

First of all I want to thank you for purchasing my work. If you have any questions that are beyond the scope of this help file, feature requests, bug report or anything else, please feel free to email me via my email pnjkenya@gmail.com or 0701628981 and i'll do my best to reply back within 24 hours. Thanks so much!

This year's project required the candidates to create a database that would enable a user perform the following fundamental tasks:

1. Maintain records of members.
2. Maintain records of vehicles.
3. Maintain records of drivers and loaders.
4. Maintain records of goods transported.
5. Maintain records of farmers who are in groups.
6. Maintain records of orders for transport.
7. Maintain records of offences committed by drivers.
8. Maintain records of expenses for each vehicle.

The system should also do a number of computations. Computations are automated calculations done by queries as soon as data is fed into the tables. The system should:

- Compute payment for a loader and a driver.
- Compute penalties surcharged on drivers
- Compute loading fee vehicle per trip.
- Compute revenue per vehicle per trip
- Compute expenses per each vehicle.
- Compute tax payable.
- Compute company expenses
- Compute total revenue for the company
- Compute overall company profit.

All the operation the system should perform and handle might sound quite a handful but trust me, it's a piece of cake. I am hereby going to discuss the MOVERS TRANSPORT SYSTEM particularly the DATABASE STRUCTURE OF THE SYSTEM. See you in the next chapter

Chapter 1

Database structure in ms access involves creating tables with well defined fields and datatypes to store the information we want for easy access manipulation and retrieval. Am going to dive straight in and point out key features which are going to help us develop this database architecture as the knec wants us to. First we need a database that should maintain records of members. We know that there are two types of memberships.

- Farmers registered as large-scale farmers (individual farmers).
- Farmers registered under groups (group farmers).

With this information, we know that we need to create tables, one to store all the individual farmers, one to store all farmers who are registered under groups (group farmers). The group farmers must be attached to a registered group. This leads to another conclusion that “: we need another table to store all the groups in the organization.”

So far, we know for sure that we need 3 fundamental tables.










Moving forward, the system requires us to maintain records of vehicles in the organization. This forces us to create another table that will hold all the vehicles that the organization has. With that, we know that we need 4 tables. Moving forward, we need to maintain records of drivers and loaders. The drivers need to be attached to vehicles, a scenario that we will tackle later. We need two additional tables. One to hold records of all drivers in the organization and another one to hold records of all loaders in the system. So far so good. We have 5 tables.

- Individual Farmers table
- Group farmers table
- Groups table
- Drivers table
- Loaders table
- Vehicles

Finally, we need to maintain records of orders made by the farmers as well as goods transported and expenses. With that we will need additional tables. First we need a table to hold all orders made all farmers. Second we will need to create a table that holds all records of verified orders. When a farmer makes an order, the orders is processed and has to be verified indicating that the order actually went through. Also we need a table to hold records of driver offenses. i.e Offenses committed by drivers.

We have 8 tables in total.


Tables

 driver_offenses
 drivers
 group_farmers
 groups
 IndividualFarmers
 loaders
 order_transport
 vehicles
 verify_order

I'll be tackling each table in details explaining all the fields and data types as well as the configuration of index, primary keys and foreign keys.

Individual Farmers table

This table as stated above will hold a record of all large scale farmers who are not under any groups. The table needs to define their personal details such as id number, their names, location, gender contact and an avatar image. See a screen shot of these details below.

IndividualFarmers		
	Field Name	Data Type
	id_number	Short Text
	first_name	Short Text
	sur_name	Short Text
	last_name	Short Text
	gender	Short Text
	contact	Short Text
	email	Hyperlink
	location	Short Text
	image	Attachment

The id number will act as a primary key which will be a unique identify that uniquely identifies every farmer in this table. The first name sur name and last name will be short text. Gender will define if they are male or female. The other fields are additional information collected that further describe this farmer. This is a simple table and does not fetch any data from any other tables.

Group farmers table

This table defines and holds records of farmers who are registered and various groups. This table is a little bit more complex than the individual farmers table since we have to attach a farmer to a group.

First we need a field which is **lookup field** which will fetch all the group registered in the organization. After that, we define details of the farmer such as id number and name and other additional details.

group_farmers	
Field Name	Data Type
group	Number
id_number	Short Text
first_name	Short Text
sur_name	Short Text
last_name	Short Text
gender	Short Text
contact	Short Text
email	Hyperlink
image	Attachment

The id number is the primary key as it uniquely identifies each farmer. The group field is a look up field which fetches all group (pk) from the groups table which are going to structure up next.

group_farmers							
group	id_number	first_name	sur_name	last_name	gender	contact	email
Wuungano	Lindsay	mwania	mutuku	maina	female	072123232	pta@gmail.co
*							

Groups Tables.

This table defines all the group registered in the organization which farmers can register through and make orders through. This table is simple and must have a minimum number of 5 farmers. Its structure is

fairly simple as we define the group name , id , location and then we define a yes/no field which defines if the group qualifies to make orders.


groups	
Field Name	Data Type
group_id	AutoNumber
group_name	Short Text
location	Short Text
qualify	Yes/No

Here the group id is the primary key is it uniquely identifies the groups in the table. We then define the group name and qualify field.


groups					
	group_id	group_name	location	qualify	Click to Add
+		3 Muungano fai makindu		<input type="checkbox"/>	
*	(New)			<input type="checkbox"/>	

Drivers table.

This table should hold all records of drivers in the organization. It's a simple table with fields that define basic information of a driver. You'll notice that this table is almost similar to loader table. Here we define the id number, names of driver, location and all relevant information about the driver. See the figure below.

drivers	
Field Name	Data Type
 id_number	Short Text
first_name	Short Text
sur_name	Short Text
last_name	Short Text
gender	Short Text
contact	Short Text
email	Hyperlink
misbehave	Yes/No
image	Attachment

The id number is the primary key here as it uniquely identifies the drivers. The image field has the datatype of **attachment** as we need to attach an image of the driver as we register them.

drivers								
	id_number	first_name	sur_name	last_name	gender	contact	email	misbehave
	445454	Testing	Trial	one	Male	9909090090	pta.com	<input type="checkbox"/>
*								<input type="checkbox"/>

Vehicles Table.

This table is a little bit complicated as we need to link a vehicle to a given driver for future computation and such details. This table needs to hold all records of vehicles that the company has. We know that the organization has 4 types of vehicle:

- Pick ups
- Lorry
- Trailer
- Refrigerated trucks.

In the fields, we need a primary key which will be unique to each vehicle owned by the company. I chose to go with **number plates**. After that, we define the type of vehicle. Here I used a lookup field where I typed the values I want to see.

After that, we define a driver field. This field define the driver who is responsible for this vehicle. This field is a look up field which fetches its data from the Drivers Table. It's a foreign key.

With this data, we record additional info such as model of the vehicle and the date the vehicle was purchased.

	Field Name	Data Type
🔑	plates	Short Text
	type	Short Text
	driver	Short Text
	available	Yes/No
	model	Short Text
	purchase_date	Date/Time

Driver offenses Table

This table holds all records of offenses committed by the drivers. Here we need a unique Id for every offense. Then we need a foreign key which links to the primary key of the Driver Table. This will ensure that an offense is linked to a driver in the organization.

driver_offenses	
Field Name	Data Type
offense_id	AutoNumber
id_number	Short Text
offense	Short Text
offense_date	Date/Time


The offense_id is a primary key while the id_number is a lookup field that fetches data from the DRIVER table. The offense is a look up field which defines all the possible offenses that a given driver can commit. i.e

offense_id	id_number	offense	offense_date
3	Testing	Over speed	15/08/2021
(New)		Over speeding	
		Overloading	
		Driving while	
		Cause perisha	




The offense date is a date/time data type which gives us a little calendar widget to select the date.

Loaders Table

This table is supposed to hold all records of loaders in the organization. It's a fairly simple table which will define the basic information of the loaders. The fields are the loaders id number which will be the primary key, the name and additional info such as location , email and telephone number.

loaders	
Field Name	Data Type
 id_number	Short Text
first_name	Short Text
sur_name	Short Text
last_name	Short Text
gender	Short Text
contact	Short Text
email	Hyperlink
image	Attachment
vehicle	Short Text

We also have an image filed which is an attachment to hold the images of the vehicles as they are being registered.

loaders								
id_number	first_name	sur_name	last_name	gender	contact	email		vehicle
 909090	the first	the new	whu	male	9898989	pta@gmaki	 (0)	Kbz/23/43
*							 (0)	

Order transport table

This is a very crucial table but a little bit complex in that this table should hold records of all orders ever made by both individual farmers and group farmers. First we need to define a primary key to identify each order uniquely. We use the order_id which will be a primary key. It's an auto number meaning it will be generated every time a new entry is made. After that we define a foreign key which will fetch all the groups that qualify to make an order. This will be a lookup field that fetches data from the group table filtering only those that are qualified to make orders. Under that we define another foreign field that is the individual farmer which holds index of the individual farmer making an order.

We then define the goods to be transported i.e cereals, livestock etc.

We have to define the vehicle that was used to transport the orders. This is a foreign key which is a lookup field fetching data from the Vehicle table.

Field Name	Data Type
order_id	AutoNumber
group_farmers	Number
individualFarmer	Short Text
goods_transported	Short Text
transport_means	Short Text
destination	Number
processed	Yes/No
order_date	Date/Time

order_id	group_farm	individualFarmer	goods_tran	transport_mean	destination	processed
5		Peter	Cereal	Pick-up	10	<input checked="" type="checkbox"/>
6			Cereal		0	<input checked="" type="checkbox"/>
(New)			Livestock		0	<input type="checkbox"/>
			Milk			
			Eggs			
			Fish			
			Flowers			
			Fertilizers			
			Manure			
			Pesticides			
			Seeds			

The transport means is a look up field.

order_id	group_farm	individualFarmer	goods_tran	transport_mean	destinatio
5		Peter	Cereal	Pick-up	
6				Pick-up	
(New)					

We then define a very crucial field, which is the destination. This is the mileage to be covered in Kilometers. This value will be used to calculate the amount charged on the order.

The processed field define orders that have been processed. It's a yes /no field. Once an order has been processed this field is checked to distinguish between orders that are still pending from orders that have been processed.

I know it looks weird that we have both the individual farmer field and the group field in the same table. So how are we going to populate these fields ? We know that only one of the fields can be populated at any given instance. The work around is simple. Have two forms. One for handling individual orders and one for handling group orders.



If a user selects the individual order, we display all the fields in the form with an exception of the Group_farmer field.

If a user selects the group order button, we display all the fields with an exception of the individual_farmer field.

Order Transport as Group

order_id: New

group_farmers: [dropdown arrow]

goods_transported: [dropdown arrow]

transport_means: [dropdown arrow]

destination: 0

Record: 1 of 1 | No Filter | Search

This allows us to dynamically populate the database depending on the user's choice.

All the other forms are straight forward.

Register Individual Farmer

[Profile Picture Placeholder]

id_number: [input field]

first_name: [input field]

sur_name: [input field]

last_name: [input field]

gender: [input field]

contact: [input field]

email: [input field]

location: [input field]

Record: 1 of 1 | No Filter | Search

A form to populate individual farmers tables.

verify_order | verify_order : Query Builder

order_transport

- * order_id
- group_farmers
- individualFarmer
- goods_transporte
- transport_means

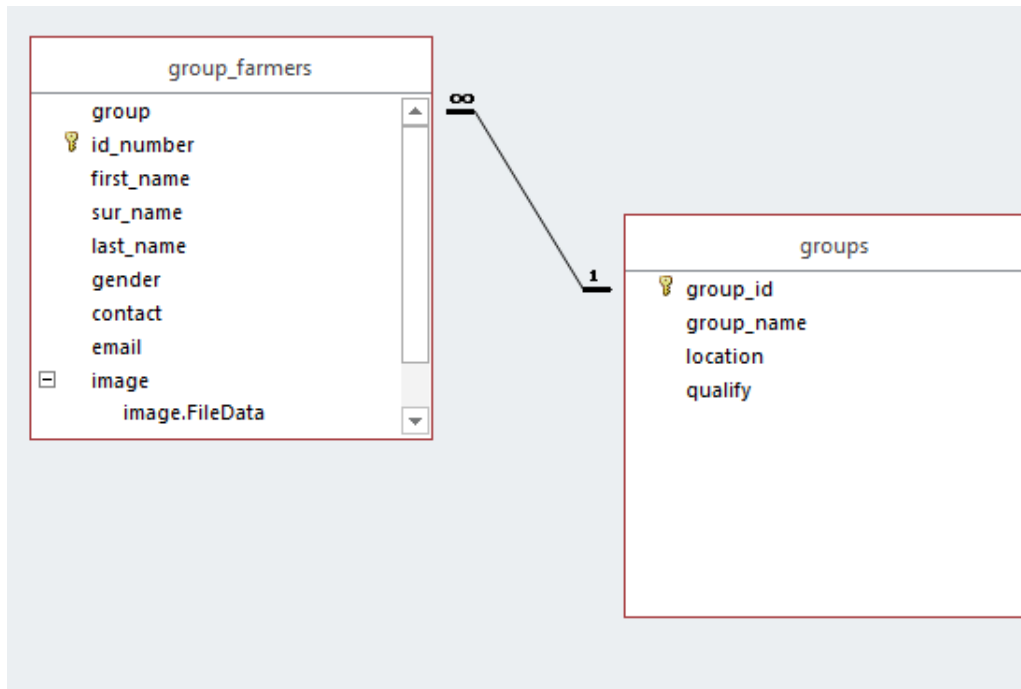
Field:	order_id	group_farmers	individualFarmer	processed
Table:	order_transport	order_transport	order_transport	order_transport
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				False
or:				

In the lookup filter query we do the filtering.

With this, we are sure that in the verify order form, we will only be able to verify orders that are not yet processed. Once an order is processed and the processed status changed to yes, the filtering algorithm will prevent duplicates.

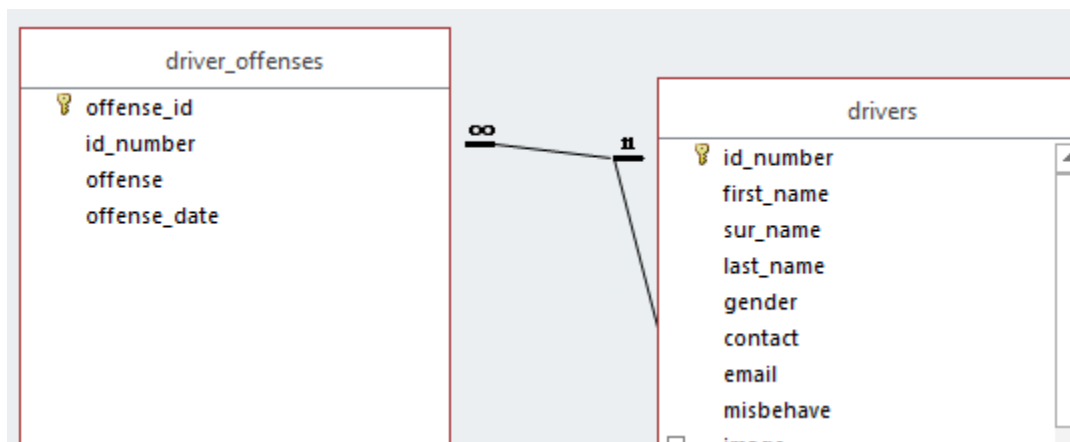
Database normalization and relationships

Now that the fields have been defined, we have to create a relationship between these tables to allow us have access to fields in other tables. Relationships are a powerful tool in ms Access as it enables the tables to communicated with one another passing data via primary key and foreign key hooks hence reducing duplications and repetitions. First we need to join the group to the group farmers.

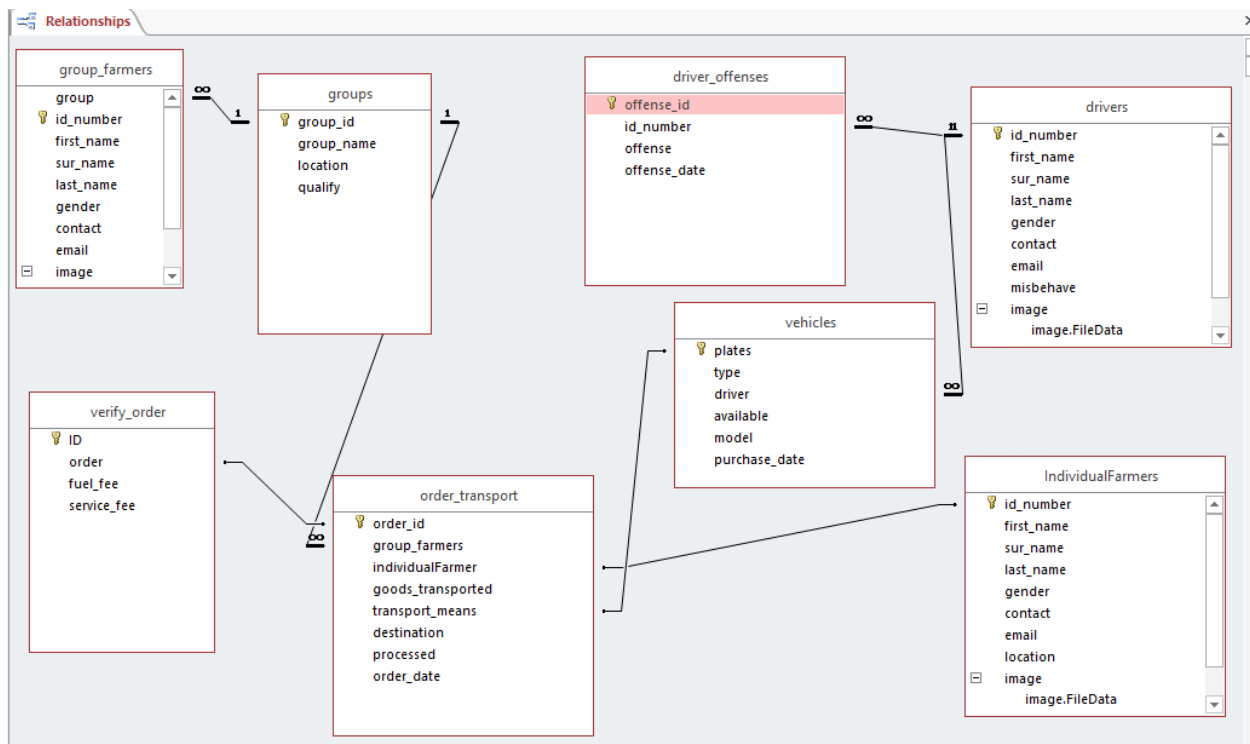


Here, we use the primary key group_id and join it to the foreign key (group) in the group_farmers table. This allows us to access data of the group from the group farmers instances.

We then create a link between a driver and driver offenses.



In the next page, I'll illustrate the overall layout of the relationships between the tables.



These relationships are very important as they will allow us to create dynamic queries and reports as well as computation.

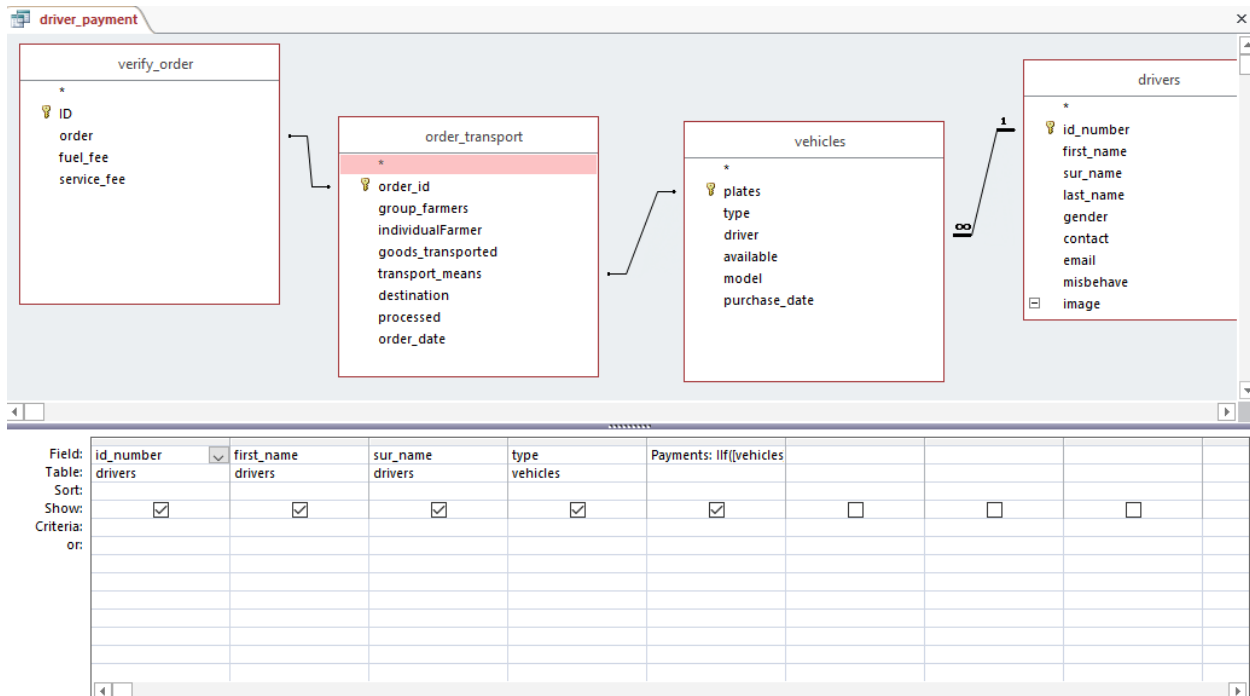
So far, we have all records of members, vehicles, loaders and drivers, farmers in groups (group farmers), orders for transport, offences committed by drivers. The records in the tables can be translated into reports.

In the next section I'll handle computation of the data as well getting data from the tables into queries.

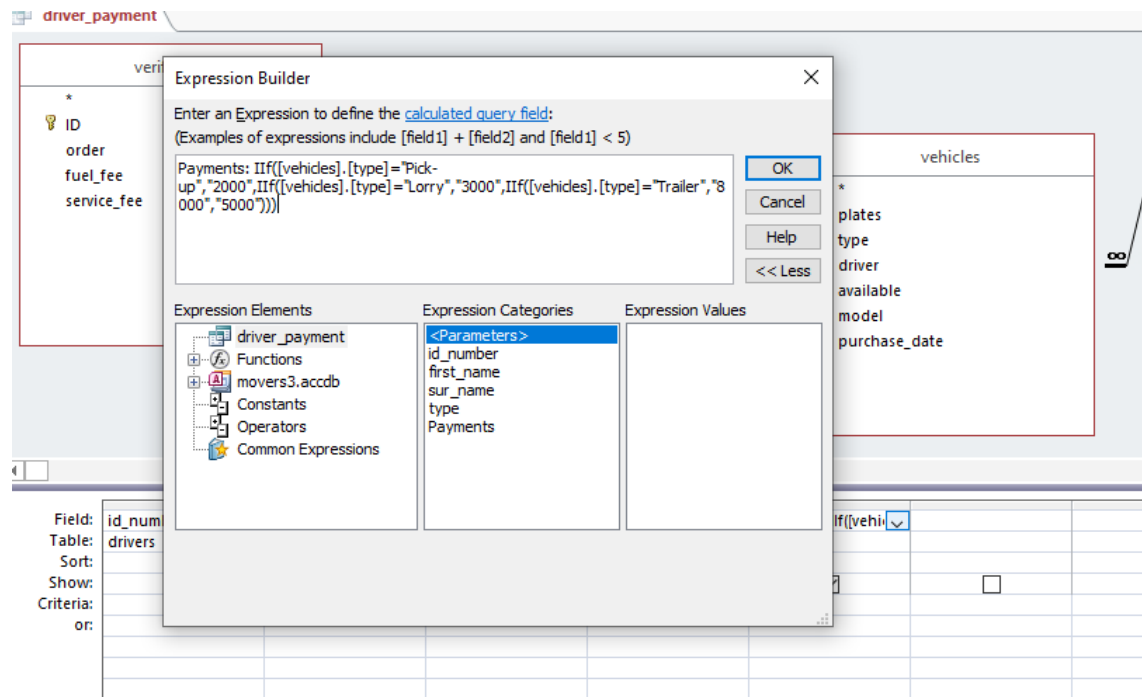
Computation of the data in the tables.

Payment for a loader and a driver.

To get the payment for a driver, we need to create a query based on each driver. From that we can add the verified order table into the query and build an expression to calculate the payment for each trip the driver makes.



Notice that we link the driver to a vehicle which is linked to an order which finally maps to a verified order. Here we list all drivers who have done a delivery. The payment field is calculated field.



We use the iff statement which allows to check the vehicle type attached to a driver. With these parameters we can then calculate the amount they should be paid depending on the vehicle type. The values used in the query corresponds to the ones in the knec documentation ie 2000 for pick up, 3000 for lorry etc. This query will give us a list of all payments for a driver per trip.

id_number	first_name	sur_name	type	Payments
445454	Testing	Trial	Pick-up	2000
389839	mutuku	maingi	Lorry	3000

With this query we can then generate a report based on every driver.

Payment for Drivers

id_number: 389839

first_name: mutuku

sur_name: maingi

type: Lorry

Payments: Kab. 3000

id_number: 445454

first_name: Testing

sur_name: Trial

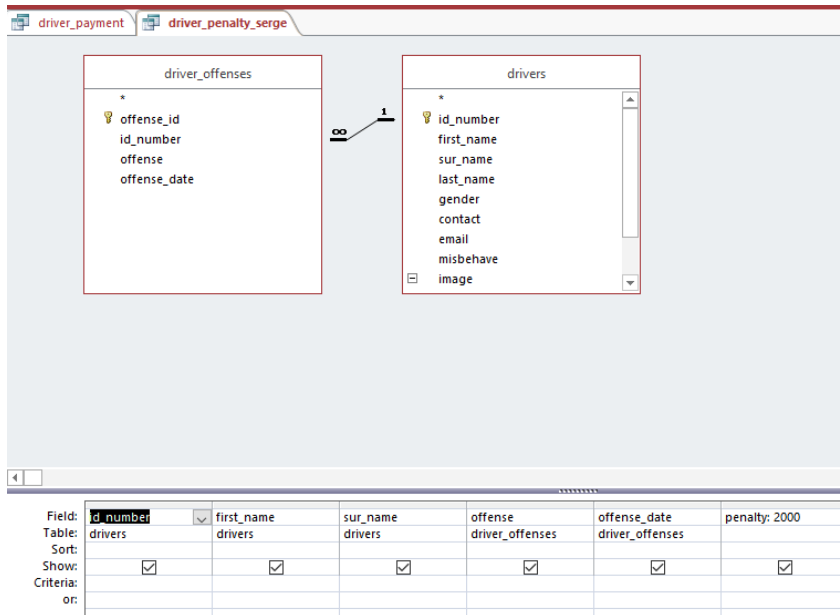
type: Pick-up

Payments: Kab. 2000

Page 1 of 1

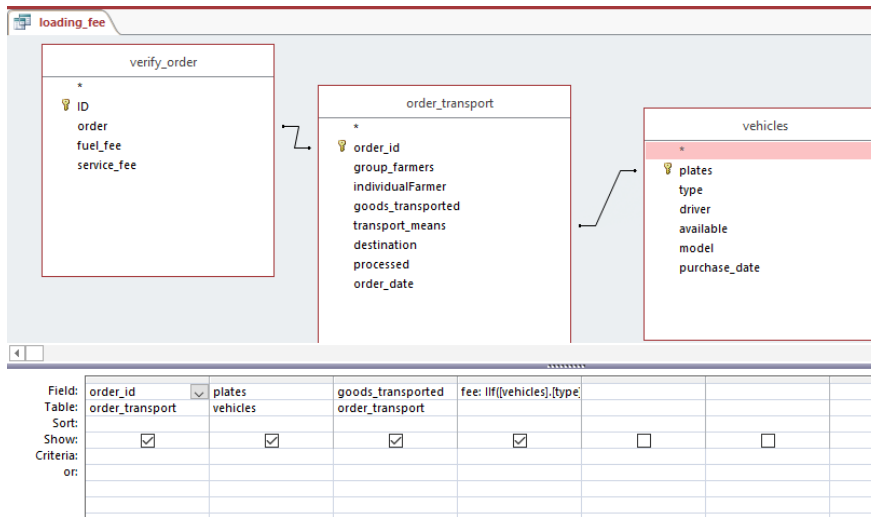
Penalty surged on drivers

This was a little tricky since the Kneec documentation does not explicitly explain how the penalty should be charged. However, I had to improvise a method. Here you create a query based on the driver offenses table. On every offense, you charge an amount of say 200 ksh. With that you'll have a list of penalty surges per every offense.



Loading fee per vehicle

To calculate the loading fee, the records have to be based on verified orders.



The fee is calculated depending on the vehicle type. Again we use the **iff** statement.

Enter an Expression to define the **calculated query field**:

(Examples of expressions include [field1] + [field2] and [field1] < 5)

fee: IIf([vehicles].[type]="Pickup","400",IIf([vehicles].[type]="Lorry","1800",IIf([vehicles].[type]="Trailer","5000","1800")))

Expression Elements: loading_fee, Functions, movers3.accdb, Constants, Operators, Common Expressions

Expression Categories: <Parameters>, order_id, plates, goods_transported, fee

Expression Values: plates, type, driver, available, model, purchase_date

Field:	order_id	plates	goods_transported	fee
Table:	order_transport	vehicles	order_transport	order_transport
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				
or:				

We know that for a pickup two loaders are required and each one is paid 200ksh. So the total loading fee for a pickup is 400. Same concept applies to lorries and other vehicle types.

order_id	plates	goods_trans	fee	type
5	Kbz/23/43	Cereal	400	Pick-up
7	KCA/23/12	Livestock	1800	Lorry
*(New)				

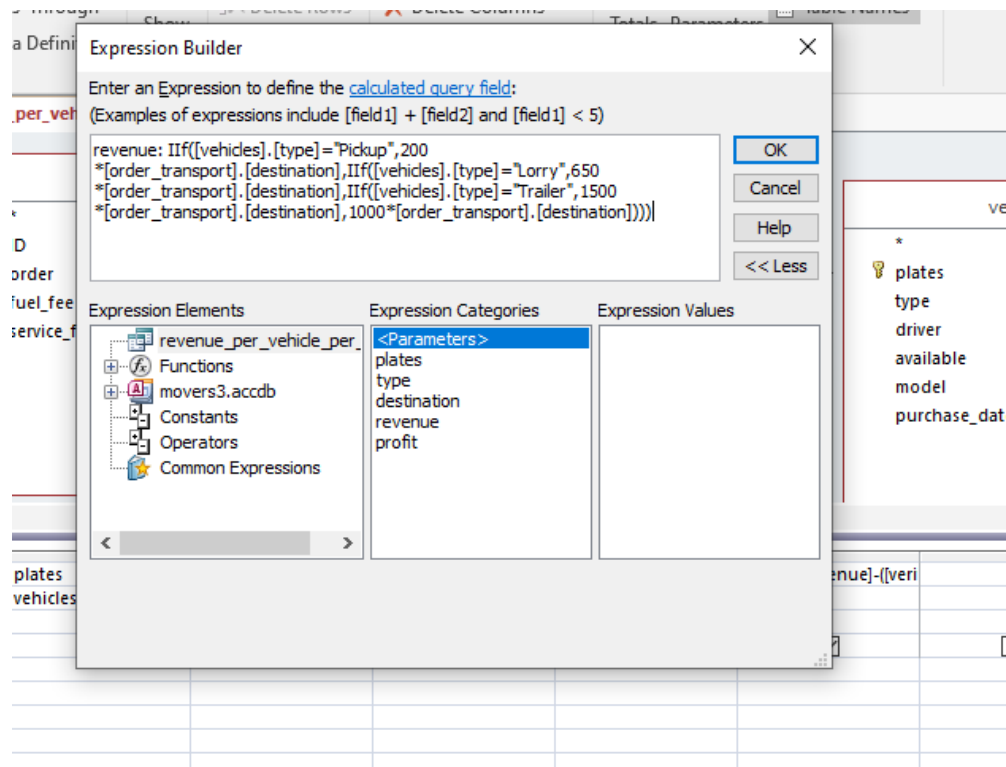
Notice how the fee changes depending on the vehicle type. With this query you can generate the corresponding report.

Revenue per vehicle per trip.

Here we need to base our query on the verified orders then use the records fetched to calculate the amount earned from these orders.

Field:	plates	type	destination	revenue: If([vehicles].	profit: [revenue]-([veri				
Table:	vehicles	vehicles	order_transport						
Sort:									
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Criteria:									
on:									

Notice how we first get the relevant records fetches. Then we define a revenue field where we calculate the revenue generated by the vehicle depending on the vehicle type. If vehicle is pick up we calculate the destination mileage with 200/= which is the cost per kilometer. If the vehicle is a lorry, we use the value 650/= and so on. See image below.



The value obtained from the query above is the revenue generated by vehicles per trip.

revenue_per_vehicle_per_trip					
plates	type	destination	revenue	profit	
Kbz/23/43	Pick-up	10	2000	-Ksh4,000.00	
KCA/23/12	Lorry	100	65000	Ksh63,000.00	
*					

With these records you can map the query to a report basing each record on a single vehicle.

With these records in the query we can generate a report based on this query.

Expenses per each Vehicle ✕

plates	Kbz/23/43		
type	Pick-up		
fuel_fee	service_fee	gross	
\$5,000.00	\$5,000.00	Ksh6,000.00	
Total	\$5,000.00	\$5,000.00	Ksh.6000
Sub Total:	\$5,000.00	\$5,000.00	Ksh6000

plates	KCA/23/12		
type	Lorry		
fuel_fee	service_fee	gross	
\$1,000.00	\$1,000.00	Ksh2,000.00	
Total	\$1,000.00	\$1,000.00	Ksh.2000
Sub Total:	\$1,000.00	\$1,000.00	Ksh2000

Grand Total:	\$4,000.00	\$4,000.00	Ksh.8000
---------------------	------------	------------	----------

The computation of the Tax payable

From The total revenue collected, 20% is remitted as tax. To Calculate this value, we need to get the total revenue collected. We base our new query on another query. Confused? What am saying is, to get the total revenue collected, you need to sum up all the revenues of the individual vehicles. To get these records, we use the “Revenue per vehicle” query. We sum up all the rows to get the total revenue.

tax_payable_calc

revenue_per_vehicle_per_trip

*

plates
type
destination
revenue
profit

Field: revenue
Table: revenue_per_vehicle_per_trip
Total: Sum
Sort:
Show: ☒
Criteria:
or:

Notice that we use the Total **Sum** function on the Revenue field in the revenue per vehicle per trip query. After getting this value, we get 20 % of this value and that is the tax payable.

tax_payable_calc

*

SumOfrevenue

Field: tax: 0.2*[SumOfrevenue]
Table:
Sort:
Show: ☒
Criteria:
or:

Total Company Expenses

To get the total company expenses, we just sum up all the expenses per vehicle.

The screenshot shows a database query interface. At the top, a table named 'vehicle_expenses' is displayed with the following fields: plates, type, fuel_fee, service_fee, and gross. Below the table, there is a query builder interface with the following fields:

Field:	Table:	Total:	Sort:	Show:	Criteria:
gross	vehicle_expenses	Sum		<input checked="" type="checkbox"/>	
				<input type="checkbox"/>	
				<input type="checkbox"/>	

Overall company profit

We know that $\text{profit} = \text{revenue} - \text{expenditure}$.

To get the profit, we get the overall revenue subtract the expenditure and also subtract the tax.

Overall_company_profit

tax_payable_calc

*

SumOfrevenue

grand_total_expenses

*

T_expenses

final_tax_payable

*

tax

Field:

T_profit: [SumOfrevenue]-[T_expenses]-[tax]

Table:

Sort:

Show:

Criteria:

or:

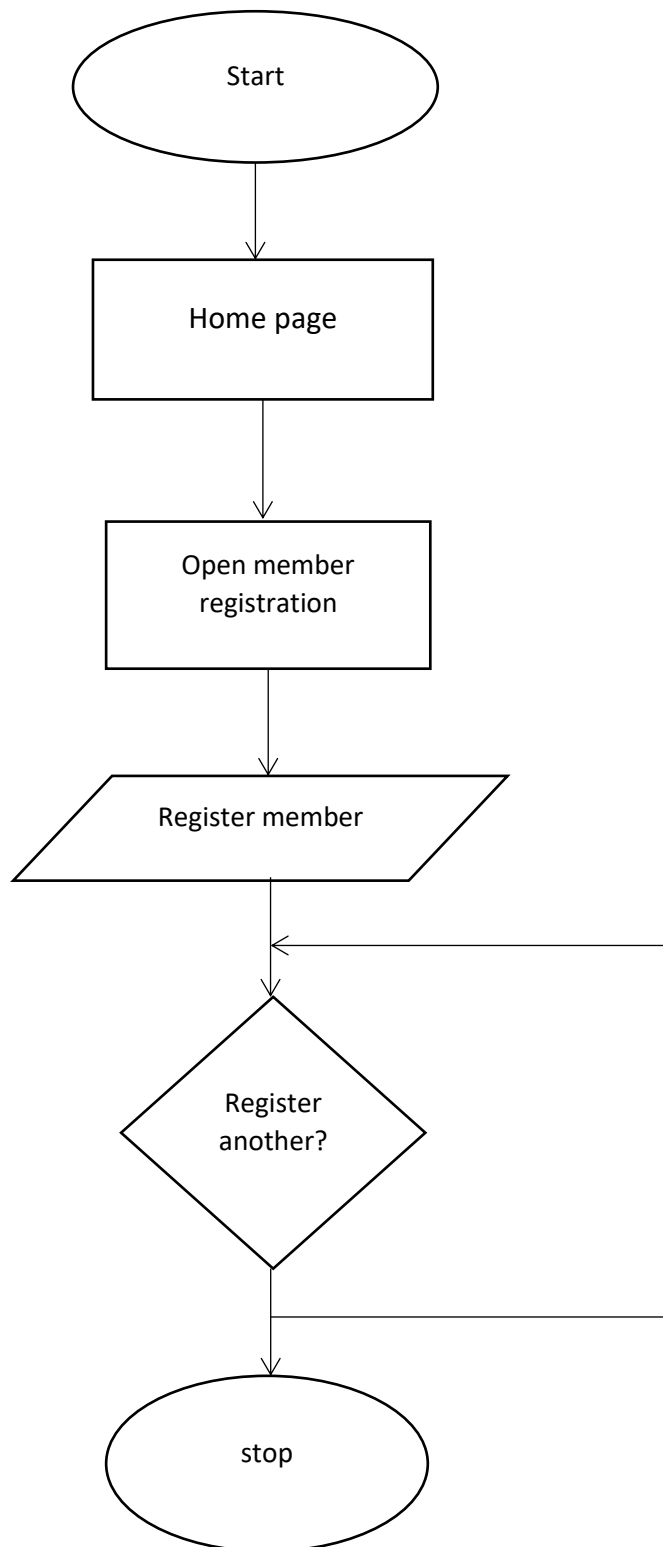
Note that these operations are based on other queries. So be careful how you implement these queries since a slight error in one parent query affects all the nested child queries.

Conclusion.

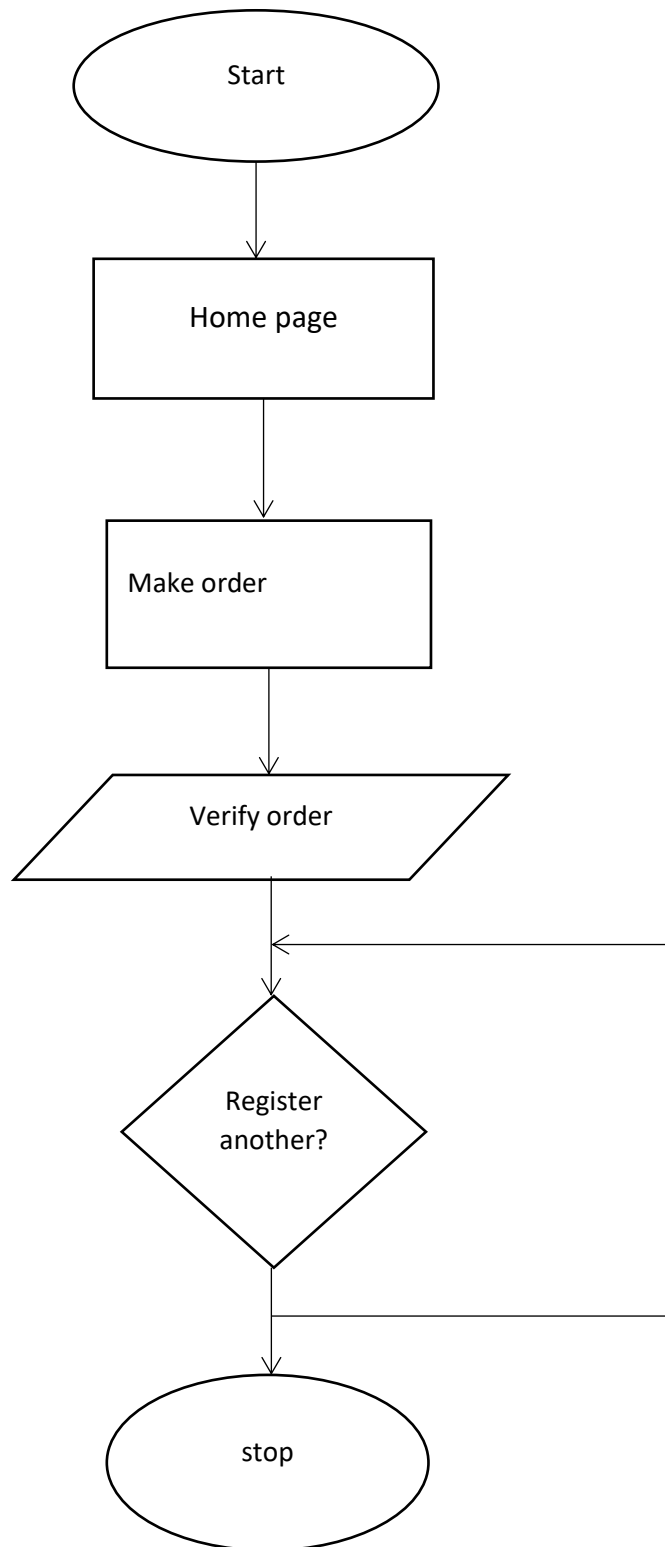
With all these tables and queries, you can generate the appropriate reports, build forms to input data into the database. That marks the end of the database structure of the Movers Transport system.

Flowcharts.

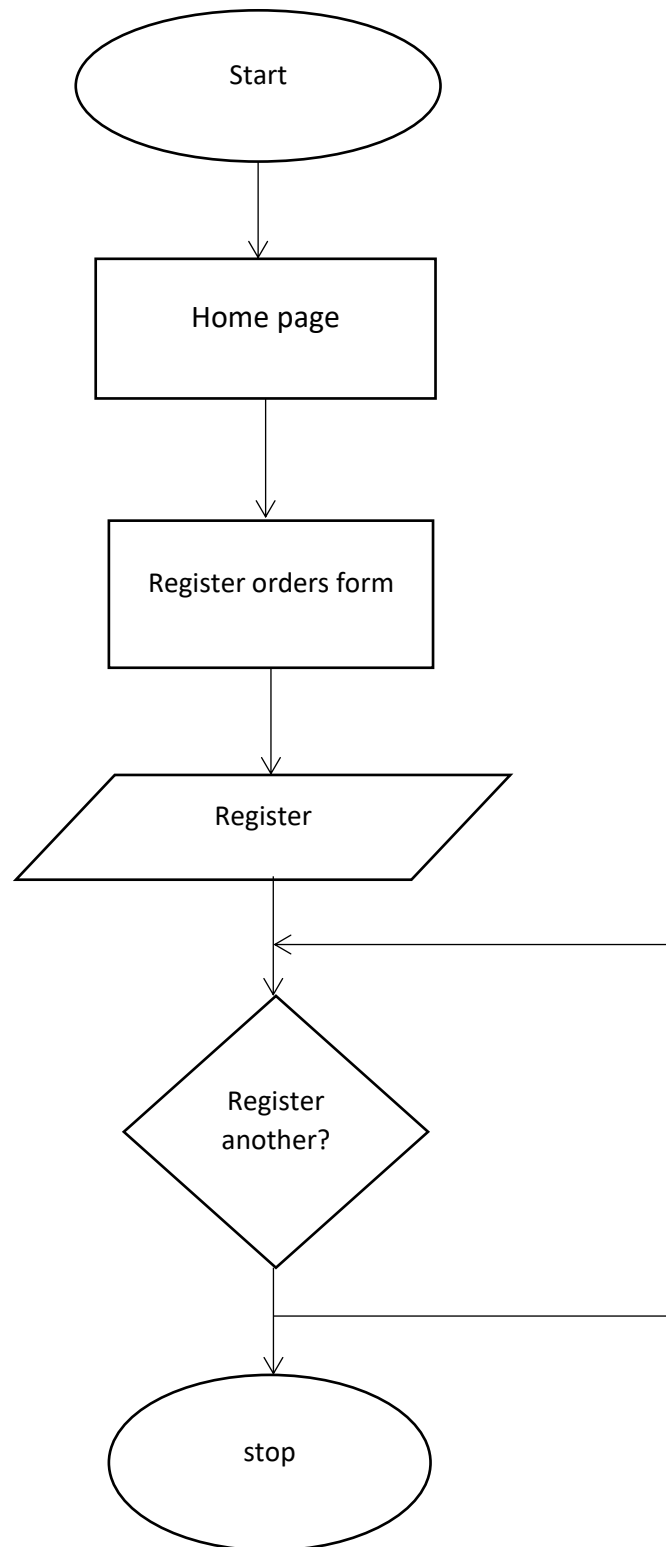
MEMBER REGISTRATION



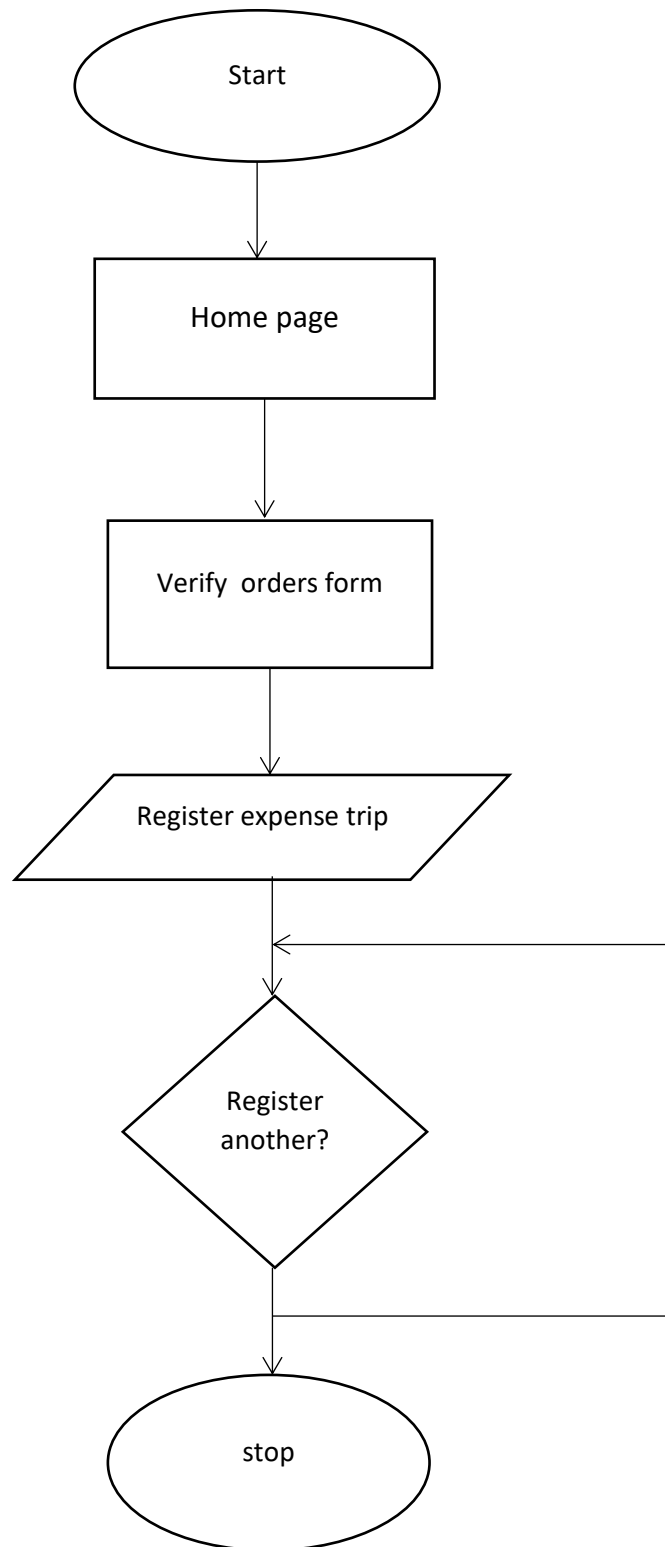
PAYMENTS OF DRIVER AND LOADER



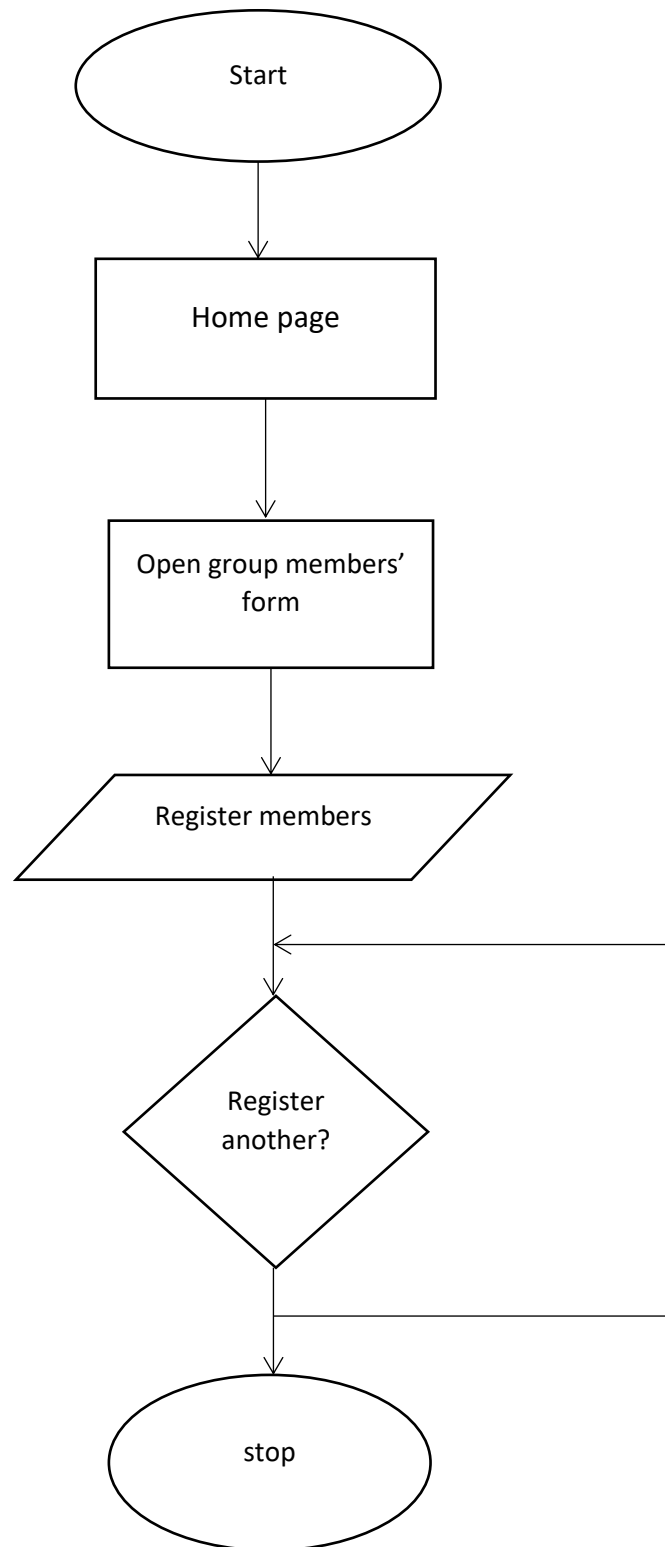
ORDERS FORM REGISTRATION



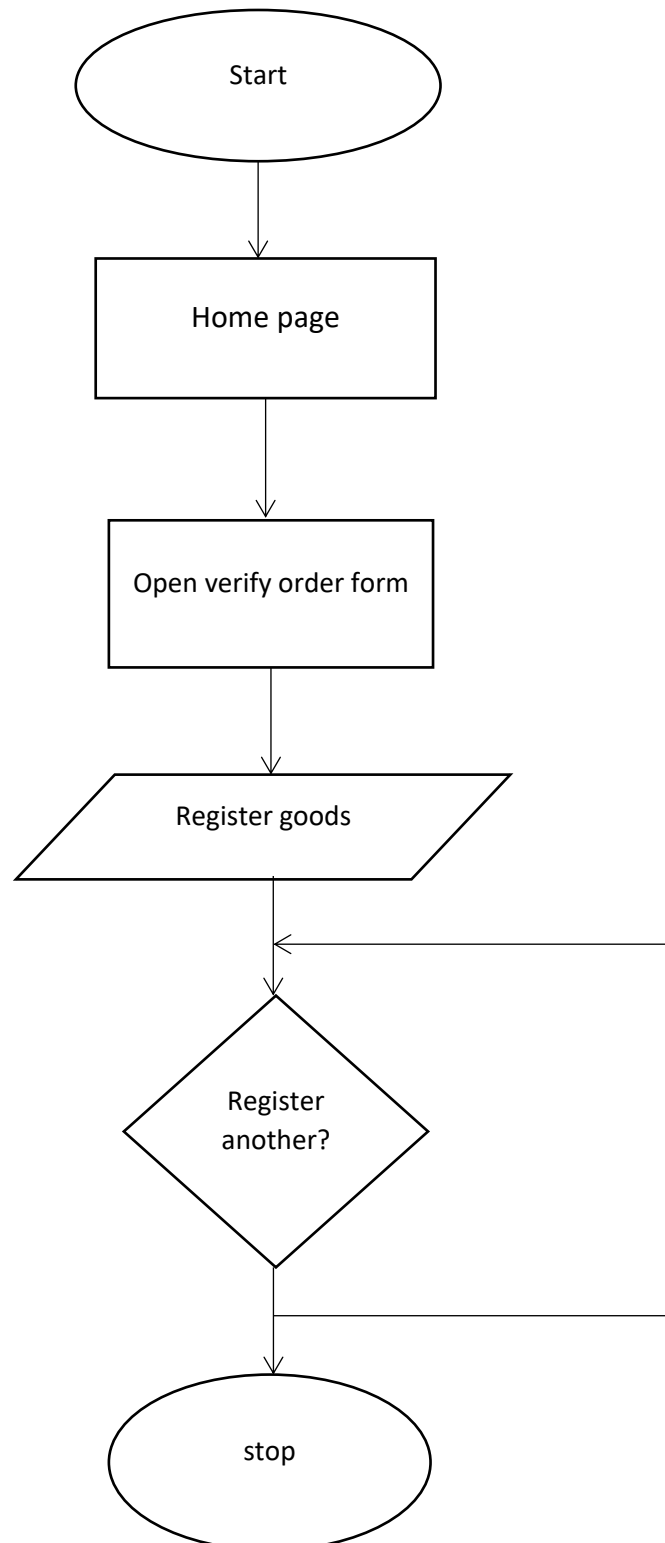
EXPENCES FORM



GROUP MEMBERS FORM



GOODS TRANSPORTED FORM



OUTPUT FLOWCHAT

