**System Requirements**

With the expansion of the taxi Station, an innovative taxi booking system was needed. The system enables easier data management, like watching data on different travel bookings, editing data, seeing statistics summaries about the current situation on drivers, taxi cabs, incomes and expenses, and order history.

For the customer, the system allows viewing self-bookings and travel data.

The manager point of view is about managing each order records, where he can observe the different operations taking place at the station and can access or edit information in a much safer and faster way.

**General description**

* The Order has an id, station id, name of driver, origin and destination address, time of travel, route distance and cost.
* Address divides to ID, street name, number, city, phone number and zip code.
* Taxi Station has an id, and address.
* Taxi Cabs have id, model, company, and owner id - that determined by different ownerships: private owners or car rental companies.
* Private owners have an id and address.
* Rent companies has an id, name, address, number of rented cars, Date of rent, Price per company.
* Cab driver has an id, name, license number, date of birth.
* Payment has id, daily cost, nightly cost, and payment method.
* Customers has id, name, and date of birth.

**The user of the system:**

**Station Manager:**

The station manager can see all the information at the station, have all the permissions to edit the drivers and taxi cabs. all the data related to booking a ride by customers. On top of that he can incomes the expenses of the financial income.

**Customer:**

The customer has the permission to edit a new trip. He also has the option to view travel details and receive summary information about his trip.

**Description of Entities**

**Address:**

-ID, Street Name, House number, City name, Phone Number, Zip Code.

**owner :**

-Owner ID, Address ID.

**Taxi Station:**

-ID, Address ID.

**Cab** **driver:**

-ID, License Number, Name, DoB.

**Rent companies:**

-Company ID, Company Name, Number Of Rent Cars, Cost Of Rental, Start Rent Date, Rent Customer ID, Address ID.

**Taxi:**

-ID, Owner ID, Model, Company.

**Customer:**

- ID, Name, DoB.

**Trip Order:**

-ID, Travel Time, Route Distance, Cost, Station ID, Origin Address ID,   
Destination Address ID, driver-taxi ID, Driver ID.

**Payment:**

-ID, Payment Method, Daily Cost, Nightly Cost.

**Customer-Order:**

Customer ID, Order ID.

**Driver-Taxi:**

Pair\_ID, Driver ID, Taxi ID.

**Normalization**

we followed the three rules of normalization.

All values in a row are atomic values that can’t be separated to smaller values.

Every table has a not null primary key and all primary key columns on the entities are not dependent on each other.

-------------------------------------------------------------------

------------------CREATION--------------------

-------------------------------------------------------------------

create database TaxiStationFinalProject;

use TaxiStationFinalProject;

create table Address(

address\_ID int NOT NULL,

street\_name char(25),

street\_num int,

city\_name char(25),

phone\_num int,

zip\_code int,

primary key (address\_ID)

)engine=InnoDB;

create table TaxiOwner(

owner\_ID int NOT NULL,

owner\_name char(25) NOT NULL,

primary key (owner\_ID)

)engine = InnoDB;

create table TaxiStation(

station\_ID int NOT NULL,

address\_ID int NOT NULL,

primary key (station\_ID),

foreign key (address\_ID) references Address(address\_ID),

foreign key (station\_ID) references TaxiOwner(owner\_ID)

)engine = innoDB;

create table Driver(

driver\_ID int NOT NULL,

lisence\_num int NOT NULL,

driver\_name char(25),

DoB date,

primary key (driver\_ID, lisence\_num)

)engine = InnoDB;

create table RentCompany(

company\_ID int NOT NULL,

company\_name char(25),

rent\_customer\_ID int,

address\_ID int,

numOfRentCars int,

costOfAllRents int,

rent\_date date,

primary key (Company\_ID),

foreign key (rent\_customer\_ID) references TaxiStation(station\_ID),

foreign key (company\_ID) references TaxiOwner(owner\_ID),

foreign key (address\_ID) references Address(Address\_ID)

)engine = InnoDB;

create table Taxi(

taxi\_ID int NOT NULL,

owner\_ID int NOT NULL,

model char(25),

company char(25),

primary key (taxi\_ID, owner\_ID),

CONSTRAINT fk\_Update\_cascade

foreign key (owner\_ID) references TaxiOwner(owner\_ID)

ON update CASCADE

)engine = InnoDB;

create table Driver\_Taxi(

pair\_ID int NOT NULL,

driver\_ID int NOT NULL,

taxi\_ID int NOT NULL,

primary key (pair\_ID),

foreign key (driver\_ID) references driver(driver\_ID),

foreign key (taxi\_ID) references taxi(taxi\_ID)

)engine = InnoDB;

create table Customer(

customer\_ID int NOT NULL,

customer\_name char(25) NOT NULL,

DoB date,

primary key (customer\_ID)

)engine = innoDB;

create table TripOrder(

station\_ID int NOT NULL,

order\_ID int NOT NULL auto\_increment,

address\_Source\_ID int,

address\_Destination\_ID int,

timeOfTravel int,

numOfKM int,

driver\_taxi\_ID int NOT NULL,

cost int,

primary key (order\_ID),

foreign key (station\_ID) references taxiStation(station\_ID),

foreign key (address\_Source\_ID) references address(address\_ID),

foreign key (address\_Destination\_ID) references address(address\_ID),

foreign key (driver\_taxi\_ID) references Driver\_Taxi(pair\_ID)

)engine = InnoDB;

create table Payment(

order\_ID int NOT NULL,

costByDay int,

costByNight int,

paymentMethod char(15),

primary key (order\_ID),

constraint fk\_tripOrder

foreign key (order\_ID) references tripOrder(order\_ID)

ON delete CASCADE

)engine = InnoDB;

create table Customer\_Order(

customer\_ID int NOT NULL,

order\_ID int NOT NULL,

foreign key (customer\_ID) references customer(customer\_ID),

CONSTRAINT fk\_UpdateOrder\_cascade

foreign key (order\_ID) references tripOrder(order\_ID)

ON update CASCADE

)engine = InnoDB;

-------------------------------------------------------------------

---------------INSERTION---------------------

-------------------------------------------------------------------

INSERT INTO Address VALUES

(300,'Ben Gurion',27,'Tel Aviv',0527342149,123000),

(301,'Herzel',1,'Rishon Lezion',0547766801,556700),

(302,'Haim Weizman',77,'Yehud',0587941212,303001),

(303,'Zeev Jabotinsky',56,'Petah Tikva',036352417,477000),

(304,'Hagfanim',30,'Kiryat Ono',0,476000),

(305,'Igal Alon',14,'Tel Aviv', 0,321400),

(306,'Menahem Begin',132,'Kiryat Ono',0,545010),

(307,'Hacarmel',56,'Petah Tikva',0,143360),

(308,'Hacalanit',56,'Kiryat Ono',0,555700),

(309,'Sokolov',56,'Herzelia',0,239070),

(310,'Levi Eshkol',115,'Kiryat Ono',035356871,669000);

INSERT INTO TaxiOwner VALUES

(1000,'taxi station'),

(700,'exclusive cars'),

(701,'champions'),

(702,'fast and furious'),

(703,'cabs to rent');

INSERT INTO TaxiStation VALUES

(1000, 310);

INSERT INTO Driver VALUES

(100,200,'Eyal Cohen', '1978-02-01'),

(101,201,'Doron Kavalio', '1980-04-13'),

(102,202,'Israel Katorza', '1960-07-08'),

(103,203,'Menahem Levi', '1967-07-12');

INSERT INTO RentCompany VALUES

(700,'exclusive cars',1000,300,4,20000, '2005-01-01'),

(701,'champions',1000,301,2,25000, '2007-5-12'),

(702,'fast and furious',1000,302,1,5700, '2008-10-14'),

(703,'cabs to rent',1000,303,3,35000, '2002-12-02');

INSERT INTO Taxi VALUES

(500, 1000, 'corolla','Toyota'),

(501, 1000, 'corolla','Toyota'),

(502, 700, 'B-class','Mercedes'),

(503, 700, 'C-class','Mercedes'),

(504, 700, 'C-class','Mercedes'),

(505, 700, 'A-class','Mercedes'),

(506, 701, 'Rio','Kaia'),

(507, 701, 'Forte','Kaia'),

(508, 702, 'Octavia','Skoda'),

(509, 703, 'Rapid','Skoda'),

(510, 703, 'GT-350','Lamborgini'),

(511, 703, 'S-8','Audi');

INSERT INTO Customer VALUES

(80, 'Ronen Badash', '1990-01-07'),

(81, 'Itay Bell', '1989-07-06'),

(82, 'Lior Shir', '1985-03-01'),

(83, 'Nir Shemesh', '1992-12-22'),

(84, 'Liad Nahum', '1996-06-13');

INSERT INTO TripOrder VALUES

(1000, 1, 304, 305, 30, 20, 100,70), #day

(1000, 2, 304, 306, 7, 5, 101, 25), #night

(1000, 3, 306, 309, 30, 25, 101,65), #day

(1000, 4, 304, 307, 25, 22, 102, 100),#night

(1000, 5, 306, 307, 15, 10, 103, 50),#day

(1000, 6, 308, 309, 30, 20, 103,70);#day

INSERT INTO Payment VALUES

(1, 70, 0, 'Credit Card'),

(2, 0, 25, 'Cash'),

(3, 65, 0, 'cash'),

(4, 0, 100, 'Credit Card'),

(5, 50, 0, 'Cash'),

(6, 70, 0, 'Cash');

INSERT INTO Customer\_Order VALUES

(80, 1),

(81, 2),

(82, 3),

(80, 4),

(84, 5),

(83, 6);

INSERT INTO Driver\_Taxi VALUES

(100, 500),

(100, 511),

(101, 504),

(102, 505),

(102, 510),

(103, 506),

(103, 510),

(103, 501);

**Manager**

1 retrive total information on trip-bookings.

**sol 1**

select count(TripOrder.order\_ID) as sumOfBookings, sum(TripOrder.cost) as sumOfErnings, sum(TripOrder.numOfKM) as totalKM

from TripOrder;

2. retrive driver information who ride on specific vehicle.

**sol 2**

SELECT d.\*

from Driver as d join Driver\_Taxi as tx

on(d.driver\_ID = tx.driver\_ID)

where tx.taxi\_ID Like '510';

3. retrieve the maximum and minimum pay for the rent company.

**sol 3**

select max(costOfAllRents), min(costOfAllRents)

from RentCompany

where (rent\_date > '2003-01-01');

4. retrive all drivers that were late at least once.

**sol 4**

SELECT d.\*, o.\*

from TripOrder as o join driver as d join Driver\_Taxi as dt

on(o.driver\_ID = d.driver\_ID) and (d.driver\_ID = dt.driver\_ID)

group by o.driver\_ID

having (o.timeOfTravel >= 15)

order by (o.order\_ID);

5.

**sol 5**

select order\_ID, costByDay

from payment

where (costByDay > 0) and costByDay < all

(select costByNight

from payment

where costByNight in (50,100))

order by costByDay;

6. retrive all viechls and their driver that been used more then twice.

**sol 6**

SELECT tx.taxi\_ID, d.\*

from Driver as d join Driver\_Taxi as tx

on(d.driver\_ID = tx.driver\_ID)

group by tx.taxi\_ID

having tx.taxi\_ID > 2

order by tx.taxi\_ID;

**Customers**

1. retrive order information by customer id.

**sol 1**

select c.customer\_name, o.\*

from customer as c join customer\_order as co join triporder as o

on (c.customer\_ID = co.customer\_ID) and (co.order\_ID = o.order\_ID)

where c.customer\_name like ('Nir Shemesh');

2. retrive driver information to complaint about the long drive , via order-ID.

**sol 2**

select c.customer\_name, a.city\_name, o.cost, d.driver\_name, d.driver\_ID

from customer as c join customer\_order as co join triporder as o join driver as d join address as a

on (c.customer\_ID = co.customer\_ID) and (co.order\_ID = o.order\_ID) and (o.driver\_ID = d.driver\_ID) and (o.address\_Source\_ID = a.address\_ID)

where o.order\_ID like 6;

3. retrive the avarge time for a drive and and avrage cost for specific destination.

**sol 3**

select a.city\_name, avg(o.timeOfTravel), avg(o.cost)

from customer as c join customer\_order as co join triporder as o join Address as a

on (c.customer\_ID = co.customer\_ID) and (co.order\_ID = o.order\_ID) and (o.address\_Destination\_ID = a.address\_ID)

group by a.city\_name

having a.city\_name like 'Kiryat Ono';

4retrive total payment for the taxi station.

**sol 4**

select c.customer\_name, sum(o.cost)

from customer as c join customer\_order as co join triporder as o

on (c.customer\_ID = co.customer\_ID) and (co.order\_ID = o.order\_ID)

group by c.customer\_ID having c.customer\_name = 'Ronen Badash';

5. retrive customer name who paid by credit card

**sol 5**

select distinct c.customer\_name

from customer as c join customer\_order as co join triporder as o join Payment as p

on (c.customer\_ID = co.customer\_ID) and (co.order\_ID = o.order\_ID) and (o.order\_ID = p.order\_ID)

where p.paymentMethod like 'Credit Card';

6. retrive cost per KM for trip taken place at day time, with minimum fee.

**sol 6**

select (cost / numOfKM) as costPerKM

from triporder as o join payment as p

on (o.order\_ID = p.order\_ID)

where (costByDay > 0) and costByDay > any

(select costByDay

from payment

where costByDay in (50,65))

order by costByDay;

**triggers**

CREATE TRIGGER deleteTaxi

after delete

on Taxi

FOR each row

DELETE FROM Taxi

WHERE company\_name like 'cabs to rent';

create trigger updateNumOfCars

before update

on RentCompany

for each row

set new.numOfRentCars = 5;