

# PARCIAL

# BASE DE DATOS I

Profesor:


Rolando Titiosky

Integrantes:

Matias Marzorati

Matias Monzalvo

Ramiro Sanabria

	<p><b>PARCIAL</b></p>	<p><i>Páginas: 39</i> <i>Versión: 1.0</i></p>
---	-----------------------	---

<b>Asignatura</b>	<b>BASE DE DATOS I</b>
<b>Carrera</b>	<b>Ing. en Informática</b>
<b>Plan</b>	<b>Ajuste 2012</b>
<b>Ciclo</b>	<b>1ero</b>
<b>Cuatrimestre</b>	<b>2do</b>
<b>Tema/Título</b>	<b>Parcial</b>
<b>Profesor</b>	<b>Prof. Titiosky</b>

### Grupo de Trabajo

<b>APELLIDO, Nombres</b>
MARZORATI, Matias
MONZALVO, Matias
SANABRIA, Ramiro

### Grilla de calificación

Concepto	Propuesta	Marco Teórico	Desarrollo propio	Conclusiones	Fuentes y Referencias
Sobresaliente (10)					
Distinguido (9-8)					
Bueno (7-6)					
Aprobado (5-4)					
Insuficiente (3-2-1)					
Reprobado (0)					
NOTA					
Comentario adicional del Profesor:					

# 1. Propuesta

## Compañía de seguros:

### Introducción:

En la industria de seguros, la gestión eficiente de las pólizas es esencial para garantizar la satisfacción de los clientes y el cumplimiento de los contratos. Para abordar esta necesidad, desarrollaremos una base de datos especializada para la gestión de pólizas en una compañía de seguros.

### Objetivos:

Los objetivos de la implementación de esta base de datos son los siguientes:

- Facilitar la creación y consulta de pólizas para agentes y clientes de manera eficiente.
- Proporcionar una plataforma segura para el registro de la información de las pólizas y los reclamos de los clientes.
- Optimizar la gestión de pagos y renovaciones de pólizas.
- Cumplir con los estándares de seguridad y regulaciones de privacidad de datos en la industria de seguros.

### DER Físico:

El Diagrama Entidad-Relación (DER) Físico representará la estructura física de la base de datos, incluyendo las tablas, los campos, los índices y las relaciones entre las entidades. Este DER proporcionará una visión detallada de la arquitectura de la base de datos de seguros.

### Conclusión:

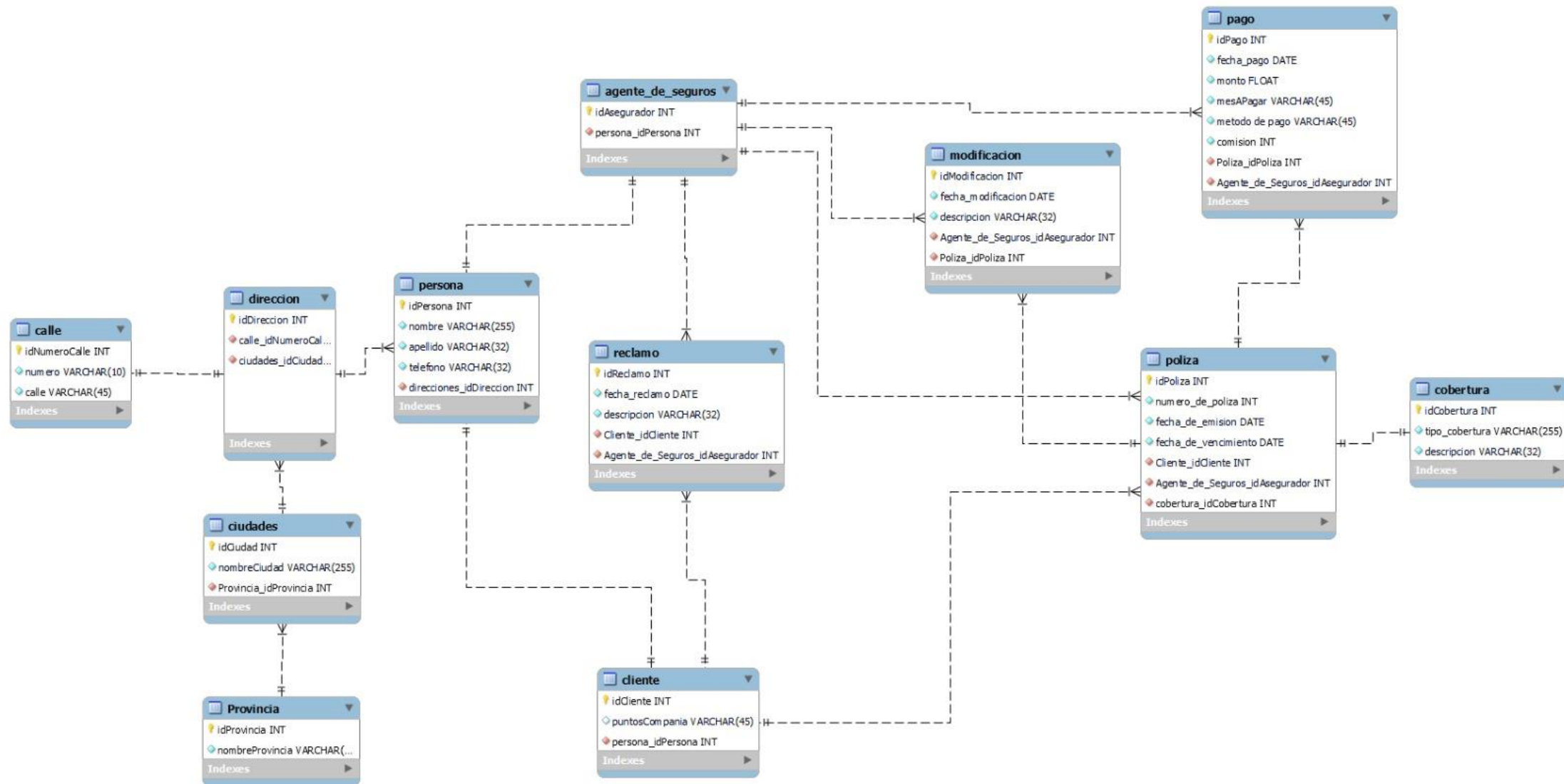
La implementación de esta base de datos de gestión de pólizas de seguros tiene como objetivo mejorar la eficiencia y la calidad de los servicios de la compañía de seguros al optimizar la gestión de pólizas, pagos y reclamos. Esta solución contribuirá a una experiencia más fluida y satisfactoria tanto para los agentes como para los clientes.

### ALCANCE:

Meta de alcance = 10

Se implementará la base de datos en un servidor de BD, junto con 1 Trigger, 6 Procedures y 1 Function. Se creará un Diagrama Entidad-Relación Físico y la documentación correspondiente para la presentación del proyecto.

## 2. DER Físico



# 3. Creates (Códigos en texto al final de la Documentación)

## 1 - Persona

```
• CREATE TABLE IF NOT EXISTS `seguros`.`persona` (  
  `idPersona` INT NOT NULL AUTO_INCREMENT,  
  `nombre` VARCHAR(255) NOT NULL,  
  `apellido` VARCHAR(32) NOT NULL,  
  `telefono` VARCHAR(32) NOT NULL,  
  `direcciones_idDireccion` INT NOT NULL,  
  PRIMARY KEY (`idPersona`),  
  INDEX `fk_persona_direcciones1_idx` (`direcciones_idDireccion` ASC) VISIBLE,  
  CONSTRAINT `fk_persona_direcciones1`  
    FOREIGN KEY (`direcciones_idDireccion`)  
      REFERENCES `seguros`.`direccion` (`idDireccion`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 2 - Agente de seguros

```
• CREATE TABLE IF NOT EXISTS `seguros`.`agente_de_seguros` (  
  `idAsegurador` INT NOT NULL AUTO_INCREMENT,  
  `persona_idPersona` INT NOT NULL,  
  PRIMARY KEY (`idAsegurador`),  
  INDEX `fk_agente_de_seguros_persona1_idx` (`persona_idPersona` ASC) VISIBLE,  
  CONSTRAINT `fk_agente_de_seguros_persona1`  
    FOREIGN KEY (`persona_idPersona`)  
      REFERENCES `seguros`.`persona` (`idPersona`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

### 3 - Cliente

```
• CREATE TABLE IF NOT EXISTS `seguros`.`cliente` (  
  `idCliente` INT NOT NULL AUTO_INCREMENT,  
  `puntosCompania` VARCHAR(45) NULL,  
  `persona_idPersona` INT NOT NULL,  
  PRIMARY KEY (`idCliente`),  
  INDEX `fk_cliente_persona1_idx` (`persona_idPersona` ASC) VISIBLE,  
  CONSTRAINT `fk_cliente_persona1`  
    FOREIGN KEY (`persona_idPersona`)  
    REFERENCES `seguros`.`persona` (`idPersona`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

### 4 - Póliza

```
• CREATE TABLE IF NOT EXISTS `seguros`.`poliza` (  
  `idPoliza` INT NOT NULL AUTO_INCREMENT,  
  `numero_de_poliza` INT NOT NULL,  
  `fecha_de_emision` DATE NOT NULL,  
  `fecha_de_vencimiento` DATE NOT NULL,  
  `Cliente_idCliente` INT NOT NULL,  
  `Agente_de_Seguros_idAsegurador` INT NOT NULL,  
  `cobertura_idCobertura` INT NOT NULL,  
  PRIMARY KEY (`idPoliza`),  
  INDEX `fk_Poliza_Cliente1_idx` (`Cliente_idCliente` ASC) VISIBLE,  
  INDEX `fk_Poliza_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,  
  INDEX `fk_poliza_cobertura1_idx` (`cobertura_idCobertura` ASC) VISIBLE,  
  CONSTRAINT `fk_Poliza_Agente_de_Seguros1`  
    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)  
    REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),  
  CONSTRAINT `fk_Poliza_Cliente1`  
    FOREIGN KEY (`Cliente_idCliente`)  
    REFERENCES `seguros`.`cliente` (`idCliente`),  
  CONSTRAINT `fk_poliza_cobertura1`  
    FOREIGN KEY (`cobertura_idCobertura`)  
    REFERENCES `seguros`.`cobertura` (`idCobertura`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 5 - Modificación

```
• CREATE TABLE IF NOT EXISTS `seguros`.`modificacion` (  
  `idModificacion` INT NOT NULL AUTO_INCREMENT,  
  `fecha_modificacion` DATE NOT NULL,  
  `descripcion` VARCHAR(32) NOT NULL,  
  `Agente_de_Seguros_idAsegurador` INT NOT NULL,  
  `Poliza_idPoliza` INT NOT NULL,  
  PRIMARY KEY (`idModificacion`),  
  INDEX `fk_Modificacion_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,  
  INDEX `fk_Modificacion_Poliza1_idx` (`Poliza_idPoliza` ASC) VISIBLE,  
  CONSTRAINT `fk_Modificacion_Agente_de_Seguros1`  
    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)  
      REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),  
  CONSTRAINT `fk_Modificacion_Poliza1`  
    FOREIGN KEY (`Poliza_idPoliza`)  
      REFERENCES `seguros`.`poliza` (`idPoliza`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 6 - Pago

```
• CREATE TABLE IF NOT EXISTS `seguros`.`pago` (  
  `idPago` INT NOT NULL AUTO_INCREMENT,  
  `fecha_pago` DATE NOT NULL,  
  `monto` FLOAT NOT NULL,  
  `mesAPagar` VARCHAR(45) NOT NULL,  
  `metodo de pago` VARCHAR(45) NOT NULL,  
  `comision` INT NOT NULL,  
  `Poliza_idPoliza` INT NOT NULL,  
  `Agente_de_Seguros_idAsegurador` INT NOT NULL,  
  PRIMARY KEY (`idPago`),  
  INDEX `fk_Pago_Poliza1_idx` (`Poliza_idPoliza` ASC) VISIBLE,  
  INDEX `fk_Pago_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,  
  CONSTRAINT `fk_Pago_Agente_de_Seguros1`  
    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)  
      REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),  
  CONSTRAINT `fk_Pago_Poliza1`  
    FOREIGN KEY (`Poliza_idPoliza`)  
      REFERENCES `seguros`.`poliza` (`idPoliza`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 7 – Cobertura

```
• CREATE TABLE IF NOT EXISTS `seguros`.`cobertura` (  
    `idCobertura` INT NOT NULL AUTO_INCREMENT,  
    `tipo_cobertura` VARCHAR(255) NOT NULL,  
    `descripcion` VARCHAR(32) NOT NULL,  
    PRIMARY KEY (`idCobertura`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 8 – Reclamo

```
• CREATE TABLE IF NOT EXISTS `seguros`.`reclamo` (  
    `idReclamo` INT NOT NULL AUTO_INCREMENT,  
    `fecha_reclamo` DATE NOT NULL,  
    `descripcion` VARCHAR(32) NOT NULL,  
    `Cliente_idCliente` INT NOT NULL,  
    `Agente_de_Seguros_idAsegurador` INT NOT NULL,  
    PRIMARY KEY (`idReclamo`),  
    INDEX `fk_Reclamo_Cliente1_idx` (`Cliente_idCliente` ASC) VISIBLE,  
    INDEX `fk_Reclamo_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,  
    CONSTRAINT `fk_Reclamo_Agente_de_Seguros1`  
        FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)  
        REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),  
    CONSTRAINT `fk_Reclamo_Cliente1`  
        FOREIGN KEY (`Cliente_idCliente`)  
        REFERENCES `seguros`.`cliente` (`idCliente`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```



## 9 – Dirección

```
• CREATE TABLE IF NOT EXISTS `seguros`.`direccion` (  
  `idDireccion` INT NOT NULL AUTO_INCREMENT,  
  `calle_idNumeroCalle` INT NOT NULL,  
  `ciudades_idCiudad` INT NOT NULL,  
  PRIMARY KEY (`idDireccion`),  
  INDEX `fk_direcciones_calle1_idx` (`calle_idNumeroCalle` ASC) VISIBLE,  
  INDEX `fk_direccion_ciudades1_idx` (`ciudades_idCiudad` ASC) VISIBLE,  
  CONSTRAINT `fk_direcciones_calle1`  
    FOREIGN KEY (`calle_idNumeroCalle`)  
      REFERENCES `seguros`.`calle` (`idNumeroCalle`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION,  
  CONSTRAINT `fk_direccion_ciudades1`  
    FOREIGN KEY (`ciudades_idCiudad`)  
      REFERENCES `seguros`.`ciudades` (`idCiudad`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 10 – Ciudad

```
• CREATE TABLE IF NOT EXISTS `seguros`.`ciudades` (  
  `idCiudad` INT NOT NULL AUTO_INCREMENT,  
  `nombreCiudad` VARCHAR(255) NOT NULL,  
  `Provincia_idProvincia` INT NOT NULL,  
  PRIMARY KEY (`idCiudad`),  
  UNIQUE INDEX `nombreCiudad_UNIQUE` (`nombreCiudad` ASC) VISIBLE,  
  INDEX `fk_ciudades_Provincial_idx` (`Provincia_idProvincia` ASC) VISIBLE,  
  CONSTRAINT `fk_ciudades_Provincial`  
    FOREIGN KEY (`Provincia_idProvincia`)  
      REFERENCES `seguros`.`Provincia` (`idProvincia`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 11 – Provincia

```
• CREATE TABLE IF NOT EXISTS `seguros`.`Provincia` (  
  `idProvincia` INT NOT NULL AUTO_INCREMENT,  
  `nombreProvincia` VARCHAR(255) NOT NULL,  
  PRIMARY KEY (`idProvincia`),  
  UNIQUE INDEX `nombreCiudad_UNIQUE` (`nombreProvincia` ASC) VISIBLE)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

## 12 – Calle

```
• CREATE TABLE IF NOT EXISTS `seguros`.`calle` (  
  `idNumeroCalle` INT NOT NULL AUTO_INCREMENT,  
  `numero` VARCHAR(10) NOT NULL,  
  `calle` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idNumeroCalle`),  
  UNIQUE INDEX `numero_UNIQUE` (`numero` ASC) VISIBLE)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```

### Detalle:

- Se definen 12 tablas junto con las relaciones entre ellas.
- Almacenan información relacionada con los seguros.
- Se establecen restricciones de clave foránea para mantener la integridad de los datos y se realizan configuraciones específicas

## **Tabla por tabla:**

La tabla "calle" almacena información sobre números de calles y nombres de calles, con un índice único en el número de calle.

La tabla "Provincia" almacena nombres de provincias con un índice único en el nombre de provincia.

La tabla "ciudades" registra nombres de ciudades relacionados con las provincias y utiliza una clave externa para establecer relaciones con la tabla "Provincia".

La tabla "direccion" relaciona calles y ciudades a través de claves foráneas y almacena información sobre direcciones.

La tabla "persona" almacena detalles sobre personas, como nombres, apellidos y números de teléfono, relacionándolos con direcciones.

La tabla "agente\_de\_seguros" está vinculada a personas y se utiliza para representar a los agentes de seguros.

La tabla "cliente" está relacionada con personas y almacena información sobre clientes, incluyendo puntos de compañía.

La tabla "cobertura" registra tipos de cobertura y sus descripciones.

La tabla "poliza" almacena información sobre pólizas de seguro, incluyendo fechas de emisión y vencimiento, con relaciones con clientes, agentes de seguros y tipos de cobertura.

La tabla "modificacion" registra cambios realizados en pólizas y está vinculada a agentes de seguros y pólizas.

La tabla "pago" almacena detalles sobre pagos relacionados con pólizas y agentes de seguros.

La tabla "reclamo" registra reclamos realizados por clientes y agentes de seguros relacionados.

# 4. Inserts

## 1 – (Provincia, Ciudades, Calle, Dirección)

```
INSERT INTO `seguros`.`provincia` (`idProvincia`, `nombreProvincia`)
VALUES
(1, 'Buenos Aires'),
(2, 'CABA');
select * from provincia;
```

```
INSERT INTO `seguros`.`ciudades` (`idCiudad`, `nombreCiudad`, `Provincia_idProvincia`)
VALUES
(1, 'Ballester', 1),
(2, 'Belgrano', 2);
select * from ciudades;
```

```
INSERT INTO `seguros`.`calle` (`idNumeroCalle`, `numero`, `calle`)
VALUES
(1, '5354', 'Pacífico Rodriguez'),
(2, '893', 'Arce');
select * from calle;
```

```
INSERT INTO `seguros`.`direccion` (`idDireccion`, `calle_idNumeroCalle`, `ciudades_idCiudad`)
VALUES
(1, 1, 1),
(2, 2, 2);
select * from direccion;
```

## Output:

	idProvincia	nombreProvincia
▶	1	Buenos Aires
	2	CABA
*	NULL	NULL

	idCiudad	nombreCiudad	Provincia_idProvincia
▶	1	Ballester	1
	2	Belgrano	2
*	NULL	NULL	NULL

	idNumeroCalle	numero	calle
▶	1	5354	Pacífico Rodriguez
	2	893	Arce
*	NULL	NULL	NULL

	idDireccion	calle_idNumeroCalle	ciudades_idCiudad
▶	1	1	1
	2	2	2
*	NULL	NULL	NULL

## 2 – (Persona, Agente de seguros, Cliente)

```
INSERT INTO `seguros`.`persona` (`idPersona`, `nombre`, `apellido`, `telefono`, `direcciones_idDireccion`)
VALUES
```

```
(1, 'Matias', 'Marzorat', '11 2513 0914', 1),
(2, 'Gisela', 'Perugorria', '11 6453 0914', 1),
(3, 'Valen', 'Werle', '11 0000 1111', 2);
select * from persona;
```

```
INSERT INTO `seguros`.`agente_de_seguros` (`idAsegurador`, `persona_idPersona`)
VALUES
```

```
(1, 1),
(2, 2);
select * from agente_de_seguros;
```

```
INSERT INTO `seguros`.`cliente` (`idCliente`, `puntosCompania`, `persona_idPersona`)
VALUES
```

```
(1,0, 3);
select * from cliente;
```

### Output:

	idPersona	nombre	apellido	telefono	direcciones_idDireccion
▶	1	Matias	Marzorat	11 2513 0914	1
	2	Gisela	Perugorria	11 6453 0914	1
	3	Valen	Werle	11 0000 1111	2
*	NULL	NULL	NULL	NULL	NULL

	idAsegurador	persona_idPersona
▶	1	1
	2	2
*	NULL	NULL

	idCliente	puntosCompania	persona_idPersona
▶	1	0	3
*	NULL	NULL	NULL

VALUES

```
(1, 'Todo Riesgo', 'Cobertura ante todo riesgo'),
```

```
(2, 'Premium', 'Cobertura premium');
```

```
select * from cobertura;
```

```
VALUES (1, 1001, '2023-09-15', '2024-09-15', 1, 1, 1),
```

```
(2, 2002, '2023-09-16', '2024-09-16', 1, 2, 2),
```

```
(3, 3003, '2023-09-17', '2024-09-17', 1, 2, 1);
```

	idCobertura	tipo_cobertura	descripcion
▶	1	Todo Riesgo	Cobertura ante todo riesgo
	2	Premium	Cobertura premium
✱	NULL	NULL	NULL

[illegible]

#### 4 - (Modificación, Pago, Reclamo)

```
INSERT INTO `seguros`.`modificacion` (`idModificacion`, `fecha_modificacion`, `descripcion`, `Agente_de_Seguros_idAsegurador`, `Poliza_idPoliza`)
VALUES (1, '2023-09-15', 'Alteracion de precios', 1, 1),
(2, '2023-09-16', 'alteracion de vehiculo', 2, 2),
(3, '2023-09-17', 'Modificación 3', 1, 3);

select * from modificacion;
```

```
INSERT INTO `seguros`.`pago` (`idPago`, `fecha_pago`, `monto`, `mesAPagar`, `metodo de pago`, `comision`, `Poliza_idPoliza`, `Agente_de_Seguros_idAsegurador`)
VALUES (1, '2023-09-15', 100.0, 'Septiembre', 'efec',10, 1, 1),
(2, '2023-09-16', 200.0, 'Septiembre', 'efec',10, 2, 2),
(3, '2023-09-17', 300.0, 'Septiembre', 'efec',10, 3, 1);

select * from pago;
```

```
INSERT INTO `seguros`.`reclamo` (`idReclamo`, `fecha_reclamo`, `descripcion`, `Cliente_idCliente`, `Agente_de_Seguros_idAsegurador`)
VALUES (1, '2023-09-15', 'Reclamo 1', 1, 1),
(2, '2023-09-16', 'Reclamo 2', 1, 2),
(3, '2023-09-17', 'Reclamo 3', 1, 1);

select * from reclamo;
```

### Output:

	idModificacion	fecha_modificacion	descripcion	Agente_de_Seguros_idAsegurador	Poliza_idPoliza
▶	1	2023-09-15	Alteracion de precios	1	1
	2	2023-09-16	alteracion de vehiculo	2	2
	3	2023-09-17	Modificación 3	1	3
*	NULL	NULL	NULL	NULL	NULL

	idReclamo	fecha_reclamo	descripcion	Cliente_idCliente	Agente_de_Seguros_idAsegurador
▶	1	2023-09-15	Reclamo 1	1	1
	2	2023-09-16	Reclamo 2	1	2
	3	2023-09-17	Reclamo 3	1	1
*	NULL	NULL	NULL	NULL	NULL

[illegible]

# 5. Procedures

## Cliente:

### 1 – getReclamosPorCliente:

Devuelve todos los reclamos asociados a un cliente específico, identificado por clienteID.

```
1  DELIMITER //
2  • CREATE PROCEDURE GetReclamosPorCliente(IN clienteID INT)
3  BEGIN
4      SELECT r.*
5      FROM reclamo r
6      WHERE r.Cliente_idCliente = clienteID;
7  END //
8  DELIMITER ;
9
10 • CALL GetReclamosPorCliente(1);
```

idReclamo	fecha_reclamo	descripcion	Cliente_idCliente	Agente_de_Seguros_idAsegurador
1	2023-09-15	Reclamo 1	1	1
2	2023-09-16	Reclamo 2	1	2
3	2023-09-17	Reclamo 3	1	1

### 2 – getPolizaPorCliente

Recupera todas las pólizas asociadas a un cliente específico, identificado por clienteID.

```
19  DELIMITER //
20  • CREATE PROCEDURE GetPolizasPorCliente(IN clienteID INT)
21  BEGIN
22      SELECT p.*
23      FROM poliza p
24      WHERE p.Cliente_idCliente = clienteID;
25  END //
26  DELIMITER ;
27
28  • call GetPolizasPorCliente(1);
```

idPoliza	numero_de_poliza	fecha_de_emision	fecha_de_vencimiento	Cliente_idCliente	Agente_de_Seguros_idAsegurador	cobertura_idCobertura
1	1001	2023-09-15	2024-09-15	1	1	1
2	2002	2023-09-16	2024-09-16	1	2	2
3	3003	2023-09-17	2024-09-17	1	2	1



### 3 - getPagosPorCliente

Obtiene todos los pagos relacionados con las pólizas de un cliente específico, identificado por clienteID

```
23 DELIMITER //
24 • CREATE PROCEDURE GetPagosPorCliente(IN clienteID INT)
25 BEGIN
26     SELECT pa.*
27     FROM pago pa
28     JOIN poliza po ON pa.Poliza_idPoliza = po.idPoliza
29     WHERE po.Cliente_idCliente = clienteID;
30 END //
31 DELIMITER ;
32
33 • call GetPagosPorCliente(1);
```

Result Grid							
Filter Rows:		Export:		Wrap Cell Content: <a href="#">IA</a>			
	idPago	fecha_pago	monto	mesAPagar	metodo de pago	Poliza_idPoliza	Agente_de_Seguros_idAsegurador
▶	1	2023-09-15	100	Septiembre	efec	1	1
	2	2023-09-16	200	Septiembre	efec	2	2
	3	2023-09-17	300	Septiembre	efec	3	1

## Agente:

### 1 – getReclamosPorAgente

Devuelve todos los reclamos asociados a un agente de seguros específico, identificado por agenteID.

```
1 DELIMITER //
2 • CREATE PROCEDURE GetReclamosPorAgente(IN agenteID INT)
3 BEGIN
4     SELECT r.*
5     FROM reclamo r
6     WHERE r.Agente_de_Seguros_idAsegurador = agenteID;
7 END //
8 DELIMITER ;
9
10 • call GetReclamosPorAgente(1);
11 • call GetReclamosPorAgente(2);
```

Result Grid					
Filter Rows:		Export:		Wrap Cell Content: <a href="#">IA</a>	
	idReclamo	fecha_reclamo	descripcion	Cliente_idCliente	Agente_de_Seguros_idAsegurador
▶	2	2023-09-16	Reclamo 2	1	2

## 2 – getPolizasPorAgente

Recupera todas las pólizas asociadas a un agente de seguros específico, identificado por agenteID.

```
13 DELIMITER //
14 • CREATE PROCEDURE GetPolizasPorAgente(IN agenteID INT)
15 BEGIN
16     SELECT p.*
17     FROM poliza p
18     WHERE p.Agente_de_Seguros_idAsegurador = agenteID;
19 END //
20 DELIMITER ;
21 • call GetPolizasPorAgente(1);
22 • call GetPolizasPorAgente(2);
23
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	idPoliza	numero_de_poliza	fecha_de_emision	fecha_de_vencimiento	Cliente_idCliente	Agente_de_Seguros_idAsegurador	cobertura_idCobertura
•	2	2002	2023-09-16	2024-09-16	1	2	2
	3	3003	2023-09-17	2024-09-17	1	2	1

## 3 – getPagosPorAgente

Obtiene todos los pagos relacionados con las pólizas gestionadas por un agente de seguros específico, identificado por agenteID.

```
24 DELIMITER //
25 • CREATE PROCEDURE GetPagosPorAgente(IN agenteID INT)
26 BEGIN
27     SELECT pa.*
28     FROM pago pa
29     WHERE pa.Agente_de_Seguros_idAsegurador = agenteID;
30 END //
31 DELIMITER ;
32
33 • call GetPagosPorAgente(1);
34 • call GetPagosPorAgente(2);
35
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	idPago	fecha_pago	monto	mesAPagar	metodo de pago	Poliza_idPoliza	Agente_de_Seguros_idAsegurador
►	1	2023-09-15	100	Septiembre	efec	1	1
	3	2023-09-17	300	Septiembre	efec	3	1

# 6. Function

## CalcularComisionAgente

Calcula la comisión total ganada por un agente de seguros específico (identificado por "p\_idAgente"). Utiliza un cursor para recorrer los registros de la tabla "pago" relacionados con el agente y acumula la comisión total calculada como un porcentaje del monto de cada pago. El resultado final se devuelve como un valor decimal con dos decimales.

```
1  DELIMITER //
2  CREATE FUNCTION CalcularComisionAgente(
3      p_idAgente INT
4  )
5      RETURNS DECIMAL(10, 2)
6      DETERMINISTIC
7  BEGIN
8      DECLARE comision_total DECIMAL(10, 2) DEFAULT 0.0;
9      DECLARE done INT DEFAULT FALSE;
10     DECLARE monto_pago DECIMAL(10, 2); -- Declarar las variables antes del bucle
11     DECLARE comision_porcentaje DECIMAL(5, 2);
12     DECLARE cur CURSOR FOR
13         SELECT monto, comision
14         FROM pago
15         WHERE Agente_de_Seguros_idAsegurador = p_idAgente;
16     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
17     OPEN cur;
18     read_loop: LOOP
19         FETCH cur INTO monto_pago, comision_porcentaje;
20         IF done THEN
21             LEAVE read_loop;
22         END IF;
23         SET comision_total = comision_total + (monto_pago * comision_porcentaje / 100);
24     END LOOP;
25     CLOSE cur;
26     RETURN comision_total;
27 END;
28 //
29 DELIMITER ;
```

Output:

Result Grid	
	CalcularComisionAgente(1)
	40.00

# 7. Trigger

## IncrementarPuntosCompania

```
1 DELIMITER //
2 • CREATE TRIGGER IncrementarPuntosCompania
3 AFTER INSERT ON poliza
4 FOR EACH ROW
5 BEGIN
6     DECLARE cliente_id INT;
7
8     SELECT Cliente_idCliente INTO cliente_id
9     FROM poliza
10    WHERE idPoliza = NEW.idPoliza;
11
12    UPDATE cliente
13    SET puntosCompania = puntosCompania + 1
14    WHERE idCliente = cliente_id;
15 END;
16 //
17 DELIMITER ;
18
```

### Output:

#### Antes:

```
21 • select * from cliente;
```

idCliente	puntosCompania	persona_idPersona
1	5	3
NULL	NULL	NULL

Con 5 puntos, ya que ese cliente a modo de prueba le cargamos 5 pólizas

#### Después:

```
23 • INSERT INTO poliza (numero_de_poliza, fecha_de_emision,
24 VALUES (4006, '2023-09-18', '2024-09-18', 1, 1, 1);
25 • select * from cliente;
26
```

idCliente	puntosCompania	persona_idPersona
1	6	3
NULL	NULL	NULL

Al insertar una nueva póliza a su nombre se suma un punto en su atributo PuntosCompania

# Código Completo

## Creates:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
```

```
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,  
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,E  
RROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
CREATE SCHEMA IF NOT EXISTS `seguros` DEFAULT CHARACTER SET utf8mb3 ;
```

```
USE `seguros` ;
```

```
CREATE TABLE IF NOT EXISTS `seguros`.`calle` (  
  
  `idNumeroCalle` INT NOT NULL AUTO_INCREMENT,  
  
  `numero` VARCHAR(10) NOT NULL,  
  
  `calle` VARCHAR(45) NOT NULL,  
  
  PRIMARY KEY (`idNumeroCalle`),  
  
  UNIQUE INDEX `numero_UNIQUE` (`numero` ASC) VISIBLE)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```

```
CREATE TABLE IF NOT EXISTS `seguros`.`Provincia` (  
  
  `idProvincia` INT NOT NULL AUTO_INCREMENT,  
  
  `nombreProvincia` VARCHAR(255) NOT NULL,  
  
  PRIMARY KEY (`idProvincia`),  
  
  UNIQUE INDEX `nombreCiudad_UNIQUE` (`nombreProvincia` ASC) VISIBLE)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```

```
CREATE TABLE IF NOT EXISTS `seguros`.`ciudades` (  
  
  `idCiudad` INT NOT NULL AUTO_INCREMENT,  
  
  `nombreCiudad` VARCHAR(255) NOT NULL,  
  
  `Provincia_idProvincia` INT NOT NULL,  
  
  PRIMARY KEY (`idCiudad`),  
  
  UNIQUE INDEX `nombreCiudad_UNIQUE` (`nombreCiudad` ASC) VISIBLE,  
  
  INDEX `fk_ciudades_Provincia1_idx` (`Provincia_idProvincia` ASC) VISIBLE,  
  
  CONSTRAINT `fk_ciudades_Provincia1`  
  
    FOREIGN KEY (`Provincia_idProvincia`)  
  
    REFERENCES `seguros`.`Provincia` (`idProvincia`)  
  
    ON DELETE NO ACTION  
  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```

```

CREATE TABLE IF NOT EXISTS `seguros`.`direccion` (

  `idDireccion` INT NOT NULL AUTO_INCREMENT,

  `calle_idNumeroCalle` INT NOT NULL,

  `ciudades_idCiudad` INT NOT NULL,

  PRIMARY KEY (`idDireccion`),

  INDEX `fk_direcciones_calle1_idx` (`calle_idNumeroCalle` ASC) VISIBLE,

  INDEX `fk_direccion_ciudades1_idx` (`ciudades_idCiudad` ASC) VISIBLE,

  CONSTRAINT `fk_direcciones_calle1`

    FOREIGN KEY (`calle_idNumeroCalle`)

      REFERENCES `seguros`.`calle` (`idNumeroCalle`)

        ON DELETE NO ACTION

        ON UPDATE NO ACTION,

  CONSTRAINT `fk_direccion_ciudades1`

    FOREIGN KEY (`ciudades_idCiudad`)

      REFERENCES `seguros`.`ciudades` (`idCiudad`)

        ON DELETE NO ACTION

        ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb3;

```

```
CREATE TABLE IF NOT EXISTS `seguros`.`persona` (  
  
  `idPersona` INT NOT NULL AUTO_INCREMENT,  
  
  `nombre` VARCHAR(255) NOT NULL,  
  
  `apellido` VARCHAR(32) NOT NULL,  
  
  `telefono` VARCHAR(32) NOT NULL,  
  
  `direcciones_idDireccion` INT NOT NULL,  
  
  PRIMARY KEY (`idPersona`),  
  
  INDEX `fk_persona_direcciones1_idx` (`direcciones_idDireccion` ASC) VISIBLE,  
  
  CONSTRAINT `fk_persona_direcciones1`  
  
    FOREIGN KEY (`direcciones_idDireccion`)  
  
    REFERENCES `seguros`.`direccion` (`idDireccion`)  
  
    ON DELETE NO ACTION  
  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```



```
CREATE TABLE IF NOT EXISTS `seguros`.`agente_de_seguros` (  
  
  `idAsegurador` INT NOT NULL AUTO_INCREMENT,  
  
  `persona_idPersona` INT NOT NULL,  
  
  PRIMARY KEY (`idAsegurador`),  
  
  INDEX `fk_agente_de_seguros_persona1_idx` (`persona_idPersona` ASC) VISIBLE,  
  
  CONSTRAINT `fk_agente_de_seguros_persona1`  
  
    FOREIGN KEY (`persona_idPersona`)  
  
    REFERENCES `seguros`.`persona` (`idPersona`)  
  
    ON DELETE NO ACTION  
  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```

```

CREATE TABLE IF NOT EXISTS `seguros`.`cliente` (

  `idCliente` INT NOT NULL AUTO_INCREMENT,

  `puntosCompania` VARCHAR(45) NULL,

  `persona_idPersona` INT NOT NULL,

  PRIMARY KEY (`idCliente`),

  INDEX `fk_cliente_persona1_idx` (`persona_idPersona` ASC) VISIBLE,

  CONSTRAINT `fk_cliente_persona1`

    FOREIGN KEY (`persona_idPersona`)

      REFERENCES `seguros`.`persona` (`idPersona`)

      ON DELETE NO ACTION

      ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb3;

```

```

CREATE TABLE IF NOT EXISTS `seguros`.`cobertura` (

  `idCobertura` INT NOT NULL AUTO_INCREMENT,

  `tipo_cobertura` VARCHAR(255) NOT NULL,

  `descripcion` VARCHAR(32) NOT NULL,

  PRIMARY KEY (`idCobertura`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb3;

```

```

CREATE TABLE IF NOT EXISTS `seguros`.`poliza` (

  `idPoliza` INT NOT NULL AUTO_INCREMENT,

  `numero_de_poliza` INT NOT NULL,

  `fecha_de_emision` DATE NOT NULL,

  `fecha_de_vencimiento` DATE NOT NULL,

  `Cliente_idCliente` INT NOT NULL,

  `Agente_de_Seguros_idAsegurador` INT NOT NULL,

  `cobertura_idCobertura` INT NOT NULL,

  PRIMARY KEY (`idPoliza`),

  INDEX `fk_Poliza_Cliente1_idx` (`Cliente_idCliente` ASC) VISIBLE,

  INDEX `fk_Poliza_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador`
ASC) VISIBLE,

  INDEX `fk_poliza_cobertura1_idx` (`cobertura_idCobertura` ASC) VISIBLE,

  CONSTRAINT `fk_Poliza_Agente_de_Seguros1`

    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)

    REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),

  CONSTRAINT `fk_Poliza_Cliente1`

    FOREIGN KEY (`Cliente_idCliente`)

    REFERENCES `seguros`.`cliente` (`idCliente`),

  CONSTRAINT `fk_poliza_cobertura1`

    FOREIGN KEY (`cobertura_idCobertura`)

    REFERENCES `seguros`.`cobertura` (`idCobertura`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION)

ENGINE = InnoDB

```

```
DEFAULT CHARACTER SET = utf8mb3;
```

```
CREATE TABLE IF NOT EXISTS `seguros`.`modificacion` (  
  
  `idModificacion` INT NOT NULL AUTO_INCREMENT,  
  
  `fecha_modificacion` DATE NOT NULL,  
  
  `descripcion` VARCHAR(32) NOT NULL,  
  
  `Agente_de_Seguros_idAsegurador` INT NOT NULL,  
  
  `Poliza_idPoliza` INT NOT NULL,  
  
  PRIMARY KEY (`idModificacion`),  
  
  INDEX `fk_Modificacion_Agente_de_Seguros1_idx`  
  (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,  
  
  INDEX `fk_Modificacion_Poliza1_idx` (`Poliza_idPoliza` ASC) VISIBLE,  
  
  CONSTRAINT `fk_Modificacion_Agente_de_Seguros1`  
  
    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)  
  
    REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),  
  
  CONSTRAINT `fk_Modificacion_Poliza1`  
  
    FOREIGN KEY (`Poliza_idPoliza`)  
  
    REFERENCES `seguros`.`poliza` (`idPoliza`))  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8mb3;
```

```

CREATE TABLE IF NOT EXISTS `seguros`.`pago` (

  `idPago` INT NOT NULL AUTO_INCREMENT,

  `fecha_pago` DATE NOT NULL,

  `monto` FLOAT NOT NULL,

  `mesAPagar` VARCHAR(45) NOT NULL,

  `metodo de pago` VARCHAR(45) NOT NULL,

  `comision` INT NOT NULL,

  `Poliza_idPoliza` INT NOT NULL,

  `Agente_de_Seguros_idAsegurador` INT NOT NULL,

  PRIMARY KEY (`idPago`),

  INDEX `fk_Pago_Poliza1_idx` (`Poliza_idPoliza` ASC) VISIBLE,

  INDEX `fk_Pago_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador` ASC) VISIBLE,

  CONSTRAINT `fk_Pago_Agente_de_Seguros1`

    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)

      REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),

  CONSTRAINT `fk_Pago_Poliza1`

    FOREIGN KEY (`Poliza_idPoliza`)

      REFERENCES `seguros`.`poliza` (`idPoliza`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb3;

```

```

CREATE TABLE IF NOT EXISTS `seguros`.`reclamo` (

  `idReclamo` INT NOT NULL AUTO_INCREMENT,

  `fecha_reclamo` DATE NOT NULL,

  `descripcion` VARCHAR(32) NOT NULL,

  `Cliente_idCliente` INT NOT NULL,

  `Agente_de_Seguros_idAsegurador` INT NOT NULL,

  PRIMARY KEY (`idReclamo`),

  INDEX `fk_Reclamo_Cliente1_idx` (`Cliente_idCliente` ASC) VISIBLE,

  INDEX `fk_Reclamo_Agente_de_Seguros1_idx` (`Agente_de_Seguros_idAsegurador`
ASC) VISIBLE,

  CONSTRAINT `fk_Reclamo_Agente_de_Seguros1`

    FOREIGN KEY (`Agente_de_Seguros_idAsegurador`)

    REFERENCES `seguros`.`agente_de_seguros` (`idAsegurador`),

  CONSTRAINT `fk_Reclamo_Cliente1`

    FOREIGN KEY (`Cliente_idCliente`)

    REFERENCES `seguros`.`cliente` (`idCliente`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb3;

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

## Trigger:

```
DELIMITER //
```

```
CREATE TRIGGER IncrementarPuntosCompania
```

```
AFTER INSERT ON poliza
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE cliente_id INT;
```

```
    SELECT Cliente_idCliente INTO cliente_id
```

```
    FROM poliza
```

```
    WHERE idPoliza = NEW.idPoliza;
```

```
    UPDATE cliente
```

```
    SET puntosCompania = puntosCompania + 1
```

```
    WHERE idCliente = cliente_id;
```

```
END;
```

```
//
```

```
DELIMITER ;
```

## Insert:

```
INSERT INTO `seguros`.`provincia` (`idProvincia`, `nombreProvincia`)
```

```
VALUES
```

```
(1, 'Buenos Aires'),
```

```
(2, 'CABA');
```

```
select * from provincia;
```

```
INSERT INTO `seguros`.`ciudades` (`idCiudad`, `nombreCiudad`,  
`Provincia_idProvincia`)
```

```
VALUES
```

```
(1, 'Ballester', 1),
```

```
(2, 'Belgrano', 2);
```

```
select * from ciudades;
```

```
INSERT INTO `seguros`.`calle` (`idNumeroCalle`, `numero`, `calle`)
```

```
VALUES
```

```
(1, '5354', 'Pacifico Rodriguez'),
```

```
(2, '893', 'Arce');
```

```
select * from calle;
```



```
INSERT INTO `seguros`.`direccion` (`idDireccion`, `calle_idNumeroCalle`,  
`ciudades_idCiudad`)
```

```
VALUES
```

```
(1, 1, 1),
```

```
(2, 2, 2);
```

```
select * from direccion;
```

```
INSERT INTO `seguros`.`persona` (`idPersona`, `nombre`, `apellido`, `telefono`,  
`direcciones_idDireccion`)
```

```
VALUES
```

```
(1, 'Matias', 'Marzorat', '11 2513 0914', 1),
```

```
(2, 'Gisela', 'Perugorria', '11 6453 0914', 1),
```

```
(3, 'Valen', 'Werle', '11 0000 1111', 2);
```

```
select * from persona;
```

```
INSERT INTO `seguros`.`agente_de_seguros` (`idAsegurador`, `persona_idPersona`)
```

```
VALUES
```

```
(1, 1),
```

```
(2, 2);
```

```
select * from agente_de_seguros;
```

```
INSERT INTO `seguros`.`cliente` (`idCliente`, `puntosCompania`,  
`persona_idPersona`)
```

```
VALUES
```

```
(1,0, 3);
```

```
select * from cliente;
```

```
INSERT INTO `seguros`.`cobertura` (`idCobertura`, `tipo_cobertura`,  
`descripcion`)
```

```
VALUES
```

```
(1, 'Todo Riesgo', 'Cobertura ante todo riesgo'),
```

```
(2, 'Premium', 'Cobertura premium');
```

```
select * from cobertura;
```

```
INSERT INTO `seguros`.`poliza` (`idPoliza`, `numero_de_poliza`,  
`fecha_de_emision`, `fecha_de_vencimiento`, `Cliente_idCliente`,  
`Agente_de_Seguros_idAsegurador`, `cobertura_idCobertura`)
```

```
VALUES (1, 1001, '2023-09-15', '2024-09-15', 1, 1, 1),
```

```
(2, 2002, '2023-09-16', '2024-09-16', 1, 2, 2),
```

```
(3, 3003, '2023-09-17', '2024-09-17', 1, 2, 1);
```

```
select * from poliza;
```

```
INSERT INTO `seguros`.`modificacion` (`idModificacion`, `fecha_modificacion`,  
`descripcion`, `Agente_de_Seguros_idAsegurador`, `Poliza_idPoliza`)
```

```
VALUES (1, '2023-09-15', 'Alteracion de precios', 1, 1),
```

```
(2, '2023-09-16', 'alteracion de vehiculo', 2, 2),
```

```
(3, '2023-09-17', 'Modificación 3', 1, 3);
```

```
select * from modificacion;
```

```
INSERT INTO `seguros`.`pago` (`idPago`, `fecha_pago`, `monto`, `mesAPagar`,  
`metodo de pago`, `comision`, `Poliza_idPoliza`,  
`Agente_de_Seguros_idAsegurador`)
```

```
VALUES (1, '2023-09-15', 100.0, 'Septiembre', 'efec',10, 1, 1),
```

```
(2, '2023-09-16', 200.0, 'Septiembre', 'efec',10, 2, 2),
```

```
(3, '2023-09-17', 300.0, 'Septiembre', 'efec',10, 3, 1);
```

```
select * from pago;
```

```
INSERT INTO `seguros`.`reclamo` (`idReclamo`, `fecha_reclamo`, `descripcion`,  
`Cliente_idCliente`, `Agente_de_Seguros_idAsegurador`)
```

```
VALUES (1, '2023-09-15', 'Reclamo 1', 1, 1),
```

```
(2, '2023-09-16', 'Reclamo 2', 1, 2),
```

```
(3, '2023-09-17', 'Reclamo 3', 1, 1);
```

```
select * from reclamo;
```

## Procedures (Cliente):

DELIMITER //

CREATE PROCEDURE GetReclamosPorCliente(IN clienteID INT)

BEGIN

SELECT r.\*

FROM reclamo r

WHERE r.Cliente\_idCliente = clienteID;

END //

DELIMITER ;

CALL GetReclamosPorCliente(1);

DELIMITER //

CREATE PROCEDURE GetPolizasPorCliente(IN clienteID INT)

BEGIN

SELECT p.\*

FROM poliza p

WHERE p.Cliente\_idCliente = clienteID;

END //

DELIMITER ;

call GetPolizasPorCliente(1);

DELIMITER //

```
CREATE PROCEDURE GetPagosPorCliente(IN clienteID INT)

BEGIN

    SELECT pa.*

    FROM pago pa

    JOIN poliza po ON pa.Poliza_idPoliza = po.idPoliza

    WHERE po.Cliente_idCliente = clienteID;

END //

DELIMITER ;
```

## **Procedures (Agente):**

```
DELIMITER //

CREATE PROCEDURE GetReclamosPorAgente(IN agenteID INT)

BEGIN

    SELECT r.*

    FROM reclamo r

    WHERE r.Agente_de_Seguros_idAsegurador = agenteID;

END //

DELIMITER ;

call GetReclamosPorAgente(1);

call GetReclamosPorAgente(2);

DELIMITER //
```

```
CREATE PROCEDURE GetPolizasPorAgente(IN agenteID INT)
```

```
BEGIN
```

```
    SELECT p.*
```

```
    FROM poliza p
```

```
    WHERE p.Agente_de_Seguros_idAsegurador = agenteID;
```

```
END //
```

```
DELIMITER ;
```

```
call GetPolizasPorAgente(1);
```

```
call GetPolizasPorAgente(2);
```

```
DELIMITER //
```

```
CREATE PROCEDURE GetPagosPorAgente(IN agenteID INT)
```

```
BEGIN
```

```
    SELECT pa.*
```

```
    FROM pago pa
```

```
    WHERE pa.Agente_de_Seguros_idAsegurador = agenteID;
```

```
END //
```

```
DELIMITER ;
```

## Function:

DELIMITER //

CREATE FUNCTION CalcularComisionAgente(

    p\_idAgente INT

)

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

    DECLARE comision\_total DECIMAL(10, 2) DEFAULT 0.0;

    DECLARE done INT DEFAULT FALSE;

    DECLARE monto\_pago DECIMAL(10, 2); -- Declarar las variables antes del bucle

    DECLARE comision\_porcentaje DECIMAL(5, 2);

    DECLARE cur CURSOR FOR

        SELECT monto, comision

        FROM pago

        WHERE Agente\_de\_Seguros\_idAsegurador = p\_idAgente;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read\_loop: LOOP

        FETCH cur INTO monto\_pago, comision\_porcentaje;

        IF done THEN

            LEAVE read\_loop;

        END IF;

```
        SET comision_total = comision_total + (monto_pago * comision_porcentaje  
/ 100);
```

```
    END LOOP;
```

```
    CLOSE cur;
```

```
    RETURN comision_total;
```

```
END;
```

```
//
```

```
DELIMITER ;
```