server

SOURCE

Budget

Azure Stream Analytics

$0.11/ hour

DAILY	WEEKLY	MONTHLY	ANNUALLY
$2.64	$18.48	$81.84	$982.08

Security

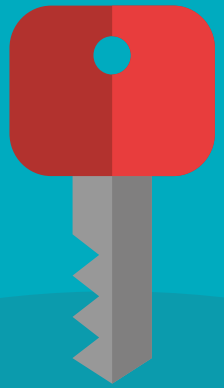
Consensus in society

Secure communication

Information encoding

05

Future Work



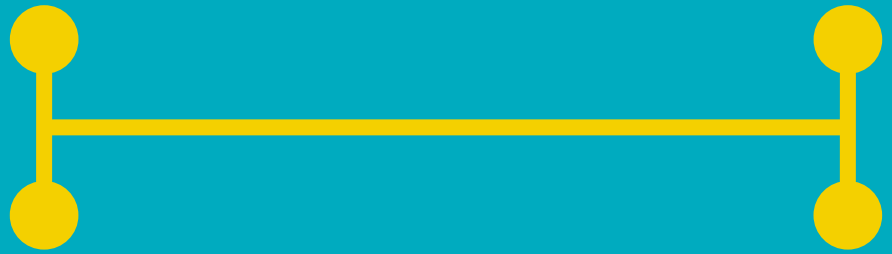
Improvements

Roads

All cities
Divided by km ranges

Environment

Keep track
of vehicles nearby



Direction

Direction of the vehicle
(cardinal points)

Streaming

From static
to streaming solution

THANKS!

Any questions?

cbarbagallo.ieu2017@student.ie.edu
pgarcia.ieu2018@student.ie.edu
jgil.ieu2018@student.ie.edu
rgonzalez.ieu2018@student.ie.edu
ovall.ieu2018@student.ie.edu
vzaldivar.ieu2018@student.ie.edu

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**

Please keep this slide for attribution.