# Taxi Company


Liliya Yerassilova

Design Project

Database Systems CMPT 308

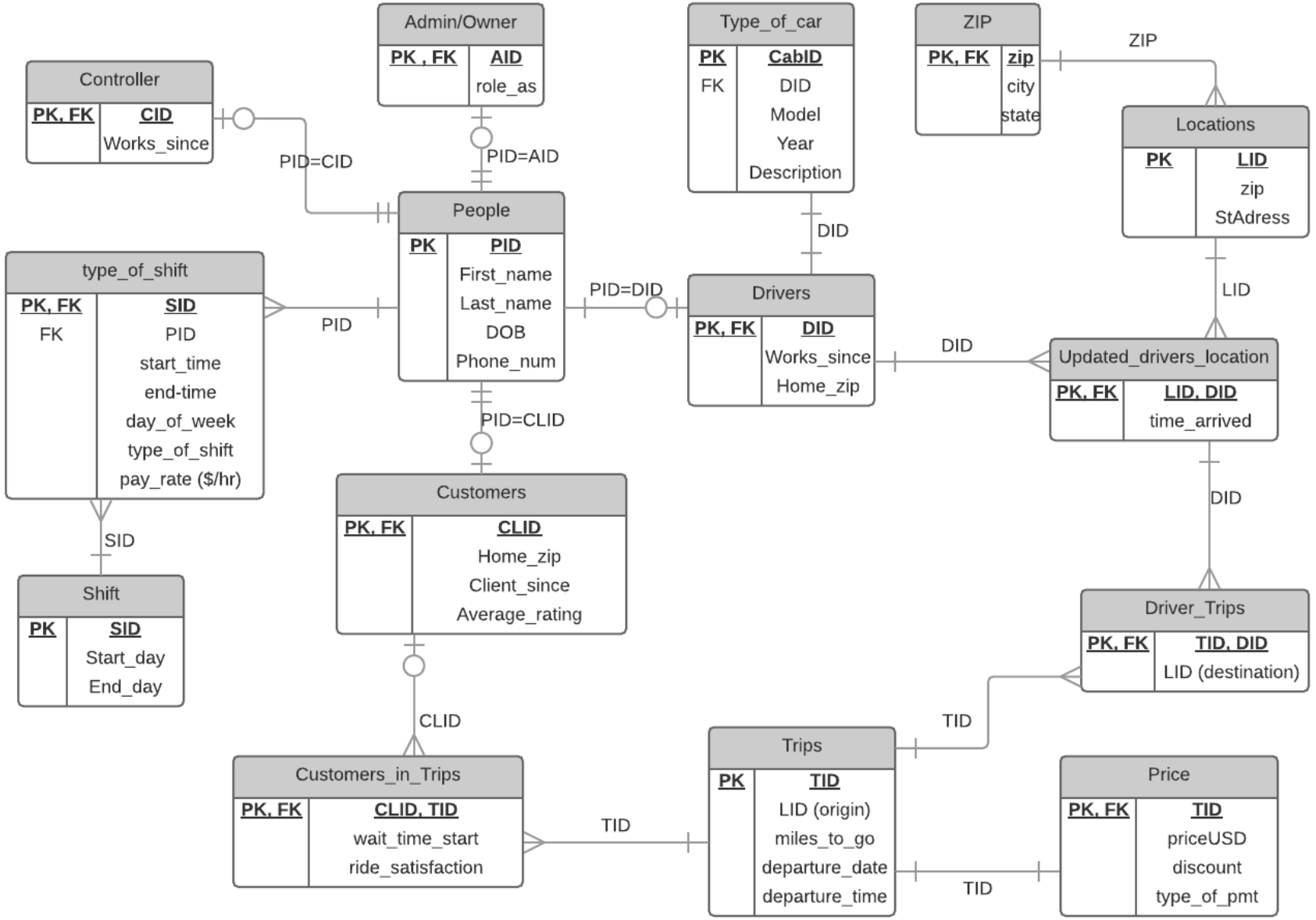
Spring 2016

# Table of Contents:

# EXECUTIVE SUMMARY

This database project is designed for a taxi company. It is not done for one specific taxi company, but rather can be implemented for various companies, since the database is expandable and can be easily changed to satisfy different business rules that might be present in some companies and not in others. In addition, most taxi companies have very similar needs from their databases and they change only a little bit due to the unique business rules, but not operations themselves. For the purposes of this project, I assume that there are controllers, people who accept orders via phone from customers, and there is also some sort of a website or an app where customers can view and order specific cabs themselves. I also assume that drivers come with their own cars, and there is a table that specifies what kind of cars are in place. Customers can make payments with either cash, or credit/debit cards. They also are encouraged to rate their trips out of five. For shifts of employees, such as administrators, controllers, and drivers, I assume that there are four types of shifts: morning that runs from 5 am till 12 pm, day shift from 12 pm till 6 pm, evening 6 pm till 11 pm, and night shift from 11 pm till 5 am. These rules, of course can be changed and they are unlikely to affect the database. The main goal of the project is to create a database that is in the Boyce-Codd normal form and that can meet the basic, essential needs of a general taxi company. The database allows administration as well as controllers to create useful queries and analyze information effectively.

# Entity Relationship Diagram



# Table People:

```
-- People --
CREATE TABLE people (
  pid                 char(5) not null unique,
  first_name       text,
  last_name        text,
  DOB                 date,
  phone_num     char(15) not null,
  primary key(pid)
);
```

| | pid character(5) | first_name text | last_name text | dob date | phone_num character(15) |
|---|---|---|---|---|---|
| 1 | 1 | Edgar | Codd | 1989-08-25 | 9145488888 |
| 2 | 2 | Maria | Sharapova | 1970-03-29 | 9145559999 |
| 3 | 3 | Sean | Connery | 1930-08-25 | 9144596365 |
| 4 | 4 | Larry | Ellison | 1991-09-05 | 9140001111 |
| 5 | 5 | Joanne | Rowling | 1985-01-29 | 9145269999 |
| 6 | 6 | Nikola | Tesla | 1953-08-16 | 9145452222 |
| 7 | 7 | Alan | Labouseur | 1960-11-11 | 9145557777 |
| 8 | 8 | Harry | Potter | 1975-02-02 | 9147777777 |
| 9 | 9 | Spider | Man | 1950-06-09 | 9146626666 |
| 10 | 10 | Franz | Schubert | 1969-07-25 | 9143331111 |
| 11 | 11 | Alexander | Pushkin | 1987-10-08 | 9143216544 |
| 12 | 12 | James | Gosling | 1974-01-05 | 9149511236 |
| 13 | 13 | Richard | Branson | 1962-03-03 | 9147419512 |
| 14 | 14 | Willis | Carrier | 1966-12-12 | 9147899631 |

FUNCTIONAL DEPENDENCIES: pid ⟶ first_name, last_name, dob, phone_num

Main table for people overall, everybody who has business with the company: employees, customers, drivers, and owners.

# Table Admin/Owner:

```
-- Admin/Owner --
CREATE TABLE admin_owner (
  aid              char(5) not null references people(pid),
  role_as       char(5) not null,
  CONSTRAINT valid_role_as
  CHECK (role_as = 'admin' OR role_as = 'owner' OR role_as = 'both'),
  primary key(aid)
);
```

| | aid character(5) | role_as character(5) |
|---|---|---|
| 1 | 1 | owner |
| 2 | 2 | admin |
| 3 | 3 | both |

FUNCTIONAL DEPENDENCIES: aid ⟶ role_as

This is a sub-entity of People table, its aid (admin id) is the same as People's pid for the same person.

# Table Controller:

```
-- Controller --
CREATE TABLE controller (
  cid                   char(5) not null references people(pid),
  works_since       DATE,
 primary key(cid)
);
```

| | cid character(5) | works_since date |
|---|---|---|
| 1 | 4 | 2011-01-03 |
| 2 | 5 | 2012-01-03 |
| 3 | 6 | 2013-01-06 |

FUNCTIONAL DEPENDENCIES: cid ⟶ works_since

This is a sub-entity of People table, its cid (controller id) is the same as People's pid for the same person.

# Table Customers:

| | clid character(5) | home_zip integer | client_since date | average_rating double precision |
|---|---|---|---|---|
| 1 | 7 | 12601 | 2015-06-07 | 5 |
| 2 | 8 | 12603 | 2012-08-08 | 4.4 |
| 3 | 9 | 12602 | 2016-09-01 | 4.8 |
| 4 | 10 | 12605 | 2014-02-02 | 4.1 |

```
-- Customers --
CREATE TABLE Customers (
  clid                  char(5) not null references people(pid),
  home_zip              int not null,
  client_since          DATE not null,
  average_rating        float,
 primary key(clid)
);
```

FUNCTIONAL DEPENDENCIES: clid ⟶ home_zip, client_since, average_rating

This is a sub-entity of People table, its clid (client id) is the same as People's pid for the same person.

# Table Drivers:

```
-- Drivers--
CREATE TABLE drivers (
  did                 char(5) not null references people(pid),
  works_since     DATE not null,
  Home_zip         int not null,
 primary key(did)
);
```

| | did character(5) | works_since date | home_zip integer |
|---|---|---|---|
| 1 | 11 | 2015-06-07 | 12601 |
| 2 | 12 | 2009-06-08 | 12602 |
| 3 | 13 | 2010-07-09 | 12603 |
| 4 | 14 | 2009-06-05 | 12601 |

FUNCTIONAL DEPENDENCIES: did $\longrightarrow$ works_since, home_zip

This is a sub-entity of People table, its did (driver id) is the same as People's pid for the same person.

# Table Type_of_Car:

| | cabid character(5) | did character(5) | model text | year date | description text |
|---|---|---|---|---|---|
| 1 | A1 | 11 | Toyota | 2005-01-01 | Fits up to 6 passengers, can fit a few medium suitcases |
| 2 | A2 | 12 | Lexus | 2009-01-01 | Fits up to 6 adults, plus up to 4 large suitcases |
| 3 | A3 | 13 | Ford | 2013-01-01 | Fits 5 adults, a few medium suitcase |
| 4 | A4 | 14 | Nissan | 2014-01-01 | Fits 5 adults, one medium or a few small bags |

```
-- Type_of_Car--
CREATE TABLE type_of_car (
  cabid              char(5) not null unique,
  did                char(5) not null references drivers(did),
  model              text not null,
  year               DATE not null,
  description        text,
 primary key(cabid)
);
```

FUNCTIONAL DEPENDENCIES: cabid ⟶ did, model, year, description

Accumulates and accounts for the cars that are in place overall and also provides information of the id of the owner for each car.

# Table Locations:

```
-- Locations--
CREATE TABLE locations (
  lid                 char(5) not null unique,
  zip                 char(5) not null,
  st_address      text not null,
 primary key(lid)
);
```

| | lid character(5) | zip character(5) | st_address text |
|---|---|---|---|
| 1 | L01 | 12601 | 3399 North road |
| 2 | L02 | 12602 | 33 Pine Bush street |
| 3 | L03 | 12603 | 55 Ave Maria Avenue |
| 4 | L04 | 12604 | 11 Lucky street |
| 5 | L05 | 12605 | 12 Green Square |

FUNCTIONAL DEPENDENCIES: lid ⟶ zip, st_address

Accounts for locations in which the company operates.

# Table ZIP:

```
-- ZIP--
CREATE TABLE zip (
  zip             char(5) not null unique,
  city            text not null,
  state           char(2) not null,
 primary key(zip)
);
```

| | zip character(5) | city text | state character(2) |
|---|---|---|---|
| 1 | 12601 | Poughkeepsie | NY |
| 2 | 12602 | Poughkeepsie | NY |
| 3 | 12603 | Poughkeepsie | NY |
| 4 | 12604 | Poughkeepsie | NY |
| 5 | 12605 | Poughkeepsie | NY |

FUNCTIONAL DEPENDENCIES: zip ⟶ city, state

ZIP table is separate in order to achieve Boyce – Codd Normal Form.

# Table Updated_Drivers_Location:

```
-- Updated_drivers_location--
CREATE TABLE updated_drivers_location (
  lid                   char(5) not null references locations(lid),
  did                   char(5) not null references people(pid),
  time_arrived      TIME with time zone not null,
 primary key(lid, did)
);
```

| | lid character(5) | did character(5) | time_arrived time with time zone |
|---|---|---|---|
| 1 | L01 | 11 | 04:53:00-04 |
| 2 | L02 | 12 | 04:45:00-04 |
| 3 | L03 | 13 | 03:58:00-04 |
| 4 | L04 | 14 | 04:35:00-04 |

FUNCTIONAL DEPENDENCIES: (lid, did) ⟶ time_arrived

This table tracks current location of each driver.

# Table Trips:

```
-- Trips--
CREATE TABLE trips (
  tid                  char(5) not null unique,
  origin_LID           char(5) not null references locations(lid),
  miles_to_go          float not null CHECK (miles_to_go > 0.0),
  departure_date       DATE not null,
  departure_time       TIME with time zone not null,
 primary key(tid)
);
```

| | tid character(5) | origin_lid character(5) | miles_to_go double precision | departure_date date | departure_time time with time zone |
|---|---|---|---|---|---|
| 1 | T001 | L02 | 15 | 2016-04-19 | 04:42:00-04 |
| 2 | T002 | L04 | 20 | 2016-04-19 | 04:00:00-04 |
| 3 | T003 | L05 | 10 | 2016-04-19 | 03:43:00-04 |
| 4 | T004 | L01 | 22 | 2016-04-19 | 03:55:00-04 |

FUNCTIONAL DEPENDENCIES: tid ⟶ origin_lid, miles_to_go, departure_date, departure_time

The main table to account for each trip by identifying them with a unique trip ID (tid).

# Table Driver_Trips:

```
-- Driver_trips--
CREATE TABLE driver_trips (
  tid                  char(5) not null references trips(tid),
  did                  char(5) not null references drivers(did),
  destination_LID   char(5) not null references locations(lid),
 primary key(tid, did)
);
```

| | tid character(5) | did character(5) | destination_lid character(5) |
|---|---|---|---|
| 1 | T001 | 11 | L01 |
| 2 | T002 | 12 | L02 |
| 3 | T003 | 13 | L03 |
| 4 | T004 | 14 | L04 |

FUNCTIONAL DEPENDENCIES: (tid, did) ⟶ destination_lid

This is an associative entity to track drivers' trips and connect that to drivers' current location.

# Table Customers_in_Trips:

```
-- Customers_in_trips--
CREATE TABLE customers_in_trips (
  clid                    char(5) not null references customers(clid),
  tid                     char(5) not null references trips(tid),
  wait_time_start     TIME not null,
  ride_satisfaction float,
 primary key(clid, tid)
);
```

| | clid character(5) | tid character(5) | wait_time_start time without time zone | ride_satisfaction double precision |
|---|---|---|---|---|
| 1 | 7 | T001 | 04:40:00 | 4.9 |
| 2 | 8 | T002 | 03:55:00 | 4.7 |
| 3 | 9 | T003 | 03:40:00 | 5 |
| 4 | 10 | T004 | 03:45:00 | 4.5 |

FUNCTIONAL DEPENDENCIES: (clid, tid) ⟶ wait_time_start, ride_satisfaction

Also an associative entity, tracks customers' trips and connects it to customers table.

# Table Shift :

```
-- Shift--
CREATE TABLE shift (
  sid               char(5) not null unique,
  start_day      DATE not null,
  end_day        DATE not null,
 primary key(sid)
);
```

| | sid character(5) | start_day date | end_day date |
|---|---|---|---|
| 1 | S1 | 2016-04-19 | 2016-04-19 |
| 2 | S2 | 2016-04-19 | 2016-04-19 |
| 3 | S3 | 2016-04-19 | 2016-04-19 |
| 4 | S4 | 2016-04-19 | 2016-04-19 |
| 5 | S5 | 2016-04-19 | 2016-04-19 |
| 6 | S6 | 2016-04-19 | 2016-04-19 |

FUNCTIONAL DEPENDENCIES: sid ⟶ start_day, end_day

The main table to account for shifts.

# Table Type_of_Shift:

```
-- Type_of_Shift--
CREATE TABLE type_of_shift (
  sid                    char(5) not null references shift(sid),
  pid                    char(5) not null references people(pid),
  start_time             TIME not null,
  end_time               TIME not null,
  day_of_week            char(2) not null,
  type_of_shift          char(1) not null,
  pay_rate_per_hr        MONEY not null default 20,
  CONSTRAINT valid_shift
  CHECK(type_of_shift = 'd' OR type_of_shift = 'n' OR type_of_shift = 'e' OR type_of_shift = 'm'),
 primary key(sid)
);
```

| | sid character(5) | pid character(5) | start_time time without time zone | end_time time without time zone | day_of_week character(2) | type_of_shift character(1) | pay_rate_per_hr money |
|---|---|---|---|---|---|---|---|
| 1 | S1 | 11 | 12:00:00 | 06:00:00 | TU | d | $22.00 |
| 2 | S2 | 12 | 12:00:00 | 06:00:00 | TU | d | $22.00 |
| 3 | S3 | 13 | 12:00:00 | 06:00:00 | TU | d | $22.00 |
| 4 | S4 | 14 | 12:00:00 | 06:00:00 | TU | d | $22.00 |
| 5 | S5 | 5 | 12:00:00 | 06:00:00 | TU | d | $21.00 |
| 6 | S6 | 5 | 06:00:00 | 11:00:00 | TU | e | $26.00 |

FUNCTIONAL DEPENDENCIES: sid ⟶ pid, start_time, end_time, day_of_week, type_of_shift, pay_rate_per_hr

A sub-entity table that is a part of shift table.

# Table Price:

```
-- Price--
CREATE TABLE price (
  tid                    char(5) not null references trips(tid),
  priceUSD               MONEY not null CHECK (priceUSD > 0.0::text::money),
  discount_percent       float default 0,
  type_of_pmt            char(7) not null,
  CONSTRAINT valid_type_of_pmt
  CHECK(type_of_pmt = 'cash' OR type_of_pmt = 'credit' OR type_of_pmt = 'debit'),
 primary key(tid)
);
```

| | tid character(5) | priceusd money | discount_percent double precision | type_of_pmt character(7) |
|---|---|---|---|---|
| 1 | T001 | $35.00 | | cash |
| 2 | T002 | $45.00 | 1.5 | credit |
| 3 | T003 | $20.00 | | cash |
| 4 | T004 | $50.00 | | debit |

FUNCTIONAL DEPENDENCIES: tid → priceUSD, discount_percent, type_of_pmt

The main table to account for prices, money paid by customers for each trip.

# View Summary:

```
CREATE VIEW summary AS
SELECT t.tid, dt.did, clid, origin_LID, destination_LID, wait_time_start, departure_time, time_arrived
FROM updated_drivers_location udl,
        trips t,
        driver_trips dt,
        customers_in_trips ct
WHERE t.tid = dt.tid
  AND   t.tid = ct.tid
  AND   udl.did = dt.did;
```

This summary can be useful for controllers to track drivers' current location, their trips, and how long it took them to get to their destination

| | tid character(5) | did character(5) | clid character(5) | origin_lid character(5) | destination_lid character(5) | wait_time_start time without time zone | departure_time time with time zone | time_arrived time with time zone |
|---|---|---|---|---|---|---|---|---|
| 1 | T001 | 11 | 7 | L02 | L01 | 04:40:00 | 04:42:00-04 | 04:53:00-04 |
| 2 | T002 | 12 | 8 | L04 | L02 | 03:55:00 | 04:00:00-04 | 04:45:00-04 |
| 3 | T003 | 13 | 9 | L05 | L03 | 03:40:00 | 03:43:00-04 | 03:58:00-04 |
| 4 | T004 | 14 | 10 | L01 | L04 | 03:45:00 | 03:55:00-04 | 04:35:00-04 |

# View forCustomers:

```
CREATE VIEW forCustomers AS
SELECT first_name, last_name, st_address, city, state, z.zip
FROM people p,
        drivers d,
        locations l,
        zip z,
        updated_drivers_location udl
WHERE p.pid = d.did
  AND l.zip = z.zip
  AND l.lid = udl.lid
  AND udl.did = d.did;
```

This view is good for customers to see what drivers are around, their current locations destination

| | first_name text | last_name text | st_address text | city text | state character(2) | zip character(5) |
|---|---|---|---|---|---|---|
| 1 | Alexander | Pushkin | 3399 North road | Poughkeepsie | NY | 12601 |
| 2 | James | Gosling | 33 Pine Bush street | Poughkeepsie | NY | 12602 |
| 3 | Richard | Branson | 55 Ave Maria Avenue | Poughkeepsie | NY | 12603 |
| 4 | Willis | Carrier | 11 Lucky street | Poughkeepsie | NY | 12604 |

# View CarLocation :

```
CREATE VIEW CarLocation AS
SELECT cabid, model, description, l.lid, st_address, l.zip
FROM type_of_car tc,
        updated_drivers_location udl,
        drivers d,
        locations l,
        zip z
WHERE tc.did = d.did
  AND   udl.did = d.did
  AND   udl.lid = l.lid
  AND   l.zip = z.zip;
```

This view is good for both controllers and customers to track current locations of specific types of cars. If a customer is leaving to the airport and has a few large bags, it would be more sensible to order a car not based on who is closer, but what car is larger.

| | cabid character(5) | model text | description text | lid character(5) | st_address text | zip character(5) |
|---|---|---|---|---|---|---|
| 1 | A1 | Toyota | Fits up to 6 passengers, can fit a few medium suitcases | L01 | 3399 North road | 12601 |
| 2 | A2 | Lexus | Fits up to 6 adults, plus up to 4 large suitcases | L02 | 33 Pine Bush street | 12602 |
| 3 | A3 | Ford | Fits 5 adults, a few medium suitcase | L03 | 55 Ave Maria Avenue | 12603 |
| 4 | A4 | Nissan | Fits 5 adults, one medium or a few small bags | L04 | 11 Lucky street | 12604 |

# Query to compare satisfaction of trip to waiting time

```
SELECT ct.clid, first_name, last_name, ride_satisfaction,
ABS (SUM((((((select cast(extract(hour from s.departure_time) as integer)) - (select cast(extract(hour from s.wait_time_start) as integer)))*60))))
- ABS(SUM(((select cast(extract(minute from s.departure_time) as integer))) - ((select cast(extract(minute from s.wait_time_start) as integer)))))) as Minutes_Waiting

FROM summary s,
        customers_in_trips ct,
        people p
WHERE s.clid = ct.clid
  AND   ct.clid = p.pid
GROUP BY ct.clid, first_name, last_name, ride_satisfaction
ORDER BY ct.clid DESC;
```

A useful query to see how waiting time can affect rating of satisfaction of customers after every trip. Good for quality control purposes.

| | clid character(5) | first_name text | last_name text | ride_satisfaction double precision | minutes_waiting integer |
|---|---|---|---|---|---|
| 1 | 7 | Alan | Labouseur | 5 | 2 |
| 2 | 8 | Harry | Potter | 4.4 | 5 |
| 3 | 9 | Spider | Man | 4.8 | 3 |
| 4 | 10 | Franz | Schubert | 4.1 | 10 |

# Query to see who works how much and their wages

```
SELECT DISTINCT p.pid, first_name, last_name, ts.pay_rate_per_hr,
ABS (SUM((((((select cast(extract(hour from ts.end_time) as integer)) - (select cast(extract(hour from ts.start_time) as integer)))))))
- ABS(SUM(((select cast(extract(minute from ts.end_time) as integer))) - ((select cast(extract(minute from ts.start_time) as integer)))))/60)) as Hours_Worked,


ABS(SUM((((((select cast(extract(hour from ts.end_time) as integer)) - (select cast(extract(hour from ts.start_time) as integer)))))
-((select cast(extract(minute from ts.end_time) as integer)) - (select cast(extract(minute from ts.start_time) as integer)))) * (ts.pay_rate_per_hr ::money::numeric::float8)))as TotalUSD

FROM people p,
        shift s,
        type_of_shift ts
WHERE p.pid = ts.pid
  AND     s.sid = ts.sid
GROUP BY p.pid, ts.pay_rate_per_hr, ts.end_time, ts.start_time;
```

Continued on next page…

Query to see who works how much and their wages …continued…

| | pid character(5) | first_name text | last_name text | pay_rate_per_hr money | hours_worked bigint | totalusd double precision |
|---|---|---|---|---|---|---|
| 1 | 11 | Alexander | Pushkin | $22.00 | 6 | 132 |
| 2 | 12 | James | Gosling | $22.00 | 6 | 132 |
| 3 | 13 | Richard | Branson | $22.00 | 6 | 132 |
| 4 | 14 | Willis | Carrier | $22.00 | 6 | 132 |
| 5 | 5 | Joanne | Rowling | $21.00 | 6 | 126 |
| 6 | 5 | Joanne | Rowling | $26.00 | 5 | 130 |

Also useful for administration and accounting purposes.

# Query to see how much customers paid to what drivers

```
SELECT pr.tid, p2.first_name as CustomerName1, p2.last_name as CustomerName, priceUSD,
        p1.first_name as DriverName1,  p1.last_name as DriverName
FROM price pr,
      driver_trips dt,
      customers_in_trips ct,
      people p1,
      people p2
WHERE pr.tid = dt.tid
   AND pr.tid = ct.tid
   AND dt.did = p1.pid
   AND ct.clid = p2.pid;
```

| | tid character(5) | customername1 text | customername text | priceusd money | drivername1 text | drivername text |
|---|---|---|---|---|---|---|
| 1 | T001 | Alan | Labouseur | $35.00 | Alexander | Pushkin |
| 2 | T002 | Harry | Potter | $45.00 | James | Gosling |
| 3 | T003 | Spider | Man | $20.00 | Richard | Branson |
| 4 | T004 | Franz | Schubert | $50.00 | Willis | Carrier |

This is good for internal control purposes, especially if a lot of customers pay in cash. It is important to know how much drivers collect/earn to better determine their wages and bonuses, if any.

# Stored Procedure to see hours worked by employees:

```
create or replace function get_hours_by_pid(char(5), REFCURSOR) returns refcursor as
$$
declare
  input_pid   char(5)   := $1;
  resultset   REFCURSOR := $2;
begin
  open resultset for
    select input_pid, first_name, last_name, start_time, end_time, day_of_week, type_of_shift, pay_rate_per_hr,
           ABS (SUM((((((select cast(extract(hour from ts.end_time) as integer)) - (select cast(extract(hour from
           ts.start_time) as integer)))))))
           - ABS(SUM(((select cast(extract(minute from ts.end_time) as integer))) - ((select cast(extract(minute from ts.start_time)
           as integer)))))/60)) as HoursWorked

    from   type_of_shift ts,
           people p
    where  input_pid = ts.pid
       and  p.pid = ts.pid
   group by ts.pid, p.first_name, p.last_name, ts.start_time, ts.end_time, ts.day_of_week, ts.type_of_shift, ts.pay_rate_per_hr;
  return resultset;
end;
```

Continued on next page…

Continued…..

```
$$
language plpgsql;

select get_hours_by_pid('11', 'results');
Fetch all from results;
```

| | input_pid character(5) | first_name text | last_name text | start_time time without time zone | end_time time without time zone | day_of_week character(2) | type_of_shift character(1) | pay_rate_per_hr money | hoursworked bigint |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 11 | Alexander | Pushkin | 12:00:00 | 06:00:00 | TU | d | $22.00 | 6 |

Shifts are important and tedious. Stored procedure will account for them.

# Stored Procedure to see how long each driver takes for certain mileage trips

```
create or replace function get_time_by_did(char(5), REFCURSOR) returns refcursor as
$$
declare
  input_did   char(5)   := $1;
  resultset   REFCURSOR := $2;
begin
  open resultset for
    select input_did, dt.tid, first_name, last_name, miles_to_go,
           ABS (SUM((((((select cast(extract(hour from udl.time_arrived) as integer)) - (select cast(extract(hour from
           t.departure_time) as integer))))*60)))
           - ABS(SUM(((select cast(extract(minute from udl.time_arrived) as integer))) - ((select cast(extract(minute from
           t.departure_time) as integer)))))) as Length_of_Trip
    from   trips t,
           updated_drivers_location udl,
           people p,
           driver_trips dt

    where  input_did = udl.did
       and  p.pid = udl.did
       and t.tid = dt.tid
       and udl.did = dt.did
```

Continued on next page…

Continued…..

```
group by dt.tid, p.first_name, p.last_name, miles_to_go;
  return resultset;
end;
$$
language plpgsql;

select get_time_by_did('11', 'result');
Fetch all from result;
```

| | input_did character(5) | tid character(5) | first_name text | last_name text | miles_to_go double precision | length_of_trip bigint |
|---|---|---|---|---|---|---|
| 1 | 11 | T001 | Alexander | Pushkin | 15 | 11 |

This stored procedure shows how much time each driver takes to travel certain mileage for each trip. In other words, how fast can a driver bring a customer from one location to another given specific mileage.

# Triggers:

Trigger to prevent inputting more than 8 hours of work per 24 hr-period as well as more than 8 hrs at a time

```
CREATE OR REPLACE FUNCTION hour_limit_trigger() RETURNS trigger as
$$
BEGIN

 IF ABS (SUM((((((select cast(extract(hour from type_of_shift.end_time) as integer)) - (select cast(extract(hour from
            type_of_shift.start_time) as integer)))))))
     - ABS(SUM(((select cast(extract(minute from type_of_shift.end_time) as integer))) - ((select cast(extract(minute from
            type_of_shift.start_time) as integer))))/60)) > 8
  FROM type_of_shift
  THEN
  RAISE EXCEPTION 'Cannot input more than 8 hours!';
  END IF;
  RETURN NEW;
END;
$$ language plpgsql;
```

Continued on next page…

Continued…..

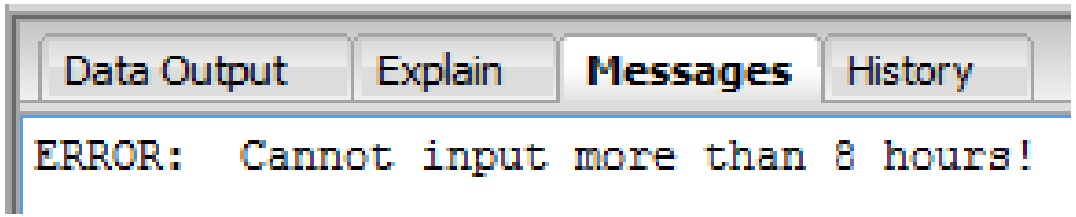
```
CREATE TRIGGER hour_limit
AFTER INSERT
ON type_of_shift
FOR each row
execute procedure hour_limit_trigger();
```

Now, check trigger:

```
INSERT INTO shift (sid, start_day, end_day)
        VALUES ('s7', '04/19/2016','04/19/2016');

INSERT INTO type_of_shift (sid, pid, start_time, end_time, day_of_week, type_of_shift, pay_rate_per_hr)
        VALUES ('s7', '2', '10:00:00', '07:00:00', 'WE', 'n', 26);
```

RESULT:



# SECURITY:

Security controls are important. These can be changed from company to company.

```
--OWNER
CREATE ROLE owner;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO owner;


--Controllers-------------------------------------------------
CREATE ROLE controller;
GRANT SELECT ON shift, type_of_shift, customers, drivers, driver_trips, locations, zip, price, customers_in_trips,
                        type_of_car, updated_drivers_location
TO controller;
--   --  --  --  --  --  --  --  --  --  --  --  --  --

GRANT INSERT ON shift, type_of_shift, customers, drivers, price, drivers, driver_trips, customers_in_trips, locations, zip
TO controller;
--  --   --  --  --  --  --  --  --  --  --  --  --  --

GRANT UPDATE ON locations, zip, updated_drivers_location, driver_trips, customers_in_trips
TO controller;
--   --  --  --  --  --  --  --  --  --  --  --  --  --
```

```
--Drivers
CREATE ROLE driver;
GRANT SELECT ON shift, type_of_shift, customers, drivers, driver_trips, locations, zip, price, customers_in_trips,
                        type_of_car, updated_drivers_location
TO driver;
--   --  --  --  --  --  --  --  --  --  --  --  --  --

GRANT INSERT ON shift, type_of_shift, customers, drivers, price, drivers, driver_trips
TO driver;
--   --  --  --  --  --  --  --  --  --  --  --  --  --

GRANT UPDATE ON updated_drivers_location, driver_trips
TO driver;
--   --  --  --  --  --  --  --  --  --  --  --  --  --
--Customers
CREATE ROLE customer;

REVOKE SELECT ON ALL TABLES IN SCHEMA PUBLIC
FROM customer;

GRANT INSERT, UPDATE ON customers, customers_in_trips
TO customer;
```

# IMPLEMENTATION NOTES

For this project I assume that controllers have a decent access to the database and are given an adequate amount of control over it. Internal management control is needed to ensure that database is true and genuine. Drivers also have some rights to change the database, however, these rights can be easily revoked. I assume that customers can see what cars are available, in what locations, what specific drivers are available, so, they are also granted a few rights. I assume that there is a website or an application or both for the customers to make orders themselves. Since wages per hour can change frequently due to shifts over holidays or breaks, or night shifts that can be more expensive, these prices have to be declared by administration manually for each shift.

# KNOWN PROBLEMS

There can be more types of employees for the business, such as maintenance, security, cafeteria, etc. These can be added, but then the database will need to be changed a little bit. This project is designed for the essential operations of a taxi company, small, start-up, or larger ones. The database will have to be changed significantly if being implemented for a very large taxi company that has an office, security guards, maintenance, cafeteria services, if they are also being accounted for in the database.

# FUTURE ENHANCEMENTS

A lot more queries and views can be added to the database for better presentation of operations and easier analysis of data. Depending on business rules that are in place for a specific business, more triggers can be implemented. If a taxi company that uses the database has agreements with other companies, then these rules can be implemented in the future. Some additional enhancements may include:

- Special customer requests
- Discount cards for frequent customers
- Services for long distance travels