

# Introducción del Negocio

TELLEVO es una plataforma tecnológica. Nuestra app para smartphones conectan a socios (drivers) de la App y usuarios que la han descargado y registrado.

Usa la app para solicitar viajes en las ciudades donde opera. Cuando un driver que está cerca acepta tu solicitud, la app te muestra su tiempo estimado de llegada al punto de partida, y te notifica cuando está a punto de llegar.

También muestra información acerca del Driver de la App con el que viajarás, como su nombre, tipo de vehículo y número de matrícula. Estos datos ayudan a que ambos se encuentren en el punto de partida, además de darle la confianza a los clientes que realizaran un viaje seguro.

Usa la app para ingresar tu destino en cualquier momento, ya sea antes o durante el viaje. Si tienes una ruta preferida, compártela con el driver de la App.

El viaje termina cuando llegas al destino y bajas del vehículo. La tarifa se calcula automáticamente, dependiendo de la ciudad y se cobra al método de pago vinculado a tu cuenta.

En algunas ciudades se puede pagar en efectivo. Esta opción se debe seleccionar antes de solicitar el viaje.

## Problemática

Actualmente, los pobladores de las diferentes ciudades de Nicaragua están viviendo un gran problema con respecto al transporte que hay en este país. Se detectó que en los últimos años ha venido un incremento, en asaltos y aumento exagerado de tarifa en los taxis de este país. Las personas que no poseen vehículo y se quieren transportar en un automóvil privado o simplemente no quieren manejar su vehículo, están teniendo la necesidad de un servicio que les de la confianza al precio justo de llevarlos de un Punto de Origen a un Punto de Destino. Por lo anterior mencionados, decidimos crear una App, en la cual las personas de Nicaragua puedan descargarla y solicitar sus viajes privados teniendo la confianza que los llevara un conductor calificado de manera segura, además a un precio justo.

# Modelo de Negocio

La App TELLEVO, es considerada únicamente una plataforma en la cual conecta los clientes con los drivers para realizar el recorrido del cliente, no es una empresa de transporte.

Los Drivers (conductores) son contratados bajo el contrato de “Servicio Profesional”, por lo cual no son meramente trabajadores de la App. Teniendo en cuenta que estan bajo contrato de Servicio Profesional, TELLEVO no esta en la obligación de pagar las prestaciones sociales como un trabajador, ya que estos solo prestan un servicio como contratista independiente, ganando un % por cada viaje recorrido, en dependencia de la distancia y tarifa que tenga la ciudad en que se realice el viaje.

La App TELLEVO, se encarga de generar la demanda de transporte, a través de la captación de nuevos clientes que requieran el servicio. Cada vez que un cliente realice un viaje, la aplicación dividirá los gastos, pago de Driver, lo necesario para cubrir los costos y el restante que serian las ganancias para la empresa.

El servicio se encuentra activo en 5 ciudades: Managua, Masaya, Leon, Chinandega y Granada. La tarifa varia en dependencia la demanda de recorridos que hay en cada ciudad.

## Introducción al Proyecto en MySQL

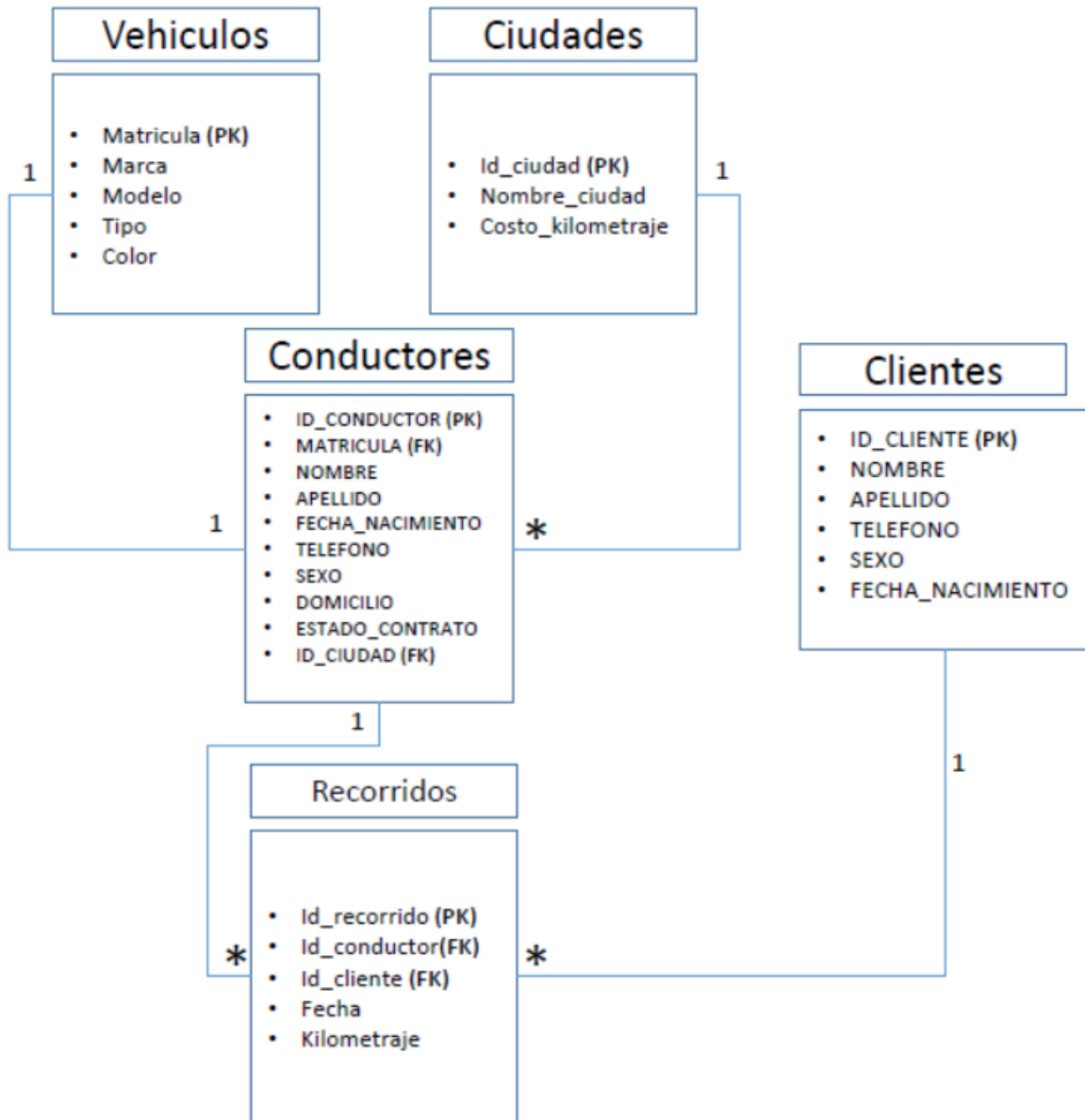
Se crea una base de datos para el fácil manejo de la información en la cual incluimos; información de los vehículos, conductor y clientes. Las ciudades donde ha sido utilizada la app y el registro de cada recorrido con toda la información desde la fecha, kilometraje recorrido, conductor y cliente que lo solicito.

Tener en cuenta que se les solicita a los conductores no utilizar más de 1 vehículo.

## Temas Desarrollados

1. Creación de Diagrama DER.
2. Descripción de tablas.
3. Creación de tablas.
4. Inserción de Datos.
5. Creación de Vistas.
6. Creación de Funciones.
7. Creación de Triggers.
8. Creación de StoredProcedures.
9. Prácticas de sentencias TCL como Rollback y Commit.
10. Prácticas de creación de Usuarios y Permisos.

# 1. Diagrama DER



## 2. Descripción de Tablas

### Vehículos

Tabla	Vehiculos						
Descripción	Contiene informacion sobre los vehiculos que manejan cada conductor						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	MATRICULA	INT		TRUE	TRUE		NUMERO DE MATRICULA UNICA EN CADA VEHICULO REGISTRADO
	MARCA	VARCHAR	50	TRUE			MARCA DEL VEHICULO EJEMPLO: TOYOTA, NISSAN, HYUNDAI
	MODELO	VARCHAR	50	TRUE			MODELO DEL VEHICULO EJEMPLO: TOYOTA COROLLA, HYUNDAI i10
	TIPO	VARCHAR	50				TIPO DE VEHICULO QUE CONDUCE EL DRIVER: SEDAN O CAMIONETA
	COLOR	VARCHAR	50	TRUE			COLOR DEL VEHICULO QUE CONDUCE EL DRIVER: AZUL, NEGRO, BLANCO

### Ciudades

Tabla	Ciudades						
Descripción	Contiene el nombre de cada ciudad donde operan los drivers y el costo x kilometraje.						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	ID_CIUADAD	INT		TRUE	TRUE		ID PARA IDENTIFICAR CADA CIUDAD DONDE OPERA EL DRIVER
	NOMBRE_CIUADAD	VARCHAR	50	TRUE			NOMBRE DE LA CIUDAD DE OPERA EL DRIVER
	COSTO_KILOMETRAJE	INT		TRUE			COSTO DE KILOMETRAJE PARA CADA CIUDAD

### Conductores

Tabla	Conductores						
Descripción	Contiene informacion sobre cada conductor que esta activo o inactivo.						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	ID_CONDUCTOR	INT		TRUE	TRUE		NUMERO DE TRABAJADOR QUE PORTA CADA DRIVER
	MATRICULA	INT		TRUE			NUMERO DE MATRICULA UNICA EN CADA VEHICULO REGISTRADO
FK	NOMBRE	VARCHAR	50	TRUE			NOMBRE DE CADA DRIVER
	APELLIDO	VARCHAR	50	TRUE			APELLIDOS DE CADA DRIVER
	FECHA_NACIMIENTO	DATE		TRUE			FECHA DE NACIMIENTO DEL DRIVER
	TELEFONO	VARCHAR	20				TELEFONO DE CONTACTO DE CADA DRIVER
	SEXO	VARCHAR	30	TRUE			GENERO DE CADA DRIVER
	DOMICILIO	VARCHAR	100				DIRECCION DONDE VIVE EL DRIVER
	ESTADO_CONTRATO	VARCHAR	30	TRUE			ESTADO DEL CONTRATO DE TRABAJO: ACTIVO O INACTIVO
	ID_CIUADAD	INT		TRUE	TRUE		ID PARA IDENTIFICAR CADA CIUDAD DONDE OPERA EL DRIVER

### Clientes

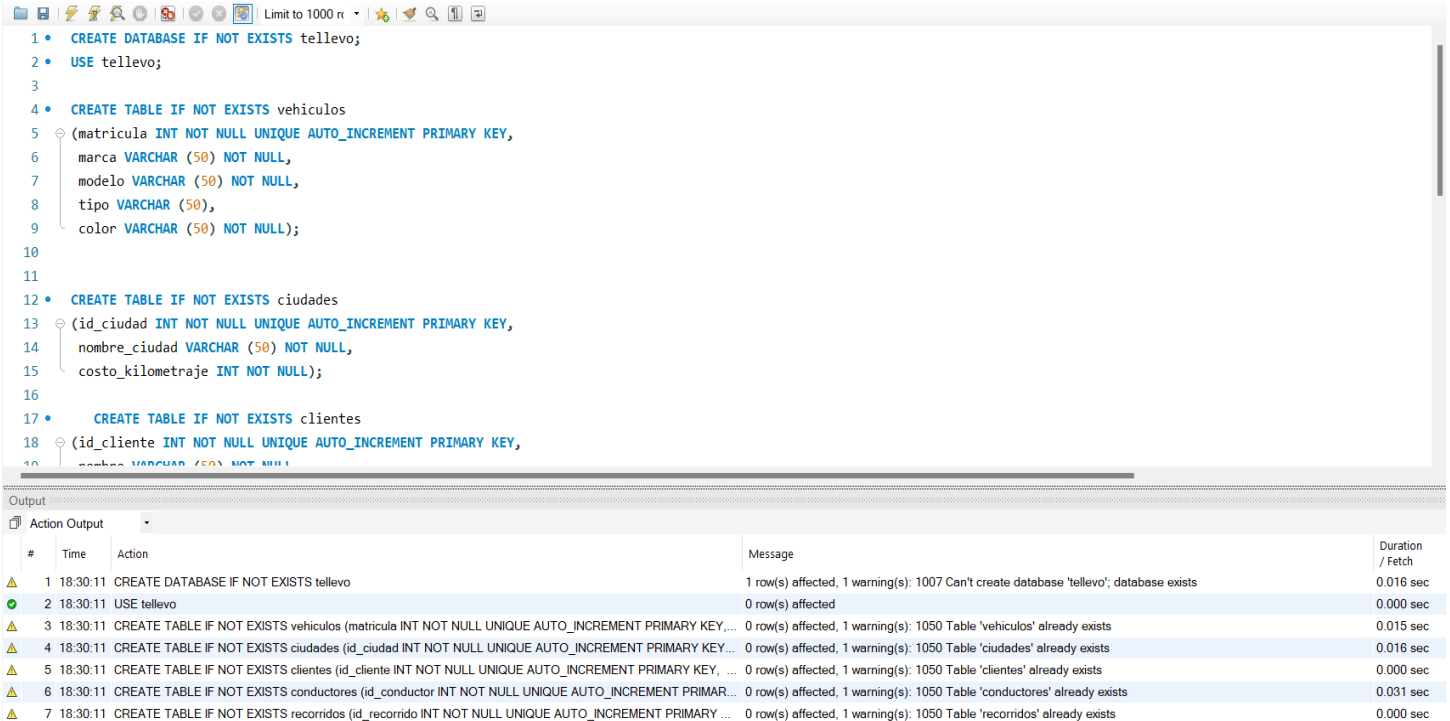
Tabla	Clientes						
Descripción	Contiene informacion de los clientes que se registran en la APP.						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	ID_CLIENTE	INT		TRUE	TRUE		NUMERO DE USUARIO REGISTRADO EN LA APP
	NOMBRE	VARCHAR	50	TRUE			NOMBRE DE CADA USUARIO
	APELLIDO	VARCHAR	50	TRUE			APELLIDO DE CADA USUARIO
	TELEFONO	VARCHAR	20				NUMERO DE CONTACTO DEL USUARIO.
	SEXO	VARCHAR	30	TRUE			GENERO DE CADA CLIENTE
	FECHA_NACIMIENTO	DATE		TRUE			FECHA DE NACIMIENTO DEL CLIENTE

### Recorridos

Tabla	Recorridos						
Descripción	Contiene informacion de todos los recorridos realizados por cada cliente y driver.						
KEY	COLUMN	TYPE	LENGHT	NOT NULL	UNIQUE	DEFAULT	NOTES
PK	ID_RECORRIDO	INT		TRUE	TRUE		NUMERO IDENTIFICADOR DE RECORRIDO
FK	ID_CONDUCTOR	INT		TRUE			NUMERO DE TRABAJADOR QUE PORTA CADA DRIVER
FK	ID_CLIENTE	INT		TRUE			NUMERO DE USUARIO REGISTRADO EN LA APP
	FECHA	DATE		TRUE		CURRENT_DATE	FECHA DEL RECORRIDO
	KILOMETRAJE	INT		TRUE			KILOMETRAJE RECORRIDO POR EL DRIVER EN CADA VIAJE

### 3. Creación de Tablas

La parte de la creación de tablas va en archivo SQL. Pero se adjuntan los resultados al momento de correr los comandos (ya se encontraban creadas las tablas al momento de esta captura).

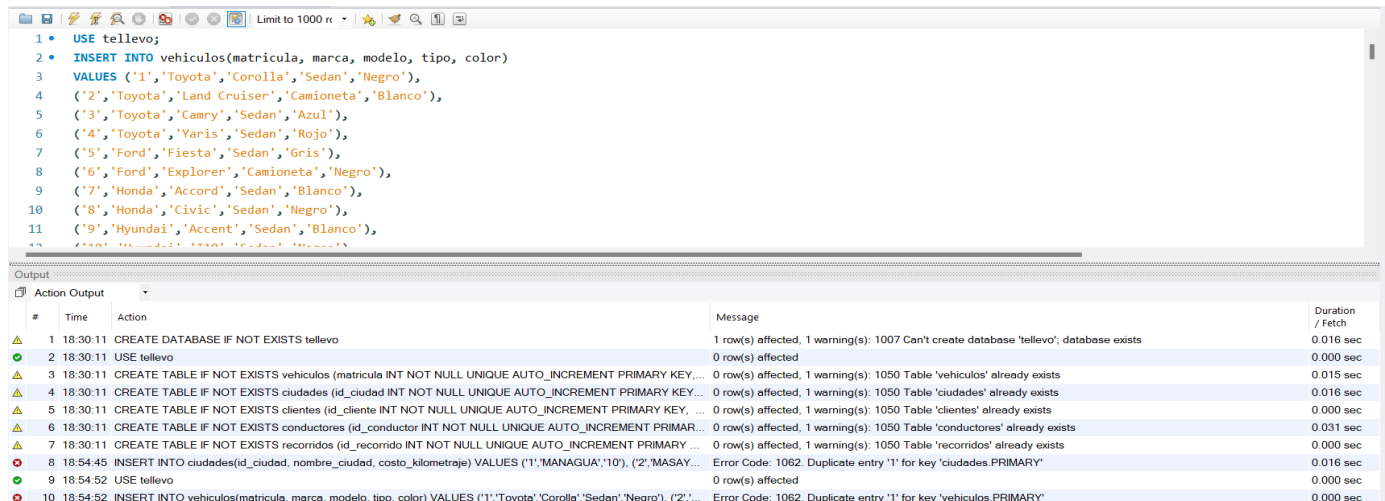


```
1 • CREATE DATABASE IF NOT EXISTS tellevoy;
2 • USE tellevoy;
3
4 • CREATE TABLE IF NOT EXISTS vehiculos
5   (matricula INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,
6    marca VARCHAR (50) NOT NULL,
7    modelo VARCHAR (50) NOT NULL,
8    tipo VARCHAR (50),
9    color VARCHAR (50) NOT NULL);
10
11
12 • CREATE TABLE IF NOT EXISTS ciudades
13   (id_ciudad INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,
14    nombre_ciudad VARCHAR (50) NOT NULL,
15    costo_kilometraje INT NOT NULL);
16
17 • CREATE TABLE IF NOT EXISTS clientes
18   (id_cliente INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,
19    nombre VARCHAR (50) NOT NULL);
```

#	Time	Action	Message	Duration / Fetch
1	18:30:11	CREATE DATABASE IF NOT EXISTS tellevoy	1 row(s) affected, 1 warning(s): 1007 Can't create database 'tellevoy'; database exists	0.016 sec
2	18:30:11	USE tellevoy	0 row(s) affected	0.000 sec
3	18:30:11	CREATE TABLE IF NOT EXISTS vehiculos (matricula INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,...	0 row(s) affected, 1 warning(s): 1050 Table 'vehiculos' already exists	0.015 sec
4	18:30:11	CREATE TABLE IF NOT EXISTS ciudades (id_ciudad INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected, 1 warning(s): 1050 Table 'ciudades' already exists	0.016 sec
5	18:30:11	CREATE TABLE IF NOT EXISTS clientes (id_cliente INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected, 1 warning(s): 1050 Table 'clientes' already exists	0.000 sec
6	18:30:11	CREATE TABLE IF NOT EXISTS conductores (id_conductor INT NOT NULL UNIQUE AUTO_INCREMENT PRIMAR...	0 row(s) affected, 1 warning(s): 1050 Table 'conductores' already exists	0.031 sec
7	18:30:11	CREATE TABLE IF NOT EXISTS recorridos (id_recorrido INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY ...	0 row(s) affected, 1 warning(s): 1050 Table 'recorridos' already exists	0.000 sec

### 4. Inserción de Datos.

La parte de la inserción de datos va en archivo SQL. Pero se adjuntan los resultados al momento de correr los comandos (ya se encontraban ingresado los valores al momento de esta captura y por eso me da el error de la Primary Key Duplicada).



```
1 • USE tellevoy;
2 • INSERT INTO vehiculos(matricula, marca, modelo, tipo, color)
3   VALUES ('1','Toyota','Corolla','Sedan','Negro'),
4   ('2','Toyota','Land Cruiser','Camioneta','Blanco'),
5   ('3','Toyota','Camry','Sedan','Azul'),
6   ('4','Toyota','Yaris','Sedan','Rojo'),
7   ('5','Ford','Fiesta','Sedan','Gris'),
8   ('6','Ford','Explorer','Camioneta','Negro'),
9   ('7','Honda','Accord','Sedan','Blanco'),
10  ('8','Honda','Civic','Sedan','Negro'),
11  ('9','Hyundai','Accent','Sedan','Blanco'),
12  ('10','Hyundai','Tucson','SUV','Negro');
```

#	Time	Action	Message	Duration / Fetch
1	18:30:11	CREATE DATABASE IF NOT EXISTS tellevoy	1 row(s) affected, 1 warning(s): 1007 Can't create database 'tellevoy'; database exists	0.016 sec
2	18:30:11	USE tellevoy	0 row(s) affected	0.000 sec
3	18:30:11	CREATE TABLE IF NOT EXISTS vehiculos (matricula INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,...	0 row(s) affected, 1 warning(s): 1050 Table 'vehiculos' already exists	0.015 sec
4	18:30:11	CREATE TABLE IF NOT EXISTS ciudades (id_ciudad INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected, 1 warning(s): 1050 Table 'ciudades' already exists	0.016 sec
5	18:30:11	CREATE TABLE IF NOT EXISTS clientes (id_cliente INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected, 1 warning(s): 1050 Table 'clientes' already exists	0.000 sec
6	18:30:11	CREATE TABLE IF NOT EXISTS conductores (id_conductor INT NOT NULL UNIQUE AUTO_INCREMENT PRIMAR...	0 row(s) affected, 1 warning(s): 1050 Table 'conductores' already exists	0.031 sec
7	18:30:11	CREATE TABLE IF NOT EXISTS recorridos (id_recorrido INT NOT NULL UNIQUE AUTO_INCREMENT PRIMARY ...	0 row(s) affected, 1 warning(s): 1050 Table 'recorridos' already exists	0.000 sec
8	18:54:45	INSERT INTO ciudades(id_ciudad, nombre_ciudad, costo_kilometraje) VALUES ('1','MANAGUA','10'), ('2','MASAY...	Error Code: 1062. Duplicate entry '1' for key 'ciudades.PRIMARY'	0.016 sec
9	18:54:52	USE tellevoy	0 row(s) affected	0.000 sec
10	18:54:52	INSERT INTO vehiculos(matricula, marca, modelo, tipo, color) VALUES ('1','Toyota','Corolla','Sedan','Negro'), ('2','...	Error Code: 1062. Duplicate entry '1' for key 'vehiculos.PRIMARY'	0.000 sec

## 5. Creación de Vistas

- Costo de KM Tabla Conductores
  - Esta vista se crea para poder añadir el costo de cada Kilometraje en dependencia de la ciudad del conductor (según tarifa). Podemos observar Id\_conductor, Id\_ciudad y el costo x km.

```

4 • USE tellevo;
5
6
7 /* AGREGANDO EL COSTO DEL KM A LA TABLA DE CONDUCTORES*/
8 • CREATE OR REPLACE VIEW Costo_km_conductores AS
9   SELECT
10     co.id_conductor,
11     co.id_ciudad,
12     ci.costo_kilometraje
13   FROM conductores AS co
14   JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad;

```

id_conductor	id_ciudad	costo_kilometraje
34	1	10
35	1	10
36	1	10
37	1	10
38	1	10
55	1	10
2	2	6
14	2	6
15	2	6
16	2	6
17	2	6
18	2	6
19	2	6
46	2	6

- Tabla de recorridos con su costo x kilometraje
  - En esta vista podemos observar cada recorrido realizado, con su fecha, el total de kilometraje y el costo x cada km recorrido.

```

/* TABLA DE RECORRIDOS CON COSTO DE CADA KM RECORRIDO*/
• CREATE OR REPLACE VIEW Costo_recorrido AS
  SELECT
    r.id_recorrido,
    r.fecha,
    r.kilometraje,
    t.id_ciudad,
    t.costo_km
  FROM recorridos AS r
  JOIN (SELECT
        co.id_conductor,
        co.id_ciudad,
        ci.costo_kilometraje AS costo_km
      FROM conductores AS co
      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad) AS t ON r.id_conductor = t.id_conductor;

```

```
1 • SELECT * FROM tellevo.costo_recorrido;
```

id_recorrido	fecha	kilometraje	id_ciudad	costo_km
1	2022-08-02	10	1	10
51	2022-05-30	5	1	10
101	2022-05-20	9	1	10
103	2022-07-16	3	1	10
105	2022-06-14	12	1	10
107	2022-07-30	10	1	10
109	2022-05-27	4	1	10
111	2022-07-17	7	1	10
113	2022-06-02	8	1	10
381	2022-08-06	7	1	10
431	2022-06-27	2	1	10

- Tabla de recorridos con su costo total x recorrido
  - Esta vista se crea para sacar el costo total por recorrido, el id del cliente, la ciudad donde se realizó y el driver que realizo el recorrido.

```

• CREATE OR REPLACE VIEW Costo_total_id_recorrido AS
  SELECT
    r.id_recorrido,
    r.fecha,
    r.kilometraje,
    r.id_cliente,
    t.id_conductor,
    t.id_ciudad,
    t.costo_km,
    r.kilometraje * t.costo_km AS costo_total
  FROM recorridos AS r
  JOIN (SELECT
        co.id_conductor,
        co.id_ciudad,
        ci.costo_kilometraje AS costo_km
      FROM conductores AS co
      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad) AS t ON r.id_conductor = t.id_conductor
  ORDER BY 1,2;

```

```
1 • SELECT * FROM tellevo.costo_total_id_recorrido;
```

id_recorrido	fecha	kilometraje	id_cliente	id_conductor	id_ciudad	costo_km	costo_total
1	2022-08-02	10	1	1	1	10	100
2	2022-05-02	8	2	2	2	6	48
3	2022-05-30	2	3	3	3	4	8
4	2022-07-18	15	4	4	4	8	120
5	2022-05-17	11	5	5	5	4	44
6	2022-07-12	3	6	6	1	10	30
7	2022-06-26	9	7	7	1	10	90
8	2022-05-21	3	8	8	1	10	30
9	2022-06-21	4	9	9	1	10	40
10	2022-07-07	3	10	10	1	10	30
11	2022-06-26	3	11	11	1	10	30
12	2022-05-06	3	12	12	1	10	30
13	2022-05-24	3	13	13	1	10	30
14	2022-05-22	9	14	14	2	6	54
15	2022-06-10	3	15	15	2	6	18
16	2022-07-30	3	16	16	2	6	18
17	2022-07-12	9	17	17	2	6	54

- Tabla de información de drivers más la ciudad donde opera
  - Esta vista se crea para obtener la información de los drivers y en que ciudades operan.

```

57  /*QUERY DE INFO DRIVERS VEHICULOS CIUDAD Y NOMBRE COMPLETO*/
58  • CREATE OR REPLACE VIEW info_drivers_vehiculo_ciudad AS
59  SELECT
60      co.id_conductor,
61      co.id_ciudad,
62      co.nombre AS nombre,
63      co.apellido AS apellido,
64      CONCAT(co.nombre, ' ', co.apellido) AS nombre_completo,
65      ci.nombre_ciudad AS ciudad,
66      vh.marca AS marca_vehiculo,
67      vh.modelo AS modelo_vehiculo,
68      vh.color AS color_vehiculo
69  FROM conductores AS co
70  JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad
71  JOIN vehiculos AS vh ON co.matricula = vh.matricula
72  ORDER BY 1 ASC;

```

```
1 • SELECT * FROM tellevo.info_drivers_vehiculo_ciudad;
```

id_conductor	id_ciudad	nombre	apellido	nombre_completo	ciudad	marca_vehiculo	modelo_vehiculo	color_vehiculo
1	1	ANTONIO	SANCHEZ	ANTONIO SANCHEZ	MANAGUA	Toyota	Corola	Rojo
2	2	MANUEL	ADELA	MANUEL ADELA	MASAYA	Toyota	Land Cruiser	Rojo
3	3	JOSE	MARTINEZ	JOSE MARTINEZ	LEON	Hyundai	Accent	Rojo
4	4	FRANCISCO	CARRASCO	FRANCISCO CARRASCO	CHINANDEGA	Hyundai	I10	Blanco
5	5	DAVID	CUEVAS	DAVID CUEVAS	GRAMADA	Hyundai	Accent	Blanco
6	1	JUAN	FERRERO	JUAN FERRERO	MANAGUA	Hyundai	I10	Blanco
7	1	JOSE ANTONIO	MADERO	JOSE ANTONIO MADERO	MANAGUA	Hyundai	Accent	Blanco
8	1	JAVIER	LEON	JAVIER LEON	MANAGUA	Hyundai	I10	Blanco
9	1	DANIEL	HERNANDEZ	DANIEL HERNANDEZ	MANAGUA	Honda	Civic	Rojo
10	1	JOSE LUIS	MARTINEZ	JOSE LUIS MARTINEZ	MANAGUA	Toyota	Land Cruiser	Negro
11	1	FRANCISCO JA...	GARCIA	FRANCISCO JAVIER GAR...	MANAGUA	Honda	Accord	Negro
12	1	CARLOS	FERNANDEZ	CARLOS FERNANDEZ	MANAGUA	Toyota	Yaris	Azul

- Tabla de Ganancia Drivers 1era semana de Julio
  - Esta vista se crea para obtener la información sobre las ganancias que tuvieron los drivers en la primera semana de Julio, aplicando técnicas de CTE.

```

88  /* GANANCIAS DE DRIVERS EN LA PRIMERA SEMANA DE JULIO*/
89  • CREATE OR REPLACE VIEW ganancia_drivers_1rasemana_julio AS
90  WITH
91      info_recorridos AS
92      (SELECT
93          r.id_recorrido,
94          r.fecha AS fecha,
95          r.kilometraje AS kilometraje,
96          r.id_cliente,
97          t.id_conductor AS id_conductor,
98          t.id_ciudad,
99          t.costo_km,
100         r.kilometraje * t.costo_km AS costo_total
101      FROM recorridos AS r
102      JOIN (SELECT
103          co.id_conductor,
104          co.id_ciudad,
105          ci.costo_kilometraje AS costo_km
106      FROM conductores AS co
107      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad) AS t ON r.id_conductor = t.id_conductor),
108      drivers AS
109      (SELECT
110          co.id_conductor AS id_conductor,
111          co.id_ciudad,
112          co.nombre AS nombre,
113          co.apellido AS apellido,
114          CONCAT(co.nombre, ' ', co.apellido) AS nombre_completo,
115          ci.nombre_ciudad AS ciudad,
116          vh.marca AS marca_vehiculo,
117          vh.modelo AS modelo_vehiculo,
118          vh.color AS color_vehiculo
119      FROM conductores AS co
120      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad
121      JOIN vehiculos AS vh ON co.matricula = vh.matricula
122      ORDER BY 1 ASC)
123
124  SELECT
125      ir.fecha,
126      d.nombre_completo AS driver,
127      SUM(ir.kilometraje) AS kilometraje_recorrido,
128      SUM(ir.costo_total) AS costo_total,
129      (SUM(ir.costo_total) * 0.20) AS ganancia_driver
130  FROM info_recorridos AS ir
131  LEFT JOIN drivers AS d ON ir.id_conductor = d.id_conductor
132  WHERE fecha BETWEEN '2022-07-01' AND '2022-07-07'
133  GROUP BY 1,2
134  ORDER BY 5 DESC;
135
136

```

```
1 • SELECT * FROM tellevo.ganancia_drivers_1rasemana_julio;
```

fecha	driver	kilometraje_recorrido	costo_total	ganancia_driver
2022-07-04	FRANCISCO JAVIER GARCIA	13	130	26.00
2022-07-06	JUAN FERRERO	11	110	22.00
2022-07-03	MARIA ROSA ALVAREZ	10	100	20.00
2022-07-01	MARIA JOSE DIAZ	12	96	19.20
2022-07-02	JOSE ANTONIO MADERO	9	90	18.00
2022-07-01	ANTONIO SANCHEZ	9	90	18.00
2022-07-04	JAVIER LEON	8	80	16.00
2022-07-03	JUAN FERRERO	8	80	16.00
2022-07-02	JESUS PEREZ	8	80	16.00
2022-07-01	LAURA GONZALEZ	13	78	15.60
2022-07-06	JAVIER LEON	7	70	14.00
2022-07-07	ANTONIO SANCHEZ	7	70	14.00
2022-07-03	MIGUEL RUIZ	10	60	12.00
2022-07-02	ANA GOMEZ	10	60	12.00
2022-07-07	PEDRO GUTIERREZ	10	60	12.00
2022-07-03	MARIA JOSE DIAZ	7	56	11.20
2022-07-07	JOSE ANTONIO MADERO	5	50	10.00
2022-07-01	CARLOS FERNANDEZ	5	50	10.00
2022-07-07	LUCIA GUTIERREZ	6	48	9.60
2022-07-02	MIGUEL RUIZ	8	48	9.60
2022-07-03	ANTONIA MARIN	6	48	9.60
2022-07-06	CRISTINA RODRIGUEZ	11	44	8.80
2022-07-03	MARIA DOLORES SAENZ	7	42	8.40
2022-07-01	MARIA ROSA ALVAREZ	4	40	8.00
2022-07-07	ANA BELEN SANTAMARIA	10	40	8.00
2022-07-06	MARIA VICTORIA IBANEZ	8	32	6.40
2022-07-04	JUAN JOSE CALVO	8	32	6.40
2022-07-02	MARIA ELENA BENITO	8	32	6.40
2022-07-07	JOSE LUIS MARTINEZ	3	30	6.00
2022-07-04	JOSE MARTINEZ	7	28	5.60
2022-07-07	LORENA MUÑOZ	7	28	5.60



- Tabla de TOP Clientes con mas KM recorrido.
  - Esta vista se crea para obtener la información sobre los clientes con más kilómetros recorridos y cantidad de recorridos, aplicando técnicas de CTE.

```

139  /* TOP 10 CLIENTES CON MAS KM RECORRIDOS*/
140  • CREATE OR REPLACE VIEW top10_clientes AS
141  WITH
142  info_recorridos AS
143  (SELECT
144  r.id_recorrido,
145  r.fecha AS fecha,
146  r.kilometraje AS kilometraje,
147  r.id_cliente,
148  t.id_conductor AS id_conductor,
149  t.id_ciudad,
150  t.costo_km,
151  r.kilometraje * t.costo_km AS costo_total
152  FROM recorridos AS r
153  JOIN (SELECT
154  co.id_conductor,
155  co.id_ciudad,
156  ci.costo_kilometraje AS costo_km
157  FROM conductores AS co
158  JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad) AS t ON r.id_conductor = t.id_conductor),
159
160  clientes_recorrido AS
161  (SELECT
162  id_cliente,
163  CONCAT(nombre, ' ', apellido) AS nombre_cliente,
164  sexo,
165  fecha_nacimiento,
166  year(current_date()) - YEAR(fecha_nacimiento) AS edad
167  FROM clientes
168  ORDER BY 1)
169
170
171  SELECT
172  cr.nombre_cliente AS cliente,
173  cr.sexo,
174  cr.edad,
175  SUM(ir.kilometraje) AS kilometraje_recorrido,
176  SUM(ir.costo_total) AS costo_total,
177  COUNT(ir.id_recorrido) AS total_recorridos
178  FROM info_recorridos AS ir
179  LEFT JOIN clientes_recorrido AS cr ON ir.id_cliente = cr.id_cliente
180  WHERE fecha >= '2022-07-01'
181  GROUP BY 1,2
182  ORDER BY 4 DESC
183  LIMIT 10;
184
185

```

cliente	sexo	edad	kilometraje_recorrido	costo_total	total_recorridos
FRANKLIN LARA	MASCULINO	30	52	284	9
MIGUEL BLANDON	MASCULINO	25	48	206	7
MARTA AGUILERA MERINO	FEMENINO	26	45	266	8
KRISTABELL MENDIETA	MASCULINO	26	43	280	5
CARLA BOIX GONZÁLEZ	FEMENINO	22	43	270	8
AINA AROCA GÓMEZ	FEMENINO	25	37	240	5
JORGE NUÑEZ	MASCULINO	27	36	186	4
KEVIN CASTILLO	MASCULINO	24	35	176	5
TEDDY CUADRA	MASCULINO	37	35	260	5
FERNANDO LOPEZ	MASCULINO	30	35	286	5

## 6. Creación de Funciones

- Función Calculo de Viajes.
  - Se crea esta función para realizar un ejercicio, de cuantos viajes se necesita realizar según la capacidad de cada vehículo.

```

1  /*FUNCION PARA OBTENER LA CANTIDAD DE VIAJES SEGUN LA
2  CAPACIDAD DEL VEHICULO Y LA CANTIDAD DE PASAJEROS*/
3  /* SE QUIERE SABER LA CANTIDAD DE VIAJES O VEHICULOS NECESARIOS
4  PARA LLEVAR 100 PASAJEROS, CADA VEHICULO TIENE UNA CAPACIDAD MAXIMA DE 4 PERSONAS */
5
6  • USE `tellev0`;
7  • DROP function IF EXISTS `calculo_viajes`;
8
9  DELIMITER $$
10 • USE `tellev0`$$
11 • CREATE FUNCTION `calculo_viajes` (capacidad INT, pasajeros INT)
12 RETURNS INTEGER
13 DETERMINISTIC
14 BEGIN
15 DECLARE resultado INT;
16 SET resultado = pasajeros / capacidad;
17 RETURN resultado;
18 END$$
19
20 DELIMITER ;
21
22 • /*SELECT `calculo_viajes` (4,100)*/
23
24

```

1 • SELECT `calculo_viajes` (4,100)		
Result Grid		
Filter Rows:		
Export:		
Wrap Cell Content:		
<table> <tr> <th>calculo_viajes` (4,100)</th></tr> <tr> <td>25</td></tr> </table>	calculo_viajes` (4,100)	25
calculo_viajes` (4,100)		
25		



- Función Driver Info

- Se crea esta función para buscar el nombre y apellido de cada conductor según su id\_conductor.

```

25  /* FUNCION PARA OBTENER EL NOMBRE Y APELLIDO DEL DRIVER SEGUN SU DRIVER ID*/
26
27  USE `tellev0`;
28  DROP function IF EXISTS `driver_info`;
29
30  USE `tellev0`;
31  DROP function IF EXISTS `tellev0`.`driver_info`;
32  ;
33
34  DELIMITER $$
35  USE `tellev0` $$
36  CREATE DEFINER=`root`@`localhost` FUNCTION `driver_info`(id INT) RETURNS varchar(100) CHARSET utf8mb4
37  READS SQL DATA
38  BEGIN
39      DECLARE info VARCHAR(100);
40      SET info = '';
41      SELECT
42          CONCAT(co.nombre, ' ', co.apellido) AS nombre_completo INTO info
43      FROM conductores AS co
44      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad
45      JOIN vehiculos AS vh ON co.matricula = vh.matricula
46      WHERE co.id_conductor = id;
47      RETURN info;
48  END$$
49
50  DELIMITER ;
51  ;
52
53  /*SELECT `driver_info`(8);*/

```

The screenshot shows a SQL client window with a toolbar at the top. The query editor contains the command: `1 • SELECT `driver_info`(8);`. Below the editor, the 'Result Grid' tab is active, displaying the result of the query in a table with one row: `JAVIER LEON`.

Result Grid
<code>`driver_info`(8)</code>
JAVIER LEON

- Función km recorridos x Driver

- Se crea esta función para obtener el total de kilómetros recorridos de cada conductor según su id\_conductor.

```

59  /*FUNCION PARA DETERMINAR EL TOTAL DE KM RECORRIDO X DRIVER SEGUN SU DRIVER ID*/
60
61  USE `tellev0`;
62  DROP function IF EXISTS `total_km_recorrido_driver`;
63
64  USE `tellev0`;
65  DROP function IF EXISTS `tellev0`.`total_km_recorrido_driver`;
66  ;
67
68  DELIMITER $$
69  USE `tellev0` $$
70  CREATE DEFINER=`root`@`localhost` FUNCTION `total_km_recorrido_driver`(id_km INT) RETURNS varchar(50) CHARSET utf8mb4
71  READS SQL DATA
72  BEGIN
73      DECLARE km VARCHAR(50);
74      SET km = '';
75      SELECT
76          SUM(r.kilometraje) INTO km
77      FROM recorridos AS r
78      JOIN (SELECT
79          co.id_conductor,
80          co.id_ciudad,
81          ci.costo_kilometraje AS costo_km
82      FROM conductores AS co
83      JOIN ciudades AS ci ON co.id_ciudad = ci.id_ciudad) AS t ON r.id_conductor = t.id_conductor
84      WHERE t.id_conductor = id_km;
85      RETURN km;
86  END$$
87
88  DELIMITER ;
89  ;
90
91  /*SELECT `total_km_recorrido_driver`(2);*/

```

The screenshot shows a SQL client window with a toolbar at the top. The query editor contains the command: `1 • SELECT `total_km_recorrido_driver`(2)`. Below the editor, the 'Result Grid' tab is active, displaying the result of the query in a table with one row: `192`.

Result Grid
<code>`total_km_recorrido_driver` (2)</code>
192

## 7. Creación de Triggers

- Trigger Registro ingreso de conductor.
  - Se crea este trigger para que luego que después de cada inserción de un nuevo conductor, nos registre en una tabla la fecha en que fue ingresado al sistema.

```
1  /*TRIGGERS PARA LAS TABLAS "CONDUCTORES"*/
2
3  /*trigger para registro de los conductores, cada vez que ingresa un conductor nuevo al sistema y la fecha en que fue ingresado*/
4  DELIMITER //
5  • CREATE TRIGGER tr_registro_conductor
6  AFTER INSERT ON conductores
7  FOR EACH ROW
8  INSERT INTO registro_conductores (id_conductor, nombre, apellido, fecha_registro)
9  VALUES (NEW.id_conductor, NEW.nombre, NEW.apellido, current_date());
10
11 END;//
12
```

1 • SELECT \* FROM tellevo.registro\_conductores;

id_conductor	nombre	apellido	fecha_registro
55	ANTONIO	SANCHEZ	2022-09-09
56	ANTONIO	SANCHEZ	2022-09-09
60	MANUEL	SANCHEZ	2022-10-13
NULL	NULL	NULL	NULL

- Trigger Bono Conductor
  - Se crea este trigger ya que actualmente se está aplicando un bono a su pago a los conductores de ciertas ciudades, si aplica al bono entonces se registra en una tabla donde están los conductores que aplican para luego pagarse.

```
15  /*trigger para ver antes de que se ingrese un conductor nuevo a la tabla de conductores, si aplica el bono de su ciudad*/
16
17  DELIMITER //
18  • CREATE TRIGGER tr_before_bonus_conductor
19  BEFORE INSERT ON conductores
20  FOR EACH ROW BEGIN
21  /* Bono para los conductores nuevos de estas 2 ciudades*/
22  IF (new.id_ciudad = 5)
23  THEN
24      INSERT INTO city_conductores_bonus (id_conductor, id_ciudad, bono) VALUES (NEW.id_conductor, NEW.id_ciudad, 1000);
25  ELSEIF
26      (new.id_ciudad = 3)
27  THEN
28      INSERT INTO city_conductores_bonus (id_conductor, id_ciudad, bono) VALUES (NEW.id_conductor, NEW.id_ciudad, 2000);
29  END IF;
30  END;//
31
32  /*PRUEBA DEL TRIGGER
33  INSERT INTO conductores (id_conductor, matricula, nombre, apellido, fecha_nacimiento, telefono, sexo, domicilio, estado_contrato, id_ciudad
34  )
35  VALUES ('60','40','MANUEL','SANCHEZ','1990-04-26','52-3519685','MASCULINO','Estatua de Montoya 2C. Sur 1 1/2 Oeste','ACTIVO','5');*/
~
1
2  • INSERT INTO conductores (id_conductor, matricula, nombre, apellido, fecha_nacimiento, telefono, sexo, domicilio, estado_contrato, id_ciudad
3  )
4  VALUES ('60','40','MANUEL','SANCHEZ','1990-04-26','52-3519685','MASCULINO','Estatua de Montoya 2C. Sur 1 1/2 Oeste','ACTIVO','5');
```

1 • SELECT \* FROM tellevo.city\_conductores\_bonus;

id_conductor	id_ciudad	bono
56	5	1000
60	5	1000
NULL	NULL	NULL

- Trigger Registro ingreso de Recorridos.
  - Se crea este trigger para que luego que después de cada recorrido registrado, nos registre en una tabla la fecha y hora en que fue ingresado al sistema.

```

38  /*TRIGGERS PARA LAS TABLAS "RECORRIDOS"*/
39
40
41  /*trigger para llevar los registros de los recorridos, que fecha y a que hora ingresa 1 recorrido nuevo*/
42  DELIMITER //
43  • CREATE TRIGGER tr_registro_recorridos
44    AFTER INSERT ON recorridos
45    FOR EACH ROW
46    INSERT INTO registro_recorrido (id_recorrido, id_conductor, fechahora)
47    VALUES (NEW.id_recorrido, NEW.id_conductor, CURRENT_TIMESTAMP());
48
49    END;//

```

1 • SELECT \* FROM tellevo.registro\_recorrido;

	id_recorrido	id_conductor	fechahora
▶	1001	29	2022-09-09 12:17:14
	1002	29	2022-09-09 12:23:16
	1003	29	2022-09-09 12:25:13
	1004	25	2022-10-13 20:26:27
•	HULL	HULL	HULL

- Trigger descuento del 15% a los mejores clientes
  - Se crea este trigger ya que actualmente se está aplicando un descuento del 15% a los clientes que cuentan con 10 recorridos o más, se guardan en una tabla para luego aplicarles el descuento.

```

51  /* Trigger para agregar el respectivo descuento del 15% a los clientes que tienen mas de 10 recorridos*/
52  DELIMITER //
53  • CREATE TRIGGER tr_before_descuento_recorrido
54    BEFORE INSERT ON recorridos
55    FOR EACH ROW BEGIN
56    /* Bono para los clientes que tengan mas de 10 recorridos*/
57    IF (new.id_cliente IN (SELECT id_cliente
58                          FROM tellevo.recorridos
59                          GROUP BY 1
60                          HAVING COUNT(distinct id_recorrido) > 10))
61    THEN
62      INSERT INTO descuento_recorrido (id_recorrido, id_cliente, descuento_porcentual) VALUES (NEW.id_recorrido, NEW.id_cliente, 15);
63    END IF;
64  END;//
65
66  /* PRUEBA DEL TRIGGER
67  INSERT INTO recorridos (id_recorrido, id_conductor, id_cliente, fecha, kilometraje)
68  VALUES ('1004', '25', '55', '2022-10-13', '5.87')
69  ;*/

```

1 • SELECT \* FROM tellevo.descuento\_recorrido;

	id_recorrido	id_cliente	descuento_porcentual
▶	1003	55	15
	1004	55	15
•	HULL	HULL	HULL

## 8. Creación de StoredProcedures

- Stored Procedures Info Clientes.
  - Se crea este stored procedures para poder ordenar ya se descendente o ascendente, en base a una columna que escojamos de la tabla clientes. Cuenta con 2 parametros.

```
1  /*ESTE SP CARGA LA TABLA DE CLIENTES, CONTIENE 2 PARAMETROS EL PRIMERO PONE ENTRE '' LA COLUMNA QUE DESEA ORDENAR
2  Y EL SEGUNDO PONE ENTRE '' DE QUE MANERA SI 'ASC' O 'DESC'*/
3  • USE `tellev0`;
4  • DROP procedure IF EXISTS `lista_clientes`;
5
6  • USE `tellev0`;
7  • DROP procedure IF EXISTS `tellev0`.`lista_clientes`;
8  ;
9
10 DELIMITER $$
11 • USE `tellev0`$$
12 • CREATE DEFINER=`root`@`localhost` PROCEDURE `lista_clientes`(IN campo CHAR(50), orden CHAR(30))
13 BEGIN
14     SELECT
15         id_cliente,
16         nombre,
17         apellido
18     FROM tellev0.clientes
19     ORDER BY
20         CASE WHEN campo = 'id_cliente' AND orden = 'ASC' THEN id_cliente END ASC,
21         CASE WHEN campo = 'id_cliente' AND orden = 'DESC' THEN id_cliente END DESC,
22         CASE WHEN campo = 'nombre' AND orden = 'ASC' THEN nombre END ASC,
23         CASE WHEN campo = 'nombre' AND orden = 'DESC' THEN nombre END DESC,
24         CASE WHEN campo = 'apellido' AND orden = 'ASC' THEN apellido END ASC,
25         CASE WHEN campo = 'apellido' AND orden = 'DESC' THEN apellido END DESC;
26
27 END$$
28
29 DELIMITER ;
30 ;
31
32 /* REALIZAMOS ESTE CALL PARA LLAMAR LA COLUMNA "APELLIDO" Y ORDENARLA DE MANERA ASC
33 call `lista_clientes` ('apellido','ASC');*/
```

```
1 • call `lista_clientes` ('apellido','ASC');
```

Result Grid			Filter Rows:	Exports	Wrap Cell Content:
id_cliente	nombre	apellido			
49	IVET	ABADIAS MASANA			
64	RAFAEL	AGUILAR			
23	MARTA	AGUILAR RAMOS			
20	JORDINA	AGUILAR RODRIGUEZ			
30	MARTA	AGUILAR SUNYÉ			
55	MARTA	AGUILERA MERINO			
44	MIREIA	AGUILERA PRAT			
33	LAURA	AGUILERA TATJÉ			
144	NORMAN	AGUIRRE			
16	GEMMA	ALAVEDRA SUNYÉ			
43	MARTA	ALCAIDE MOLINA			
96	JOAQUIN	ALFARO			
17	MARIA L...	ALIGUÉ BONVEHÍ			
28	MARIONA	ALIGUÉ RIVERA			
26	CRISTINA	ALINS GONZÁLEZ			
37	INGRID	ALOY FARRANDO			
163	ANTONY	ALTAMIRANO			
19	SANDRA	ALTIMIRAS ARMENTE...			
39	GEMMA	ALTIMIRAS SERAROLS			
126	HENRY	ALVAREZ			
142	LENIN	ALVAREZ			
41	VIRGINIA	ALVAREZ ARMENTER...			
53	LUCIA	ALVAREZ DOMENECH			
51	IRENE	ALVAREZ PARCERISA			
25	MARIA N...	ALVAREZ TROYANO			
109	RIDER	ARAUZ			
91	CRISTIAN	ARAUZ			
92	JORGE	ARBUROLA			
145	AYRAM	ARCE			
114	OSCAR	AREVALO			
50	JÚLIA	AREVALO SANCHEZ			

- Stored Insertar o Eliminar Filas.
  - Se crea este stored procedures que nos ayuda a insertar o eliminar filas en la tabla de Vehiculos, dependiendo del “call” que realicemos.

Insertando el dato.

```

36  /*STORED PROCEDURE QUE NOS AYUDA A INSERTAR Y ELIMINAR FILAS EN LA TABLA DE "VEHICULOS" */
37  •  USE `tellevo`;
38  •  DROP procedure IF EXISTS `insert_delete_vehiculos`;
39
40  DELIMITER $$
41  •  USE `tellevo`$$
42  •  CREATE PROCEDURE `insert_delete_vehiculos` ( IN
43      accion VARCHAR(30)
44      ,spmatricula INT
45      ,spmarca VARCHAR(50)
46      ,spmodelo VARCHAR(50)
47      ,sptipo VARCHAR(50)
48      ,spcolor VARCHAR(50)
49  )
50  BEGIN
51      -- INSERT
52      IF accion = "INSERT" THEN
53          INSERT INTO vehiculos(matricula ,marca ,modelo ,tipo ,color )
54          VALUES (spmatricula ,spmarca ,spmodelo ,sptipo ,spcolor );
55      END IF;
56
57      -- DELETE
58      IF accion ="DELETE" THEN
59          DELETE FROM vehiculos
60          WHERE matricula = spmatricula;
61      END IF;
62  END$$
63
64  DELIMITER ;

```

```
19 • call insert_delete_vehiculos ("INSERT",51,"Toyota","Hilux","Camioneta","Blanca");
```

Output			
Action Output			
#	Time	Action	Message
37	20:26:27	INSERT INTO recorridos (id_recorrido, id_conductor, id_cliente, fecha, kilometraje) VALUES ('1004','25','55',...	1 row(s) affected
38	20:26:31	SELECT * FROM tellevo.registro_recorrido LIMIT 0, 1000	4 row(s) returned
39	20:26:34	SELECT * FROM tellevo.descuento_recorrido LIMIT 0, 1000	2 row(s) returned
40	20:33:57	call 'lista_clientes' ('apellido','ASC')	250 row(s) returned
41	20:36:07	SELECT * FROM tellevo.vehiculos LIMIT 0, 1000	50 row(s) returned
42	20:38:11	call insert_delete_vehiculos ("INSERT",51,"Toyota","Hilux","Camioneta","Blanca")	1 row(s) affected

```

1 • SELECT * FROM tellevo.vehiculos
2 WHERE matricula = 51;

```

matricula	marca	modelo	tipo	color
51	Toyota	Hilux	Camioneta	Blanca
* NULL	NULL	NULL	NULL	NULL

Eliminando el registro.

```
19 • call insert_delete_vehiculos ("DELETE",51,"","","","");
```

Output			
Action Output			
#	Time	Action	Message
38	20:26:31	SELECT * FROM tellevo.registro_recorrido LIMIT 0, 1000	4 row(s) returned
39	20:26:34	SELECT * FROM tellevo.descuento_recorrido LIMIT 0, 1000	2 row(s) returned
40	20:33:57	call 'lista_clientes' ('apellido','ASC')	250 row(s) returned
41	20:36:07	SELECT * FROM tellevo.vehiculos LIMIT 0, 1000	50 row(s) returned
42	20:38:11	call insert_delete_vehiculos ("INSERT",51,"Toyota","Hilux","Camioneta","Blanca")	1 row(s) affected
43	20:38:19	SELECT * FROM tellevo.vehiculos LIMIT 0, 1000	51 row(s) returned
44	20:38:41	SELECT * FROM tellevo.vehiculos WHERE matricula = 51 LIMIT 0, 1000	1 row(s) returned
45	20:42:28	call insert_delete_vehiculos ("DELETE",51,"","","","")	1 row(s) affected
46	20:42:33	SELECT * FROM tellevo.vehiculos WHERE matricula = 51 LIMIT 0, 1000	0 row(s) returned

```

1 • SELECT * FROM tellevo.vehiculos
2 WHERE matricula = 51;

```

matricula	marca	modelo	tipo	color
* NULL	NULL	NULL	NULL	NULL

## 9. Practicas de Sentencias TCL Rollback y Commit

- Rollback y Commit.
  - En la primera imagen se puede aplicar un rollback y commit, eliminando los registros del 997 al 1000. En la segunda imagen se practica la sentencia “savepoint” y luego rollback directo al “savepoint” del primer\_lote.

```
272  -- SENTENCIAS ROLLBACK Y COMMIT
273
274
275  /* Eliminando los registros del 1000 al 997 en la tabla de recorridos*/
276
277  START TRANSACTION;
278 • DELETE FROM tellevo.recorridos
279 WHERE
280 id_recorrido IN (1000,999,998,997);
281
282  -- ROLLBACK;
283  -- COMMIT;
284  /* DEJO COMENTADO LOS VALORES ELIMINADOS*/
285  /* INSERT INTO recorridos (id_recorrido, id_conductor, id_cliente, fecha, kilometraje)
286  VALUES ('997','26','112','2022-07-01','12.48'),
287  ('998','27','113','2022-06-15','6.68'),
288  ('999','28','114','2022-08-05','5.43'),
289  ('1000','29','115','2022-08-10','5.87')
290  ;*/

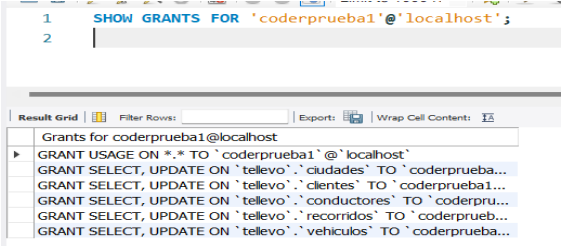
292  -- Inicio la transaccion para insertar 8 clientes nuevos en mi tabla de clientes
293 • START TRANSACTION;
294 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
295 VALUES ('251','MARTHA','RUIZ TABOADA','938265580','FEMENINO','1997-06-25');
296 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
297 VALUES ('252','CARLOS','CASTILLO MOLINA','938234580','MASCULINO','1993-02-28');
298 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
299 VALUES ('253','MARIANA','LOPEZ CABRERA','938205576','FEMENINO','1998-05-28');
300 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
301 VALUES ('254','ALEJANDRA','MOLINA CARBALLO','938355580','FEMENINO','1990-02-28');
302
303 • savepoint primer_lote;
304 -- Se guarda el primer lote de 4 clientes
305
306 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
307 VALUES ('255','MARTHA','CASTELLON MARTINEZ','955265580','FEMENINO','1992-06-25');
308 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
309 VALUES ('256','CARLOS','ALVAREZ BRAVO','938234770','MASCULINO','1993-03-22');
310 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
311 VALUES ('257','ALBERTO','LOPEZ LACAYO','938203276','MASCULINO','1998-09-28');
312 • INSERT INTO clientes (id_cliente, nombre, apellido, telefono, sexo, fecha_nacimiento)
313 VALUES ('258','JOAQUIN','GUZMAN CARBALLO','932255580','MASCULINO','1990-02-28');
314
315 • savepoint segundo_lote;
316 -- Se guarda el segundo lote de 4 clientes y se deja comentado un rollback al primer lote para mostrar los 4 primeros clientes.
317
318  -- ROLLBACK TO primer_lote;
```

## 10. Prácticas de creación de Usuarios y Permisos

- Creando usuario y otorgando permisos de Select y Update.
  - Se crea este usuario **'coderprueba1'@'localhost'** al cual solo se le otorgan permisos de Select y Update en algunas tablas.

```
/* Empezamos creando el usuario 'coderprueba1'@'localhost' */
• CREATE USER 'coderprueba1'@'localhost';
/* Vemos que permisos tiene el usuario 'coderprueba1'@'localhost', observando que aun no se le ha otorgado ninguno */
• SHOW GRANTS FOR 'coderprueba1'@'localhost';

/* Le damos permiso de Lectura para las 5 tablas principales de la base tellevo al usuario 'coderprueba1'@'localhost' */
• GRANT SELECT, UPDATE ON tellevo.ciudades TO 'coderprueba1'@'localhost';
• GRANT SELECT, UPDATE ON tellevo.clientes TO 'coderprueba1'@'localhost';
• GRANT SELECT, UPDATE ON tellevo.conductores TO 'coderprueba1'@'localhost';
• GRANT SELECT, UPDATE ON tellevo.recorridos TO 'coderprueba1'@'localhost';
• GRANT SELECT, UPDATE ON tellevo.vehiculos TO 'coderprueba1'@'localhost';
```



1	SHOW GRANTS FOR 'coderprueba1'@'localhost';
2	

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Grants for coderprueba1@localhost			
▶ GRANT USAGE ON *.* TO 'coderprueba1'@'localhost'			
GRANT SELECT, UPDATE ON `tellevo`.`ciudades` TO 'coderprueba1'@'localhost';			
GRANT SELECT, UPDATE ON `tellevo`.`clientes` TO 'coderprueba1'@'localhost';			
GRANT SELECT, UPDATE ON `tellevo`.`conductores` TO 'coderprueba1'@'localhost';			
GRANT SELECT, UPDATE ON `tellevo`.`recorridos` TO 'coderprueba1'@'localhost';			
GRANT SELECT, UPDATE ON `tellevo`.`vehiculos` TO 'coderprueba1'@'localhost';			

- Creando usuario y otorgando todos los permisos.



- Se crea este usuario '**coderpruebatodos**'@'**localhost**' al cual se le otorgan permisos de SELECT, UPDATE, INSERT, CREATE, DROP en algunas tablas, este usuario tiene más permisos que el anterior.

```
17 /* Empezamos creando el usuario 'coderpruebatodos'@'localhost' */
18 • CREATE USER 'coderpruebatodos'@'localhost';
19 /* Vemos que permisos tiene el usuario 'coderpruebatodos'@'localhost', observando que aun no se le ha otorgado ninguno */
20 • SHOW GRANTS FOR 'coderpruebatodos'@'localhost';
21
22 /* Le damos permisos de Lectura, insercion y modificacion de valores para las 5 tablas principales de la base tellevo al usuario 'coderpruebatodos'@'localhost' */
23 • GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON tellevo.ciudades TO 'coderpruebatodos'@'localhost';
24 • GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON tellevo.clientes TO 'coderpruebatodos'@'localhost';
25 • GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON tellevo.conductores TO 'coderpruebatodos'@'localhost';
26 • GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON tellevo.recorridos TO 'coderpruebatodos'@'localhost';
27 • GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON tellevo.vehiculos TO 'coderpruebatodos'@'localhost';
```

```
1 SHOW GRANTS FOR 'coderpruebatodos'@'localhost';
2
```