# Startups de Impacto Social en Perú

# ÍNDICE


I. Introducción……3
II. Descripción general del problema……3
III. Sobre la data……3
IV. Objetivo del proyecto……3
V. Modelo Entidad Relación……4
VI. Diccionario de Datos……5
VII. Implementación de la base de datos……10
VIII. Consultas-Optimización y Experimentación……23
IX. Indices……33

# 1. Introducción

Este proyecto busca diseñar e implementar una base de datos que permita registrar, organizar y analizar información sobre startups peruanas que generan impacto en distintos sectores como: financiero, educativo, ambiental, salubre, etc. con énfasis en el impacto social. Esta base de datos facilitará la consulta de información relevante para investigadores, inversionistas, organizaciones de apoyo y demás stakeholders.

# 2. Descripción general del problema

En el Perú muchas startups de impacto social carecen de visibilidad y seguimiento sistemático. A pesar de que existen incubadoras, programas de gobierno e inversionistas interesados; no hay una plataforma unificada que registre su información de manera estructurada ni su evolución en el tiempo.

La base de datos permitirá registrar las startups desde su etapa inicial (pre-seed) hasta su expansión o salida del mercado, así como su relación con objetivos de desarrollo sostenible (ODS), participación de personas clave, y fuentes de financiamiento. Será útil para estudios de impacto, decisiones de inversión, y diseño de políticas públicas.

La información sobre las startups que se están desarrollando, tal cual lo habíamos comentando anteriormente, no se encuentran en un lugar específico; pero se puede encontrar en páginas web del gobierno peruano como ProInnovate-Ministerio de Producción información relativa a financiamientos y consejería en general. Ahora, lamentablemente la información es limitada; es decir no se puede conocer sobre la etapa en la que está, ni sus participantes, entre otros. Además, para el proyecto actual es necesario realizar consultas en escenarios con 1000 datos, 10 000 datos y 100 000 datos, los cuales serán generados de manera aleatoria respetando las especificaciones de nuestras entidades y relaciones.

# 3. Sobre la data

La data fue creada de manera híbrida (PostgreSQL + Faker). Se usaron sentencias SQL mediante INSERTS, generación de series y reglas de negocio; de la misma manera se usó Python a través de la librería FAKER con el propósito de simular información más realista (nombres, fechas, montos, relaciones, etc.).
A pesar de ello el foco de este proyecto está en el diseño relacional, consultas analíticas y optimización con índices; pero de igual manera se adjuntarán los scripts de las creaciones de la base de datos, los cuales únicamente se incluyen con fines de reproducibilidad.

# 4. Objetivo del Proyecto

Los principales objetivos del proyecto son:

-Diseño de una base de datos que represente adecuadamente las múltiples relaciones entre startups, personas, organizaciones, inversiones e impacto social; con un enfoque estructurado que permita un análisis integral del ecosistema emprendedor peruano.

-Implementación de un modelo diseñado en PostgreSQL que permita realizar consultas complejas orientadas a la toma de decisiones por parte de actores como investigadores, instituciones públicas,

organizaciones de apoyo y potenciales inversionistas.

-Optimizar los tiempos de respuesta de las consultas a la base de datos para escalar esta misma a grandes cantidades de datos (1k,10k,100k)

# 5. Modelo Entidad-Relación

## 5.1 Reglas Semánticas

Cada persona tiene un ID único (DNI), nombre completo, edad y profesión. Una persona puede ser o bien un fundador o un participante o un inversionista. Un fundador presenta un rol (CEO en su mayoría al ser startups), el participante tiene una función y el inversionista aporta un capital.

Un fundador puede fundar muchas startups, mientras que una startup puede ser fundada por muchos fundadores. Asimismo, un participante puede participar en muchas startups y en una startup pueden participar muchos participantes. Cada fundación y participación tiene una fecha que denota su inicio de dicha acción.

Cada instancia de la entidad Fundador debe estar asociada obligatoriamente con al menos una instancia de la relación StartUp. No existen instancias de Fundador que estén fuera de esta relación. De igual manera, cada instancia de la entidad Participante debe estar asociada obligatoriamente con al menos una instancia de la relación StartUp. No existen instancias de Participante que estén fuera de esta relación.

Una StartUp tiene un ID único, nombre y sector. Una startup impacta en muchas ODS (objetivo de desarrollo sostenible), mientras que una ODS puede impactar en muchas StartUps. Una ODS tiene un código identificador ID ODS.

Las StartUps tienen diversas fases o etapas: Pre-Seed, Seed, Early-Stage, Growth-Stage, Expansion-Stage y Exit. Cada una de las fases tiene su propio código identificador y se añade el flujo de caja en la fase Growth. Cada instancia de una fase “posterior” debe estar asociada obligatoriamente con al menos una instancia de su fase predecesora (ejem. cada StartUp que haya pasado o “sobrevivido” a Seed debe estar relacionada con las de Pre-Seed). Además, no existe la entidad Pre-seed sin un StartUp.

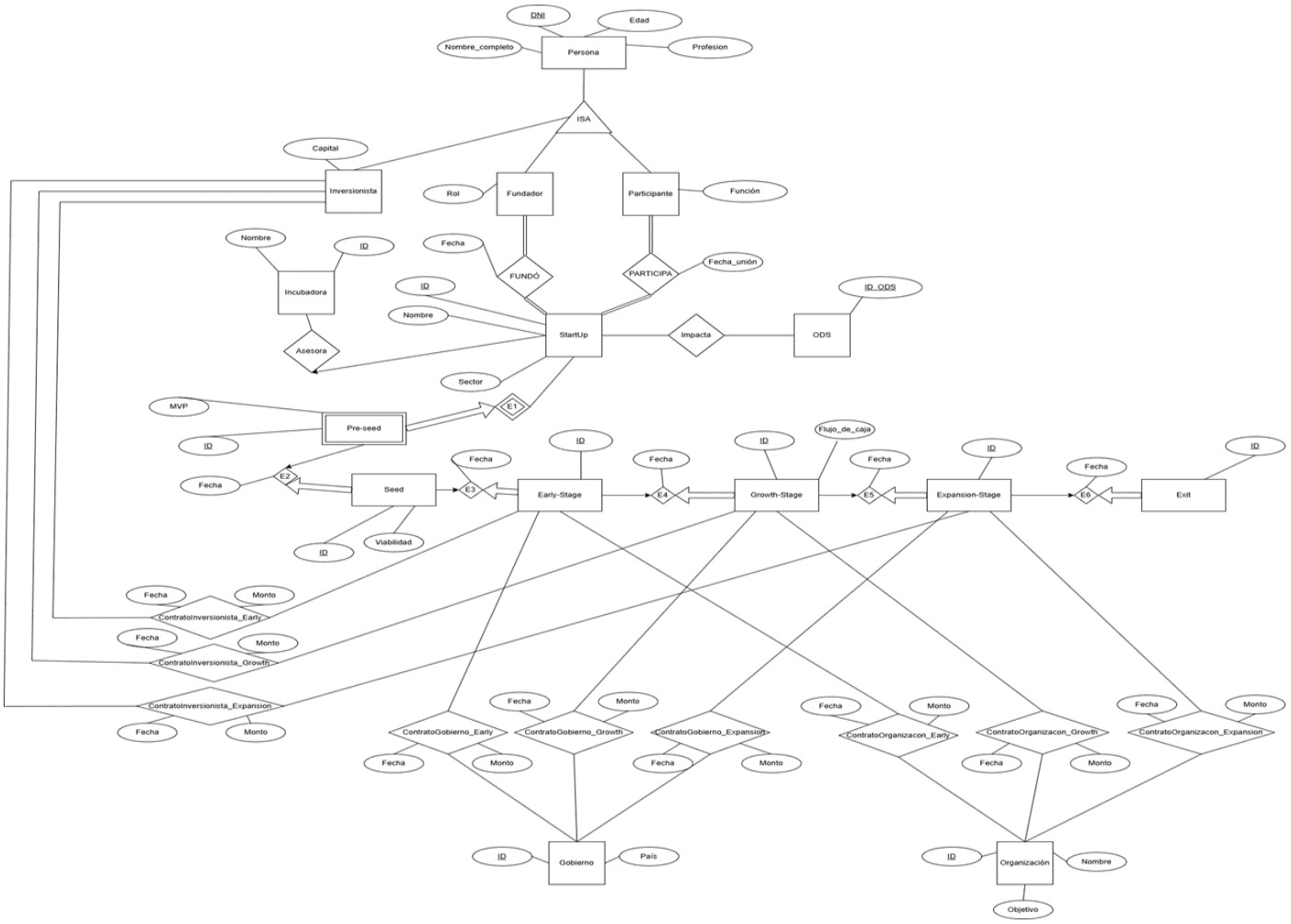
Toda aquella persona, empresa o entidad que desee invertir en una StartUp lo puede hacer solo en sus fases Early, Growth y Expansion.

Un inversionista puede hacer muchas inversiones en StartUps. En una StartUp puede haber inversiones de muchos inversionistas. Cada inversión tiene su respectiva fecha y monto.

Aparte de los inversionistas están también el Gobierno y las Organizaciones. Los gobiernos tienen su propio ID y país, mientras que las organizaciones tienen su ID, nombre y objetivo.

Una incubadora tiene un identificador único y un nombre. Una incubadora puede asesorar a muchas startups, pero una startup solo puede ser asesorada por una incubadora.

## 5.2 Modelo Entidad-Relación




# 6. Diccionario de datos

## 6.1 Entidades

### Organización

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Organizacion | INT | Si | No | Identificador único de la organización |
| Nombre | VARCHAR(50) | No | No | Nombre de la organización |
| Objetivo | VARCHAR(100) | No | No | Objetivo principal de la organización |

### Gobierno

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Gobierno | INT | Si | No | Identificador del gobierno |
| Pais | VARCHAR(50) | No | No | País correspondiente al gobierno |

### Exit

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Exit | INT | Si | No | Identificador de la etapa de exit |
| Id_Expansion_Stage | INT | No | Sí | Etapa de expansión asociada |

### Expansion-Stage

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Expansion-Stage | INT | Si | No | Identificador único de la etapa expansion |
| Id_Growth-Stage | INT | No | Sí | Etapa de crecimiento asociada |

### Growth-Stage

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Growth-Stage | INT | Si | No | Identificador de la etapa growth |
| Id_Early_Stage | INT | No | Sí | Etapa early correspondiente |
| Flujo de caja | DECIMAL | No | No | Flujo de caja reportado |

### Early-stage

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Early-Stage | INT | Si | No | Identificador etapa early |
| Id_Seed | INT | No | Sí | Etapa seed correspondiente |

### Seed

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Seed-Stage | INT | Si | No | Identificador etapa seed |
| Id_Pre-seed | INT | No | Sí | Etapa pre-seed vinculada |
| ID_StartUp | INT | No | Sí | StartUp correspondiente |
| Viabilidad | VARCHAR(50) | No | No | Nivel de viabilidad del proyecto |

### Incubadora

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Incubadora | INT | Sí | No | Identificador de la incubadora |
| Nombre | VARCHAR(50) | No | No | Nombre de la incubadora |

### StartUp

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_StartUp | INT | Si | No | Identificador único de la StartUp |
| Id_Incubadora | INT | No | Sí | Incubadora relacionada |
| Nombre | VARCHAR(50) | No | No | Nombre de la StartUp |
| Sector | VARCHAR(50) | No | No | Sector en el que opera |

### ODS

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_ODS | INT | Sí | No | Identificador del objetivo de desarrollo sostenible |

## 6.2 Entidades débiles

### Pre-seed

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Id_Pre-seed | INT | Si | No | Llave parcial e identificador de etapa pre-seed |
| Id_StartUp | INT | Si | Sí | Identificador de la startup que identifica a pre-seed |
| MVP | VARCHAR(500) | No | No | Modelo de producto mínimo viable |

## 6.3 SuperClases

### Persona

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| DNI | INT | Si | No | Documento de identificación |
| Edad | INT | No | No | Edad de la persona |
| Profesion | VARCHAR(50) | No | No | Profesión de la persona |
| Nombre completo | VARCHAR(100) | No | No | Nombre completo de la persona |

## 6.4 Subclases

### Fundador

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Persona.DNI | INT | Sí | Sí | Documento de identificación |
| Rol | VARCHAR(50) | No | No | Rol actual del fundador en la startup |

### Participante

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Persona.DNI | INT | Sí | Sí | Documento de identificación |
| Funcion | VARCHAR(50) | No | No | Función del participante en el startup |

Inversionista

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Persona.DNI | INT | Sí | Sí | Documento de identificación |
| Capital | INT | No | No | Capital invertido en la startup |

## 6.5 Relaciones binarias

### ContratoInversionista_Growth

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Persona.DNI | INT | Si | Si | Identificador del inversionista |
| ID_Growth-Stage | INT | Sí | Sí | Etapa growth de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto de la inversión |

### ContratoInversionista_Expansion

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| Persona.DNI | INT | Si | Si | Identificador del inversionista |
| ID_Expansion-Stage | INT | Sí | Sí | Etapa expansión de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto de la inversión |

### ContratoOrganizacion_Expansion

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Organización | INT | Si | Si | Organización que financia |
| ID_Expansion-Stage | INT | Sí | Sí | Etapa expansión de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### ContratoOrganizacion_Growth

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Organización | INT | Si | Si | Organización que financia |
| ID_Growth-Stage | INT | Sí | Sí | Etapa growth de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### ContratoOrganizacion_Early

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Organización | INT | Si | Si | Organización que financia |
| ID_Early-Stage | INT | Sí | Sí | Etapa early de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### ContratoGobierno_Expansion

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Gobierno | INT | Si | Si | Gobierno que financia |
| ID_Expansion-stage | INT | Sí | Sí | Etapa expansión de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### ContratoGobierno_Growth

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Gobierno | INT | Si | Si | Gobierno que financia |
| ID_Growth-stage | INT | Sí | Sí | Etapa growth de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### ContratoGobierno_Early

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_Gobierno | INT | Si | Si | Gobierno que financia |
| ID_Early-stage | INT | Sí | Sí | Etapa early de la startup |
| Fecha | DATE | No | No | Fecha del contrato |
| Monto | DECIMAL | No | No | Monto del financiamiento |

### Impacta

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_StartUp | INT | Sí | Sí | StartUp que impacta en un ODS |
| ID_ODS | INT | Sí | Sí | Objetivo de desarrollo sostenible |

### Fundo

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_StartUp | INT | Sí | Sí | Identificador de la startup fundada |
| Persona.DNI | INT | Sí | Sí | Persona que funda |
| Fecha | DATE | No | No | Fecha de fundación |

**Participa**

| Campo | Tipo | PK | FK | Descripción |
|---|---|---|---|---|
| ID_StartUp | INT | Sí | Sí | StartUp en la que participa |
| Persona.DNI | INT | Sí | Sí | Participante en la StartUp |
| Fecha_union | DATE | No | No | Fecha en que se unió |

# 7. Implementación de la base de datos

## 7.1 Creación de Tablas en PostgreSQL:

```
--creaciones de tablas

CREATE TABLE Organizacion
 (
    ID_Organizacion INT PRIMARY KEY ,
    nombre VARCHAR (50) ,
    objetivo VARCHAR (100)
);

CREATE TABLE Gobierno
(
 ID_Gobierno INT PRIMARY KEY ,
 pais VARCHAR (50)
 );

 CREATE TABLE ODS
 (
    ID_ODS INT PRIMARY KEY
 );

 CREATE TABLE Incubadora
 (
    ID_Incubadora INT PRIMARY KEY ,
    nombre VARCHAR (50)
 );
```

```sql
CREATE TABLE StartUp
(
   ID_StartUp INT PRIMARY KEY ,
   ID_Incubadora INT ,
   nombre VARCHAR (50) ,
   sector VARCHAR (50) ,
   FOREIGN KEY ( ID_Incubadora ) REFERENCES Incubadora ( ID_Incubadora )
);

CREATE TABLE Persona
(
    DNI INT PRIMARY KEY ,
    edad INT ,
    profesion VARCHAR (50) ,
    nombre_completo VARCHAR (100)
);

CREATE TABLE Fundador
 (
   DNI INT PRIMARY KEY ,
   rol VARCHAR (50) ,
   FOREIGN KEY (DNI ) REFERENCES Persona (DNI)
  );


CREATE TABLE Participante
 (
    DNI INT PRIMARY KEY ,
    funcion VARCHAR (50) ,
    FOREIGN KEY (DNI) REFERENCES Persona (DNI)
);

CREATE TABLE Inversionista
 (
    DNI INT PRIMARY KEY,
    capital INT,
    FOREIGN KEY (DNI) REFERENCES Persona (DNI)
);

CREATE TABLE Pre_Seed
  (
    ID_Pre_Seed INT PRIMARY KEY ,
    ID_StartUp INT ,
    MVP VARCHAR (500) ,
    FOREIGN KEY (ID_StartUp) REFERENCES StartUp (ID_StartUp)
    );
```

```sql
CREATE TABLE Seed
    (
        ID_Seed INT PRIMARY KEY ,
        ID_Pre_Seed INT ,
        ID_StartUp INT ,
        viabilidad VARCHAR (50) ,
        FOREIGN KEY ( ID_Pre_Seed ) REFERENCES Pre_Seed ( ID_Pre_Seed ),
        FOREIGN KEY ( ID_StartUp ) REFERENCES StartUp ( ID_StartUp )
    );

CREATE TABLE Early_Stage
(
    ID_Early_Stage INT PRIMARY KEY ,
    ID_Seed INT ,
    FOREIGN KEY (ID_Seed) REFERENCES Seed (ID_Seed)
);

CREATE TABLE Growth_Stage
    (
    ID_Growth_Stage INT PRIMARY KEY ,
    flujo_de_caja DECIMAL ,
    ID_Early_Stage INT ,
    FOREIGN KEY ( ID_Early_Stage ) REFERENCES Early_Stage ( ID_Early_Stage )
    );

CREATE TABLE Expansion_Stage
 (
    ID_Expansion_Stage INT PRIMARY KEY ,
    ID_Growth_Stage INT ,
    FOREIGN KEY ( ID_Growth_Stage ) REFERENCES Growth_Stage ( ID_Growth_Stage )
 );

CREATE TABLE Exit
 (
    ID_Exit INT PRIMARY KEY ,
    ID_Expansion_Stage INT ,
    FOREIGN KEY ( ID_Expansion_Stage ) REFERENCES Expansion_Stage (
    ID_Expansion_Stage )
 );

 CREATE TABLE ContratoGobierno_Early
 (
    ID_Gobierno INT ,
    ID_Early_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY ( ID_Gobierno , ID_Early_Stage ),
    FOREIGN KEY ( ID_Gobierno ) REFERENCES Gobierno ( ID_Gobierno ),
    FOREIGN KEY ( ID_Early_Stage ) REFERENCES Early_Stage ( ID_Early_Stage )
 );
```

```sql
CREATE TABLE ContratoGobierno_Growth
(
    ID_Gobierno INT ,
    ID_Growth_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY ( ID_Gobierno , ID_Growth_Stage ),
    FOREIGN KEY ( ID_Gobierno ) REFERENCES Gobierno ( ID_Gobierno ),
    FOREIGN KEY ( ID_Growth_Stage ) REFERENCES Growth_Stage ( ID_Growth_Stage )
    );

CREATE TABLE ContratoGobierno_Expansion
    (
        ID_Gobierno INT ,
        ID_Expansion_Stage INT ,
        fecha DATE ,
        monto DECIMAL ,
        PRIMARY KEY ( ID_Gobierno , ID_Expansion_Stage ),
        FOREIGN KEY ( ID_Gobierno ) REFERENCES Gobierno ( ID_Gobierno ),
        FOREIGN KEY ( ID_Expansion_Stage ) REFERENCES Expansion_Stage (
        ID_Expansion_Stage )
        );


CREATE TABLE ContratoOrganizacion_Early
(
    ID_Organizacion INT ,
    ID_Early_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY ( ID_Organizacion , ID_Early_Stage ),
    FOREIGN KEY ( ID_Organizacion ) REFERENCES Organizacion ( ID_Organizacion ),
    FOREIGN KEY ( ID_Early_Stage ) REFERENCES Early_Stage ( ID_Early_Stage )
    );

CREATE TABLE ContratoOrganizacion_Growth
(
    ID_Organizacion INT ,
    ID_Growth_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY ( ID_Organizacion , ID_Growth_Stage ),
    FOREIGN KEY ( ID_Organizacion ) REFERENCES Organizacion ( ID_Organizacion ),
    FOREIGN KEY ( ID_Growth_Stage ) REFERENCES Growth_Stage ( ID_Growth_Stage )
 );
```

```sql
CREATE TABLE ContratoOrganizacion_Expansion
 (
    ID_Organizacion INT ,
    ID_Expansion_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY ( ID_Organizacion , ID_Expansion_Stage ),
    FOREIGN KEY ( ID_Organizacion ) REFERENCES Organizacion ( ID_Organizacion ),
    FOREIGN KEY ( ID_Expansion_Stage ) REFERENCES Expansion_Stage (
    ID_Expansion_Stage )
 );

 CREATE TABLE ContratoInversionista_Early
(
    DNI INT ,
    ID_Early_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY (DNI , ID_Early_Stage ),
    FOREIGN KEY (DNI ) REFERENCES Persona (DNI ),
    FOREIGN KEY ( ID_Early_Stage ) REFERENCES Early_Stage ( ID_Early_Stage )
 );
```

```sql
 CREATE TABLE ContratoInversionista_Growth
 (
    DNI INT ,
    ID_Growth_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY (DNI , ID_Growth_Stage ),
    FOREIGN KEY (DNI ) REFERENCES Persona (DNI ),
    FOREIGN KEY ( ID_Growth_Stage ) REFERENCES Growth_Stage ( ID_Growth_Stage )
 );

 CREATE TABLE ContratoInversionista_Expansion
(
    DNI INT ,
    ID_Expansion_Stage INT ,
    fecha DATE ,
    monto DECIMAL ,
    PRIMARY KEY (DNI , ID_Expansion_Stage ),
    FOREIGN KEY (DNI ) REFERENCES Persona (DNI ),
    FOREIGN KEY ( ID_Expansion_Stage ) REFERENCES Expansion_Stage (
    ID_Expansion_Stage )
 );
```

```sql
CREATE TABLE Impacta
(
    ID_StartUp INT ,
    ID_ODS INT ,
    PRIMARY KEY ( ID_StartUp , ID_ODS ),
    FOREIGN KEY ( ID_StartUp ) REFERENCES StartUp ( ID_StartUp ),
    FOREIGN KEY ( ID_ODS ) REFERENCES ODS ( ID_ODS )
 );

CREATE TABLE Fundo
(
    ID_StartUp INT ,
    DNI INT ,
    fecha DATE ,
    PRIMARY KEY ( ID_StartUp , DNI ),
    FOREIGN KEY ( ID_StartUp ) REFERENCES StartUp ( ID_StartUp ),
    FOREIGN KEY (DNI ) REFERENCES Persona (DNI )
 );

 CREATE TABLE Participa
 (
    ID_StartUp INT ,
    DNI INT ,
    fecha_union DATE ,
    PRIMARY KEY ( ID_StartUp , DNI ),
    FOREIGN KEY ( ID_StartUp ) REFERENCES StartUp ( ID_StartUp ),
    FOREIGN KEY (DNI ) REFERENCES Persona (DNI )
 );
```

```sql
--alter table, checks,constraints

-- Agregamos fechas en cada etapa
ALTER TABLE seed ADD COLUMN fecha DATE;
ALTER TABLE early_stage ADD COLUMN fecha DATE;
ALTER TABLE growth_stage ADD COLUMN fecha DATE;
ALTER TABLE expansion_stage ADD COLUMN fecha DATE;
ALTER TABLE exit ADD COLUMN fecha DATE;


-- Fundación de Startup (fecha no futura)
ALTER TABLE fundo
ADD CONSTRAINT chk_fundo_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

-- Participación persona-startup (fecha de unión no futura)
ALTER TABLE participa
ADD CONSTRAINT chk_participa_fecha_union_no_futura
CHECK (fecha_union <= CURRENT_DATE);

-- Etapas de crecimiento (fecha no futura)

ALTER TABLE seed
ADD CONSTRAINT chk_seed_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE early_stage
ADD CONSTRAINT chk_early_stage_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);
```

```
ALTER TABLE growth_stage
ADD CONSTRAINT chk_growth_stage_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE expansion_stage
ADD CONSTRAINT chk_expansion_stage_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE exit
ADD CONSTRAINT chk_exit_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);


-- Contratos (fecha de firma no futura)


ALTER TABLE contratoinversionista_early
ADD CONSTRAINT chk_cinv_early_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratoinversionista_growth
ADD CONSTRAINT chk_cinv_growth_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratoinversionista_expansion
ADD CONSTRAINT chk_cinv_expansion_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratogobierno_early
ADD CONSTRAINT chk_cgob_early_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratogobierno_growth
ADD CONSTRAINT chk_cgob_growth_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratogobierno_expansion
ADD CONSTRAINT chk_cgob_expansion_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratoorganizacion_early
ADD CONSTRAINT chk_corg_early_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratoorganizacion_growth
ADD CONSTRAINT chk_corg_growth_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);

ALTER TABLE contratoorganizacion_expansion
ADD CONSTRAINT chk_corg_expansion_fecha_no_futura
CHECK (fecha <= CURRENT_DATE);
```

Ahora vamos con los triggers: se valida que la fecha al pasar de una etapa a otra sea mayor que la fecha a la etapa anterior

```
-- Fundo -> Seed
CREATE OR REPLACE FUNCTION validarFechaDeFundoSeed()
RETURNS TRIGGER AS $$
DECLARE
  fechaFundo DATE;
BEGIN
  SELECT f.fecha
    INTO fechaFundo
  FROM fundo f
  WHERE f.id_startup = NEW.id_startup;

  IF fechaFundo IS NULL THEN
    RAISE EXCEPTION 'No existe registro en FUNDO para la startup %', NEW.id_startup;
  END IF;

  IF NEW.fecha <= fechaFundo THEN
    RAISE EXCEPTION 'La fecha en Seed debe ser mayor que la fecha de Fundo (startup %)', NEW.id_startup;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarFechaDeFundoSeed ON seed;

CREATE TRIGGER trigger_validarFechaDeFundoSeed
BEFORE INSERT OR UPDATE ON seed
FOR EACH ROW
EXECUTE FUNCTION validarFechaDeFundoSeed();
```

```
-- Seed -> EarlyStage (early_stage tiene id_seed)
CREATE OR REPLACE FUNCTION validarFechaDeSeedEarlyStage()
RETURNS TRIGGER AS $$
DECLARE
  fechaSeed DATE;
BEGIN
  SELECT s.fecha
    INTO fechaSeed
  FROM seed s
  WHERE s.id_seed = NEW.id_seed;

  IF fechaSeed IS NULL THEN
    RAISE EXCEPTION 'No existe registro SEED con id_seed %', NEW.id_seed;
  END IF;

  IF NEW.fecha <= fechaSeed THEN
    RAISE EXCEPTION 'La fecha en Early Stage debe ser mayor que la fecha de Seed (id_seed %)', NEW.id_seed;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarFechaDeSeedEarlyStage ON early_stage;

CREATE TRIGGER trigger_validarFechaDeSeedEarlyStage
BEFORE INSERT OR UPDATE ON early_stage
FOR EACH ROW
EXECUTE FUNCTION validarFechaDeSeedEarlyStage();
```

```sql
-- EarlyStage -> GrowthStage (growth_stage tiene id_early_stage)
CREATE OR REPLACE FUNCTION validarEarlyStageGrowth()
RETURNS TRIGGER AS $$
DECLARE
  fechaEarly DATE;
BEGIN
  SELECT e.fecha
    INTO fechaEarly
  FROM early_stage e
  WHERE e.id_early_stage = NEW.id_early_stage;

  IF fechaEarly IS NULL THEN
    RAISE EXCEPTION 'No existe registro EARLY_STAGE con id_early_stage %', NEW.id_early_stage;
  END IF;

  IF NEW.fecha <= fechaEarly THEN
    RAISE EXCEPTION 'La fecha en Growth Stage debe ser mayor que la fecha de Early Stage (id_early_stage %)',
      NEW.id_early_stage;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarEarlyStageGrowth ON growth_stage;

CREATE TRIGGER trigger_validarEarlyStageGrowth
BEFORE INSERT OR UPDATE ON growth_stage
FOR EACH ROW
EXECUTE FUNCTION validarEarlyStageGrowth();
```

```sql
-- GrowthStage -> ExpansionStage (expansion_stage tiene id_growth_stage)
CREATE OR REPLACE FUNCTION validarGrowthExpansionStage()
RETURNS TRIGGER AS $$
DECLARE
  fechaGrowth DATE;
BEGIN
  SELECT g.fecha
    INTO fechaGrowth
  FROM growth_stage g
  WHERE g.id_growth_stage = NEW.id_growth_stage;

  IF fechaGrowth IS NULL THEN
    RAISE EXCEPTION 'No existe registro GROWTH_STAGE con id_growth_stage %', NEW.id_growth_stage;
  END IF;

  IF NEW.fecha <= fechaGrowth THEN
    RAISE EXCEPTION 'La fecha en Expansion Stage debe ser mayor que la fecha de Growth (id_growth_stage %)',
      NEW.id_growth_stage;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarGrowthExpansionStage ON expansion_stage;

CREATE TRIGGER trigger_validarGrowthExpansionStage
BEFORE INSERT OR UPDATE ON expansion_stage
FOR EACH ROW
EXECUTE FUNCTION validarGrowthExpansionStage();
```

```sql
-- ExpansionStage -> Exit (exit tiene id_expansion_stage)
CREATE OR REPLACE FUNCTION validarExpansionStageExit()
RETURNS TRIGGER AS $$
DECLARE
  fechaExpansion DATE;
BEGIN
  SELECT ex.fecha
    INTO fechaExpansion
  FROM expansion_stage ex
  WHERE ex.id_expansion_stage = NEW.id_expansion_stage;

  IF fechaExpansion IS NULL THEN
    RAISE EXCEPTION 'No existe registro EXPANSION_STAGE con id_expansion_stage %', NEW.id_expansion_stage;
  END IF;

  IF NEW.fecha <= fechaExpansion THEN
    RAISE EXCEPTION 'La fecha en Exit debe ser mayor que la fecha de Expansion Stage (id_expansion_stage %)',
      NEW.id_expansion_stage;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarExpansionStageExit ON exit;

CREATE TRIGGER trigger_validarExpansionStageExit
BEFORE INSERT OR UPDATE ON exit
FOR EACH ROW
EXECUTE FUNCTION validarExpansionStageExit();
```

```sql
-- Viabilidad: Solo Alta/Media/Baja
CREATE OR REPLACE FUNCTION validarViabilidadSeed()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.viabilidad NOT IN ('Alta', 'Media', 'Baja') THEN
    RAISE EXCEPTION 'Valor inválido para viabilidad: solo se permite Alta, Media o Baja.';
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_validarViabilidadSeed ON seed;

CREATE TRIGGER trigger_validarViabilidadSeed
BEFORE INSERT OR UPDATE ON seed
FOR EACH ROW
EXECUTE FUNCTION validarViabilidadSeed();
```

```sql
-- FUNCIONES: PASAR DE ETAPA

-- Pre-seed -> Seed
CREATE OR REPLACE FUNCTION Preseed_A_Seed(id_preseed BIGINT, viability VARCHAR(50))
RETURNS VOID AS $$
DECLARE
  id_startup_result BIGINT;
  nuevo_id_seed BIGINT;
BEGIN
  -- Validar viabilidad
  IF viability NOT IN ('Alta','Media','Baja') THEN
    RAISE EXCEPTION 'Valor inválido para viabilidad: solo Alta, Media o Baja.';
  END IF;

  -- Verificar que exista pre_seed y obtener startup
  SELECT p.id_startup
    INTO id_startup_result
  FROM pre_seed p
  WHERE p.id_pre_seed = id_preseed;

  IF id_startup_result IS NULL THEN
    RAISE EXCEPTION 'No existe registro PRE_SEED con id_pre_seed %', id_preseed;
  END IF;

  -- Evitar duplicidad: una startup no debería tener 2 seeds
  IF EXISTS (SELECT 1 FROM seed s WHERE s.id_startup = id_startup_result) THEN
    RAISE EXCEPTION 'La startup % ya tiene un registro en SEED', id_startup_result;
  END IF;



  -- Generar nuevo id (enfoque actual)
  SELECT COALESCE(MAX(id_seed), 0) + 1
    INTO nuevo_id_seed
  FROM seed;

  INSERT INTO seed (id_seed, id_pre_seed, id_startup, viabilidad, fecha)
  VALUES (nuevo_id_seed, id_preseed, id_startup_result, viability, CURRENT_DATE);
END;
$$ LANGUAGE plpgsql;


-- Seed -> Early Stage
CREATE OR REPLACE FUNCTION Seed_A_EarlyStage(bid_seed BIGINT)
RETURNS VOID AS $$
DECLARE
  nuevo_id_earlystage BIGINT;
  fechaSeed DATE;
BEGIN
  -- Verificar existencia de seed
  SELECT s.fecha
    INTO fechaSeed
  FROM seed s
  WHERE s.id_seed = bid_seed;

  IF fechaSeed IS NULL THEN
    RAISE EXCEPTION 'No existe registro SEED con id_seed %', bid_seed;
  END IF;
```

```
  -- Evitar duplicidad: un seed no debería tener 2 early_stage
  IF EXISTS (SELECT 1 FROM early_stage e WHERE e.id_seed = bid_seed) THEN
    RAISE EXCEPTION 'El SEED % ya tiene un registro en EARLY_STAGE', bid_seed;
  END IF;

  SELECT COALESCE(MAX(id_early_stage), 0) + 1
    INTO nuevo_id_earlystage
  FROM early_stage;

  INSERT INTO early_stage (id_early_stage, id_seed, fecha)
  VALUES (nuevo_id_earlystage, bid_seed, CURRENT_DATE);
END;
$$ LANGUAGE plpgsql;


-- Early Stage -> Growth Stage
CREATE OR REPLACE FUNCTION EarlyStage_A_GrowthStage(bid_EarlyStage BIGINT, flujoCaja DECIMAL)
RETURNS VOID AS $$
DECLARE
  nuevo_id_GrowthStage BIGINT;
  fechaEarly DATE;
BEGIN
  -- Verificar existencia de early_stage
  SELECT e.fecha
    INTO fechaEarly
  FROM early_stage e
  WHERE e.id_early_stage = bid_EarlyStage;


  IF fechaEarly IS NULL THEN
    RAISE EXCEPTION 'No existe registro EARLY_STAGE con id_early_stage %', bid_EarlyStage;
  END IF;

  -- Evitar duplicidad
  IF EXISTS (SELECT 1 FROM growth_stage g WHERE g.id_early_stage = bid_EarlyStage) THEN
    RAISE EXCEPTION 'El EARLY_STAGE % ya tiene un registro en GROWTH_STAGE', bid_EarlyStage;
  END IF;

  SELECT COALESCE(MAX(id_growth_stage), 0) + 1
    INTO nuevo_id_GrowthStage
  FROM growth_stage;

  INSERT INTO growth_stage (id_growth_stage, flujo_de_caja, id_early_stage, fecha)
  VALUES (nuevo_id_GrowthStage, flujoCaja, bid_EarlyStage, CURRENT_DATE);
END;
$$ LANGUAGE plpgsql;


-- Growth Stage -> Expansion Stage
CREATE OR REPLACE FUNCTION GrowthStage_A_ExpansionStage(bid_GrowthStage BIGINT)
RETURNS VOID AS $$
DECLARE
  nuevo_id_ExpansionStage BIGINT;
  fechaGrowth DATE;
```

```
BEGIN
  -- Verificar existencia de growth_stage
  SELECT g.fecha
    INTO fechaGrowth
  FROM growth_stage g
  WHERE g.id_growth_stage = bid_GrowthStage;

  IF fechaGrowth IS NULL THEN
    RAISE EXCEPTION 'No existe registro GROWTH_STAGE con id_growth_stage %', bid_GrowthStage;
  END IF;

  -- Evitar duplicidad
  IF EXISTS (SELECT 1 FROM expansion_stage ex WHERE ex.id_growth_stage = bid_GrowthStage) THEN
    RAISE EXCEPTION 'El GROWTH_STAGE % ya tiene un registro en EXPANSION_STAGE', bid_GrowthStage;
  END IF;

  SELECT COALESCE(MAX(id_expansion_stage), 0) + 1
    INTO nuevo_id_ExpansionStage
  FROM expansion_stage;

  INSERT INTO expansion_stage (id_expansion_stage, id_growth_stage, fecha)
  VALUES (nuevo_id_ExpansionStage, bid_GrowthStage, CURRENT_DATE);
END;
$$ LANGUAGE plpgsql;



-- Expansion Stage -> Exit
CREATE OR REPLACE FUNCTION ExpansionStage_A_Exit(bid_ExpansionStage BIGINT)
RETURNS VOID AS $$
DECLARE
  nuevo_id_Exit BIGINT;
  fechaExpansion DATE;
BEGIN
  -- Verificar existencia de expansion_stage
  SELECT ex.fecha
    INTO fechaExpansion
  FROM expansion_stage ex
  WHERE ex.id_expansion_stage = bid_ExpansionStage;

  IF fechaExpansion IS NULL THEN
    RAISE EXCEPTION 'No existe registro EXPANSION_STAGE con id_expansion_stage %', bid_ExpansionStage;
  END IF;

  -- Evitar duplicidad
  IF EXISTS (SELECT 1 FROM exit e WHERE e.id_expansion_stage = bid_ExpansionStage) THEN
    RAISE EXCEPTION 'El EXPANSION_STAGE % ya tiene un registro en EXIT', bid_ExpansionStage;
  END IF;

  SELECT COALESCE(MAX(id_exit), 0) + 1
    INTO nuevo_id_Exit
  FROM exit;
```

```
  INSERT INTO exit (id_exit, id_expansion_stage, fecha)
  VALUES (nuevo_id_Exit, bid_ExpansionStage, CURRENT_DATE);
END;
$$ LANGUAGE plpgsql;
```

# 8. Consultas-Optimización y Experimentación

En esta sección, se presentan ocho consultas SQL de complejidad media y alta, formuladas específicamente para explorar y evaluar datos relevantes del ecosistema de startups de impacto social en el Perú. Estas consultas serán ejecutadas sobre bases de datos con distintos volúmenes de datos: 1,000; 10,000 y 100,000 registros, con el fin de medir su desempeño y analizar posibles cuellos de botella.
Para mejorar su eficiencia, se implementarán índices sobre tres de dichas con el propósito de comparar medidas de antes vs después de índices.

## 8.1 Implementación de consultas en SQL

Consulta 1: ¿Cuáles son las tres primeras incubadoras que tienen el mayor porcentaje de proyectos que llegaron a la etapa expansión?

Cuando se desea iniciar una startup el paso recomendado es buscar una incubadora, debido a que esta ayuda al proyecto no solo en la parte financiera, sino también en el asesoramiento. Todo esto con el fin de que su proyecto tenga una mayor probabilidad de poder llegar a una etapa avanzada.
La consulta arroja las incubadoras (top 3),su total de startups, la cantidad de las mismas que llegaron a la etapa de 'expansión' y su respectivo porcentaje.

```
WITH total_por_incubadora AS (
  SELECT i.id_incubadora, i.nombre AS incubadora, COUNT(*) AS total_startups
  FROM incubadora i
  JOIN startup s ON s.id_incubadora = i.id_incubadora
  GROUP BY i.id_incubadora, i.nombre
),
exp_por_incubadora AS (
  SELECT i.id_incubadora, COUNT(DISTINCT s.id_startup) AS startups_expansion
  FROM incubadora i
  JOIN startup s ON s.id_incubadora = i.id_incubadora
  JOIN seed sd ON sd.id_startup = s.id_startup
  JOIN early_stage e ON e.id_seed = sd.id_seed
  JOIN growth_stage g ON g.id_early_stage = e.id_early_stage
  JOIN expansion_stage ex ON ex.id_growth_stage = g.id_growth_stage
  GROUP BY i.id_incubadora
)
SELECT
  t.incubadora,
  t.total_startups,
  COALESCE(e.startups_expansion, 0) AS startups_expansion,
  ROUND(100.0 * COALESCE(e.startups_expansion,0) / NULLIF(t.total_startups,0), 2) AS pct_expansio
FROM total_por_incubadora t
LEFT JOIN exp_por_incubadora e ON e.id_incubadora = t.id_incubadora
ORDER BY pct_expansion DESC, startups_expansion DESC, total_startups DESC
LIMIT 3;
```

| | incubadora character varying (50) | total_startups bigint | startups_expansion bigint | pct_expansion numeric |
|---|---|---|---|---|
| 1 | SeedLab | 119 | 2 | 1.68 |
| 2 | UTEC Ventures | 130 | 2 | 1.54 |
| 3 | Wayra | 108 | 1 | 0.93 |

Consulta 2 ¿Cuáles son las startups y sus inversionistas sea persona, gobierno u organización (si es que lo hay) que se encuentran estancadas por más de 6 meses en la etapa early stage?

Para un inversionista de cualquier tipo es necesario saber que startups no pueden pasar de la etapa Early Stage, debido a que permite evaluar de manera más exhaustiva la idea de negocio, cuáles son las debilidades de dichas startups y ver como se le puede apoyar. Identificamos startups que llevan estancadas durante 6 meses sin crecer, lo cual es una señal interesante para 'hacer zoom' ahí y poder determinar en que se está fallando

```
WITH early_stuck AS (
  SELECT
    s.id_startup,
    s.nombre AS startup,
    e.id_early_stage,
    e.fecha AS early_fecha
  FROM early_stage e
  JOIN seed sd ON sd.id_seed = e.id_seed
  JOIN startup s ON s.id_startup = sd.id_startup
  LEFT JOIN growth_stage g ON g.id_early_stage = e.id_early_stage
  WHERE g.id_growth_stage IS NULL
    AND e.fecha <= (CURRENT_DATE - INTERVAL '6 months')
),
inv_persona AS (
  SELECT
    ci.id_early_stage,
    STRING_AGG(DISTINCT (p.nombre_completo || ' (DNI ' || p.dni || ')'), ', ') AS inversionistas_persona
  FROM contratoinversionista_early ci
  JOIN persona p ON p.dni = ci.dni
  GROUP BY ci.id_early_stage
),
inv_gob AS (
  SELECT
    cg.id_early_stage,
    STRING_AGG(DISTINCT (g.pais || ' (ID ' || g.id_gobierno || ')'), ', ') AS inversionistas_gobierno
  FROM contratogobierno_early cg
  JOIN gobierno g ON g.id_gobierno = cg.id_gobierno
  GROUP BY cg.id_early_stage
),
inv_org AS (
  SELECT
    co.id_early_stage,
    STRING_AGG(DISTINCT (o.nombre || ' (ID ' || o.id_organizacion || ')'), ', ') AS inversionistas_organizacion
  FROM contratoorganizacion_early co
  JOIN organizacion o ON o.id_organizacion = co.id_organizacion
  GROUP BY co.id_early_stage
)
SELECT
  es.startup,
  es.early_fecha,
  COALESCE(ip.inversionistas_persona, 'N/A') AS inversionista_persona,
  COALESCE(ig.inversionistas_gobierno, 'N/A') AS inversionista_gobierno,
  COALESCE(io.inversionistas_organizacion, 'N/A') AS inversionista_organizacion
FROM early_stuck es
LEFT JOIN inv_persona ip ON ip.id_early_stage = es.id_early_stage
LEFT JOIN inv_gob ig ON ig.id_early_stage = es.id_early_stage
LEFT JOIN inv_org io ON io.id_early_stage = es.id_early_stage
ORDER BY es.early_fecha ASC, es.startup;
```

Mostrando solo las cinco primeras filas:

| | startup character varying (50) | early_fecha date | inversionista_persona text | inversionista_gobierno text | inversionista_organizacion text |
|---|---|---|---|---|---|
| 1 | Sevillano y Campillo S.L. | 2020-08-26 | N/A | Colombia (ID 3) | N/A |
| 2 | Llopis y asociados S.Coop. | 2020-08-27 | Elisabet Casado (DNI 10003343) | México (ID 4) | N/A |
| 3 | Banca Privada XYE S.Com. | 2020-09-15 | Elisabet Casado (DNI 10003343) | Chile (ID 2) | N/A |
| 4 | Hermanos Infante S.Com. | 2020-10-10 | Elisabet Casado (DNI 10003343) | Bolivia (ID 8) | N/A |
| 5 | Restauración Hoyos S.L. | 2020-10-10 | Elisabet Casado (DNI 10003343) | N/A | N/A |

Consulta 3 ¿Qué startups han recibido el mayor monto total de inversión y en qué etapas?

En el ecosistema emprendedor peruano, uno de los factores más determinantes para el crecimiento de una startup es la cantidad de inversión recibida y la etapa en la que esta inversión se concentra. Esta información resulta clave tanto para los inversionistas actuales como para entidades gubernamentales y organizaciones que buscan apoyar emprendimientos con alto potencial de impacto social.

Dado que las inversiones pueden realizarse en diferentes etapas del crecimiento de una startup —como Early Stage, Growth Stage y Expansion Stage—, es fundamental analizar no solo el monto total acumulado por cada emprendimiento, sino también identificar en qué etapas se ha concentrado dicha inversión. Esto permitirá identificar patrones de financiamiento, comparar trayectorias de crecimiento y focalizar políticas de apoyo según el ciclo de vida de las startups.

```
WITH inv AS (
  -- EARLY (persona/gob/org)
  SELECT sd.id_startup, 'Early' AS etapa, ci.monto
  FROM contratoinversionista_early ci
  JOIN early_stage e ON e.id_early_stage = ci.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Early', cg.monto
  FROM contratogobierno_early cg
  JOIN early_stage e ON e.id_early_stage = cg.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Early', co.monto
  FROM contratoorganizacion_early co
  JOIN early_stage e ON e.id_early_stage = co.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed

  -- GROWTH
  UNION ALL
  SELECT sd.id_startup, 'Growth', ci.monto
  FROM contratoinversionista_growth ci
  JOIN growth_stage g ON g.id_growth_stage = ci.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Growth', cg.monto
  FROM contratogobierno_growth cg
  JOIN growth_stage g ON g.id_growth_stage = cg.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Growth', co.monto
  FROM contratoorganizacion_growth co
  JOIN growth_stage g ON g.id_growth_stage = co.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
```

```sql
  -- EXPANSION
  UNION ALL
  SELECT sd.id_startup, 'Expansion', ci.monto
  FROM contratoinversionista_expansion ci
  JOIN expansion_stage ex ON ex.id_expansion_stage = ci.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Expansion', cg.monto
  FROM contratogobierno_expansion cg
  JOIN expansion_stage ex ON ex.id_expansion_stage = cg.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, 'Expansion', co.monto
  FROM contratoorganizacion_expansion co
  JOIN expansion_stage ex ON ex.id_expansion_stage = co.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
),
tot AS (
  SELECT id_startup, SUM(monto) AS monto_total
  FROM inv
  GROUP BY id_startup
),
top10 AS (
  SELECT id_startup, monto_total
  FROM tot
  ORDER BY monto_total DESC
  LIMIT 10
),
by_stage AS (
  SELECT id_startup, etapa, SUM(monto) AS monto_etapa
  FROM inv
  GROUP BY id_startup, etapa
)
SELECT
  s.nombre AS startup,
  t.monto_total,
  STRING_AGG(bs.etapa || ': ' || ROUND(bs.monto_etapa::numeric,2), ' | ' ORDER BY bs.monto_etapa DESC) AS detalle_por_
FROM top10 t
JOIN startup s ON s.id_startup = t.id_startup
JOIN by_stage bs ON bs.id_startup = t.id_startup
GROUP BY s.nombre, t.monto_total
ORDER BY t.monto_total DESC;
```

Mostrando las cinco primeras filas:

| | startup<br>character varying (50) | monto_total<br>numeric | detalle_por_etapa<br>text |
|---|---|---|---|
| 1 | Tamara Bonilla Valencia S.Coop. | 734694.9464469212 | Expansion: 549425.64 \| Growth: 185269.31 |
| 2 | Compañía Alberdi S.Coop. | 470042.2864710122 | Expansion: 362919.09 \| Growth: 107123.19 |
| 3 | Familia Roldan S.Coop. | 348686.6560452091 | Growth: 348686.66 |
| 4 | Uría & Asociados S.A. | 337922.76156874716 | Expansion: 223653.17 \| Growth: 109133.62 \| Early: 5135.97 |
| 5 | Inmobiliaria Blasco S.L.N.E | 327210.5895978795 | Growth: 219604.29 \| Early: 107606.30 |

Consulta 4 ¿Qué startups en etapas avanzadas (Growth o Expansion) con alta viabilidad han recibido inversión de otros actores?

Para identificar startups con potencial de crecimiento, se seleccionaron aquellas que han alcanzado la etapa Growth-Expansion, presentan un flujo de caja registrado y fueron calificadas con “alta viabilidad” en la etapa Seed. Esta consulta permite destacar emprendimientos que avanzan hacia la consolidación y han sido capaces de atraer inversión privada. La información puede ser útil para estudios de madurez del ecosistema emprendedor o para actores interesados en identificar proyectos sólidos con respaldo financiero. En el CTE ‘otros_actores’ nos interesa rastrear el dinero que no proviene de personas naturales. Buscas específicamente el apoyo institucional (Gobiernos y Organizaciones) en las fases más costosas (Growth y Expansion).

```
WITH advanced AS (
  SELECT
    s.id_startup,
    s.nombre AS startup,
    sd.viabilidad,
    CASE
      WHEN ex.id_expansion_stage IS NOT NULL THEN 'Expansion'
      WHEN g.id_growth_stage IS NOT NULL THEN 'Growth'
    END AS etapa_avanzada,
    g.id_growth_stage,
    ex.id_expansion_stage
  FROM startup s
  JOIN seed sd ON sd.id_startup = s.id_startup
  LEFT JOIN early_stage e ON e.id_seed = sd.id_seed
  LEFT JOIN growth_stage g ON g.id_early_stage = e.id_early_stage
  LEFT JOIN expansion_stage ex ON ex.id_growth_stage = g.id_growth_stage
  WHERE sd.viabilidad = 'Alta'
    AND (g.id_growth_stage IS NOT NULL OR ex.id_expansion_stage IS NOT NULL)
),
otros_actores AS (
  SELECT a.id_startup, 'Gobierno' AS actor, 'Growth' AS etapa, SUM(cg.monto) AS monto
  FROM advanced a
  JOIN contratogobierno_growth cg ON cg.id_growth_stage = a.id_growth_stage
  GROUP BY a.id_startup
  UNION ALL
  SELECT a.id_startup, 'Organización', 'Growth', SUM(co.monto)
  FROM advanced a
  JOIN contratoorganizacion_growth co ON co.id_growth_stage = a.id_growth_stage
  GROUP BY a.id_startup
  UNION ALL
  SELECT a.id_startup, 'Gobierno', 'Expansion', SUM(cg.monto)
  FROM advanced a
  JOIN contratogobierno_expansion cg ON cg.id_expansion_stage = a.id_expansion_stage
  GROUP BY a.id_startup
  UNION ALL
  SELECT a.id_startup, 'Organización', 'Expansion', SUM(co.monto)
  FROM advanced a
  JOIN contratoorganizacion_expansion co ON co.id_expansion_stage = a.id_expansion_stage
  GROUP BY a.id_startup
)
```

```sql
SELECT
  a.startup,
  a.etapa_avanzada,
  a.viabilidad,
  STRING_AGG(oa.actor || ' en ' || oa.etapa || ' (monto=' || ROUND(oa.monto::numeric,2) || ')', ' | ') AS inversion_otros_actores
FROM advanced a
JOIN otros_actores oa ON oa.id_startup = a.id_startup
GROUP BY a.startup, a.etapa_avanzada, a.viabilidad
ORDER BY a.etapa_avanzada DESC, a.startup;
```

| | startup character varying (50) | etapa_avanzada text | viabilidad character varying (50) | inversion_otros_actores text |
|---|---|---|---|---|
| 1 | Anaya y Crespi S.A. | Growth | Alta | Organización en Growth (monto=25289.35) |
| 2 | Compañía Figuerola & Asociados S.L. | Growth | Alta | Organización en Growth (monto=62392.23) |
| 3 | Dora Solé Viña S.A. | Growth | Alta | Gobierno en Growth (monto=121201.21) |
| 4 | Fábrica FTK S.C.P | Growth | Alta | Organización en Growth (monto=32284.12) |
| 5 | Familia Bartolomé S.A. | Growth | Alta | Gobierno en Growth (monto=76934.37) |
| 6 | Familia Figueras S.C.P | Growth | Alta | Gobierno en Growth (monto=134214.79) |
| 7 | Familia Roldan S.Coop. | Growth | Alta | Gobierno en Growth (monto=140774.71) \| Organización en Growth (monto=63634.72) |
| 8 | Inmobiliaria Blasco S.L.N.E | Growth | Alta | Gobierno en Growth (monto=108258.50) \| Organización en Growth (monto=20784.08) |
| 9 | Martin & Asociados S.L. | Growth | Alta | Organización en Growth (monto=55662.76) |

Consulta 5: Top 3 startups por incubadora según monto total invertido (todas las etapas)

La consulta es clara, estamos en un ranking de rendimiento por incubadora. Esa consulta nos permite comparar top 3 de distintas incubadoras cada una con sus respectivas startups revelando que instituciones están atrayendo los proyectos con mejor tracción financiera

```sql
WITH inv AS (
  SELECT sd.id_startup, ci.monto
  FROM contratoinversionista_early ci
  JOIN early_stage e ON e.id_early_stage = ci.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, cg.monto
  FROM contratogobierno_early cg
  JOIN early_stage e ON e.id_early_stage = cg.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, co.monto
  FROM contratoorganizacion_early co
  JOIN early_stage e ON e.id_early_stage = co.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed

  UNION ALL
  SELECT sd.id_startup, ci.monto
  FROM contratoinversionista_growth ci
  JOIN growth_stage g ON g.id_growth_stage = ci.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, cg.monto
  FROM contratogobierno_growth cg
  JOIN growth_stage g ON g.id_growth_stage = cg.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, co.monto
  FROM contratoorganizacion_growth co
  JOIN growth_stage g ON g.id_growth_stage = co.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
```

```sql
  UNION ALL
  SELECT sd.id_startup, ci.monto
  FROM contratoinversionista_expansion ci
  JOIN expansion_stage ex ON ex.id_expansion_stage = ci.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, cg.monto
  FROM contratogobierno_expansion cg
  JOIN expansion_stage ex ON ex.id_expansion_stage = cg.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
  UNION ALL
  SELECT sd.id_startup, co.monto
  FROM contratoorganizacion_expansion co
  JOIN expansion_stage ex ON ex.id_expansion_stage = co.id_expansion_stage
  JOIN growth_stage g ON g.id_growth_stage = ex.id_growth_stage
  JOIN early_stage e ON e.id_early_stage = g.id_early_stage
  JOIN seed sd ON sd.id_seed = e.id_seed
),
tot AS (
  SELECT id_startup, SUM(monto) AS monto_total
  FROM inv
  GROUP BY id_startup
),
ranked AS (
  SELECT
    i.nombre AS incubadora,
    s.nombre AS startup,
    t.monto_total,
    ROW_NUMBER() OVER (PARTITION BY s.id_incubadora ORDER BY t.monto_total DESC) AS rn
  FROM tot t
  JOIN startup s ON s.id_startup = t.id_startup
  JOIN incubadora i ON i.id_incubadora = s.id_incubadora
)
SELECT incubadora, startup, round(monto_total,2)
FROM ranked
WHERE rn <= 3
ORDER BY incubadora, monto_total DESC;
```

Mostrando los seis primeros resultados:

| | incubadora character varying (50) | startup character varying (50) | round numeric |
|---|---|---|---|
| 1 | AndesHub | Comercial Berenguer S.Com. | 302955.90 |
| 2 | AndesHub | Familia Figueras S.C.P | 282123.92 |
| 3 | AndesHub | Esmeralda Solana García S.A. | 216870.75 |
| 4 | ImpactLab | Uría & Asociados S.A. | 337922.76 |
| 5 | ImpactLab | Ruiz y asociados S.A.D | 225591.23 |
| 6 | ImpactLab | Farmaceútica Figueroa & Asociados S.L.N.E | 217537.84 |

Consulta6: incubadoras con su respectivo nro de startups, promedio y mediana entre el número de días que hay entre las fases seed y early

Medimos el tiempo de transición entre el primer 'salto' crítico del ciclo de vida. Mientras menos sea el número de días que la incubadora ayuda a su startups a pasar de dicha fase, más eficiente es. Asimismo, nos ayuda a medir el tiempo; o sea, a analizar porque algunas incubadoras son más lentas que otras.

```
WITH base AS (
  SELECT
    s.id_incubadora,
    (e.fecha - sd.fecha) AS dias_seed_a_early
  FROM seed sd
  JOIN early_stage e ON e.id_seed = sd.id_seed
  JOIN startup s ON s.id_startup = sd.id_startup
)
SELECT
  i.nombre AS incubadora,
  COUNT(*) AS n_startups,
  ROUND(AVG(dias_seed_a_early)::numeric, 2) AS avg_dias,
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY dias_seed_a_early) AS mediana_dias
FROM base b
JOIN incubadora i ON i.id_incubadora = b.id_incubadora
GROUP BY i.nombre
ORDER BY avg_dias DESC;
```

| | incubadora character varying (50) | n_startups bigint | avg_dias numeric | mediana_dias double precision |
|---|---|---|---|---|
| 1 | Startup Perú | 29 | 262.17 | 273 |
| 2 | Wayra | 27 | 261.96 | 289 |
| 3 | Pacific Accelerator | 40 | 244.63 | 238 |
| 4 | SeedLab | 32 | 242.16 | 250.5 |
| 5 | AndesHub | 40 | 241.03 | 232 |
| 6 | InnovateX | 45 | 238.04 | 248 |
| 7 | ImpactLab | 41 | 235.73 | 248 |
| 8 | UTEC Ventures | 43 | 205.40 | 195 |

Consulta 7: se solicita un análisis de tendencia temporal por mes con su respectivo crecimiento histórico

Al ver los datos mes a mes, podríamos descubrir patrones o meses en los que se invierta más. Asimismo la columna 'inversion_acumulada' es buena para crear un gráfico de área o de líneas que muestre como ha crecido del capital total inyectado en el país

```
WITH inv_dates AS (
  SELECT fecha, monto FROM contratoinversionista_early
  UNION ALL SELECT fecha, monto FROM contratoinversionista_growth
  UNION ALL SELECT fecha, monto FROM contratoinversionista_expansion
  UNION ALL SELECT fecha, monto FROM contratogobierno_early
  UNION ALL SELECT fecha, monto FROM contratogobierno_growth
  UNION ALL SELECT fecha, monto FROM contratogobierno_expansion
  UNION ALL SELECT fecha, monto FROM contratoorganizacion_early
  UNION ALL SELECT fecha, monto FROM contratoorganizacion_growth
  UNION ALL SELECT fecha, monto FROM contratoorganizacion_expansion
),
by_month AS (
  SELECT
    DATE_TRUNC('month', fecha)::date AS mes,
    round(SUM(monto),2) AS inversion_mensual
  FROM inv_dates
  GROUP BY 1
)
SELECT
  mes,
  inversion_mensual,
  SUM(inversion_mensual) OVER (ORDER BY mes) AS inversion_acumulada
FROM by_month
ORDER BY mes;
```

Mostrando los 5 primeros:

| | mes<br>date | inversion_mensual<br>numeric | inversion_acumulada<br>numeric |
|---|---|---|---|
| 1 | 2020-07-01 | 72293.71 | 72293.71 |
| 2 | 2020-08-01 | 173085.72 | 245379.43 |
| 3 | 2020-09-01 | 205231.92 | 450611.35 |
| 4 | 2020-10-01 | 10567.30 | 461178.65 |
| 5 | 2020-11-01 | 63887.99 | 525066.64 |

Consulta 8 : ranking de estancamiento en etapas

Esta es la consulta de Análisis de Supervivencia y Riesgo de Estancamiento

CTE **current_stage**: Utiliza una estructura de CASE WHEN para determinar la jerarquía. Detecta cuál es la etapa más avanzada alcanzada por la startup y extrae la fecha exacta en la que entró en ella.

CTE **stagnation**: Calcula la métrica operativa clave: CURRENT_DATE - fecha_inicio_etapa. Esto nos da el número de días que la empresa lleva "congelada" en su estado actual.

Podemos identificar startups que tienen un dias_en_etapa excesivamente alto. En el ecosistema peruano, esto puede indicar falta de tracción, problemas de gestión o un mercado que ya se saturó.

```
WITH current_stage AS (
  SELECT
    s.id_startup,
    s.nombre AS startup,
    sd.fecha AS seed_fecha,
    e.fecha  AS early_fecha,
    g.fecha  AS growth_fecha,
    ex.fecha AS expansion_fecha,
    -- Detectar etapa actual: la más avanzada existente
    CASE
      WHEN ex.id_expansion_stage IS NOT NULL THEN 'Expansion'
      WHEN g.id_growth_stage IS NOT NULL THEN 'Growth'
      WHEN e.id_early_stage IS NOT NULL THEN 'Early'
      WHEN sd.id_seed IS NOT NULL THEN 'Seed'
      ELSE 'Pre_Seed'
    END AS etapa_actual,
    -- Fecha de inicio de la etapa actual
    CASE
      WHEN ex.id_expansion_stage IS NOT NULL THEN ex.fecha
      WHEN g.id_growth_stage IS NOT NULL THEN g.fecha
      WHEN e.id_early_stage IS NOT NULL THEN e.fecha
      WHEN sd.id_seed IS NOT NULL THEN sd.fecha
      ELSE NULL
    END AS fecha_inicio_etapa
  FROM startup s
  LEFT JOIN seed sd ON sd.id_startup = s.id_startup
  LEFT JOIN early_stage e ON e.id_seed = sd.id_seed
  LEFT JOIN growth_stage g ON g.id_early_stage = e.id_early_stage
  LEFT JOIN expansion_stage ex ON ex.id_growth_stage = g.id_growth_stage
),
stagnation AS (
  SELECT
    startup,
    etapa_actual,
    fecha_inicio_etapa,
    (CURRENT_DATE - fecha_inicio_etapa) AS dias_en_etapa
  FROM current_stage
  WHERE etapa_actual <> 'Pre_Seed'
    AND fecha_inicio_etapa IS NOT NULL
),
```

```sql
ranked AS (
  SELECT
    startup,
    etapa_actual,
    fecha_inicio_etapa,
    dias_en_etapa,
    ROW_NUMBER() OVER (PARTITION BY etapa_actual ORDER BY dias_en_etapa DESC) AS rank_en_etapa,
    PERCENT_RANK() OVER (PARTITION BY etapa_actual ORDER BY dias_en_etapa) AS percentil_en_etapa
  FROM stagnation
)
SELECT
  startup,
  etapa_actual,
  fecha_inicio_etapa,
  dias_en_etapa,
  rank_en_etapa,
  ROUND((100 * percentil_en_etapa)::numeric, 2) AS percentil_en_etapa
FROM ranked
ORDER BY dias_en_etapa DESC
LIMIT 50;
```

Mostrando los 5 primeros resultados

| | startup<br>character varying (50) | etapa_actual<br>text | fecha_inicio_etapa<br>date | dias_en_etapa<br>integer | rank_en_etapa<br>bigint | percentil_en_etapa<br>numeric |
|---|---|---|---|---|---|---|
| 1 | Gascón y Romeu S.A. | Seed | 2020-03-11 | 2111 | 1 | 100.00 |
| 2 | Ulises Campo Ruano S.Coop. | Seed | 2020-03-16 | 2106 | 2 | 99.75 |
| 3 | Manufacturas Baena S.L.N.E | Seed | 2020-04-06 | 2085 | 3 | 99.51 |
| 4 | Transportes XHR S.L. | Seed | 2020-04-14 | 2077 | 4 | 99.26 |
| 5 | Fito Yuste Amaya S.Coop. | Seed | 2020-04-17 | 2074 | 5 | 99.01 |

# 9. Índices

En esta sección voy a detallar los planes del cómo se tratarán la parte de ‘índices’ del presente proyecto:

Se crearon bases de datos: startup_1k, startup_10k y startup_100k. Para cada una de ellas se tiene una versión ‘sin índices’ y otra versión ‘con índices’. Se hace esto con fines comparativos para poder ilustrar dicha diferencia para bases de datos con diferentes cantidades de registros.

Se usó:

### 9.1 Base de datos de 1k sin índices: consulta ‘2’:

| | QUERY PLAN text |
|---|---|
| 1 | Sort (cost=57.97..57.98 rows=1 width=126) (actual time=10.469..10.496 rows=237 loops=1) |
| 2 | Output: s.nombre, e.fecha, (COALESCE((string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), 'N/A'::text)), (COALESCE((stri |
| 3 | Sort Key: e.fecha, s.nombre |
| 4 | Sort Method: quicksort Memory: 45kB |
| 5 | Buffers: shared hit=1434 |
| 6 | -> Merge Left Join (cost=42.06..57.96 rows=1 width=126) (actual time=1.524..10.264 rows=237 loops=1) |
| 7 | Output: s.nombre, e.fecha, COALESCE((string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), 'N/A'::text), COALESCE((str |
| 8 | Inner Unique: true |
| 9 | Merge Cond: (e.id_early_stage = co.id_early_stage) |
| 10 | Buffers: shared hit=1434 |
| 11 | -> Merge Left Join (cost=36.07..48.22 rows=1 width=98) (actual time=1.306..9.634 rows=237 loops=1) |
| 12 | Output: e.fecha, e.id_early_stage, s.nombre, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), (string_agg(DISTI |
| 13 | Inner Unique: true |
| 14 | Merge Cond: (e.id_early_stage = cg.id_early_stage) |
| 15 | Buffers: shared hit=1432 |
| 16 | -> Nested Loop (cost=28.16..34.99 rows=1 width=66) (actual time=1.004..8.880 rows=237 loops=1) |
| 17 | Output: e.fecha, e.id_early_stage, s.nombre, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)) |
| 18 | Inner Unique: true |
| 19 | Buffers: shared hit=1430 |
| 20 | -> Nested Loop (cost=27.88..34.61 rows=1 width=44) (actual time=0.989..2.423 rows=237 loops=1) |
| 21 | Output: e.fecha, e.id_early_stage, sd.id_startup, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)) |
| 22 | Inner Unique: true |
| 23 | Buffers: shared hit=719 |

| |
|---|
| Planning: |
| Buffers: shared hit=106 |
| Planning Time: 4.806 ms |
| Execution Time: 10.807 ms |

## 9.2 Base de datos de 1k con índices: consulta ‘2’

| | QUERY PLAN text |
|---|---|
| 1 | Sort (cost=57.97..57.98 rows=1 width=126) (actual time=4.585..4.604 rows=237 loops=1) |
| 2 | Output: s.nombre, e.fecha, (COALESCE((string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), 'N/A'::text)), (COALESCE((string |
| 3 | Sort Key: e.fecha, s.nombre |
| 4 | Sort Method: quicksort Memory: 45kB |
| 5 | Buffers: shared hit=1434 |
| 6 | -> Merge Left Join (cost=42.06..57.96 rows=1 width=126) (actual time=0.801..4.399 rows=237 loops=1) |
| 7 | Output: s.nombre, e.fecha, COALESCE((string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), 'N/A'::text), COALESCE((stri |
| 8 | Inner Unique: true |
| 9 | Merge Cond: (e.id_early_stage = co.id_early_stage) |
| 10 | Buffers: shared hit=1434 |
| 11 | -> Merge Left Join (cost=36.07..48.22 rows=1 width=98) (actual time=0.683..3.878 rows=237 loops=1) |
| 12 | Output: e.fecha, e.id_early_stage, s.nombre, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)), (string_agg(DISTIN |
| 13 | Inner Unique: true |
| 14 | Merge Cond: (e.id_early_stage = cg.id_early_stage) |
| 15 | Buffers: shared hit=1432 |
| 16 | -> Nested Loop (cost=28.16..34.99 rows=1 width=66) (actual time=0.424..3.065 rows=237 loops=1) |
| 17 | Output: e.fecha, e.id_early_stage, s.nombre, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)) |
| 18 | Inner Unique: true |
| 19 | Buffers: shared hit=1430 |
| 20 | -> Nested Loop (cost=27.88..34.61 rows=1 width=44) (actual time=0.416..2.289 rows=237 loops=1) |
| 21 | Output: e.fecha, e.id_early_stage, sd.id_startup, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)) |
| 22 | Inner Unique: true |
| 23 | Buffers: shared hit=719 |
| 24 | -> Merge Left Join (cost=27.61..34.20 rows=1 width=44) (actual time=0.404..1.184 rows=237 loops=1) |
| 25 | Output: e.fecha, e.id_seed, e.id_early_stage, (string_agg(DISTINCT (((((p.nombre_completo)::text \|\| ' (DNI '::text) \|\| (p.dni)::text) \|\| ')'::text)), ', '::text)) |
| 26 | Inner Unique: true |
| 27 | Merge Cond: (e.id_early_stage = ci.id_early_stage) |

| |
|---|
| Planning: |
| Buffers: shared hit=105 |
| Planning Time: 2.318 ms |
| Execution Time: 4.801 ms |

### 9.3 Base de datos de 10k sin índices: consulta ‘7’

| | QUERY PLAN text |
|---|---|
| 1 | WindowAgg (cost=154.23..157.71 rows=200 width=68) (actual time=6.785..6.837 rows=58 loops=1) |
| 2 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), (round(sum(contratoinversionista_early.monto), 2)), sum |
| 3 | Buffers: shared hit=3 read=24 |
| 4 | -> Sort (cost=154.21..154.71 rows=200 width=36) (actual time=6.737..6.743 rows=58 loops=1) |
| 5 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), (round(sum(contratoinversionista_early.monto), 2)) |
| 6 | Sort Key: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date) |
| 7 | Sort Method: quicksort Memory: 26kB |
| 8 | Buffers: shared hit=3 read=24 |
| 9 | -> HashAggregate (cost=142.06..146.56 rows=200 width=36) (actual time=6.630..6.658 rows=58 loops=1) |
| 10 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), round(sum(contratoinversionista_early.monto), 2) |
| 11 | Group Key: (date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date |
| 12 | Batches: 1 Memory Usage: 64kB |
| 13 | Buffers: shared read=24 |
| 14 | -> Result (cost=0.00..126.31 rows=3150 width=16) (actual time=1.276..4.457 rows=3144 loops=1) |
| 15 | Output: (date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date, contratoinversionista_early.monto |
| 16 | Buffers: shared read=24 |
| 17 | -> Append (cost=0.00..71.19 rows=3150 width=16) (actual time=0.052..1.433 rows=3144 loops=1) |
| 18 | Buffers: shared read=24 |
| 19 | -> Seq Scan on public.contratoinversionista_early (cost=0.00..23.61 rows=1361 width=16) (actual time=0.051..0.626 rows=1361 loops=1) |
| 20 | Output: contratoinversionista_early.fecha, contratoinversionista_early.monto |
| 21 | Buffers: shared read=10 |
| 22 | -> Seq Scan on public.contratoinversionista_growth (cost=0.00..0.00 rows=1 width=36) (actual time=0.027..0.027 rows=0 loops=1) |
| 23 | Output: contratoinversionista_growth.fecha, contratoinversionista_growth.monto |
| 24 | -> Seq Scan on public.contratoinversionista_expansion (cost=0.00..0.00 rows=1 width=36) (actual time=0.006..0.007 rows=0 loops=1) |
| 25 | Output: contratoinversionista_expansion.fecha, contratoinversionista_expansion.monto |
| 26 | -> Seq Scan on public.contratogobierno_early (cost=0.00..17.84 rows=984 width=16) (actual time=0.050..0.236 rows=984 loops=1) |
| 27 | Output: contratogobierno_early.fecha, contratogobierno_early.monto |
| 28 | Buffers: shared read=8 |

Planning:

Buffers: shared hit=305 read=23

Planning Time: 162.529 ms

Execution Time: 7.488 ms

### 9.4 Base de datos de 10k con índices: consulta ‘7’

| | QUERY PLAN<br>text |
|---|---|
| 1 | WindowAgg (cost=154.23..157.71 rows=200 width=68) (actual time=12.510..12.629 rows=58 loops=1) |
| 2 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), (round(sum(contratoinversionista_early.monto), 2)), sum((round( |
| 3 | Buffers: shared hit=24 |
| 4 | -> Sort (cost=154.21..154.71 rows=200 width=36) (actual time=12.476..12.486 rows=58 loops=1) |
| 5 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), (round(sum(contratoinversionista_early.monto), 2)) |
| 6 | Sort Key: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date) |
| 7 | Sort Method: quicksort Memory: 26kB |
| 8 | Buffers: shared hit=24 |
| 9 | -> HashAggregate (cost=142.06..146.56 rows=200 width=36) (actual time=12.379..12.436 rows=58 loops=1) |
| 10 | Output: ((date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date), round(sum(contratoinversionista_early.monto), 2) |
| 11 | Group Key: (date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date |
| 12 | Batches: 1 Memory Usage: 64kB |
| 13 | Buffers: shared hit=24 |
| 14 | -> Result (cost=0.00..126.31 rows=3150 width=16) (actual time=0.069..8.508 rows=3144 loops=1) |
| 15 | Output: (date_trunc('month'::text, (contratoinversionista_early.fecha)::timestamp with time zone))::date, contratoinversionista_early.monto |
| 16 | Buffers: shared hit=24 |
| 17 | -> Append (cost=0.00..71.19 rows=3150 width=16) (actual time=0.058..3.062 rows=3144 loops=1) |
| 18 | Buffers: shared hit=24 |
| 19 | -> Seq Scan on public.contratoinversionista_early (cost=0.00..23.61 rows=1361 width=16) (actual time=0.056..1.259 rows=1361 loops=1) |
| 20 | Output: contratoinversionista_early.fecha, contratoinversionista_early.monto |
| 21 | Buffers: shared hit=10 |
| 22 | -> Seq Scan on public.contratoinversionista_growth (cost=0.00..0.00 rows=1 width=36) (actual time=0.187..0.188 rows=0 loops=1) |
| 23 | Output: contratoinversionista_growth.fecha, contratoinversionista_growth.monto |
| 24 | -> Seq Scan on public.contratoinversionista_expansion (cost=0.00..0.00 rows=1 width=36) (actual time=0.026..0.026 rows=0 loops=1) |
| 25 | Output: contratoinversionista_expansion.fecha, contratoinversionista_expansion.monto |
| 26 | -> Seq Scan on public.contratogobierno_early (cost=0.00..17.84 rows=984 width=16) (actual time=0.059..0.412 rows=984 loops=1) |
| 27 | Output: contratogobierno_early.fecha, contratogobierno_early.monto |
| 28 | Buffers: shared hit=8 |
| 29 | -> Seq Scan on public.contratogobierno_growth (cost=0.00..0.00 rows=1 width=36) (actual time=0.030..0.030 rows=0 loops=1) |
| 30 | Output: contratogobierno_growth.fecha, contratogobierno_growth.monto |

| |
|---|
| Planning: |
| Buffers: shared hit=6 |
| Planning Time: 2.002 ms |
| Execution Time: 12.897 ms |

### 9.5 Base de datos de 100k sin índices: consulta ‘8’

| | QUERY PLAN text |
|---|---|
| 1 | Limit (cost=32407.30..32407.42 rows=50 width=93) (actual time=1430.073..1430.084 rows=50 loops=1) |
| 2 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, (round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2)) |
| 3 | Buffers: shared hit=6 read=1357, temp read=1926 written=847 |
| 4 | -> Sort (cost=32407.30..32654.80 rows=99002 width=93) (actual time=1430.071..1430.079 rows=50 loops=1) |
| 5 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, (round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2)) |
| 6 | Sort Key: ranked.dias_en_etapa DESC |
| 7 | Sort Method: top-N heapsort Memory: 32kB |
| 8 | Buffers: shared hit=6 read=1357, temp read=1926 written=847 |
| 9 | -> Subquery Scan on ranked (cost=19218.36..29118.52 rows=99002 width=93) (actual time=1032.485..1393.627 rows=69852 loops=1) |
| 10 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2) |
| 11 | Buffers: shared hit=3 read=1357, temp read=1926 written=847 |
| 12 | -> WindowAgg (cost=19218.36..28376.01 rows=99002 width=69) (actual time=1031.566..1181.887 rows=69852 loops=1) |

```
Planning:
  Buffers: shared hit=236 read=34
Planning Time: 175.931 ms
Execution Time: 1435.435 ms
```

### 9.6 Base de datos de 100k con índices: consulta ‘8’

| | QUERY PLAN text |
|---|---|
| 1 | Limit (cost=32407.30..32407.42 rows=50 width=93) (actual time=982.587..982.608 rows=50 loops=1) |
| 2 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, (round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2)) |
| 3 | Buffers: shared hit=1357, temp read=1926 written=847 |
| 4 | -> Sort (cost=32407.30..32654.80 rows=99002 width=93) (actual time=982.585..982.600 rows=50 loops=1) |
| 5 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, (round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2)) |
| 6 | Sort Key: ranked.dias_en_etapa DESC |
| 7 | Sort Method: top-N heapsort Memory: 32kB |
| 8 | Buffers: shared hit=1357, temp read=1926 written=847 |
| 9 | -> Subquery Scan on ranked (cost=19218.36..29118.52 rows=99002 width=93) (actual time=446.015..919.140 rows=69852 loops=1) |
| 10 | Output: ranked.startup, ranked.etapa_actual, ranked.fecha_inicio_etapa, ranked.dias_en_etapa, ranked.rank_en_etapa, round((('100'::double precision * ranked.percentil_en_etapa))::numeric, 2) |
| 11 | Buffers: shared hit=1357, temp read=1926 written=847 |

```
Planning:
  Buffers: shared hit=34
Planning Time: 1.593 ms
Execution Time: 988.129 ms
```