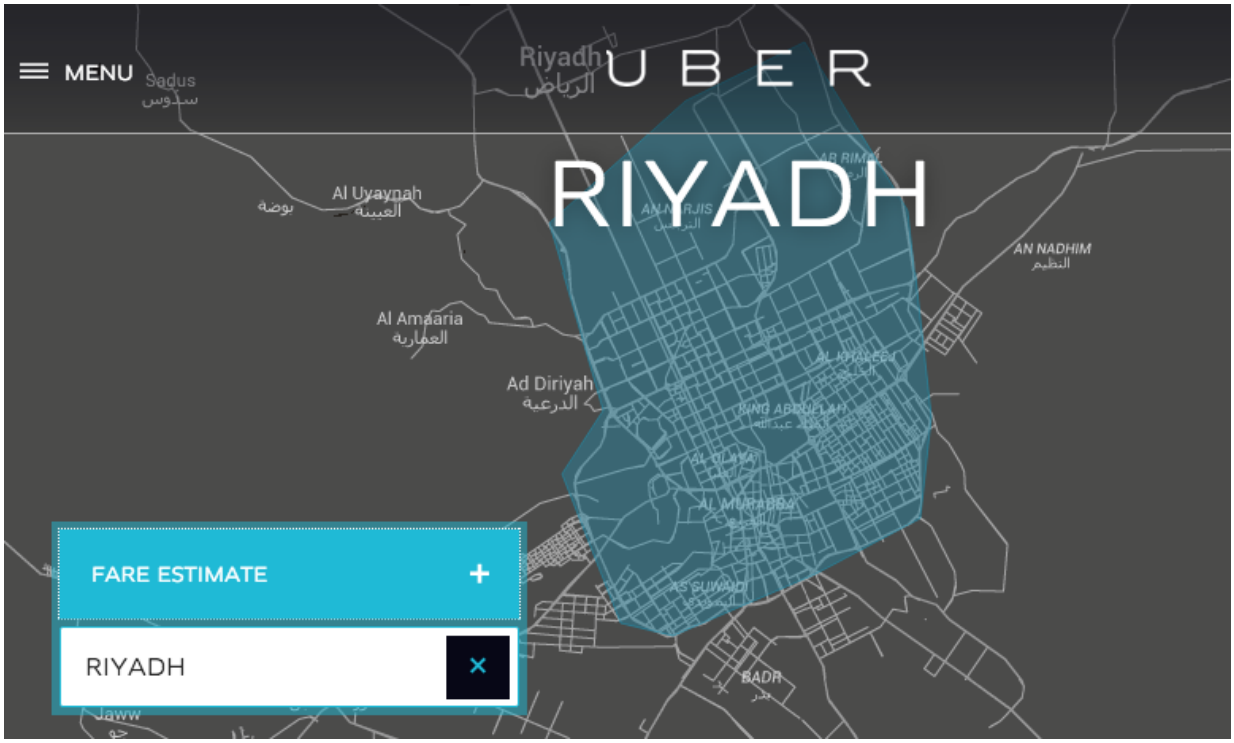
Uber - 2020 strategy for Riyadh Market (Part 1): SQL Query

In [10]:
```
from IPython.core.display import HTML
HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
The raw code for this IPython notebook is by default hidden for easier reading
To toggle on/off the raw code, click <a href="javascript:code_toggle()">here</
```

Out[10]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

In [11]:
```
from IPython.display import Image
Image(filename='output1.png',width=1000, height=400)
```

Out[11]:



Introduction

Uber Technologies Inc. is investing $250 million to expand in the Middle East and North Africa, which have some of the ride-sharing service's fastest-growing markets, Bloomberg reports.

Uber is already in Saudi Arabia, and the ride-sharing app is having a significant impact on the transportation economy there.

In [12]:
```
import pandas as pd
import sqlite3 as sql

df_riyadh=pd.read_csv('riyadh_sample.csv')
```

In [13]:
```
d_tables = {
    'trips': [
      'city_id',
      'completed_trip',
      'distance_to_pickup',
      'driver_id',
      'dropoff_geo',
      'dropoff_local_time',
      'dropoff_utc_time',
      'entered_destination',
      'esttime_to_pickup',
      'pickup_geo',
      'pickup_local_time',
      'pickup_utc_time',
      'request_geo',
      'request_local_time',
      'request_type',
      'request_utc_time',
      'rider_id',
      'surged_trip',
      'time_to_pickup',
      'trip_id',
      'trip_status',
      'vehicle_id'
    ],
    'cities': [
      'city_id',
      'city_name',
      'country_id',
      'country_name',
      'distance_unit',
      'lat',
      'lng',
      'local_currency'
    ],
    'riders': [
      'active_city_id',
      'first_trip_id',
      'preferred_language',
      'rider_app',
      'rider_device',
      'rider_email',
      'rider_trip_count',
      'signup_date'
    ],
    'drivers': [
      'active_city_id',
      'driver_app',
      'driver_device',
      'driver_email',
      'driver_id',
      'driver_trip_count',
      'first_trip_id',
      'preferred_language',
      'signup_date'
    ],
```

```
    'bills': [
      'bill_id',
      'cancel_fee_local',
      'cancel_fee_usd',
      'completed_trip',
      'driver_id',
      'entered_destination',
      'exchange_rate',
      'local_currency',
      'paid_cash',
      'partner_id',
      'payment_type',
      'product_category',
      'request_type',
      'rider_id',
      'surged_trip',
      'trip_distance_miles',
      'trip_duration_seconds',
      'trip_fare_local',
      'trip_fare_usd',
      'trip_id'
    ],
    'vehicles': [
      'seat_count',
      'vehicle_color',
      'vehicle_id',
      'vehicle_trip_count',
      'vehicle_type'
    ],

}
```

Displays the tables and columns of each table of the dataset: trips, cities, riders, drivers, bills, vehicle

In [14]:
```
d_tables
```

Out[14]:
```
{'trips': ['city_id',
  'completed_trip',
  'distance_to_pickup',
  'driver_id',
  'dropoff_geo',
  'dropoff_local_time',
  'dropoff_utc_time',
  'entered_destination',
  'esttime_to_pickup',
  'pickup_geo',
  'pickup_local_time',
  'pickup_utc_time',
  'request_geo',
  'request_local_time',
  'request_type',
  'request_utc_time',
  'rider_id',
  'surged_trip',
  'time_to_pickup',
  'trip_id',
  'trip_status',
  'vehicle_id'],
 'cities': ['city_id',
  'city_name',
  'country_id',
  'country_name',
  'distance_unit',
  'lat',
  'lng',
  'local_currency'],
 'riders': ['active_city_id',
  'first_trip_id',
  'preferred_language',
  'rider_app',
  'rider_device',
  'rider_email',
  'rider_trip_count',
  'signup_date'],
 'drivers': ['active_city_id',
  'driver_app',
  'driver_device',
  'driver_email',
  'driver_id',
  'driver_trip_count',
  'first_trip_id',
  'preferred_language',
  'signup_date'],
 'bills': ['bill_id',
  'cancel_fee_local',
  'cancel_fee_usd',
  'completed_trip',
  'driver_id',
  'entered_destination',
  'exchange_rate',
```

```
 'local_currency',
 'paid_cash',
 'partner_id',
 'payment_type',
 'product_category',
 'request_type',
 'rider_id',
 'surged_trip',
 'trip_distance_miles',
 'trip_duration_seconds',
 'trip_fare_local',
 'trip_fare_usd',
 'trip_id'],
'vehicles': ['seat_count',
 'vehicle_color',
 'vehicle_id',
 'vehicle_trip_count',
 'vehicle_type']}
```

In [15]:
```
for each_element in list(df_riyadh):
  flag = 0
  for each_table in list(d_tables):
    for each_column in d_tables[each_table]:
      if each_element == each_column:
        print(each_table+'.'+each_column+',')
        flag = 1

  if flag == 0:
    print('     '+each_element+',')
```

```
trips.pickup_local_time,
trips.pickup_utc_time,
bills.cancel_fee_local,
bills.cancel_fee_usd,
trips.city_id,
cities.city_id,
riders.rider_app,
riders.rider_device,
riders.rider_trip_count,
trips.rider_id,
bills.rider_id,
     partner_vehicle_count,
drivers.driver_trip_count,
trips.driver_id,
drivers.driver_id,
bills.driver_id,
trips.dropoff_local_time,
trips.dropoff_utc_time,
trips.esttime_to_pickup,
trips.request_type,
bills.request_type,
trips.entered_destination,
bills.entered_destination,
bills.paid_cash,
trips.completed_trip,
bills.completed_trip,
trips.surged_trip,
bills.surged_trip,
bills.trip_fare_local,
bills.trip_fare_usd,
bills.partner_id,
trips.request_local_time,
trips.request_utc_time,
trips.distance_to_pickup,
trips.time_to_pickup,
trips.trip_status,
bills.trip_distance_miles,
bills.trip_duration_seconds,
trips.trip_id,
bills.trip_id,
vehicles.vehicle_trip_count,
trips.vehicle_id,
vehicles.vehicle_id,
vehicles.vehicle_type,
```

```
trips.pickup_geo,
trips.dropoff_geo,
```

Part 1: SQL query

In [16]:

```
query = '''
select
    trips.pickup_local_time,
    trips.pickup_utc_time,
    bills.cancel_fee_local,
    bills.cancel_fee_usd,
    trips.city_id,
    riders.rider_app,
    riders.rider_device,
    riders.rider_trip_count,
    trips.rider_id,
    -- Assuming that the SQL dialect is Snowflake
    array_size(drivers.vehicle_ids) as partner_vehicle_count,
    drivers.driver_trip_count,
    trips.driver_id,
    trips.dropoff_local_time,
    trips.dropoff_utc_time,
    trips.esttime_to_pickup,
    trips.request_type,
    trips.entered_destination,
    bills.paid_cash,
    trips.completed_trip,
    trips.surged_trip,
    bills.trip_fare_local,
    bills.trip_fare_usd,
    bills.partner_id,
    trips.request_local_time,
    trips.request_utc_time,
    trips.distance_to_pickup,
    trips.time_to_pickup,
    trips.trip_status,
    bills.trip_distance_miles,
    bills.trip_duration_seconds,
    trips.trip_id,
    vehicles.vehicle_trip_count,
    trips.vehicle_id,
    vehicles.vehicle_id,
    vehicles.vehicle_type,
    trips.pickup_geo,
    trips.dropoff_geo
from bills
left join trips on bills.trip_id = trips.trip_id
left join riders on bills.rider_id = riders.rider_id
left join drivers on bills.driver_id = drivers.driver_id
left join vehicles on bills.vehicle_id = vehicles.vehicle_id
where
  trips.city_id = 1
  and request_utc_time between '2018-05-06 21:00:00' and '2018-07-01 20:00:00'
'''
```

Click the 'here' below to see the SQL query

In [17]:
```
from IPython.core.display import HTML
HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
The raw code for this IPython notebook is by default hidden for easier reading
To toggle on/off the raw code, click <a href="javascript:code_toggle()">here</
```

Out[17]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

In [18]:
```
query
```

Out[18]:
```
"\nselect\n    trips.pickup_local_time,\n    trips.pickup_utc_time,\n    bills.
cancel_fee_local,\n    bills.cancel_fee_usd,\n    trips.city_id,\n    riders.ri
der_app,\n    riders.rider_device,\n    riders.rider_trip_count,\n    trips.rid
er_id,\n    -- Assuming that the SQL dialect is Snowflake\n    array_size(drive
rs.vehicle_ids) as partner_vehicle_count,\n    drivers.driver_trip_count,\n
trips.driver_id,\n    trips.dropoff_local_time,\n    trips.dropoff_utc_time,\n
trips.esttime_to_pickup,\n    trips.request_type,\n    trips.entered_destinatio
n,\n    bills.paid_cash,\n    trips.completed_trip,\n    trips.surged_trip,\n
bills.trip_fare_local,\n    bills.trip_fare_usd,\n    bills.partner_id,\n    tr
ips.request_local_time,\n    trips.request_utc_time,\n    trips.distance_to_pic
kup,\n    trips.time_to_pickup,\n    trips.trip_status,\n    bills.trip_distanc
e_miles,\n    bills.trip_duration_seconds,\n    trips.trip_id,\n    vehicles.ve
hicle_trip_count,\n    trips.vehicle_id,\n    vehicles.vehicle_id,\n    vehicle
s.vehicle_type,\n    trips.pickup_geo,\n    trips.dropoff_geo\nfrom bills\nleft
join trips on bills.trip_id = trips.trip_id\nleft join riders on bills.rider_id
= riders.rider_id\nleft join drivers on bills.driver_id = drivers.driver_id\nle
ft join vehicles on bills.vehicle_id = vehicles.vehicle_id\nwhere\n  trips.city
_id = 1\n  and request_utc_time between '2018-05-06 21:00:00' and '2018-07-01 2
0:00:00'\n"
```

In [ ]:

In [ ]: